Automotive ProASIC3 FPGA Fabric User’s Guide

Table of Contents

Introduction

Contents

This user’s guide contains information to help designers understand and use Microsemi's Automotive ProASIC®3 devices. Each chapter addresses a specific topic. Most of these chapters apply to other Microsemi device families as well. When a feature or description applies only to a specific device family, this is made clear in the text.

Revision History

The revision history for each chapter is listed at the end of the chapter. Most of these chapters were formerly included in device handbooks. Some were originally application notes or information included in device datasheets.

A "Summary of Changes" table at the end of this user’s guide lists the chapters that were changed in each revision of the document, with links to the "List of Changes" sections for those chapters.

Related Information

Refer to the *Automotive ProASIC3 Flash Family FPGAs* datasheet for detailed specifications, timing, and package and pin information.

Automotive ProASIC3 device solutions can be found on the website:

/http://www.microsemi.com/soc/products/solutions/auto/guide.aspx#pa3.

1 – FPGA Array Architecture in Low Power Flash Devices

Device Architecture

Advanced Flash Switch

Unlike SRAM FPGAs, the low power flash devices use a live-at-power-up ISP flash switch as their programming element. Flash cells are distributed throughout the device to provide nonvolatile, reconfigurable programming to connect signal lines to the appropriate VersaTile inputs and outputs. In the flash switch, two transistors share the floating gate, which stores the programming information (Figure 1-1). One is the sensing transistor, which is only used for writing and verification of the floating gate voltage. The other is the switching transistor. The latter is used to connect or separate routing nets, or to configure VersaTile logic. It is also used to erase the floating gate. Dedicated high-performance lines are connected as required using the flash switch for fast, low-skew, global signal distribution throughout the device core. Maximum core utilization is possible for virtually any design. The use of the flash switch technology also removes the possibility of firm errors, which are increasingly common in SRAM-based FPGAs.

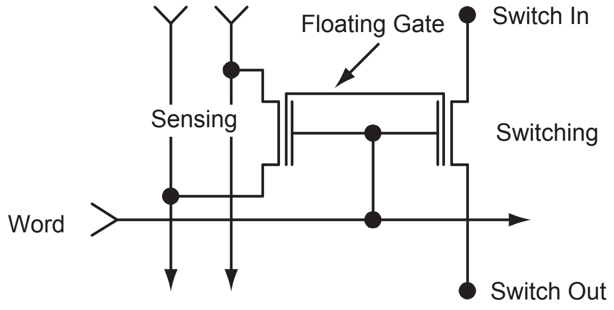


***Figure 1-1 •* Flash-Based Switch**

FPGA Array Architecture Support

The flash FPGAs listed in Table 1-1 support the architecture features described in this document.

Table 1-1 **• Flash-Based FPGAs**

Series	Family*	Description
IGLOO®	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry’s lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC®3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

*Note: *The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 1-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

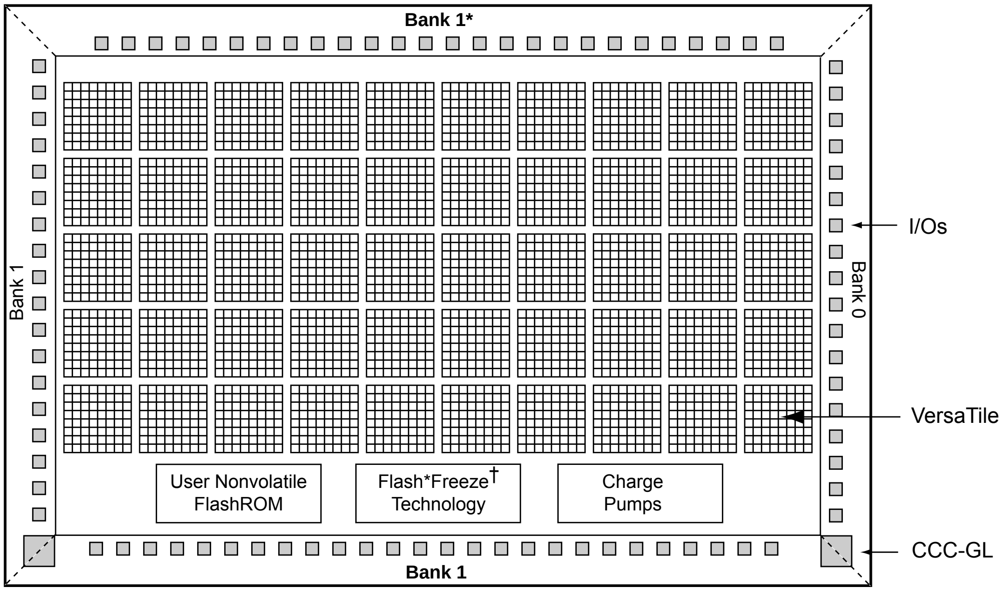
In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 1-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry’s Lowest Power FPGAs Portfolio*.

Device Overview

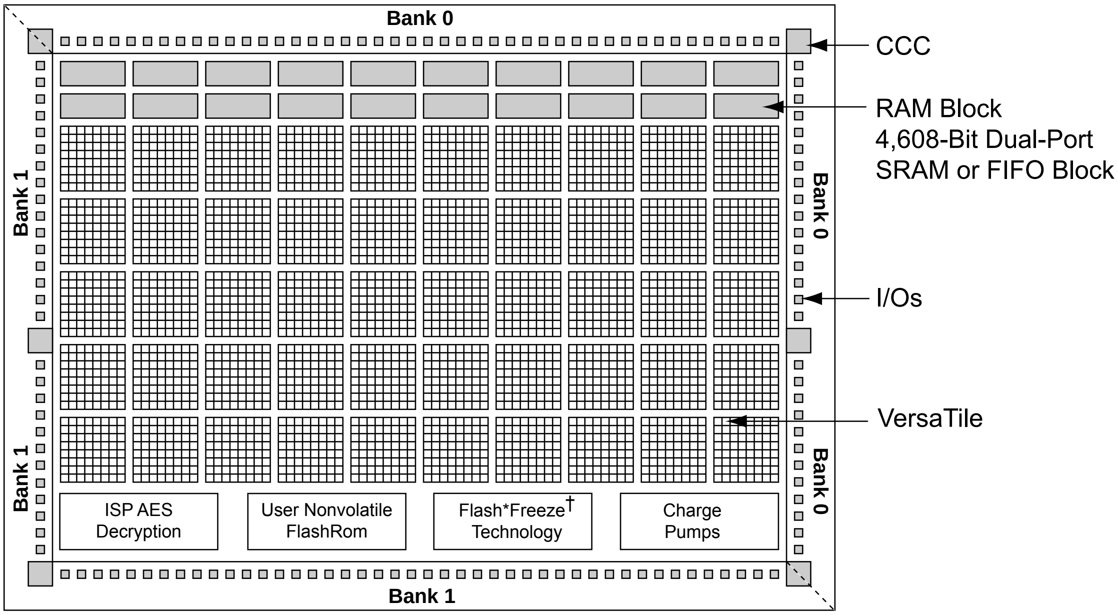
Low power flash devices consist of multiple distinct programmable architectural features (Figure 1-5 on page 15 through Figure 1-7 on page 16):

- FPGA fabric/core (VersaTiles)
- Routing and clock resources (VersaNets)
- FlashROM
- Dedicated SRAM and/or FIFO
  - 30 k gate and smaller device densities do not support SRAM or FIFO.
  - Automotive devices do not support FIFO operation.
- I/O structures
- Flash*Freeze technology and low power modes



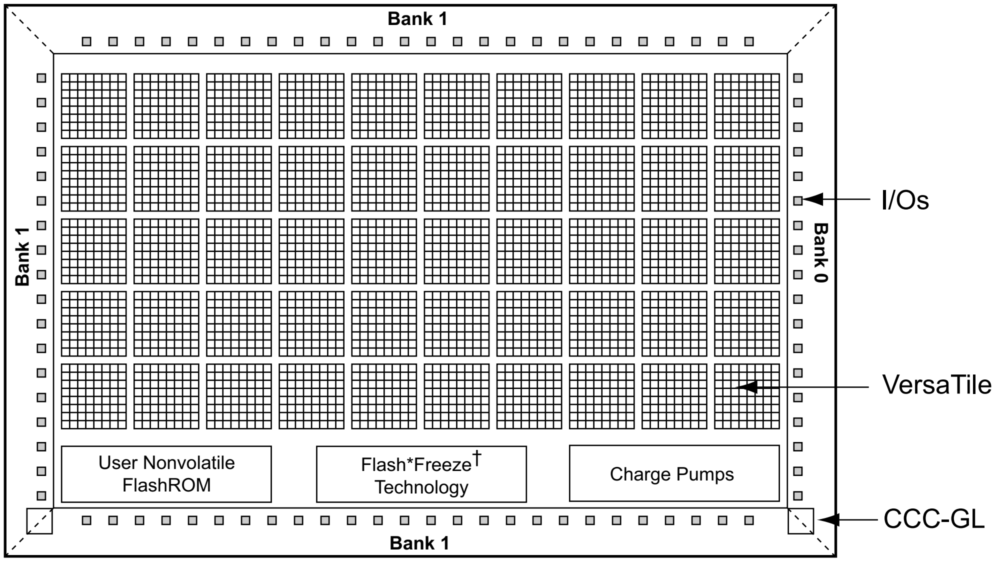
Notes: * Bank 0 for the 30 k devices
† Flash*Freeze mode is supported on IGLOO devices.

Figure 1-2 • **IGLOO and ProASIC3 nano Device Architecture Overview with Two I/O Banks (applies to 10 k and 30 k device densities, excluding IGLOO PLUS devices)**



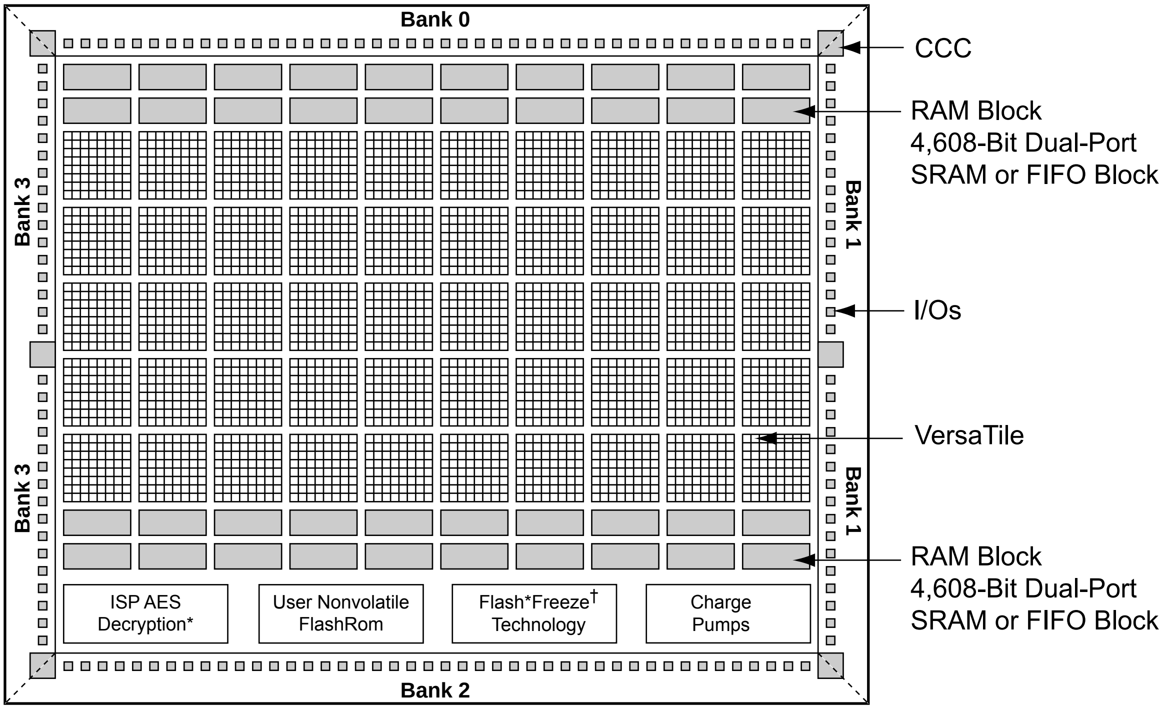
*Note: † Flash*Freeze mode is supported on IGLOO devices.*

Figure 1-3 • **IGLOO Device Architecture Overview with Two I/O Banks with RAM and PLL (60 k and 125 k gate densities)**



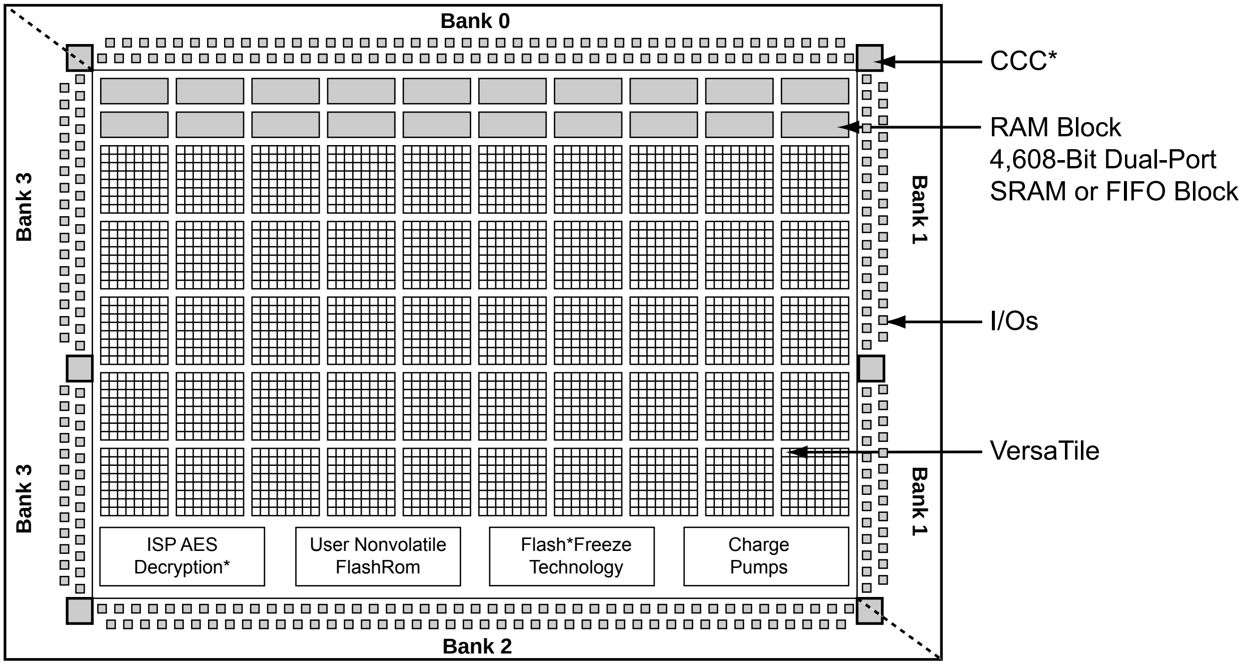
*Note: † Flash*Freeze mode is supported on IGLOO devices.*

Figure 1-4 • **IGLOO Device Architecture Overview with Three I/O Banks (AGLN015, AGLN020, A3PN015, and A3PN020)**



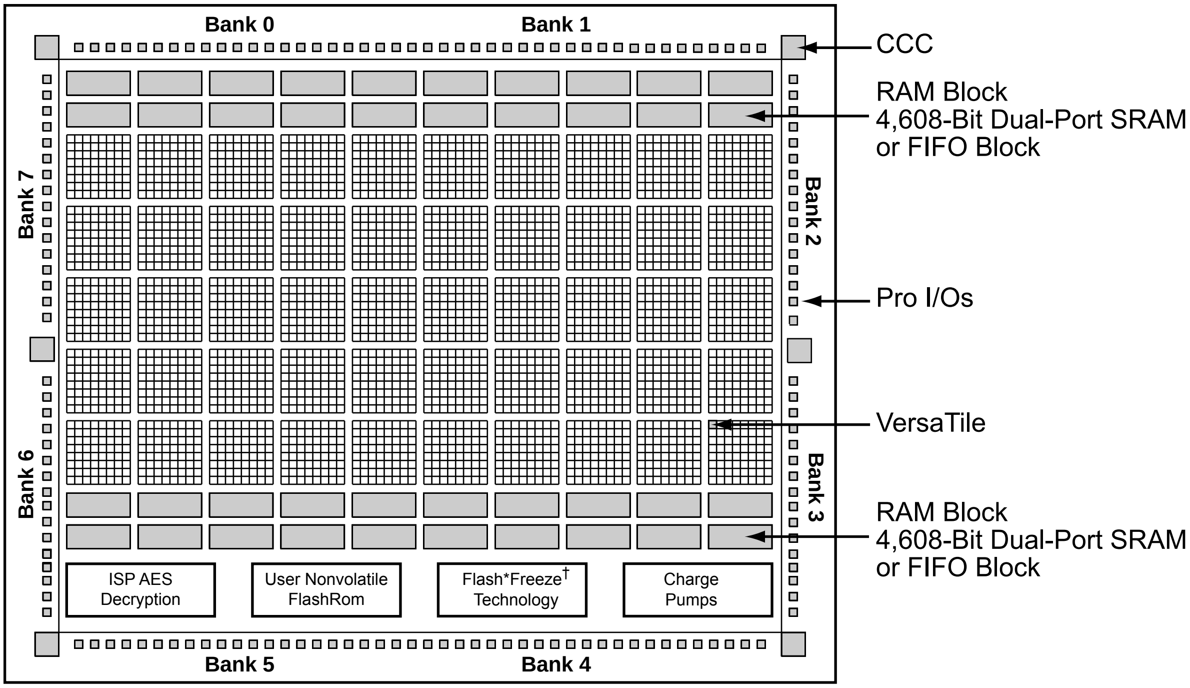
Note: Flash*Freeze technology only applies to IGLOO and ProASIC3L families.

Figure 1-5 • **IGLOO, IGLOO nano, ProASIC3 nano, and ProASIC3/L Device Architecture Overview with Four I/O Banks (AGL600 device is shown)**



Note: * AGLP030 does not contain a PLL or support AES security.

Figure 1-6 • **IGLOO PLUS Device Architecture Overview with Four I/O Banks**



*Note: Flash*Freeze technology only applies to IGLOOe devices.*

Figure 1-7 • **IGLOOe and ProASIC3E Device Architecture Overview (AGLE600 device is shown)**

I/O State of Newly Shipped Devices

Devices are shipped from the factory with a test design in the device. The power-on switch for VCC is OFF by default in this test design, so I/Os are tristated by default. Tristated means the I/O is not actively driven and floats. The exact value cannot be guaranteed when it is floating. Even in simulation software, a tristate value is marked as unknown. Due to process variations and shifts, tristated I/Os may float toward High or Low, depending on the particular device and leakage level.

If there is concern regarding the exact state of unused I/Os, weak pull-up/pull-down should be added to the floating I/Os so their state is controlled and stabilized.

Core Architecture

VersaTile

The proprietary IGLOO and ProASIC3 device architectures provide granularity comparable to gate arrays. The device core consists of a sea-of-VersaTiles architecture.

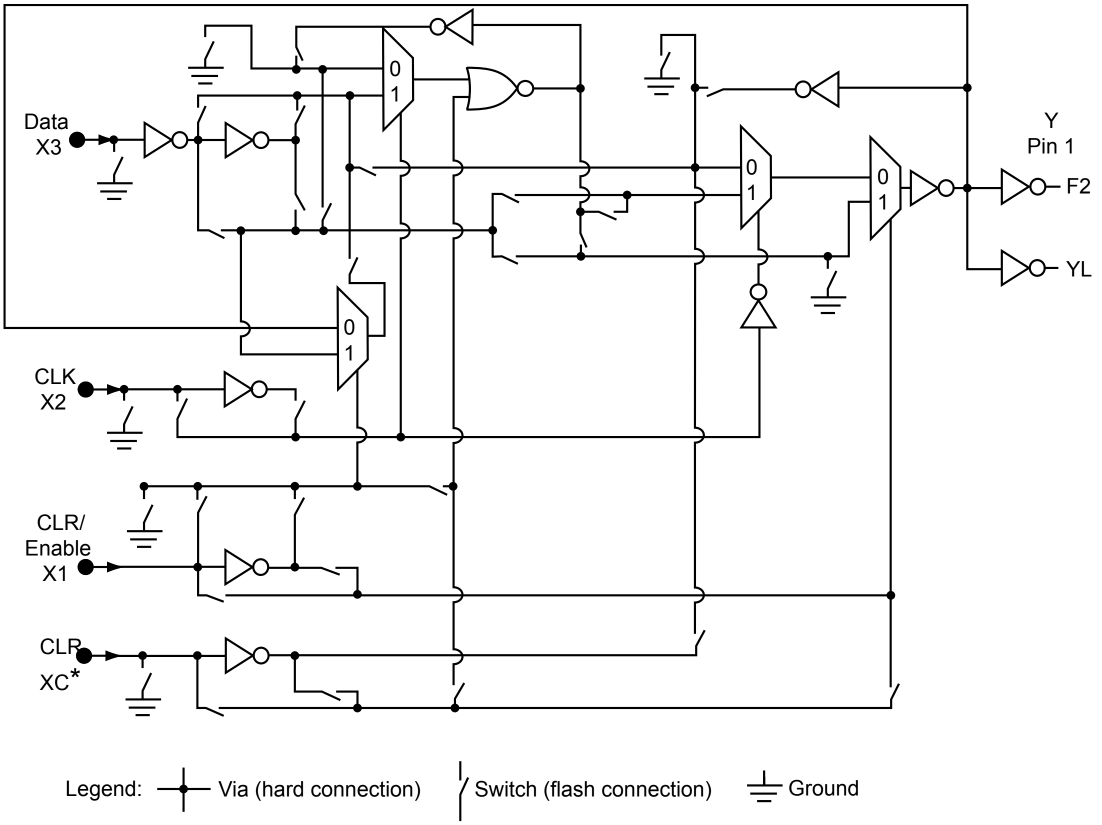
As illustrated in Figure 1-8, there are four inputs in a logic VersaTile cell, and each VersaTile can be configured using the appropriate flash switch connections:

- Any 3-input logic function
- Latch with clear or set
- D-flip-flop with clear or set
- Enable D-flip-flop with clear or set (on a 4th input)

VersaTiles can flexibly map the logic and sequential gates of a design. The inputs of the VersaTile can be inverted (allowing bubble pushing), and the output of the tile can connect to high-speed, very-long-line routing resources. VersaTiles and larger functions can be connected with any of the four levels of routing hierarchy.

When the VersaTile is used as an enable D-flip-flop, SET/CLR is supported by a fourth input. The SET/CLR signal can only be routed to this fourth input over the VersaNet (global) network. However, if, in the user's design, the SET/CLR signal is not routed over the VersaNet network, a compile warning message will be given, and the intended logic function will be implemented by two VersaTiles instead of one.

The output of the VersaTile is F2 when the connection is to the ultra-fast local lines, or YL when the connection is to the efficient long-line or very-long-line resources.



* This input can only be connected to the global clock distribution network.

Figure 1-8 **• Low Power Flash Device Core VersaTile**

Array Coordinates

During many place-and-route operations in the Microsemi Designer software tool, it is possible to set constraints that require array coordinates. Table 1-2 provides array coordinates of core cells and memory blocks for IGLOO and ProASIC3 devices. Table 1-3 provides the information for IGLOO PLUS devices. Table 1-4 on page 19 provides the information for IGLOO nano and ProASIC3 nano devices. The array coordinates are measured from the lower left (0, 0). They can be used in region constraints for specific logic groups/blocks, designated by a wildcard, and can contain core cells, memories, and I/Os.

I/O and cell coordinates are used for placement constraints. Two coordinate systems are needed because there is not a one-to-one correspondence between I/O cells and core cells. In addition, the I/O coordinate system changes depending on the die/package combination. It is not listed in Table 1-2. The Designer ChipPlanner tool provides the array coordinates of all I/O locations. I/O and cell coordinates are used for placement constraints. However, I/O placement is easier by package pin assignment.

Figure 1-9 on page 19 illustrates the array coordinates of a 600 k gate device. For more information on how to use array coordinates for region/placement constraints, see the *Designer User's Guide* or online help (available in the software) for software tools.

Table 1-2 **• IGLOO and ProASIC3 Array Coordinates**

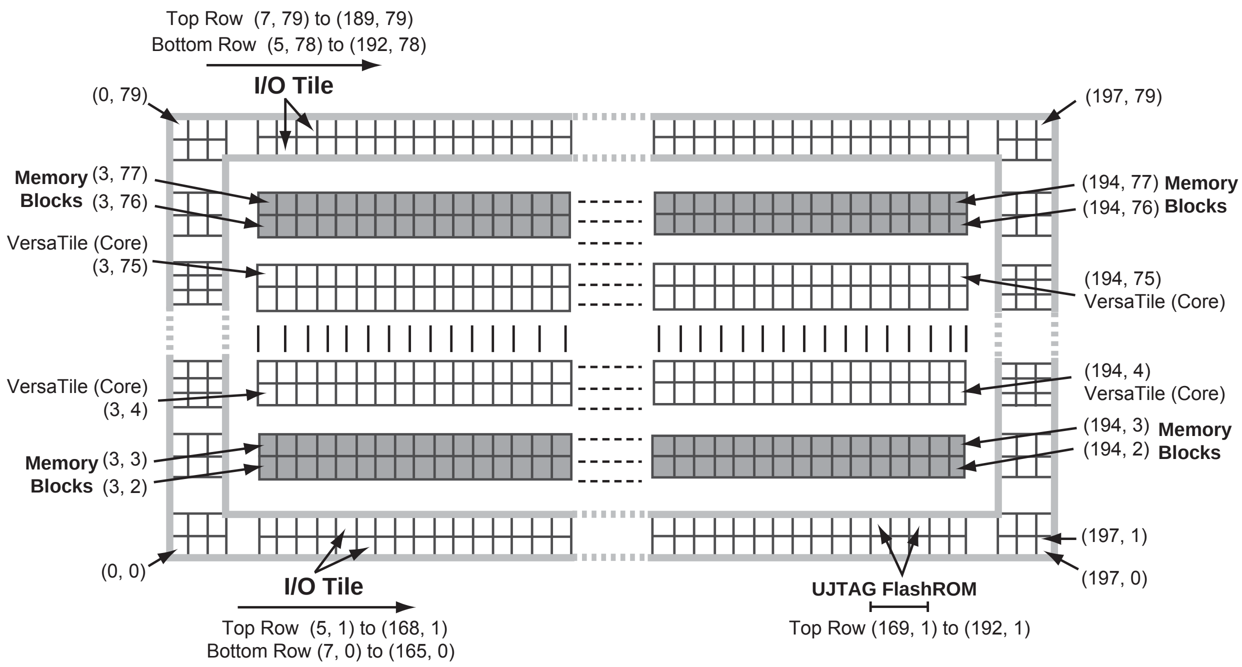
Device		**VersaTiles**				**Memory Rows**		**Entire Die**	
		Min.		**Max.**		**Bottom**	**Top**	**Min.**	**Max.**
IGLOO	**ProASIC3/ ProASIC3L**	**x**	**y**	**x**	**y**	**(x, y)**	**(x, y)**	**(x, y)**	**(x, y)**
AGL015	A3P015	3	2	34	13	None	None	(0, 0)	(37, 15)
AGL030	A3P030	3	3	66	13	None	None	(0, 0)	(69, 15)
AGL060	A3P060	3	2	66	25	None	(3, 26)	(0, 0)	(69, 29)
AGL125	A3P125	3	2	130	25	None	(3, 26)	(0, 0)	(133, 29)
AGL250	A3P250/L	3	2	130	49	None	(3, 50)	(0, 0)	(133, 53)
AGL400	A3P400	3	2	194	49	None	(3, 50)	(0, 0)	(197, 53)
AGL600	A3P600/L	3	4	194	75	(3, 2)	(3, 76)	(0, 0)	(197, 79)
AGL1000	A3P1000/L	3	4	258	99	(3, 2)	(3, 100)	(0, 0)	(261, 103)
AGLE600	A3PE600/L, RT3PE600L	3	4	194	75	(3, 2)	(3, 76)	(0, 0)	(197, 79)
	A3PE1500	3	4	322	123	(3, 2)	(3, 124)	(0, 0)	(325, 127)
AGLE3000	A3PE3000/L, RT3PE3000L	3	6	450	173	(3, 2) or (3, 4)	(3, 174) or (3, 176)	(0, 0)	(453, 179)

Table 1-3 **• IGLOO PLUS Array Coordinates**

Device	**VersaTiles**				**Memory Rows**		**Entire Die**	
	Min.		**Max.**		**Bottom**	**Top**	**Min.**	**Max.**
IGLOO PLUS	**x**	**y**	**x**	**y**	**(x, y)**	**(x, y)**	**(x, y)**	**(x, y)**
AGLP030	2	3	67	13	None	None	(0, 0)	(69, 15)
AGLP060	2	2	67	25	None	(3, 26)	(0, 0)	(69, 29)
AGLP125	2	2	131	25	None	(3, 26)	(0, 0)	(133, 29)

Table 1-4 • **IGLOO nano and ProASIC3 nano Array Coordinates**

Device		VersaTiles		Memory Rows		Entire Die	
		Min.	**Max.**	**Bottom**	**Top**	**Min.**	**Max.**
IGLOO nano	**ProASIC3 nano**	**(x, y)**	**(x, y)**	**(x, y)**	**(x, y)**	**(x, y)**	**(x, y)**
AGLN010	A3P010	(0, 2)	(32, 5)	None	None	(0, 0)	(34, 5)
AGLN015	A3PN015	(0, 2)	(32, 9)	None	None	(0, 0)	(34, 9)
AGLN020	A3PN020	(0, 2)	32, 13)	None	None	(0, 0)	(34, 13)
AGLN060	A3PN060	(3, 2)	(66, 25)	None	(3, 26)	(0, 0)	(69, 29)
AGLN125	A3PN125	(3, 2)	(130, 25)	None	(3, 26)	(0, 0)	(133, 29)
AGLN250	A3PN250	(3, 2)	(130, 49)	None	(3, 50)	(0, 0)	(133, 49)



Note: *The vertical I/O tile coordinates are not shown. West-side coordinates are {(0, 2) to (2, 2)} to {(0, 77) to (2, 77)}; east-side coordinates are {(195, 2) to (197, 2)} to {(195, 77) to (197, 77)}.*

Figure 1-9 • **Array Coordinates for AGL600, AGLE600, A3P600, and A3PE600**

Routing Architecture

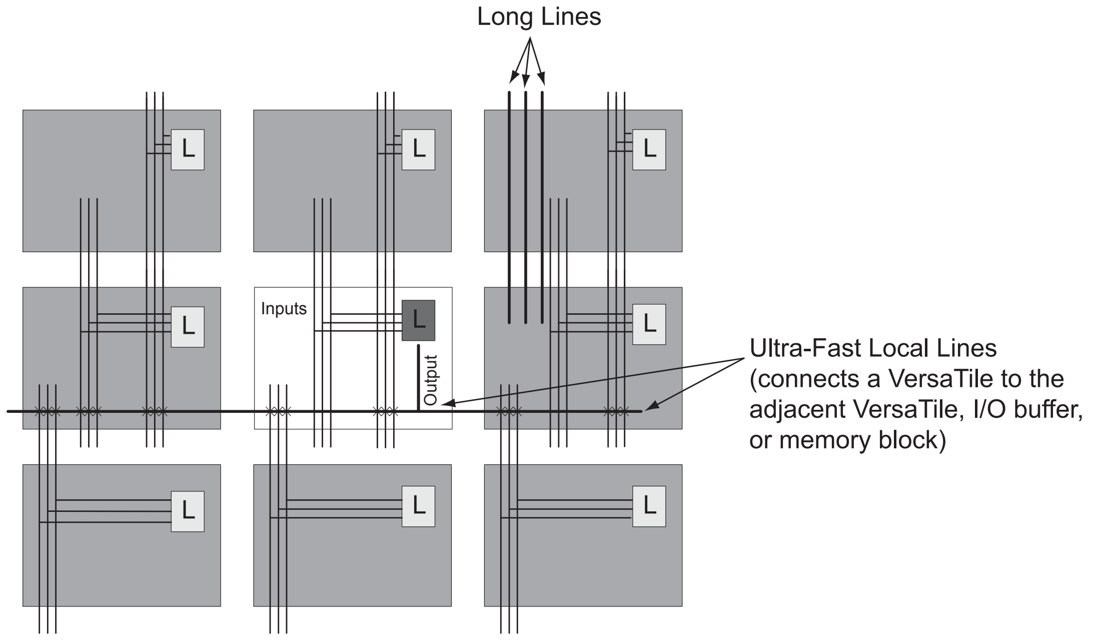
The routing structure of low power flash devices is designed to provide high performance through a flexible four-level hierarchy of routing resources: ultra-fast local resources; efficient long-line resources; high-speed, very-long-line resources; and the high-performance VersaNet networks.

The ultra-fast local resources are dedicated lines that allow the output of each VersaTile to connect directly to every input of the eight surrounding VersaTiles (Figure 1-10). The exception to this is that the SET/CLR input of a VersaTile configured as a D-flip-flop is driven only by the VersaTile global network.

The efficient long-line resources provide routing for longer distances and higher-fanout connections. These resources vary in length (spanning one, two, or four VersaTiles), run both vertically and horizontally, and cover the entire device (Figure 1-11 on page 21). Each VersaTile can drive signals onto the efficient long-line resources, which can access every input of every VersaTile. Routing software automatically inserts active buffers to limit loading effects.

The high-speed, very-long-line resources, which span the entire device with minimal delay, are used to route very long or high-fanout nets: length ±12 VersaTiles in the vertical direction and length ±16 in the horizontal direction from a given core VersaTile (Figure 1-12 on page 21). Very long lines in low power flash devices have been enhanced over those in previous ProASIC families. This provides a significant performance boost for long-reach signals.

The high-performance VersaNet global networks are low-skew, high-fanout nets that are accessible from external pins or internal logic. These nets are typically used to distribute clocks, resets, and other high-fanout nets requiring minimum skew. The VersaNet networks are implemented as clock trees, and signals can be introduced at any junction. These can be employed hierarchically, with signals accessing every input of every VersaTile. For more details on VersaNets, refer to the "Global Resources in Low Power Flash Devices" section on page 33.



Note: *Input to the core cell for the D-flip-flop set and reset is only available via the VersaNet global network connection.*

***Figure 1-10* • Ultra-Fast Local Lines Connected to the Eight Nearest Neighbors**

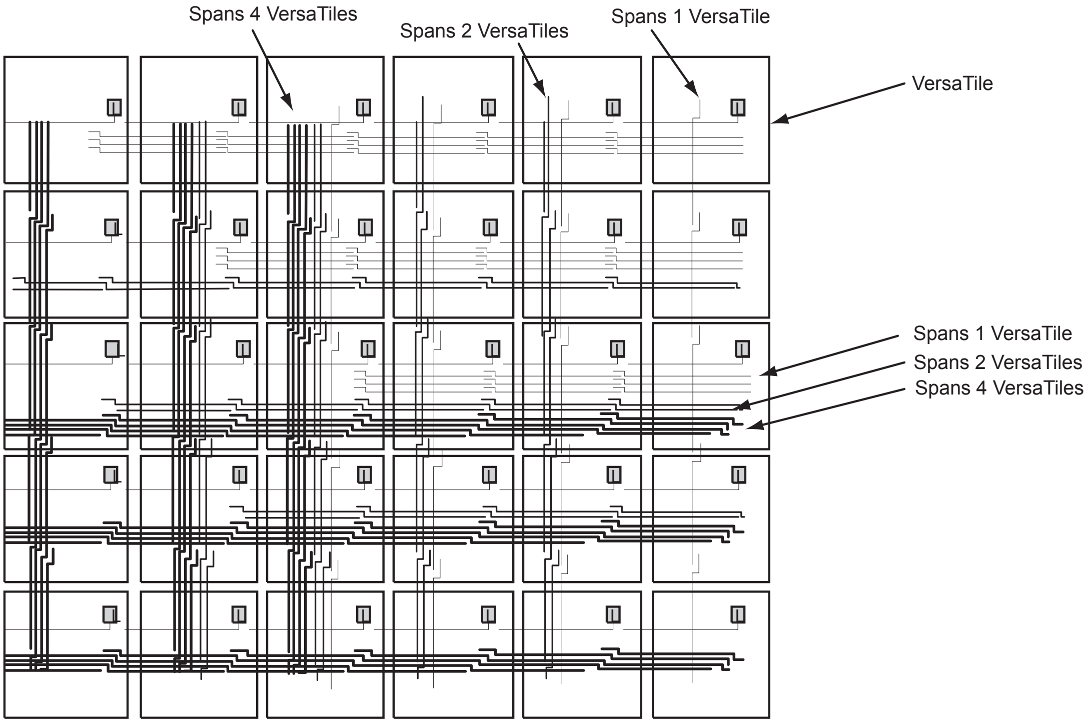


Figure 1-11 **• Efficient Long-Line Resources**

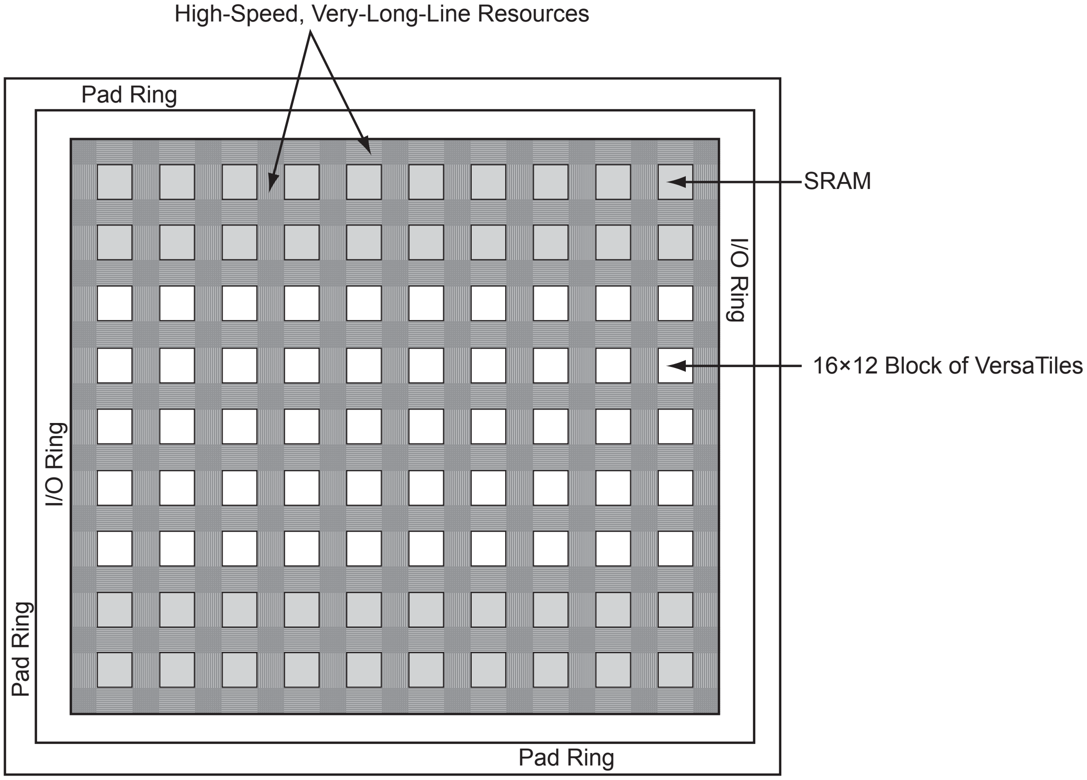

High-Speed, Very-Long-Line Resources
Pad Ring
SRAM
I/O Ring
16×12 Block of VersaTiles
I/O Ring
Pad Ring
Pad Ring

Figure 1-12 **• Very-Long-Line Resources**

Related Documents

User's Guides

Designer User's Guide

http://www.microsemi.com/soc/documents/designer_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	The "I/O State of Newly Shipped Devices" section is new (SAR 39542).	16
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 1-1 • Flash-Based FPGAs.	12
	Figure 1-2 • IGLOO and ProASIC3 nano Device Architecture Overview with Two I/O Banks (applies to 10 k and 30 k device densities, excluding IGLOO PLUS devices) through Figure 1-5 • IGLOO, IGLOO nano, ProASIC3 nano, and ProASIC3/L Device Architecture Overview with Four I/O Banks (AGL600 device is shown) are new.	13, 14
	Table 1-4 • IGLOO nano and ProASIC3 nano Array Coordinates is new.	19
v1.3 (October 2008)	The title of this document was changed from "Core Architecture of IGLOO and ProASIC3 Devices" to "FPGA Array Architecture in Low Power Flash Devices."	11
	The "FPGA Array Architecture Support" section was revised to include new families and make the information more concise.	12
	Table 1-2 • IGLOO and ProASIC3 Array Coordinates was updated to include Military ProASIC3/EL and RT ProASIC3 devices.	18
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 1-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	12
v1.1 (March 2008)	Table 1-1 • Flash-Based FPGAs and the accompanying text was updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "Device Overview" section are new.	12
	The "Device Overview" section was updated to note that 15 k devices do not support SRAM or FIFO.	13
	Figure 1-6 • IGLOO PLUS Device Architecture Overview with Four I/O Banks is new.	15
	Table 1-2 • IGLOO and ProASIC3 Array Coordinates was updated to add A3P015 and AGL015.	18
	Table 1-3 • IGLOO PLUS Array Coordinates is new.	18

2 – Low Power Modes in ProASIC3/E and ProASIC3 nano FPGAs

Introduction

The demand for low power systems and semiconductors, combined with the strong growth observed for value-based FPGAs, is driving growing demand for low power FPGAs. For portable and battery-operated applications, power consumption has always been the greatest challenge. The battery life of a system and on-board devices has a direct impact on the success of the product. As a result, FPGAs used in these applications should meet low power consumption requirements.

ProASIC®3/E and ProASIC3 nano FPGAs offer low power consumption capability inherited from their nonvolatile and live-at-power-up (LAPU) flash technology. This application note describes the power consumption and how to use different power saving modes to further reduce power consumption for power-conscious electronics design.

Power Consumption Overview

In evaluating the power consumption of FPGA technologies, it is important to consider it from a system point of view. Generally, the overall power consumption should be based on static, dynamic, inrush, and configuration power. Few FPGAs implement ways to reduce static power consumption utilizing sleep modes.

SRAM-based FPGAs use volatile memory for their configuration, so the device must be reconfigured after each power-up cycle. Moreover, during this initialization state, the logic could be in an indeterminate state, which might cause inrush current and power spikes. More complex power supplies are required to eliminate potential system power-up failures, resulting in higher costs. For portable electronics requiring frequent power-up and -down cycles, this directly affects battery life, requiring more frequent recharging or replacement.

$$\text{SRAM-Based FPGA Total Power Consumption} = P_{static} + P_{dynamic} + P_{inrush} + P_{config} \qquad \textit{EQ 1}$$

$$\text{ProASIC3/E Total Power Consumption} = P_{static} + P_{dynamic} \qquad \textit{EQ 2}$$

Unlike SRAM-based FPGAs, Microsemi flash-based FPGAs are nonvolatile and do not require power-up configuration. Additionally, Microsemi nonvolatile flash FPGAs are live at power-up and do not require additional support components. Total power consumption is reduced as the inrush current and configuration power components are eliminated.

Note that the static power component can be reduced in flash FPGAs (such as the ProASIC3/E devices) by entering User Low Static mode or Sleep mode. This leads to an extremely low static power component contribution to the total system power consumption.

The following sections describe the usage of Static (Idle) mode to reduce the power component, User Low Static mode to reduce the static power component, and Sleep mode and Shutdown mode to achieve a range of power consumption when the FPGA or system is idle. Table 2-1 on page 24 summarizes the different low power modes offered by ProASIC3/E devices.

Table 2-1 **• ProASIC3/E/nano Low Power Modes Summary**

Mode	Power Supplies / Clock Status	Needed to Start Up
Active	On – All, clock Off – None	N/A (already active)
Static (Idle)	On – All Off – No active clock in FPGA Optional: Enter User Low Static (Idle) Mode by enabling ULSICC macro to further reduce power consumption by powering down FlashROM.	Initiate clock source. No need to initialize volatile contents.
Sleep	On – VCCI Off – VCC (core voltage), VJTAG (JTAG DC voltage), and VPUMP (programming voltage) LAPU enables immediate operation when power returns. Optional: Save state of volatile contents in external memory.	Need to turn on core. Load states from external memory. As needed, restore volatile contents from external memory.
Shutdown	On – None Off – All power supplies Applicable to all ProASIC3 nano devices, cold-sparing and hot-insertion allow the device to be powered down without bringing down the system. LAPU enables immediate operation when power returns.	Need to turn on VCC, VCCI.

Static (Idle) Mode

In Static (Idle) mode, the clock inputs are not switching and the static power consumption is the minimum power required to keep the device powered up. In this mode, I/Os are only drawing the minimum leakage current specified in the datasheet. Also, in Static (Idle) mode, embedded SRAM, I/Os, and registers retain their values, so the device can enter and exit this mode without any penalty.

If the embedded PLLs are used as the clock source, Static (Idle) mode can be entered easily by pulling LOW the PLL POWERDOWN pin (active-low). By pulling the PLL POWERDOWN pin to LOW, the PLL is turned off. Refer to Figure 2-1 on page 25 for more information.

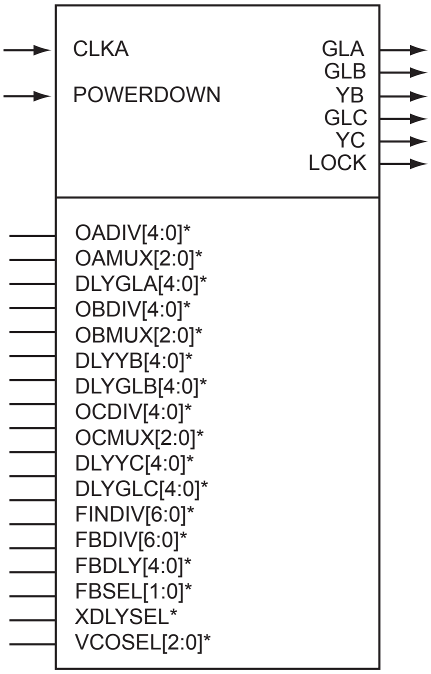


***Figure 2-1* • CCC/PLL Macro**

User Low Static (Idle) Mode

User Low Static (Idle) mode is an advanced feature supported by ProASIC3/E devices to reduce static (idle) power consumption. Entering and exiting this mode is made possible using the ULSICC macro by setting its value to disable/enable the User Low Static (Idle) mode. Under typical operating conditions, characterization results show up to 25% reduction of the static (idle) power consumption. The greatest power savings in terms of percentage are seen in the smaller members of the ProASIC3 family. The active-high control signal for User Low Static (Idle) mode can be generated by internal or external logic. When the device is operating in User Low Static (Idle) mode, FlashROM functionality is temporarily disabled to save power. If FlashROM functionality is needed, the device can exit User Low Static mode temporarily and re-enter the mode once the functionality is no longer needed.

To utilize User Low Static (Idle) mode, simply instantiate the ULSICC macro (Table 2-2 on page 26) in your design, and connect the input port to either an internal logic signal or a device package pin, as illustrated in Figure 2-2 on page 26 or Figure 2-3 on page 27, respectively. The attribute is used so the Synplify® synthesis tool will not optimize the instance with no output port.

This mode can be used to lower standard static (idle) power consumption when the FlashROM feature is not needed. Configuring the device to enter User Low Static (Idle) mode is beneficial when the FPGA enters and exits static mode frequently and lowering power consumption as much as possible is desired. The device is still functional, and data is retained in this state so the device can enter and exit this mode quickly, resulting in reduced total power consumption. The device can also stay in User Low Static mode when the FlashROM feature is not used in the device.

Table 2-2 • **Using ULSICC Macro***

VHDL	Verilog

COMPONENT ULSICC
       port (
             LSICC   : in   STD_ULOGIC);
END COMPONENT;
 	
Example:
COMPONENT ULSICC
       port (
             LSICC   : in   STD_ULOGIC);
END COMPONENT;
 	
attribute syn_noprune : boolean;
attribute syn_noprune of u1 : label is true;
u1: ULSICC port map(myInputSignal);
	module ULSICC(LSICC);
	 input LSICC;
	endmodule
	
	Example:
	ULSICC U1(.LSICC(myInputSignal))
	/* synthesis syn_noprune=1 */;

*Note: *Supported in Libero® software v7.2 and newer versions.*

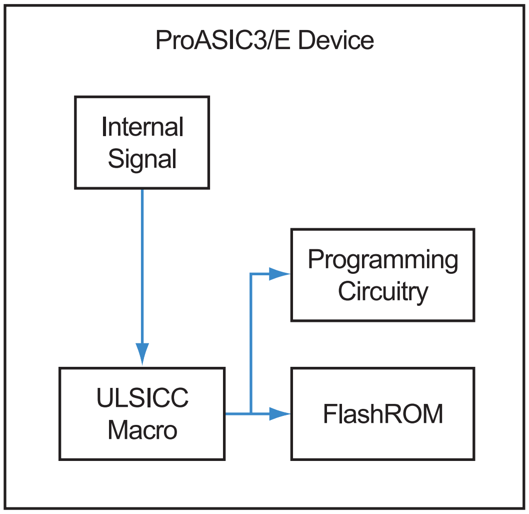


Figure 2-2 • **User Low Static (Idle) Mode Application—Internal Control Signal**

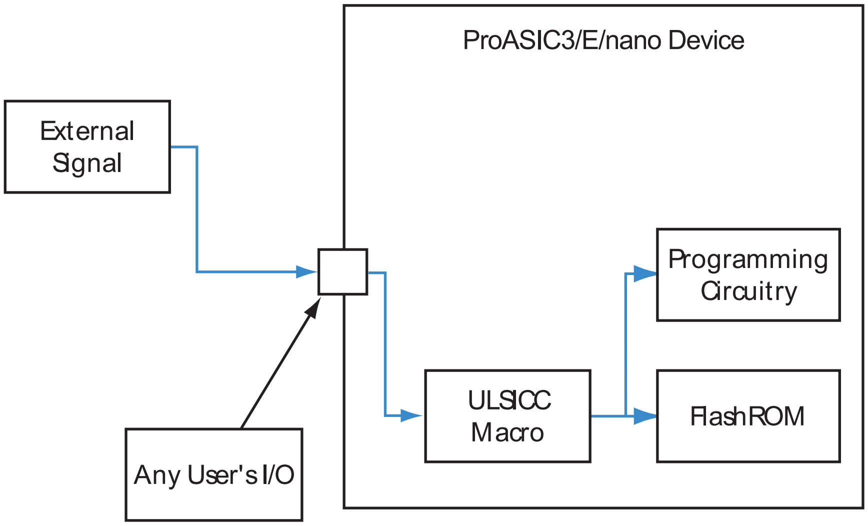


Figure 2-3 **• User Low Static (Idle) Mode Application—External Control Signal**

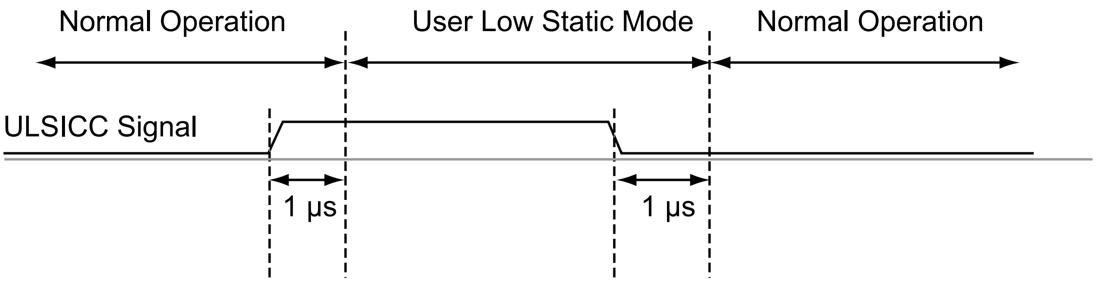


Figure 2-4 **• User Low Static (Idle) Mode Timing Diagram**

Sleep Mode

ProASIC3/E and ProASIC3 nano FPGAs support Sleep mode when device functionality is not required. In Sleep mode, the VCC (core voltage), VJTAG (JTAG DC voltage), and VPUMP (programming voltage) are grounded, resulting in the FPGA core being turned off to reduce power consumption. While the ProASIC3/E device is in Sleep mode, the rest of the system is still operating and driving the input buffers of the ProASIC3/E device. The driven inputs do not pull up power planes, and the current draw is limited to a minimal leakage current.

Table 2-3 shows the status of the power supplies in Sleep mode. When a power supply is powered off, the corresponding power pin can be left floating or grounded.

Table 2-3 **• Sleep Mode—Power Supply Requirements for ProASIC3/E/nano Devices**

Power Supplies	ProASIC3/E/nano Device
VCC	Powered off
VCCI = VMV	Powered on
VJTAG	Powered off
VPUMP	Powered off

Table 2-4 shows the current draw in Sleep mode for an A3P250 device with the following test conditions: VCCI = VMV; VCC = VJTAG = VPUMP = GND.

***Table 2-4* • A3P250 Current Draw in Sleep Mode**

Typical Conditions	A3P250	
	I_{CCI} (µA)	**I_{CCI} (µA) per Bank**
VCCI = 3.3 V	31.57	7.89
VCCI = 2.5 V	23.96	5.99
VCCI = 1.8 V	17.32	4.33
VCCI = 1.5 V	14.46	3.62
I_{CC} FPGA Core	0.0	0.0
Leakage Current per I/O	0.1	0.1
VPUMP	0.0	0.0

Note: The data in this table were taken under typical conditions and are based on characterization. The data is not guaranteed.

Table 2-5 shows the current draw in Sleep mode for an A3PE600 device with the following test conditions: VCCI = VMV; VCC = VJTAG = VPUMP = GND.

***Table 2-5* • A3PE600 Current Draw in Sleep Mode**

Typical Conditions	A3PE600	
	I_{CCI} (µA)	**I_{CCI} (µA) per Bank**
VCCI = 3.3 V	59.85	7.48
VCCI = 2.5 V	45.50	5.69
VCCI = 1.8 V	32.98	4.12
VCCI = 1.5 V	27.66	3.46
VCCI = 0 V or Floating	0.0	0.0
I_{CC} FPGA Core	0.0	0.0
Leakage Current per I/O	0.1	0.1
I_{PUMP}	0.0	0.0

Note: The data in this table were taken under typical conditions and are based on characterization. The data is not guaranteed.

ProASIC3/E and ProASIC3 nano devices were designed such that before device power-up, all I/Os are in tristate mode. The I/Os will remain tristated during power-up until the last voltage supply (VCC or VCCI) is powered to its functional level. After the last supply reaches the functional level, the outputs will exit the tristate mode and drive the logic at the input of the output buffer. The behavior of user I/Os is independent of the VCC and VCCI sequence or the state of other FPGA voltage supplies (VPUMP and VJTAG). During power-down, device I/Os become tristated once the first power supply (VCC or VCCI) drops below its brownout voltage level. The I/O behavior during power-down is also independent of voltage supply sequencing.

Figure 2-5 on page 29 shows a timing diagram for the FPGA core entering the activation and deactivation trip points for a typical application when the VCC power supply ramp rate is 100 µs (ramping from 0 V to 1.5 V). This is, in fact, the timing diagram for the FPGA entering and exiting Sleep mode, as it is dependent on powering down or powering up VCC. Depending on the ramp rate of the power supply and board-level configurations, the user can easily calculate how long it takes for the core to become active or inactive. For more information, refer to the "Power-Up/-Down Behavior of Low Power Flash Devices" section on page 387.

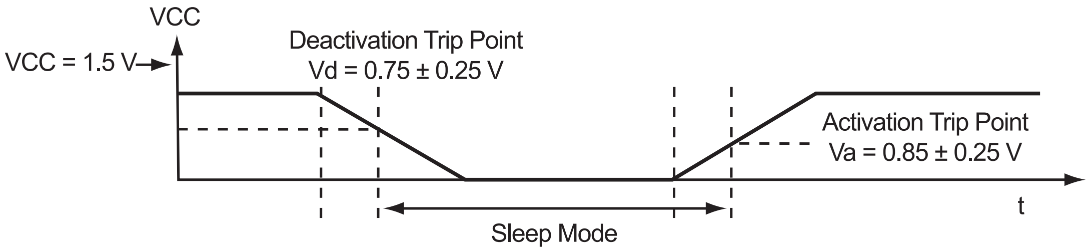


Figure 2-5 **• Entering and Exiting Sleep Mode—Typical Timing Diagram**

Shutdown Mode

For all ProASIC3/E and ProASIC3 nano devices, shutdown mode can be entered by turning off all power supplies when device functionality is not needed. Cold-sparing and hot-insertion features in ProASIC3 nano devices enable the device to be powered down without turning off the entire system. When power returns, the live at power-up feature enables immediate operation of the device.

Using Sleep Mode or Shutdown Mode in the System

Depending on the power supply and components used in an application, there are many ways to turn the power supplies connected to the device on or off. For example, Figure 2-6 shows how a microprocessor is used to control a power FET. It is recommended that power FETs with low on resistance be used to perform the switching action.

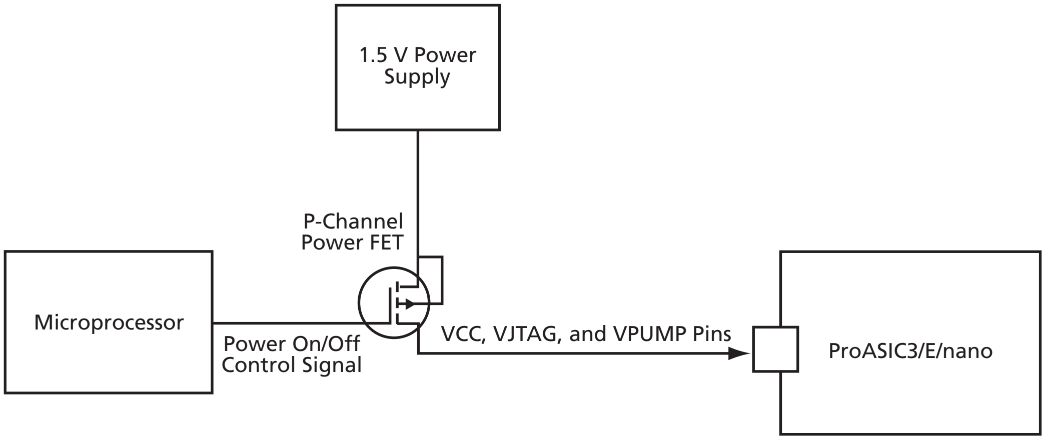


Figure 2-6 **• Controlling Power On/Off State Using Microprocessor and Power FET**

Alternatively, Figure 2-7 shows how a microprocessor can be used with a voltage regulator's shutdown pin to turn the power supplies connected to the device on or off.

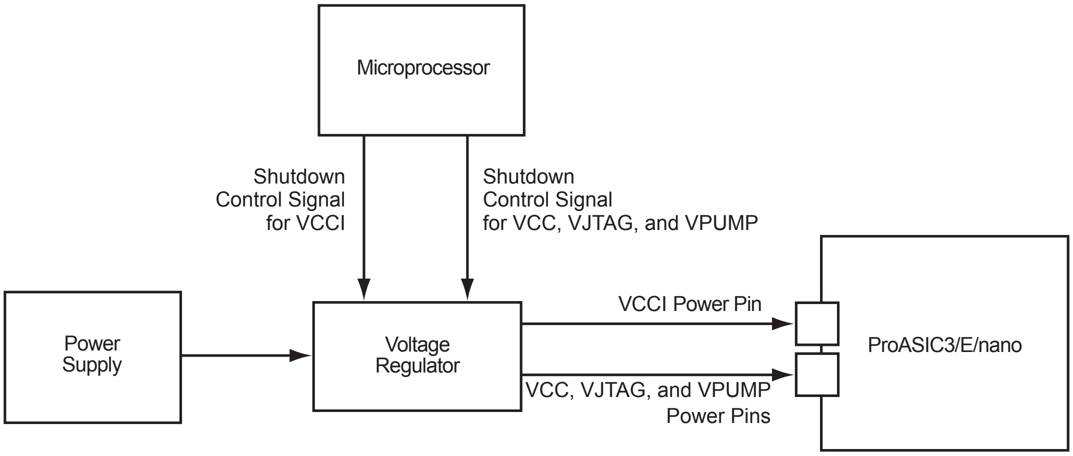

Microprocessor
Shutdown Control Signal for VCCI
Shutdown Control Signal for VCC, VJTAG, and VPUMP
Power Supply
Voltage Regulator
VCCI Power Pin
VCC, VJTAG, and VPUMP Power Pins
ProASIC3/E/nano

Figure 2-7 • **Controlling Power On/Off State Using Microprocessor and Voltage Regulator**

Though Sleep mode or Shutdown mode can be used to save power, the content of the SRAM and the state of the registers is lost when power is turned off if no other measure is taken. To keep the original contents of the device, a low-cost external serial EEPROM can be used to save and restore the device contents when entering and exiting Sleep mode. In the *Embedded SRAM Initialization Using External Serial EEPROM* application note, detailed information and a reference design are provided to initialize the embedded SRAM using an external serial EEPROM. The user can easily customize the reference design to save and restore the FPGA state when entering and exiting Sleep mode. The microcontroller will need to manage this activity, so before powering down VCC, the data must be read from the FPGA and stored externally. Similarly, after the FPGA is powered up, the microcontroller must allow the FPGA to load the data from external memory and restore its original state.

Conclusion

Microsemi ProASIC3/E and ProASIC3 nano FPGAs inherit low power consumption capability from their nonvolatile and live-at-power-up flash-based technology. Power consumption can be reduced further using the Static (Idle), User Low Static (Idle), Sleep, or Shutdown power modes. All these features result in a low-power, cost-effective, single-chip solution designed specifically for power-sensitive electronics applications.

Related Documents

Application Notes

Embedded SRAM Initialization Using External Serial EEPROM

http://www.microsemi.com/soc/documents/EmbeddedSRAMInit_AN.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
June 2011	Table 2-1 • ProASIC3/E/nano Low Power Modes Summary and the "Shutdown Mode" section were revised to remove reference to ProASIC3/E devices (SAR 24526).	24, 29
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.2 (August 2008)	References to ProASIC3 nano devices were added to the document where appropriate.	N/A
	VJTAG and VPUMP were noted as "Off" in the Sleep Mode section of Table 2-1 • ProASIC3/E/nano Low Power Modes Summary.	24
	The "Sleep Mode" section, including Table 2-3 • Sleep Mode—Power Supply Requirements for ProASIC3/E/nano Devices, was revised to state that VJTAG and VPUMP are powered off during Sleep mode.	27
	The text above Table 2-4 • A3P250 Current Draw in Sleep Mode and Table 2-5 • A3PE600 Current Draw in Sleep Mode was revised to state "VCC = VJTAG = VPUMP = GND."	28
	Figure 2-6 • Controlling Power On/Off State Using Microprocessor and Power FET and Figure 2-7 • Controlling Power On/Off State Using Microprocessor and Voltage Regulator were revised to show shutdown of VJTAG and VPUMP during Sleep mode.	29, 30
v1.1 (March 2008)	The part number for this document was changed from 51700094-002-0 to 51700094-003-1.	N/A
v1.0 (January 2008)	The Power Supplies / Clock Status description was updated for Static (Idle) in Table 2-1 • ProASIC3/E/nano Low Power Modes Summary.	24
	Programming information was updated in the "User Low Static (Idle) Mode" section.	25
51900138-2/10.06	The "User Low Static (Idle) Mode" section was updated to include information about allowing programming in the ULSICC mode.	25
	Figure 2-2 • User Low Static (Idle) Mode Application—Internal Control Signal was updated.	26
	Figure 2-3 • User Low Static (Idle) Mode Application—External Control Signal was updated.	27
51900138-1/6.06	In Table 2-4 • A3P250 Current Draw in Sleep Mode, "VCCI = 1.5 V" was changed from 3.6158 to 3.62.	28
	In Table 2-5 • A3PE600 Current Draw in Sleep Mode, "VCCI = 2.5 V" was changed from 5.6875 to 3.69.	28

3 – Global Resources in Low Power Flash Devices

Introduction

IGLOO, Fusion, and ProASIC3 FPGA devices offer a powerful, low-delay VersaNet global network scheme and have extensive support for multiple clock domains. In addition to the Clock Conditioning Circuits (CCCs) and phase-locked loops (PLLs), there is a comprehensive global clock distribution network called a VersaNet global network. Each logical element (VersaTile) input and output port has access to these global networks. The VersaNet global networks can be used to distribute low-skew clock signals or high-fanout nets. In addition, these highly segmented VersaNet global networks contain spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside their region. This allows users the flexibility to create low-skew local clock networks using spines. This document describes VersaNet global networks and discusses how to assign signals to these global networks and spines in a design flow. Details concerning low power flash device PLLs are described in the "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 63. This chapter describes the low power flash devices' global architecture and uses of these global networks in designs.

Global Architecture

Low power flash devices offer powerful and flexible control of circuit timing through the use of global circuitry. Each chip has up to six CCCs, some with PLLs.

- In IGLOOe, ProASIC3EL, and ProASIC3E devices, all CCCs have PLLs—hence, 6 PLLs per device (except the PQ208 package, which has only 2 PLLs).
- In IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3, and ProASIC3L devices, the west CCC contains a PLL core (except in 10 k through 30 k devices).
- In Fusion devices, the west CCC also contains a PLL core. In the two larger devices (AFS600 and AFS1500), the west and east CCCs each contain a PLL.

Refer to Table 4-6 on page 86 for details. Each PLL includes delay lines, a phase shifter (0°, 90°, 180°, 270°), and clock multipliers/dividers. Each CCC has all the circuitry needed for the selection and interconnection of inputs to the VersaNet global network. The east and west CCCs each have access to three chip global lines on each side of the chip (six chip global lines total). The CCCs at the four corners each have access to three quadrant global lines in each quadrant of the chip (except in 10 k through 30 k gate devices).

The nano 10 k, 15 k, and 20 k devices support four VersaNet global resources, and 30 k devices support six global resources. The 10 k through 30 k devices have simplified CCCs called CCC-GLs.

The flexible use of the VersaNet global network allows the designer to address several design requirements. User applications that are clock-resource-intensive can easily route external or gated internal clocks using VersaNet global routing networks. Designers can also drastically reduce delay penalties and minimize resource usage by mapping critical, high-fanout nets to the VersaNet global network.

Note: Microsemi recommends that you choose the appropriate global pin and use the appropriate global resource so you can realize these benefits.

The following sections give an overview of the VersaNet global network, the structure of the global network, access point for the global networks, and the clock aggregation feature that enables a design to have very low clock skew using spines.

Global Resource Support in Flash-Based Devices

The flash FPGAs listed in Table 3-1 support the global resources and the functions described in this document.

***Table 3-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
	IGLOO nano	The industry's lowest-power, smallest-size solution
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO products as listed in Table 3-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 3-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

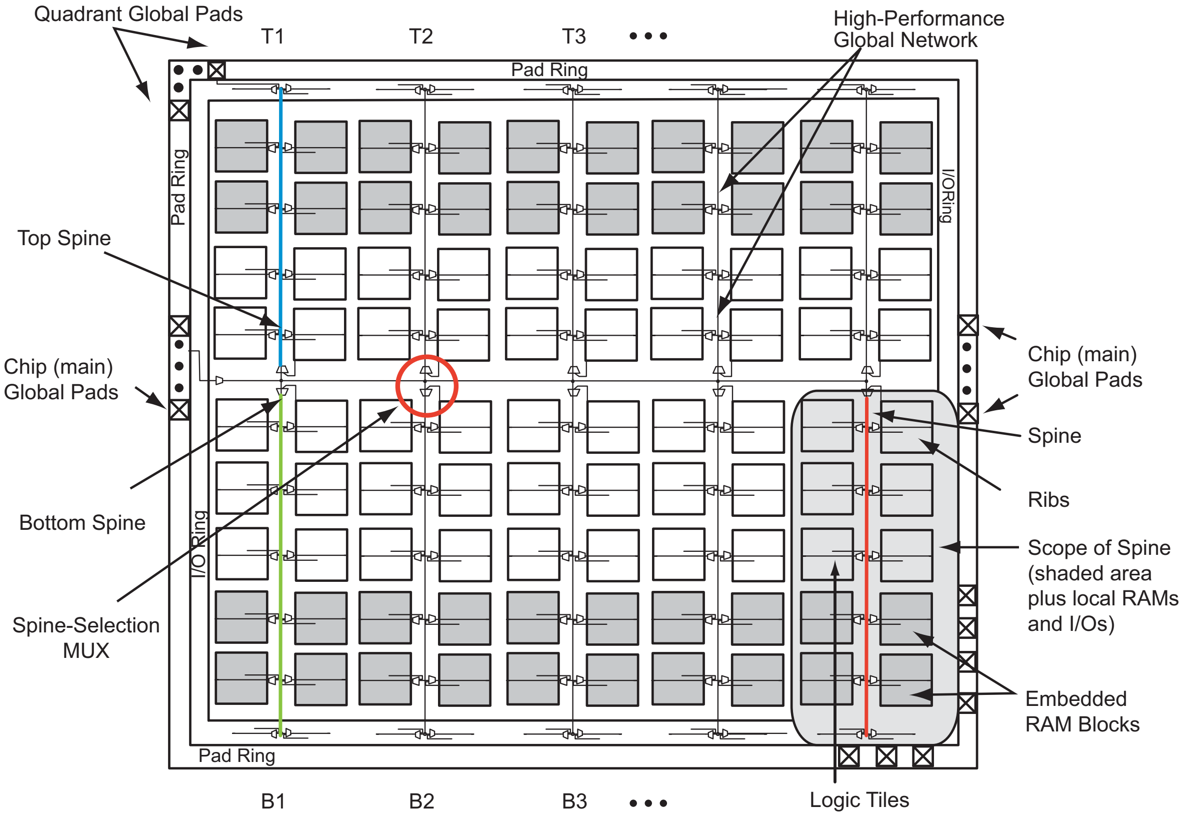
VersaNet Global Network Distribution

One of the architectural benefits of low power flash architecture is the set of powerful, low-delay VersaNet global networks that can access the VersaTiles, SRAM, and I/O tiles of the device. Each device offers a chip global network with six global lines (except for nano 10 k, 15 k, and 20 k gate devices) that are distributed from the center of the FPGA array. In addition, each device (except the 10 k through 30 k gate device) has four quadrant global networks, each consisting of three quadrant global net resources. These quadrant global networks can only drive a signal inside their own quadrant. Each VersaTile has access to nine global line resources—three quadrant and six chip-wide (main) global networks—and a total of 18 globals are available on the device (3 × 4 regional from each quadrant and 6 global).

Figure 3-1 shows an overview of the VersaNet global network and device architecture for devices 60 k and above. Figure 3-2 and Figure 3-3 on page 36 show simplified VersaNet global networks.

The VersaNet global networks are segmented and consist of spines, global ribs, and global multiplexers (MUXes), as shown in Figure 3-1. The global networks are driven from the global rib at the center of the die or quadrant global networks at the north or south side of the die. The global network uses the MUX trees to access the spine, and the spine uses the clock ribs to access the VersaTile. Access is available to the chip or quadrant global networks and the spines through the global MUXes. Access to the spine using the global MUXes is explained in the "Spine Architecture" section on page 43.

These VersaNet global networks offer fast, low-skew routing resources for high-fanout nets, including clock signals. In addition, these highly segmented global networks offer users the flexibility to create low-skew local clock networks using spines for up to 252 internal/external clocks or other high-fanout nets in low power flash devices. Optimal usage of these low-skew networks can result in significant improvement in design performance.



Note: Not applicable to 10 k through 30 k gate devices

***Figure 3-1* • Overview of VersaNet Global Network and Device Architecture**

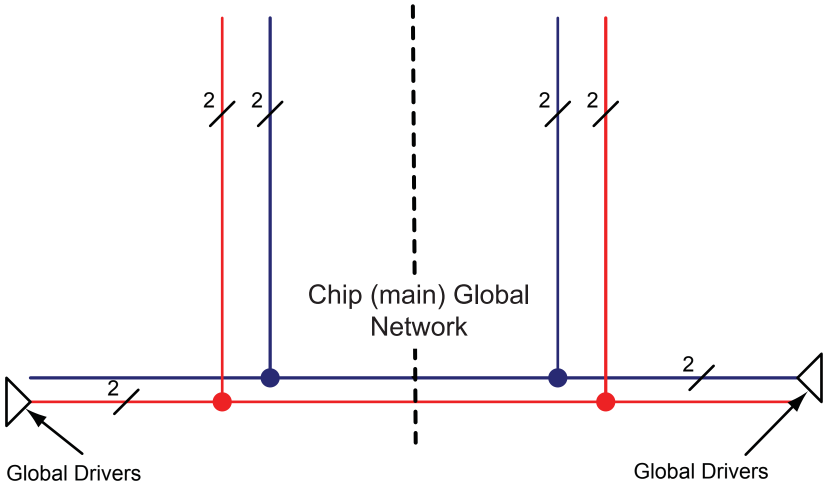


***Figure 3-2* • Simplified VersaNet Global Network (30 k gates and below)**

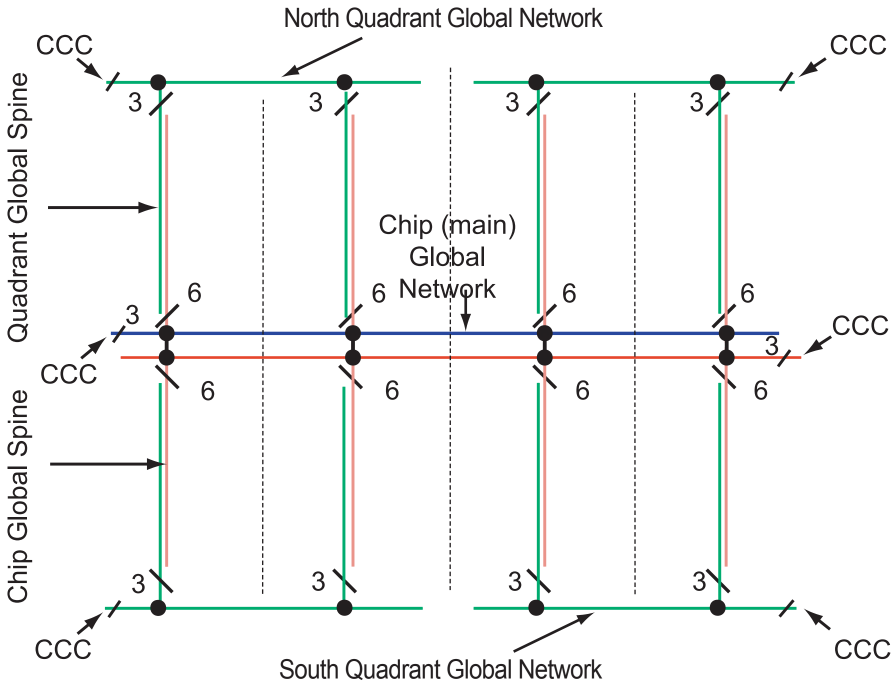

North Quadrant Global Network
CCC
CCC
Quadrant Global Spine
3
3
3
3
Chip (main) Global Network
6
6
6
6
3
CCC
CCC
3
6
6
6
6
Chip Global Spine
3
3
3
3
CCC
CCC
South Quadrant Global Network

***Figure 3-3* • Simplified VersaNet Global Network (60 k gates and above)**

Chip and Quadrant Global I/Os

The following sections give an overview of naming conventions and other related I/O information.

Naming of Global I/Os

In low power flash devices, the global I/Os have access to certain clock conditioning circuitry and have direct access to the global network. Additionally, the global I/Os can be used as regular I/Os, since they have identical capabilities to those of regular I/Os. Due to the comprehensive and flexible nature of the I/Os in low power flash devices, a naming scheme is used to show the details of the I/O. The global I/O uses the generic name Gmn/IOuxwByVz. Note that Gmn refers to a global input pin and IOuxwByVz refers to a regular I/O Pin, as these I/Os can be used as either global or regular I/Os. Refer to the I/O Structures chapter of the user's guide for the device that you are using for more information on this naming convention.

Figure 3-4 represents the global input pins connection. It shows all 54 global pins available to access the 18 global networks in ProASIC3E families.

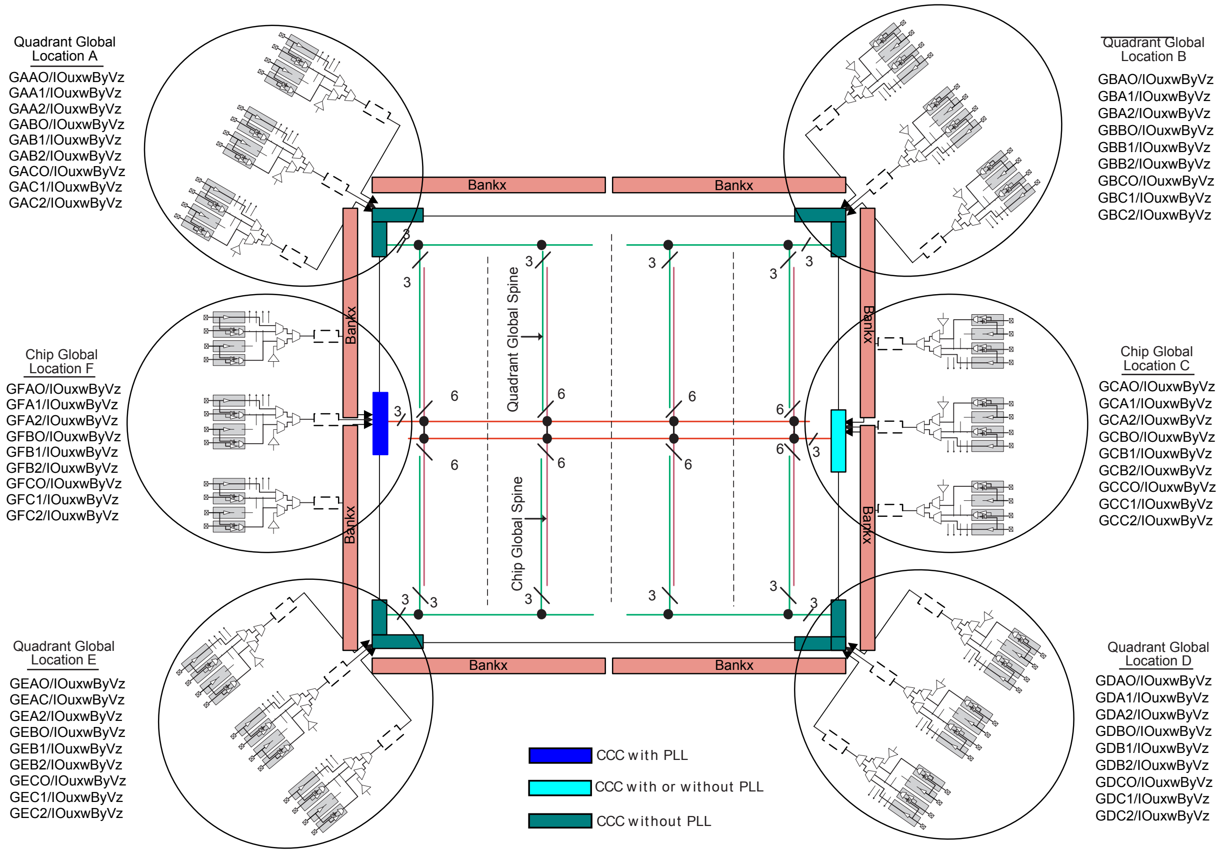
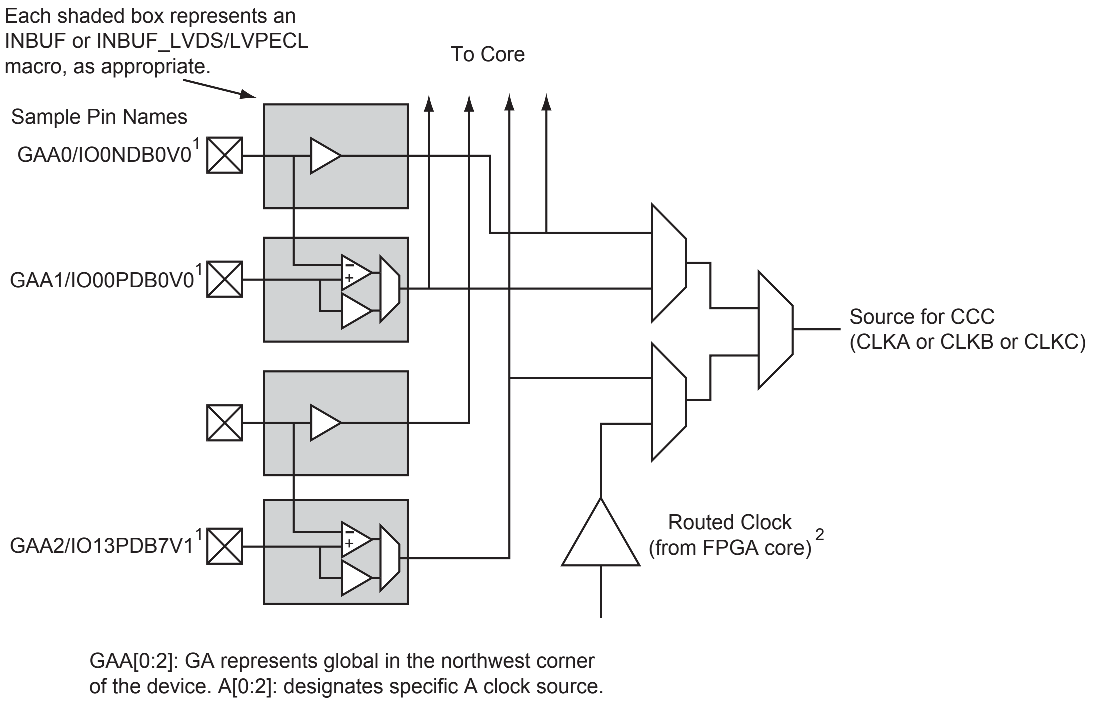


Figure 3-4 **• Global Connections Details**

Figure 3-5 shows more detailed global input connections. It shows the global input pins connection to the northwest quadrant global networks. Each global buffer, as well as the PLL reference clock, can be driven from one of the following:

- 3 dedicated single-ended I/Os using a hardwired connection
- 2 dedicated differential I/Os using a hardwired connection (not supported for IGLOO nano or ProASIC3 nano devices)
- The FPGA core


Each shaded box represents an INBUF or INBUF_LVDS/LVPECL macro, as appropriate.
To Core
Sample Pin Names
GAA0/IO0NDB0V0¹
GAA1/IO00PDB0V0¹
GAA2/IO13PDB7V1¹
Source for CCC (CLKA or CLKB or CLKC)
Routed Clock (from FPGA core)²
GAA[0:2]: GA represents global in the northwest corner of the device. A[0:2]: designates specific A clock source.

Note: Differential inputs are not supported for IGLOO nano or ProASIC3 nano devices.

***Figure 3-5* • Global I/O Overview**

Figure 3-6 shows all nine global inputs for the location A connected to the top left quadrant global network via CCC.

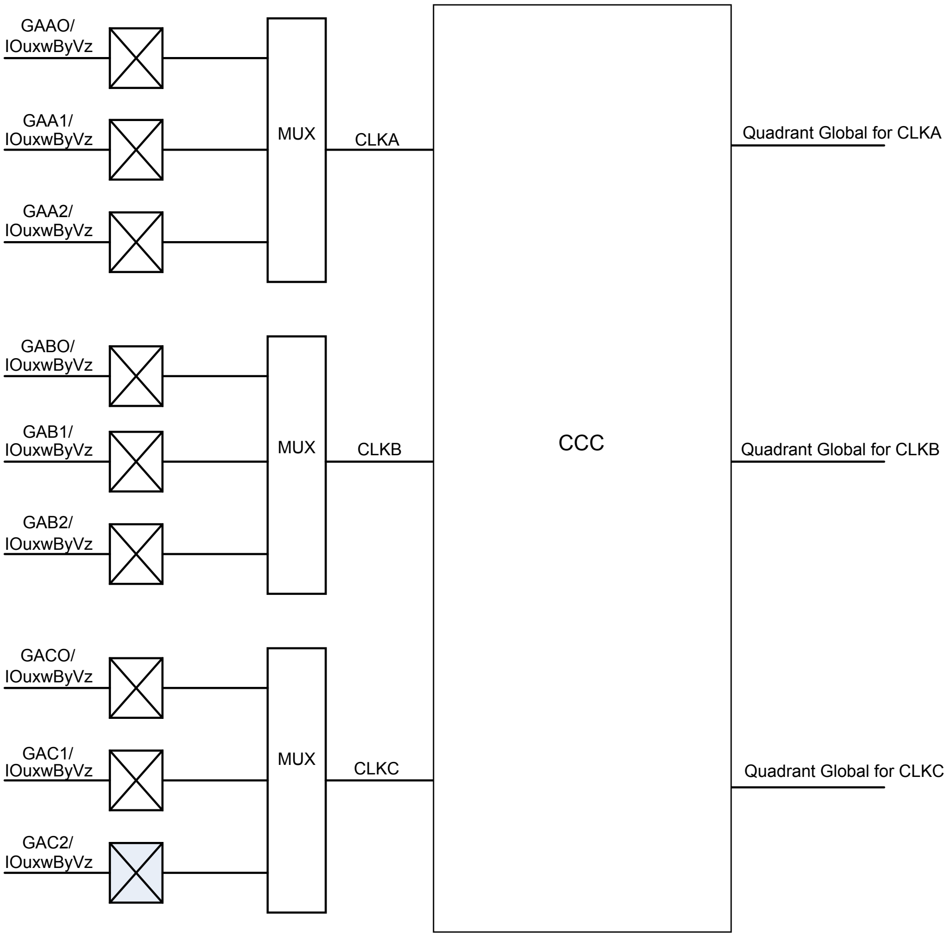


***Figure 3-6* • Global Inputs**

Since each bank can have a different I/O standard, the user should be careful to choose the correct global I/O for the design. There are 54 global pins available to access 18 global networks. For the single-ended and voltage-referenced I/O standards, you can use any of these three available I/Os to access the global network. For differential I/O standards such as LVDS and LVPECL, the I/O macro needs to be placed on (A0, A1), (B0, B1), (C0, C1), or a similar location. The unassigned global I/Os can be used as regular I/Os. Note that pin names starting with GF and GC are associated with the chip global networks, and GA, GB, GD, and GE are used for quadrant global networks. Table 3-2 on page 40 and Table 3-3 on page 41 show the general chip and quadrant global pin names.

Table 3-2 • **Chip Global Pin Name**

I/O Type	Beginning of I/O Name	Notes
Single-Ended	GFAO/IOuxwByVz GFA1/IOuxwByVz GFA2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GFBO/IOuxwByVz GFB1/IOuxwByVz GFB2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GFC0/IOuxwByVz GFC1/IOuxwByVz GFC2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GCAO/IOuxwByVz GCA1/IOuxwByVz GCA2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GCBO/IOuxwByVz GCB1/IOuxwByVz GCB2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GCC0/IOuxwByVz GCC1/IOuxwByVz GCC2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
Differential I/O Pairs	GFAO/IOuxwByVz GFA1/IOuxwByVz	The output of the different pair will drive the chip global.
	GFBO/IOuxwByVz GFB1/IOuxwByVz	The output of the different pair will drive the chip global.
	GFCO/IOuxwByVz GFC1/IOuxwByVz	The output of the different pair will drive the chip global.
	GCAO/IOuxwByVz GCA1/IOuxwByVz	The output of the different pair will drive the chip global.
	GCBO/IOuxwByVz GCB1/IOuxwByVz	The output of the different pair will drive the chip global.
	GCCO/IOuxwByVz GCC1/IOuxwByVz	The output of the different pair will drive the chip global.

Note: Only one of the I/Os can be directly connected to a quadrant at a time.

Table 3-3 **• Quadrant Global Pin Name**

I/O Type	Beginning of I/O Name	Notes
Single-Ended	GAAO/IOuxwByVz GAA1/IOuxwByVz GAA2/IOuxwByVz	Only one of the I/Os can be directly connected to a quadrant global at a time
	GABO/IOuxwByVz GAB1/IOuxwByVz GAB2/IOuxwByVz	Only one of the I/Os can be directly connected to a quadrant global at a time.
	GAC0/IOuxwByVz GAC1/IOuxwByVz GAC2/IOuxwByVz	Only one of the I/Os can be directly connected to a quadrant global at a time.
	GBAO/IOuxwByVz GBA1/IOuxwByVz GBA2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GBBO/IOuxwByVz GBB1/IOuxwByVz GBB2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GBC0/IOuxwByVz GBC1/IOuxwByVz GBC2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GDAO/IOuxwByVz GDA1/IOuxwByVz GDA2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GDBO/IOuxwByVz GDB1/IOuxwByVz GDB2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GDC0/IOuxwByVz GDC1/IOuxwByVz GDC2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GEAO/IOuxwByVz GEA1/IOuxwByVz GEA2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GEBO/IOuxwByVz GEB1/IOuxwByVz GEB2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GEC0/IOuxwByVz GEC1/IOuxwByVz GEC2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.

Note: Only one of the I/Os can be directly connected to a quadrant at a time.

Table 3-3 • **Quadrant Global Pin Name (continued)**

Differential I/O Pairs	GAAO/IOuxwByVz GAA1/IOuxwByVz	The output of the different pair will drive the global.
	GABO/IOuxwByVz GAB1/IOuxwByVz	The output of the different pair will drive the global.
	GACO/IOuxwByVz GAC1/IOuxwByVz	The output of the different pair will drive the global.
	GBAO/IOuxwByVz GBA1/IOuxwByVz	The output of the different pair will drive the global.
	GBBO/IOuxwByVz GBB1/IOuxwByVz	The output of the different pair will drive the global.
	GBCO/IOuxwByVz GBC1/IOuxwByVz	The output of the different pair will drive the global.
	GDAO/IOuxwByVz GDA1/IOuxwByVz	The output of the different pair will drive the global.
	GDBO/IOuxwByVz GDB1/IOuxwByVz	The output of the different pair will drive the global.
	GDCO/IOuxwByVz GDC1/IOuxwByVz	The output of the different pair will drive the global.
	GEAO/IOuxwByVz GEA1/IOuxwByVz	The output of the different pair will drive the global.
	GEBO/IOuxwByVz GEB1/IOuxwByVz	The output of the different pair will drive the global.
	GECO/IOuxwByVz GEC1/IOuxwByVz	The output of the different pair will drive the global.

Note: Only one of the I/Os can be directly connected to a quadrant at a time.

Unused Global I/O Configuration

The unused clock inputs behave similarly to the unused Pro I/Os. The Microsemi Designer software automatically configures the unused global pins as inputs with pull-up resistors if they are not used as regular I/O.

I/O Banks and Global I/O Standards

In low power flash devices, any I/O or internal logic can be used to drive the global network. However, only the global macro placed at the global pins will use the hardwired connection between the I/O and global network. Global signal (signal driving a global macro) assignment to I/O banks is no different from regular I/O assignment to I/O banks with the exception that you are limited to the pin placement location available. Only global signals compatible with both the VCCI and VREF standards can be assigned to the same bank.

Spine Architecture

The low power flash device architecture allows the VersaNet global networks to be segmented. Each of these networks contains spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside its region. The nine spines available in a vertical column reside in global networks with two separate regions of scope: the quadrant global network, which has three spines, and the chip (main) global network, which has six spines. Note that the number of quadrant globals and globals/spines per tree varies depending on the specific device. Refer to Table 3-4 for the clocking resources available for each device. The spines are the vertical branches of the global network tree, shown in Figure 3-3 on page 36. Each spine in a vertical column of a chip (main) global network is further divided into two spine segments of equal lengths: one in the top and one in the bottom half of the die (except in 10 k through 30 k gate devices).

Top and bottom spine segments radiating from the center of a device have the same height. However, just as in the ProASIC$^{\underline{PLUS}®}$ family, signals assigned only to the top and bottom spine cannot access the middle two rows of the die. The spines for quadrant clock networks do not cross the middle of the die and cannot access the middle two rows of the architecture.

Each spine and its associated ribs cover a certain area of the device (the "scope" of the spine; see Figure 3-3 on page 36). Each spine is accessed by the dedicated global network MUX tree architecture, which defines how a particular spine is driven—either by the signal on the global network from a CCC, for example, or by another net defined by the user. Details of the chip (main) global network spine-selection MUX are presented in Figure 3-8 on page 46. The spine drivers for each spine are located in the middle of the die.

Quadrant spines can be driven from user I/Os or an internal signal from the north and south sides of the die. The ability to drive spines in the quadrant global networks can have a significant effect on system performance for high-fanout inputs to a design. Access to the top quadrant spine regions is from the top of the die, and access to the bottom quadrant spine regions is from the bottom of the die. The A3PE3000 device has 28 clock trees and each tree has nine spines; this flexible global network architecture enables users to map up to 252 different internal/external clocks in an A3PE3000 device.

***Table 3-4* • Globals/Spines/Rows for IGLOO and ProASIC3 Devices**

ProASIC3/ ProASIC3L Devices	IGLOO Devices	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
A3PN010	AGLN010	4	0	1	0	0	260	260	4
A3PN015	AGLN015	4	0	1	0	0	384	384	6
A3PN020	AGLN020	4	0	1	0	0	520	520	6
A3PN060	AGLN060	6	12	4	9	36	384	1,536	12
A3PN125	AGLN125	6	12	8	9	72	384	3,072	12
A3PN250	AGLN250	6	12	8	9	72	768	6,144	24
A3P015	AGL015	6	0	1	9	9	384	384	12
A3P030	AGL030	6	0	2	9	18	384	768	12
A3P060	AGL060	6	12	4	9	36	384	1,536	12
A3P125	AGL125	6	12	8	9	72	384	3,072	12
A3P250/L	AGL250	6	12	8	9	72	768	6,144	24
A3P400	AGL400	6	12	12	9	108	768	9,216	24
A3P600/L	AGL600	6	12	12	9	108	1,152	13,824	36
A3P1000/L	AGL1000	6	12	16	9	144	1,536	24,576	48
A3PE600/L	AGLE600	6	12	12	9	108	1,120	13,440	35
A3PE1500		6	12	20	9	180	1,888	37,760	59
A3PE3000/L	AGLE3000	6	12	28	9	252	2,656	74,368	83

Table 3-5 • **Globals/Spines/Rows for IGLOO PLUS Devices**

IGLOO PLUS Devices	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
AGLP030	6	0	2	9	18	384*	792	12
AGLP060	6	12	4	9	36	384*	1,584	12
AGLP125	6	12	8	9	72	384*	3,120	12

*Note: *Clock trees that are located at far left and far right will support more VersaTiles.*

Table 3-6 • **Globals/Spines/Rows for Fusion Devices**

Fusion Device	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
AFS090	6	12	6	9	54	384	2,304	12
AFS250	6	12	8	9	72	768	6,144	24
AFS600	6	12	12	9	108	1,152	13,824	36
AFS1500	6	12	20	9	180	1,920	38,400	60

Spine Access

The physical location of each spine is identified by the letter T (top) or B (bottom) and an accompanying number (T*n* or B*n*). The number *n* indicates the horizontal location of the spine; 1 refers to the first spine on the left side of the die. Since there are six chip spines in each spine tree, there are up to six spines available for each combination of T (or B) and *n* (for example, six T1 spines). Similarly, there are three quadrant spines available for each combination of T (or B) and *n* (for example, four T1 spines), as shown in Figure 3-7.

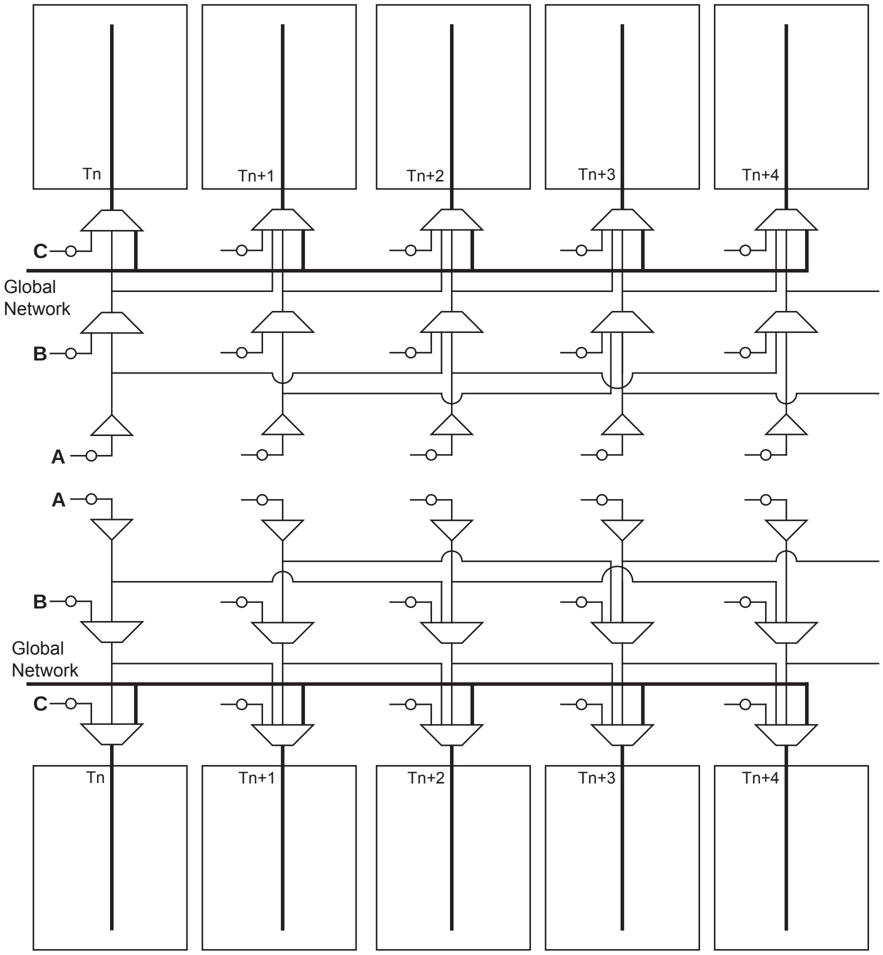


Figure 3-7 **• Chip Global Aggregation**

A spine is also called a local clock network, and is accessed by the dedicated global MUX architecture. These MUXes define how a particular spine is driven. Refer to Figure 3-8 on page 46 for the global MUX architecture. The MUXes for each chip global spine are located in the middle of the die. Access to the top and bottom chip global spine is available from the middle of the die. There is no control dependency between the top and bottom spines. If a top spine, T1, of a chip global network is assigned to a net, B1 is not wasted and can be used by the global clock network. The signal assigned only to the top or bottom spine cannot access the middle two rows of the architecture. However, if a spine is using the top and bottom at the same time (T1 and B1, for instance), the previous restriction is lifted.

The MUXes for each quadrant global spine are located in the north and south sides of the die. Access to the top and bottom quadrant global spines is available from the north and south sides of the die. Since the MUXes for quadrant spines are located in the north and south sides of the die, you should not try to drive T1 and B1 quadrant spines from the same signal.

Using Clock Aggregation

Clock aggregation allows for multi-spine clock domains to be assigned using hardwired connections, without adding any extra skew. A MUX tree, shown in Figure 3-8, provides the necessary flexibility to allow long lines, local resources, or I/Os to access domains of one, two, or four global spines. Signal access to the clock aggregation system is achieved through long-line resources in the central rib in the center of the die, and also through local resources in the north and south ribs, allowing I/Os to feed directly into the clock system. As Figure 3-9 indicates, this access system is contiguous.

There is no break in the middle of the chip for the north and south I/O VersaNet access. This is different from the quadrant clocks located in these ribs, which only reach the middle of the rib.

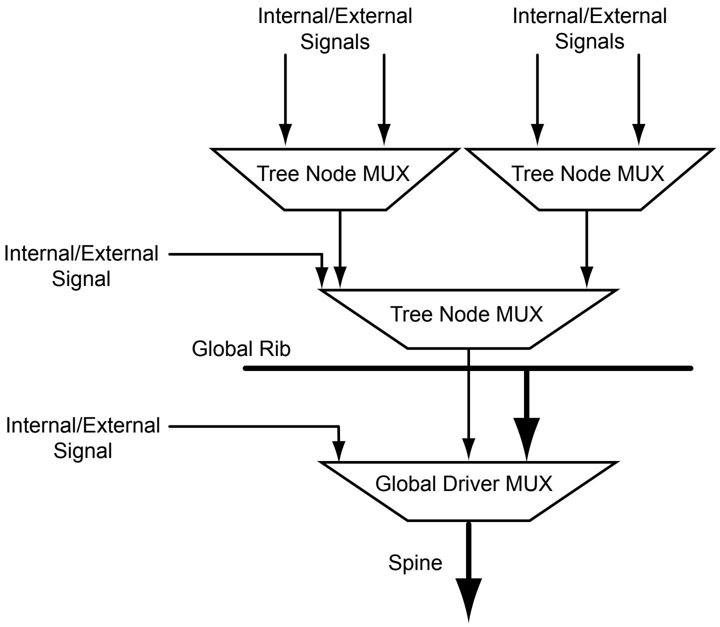


Figure 3-8 • **Spine Selection MUX of Global Tree**

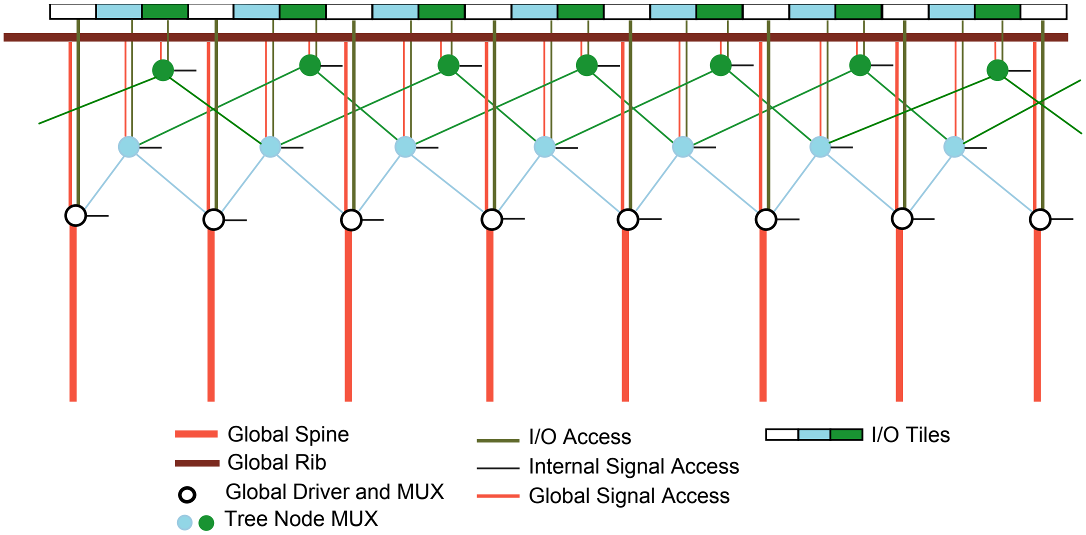


Figure 3-9 • **Clock Aggregation Tree Architecture**

Clock Aggregation Architecture

This clock aggregation feature allows a balanced clock tree, which improves clock skew. The physical regions for clock aggregation are defined from left to right and shift by one spine. For chip global networks, there are three types of clock aggregation available, as shown in Figure 3-10:

- Long lines that can drive up to four adjacent spines (A)
- Long lines that can drive up to two adjacent spines (B)
- Long lines that can drive one spine (C)

There are three types of clock aggregation available for the quadrant spines, as shown in Figure 3-10:

- I/Os or local resources that can drive up to four adjacent spines
- I/Os or local resources that can drive up to two adjacent spines
- I/Os or local resources that can drive one spine

As an example, A3PE600 and AFS600 devices have twelve spine locations: T1, T2, T3, T4, T5, T6, B1, B2, B3, B4, B5, and B6. Table 3-7 shows the clock aggregation you can have in A3PE600 and AFS600.

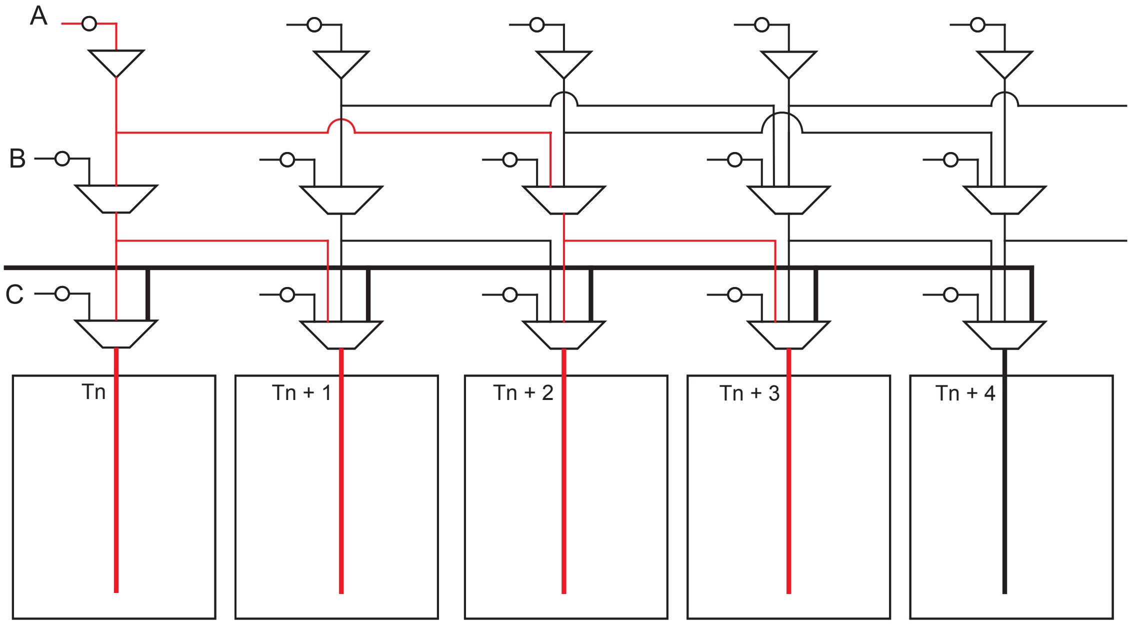


***Figure 3-10* • Four Spines Aggregation**

***Table 3-7* • Spine Aggregation in A3PE600 or AFS600**

Clock Aggregation	**Spine**
1 spine	T1, T2, T3, T4, T5, T6, B1, B2, B3, B4, B5, B6
2 spines	T1:T2, T2:T3, T3:T4, T4:T5, T5:T6, B1:B2, B2:B3, B3:B4, B4:B5, B5:B6
4 spines	B1:B4, B2:B5, B3:B6, T1:T4, T2:T5, T3:T6

The clock aggregation for the quadrant spines can cross over from the left to right quadrant, but not from top to bottom. The quadrant spine assignment T1:T4 is legal, but the quadrant spine assignment T1:B1 is not legal. Note that this clock aggregation is hardwired. You can always assign signals to spine T1 and B2 by instantiating a buffer, but this may add skew in the signal.

Design Recommendations

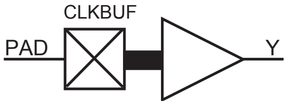
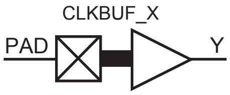
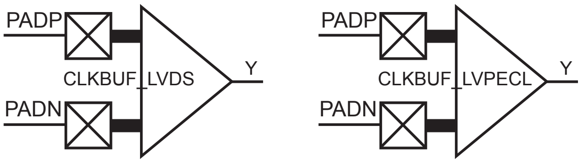
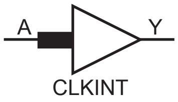
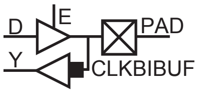
The following sections provide design flow recommendations for using a global network in a design.

- "Global Macros and I/O Standards"
- "Global Macro and Placement Selections" on page 50
- "Using Global Macros in Synplicity" on page 52
- "Global Promotion and Demotion Using PDC" on page 53
- "Spine Assignment" on page 54
- "Designer Flow for Global Assignment" on page 55
- "Simple Design Example" on page 57
- "Global Management in PLL Design" on page 59
- "Using Spines of Occupied Global Networks" on page 60

Global Macros and I/O Standards

The larger low power flash devices have six chip global networks and four quadrant global networks. However, the same clock macros are used for assigning signals to chip globals and quadrant globals. Depending on the clock macro placement or assignment in the Physical Design Constraint (PDC) file or MultiView Navigator (MVN), the signal will use the chip global network or quadrant network. Table 3-8 lists the clock macros available for low power flash devices. Refer to the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide* for details.

***Table 3-8* • Clock Macros**

Macro Name	Description	Symbol
CLKBUF	Input macro for Clock Network	
CLKBUF_x	Input macro for Clock Network with specific I/O standard	
CLKBUF_LVDS/LVPECL	LVDS or LVPECL input macro for Clock Network (not supported for IGLOO nano or ProASIC3 nano devices)	 PADP CLKBUF_LVDS PADN Y PADP CLKBUF_LVPECL PADN Y
CLKINT	Macro for internal clock interface	
CLKBIBUF	Bidirectional macro with input dedicated to routed Clock Network	

Use these available macros to assign a signal to the global network. In addition to these global macros, PLL and CLKDLY macros can also drive the global networks. Use I/O–standard–specific clock macros (CLKBUF_x) to instantiate a specific I/O standard for the global signals. Table 3-9 on page 49 shows the list of these I/O–standard–specific macros. Note that if you use these I/O–standard–specific clock macros, you cannot change the I/O standard later in the design stage. If you use the regular CLKBUF macro, you can use MVN or the PDC file in Designer to change the I/O standard. The default I/O standard for CLKBUF is LVTTL in the current Microsemi Libero® System-on-Chip (SoC) and Designer software.

Table 3-9 • **I/O Standards within CLKBUF**

Name	Description
CLKBUF_LVCMOS5	LVCMOS clock buffer with 5.0 V CMOS voltage level
CLKBUF_LVCMOS33	LVCMOS clock buffer with 3.3 V CMOS voltage level
CLKBUF_LVCMOS25	LVCMOS clock buffer with 2.5 V CMOS voltage level[1]
CLKBUF_LVCMOS18	LVCMOS clock buffer with 1.8 V CMOS voltage level
CLKBUF_LVCMOS15	LVCMOS clock buffer with 1.5 V CMOS voltage level
CLKBUF_LVCMOS12	LVCMOS clock buffer with 1.2 V CMOS voltage level
CLKBUF_PCI	PCI clock buffer
CLKBUF_PCIX	PCIX clock buffer
CLKBUF_GTL25	GTL clock buffer with 2.5 V CMOS voltage level[1]
CLKBUF_GTL33	GTL clock buffer with 3.3 V CMOS voltage level[1]
CLKBUF_GTLP25	GTL+ clock buffer with 2.5 V CMOS voltage level[1]
CLKBUF_GTLP33	GTL+ clock buffer with 3.3 V CMOS voltage level[1]
CLKBUF_ HSTL _I	HSTL Class I clock buffer[1]
CLKBUF_ HSTL _II	HSTL Class II clock buffer[1]
CLKBUF_SSTL2_I	SSTL2 Class I clock buffer[1]
CLKBUF_SSTL2_II	SSTL2 Class II clock buffer[1]
CLKBUF_SSTL3_I	SSTL3 Class I clock buffer[1]
CLKBUF_SSTL3_II	SSTL3 Class II clock buffer[1]

Notes:

1. *Supported in only the IGLOOe, ProASIC3E, AFS600, and AFS1500 devices*
2. *By default, the CLKBUF macro uses the 3.3 V LVTTL I/O technology.*

The current synthesis tool libraries only infer the CLKBUF or CLKINT macros in the netlist. All other global macros must be instantiated manually into your HDL code. The following is an example of CLKBUF_LVCMOS25 global macro instantiations that you can copy and paste into your code:

VHDL

```
component clkbuf_lvcmos25
  port (pad : in std_logic; y : out std_logic);
end component

begin
-- concurrent statements
u2 : clkbuf_lvcmos25 port map (pad =>  ext_clk, y => int_clk);
end
```

Verilog

```
module design (______);

input _____;
output ______;

clkbuf_lvcmos25 u2 (.y(int_clk), .pad(ext_clk);

endmodule
```

Global Macro and Placement Selections

Low power flash devices provide the flexibility of choosing one of the three global input pad locations available to connect to a global / quadrant global network. For 60K gate devices and above, if the single-ended I/O standard is chosen, there is flexibility to choose one of the global input pads (the first, second, and fourth input). Once chosen, the other I/O locations are used as regular I/Os. If the differential I/O standard is chosen, the first and second inputs are considered as paired, and the third input is paired with a regular I/O. The user then has the choice of selecting one of the two sets to be used as the global input source. There is also the option to allow an internal clock signal to feed the global network. A multiplexer tree selects the appropriate global input for routing to the desired location. Note that the global I/O pads do not need to feed the global network; they can also be used as regular I/O pads.

Hardwired I/O Clock Source

Hardwired I/O refers to global input pins that are hardwired to the multiplexer tree, which directly accesses the global network. These global input pins have designated pin locations and are indicated with the I/O naming convention Gmn (m refers to any one of the positions where the global buffers is available, and n refers to any one of the three global input MUXes and the pin number of the associated global location, m). Choosing this option provides the benefit of directly connecting to the global buffers, which provides less delay. See Figure 3-11 for an example illustration of the connections, shown in red. If a CLKBUF macro is initiated, the clock input can be placed at one of nine dedicated global input pin locations: GmA0, GmA1, GmA2, GmB0, GmB1, GmB2, GmC0, GmC1, or GmC2. Note that the placement of the global will determine whether you are using chip global or quadrant global. For example, if the CLKBIF is placed in one of the GF pin locations, it will use the chip global network; if the CLKBIF is placed in one of the GA pin locations, it will use quadrant global network. This is shown in Figure 3-12 on page 51 and Figure 3-13 on page 51.

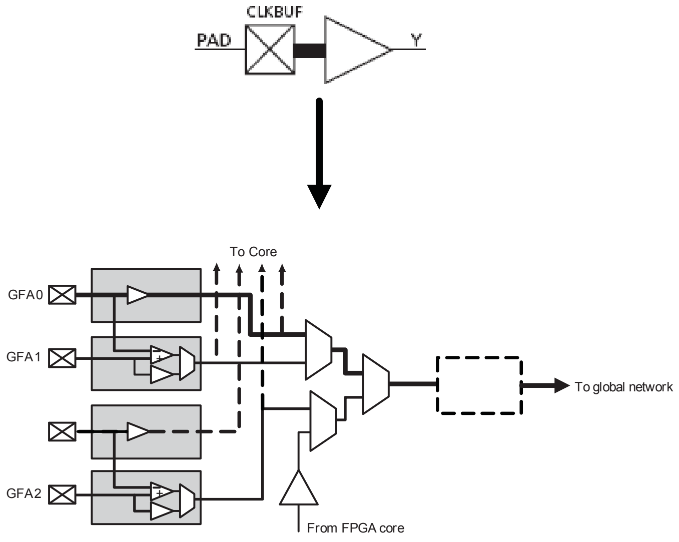


Figure 3-11 • **CLKBUF Macro**

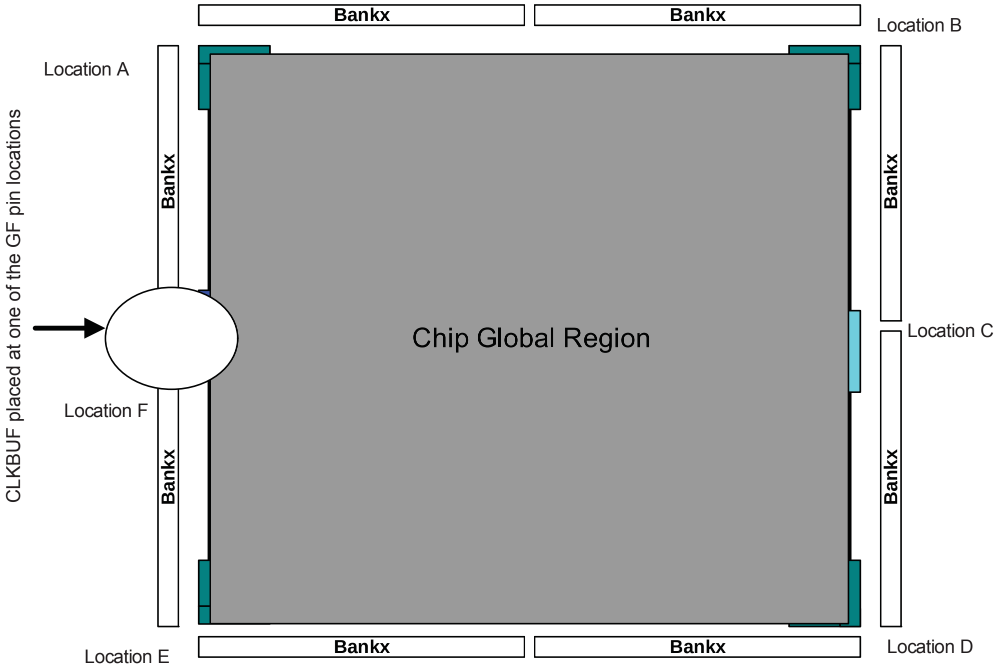


Figure 3-12 • **Chip Global Region**

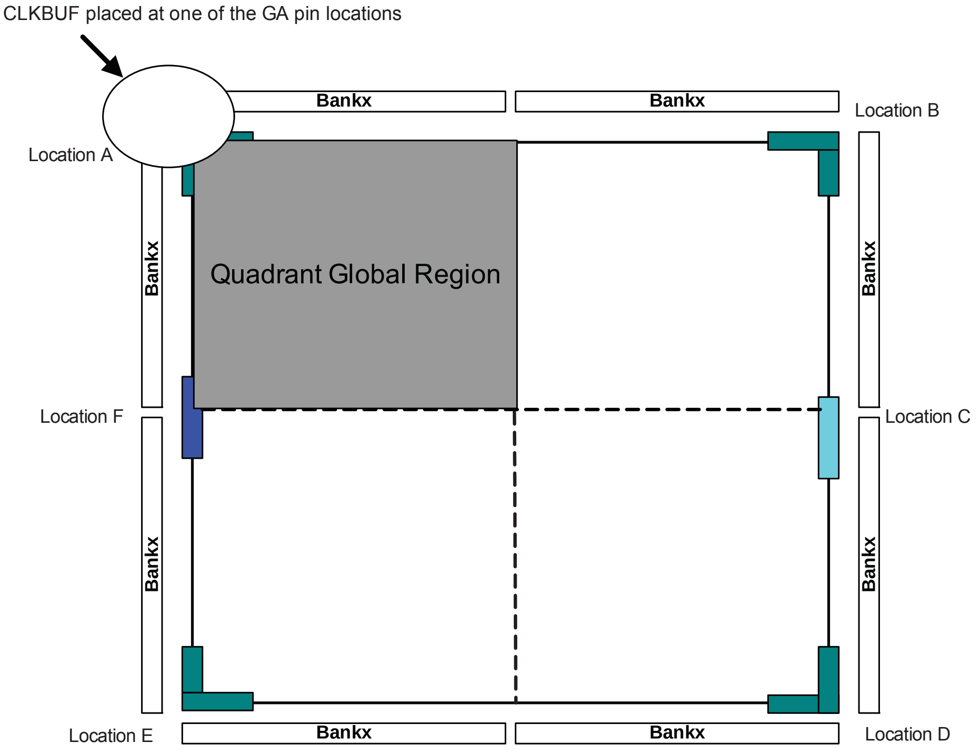


Figure 3-13 • **Quadrant Global Region**

External I/O or Local signal as Clock Source

External I/O refers to regular I/O pins are labeled with the I/O convention IOuxwByVz. You can allow the external I/O or internal signal to access the global. To allow the external I/O or internal signal to access the global network, you need to instantiate the CLKINT macro. Refer to Figure 3-4 on page 37 for an example illustration of the connections. Instead of using CLKINT, you can also use PDC to promote signals from external I/O or internal signal to the global network. However, it may cause layout issues because of synthesis logic replication. Refer to the "Global Promotion and Demotion Using PDC" section on page 53 for details.

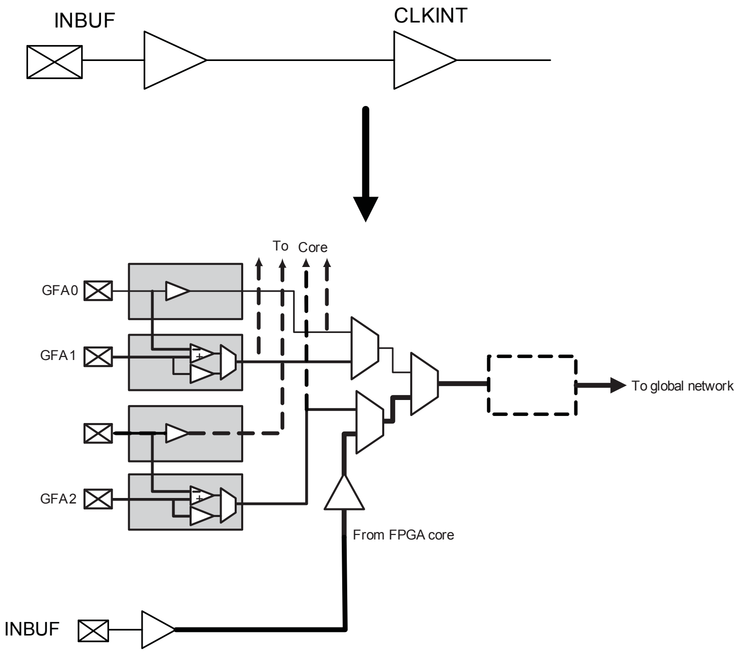
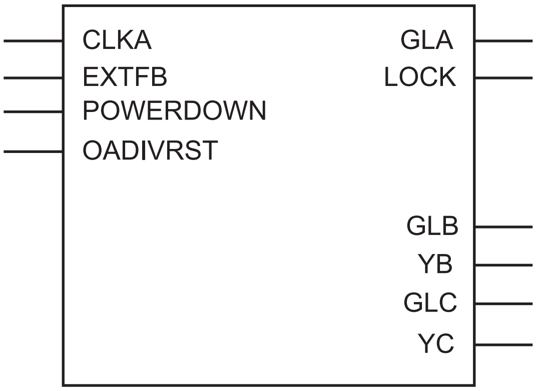


Figure 3-14 • **CLKINT Macro**

Using Global Macros in Synplicity

The Synplify® synthesis tool automatically inserts global buffers for nets with high fanout during synthesis. By default, Synplicity® puts six global macros (CLKBUF or CLKINT) in the netlist, including any global instantiation or PLL macro. Synplify always honors your global macro instantiation. If you have a PLL (only primary output is used) in the design, Synplify adds five more global buffers in the netlist. Synplify uses the following global counting rule to add global macros in the netlist:

1. CLKBUF: 1 global buffer
2. CLKINT: 1 global buffer
3. CLKDLY: 1 global buffer
4. PLL: 1 to 3 global buffers
   – GLA, GLB, GLC, YB, and YC are counted as 1 buffer.
   – GLB or YB is used or both are counted as 1 buffer.
   – GLC or YC is used or both are counted as 1 buffer.



Note: *OAVDIVRST exists only in the Fusion PLL.*

Figure 3-15 **• PLLs in Low Power Flash Devices**

You can use the syn_global_buffers attribute in Synplify to specify a maximum number of global macros to be inserted in the netlist. This can also be used to restrict the number of global buffers inserted. In the Synplicity 8.1 version or newer, a new attribute, syn_global_minfanout, has been added for low power flash devices. This enables you to promote only the high-fanout signal to global. However, be aware that you can only have six signals assigned to chip global networks, and the rest of the global signals should be assigned to quadrant global networks. So, if the netlist has 18 global macros, the remaining 12 global macros should have fanout that allows the instances driven by these globals to be placed inside a quadrant.

Global Promotion and Demotion Using PDC

The HDL source file or schematic is the preferred place for defining which signals should be assigned to a clock network using clock macro instantiation. This method is preferred because it is guaranteed to be honored by the synthesis tools and Designer software and stop any replication on this net by the synthesis tool. Note that a signal with fanout may have logic replication if it is not promoted to global during synthesis. In that case, the user cannot promote that signal to global using PDC. See Synplicity Help for details on using this attribute. To help you with global management, Designer allows you to promote a signal to a global network or demote a global macro to a regular macro from the user netlist using the compile options and/or PDC commands.

The following are the PDC constraints you can use to promote a signal to a global network:

1. PDC syntax to promote a regular net to a chip global clock:

   ```
   assign_global_clock –net netname
   ```

   The following will happen during promotion of a regular signal to a global network:

   – If the net is external, the net will be driven by a CLKINT inserted automatically by Compile.

   – The I/O macro will not be changed to CLKBUF macros.

   – If the net is an internal net, the net will be driven by a CLKINT inserted automatically by Compile.

2. PDC syntax to promote a net to a quadrant clock:

   ```
   assign_local_clock –net netname –type quadrant UR|UL|LR|LL
   ```

   This follows the same rule as the chip global clock network.

The following PDC command demotes the clock nets to regular nets.

```
unassign_global_clock -net netname
```

The following will happen during demotion of a global signal to regular nets:

- CLKBUF_x becomes INBUF_x; CLKINT is removed from the netlist.
- The essential global macro, such as the output of the Clock Conditioning Circuit, cannot be demoted.
- No automatic buffering will happen.

Since no automatic buffering happens when a signal is demoted, this net may have a high delay due to large fanout. This may have a negative effect on the quality of the results. Microsemi recommends that the automatic global demotion only be used on small-fanout nets. Use clock networks for high-fanout nets to improve timing and routability.

Spine Assignment

The low power flash device architecture allows the global networks to be segmented and used as clock spines. These spines, also called local clock networks, enable the use of PDC or MVN to assign a signal to a spine.

PDC syntax to promote a net to a spine/local clock:

```
assign_local_clock –net netname –type [quadrant|chip] Tn|Bn|Tn:Bm
```

If the net is driven by a clock macro, Designer automatically demotes the clock net to a regular net before it is assigned to a spine. Nets driven by a PLL or CLKDLY macro cannot be assigned to a local clock.

When assigning a signal to a spine or quadrant global network using PDC (pre-compile), the Designer software will legalize the shared instances. The number of shared instances to be legalized can be controlled by compile options. If these networks are created in MVN (only quadrant globals can be created), no legalization is done (as it is post-compile). Designer does not do legalization between non-clock nets.

As an example, consider two nets, net_clk and net_reset, driving the same flip-flop. The following PDC constraints are used:

```
assign_local_clock –net net_clk –type chip T3
assign_local_clock –net net_reset –type chip T1:T2
```

During Compile, Designer adds a buffer in the reset net and places it in the T1 or T2 region, and places the flip-flop in the T3 spine region (Figure 3-16).

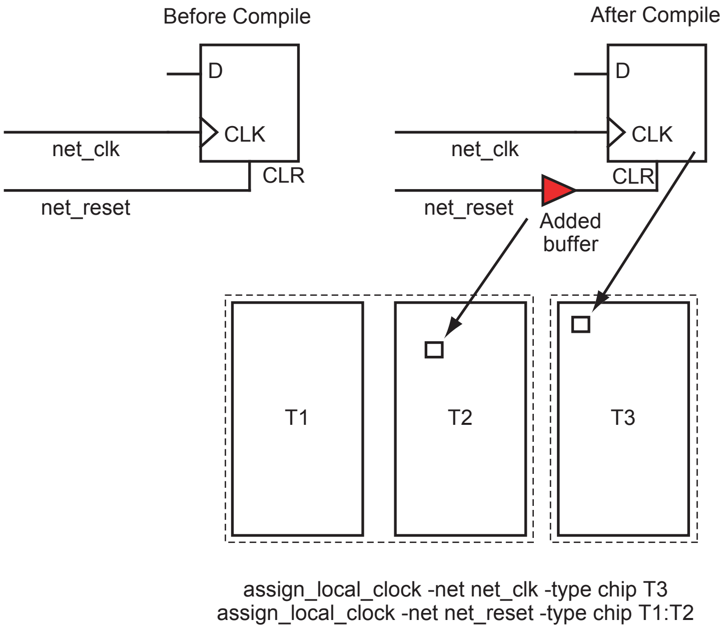
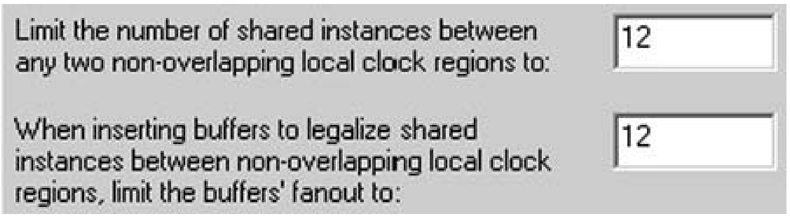

Before Compile
After Compile
D
CLK
CLR
net_clk
net_reset
D
CLK
CLR
net_clk
net_reset
Added
buffer
T1
T2
T3
assign_local_clock -net net_clk -type chip T3
assign_local_clock -net net_reset -type chip T1:T2

Figure 3-16 • **Adding a Buffer for Shared Instances**

You can control the maximum number of shared instances allowed for the legalization to take place using the Compile Option dialog box shown in Figure 3-17. Refer to Libero SoC / Designer online help for details on the Compile Option dialog box. A large number of shared instances most likely indicates a floorplanning problem that you should address.



Limit the number of shared instances between any two non-overlapping local clock regions to: 12

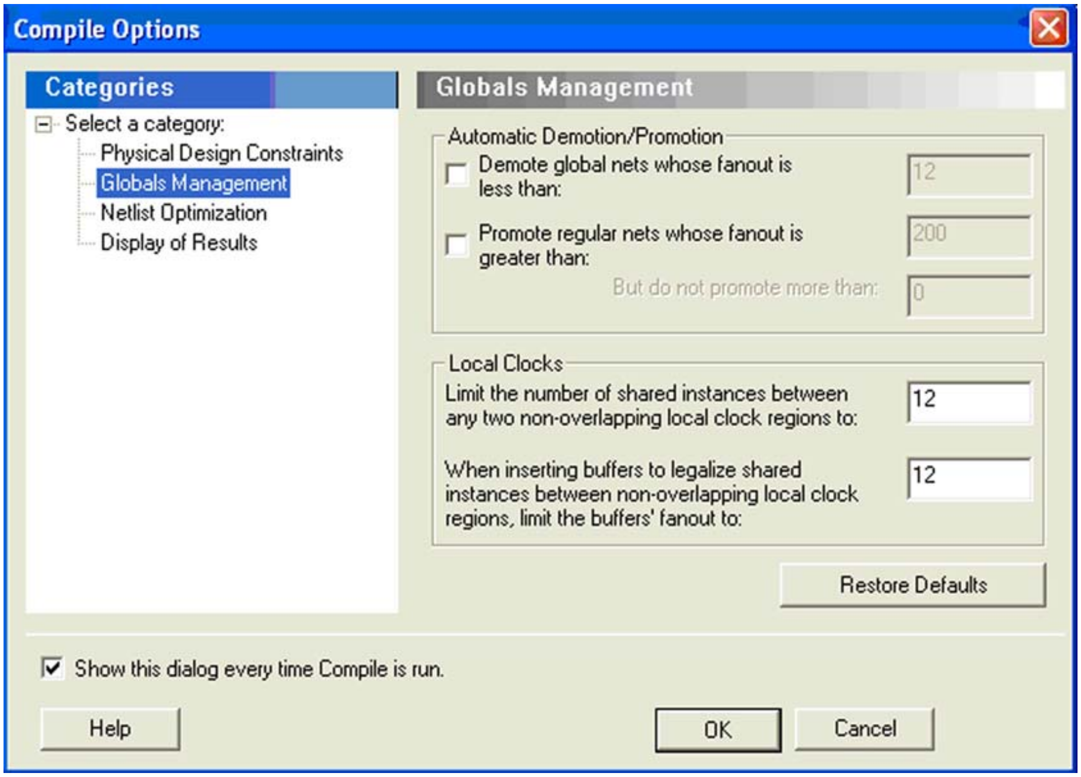
When inserting buffers to legalize shared instances between non-overlapping local clock regions, limit the buffers' fanout to: 12

***Figure 3-17* • Shared Instances in the Compile Option Dialog Box**

Designer Flow for Global Assignment

To achieve the desired result, pay special attention to global management during synthesis and place-and-route. The current Synplify tool does not insert more than six global buffers in the netlist by default. Thus, the default flow will not assign any signal to the quadrant global network. However, you can use attributes in Synplify and increase the default global macro assignment in the netlist. Designer v6.2 supports automatic quadrant global assignment, which was not available in Designer v6.1. Layout will make the choice to assign the correct signals to global. However, you can also utilize PDC and perform manual global assignment to overwrite any automatic assignment. The following step-by-step suggestions guide you in the layout of your design and help you improve timing in Designer:

1. Run Compile and check the Compile report. The Compile report has global information in the "Device Utilization" section that describes the number of chip and quadrant signals in the design. A "Net Report" section describes chip global nets, quadrant global nets, local clock nets, a list of nets listed by fanout, and net candidates for local clock assignment. Review this information. Note that YB or YC are counted as global only when they are used in isolation; if you use YB only and not GLB, this net is not shown in the global/quadrant nets report. Instead, it appears in the Global Utilization report.
2. If some signals have a very high fanout and are candidates for global promotion, promote those signals to global using the compile options or PDC commands. Figure 3-18 on page 56 shows the Globals Management section of the compile options. Select **Promote regular nets whose fanout is greater than** and enter a reasonable value for fanouts.



Compile Options

Categories

Select a category:
- Physical Design Constraints
- Globals Management
- Netlist Optimization
- Display of Results

Globals Management

Automatic Demotion/Promotion

□ Demote global nets whose fanout is less than: 12

□ Promote regular nets whose fanout is greater than: 200

But do not promote more than: 0

Local Clocks

Limit the number of shared instances between any two non-overlapping local clock regions to: 12

When inserting buffers to legalize shared instances between non-overlapping local clock regions, limit the buffers' fanout to: 12

☑ Show this dialog every time Compile is run.

Figure 3-18 **• Globals Management GUI in Designer**

3. Occasionally, the synthesis tool assigns a global macro to clock nets, even though the fanout is significantly less than other asynchronous signals. Select **Demote global nets whose fanout is less than** and enter a reasonable value for fanouts. This frees up some global networks from the signals that have very low fanouts. This can also be done using PDC.
4. Use a local clock network for the signals that do not need to go to the whole chip but should have low skew. This local clock network assignment can only be done using PDC.
5. Assign the I/O buffer using MVN if you have fixed I/O assignment. As shown in Figure 3-10 on page 47, there are three sets of global pins that have a hardwired connection to each global network. Do not try to put multiple CLKBUF macros in these three sets of global pins. For example, do not assign two CLKBUFs to GAA0x and GAA2x pins.
6. You must click **Commit** at the end of MVN assignment. This runs the pre-layout checker and checks the validity of global assignment.
7. Always run Compile with the **Keep existing physical constraints** option on. This uses the quadrant clock network assignment in the MVN assignment and checks if you have the desired signals on the global networks.
8. Run Layout and check the timing.

Simple Design Example

Consider a design consisting of six building blocks (shift registers) and targeted for an A3PE600-PQ208 (Figure 3-16 on page 54). The example design consists of two PLLs (PLL1 has GLA only; PLL2 has both GLA and GLB), a global reset (ACLR), an enable (EN_ALL), and three external clock domains (QCLK1, QCLK2, and QCLK3) driving the different blocks of the design. Note that the PQ208 package only has two PLLs (which access the chip global network). Because of fanout, the global reset and enable signals need to be assigned to the chip global resources. There is only one free chip global for the remaining global (QCLK1, QCLK2, QCLK3). Place two of these signals on the quadrant global resource. The design example demonstrates manually assignment of QCLK1 and QCLK2 to the quadrant global using the PDC command.

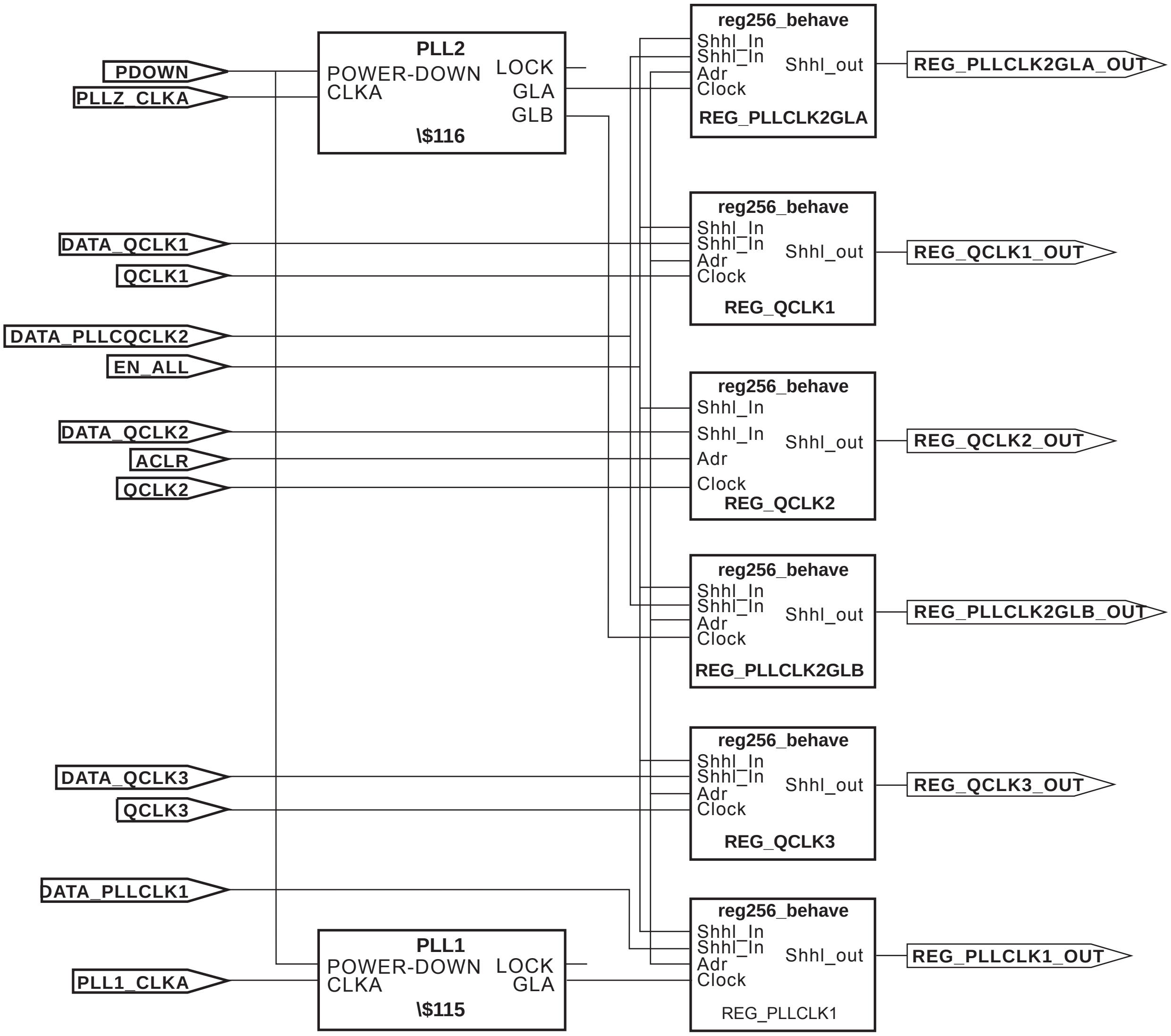


Figure 3-19 • **Block Diagram of the Global Management Example Design**

Step 1

Run Synthesis with default options. The Synplicity log shows the following device utilization:

Cell usage:

	cell count	area	count*area
DFN1E1C1	1536	2.0	3072.0
BUFF	278	1.0	278.0
INBUF	10	0.0	0.0
VCC	9	0.0	0.0
GND	9	0.0	0.0
OUTBUF	6	0.0	0.0
CLKBUF	3	0.0	0.0
PLL	2	0.0	0.0
TOTAL	1853		3350.0

Step 2

Run Compile with the **Promote regular nets whose fanout is greater than** option selected in Designer; you will see the following in the Compile report:

```
Device utilization report:
==========================
CORE                Used:   1536  Total:  13824   (11.11%)
IO (W/ clocks)      Used:     19  Total:    147   (12.93%)
Differential IO     Used:      0  Total:     65    (0.00%)
GLOBAL              Used:      8  Total:     18   (44.44%)
PLL                 Used:      2  Total:      2  (100.00%)
RAM/FIFO            Used:      0  Total:     24    (0.00%)
FlashROM            Used:      0  Total:      1    (0.00%)
……………………
The following nets have been assigned to a global resource:
Fanout  Type          Name
--------------------------
1536    INT_NET       Net   : EN_ALL_c
                      Driver: EN_ALL_pad_CLKINT
                      Source: AUTO PROMOTED
1536    SET/RESET_NET Net   : ACLR_c
                      Driver: ACLR_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : QCLK1_c
                      Driver: QCLK1_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : QCLK2_c
                      Driver: QCLK2_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : QCLK3_c
                      Driver: QCLK3_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : $1N14
                      Driver: $1I5/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N12
                      Driver: $1I6/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N10
                      Driver: $1I6/Core
                      Source: ESSENTIAL
```

Designer will promote five more signals to global due to high fanout. There are eight signals assigned to global networks.

During Layout, Designer will assign two of the signals to quadrant global locations.

Step 3 (optional)

You can also assign the QCLK1_c and QCLK2_c nets to quadrant regions using the following PDC commands:

```
assign_local_clock –net QCLK1_c  –type quadrant UL
assign_local_clock –net QCLK2_c  –type quadrant LL
```

Step 4

Import this PDC with the netlist and run Compile again. You will see the following in the Compile report:

```
The following nets have been assigned to a global resource:
Fanout  Type          Name
--------------------------
1536    INT_NET       Net   : EN_ALL_c
                      Driver: EN_ALL_pad_CLKINT
                      Source: AUTO PROMOTED
1536    SET/RESET_NET Net   : ACLR_c
                      Driver: ACLR_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : QCLK3_c
                      Driver: QCLK3_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : $1N14
                      Driver: $1I5/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N12
                      Driver: $1I6/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N10
                      Driver: $1I6/Core
                      Source: ESSENTIAL
The following nets have been assigned to a quadrant clock resource using PDC:
Fanout  Type          Name
--------------------------
256     CLK_NET       Net   : QCLK1_c
                      Driver: QCLK1_pad_CLKINT
                      Region: quadrant_UL
256     CLK_NET       Net   : QCLK2_c
                      Driver: QCLK2_pad_CLKINT
                      Region: quadrant_LL
```

Step 5

Run Layout.

Global Management in PLL Design

This section describes the legal global network connections to PLLs in the low power flash devices. For detailed information on using PLLs, refer to "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 63. Microsemi recommends that you use the dedicated global pins to directly drive the reference clock input of the associated PLL for reduced propagation delays and clock distortion. However, low power flash devices offer the flexibility to connect other signals to reference clock inputs. Each PLL is associated with three global networks (Figure 3-5 on page 38). There are some limitations, such as when trying to use the global and PLL at the same time:

- If you use a PLL with only primary output, you can still use the remaining two free global networks.
- If you use three globals associated with a PLL location, you cannot use the PLL on that location.
- If the YB or YC output is used standalone, it will occupy one global, even though this signal does not go to the global network.

Using Spines of Occupied Global Networks

When a signal is assigned to a global network, the flash switches are programmed to set the MUX select lines (explained in the "Clock Aggregation Architecture" section on page 47) to drive the spines of that network with the global net. However, if the global net is restricted from reaching into the scope of a spine, the MUX drivers of that spine are available for other high-fanout or critical signals (Figure 3-20).

For example, if you want to limit the CLK1_c signal to the left half of the chip and want to use the right side of the same global network for CLK2_c, you can add the following PDC commands:

```
define_region -name region1 -type inclusive 0 0 34 29
assign_net_macros region1 CLK1_c
assign_local_clock –net CLK2_c –type chip B2
```

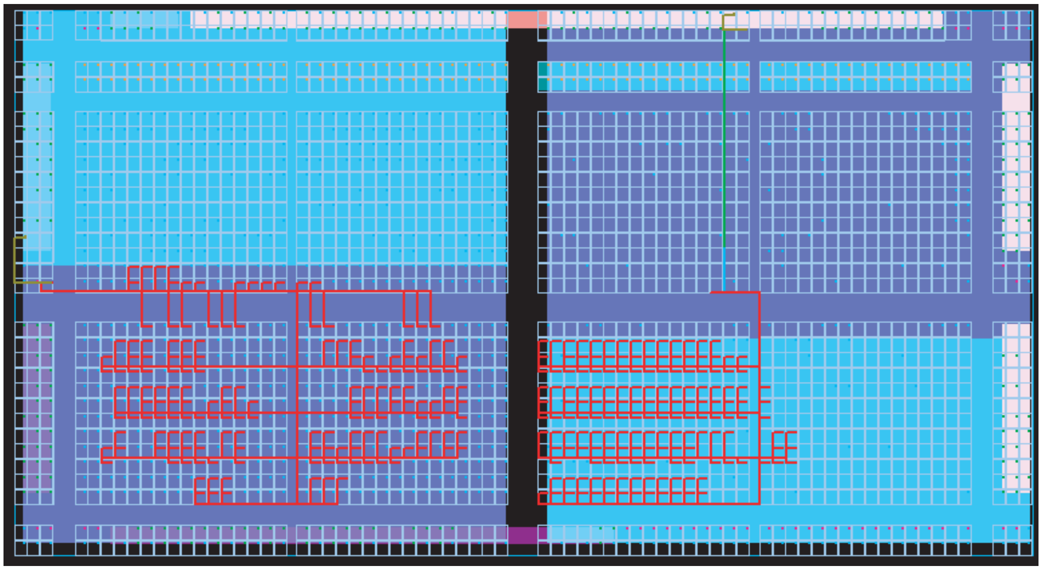


Figure 3-20 • **Design Example Using Spines of Occupied Global Networks**

Conclusion

IGLOO, Fusion, and ProASIC3 devices contain 18 global networks: 6 chip global networks and 12 quadrant global networks. These global networks can be segmented into local low-skew networks called spines. The spines provide low-skew networks for the high-fanout signals of a design. These allow you up to 252 different internal/external clocks in an A3PE3000 device. This document describes the architecture for the global network, plus guidelines and methodologies in assigning signals to globals and spines.

Related Documents

User’s Guides

IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide

http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
	The "Global Architecture" section and "VersaNet Global Network Distribution" section were revised for clarity (SARs 20646, 24779).	33, 35
	The "I/O Banks and Global I/Os" section was moved earlier in the document, renamed to "Chip and Quadrant Global I/Os", and revised for clarity. Figure 3-4 • Global Connections Details, Figure 3-6 • Global Inputs, Table 3-2 • Chip Global Pin Name, and Table 3-3 • Quadrant Global Pin Name are new (SARs 20646, 24779).	37
	The "Clock Aggregation Architecture" section was revised (SARs 20646, 24779).	43
	Figure 3-7 • Chip Global Aggregation was revised (SARs 20646, 24779).	45
	The "Global Macro and Placement Selections" section is new (SARs 20646, 24779).	50
v1.4 (December 2008)	The "Global Architecture" section was updated to include 10 k devices, and to include information about VersaNet global support for IGLOO nano devices.	33
	The Table 3-1 • Flash-Based FPGAs was updated to include IGLOO nano and ProASIC3 nano devices.	34
	The "VersaNet Global Network Distribution" section was updated to include 10 k devices and to note an exception in global lines for nano devices.	35
	Figure 3-2 • Simplified VersaNet Global Network (30 k gates and below) is new.	36
	The "Spine Architecture" section was updated to clarify support for 10 k and nano devices.	43
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include IGLOO nano and ProASIC3 nano devices.	43
	The figure in the CLKBUF_LVDS/LVPECL row of Table 3-8 • Clock Macros was updated to change CLKBIBUF to CLKBUF.	48
v1.3 (October 2008)	A third bullet was added to the beginning of the "Global Architecture" section: In Fusion devices, the west CCC also contains a PLL core. In the two larger devices (AFS600 and AFS1500), the west and east CCCs each contain a PLL.	33
	The "Global Resource Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	34
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include A3PE600/L in the device column.	43
	Table note 1 was revised in Table 3-9 • I/O Standards within CLKBUF to include AFS600 and AFS1500.	49
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 3-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	34

Date	Changes	Page
v1.1 (March 2008)	The "Global Architecture" section was updated to include the IGLOO PLUS family. The bullet was revised to include that the west CCC does not contain a PLL core in 15 k and 30 k devices. Instances of "A3P030 and AGL030 devices" were replaced with "15 k and 30 k gate devices."	33
v1.1 (continued)	Table 3-1 • Flash-Based FPGAs and the accompanying text was updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	34
	The "VersaNet Global Network Distribution" section, "Spine Architecture" section, the note in Figure 3-1 • Overview of VersaNet Global Network and Device Architecture, and the note in Figure 3-3 • Simplified VersaNet Global Network (60 k gates and above) were updated to include mention of 15 k gate devices.	35, 36
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to add the A3P015 device, and to revise the values for clock trees, globals/spines per tree, and globals/spines per device for the A3P030 and AGL030 devices.	43
	Table 3-5 • Globals/Spines/Rows for IGLOO PLUS Devices is new.	44
	CLKBUF_LVCMOS12 was added to Table 3-9 • I/O Standards within CLKBUF.	49
	The "User's Guides" section was updated to include the three different I/O Structures chapters for ProASIC3 and IGLOO device families.	60
v1.0 (January 2008)	Figure 3-3 • Simplified VersaNet Global Network (60 k gates and above) was updated.	36
	The "Naming of Global I/Os" section was updated.	37
	The "Using Global Macros in Synplicity" section was updated.	52
	The "Global Promotion and Demotion Using PDC" section was updated.	53
	The "Designer Flow for Global Assignment" section was updated.	55
	The "Simple Design Example" section was updated.	57
51900087-0/1.05 (January 2005)	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated.	43

4 – Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs

Introduction

This document outlines the following device information: Clock Conditioning Circuit (CCC) features, PLL core specifications, functional descriptions, software configuration information, detailed usage information, recommended board-level considerations, and other considerations concerning clock conditioning circuits and global networks in low power flash devices or mixed signal FPGAs.

Overview of Clock Conditioning Circuitry

In Fusion, IGLOO, and ProASIC3 devices, the CCCs are used to implement frequency division, frequency multiplication, phase shifting, and delay operations. The CCCs are available in six chip locations—each of the four chip corners and the middle of the east and west chip sides. For device-specific variations, refer to the "Device-Specific Layout" section on page 80.

The CCC is composed of the following:

- PLL core
- 3 phase selectors
- 6 programmable delays and 1 fixed delay that advances/delays phase
- 5 programmable frequency dividers that provide frequency multiplication/division (not shown in Figure 4-6 on page 73 because they are automatically configured based on the user's required frequencies)
- 1 dynamic shift register that provides CCC dynamic reconfiguration capability

Figure 4-1 provides a simplified block diagram of the physical implementation of the building blocks in each of the CCCs.

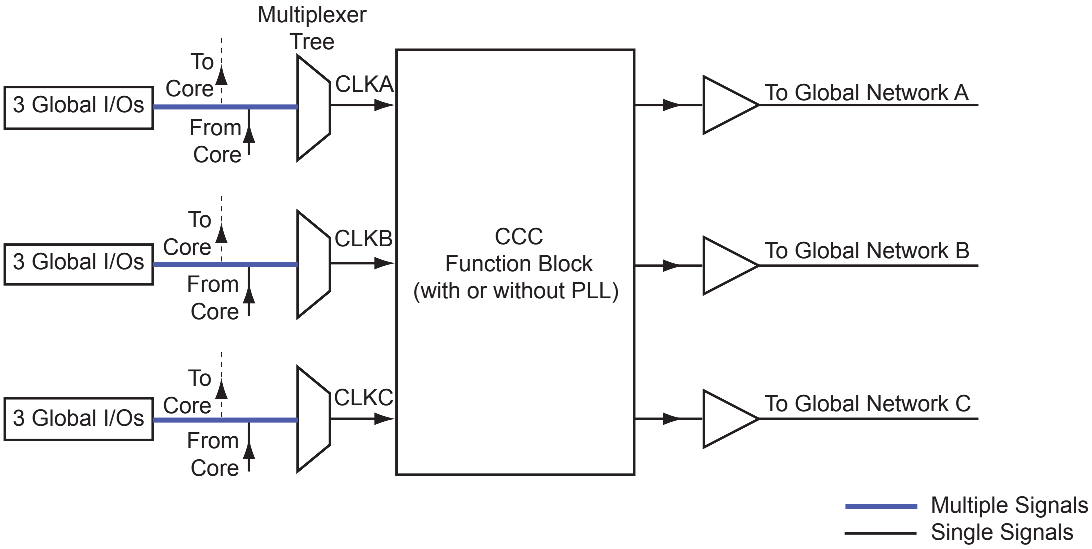


***Figure 4-1* • Overview of the CCCs Offered in Fusion, IGLOO, and ProASIC3**

Each CCC can implement up to three independent global buffers (with or without programmable delay) or a PLL function (programmable frequency division/multiplication, phase shift, and delays) with up to three global outputs. Unused global outputs of a PLL can be used to implement independent global buffers, up to a maximum of three global outputs for a given CCC.

CCC Programming

The CCC block is fully configurable, either via flash configuration bits set in the programming bitstream or through an asynchronous interface. This asynchronous dedicated shift register interface is dynamically accessible from inside the low power flash devices to permit parameter changes, such as PLL divide ratios and delays, during device operation.

To increase the versatility and flexibility of the clock conditioning system, the CCC configuration is determined either by the user during the design process, with configuration data being stored in flash memory as part of the device programming procedure, or by writing data into a dedicated shift register during normal device operation.

This latter mode allows the user to dynamically reconfigure the CCC without the need for core programming. The shift register is accessed through a simple serial interface. Refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 377 or the application note *Using Global Resources in Actel Fusion Devices*.

Global Resources

Low power flash and mixed signal devices provide three global routing networks (GLA, GLB, and GLC) for each of the CCC locations. There are potentially many I/O locations; each global I/O location can be chosen from only one of three possibilities. This is controlled by the multiplexer tree circuitry in each global network. Once the I/O location is selected, the user has the option to utilize the CCCs before the signals are connected to the global networks. The CCC in each location (up to six) has the same structure, so generating the CCC macros is always done with an identical software GUI. The CCCs in the corner locations drive the quadrant global networks, and the CCCs in the middle of the east and west chip sides drive the chip global networks. The quadrant global networks span only a quarter of the device, while the chip global networks span the entire device. For more details on global resources offered in low power flash devices, refer to the "Global Resources in Low Power Flash Devices" section on page 33.

A global buffer can be placed in any of the three global locations (CLKA-GLA, CLKB-GLB, or CLKC-GLC) of a given CCC. A PLL macro uses the CLKA CCC input to drive its reference clock. It uses the GLA and, optionally, the GLB and GLC global outputs to drive the global networks. A PLL macro can also drive the YB and YC regular core outputs. The GLB (or GLC) global output cannot be reused if the YB (or YC) output is used. Refer to the "PLL Macro Signal Descriptions" section on page 70 for more information.

Each global buffer, as well as the PLL reference clock, can be driven from one of the following:

- 3 dedicated single-ended I/Os using a hardwired connection
- 2 dedicated differential I/Os using a hardwired connection (not supported for IGLOO nano or ProASIC3 nano devices)
- The FPGA core

CCC Support in Microsemi's Flash Devices

The flash FPGAs listed in Table 4-1 support the CCC feature and the functions described in this document.

Table 4-1 **• Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
	IGLOO nano	The industry's lowest-power, smallest-size solution
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

*Note: *The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 4-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 4-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Global Buffers with No Programmable Delays

Access to the global / quadrant global networks can be configured directly from the global I/O buffer, bypassing the CCC functional block (as indicated by the dotted lines in Figure 4-1 on page 63). Internal signals driven by the FPGA core can use the global / quadrant global networks by connecting via the routed clock input of the multiplexer tree.

There are many specific CLKBUF macros supporting the wide variety of single-ended I/O inputs (CLKBUF) and differential I/O standards (CLKBUF_LVDS/LVPECL) in the low power flash families. They are used when connecting global I/Os directly to the global/quadrant networks.

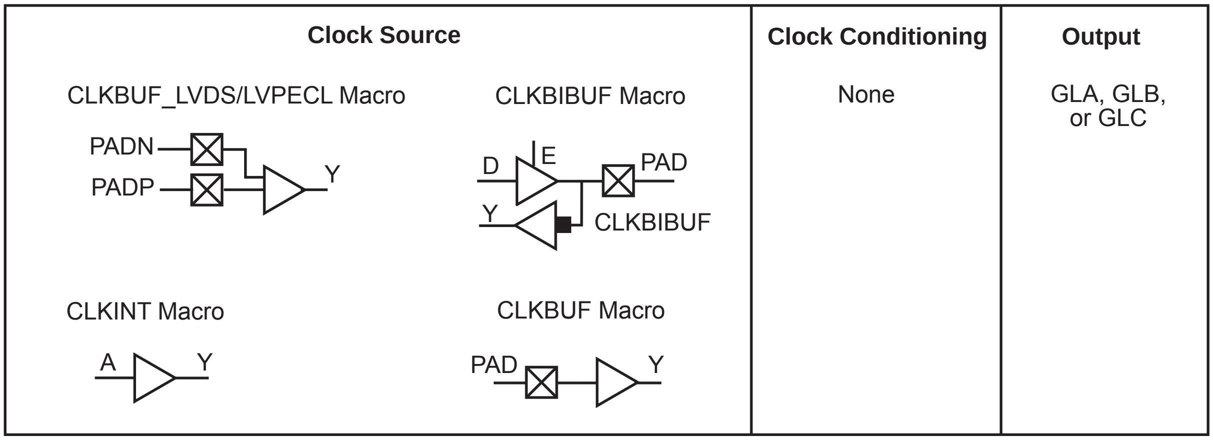
Note: IGLOO nano and ProASIC nano devices do not support differential inputs.

When an internal signal needs to be connected to the global/quadrant network, the CLKINT macro is used to connect the signal to the routed clock input of the network's MUX tree.

To utilize direct connection from global I/Os or from internal signals to the global/quadrant networks, CLKBUF, CLKBUF_LVPECL/LVDS, and CLKINT macros are used (Figure 4-2).

- The CLKBUF and CLKBUF_LVPECL/LVDS[1] macros are composite macros that include an I/O macro driving a global buffer, which uses a hardwired connection.
- The CLKBUF, CLKBUF_LVPECL/LVDS[1] and CLKINT macros are pass-through clock sources and do not use the PLL or provide any programmable delay functionality.
- The CLKINT macro provides a global buffer function driven internally by the FPGA core.

The available CLKBUF macros are described in the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide*.



Note: IGLOO nano and ProASIC nano devices do not support differential inputs.

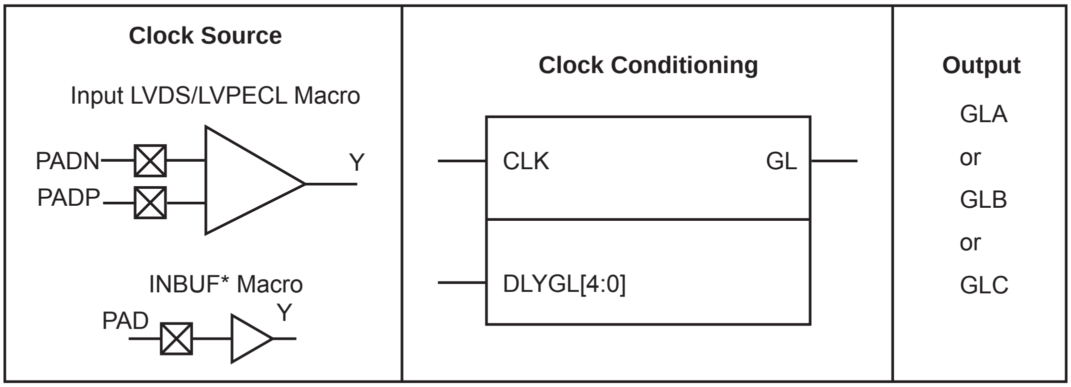
***Figure 4-2* • CCC Options: Global Buffers with No Programmable Delay**

Global Buffer with Programmable Delay

Clocks requiring clock adjustments can utilize the programmable delay cores before connecting to the global / quadrant global networks. A maximum of 18 CCC global buffers can be instantiated in a device—three per CCC and up to six CCCs per device.

Each CCC functional block contains a programmable delay element for each of the global networks (up to three), and users can utilize these features by using the corresponding macro (Figure 4-3 on page 67).

[1] *B-LVDS and M-LVDS are supported with the LVDS macro.*



Notes:

1. *For INBUF* driving a PLL macro or CLKDLY macro, the I/O will be hard-routed to the CCC; i.e., will be placed by software to a dedicated Global I/O.*
2. *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

***Figure 4-3* • CCC Options: Global Buffers with Programmable Delay**

The CLKDLY macro is a pass-through clock source that does not use the PLL, but provides the ability to delay the clock input using a programmable delay. The CLKDLY macro takes the selected clock input and adds a user-defined delay element. This macro generates an output clock phase shift from the input clock.

The CLKDLY macro can be driven by an INBUF* macro to create a composite macro, where the I/O macro drives the global buffer (with programmable delay) using a hardwired connection. In this case, the software will automatically place the dedicated global I/O in the appropriate locations. Many specific INBUF macros support the wide variety of single-ended and differential I/O standards supported by the low power flash family. The available INBUF macros are described in the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide.*

The CLKDLY macro can be driven directly from the FPGA core. The CLKDLY macro can also be driven from an I/O that is routed through the FPGA regular routing fabric. In this case, users must instantiate a special macro, PLLINT, to differentiate the clock input driven by the hardwired I/O connection.

The visual CLKDLY configuration in the SmartGen area of the Microsemi Libero System-on-Chip (SoC) and Designer tools allows the user to select the desired amount of delay and configures the delay elements appropriately. SmartGen also allows the user to select the input clock source. SmartGen will automatically instantiate the special macro, PLLINT, when needed.

CLKDLY Macro Signal Descriptions

The CLKDLY macro supports one input and one output. Each signal is described in Table 4-2.

***Table 4-2* • Input and Output Description of the CLKDLY Macro**

Signal	Name	I/O	Description
CLK	Reference Clock	Input	Reference clock input
GL	Global Output	Output	Primary output clock to respective global/quadrant clock networks

CLKDLY Macro Usage

When a CLKDLY macro is used in a CCC location, the programmable delay element is used to allow the clock delays to go to the global network. In addition, the user can bypass the PLL in a CCC location integrated with a PLL, but use the programmable delay that is associated with the global network by instantiating the CLKDLY macro. The same is true when using programmable delay elements in a CCC location with no PLLs (the user needs to instantiate the CLKDLY macro). There is no difference between the programmable delay elements used for the PLL and the CLKDLY macro. The CCC will be configured to use the programmable delay elements in accordance with the macro instantiated by the user.

As an example, if the PLL is not used in a particular CCC location, the designer is free to specify up to three CLKDLY macros in the CCC, each of which can have its own input frequency and delay adjustment options. If the PLL core is used, assuming output to only one global clock network, the other two global clock networks are free to be used by either connecting directly from the global inputs or connecting from one or two CLKDLY macros for programmable delay.

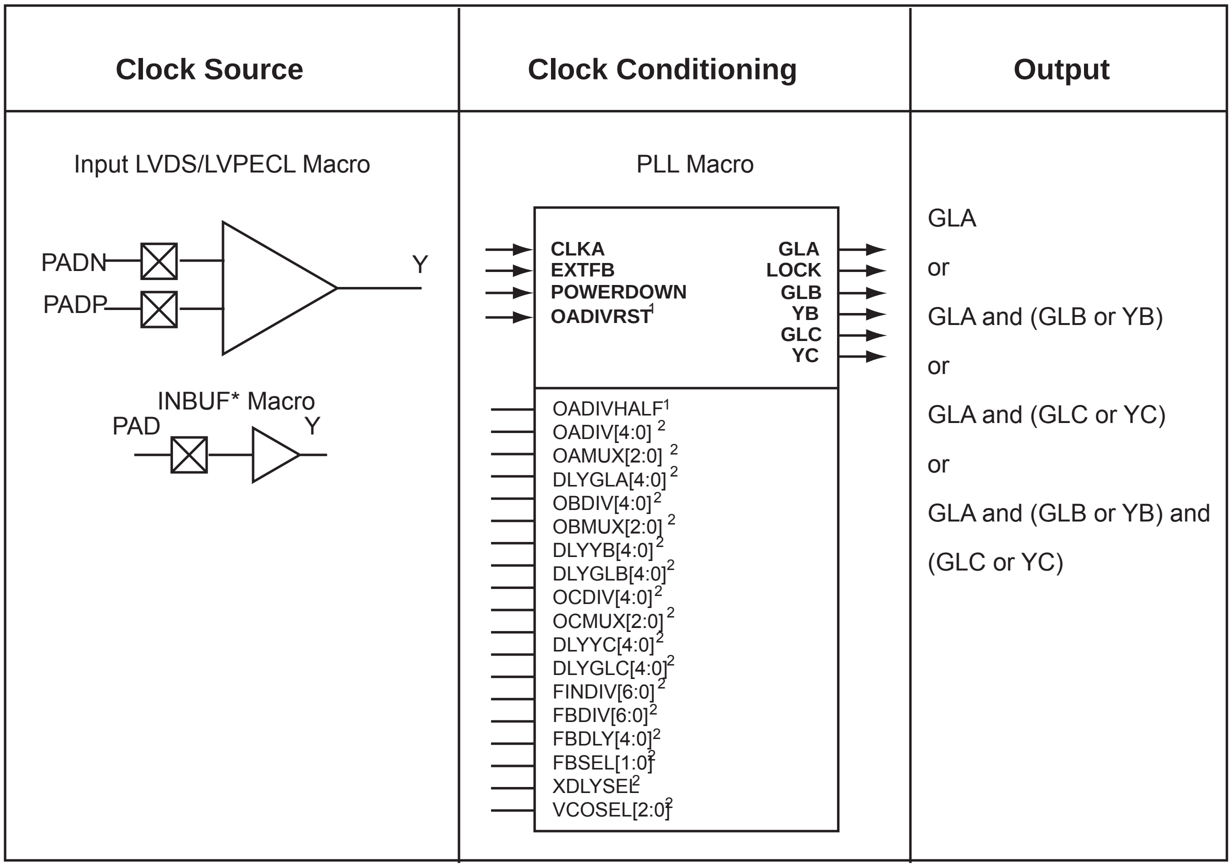
The programmable delay elements are shown in the block diagram of the PLL block shown in Figure 4-6 on page 73. Note that any CCC locations with no PLL present contain only the programmable delay blocks going to the global networks (labeled "Programmable Delay Type 2"). Refer to the "Clock Delay Adjustment" section on page 88 for a description of the programmable delay types used for the PLL. Also refer to Table 4-14 on page 96 for Programmable Delay Type 1 step delay values, and Table 4-15 on page 96 for Programmable Delay Type 2 step delay values. CCC locations with a PLL present can be configured to utilize only the programmable delay blocks (Programmable Delay Type 2) going to the global networks A, B, and C.

Global network A can be configured to use only the programmable delay element (bypassing the PLL) if the PLL is not used in the design. Figure 4-6 on page 73 shows a block diagram of the PLL, where the programmable delay elements are used for the global networks (Programmable Delay Type 2).

Global Buffers with PLL Function

Clocks requiring frequency synthesis or clock adjustments can utilize the PLL core before connecting to the global / quadrant global networks. A maximum of 18 CCC global buffers can be instantiated in a device—three per CCC and up to six CCCs per device. Each PLL core can generate up to three global/quadrant clocks, while a clock delay element provides one.

The PLL functionality of the clock conditioning block is supported by the PLL macro.



Notes:

1. *For Fusion only.*
2. *Refer to the* IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide *for more information.*
3. *For INBUF* driving a PLL macro or CLKDLY macro, the I/O will be hard-routed to the CCC; i.e., will be placed by software to a dedicated Global I/O.*
4. *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

***Figure 4-4* • CCC Options: Global Buffers with PLL**

The PLL macro provides five derived clocks (three independent) from a single reference clock. The PLL macro also provides power-down input and lock output signals. The additional inputs shown on the macro are configuration settings, which are configured through the use of SmartGen. For manual setting of these bits refer to the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide* for details.

Figure 4-6 on page 73 illustrates the various clock output options and delay elements.

PLL Macro Signal Descriptions

The PLL macro supports two inputs and up to six outputs. Table 4-3 gives a description of each signal.

Table 4-3 • **Input and Output Signals of the PLL Block**

Signal	Name	I/O	Description
CLKA	Reference Clock	Input	Reference clock input for PLL core; input clock for primary output clock, GLA
OADIVRST	Reset Signal for the Output Divider A	Input	For Fusion only. OADIVRST can be used when you bypass the PLL core (i.e., OAMUX = 001). The purpose of the OADIVRST signals is to reset the output of the final clock divider to synchronize it with the input to that divider when the PLL is bypassed. The signal is active on a low to high transition. The signal must be low for at least one divider input. If PLL core is used, this signal is "don't care" and the internal circuitry will generate the reset signal for the synchronization purpose.
OADIVHALF	Output A Division by Half	Input	For Fusion only. Active high. Division by half feature. This feature can only be used when users bypass the PLL core (i.e., OAMUX = 001) and the RC Oscillator (RCOSC) drives the CLKA input. This can be used to divide the 100 MHz RC oscillator by a factor of 1.5, 2.5, 3.5, 4.5 ... 14.5). Refer to Table 4-18 on page 97 for more information.
EXTFB	External Feedback	Input	Allows an external signal to be compared to a reference clock in the PLL core's phase detector.
POWERDOWN	Power Down	Input	Active low input that selects power-down mode and disables the PLL. With the POWERDOWN signal asserted, the PLL core sends 0 V signals on all of the outputs.
GLA	Primary Output	Output	Primary output clock to respective global/quadrant clock networks
GLB	Secondary 1 Output	Output	Secondary 1 output clock to respective global/quadrant clock networks
YB	Core 1 Output	Output	Core 1 output clock to local routing network
GLC	Secondary 2 Output	Output	Secondary 2 output clock to respective global/quadrant clock networks
YC	Core 2 Output	Output	Core 2 output clock to local routing network
LOCK	PLL Lock Indicator	Output	Active high signal indicating that steady-state lock has been achieved between CLKA and the PLL feedback signal

Input Clock

The inputs to the input reference clock (CLKA) of the PLL can come from global input pins, regular I/O pins, or internally from the core. For Fusion families, the input reference clock can also be from the embedded RC oscillator or crystal oscillator.

Global Output Clocks

GLA (Primary), GLB (Secondary 1), and GLC (Secondary 2) are the outputs of Global Multiplexer 1, Global Multiplexer 2, and Global Multiplexer 3, respectively. These signals (GLx) can be used to drive the high-speed global and quadrant networks of the low power flash devices.

A global multiplexer block consists of the input routing for selecting the input signal for the GLx clock and the output multiplexer, as well as delay elements associated with that clock.

Core Output Clocks

YB and YC are known as Core Outputs and can be used to drive internal logic without using global network resources. This is especially helpful when global network resources must be conserved and utilized for other timing-critical paths.

YB and YC are identical to GLB and GLC, respectively, with the exception of a higher selectable final output delay. The SmartGen PLL Wizard will configure these outputs according to user specifications and can enable these signals with or without the enabling of Global Output Clocks.

The above signals can be enabled in the following output groupings in both internal and external feedback configurations of the static PLL:

- One output – GLA only
- Two outputs – GLA + (GLB and/or YB)
- Three outputs – GLA + (GLB and/or YB) + (GLC and/or YC)

PLL Macro Block Diagram

As illustrated, the PLL supports three distinct output frequencies from a given input clock. Two of these (GLB and GLC) can be routed to the B and C global network access, respectively, and/or routed to the device core (YB and YC).

There are five delay elements to support phase control on all five outputs (GLA, GLB, GLC, YB, and YC).

There are delay elements in the feedback loop that can be used to advance the clock relative to the reference clock.

The PLL macro reference clock can be driven in the following ways:

1. By an INBUF* macro to create a composite macro, where the I/O macro drives the global buffer (with programmable delay) using a hardwired connection. In this case, the I/O must be placed in one of the dedicated global I/O locations.
2. Directly from the FPGA core.
3. From an I/O that is routed through the FPGA regular routing fabric. In this case, users must instantiate a special macro, PLLINT, to differentiate from the hardwired I/O connection described earlier.

During power-up, the PLL outputs will toggle around the maximum frequency of the voltage-controlled oscillator (VCO) gear selected. Toggle frequencies can range from 40 MHz to 250 MHz. This will continue as long as the clock input (CLKA) is constant (HIGH or LOW). This can be prevented by LOW assertion of the POWERDOWN signal.

The visual PLL configuration in SmartGen, a component of the Libero SoC and Designer tools, will derive the necessary internal divider ratios based on the input frequency and desired output frequencies selected by the user.

Implementing EXTFB in ProASIC3/E Devices

When the external feedback (EXTFB) signal of the PLL in the ProASIC3/E devices is implemented, the phase detector of the PLL core receives the reference clock (CLKA) and EXTFB as inputs. EXTFB must be sourced as an INBUF macro and located at the global/chip clock location associated with the target PLL by Designer software. EXTFB cannot be sourced from the FPGA fabric.

The following example shows CLKA and EXTFB signals assigned to two global I/Os in the same global area of ProASIC3E device.

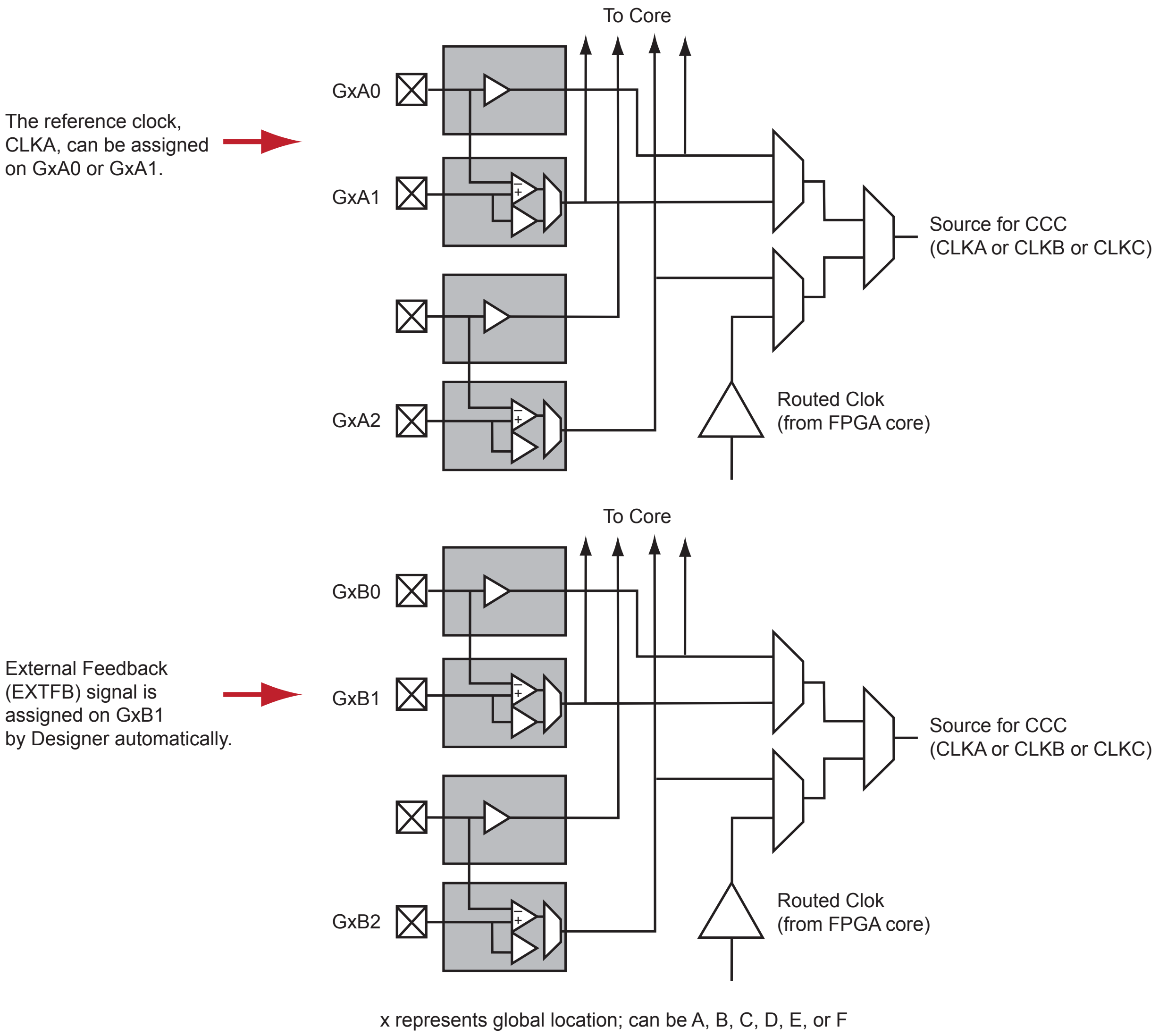
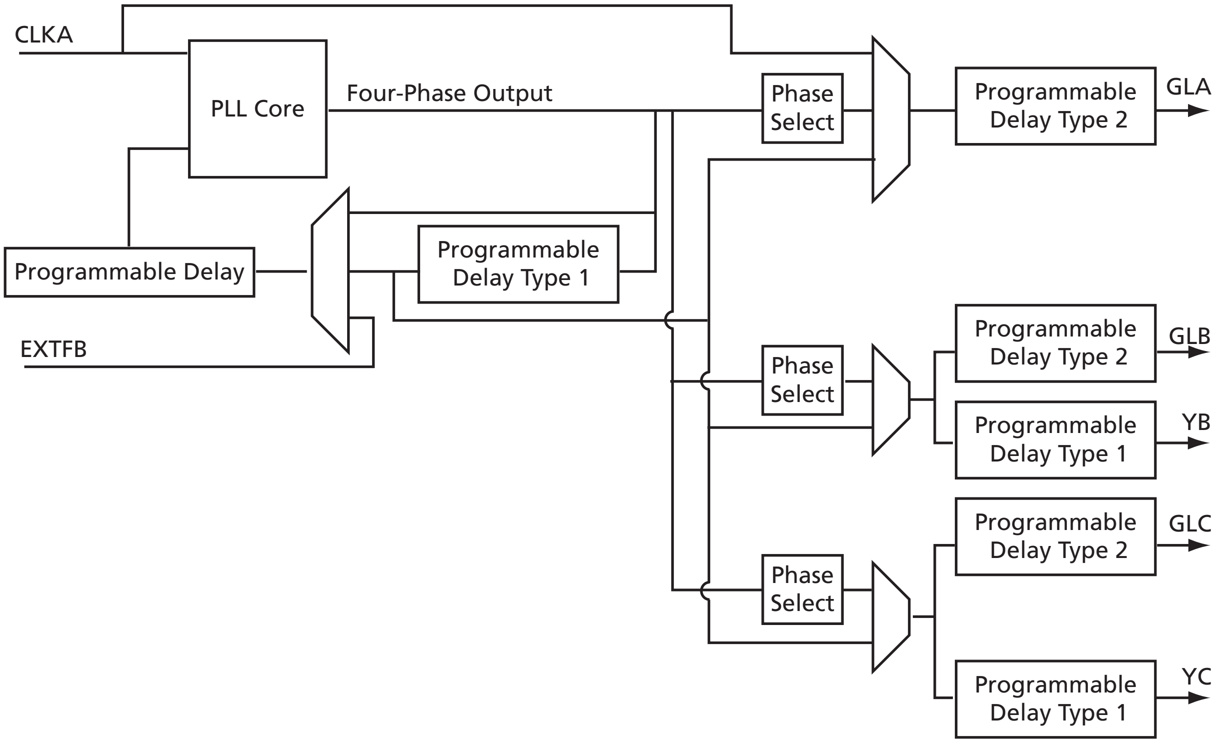

To Core
GxA0
GxA1
GxA2
The reference clock, CLKA, can be assigned on GxA0 or GxA1.
Source for CCC (CLKA or CLKB or CLKC)
Routed Clok (from FPGA core)
To Core
GxB0
GxB1
GxB2
External Feedback (EXTFB) signal is assigned on GxB1 by Designer automatically.
Source for CCC (CLKA or CLKB or CLKC)
Routed Clok (from FPGA core)
x represents global location; can be A, B, C, D, E, or F

Figure 4-5 • **CLKA and EXTFB Assigned to Global I/Os**

SmartGen also allows the user to select the various delays and phase shift values necessary to adjust the phases between the reference clock (CLKA) and the derived clocks (GLA, GLB, GLC, YB, and YC). SmartGen allows the user to select the input clock source. SmartGen automatically instantiates the special macro, PLLINT, when needed.


CLKA
PLL Core
Four-Phase Output
Phase Select
Programmable Delay Type 2
GLA
Programmable Delay
Programmable Delay Type 1
EXTFB
Phase Select
Programmable Delay Type 2
GLB
Programmable Delay Type 1
YB
Programmable Delay Type 2
GLC
Phase Select
Programmable Delay Type 1
YC

Note: *Clock divider and clock multiplier blocks are not shown in this figure or in SmartGen. They are automatically configured based on the user's required frequencies.*

***Figure 4-6* • CCC with PLL Block**

Global Input Selections

Low power flash devices provide the flexibility of choosing one of the three global input pad locations available to connect to a CCC functional block or to a global / quadrant global network. Figure 4-7 on page 74 and Figure 4-8 on page 74 show the detailed architecture of each global input structure for 30 k gate devices and below, as well as 60 k gate devices and above, respectively. For 60 k gate devices and above (Figure 4-7 on page 74), if the single-ended I/O standard is chosen, there is flexibility to choose one of the global input pads (the first, second, and fourth input). Once chosen, the other I/O locations are used as regular I/Os. If the differential I/O standard is chosen (not applicable for IGLOO nano and ProASIC3 nano devices), the first and second inputs are considered as paired, and the third input is paired with a regular I/O.

The user then has the choice of selecting one of the two sets to be used as the clock input source to the CCC functional block. There is also the option to allow an internal clock signal to feed the global network or the CCC functional block. A multiplexer tree selects the appropriate global input for routing to the desired location. Note that the global I/O pads do not need to feed the global network; they can also be used as regular I/O pads.

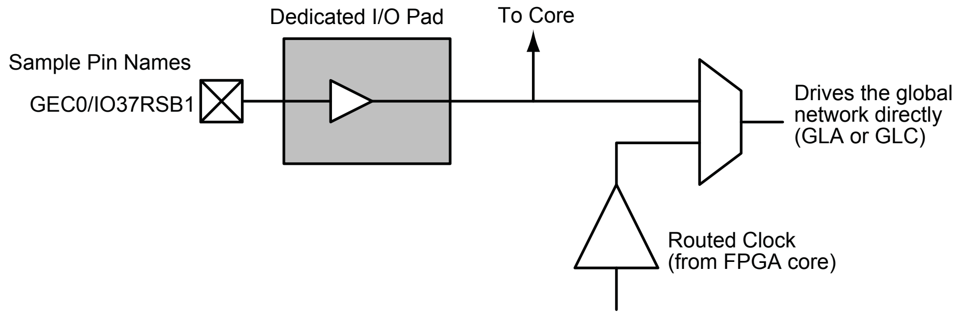
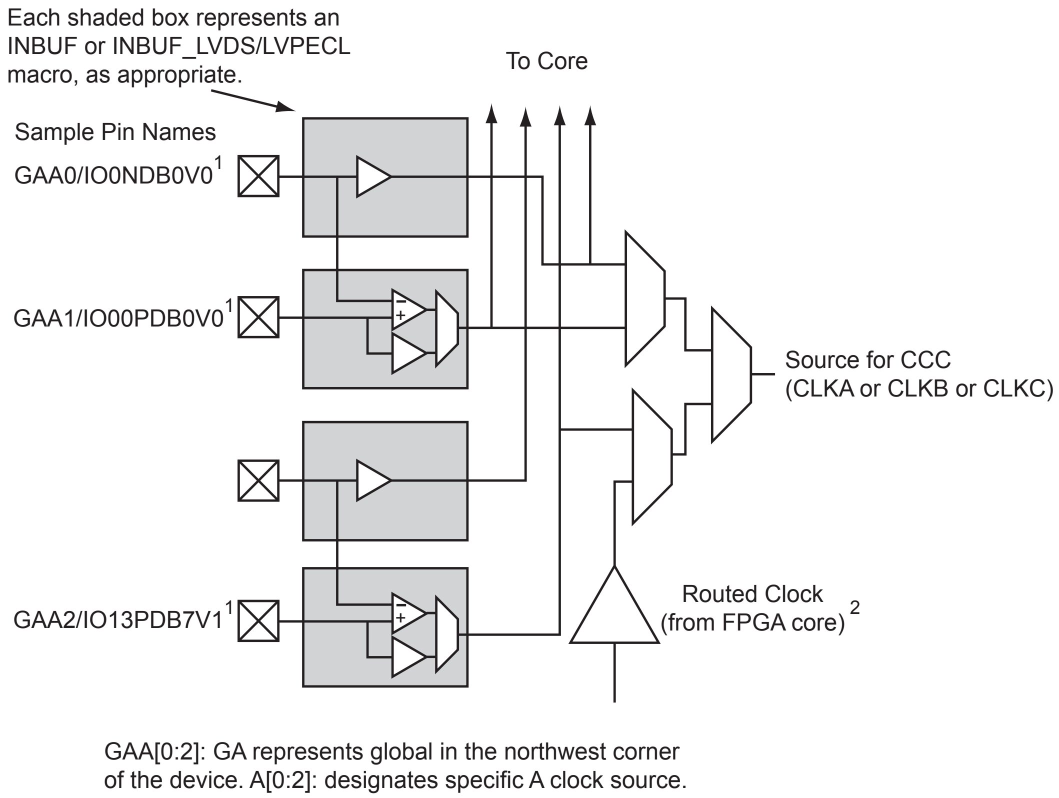

Dedicated I/O Pad
To Core
Sample Pin Names
GEC0/IO37RSB1
Drives the global network directly (GLA or GLC)
Routed Clock (from FPGA core)

Figure 4-7 **• Clock Input Sources (30 k gates devices and below)**


Each shaded box represents an INBUF or INBUF_LVDS/LVPECL macro, as appropriate.
To Core
Sample Pin Names
GAA0/IO0NDB0V0[1]
GAA1/IO00PDB0V0[1]
GAA2/IO13PDB7V1[1]
Source for CCC (CLKA or CLKB or CLKC)
Routed Clock (from FPGA core)[2]
GAA[0:2]: GA represents global in the northwest corner of the device. A[0:2]: designates specific A clock source.

Notes:

1. *Represents the global input pins. Globals have direct access to the clock conditioning block and are not routed via the FPGA fabric. Refer to the "User I/O Naming Conventions in I/O Structures" chapter of the appropriate device user's guide.*
2. *Instantiate the routed clock source input as follows:*
   a) *Connect the output of a logic element to the clock input of a PLL, CLKDLY, or CLKINT macro.*
   b) *Do not place a clock source I/O (INBUF or INBUF_LVPECL/LVDS/B-LVDS/M-LVDS/DDR) in a relevant global pin location.*
3. *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

Figure 4-8 **• Clock Input Sources Including CLKBUF, CLKBUF_LVDS/LVPECL, and CLKINT (60 k gates devices and above)**

Each global buffer, as well as the PLL reference clock, can be driven from one of the following:

- 3 dedicated single-ended I/Os using a hardwired connection
- 2 dedicated differential I/Os using a hardwired connection (not applicable for IGLOO nano and ProASIC3 nano devices)
- The FPGA core

Since the architecture of the devices varies as size increases, the following list details I/O types supported for globals:

IGLOO and ProASIC3

- LVDS-based clock sources are available only on 250 k gate devices and above (IGLOO nano and ProASIC3 nano devices do not support differential inputs).
- 60 k and 125 k gate devices support single-ended clock sources only.
- 15 k and 30 k gate devices support these inputs for CCC only and do not contain a PLL.
- nano devices:
  - 10 k, 15 k, and 20 k devices do not contain PLLs in the CCCs, and support only CLKBUF and CLKINT.
  - 60 k, 125 k, and 250 k devices support one PLL in the middle left CCC position. In the absence of the PLL, this CCC can be used by CLKBUF, CLKINT, and CLKDLY macros. The corner CCCs support CLKBUF, CLKINT, and CLKDLY.

Fusion

- AFS600 and AFS1500: All single-ended, differential, and voltage-referenced I/O standards (Pro I/O).
- AFS090 and AFS250: All single-ended and differential I/O standards.

Clock Sources for PLL and CLKDLY Macros

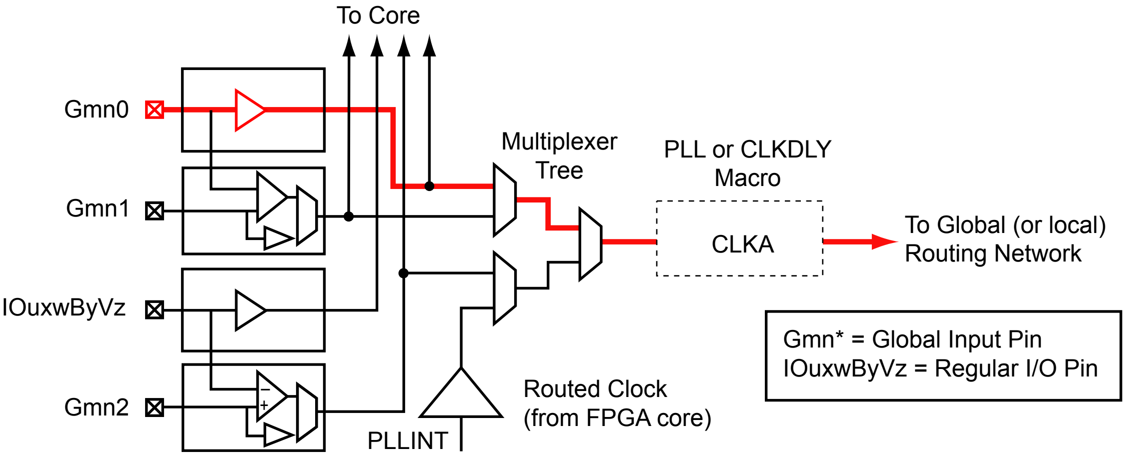
The input reference clock (CLKA for a PLL macro, CLK for a CLKDLY macro) can be accessed from different sources via the associated clock multiplexer tree. Each CCC has the option of choosing the source of the input clock from one of the following:

- Hardwired I/O
- External I/O
- Core Logic
- RC Oscillator (Fusion only)
- Crystal Oscillator (Fusion only)

The SmartGen macro builder tool allows users to easily create the PLL and CLKDLY macros with the desired settings. Microsemi strongly recommends using SmartGen to generate the CCC macros.

Hardwired I/O Clock Source

Hardwired I/O refers to global input pins that are hardwired to the multiplexer tree, which directly accesses the CCC global buffers. These global input pins have designated pin locations and are indicated with the I/O naming convention *Gmn* (*m* refers to any one of the positions where the PLL core is available, and *n* refers to any one of the three global input MUXes and the pin number of the associated global location, *m*). Choosing this option provides the benefit of directly connecting to the CCC reference clock input, which provides less delay. See Figure 4-9 on page 76 for an example illustration of the connections, shown in red. If a CLKDLY macro is initiated to utilize the programmable delay element of the CCC, the clock input can be placed at one of nine dedicated global input pin locations. In other words, if Hardwired I/O is chosen as the input source, the user can decide to place the input pin in one of the GmA0, GmA1, GmA2, GmB0, GmB1, GmB2, GmC0, GmC1, or GmC2 locations of the low power flash devices. When a PLL macro is used to utilize the PLL core in a CCC location, the clock input of the PLL can only be connected to one of three GmA* global pin locations: GmA0, GmA1, or GmA2.



Note: *Fusion CCCs have additional source selections (RCOSC, XTAL).*

Figure 4-9 **• Illustration of Hardwired I/O (global input pins) Usage for IGLOO and ProASIC3 devices 60 k Gates and Larger**

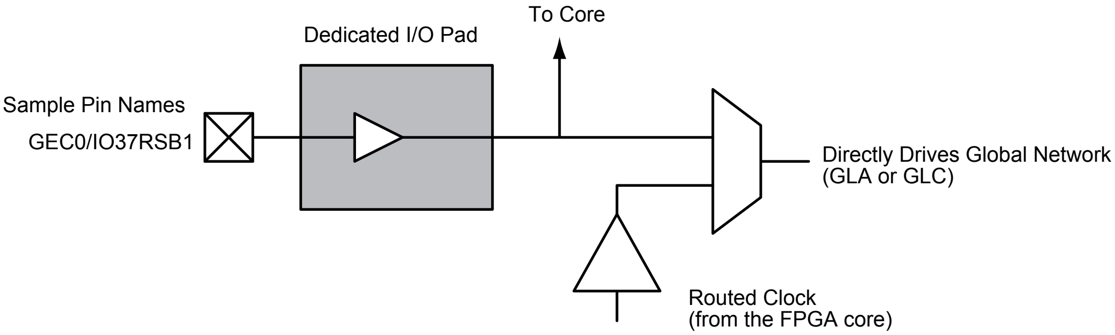

To Core
Dedicated I/O Pad
Sample Pin Names
GEC0/IO37RSB1
Directly Drives Global Network (GLA or GLC)
Routed Clock (from the FPGA core)

Figure 4-10 **• Illustration of Hardwired I/O (global input pins) Usage for IGLOO and ProASIC3 devices 30 k Gates and Smaller**

External I/O Clock Source

External I/O refers to regular I/O pins. The clock source is instantiated with one of the various INBUF options and accesses the CCCs via internal routing. The user has the option of assigning this input to any of the I/Os labeled with the I/O convention *IOuxwByVz*. Refer to the "User I/O Naming Conventions in I/O Structures" chapter of the appropriate device user's guide, and for Fusion, refer to the *Fusion Family of Mixed Signal FPGAs* datasheet for more information. Figure 4-11 gives a brief explanation of external I/O usage. Choosing this option provides the freedom of selecting any user I/O location but introduces additional delay because the signal connects to the routed clock input through internal routing before connecting to the CCC reference clock input.

For the External I/O option, the routed signal would be instantiated with a PLLINT macro before connecting to the CCC reference clock input. This instantiation is conveniently done automatically by SmartGen when this option is selected. Microsemi recommends using the SmartGen tool to generate the CCC macro. The instantiation of the PLLINT macro results in the use of the routed clock input of the I/O to connect to the PLL clock input. If not using SmartGen, manually instantiate a PLLINT macro before the PLL reference clock to indicate that the regular I/O driving the PLL reference clock should be used (see Figure 4-11 for an example illustration of the connections, shown in red).

In the above two options, the clock source must be instantiated with one of the various INBUF macros. The reference clock pins of the CCC functional block core macros must be driven by regular input macros (INBUFs), not clock input macros.

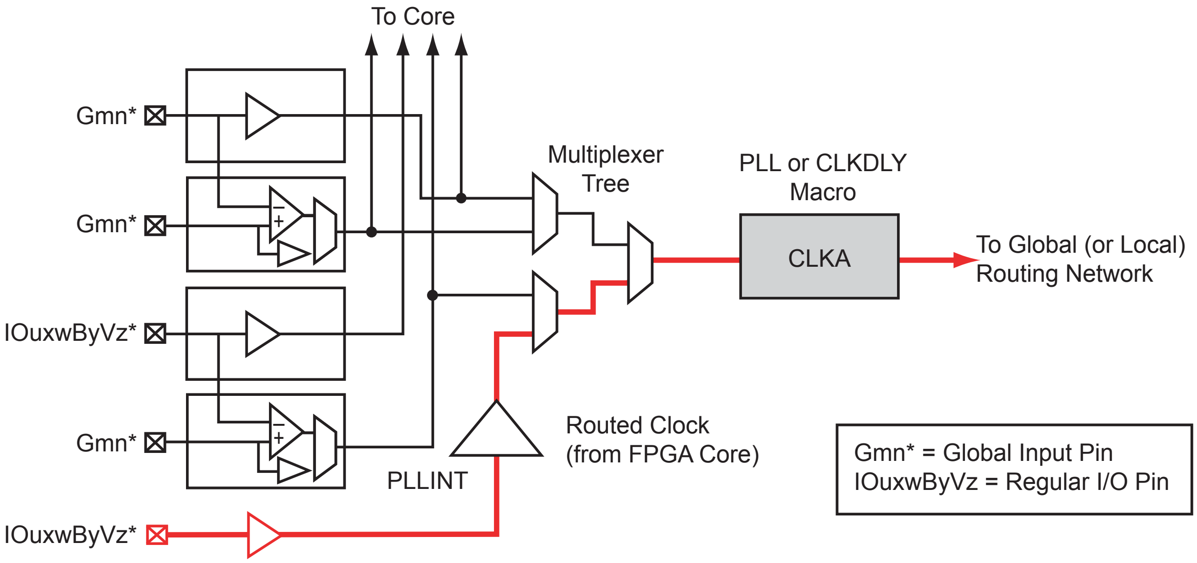


Figure 4-11 • **Illustration of External I/O Usage**

For Fusion devices, the input reference clock can also be from the embedded RC oscillator and crystal oscillator. In this case, the CCC configuration is the same as the hardwired I/O clock source, and users are required to instantiate the RC oscillator or crystal oscillator macro and connect its output to the input reference clock of the CCC block.

Core Logic Clock Source

Core logic refers to internal routed nets. Internal routed signals access the CCC via the FPGA Core Fabric. Similar to the External I/O option, whenever the clock source comes internally from the core itself, the routed signal is instantiated with a PLLINT macro before connecting to the CCC clock input (see Figure 4-12 for an example illustration of the connections, shown in red).

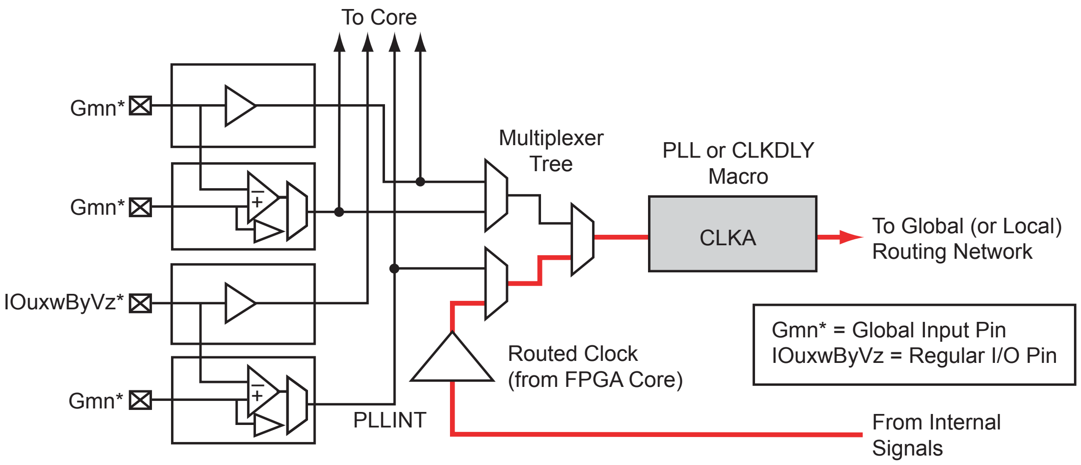


Figure 4-12 **• Illustration of Core Logic Usage**

For Fusion devices, the input reference clock can also be from the embedded RC oscillator and crystal oscillator. In this case, the CCC configuration is the same as the hardwired I/O clock source, and users are required to instantiate the RC oscillator or crystal oscillator macro and connect its output to the input reference clock of the CCC block.

Available I/O Standards

***Table 4-4* • Available I/O Standards within CLKBUF and CLKBUF_LVDS/LVPECL Macros**

CLKBUF_LVCMOS5
CLKBUF_LVCMOS33 [1]
CLKBUF_LVCMOS25 [2]
CLKBUF_LVCMOS18
CLKBUF_LVCMOS15
CLKBUF_PCI
CLKBUF_PCIX [3]
CLKBUF_GTL25 [2,3]
CLKBUF_GTL33 [2,3]
CLKBUF_GTLP25 [2,3]
CLKBUF_GTLP33 [2,3]
CLKBUF_HSTL_I [2,3]
CLKBUF_HSTL_II [2,3]
CLKBUF_SSTL3_I [2,3]
CLKBUF_SSTL3_II [2,3]
CLKBUF_SSTL2_I [2,3]
CLKBUF_SSTL2_II [2,3]
CLKBUF_LVDS [4,5]
CLKBUF_LVPECL[5]

Notes:

1. *By default, the CLKBUF macro uses 3.3 V LVTTL I/O technology. For more details, refer to the IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide.*
2. *I/O standards only supported in ProASIC3E and IGLOOe families.*
3. *I/O standards only supported in the following Fusion devices: AFS600 and AFS1500.*
4. *B-LVDS and M-LVDS standards are supported by CLKBUF_LVDS.*
5. *Not supported for IGLOO nano and ProASIC3 nano devices.*

Global Synthesis Constraints

The Synplify® synthesis tool, by default, allows six clocks in a design for Fusion, IGLOO, and ProASIC3. When more than six clocks are needed in the design, a user synthesis constraint attribute, syn_global_buffers, can be used to control the maximum number of clocks (up to 18) that can be inferred by the synthesis engine.

High-fanout nets will be inferred with clock buffers and/or internal clock buffers. If the design consists of CCC global buffers, they are included in the count of clocks in the design.

The subsections below discuss the clock input source (global buffers with no programmable delays) and the clock conditioning functional block (global buffers with programmable delays and/or PLL function) in detail.

Device-Specific Layout

Two kinds of CCCs are offered in low power flash devices: CCCs with integrated PLLs, and CCCs without integrated PLLs (simplified CCCs). Table 4-5 lists the number of CCCs in various devices.

***Table 4-5* • Number of CCCs by Device Size and Package**

Device			CCCs with Integrated PLLs	CCCs without Integrated PLLs (simplified CCC)
ProASIC3	**IGLOO**	**Package**		
A3PN010	AGLN010	All	0	2
A3PN015	AGLN015	All	0	2
A3PN020	AGLN020	All	0	2
	AGLN060	CS81	0	6
A3PN060	AGLN060	All other packages	1	5
	AGLN125	CS81	0	6
A3PN125	AGLN125	All other packages	1	5
	AGLN250	CS81	0	6
A3PN250	AGLN250	All other packages	1	5
A3P015	AGL015	All	0	2
A3P030	AGL030/AGLP030	All	0	2
	AGL060/AGLP060	CS121/CS201	0	6
A3P060	AGL060/AGLP060	All other packages	1	5
A3P125	AGL125/AGLP125	All	1	5
A3P250/L	AGL250	All	1	5
A3P400	AGL400	All	1	5
A3P600/L	AGL600	All	1	5
A3P1000/L	AGL1000	All	1	5
A3PE600	AGLE600	PQ208	2	4
A3PE600/L		All other packages	6	0
A3PE1500		PQ208	2	4
A3PE1500		All other packages	6	0
A3PE3000/L		PQ208	2	4
A3PE3000/L	AGLE3000	All other packages	6	0
Fusion Devices				
AFS090		All	1	5
AFS250, M1AFS250		All	1	5
AFS600, M7AFS600, M1AFS600		All	2	4
AFS1500, M1AFS1500		All	2	4

Note: nano 10 k, 15 k, and 20 k offer 6 global MUXes instead of CCCs.

This section outlines the following device information: CCC features, PLL core specifications, functional descriptions, software configuration information, detailed usage information, recommended board-level considerations, and other considerations concerning global networks in low power flash devices.

Clock Conditioning Circuits with Integrated PLLs

Each of the CCCs with integrated PLLs includes the following:

- 1 PLL core, which consists of a phase detector, a low-pass filter, and a four-phase voltage-controlled oscillator
- 3 global multiplexer blocks that steer signals from the global pads and the PLL core onto the global networks
- 6 programmable delays and 1 fixed delay for time advance/delay adjustments
- 5 programmable frequency divider blocks to provide frequency synthesis (automatically configured by the SmartGen macro builder tool)

Clock Conditioning Circuits without Integrated PLLs

There are two types of simplified CCCs without integrated PLLs in low power flash devices.

1. The simplified CCC with programmable delays, which is composed of the following:
   - 3 global multiplexer blocks that steer signals from the global pads and the programmable delay elements onto the global networks
   - 3 programmable delay elements to provide time delay adjustments
2. The simplified CCC (referred to as CCC-GL) without programmable delay elements, which is composed of the following:
   - A global multiplexer block that steer signals from the global pads onto the global networks

CCC Locations

CCCs located in the middle of the east and west sides of the device access the three VersaNet global networks on each side (six total networks), while the four CCCs located in the four corners access three quadrant global networks (twelve total networks). See Figure 4-13.

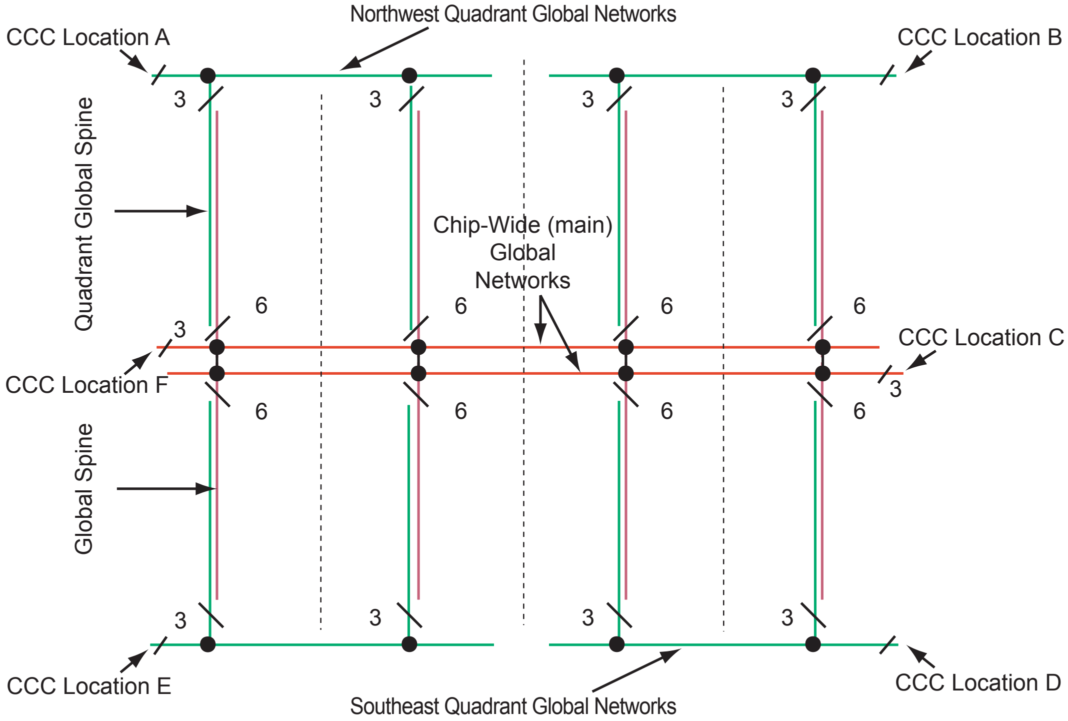


Figure 4-13 • **Global Network Architecture for 60 k Gate Devices and Above**

The following explains the locations of the CCCs in IGLOO and ProASIC3 devices:

In Figure 4-15 on page 84 through Figure 4-16 on page 84, CCCs with integrated PLLs are indicated in red, and simplified CCCs are indicated in yellow. There is a letter associated with each location of the CCC, in clockwise order. The upper left corner CCC is named "A," the upper right is named "B," and so on. These names finish up at the middle left with letter "F."

IGLOO and ProASIC3 CCC Locations

In all IGLOO and ProASIC3 devices (except 10 k through 30 k gate devices, which do not contain PLLs), six CCCs are located in the same positions as the IGLOOe and ProASIC3E CCCs. Only one of the CCCs has an integrated PLL and is located in the middle of the west (middle left) side of the device. The other five CCCs are simplified CCCs and are located in the four corners and the middle of the east side of the device (Figure 4-14).

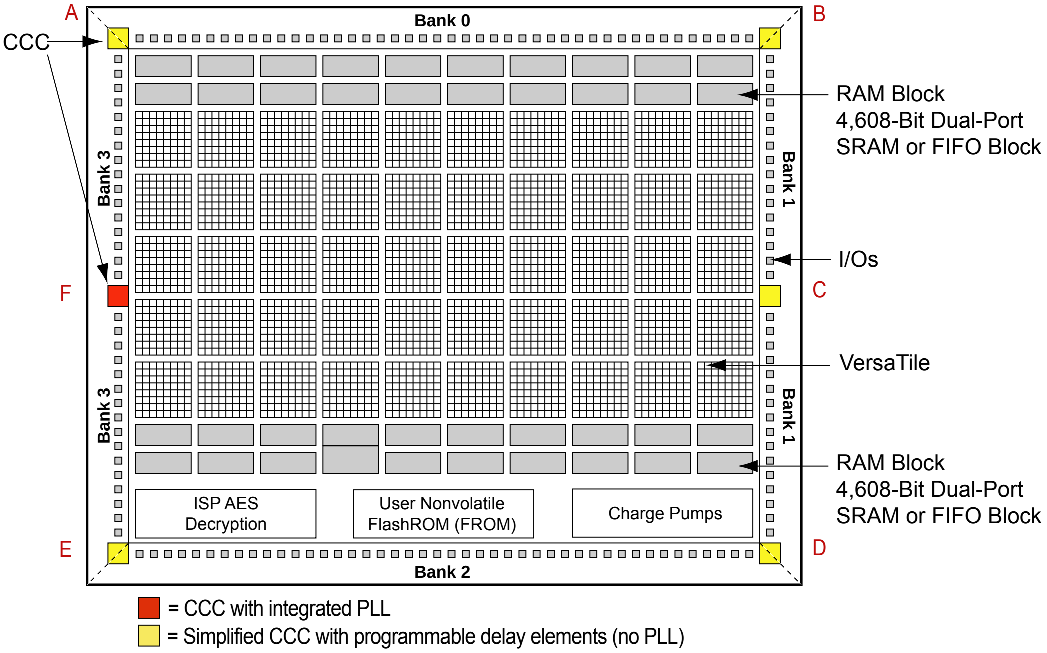


***Figure 4-14* • CCC Locations in IGLOO and ProASIC3 Family Devices (except 10 k through 30 k gate devices)**

Note: The number and architecture of the banks are different for some devices.

10 k through 30 k gate devices do not support PLL features. In these devices, there are two CCC-GLs at the lower corners (one at the lower right, and one at the lower left). These CCC-GLs do not have programmable delays.

IGLOOe and ProASIC3E CCC Locations

IGLOOe and ProASIC3E devices have six CCCs—one in each of the four corners and one each in the middle of the east and west sides of the device (Figure 4-15).

All six CCCs are integrated with PLLs, except in PQFP-208 package devices. PQFP-208 package devices also have six CCCs, of which two include PLLs and four are simplified CCCs. The CCCs with PLLs are implemented in the middle of the east and west sides of the device (middle right and middle left). The simplified CCCs without PLLs are located in the four corners of the device (Figure 4-16).

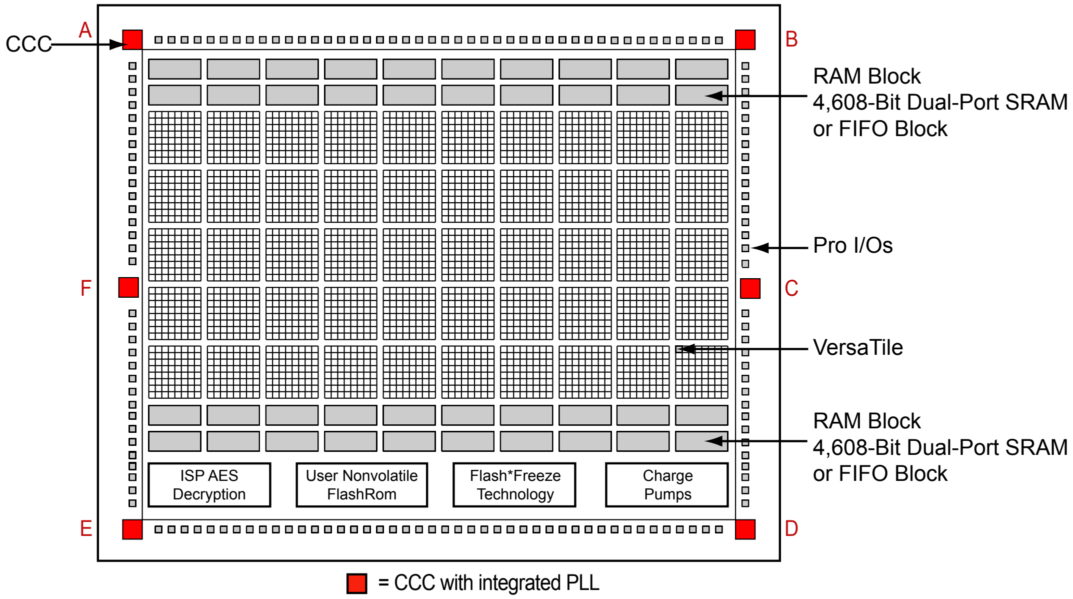


Figure 4-15 • **CCC Locations in IGLOOe and ProASIC3E Family Devices (except PQFP-208 package)**

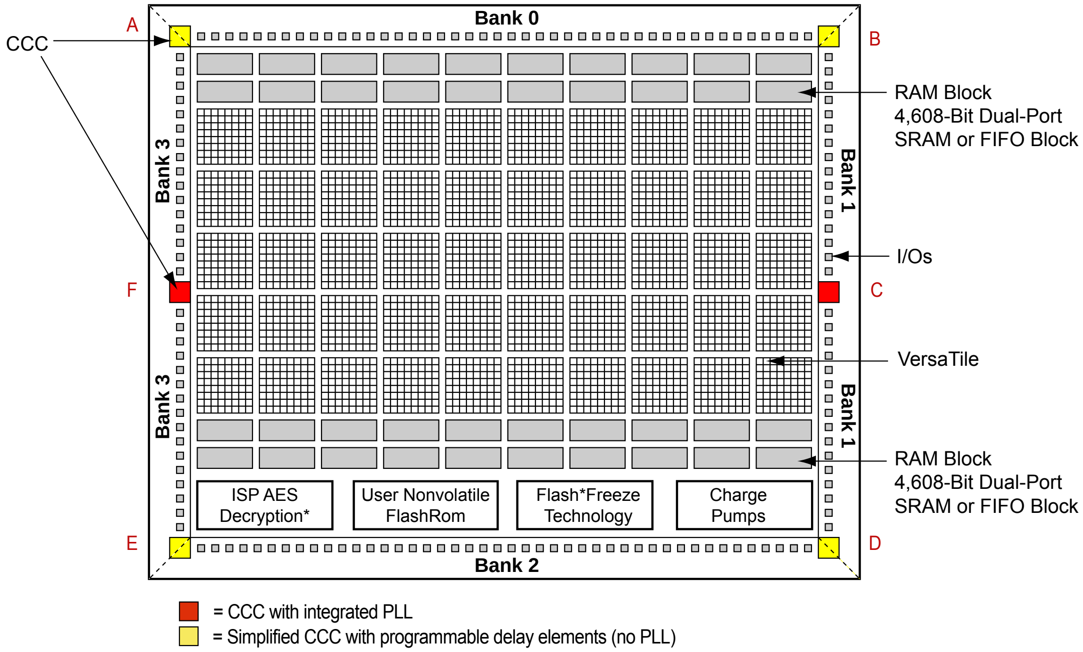


Figure 4-16 • **CCC Locations in ProASIC3E Family Devices (PQFP-208 package)**

Fusion CCC Locations

Fusion devices have six CCCs: one in each of the four corners and one each in the middle of the east and west sides of the device (Figure 4-17 and Figure 4-18). The device can have one integrated PLL in the middle of the west side of the device or two integrated PLLs in the middle of the east and west sides of the device (middle right and middle left).

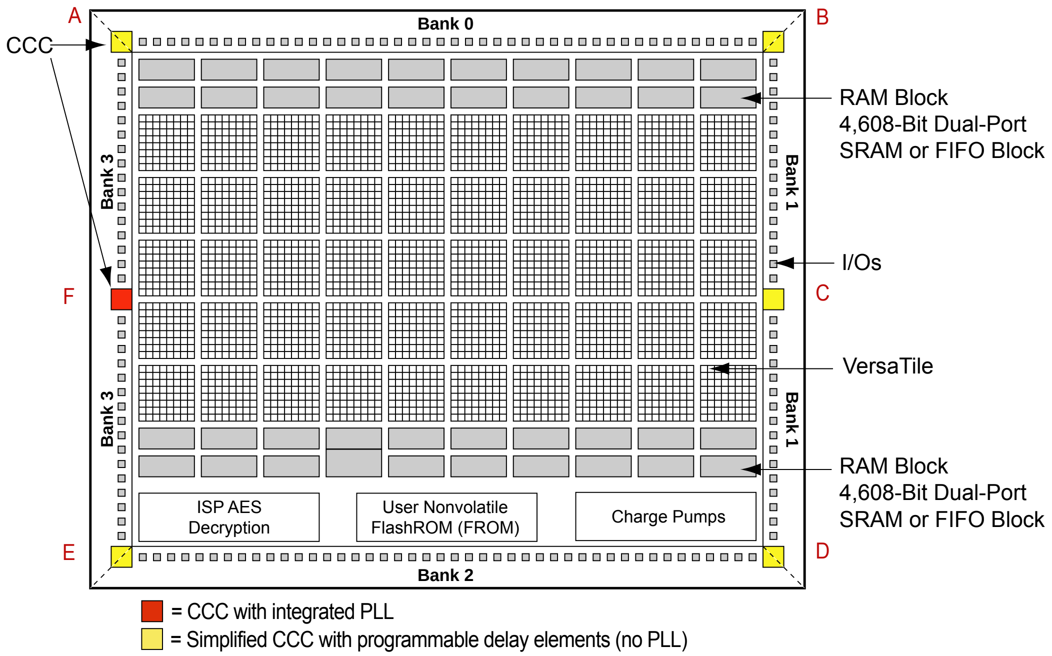


Figure 4-17 **• CCC Locations in Fusion Family Devices (AFS090, AFS250, M1AFS250)**

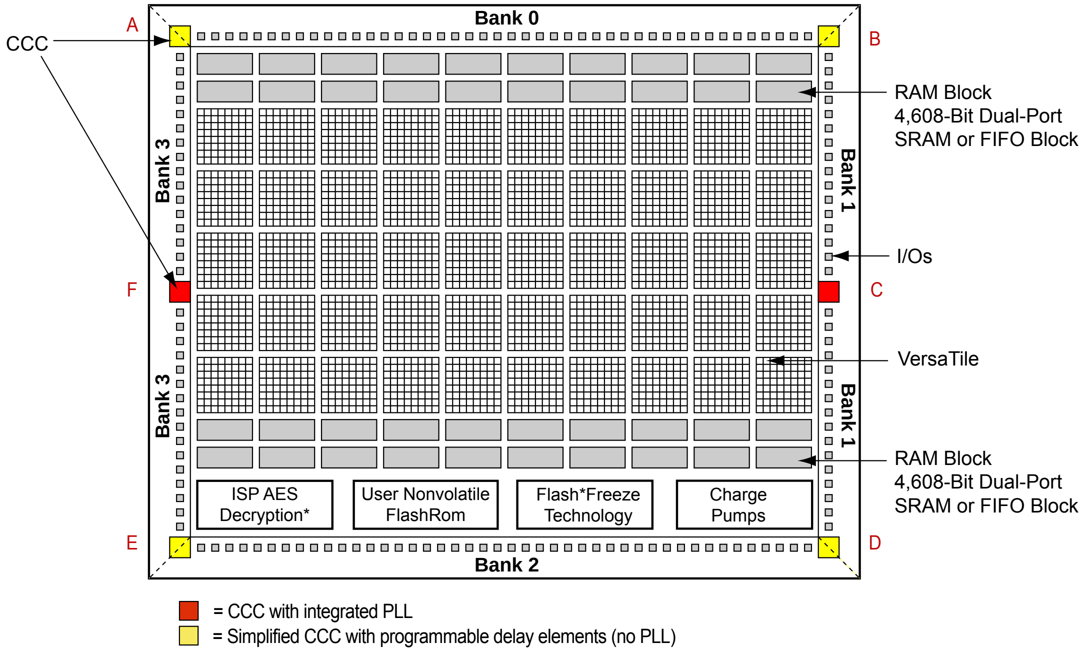


Figure 4-18 **• CCC Locations in Fusion Family Devices (except AFS090, AFS250, M1AFS250)**

PLL Core Specifications

PLL core specifications can be found in the DC and Switching Characteristics chapter of the appropriate family datasheet.

Loop Bandwidth

Common design practice for systems with a low-noise input clock is to have PLLs with small loop bandwidths to reduce the effects of noise sources at the output. Table 4-6 shows the PLL loop bandwidth, providing a measure of the PLL's ability to track the input clock and jitter.

***Table 4-6* • –3 dB Frequency of the PLL**

	Minimum **(T_a = +125°C, VCCA = 1.4 V)**	**Typical** **(T_a = +25°C, VCCA = 1.5 V)**	**Maximum** **(T_a = –55°C, VCCA = 1.6 V)**
–3 dB Frequency	15 kHz	25 kHz	45 kHz

PLL Core Operating Principles

This section briefly describes the basic principles of PLL operation. The PLL core is composed of a phase detector (PD), a low-pass filter (LPF), and a four-phase voltage-controlled oscillator (VCO). Figure 4-19 illustrates a basic single-phase PLL core with a divider and delay in the feedback path.

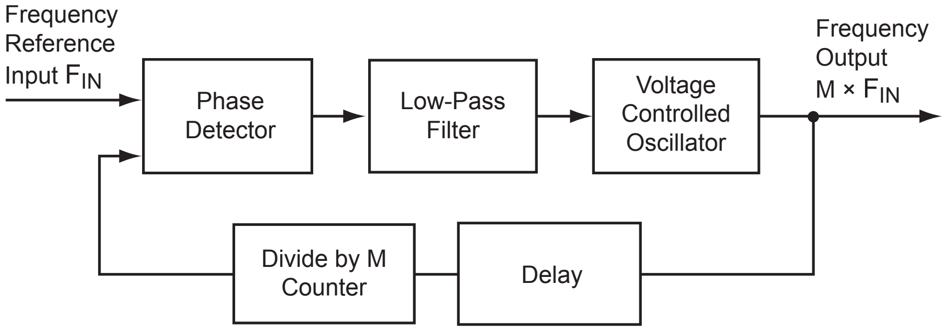

Frequency Reference Input F_{IN}
Phase Detector
Low-Pass Filter
Voltage Controlled Oscillator
Frequency Output M × F_{IN}
Divide by M Counter
Delay

***Figure 4-19* • Simplified PLL Core with Feedback Divider and Delay**

The PLL is an electronic servo loop that phase-aligns the PD feedback signal with the reference input. To achieve this, the PLL dynamically adjusts the VCO output signal according to the average phase difference between the input and feedback signals.

The first element is the PD, which produces a voltage proportional to the phase difference between its inputs. A simple example of a digital phase detector is an Exclusive-OR gate. The second element, the LPF, extracts the average voltage from the phase detector and applies it to the VCO. This applied voltage alters the resonant frequency of the VCO, thus adjusting its output frequency.

Consider Figure 4-19 with the feedback path bypassing the divider and delay elements. If the LPF steadily applies a voltage to the VCO such that the output frequency is identical to the input frequency, this steady-state condition is known as lock. Note that the input and output phases are also identical. The PLL core sets a LOCK output signal HIGH to indicate this condition.

Should the input frequency increase slightly, the PD detects the frequency/phase difference between its reference and feedback input signals. Since the PD output is proportional to the phase difference, the change causes the output from the LPF to increase. This voltage change increases the resonant frequency of the VCO and increases the feedback frequency as a result. The PLL dynamically adjusts in this manner until the PD senses two phase-identical signals and steady-state lock is achieved. The opposite (decreasing PD output signal) occurs when the input frequency decreases.

Now suppose the feedback divider is inserted in the feedback path. As the division factor M (shown in Figure 4-20 on page 87) is increased, the average phase difference increases. The average phase difference will cause the VCO to increase its frequency until the output signal is phase-identical to the input after undergoing division. In other words, lock in both frequency and phase is achieved when the output frequency is M times the input. Thus, clock division in the feedback path results in multiplication at the output.

A similar argument can be made when the delay element is inserted into the feedback path. To achieve steady-state lock, the VCO output signal will be delayed by the input period *less* the feedback delay. For periodic signals, this is equivalent to time-advancing the output clock by the feedback delay.

Another key parameter of a PLL system is the acquisition time. Acquisition time is the amount of time it takes for the PLL to achieve lock (i.e., phase-align the feedback signal with the input reference clock). For example, suppose there is no voltage applied to the VCO, allowing it to operate at its free-running frequency. Should an input reference clock suddenly appear, a lock would be established within the maximum acquisition time.

Functional Description

This section provides detailed descriptions of PLL block functionality: clock dividers and multipliers, clock delay adjustment, phase adjustment, and dynamic PLL configuration.

Clock Dividers and Multipliers

The PLL block contains five programmable dividers. Figure 4-20 shows a simplified PLL block.

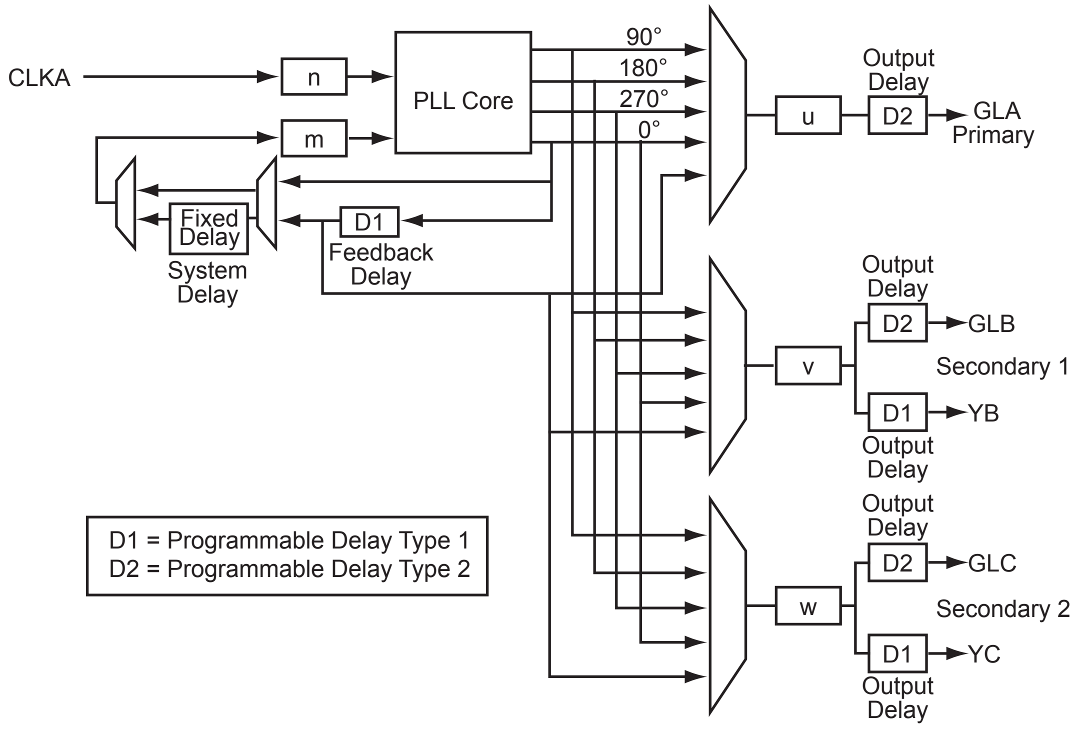


Figure 4-20 • **PLL Block Diagram**

Dividers *n* and *m* (the input divider and feedback divider, respectively) provide integer frequency division factors from 1 to 128. The output dividers *u*, *v*, and *w* provide integer division factors from 1 to 32. Frequency scaling of the reference clock CLKA is performed according to the following formulas:

$f_{GLA} = f_{CLKA} \times m / (n \times u)$ – GLA Primary PLL Output Clock

EQ 4-1

$f_{GLB} = f_{YB} = f_{CLKA} \times m / (n \times v)$ – GLB Secondary 1 PLL Output Clock(s)

EQ 4-2

$f_{GLC} = f_{YC} = f_{CLKA} \times m / (n \times w)$ – GLC Secondary 2 PLL Output Clock(s)

EQ 4-3

SmartGen provides a user-friendly method of generating the configured PLL netlist, which includes automatically setting the division factors to achieve the closest possible match to the requested frequencies. Since the five output clocks share the *n* and *m* dividers, the achievable output frequencies are interdependent and related according to the following formula:

$f_{GLA} = f_{GLB} \times (v / u) = f_{GLC} \times (w / u)$

EQ 4-4

Clock Delay Adjustment

There are a total of seven configurable delay elements implemented in the PLL architecture.

Two of the delays are located in the feedback path, entitled System Delay and Feedback Delay. System Delay provides a fixed delay of 2 ns (typical), and Feedback Delay provides selectable delay values from 0.6 ns to 5.56 ns in 160 ps increments (typical). For PLLs, delays in the feedback path will effectively advance the output signal from the PLL core with respect to the reference clock. Thus, the System and Feedback delays generate negative delay on the output clock. Additionally, each of these delays can be independently bypassed if necessary.

The remaining five delays perform traditional time delay and are located at each of the outputs of the PLL. Besides the fixed global driver delay of 0.755 ns for each of the global networks, the global multiplexer outputs (GLA, GLB, and GLC) each feature an additional selectable delay value, as given in Table 4-7.

***Table 4-7* • Delay Values in Libero SoC Software per Device Family**

Device	Typical	Starting Values	Increments	Ending Value
ProASIC3	200 ps	0 to 735 ps	200 ps	6.735 ns
IGLOO/ProASIC3L 1.5 V	360 ps	0 to 1.610 ns	360 ps	12.410 ns
IGLOO/ProASIC3L 1.2 V	580 ps	0 to 2.880 ns	580 ps	20.280 ns

The additional YB and YC signals have access to a selectable delay from 0.6 ns to 5.56 ns in 160 ps increments (typical). This is the same delay value as the CLKDLY macro. It is similar to CLKDLY, which bypasses the PLL core just to take advantage of the phase adjustment option with the delay value.

The following parameters must be taken into consideration to achieve minimum delay at the outputs (GLA, GLB, GLC, YB, and YC) relative to the reference clock: routing delays from the PLL core to CCC outputs, core outputs and global network output delays, and the feedback path delay. The feedback path delay acts as a time advance of the input clock and will offset any delays introduced beyond the PLL core output. The routing delays are determined from back-annotated simulation and are configuration-dependent.

Phase Adjustment

The four phases available (0, 90, 180, 270) are phases with respect to VCO (PLL output). The VCO is divided to achieve the user's CCC required output frequency (GLA, YB/GLB, YC/GLC). The division happens after the selection of the VCO phase. The effective phase shift is actually the VCO phase shift divided by the output divider. This is why the visual CCC shows both the actual achievable phase and more importantly the actual delay that is equivalent to the phase shift that can be achieved.

Dynamic PLL Configuration

The CCCs can be configured both statically and dynamically.

In addition to the ports available in the Static CCC, the Dynamic CCC has the dynamic shift register signals that enable dynamic reconfiguration of the CCC. With the Dynamic CCC, the ports CLKB and CLKC are also exposed. All three clocks (CLKA, CLKB, and CLKC) can be configured independently.

The CCC block is fully configurable. The following two sources can act as the CCC configuration bits.

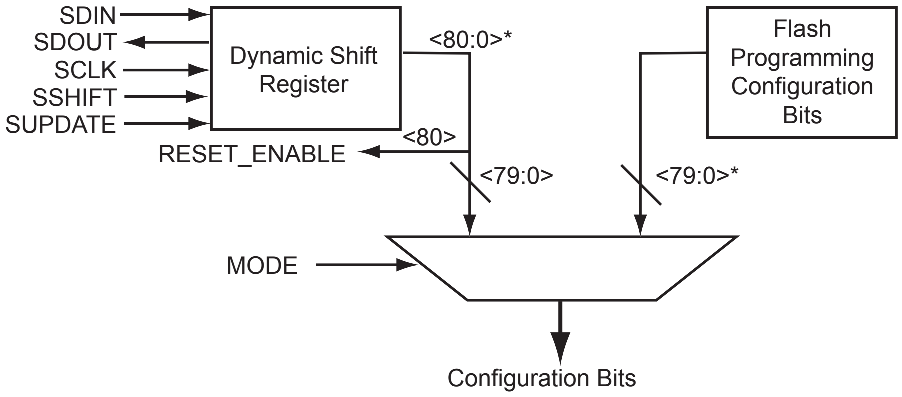
Flash Configuration Bits

The flash configuration bits are the configuration bits associated with programmed flash switches. These bits are used when the CCC is in static configuration mode. Once the device is programmed, these bits cannot be modified. They provide the default operating state of the CCC.

Dynamic Shift Register Outputs

This source does not require core reprogramming and allows core-driven dynamic CCC reconfiguration. When the dynamic register drives the configuration bits, the user-defined core circuit takes full control over SDIN, SDOUT, SCLK, SSHIFT, and SUPDATE. The configuration bits can consequently be dynamically changed through shift and update operations in the serial register interface. Access to the logic core is accomplished via the dynamic bits in the specific tiles assigned to the PLLs.

Figure 4-21 illustrates a simplified block diagram of the MUX architecture in the CCCs.



*Note: *For Fusion, bit <88:81> is also needed.*

Figure 4-21 **• The CCC Configuration MUX Architecture**

The selection between the flash configuration bits and the bits from the configuration register is made using the MODE signal shown in Figure 4-21. If the MODE signal is logic HIGH, the dynamic shift register configuration bits are selected. There are 81 control bits to configure the different functions of the CCC.

Each group of control bits is assigned a specific location in the configuration shift register. For a list of the 81 configuration bits (C[80:0]) in the CCC and a description of each, refer to "PLL Configuration Bits Description" on page 92. The configuration register can be serially loaded with the new configuration data and programmed into the CCC using the following ports:

- SDIN: The configuration bits are serially loaded into a shift register through this port. The LSB of the configuration data bits should be loaded first.
- SDOUT: The shift register contents can be shifted out (LSB first) through this port using the shift operation.
- SCLK: This port should be driven by the shift clock.
- SSHIFT: The active-high shift enable signal should drive this port. The configuration data will be shifted into the shift register if this signal is HIGH. Once SSHIFT goes LOW, the data shifting will be halted.
- SUPDATE: The SUPDATE signal is used to configure the CCC with the new configuration bits when shifting is complete.

To access the configuration ports of the shift register (SDIN, SDOUT, SSHIFT, etc.), the user should instantiate the CCC macro in his design with appropriate ports. Microsemi recommends that users choose SmartGen to generate the CCC macros with the required ports for dynamic reconfiguration.

Users must familiarize themselves with the architecture of the CCC core and its input, output, and configuration ports to implement the desired delay and output frequency in the CCC structure. Figure 4-22 shows a model of the CCC with configurable blocks and switches.

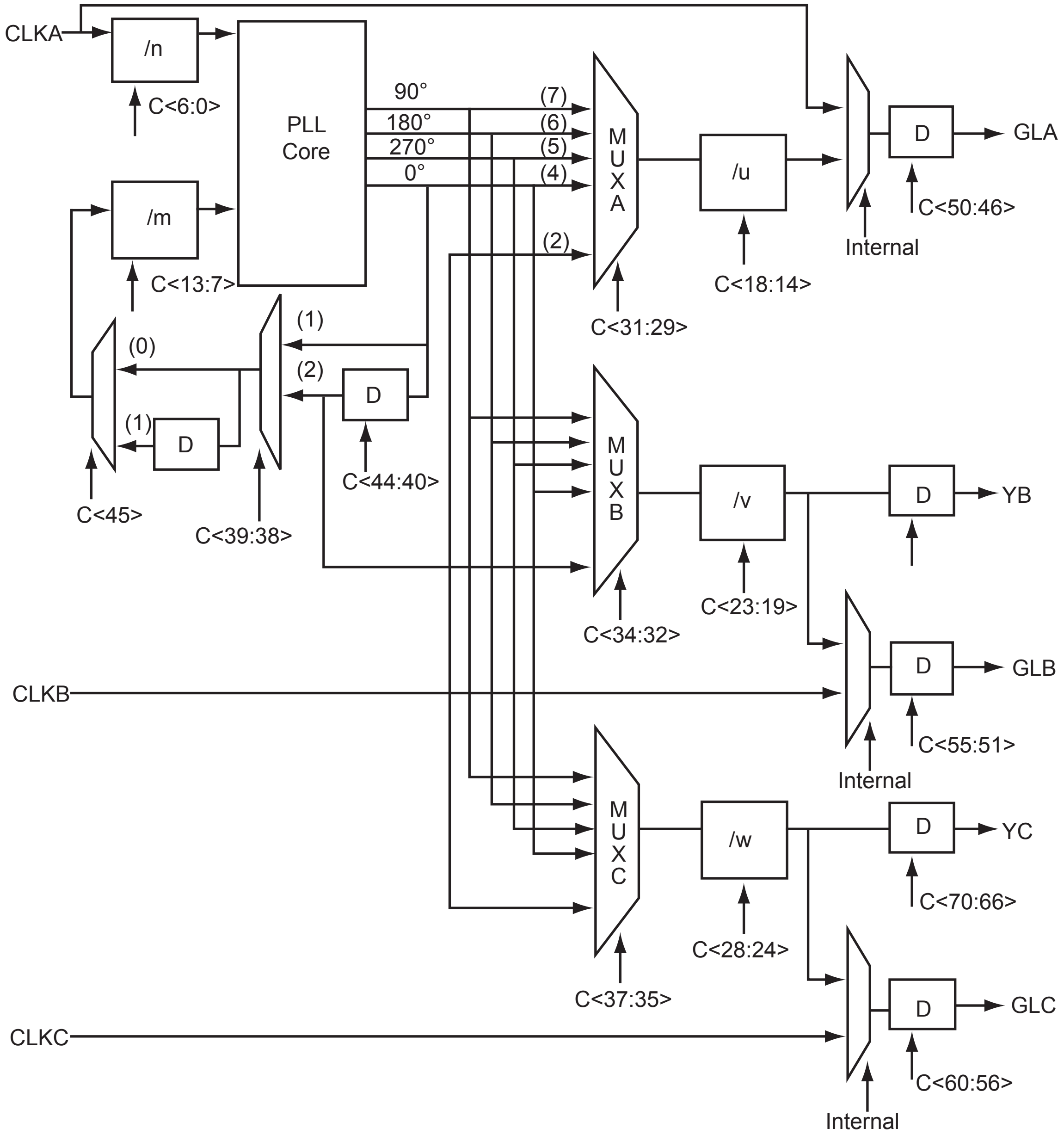


Figure 4-22 • **CCC Block Control Bits – Graphical Representation of Assignments**

Loading the Configuration Register

The most important part of CCC dynamic configuration is to load the shift register properly with the configuration bits. There are different ways to access and load the configuration shift register:

- JTAG interface
- Logic core
- Specific I/O tiles

JTAG Interface

The JTAG interface requires no additional I/O pins. The JTAG TAP controller is used to control the loading of the CCC configuration shift register.

Low power flash devices provide a user interface macro between the JTAG pins and the device core logic. This macro is called UJTAG. A user should instantiate the UJTAG macro in his design to access the configuration register ports via the JTAG pins.

For more information on CCC dynamic reconfiguration using UJTAG, refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 377.

Logic Core

If the logic core is employed, the user must design a module to provide the configuration data and control the shifting and updating of the CCC configuration shift register. In effect, this is a user-designed TAP controller, which requires additional chip resources.

Specific I/O Tiles

If specific I/O tiles are used for configuration, the user must provide the external equivalent of a TAP controller. This does not require additional core resources but does use pins.

Shifting the Configuration Data

To enter a new configuration, all 81 bits must shift in via SDIN. After all bits are shifted, SSHIFT must go LOW and SUPDATE HIGH to enable the new configuration. For simulation purposes, bits <71:73> and <77:80> are "don't care."

The SUPDATE signal must be LOW during any clock cycle where SSHIFT is active. After SUPDATE is asserted, it must go back to the LOW state until a new update is required.

PLL Configuration Bits Description

***Table 4-8* • Configuration Bit Descriptions for the CCC Blocks**

Config. Bits	Signal	Name	Description
<88:87>	GLMUXCFG [1:0][1]	NGMUX configuration	The configuration bits specify the input clocks to the NGMUX (refer to Table 4-17 on page 96).[2]
86	OCDIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by the divider factor in Table 4-18 on page 97.
85	OBDIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by a 0.5 factor (refer to Table 4-18 on page 97).
84	OADIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by certain 0.5 factor (refer to Table 4-16 on page 96).

Notes:

1. *The <88:81> configuration bits are only for the Fusion dynamic CCC.*
2. *This value depends on the input clock source, so Layout must complete before these bits can be set. After completing Layout in Designer, generate the "CCC_Configuration" report by choosing **Tools** > **Report** > **CCC_Configuration**. The report contains the appropriate settings for these bits.*

Table 4-8 **• Configuration Bit Descriptions for the CCC Blocks (continued)**

Config. Bits	Signal	Name	Description
83	RXCSEL[1]	CLKC input selection	Select the CLKC input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 96).[2]
82	RXBSEL[1]	CLKB input selection	Select the CLKB input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 96).[2]
81	RXASEL[1]	CLKA input selection	Select the CLKA input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 96).[2]
80	RESETEN	Reset Enable	Enables (active high) the synchronization of PLL output dividers after dynamic reconfiguration (SUPDATE). The Reset Enable signal is READ-ONLY.
79	DYNCSEL	Clock Input C Dynamic Select	Configures clock input C to be sent to GLC for dynamic control.[2]
78	DYNBSEL	Clock Input B Dynamic Select	Configures clock input B to be sent to GLB for dynamic control.[2]
77	DYNASEL	Clock Input A Dynamic Select	Configures clock input A for dynamic PLL configuration.[2]
<76:74>	VCOSEL[2:0]	VCO Gear Control	Three-bit VCO Gear Control for four frequency ranges (refer to Table 4-19 on page 97 and Table 4-20 on page 97).
73	STATCSEL	MUX Select on Input C	MUX selection for clock input C[2]
72	STATBSEL	MUX Select on Input B	MUX selection for clock input B[2]
71	STATASEL	MUX Select on Input A	MUX selection for clock input A[2]
<70:66>	DLYC[4:0]	YC Output Delay	Sets the output delay value for YC.
<65:61>	DLYB[4:0]	YB Output Delay	Sets the output delay value for YB.
<60:56>	DLYGLC[4:0]	GLC Output Delay	Sets the output delay value for GLC.
<55:51>	DLYGLB[4:0]	GLB Output Delay	Sets the output delay value for GLB.
<50:46>	DLYGLA[4:0]	Primary Output Delay	Primary GLA output delay
45	XDLYSEL	System Delay Select	When selected, inserts System Delay in the feedback path in Figure 4-20 on page 87.
<44:40>	FBDLY[4:0]	Feedback Delay	Sets the feedback delay value for the feedback element in Figure 4-20 on page 87.
<39:38>	FBSEL[1:0]	Primary Feedback Delay Select	Controls the feedback MUX: no delay, include programmable delay element, or use external feedback.
<37:35>	OCMUX[2:0]	Secondary 2 Output Select	Selects from the VCO's four phase outputs for GLC/YC.
<34:32>	OBMUX[2:0]	Secondary 1 Output Select	Selects from the VCO's four phase outputs for GLB/YB.

Notes:

1. *The <88:81> configuration bits are only for the Fusion dynamic CCC.*
2. *This value depends on the input clock source, so Layout must complete before these bits can be set. After completing Layout in Designer, generate the "CCC_Configuration" report by choosing* ***Tools*** *>* ***Report*** *>* ***CCC_Configuration****. The report contains the appropriate settings for these bits.*

***Table 4-8* • Configuration Bit Descriptions for the CCC Blocks (continued)**

Config. Bits	Signal	Name	Description
<31:29>	OAMUX[2:0]	GLA Output Select	Selects from the VCO's four phase outputs for GLA.
<28:24>	OCDIV[4:0]	Secondary 2 Output Divider	Sets the divider value for the GLC/YC outputs. Also known as divider *w* in Figure 4-20 on page 87. The divider value will be OCDIV[4:0] + 1.
<23:19>	OBDIV[4:0]	Secondary 1 Output Divider	Sets the divider value for the GLB/YB outputs. Also known as divider *v* in Figure 4-20 on page 87. The divider value will be OBDIV[4:0] + 1.
<18:14>	OADIV[4:0]	Primary Output Divider	Sets the divider value for the GLA output. Also known as divider *u* in Figure 4-20 on page 87. The divider value will be OADIV[4:0] + 1.
<13:7>	FBDIV[6:0]	Feedback Divider	Sets the divider value for the PLL core feedback. Also known as divider *m* in Figure 4-20 on page 87. The divider value will be FBDIV[6:0] + 1.
<6:0>	FINDIV[6:0]	Input Divider	Input Clock Divider (/n). Sets the divider value for the input delay on CLKA. The divider value will be FINDIV[6:0] + 1.

Notes:

1. *The <88:81> configuration bits are only for the Fusion dynamic CCC.*
2. *This value depends on the input clock source, so Layout must complete before these bits can be set. After completing Layout in Designer, generate the "CCC_Configuration" report by choosing **Tools** > **Report** > **CCC_Configuration**. The report contains the appropriate settings for these bits.*

Table 4-9 to Table 4-15 on page 96 provide descriptions of the configuration data for the configuration bits.

Table 4-9 • **Input Clock Divider, FINDIV[6:0] (/n)**

FINDIV<6:0> State	Divisor	New Frequency Factor
0	1	1.00000
1	2	0.50000
⋮	⋮	⋮
127	128	0.0078125

Table 4-10 • **Feedback Clock Divider, FBDIV[6:0] (/m)**

FBDIV<6:0> State	Divisor	New Frequency Factor
0	1	1
1	2	2
⋮	⋮	⋮
127	128	128

Table 4-11 • **Output Frequency Dividers**
A **Output Divider, OADIV <4:0> (/u);**
B **Output Divider, OBDIV <4:0> (/v);**
C **Output Divider, OCDIV <4:0> (/w)**

OADIV<4:0>; OBDIV<4:0>; CDIV<4:0> State	Divisor	New Frequency Factor
0	1	1.00000
1	2	0.50000
⋮	⋮	⋮
31	32	0.03125

Table 4-12 • **MUXA, MUXB, MUXC**

OAMUX<2:0>; OBMUX<2:0>; OCMUX<2:0> State	MUX Input Selected
0	None. Six-input MUX and PLL are bypassed. Clock passes only through global MUX and goes directly into HC ribs.
1	Not available
2	PLL feedback delay line output
3	Not used
4	PLL VCO 0° phase shift
5	PLL VCO 270° phase shift
6	PLL VCO 180° phase shift
7	PLL VCO 90° phase shift

Table 4-13 • **2-Bit Feedback MUX**

FBSEL<1:0> State	MUX Input Selected
0	Ground. Used for power-down mode in power-down logic block.
1	PLL VCO 0° phase shift
2	PLL delayed VCO 0° phase shift
3	N/A

Table 4-14 • **Programmable Delay Selection for Feedback Delay and Secondary Core Output Delays**

FBDLY<4:0>; DLYYB<4:0>; DLYYC<4:0> State	Delay Value
0	Typical delay = 600 ps
1	Typical delay = 760 ps
2	Typical delay = 920 ps
⋮	⋮
31	Typical delay = 5.56 ns

Table 4-15 • **Programmable Delay Selection for Global Clock Output Delays**

DLYGLA<4:0>; DLYGLB<4:0>; DLYGLC<4:0> State	Delay Value
0	Typical delay = 225 ps
1	Typical delay = 760 ps
2	Typical delay = 920 ps
⋮	⋮
31	Typical delay = 5.56 ns

Table 4-16 • **Fusion Dynamic CCC Clock Source Selection**

RXASEL	DYNASEL	Source of CLKA
1	0	RC Oscillator
1	1	Crystal Oscillator
RXBSEL	**DYNBSEL**	**Source of CLKB**
1	0	RC Oscillator
1	1	Crystal Oscillator
RXBSEL	**DYNCSEL**	**Source of CLKC**
1	0	RC Oscillator
1	1	Crystal Oscillator

Table 4-17 • **Fusion Dynamic CCC NGMUX Configuration**

GLMUXCFG<1:0>	NGMUX Select Signal	Supported Input Clocks to NGMUX
00	0	GLA
	1	GLC
01	0	GLA
	1	GLINT
10	0	GLC
	1	GLINT

Table 4-18 • **Fusion Dynamic CCC Division by Half Configuration**

OADIVHALF / OBDIVHALF / OCDIVHALF	OADIV<4:0> / OBDIV<4:0> / OCDIV<4:0> (in decimal)	Divider Factor	Input Clock Frequency	Output Clock Frequency (MHz)
1	2	1.5	100 MHz RC Oscillator	66.7
	4	2.5		40.0
	6	3.5		28.6
	8	4.5		22.2
	10	5.5		18.2
	12	6.5		15.4
	14	7.5		13.3
	16	8.5		11.8
	18	9.5		10.5
	20	10.5		9.5
	22	11.5		8.7
	24	12.5		8.0
	26	13.5		7.4
	28	14.5		6.9
0	0–31	1–32	Other Clock Sources	Depends on other divider settings

Table 4-19 • **Configuration Bit <76:75> / VCOSEL<2:1> Selection for All Families**

	VCOSEL[2:1]							
	00		01		10		11	
Voltage	Min. (MHz)	Max. (MHz)	Min. (MHz)	Max. (MHz)	Min. (MHz)	Max. (MHz)	Min. (MHz)	Max. (MHz)
IGLOO and IGLOO PLUS								
1.2 V ± 5%	24	35	30	70	60	140	135	160
1.5 V ± 5%	24	43.75	30	87.5	60	175	135	250
ProASIC3L, RT ProASIC3, and Military ProASIC3/L								
1.2 V ± 5%	24	35	30	70	60	140	135	250
1.5 V ± 5%	24	43.75	30	70	60	175	135	350
ProASIC3 and Fusion								
1.5 V ± 5%	24	43.75	33.75	87.5	67.5	175	135	350

Table 4-20 • **Configuration Bit <74> / VCOSEL<0> Selection for All Families**

VCOSEL[0]	Description
0	Fast PLL lock acquisition time with high tracking jitter. Refer to the corresponding datasheet for specific value and definition.
1	Slow PLL lock acquisition time with low tracking jitter. Refer to the corresponding datasheet for specific value and definition.

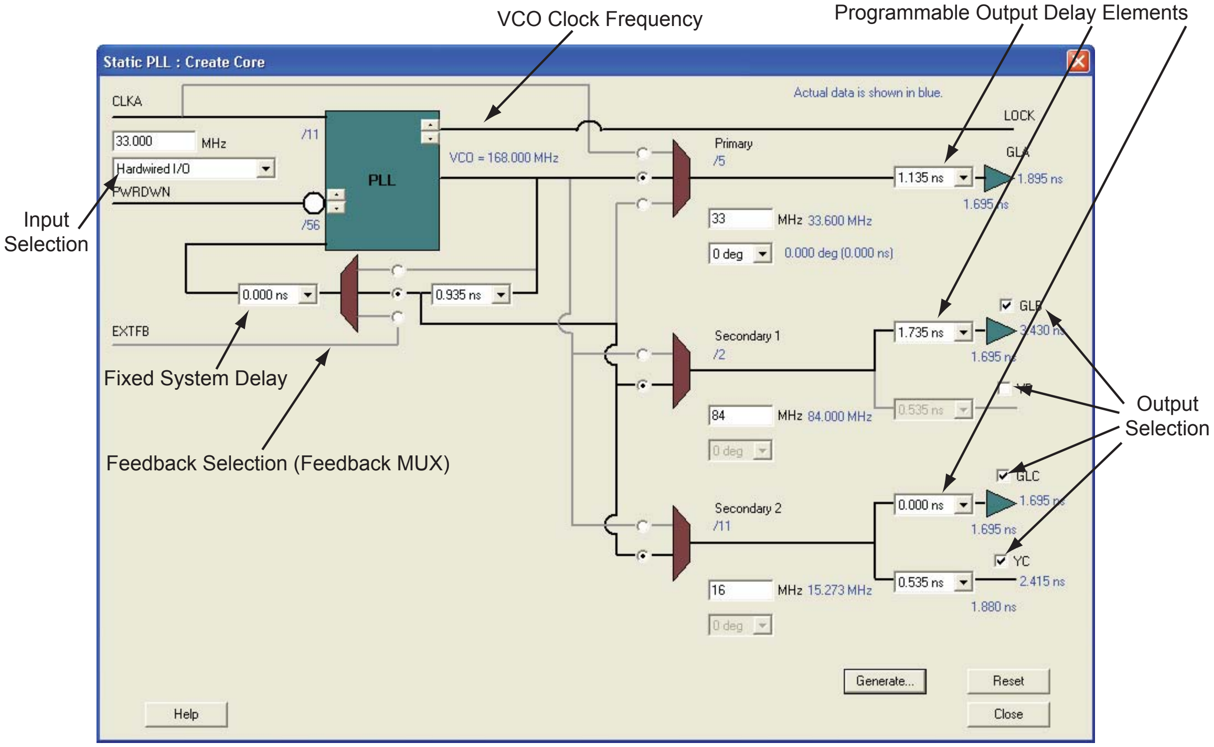
Software Configuration

SmartGen automatically generates the desired CCC functional block by configuring the control bits, and allows the user to select two CCC modes: Static PLL and Delayed Clock (CLKDLY).

Static PLL Configuration

The newly implemented Visual PLL Configuration Wizard feature provides the user a quick and easy way to configure the PLL with the desired settings (Figure 4-23). The user can invoke SmartGen to set the parameters and generate the netlist file with the appropriate flash configuration bits set for the CCCs. As mentioned in "PLL Macro Block Diagram" on page 71, the input reference clock CLKA can be configured to be driven by Hardwired I/O, External I/O, or Core Logic. The user enters the desired settings for all the parameters (output frequency, output selection, output phase adjustment, clock delay, feedback delay, and system delay). Notice that the actual values (divider values, output frequency, delay values, and phase) are shown to aid the user in reaching the desired design frequency in real time. These values are typical-case data. Best- and worst-case data can be observed through static timing analysis in SmartTime within Designer.

For dynamic configuration, the CCC parameters are defined using either the external JTAG port or an internally defined serial interface via the built-in dynamic shift register. This feature provides the ability to compensate for changes in the external environment.



CLKA: 33.000 MHz, Hardwired I/O, /11, /56

VCO = 168.000 MHz

Feedback: 0.000 ns, 0.935 ns

Primary /5: 33 MHz 33.600 MHz; 0 deg 0.000 deg (0.000 ns); GLA 1.135 ns, 1.895 ns, 1.695 ns

Secondary 1 /2: 84 MHz 84.000 MHz; 0 deg; ☑ GLB 1.735 ns, 3.430 ns, 1.695 ns; □ YB 0.535 ns

Secondary 2 /11: 16 MHz 15.273 MHz; 0 deg; ☑ GLC 0.000 ns, 1.695 ns, 1.695 ns; ☑ YC 0.535 ns, 2.415 ns, 1.880 ns

Figure 4-23 • **Visual PLL Configuration Wizard**

Feedback Configuration

The PLL provides both internal and external feedback delays. Depending on the configuration, various combinations of feedback delays can be achieved.

Internal Feedback Configuration

This configuration essentially sets the feedback multiplexer to route the VCO output of the PLL core as the input to the feedback of the PLL. The feedback signal can be processed with the fixed system and the adjustable feedback delay, as shown in Figure 4-24. The dividers are automatically configured by SmartGen based on the user input.

Indicated below is the System Delay pull-down menu. The System Delay can be bypassed by setting it to 0. When set, it adds a 2 ns delay to the feedback path (which results in delay advancement of the output clock by 2 ns).

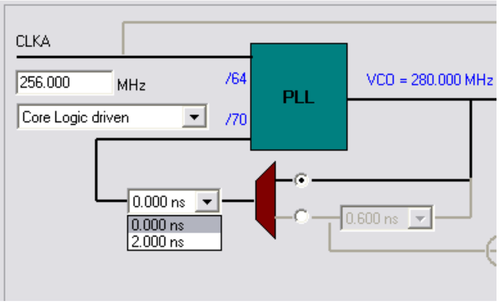


Figure 4-24 **• Internal Feedback with Selectable System Delay**

Figure 4-25 shows the controllable Feedback Delay. If set properly in conjunction with the fixed System Delay, the total output delay can be advanced significantly.

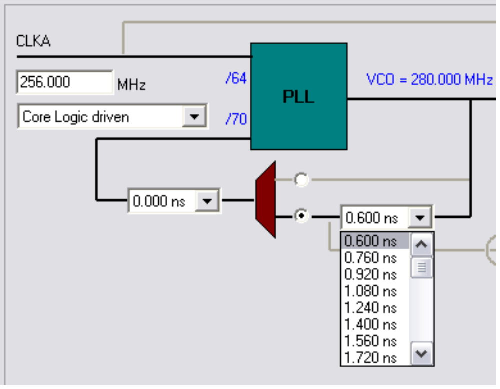


Figure 4-25 **• Internal Feedback with Selectable Feedback Delay**

External Feedback Configuration

For certain applications, such as those requiring generation of PCB clocks that must be matched with existing board delays, it is useful to implement an external feedback, EXTFB. The Phase Detector of the PLL core will receive CLKA and EXTFB as inputs. EXTFB may be processed by the fixed System Delay element as well as the *M* divider element. The EXTFB option is currently not supported.

After setting all the required parameters, users can generate one or more PLL configurations with HDL or EDIF descriptions by clicking the **Generate** button. SmartGen gives the option of saving session results and messages in a log file:

```
****************
Macro Parameters
****************

Name                            : test_pll
Family                          : ProASIC3E
Output Format                   : VHDL
Type                            : Static PLL
Input Freq(MHz)                 : 10.000
CLKA Source                     : Hardwired I/O
Feedback Delay Value Index      : 1
Feedback Mux Select             : 2
XDLY Mux Select                 : No
Primary Freq(MHz)               : 33.000
Primary PhaseShift              : 0
Primary Delay Value Index       : 1
Primary Mux Select              : 4
Secondary1 Freq(MHz)            : 66.000
Use GLB                         : YES
Use YB                          : YES
GLB Delay Value Index           : 1
YB Delay Value Index            : 1
Secondary1 PhaseShift           : 0
Secondary1 Mux Select           : 4
Secondary2 Freq(MHz)            : 101.000
Use GLC                         : YES
Use YC                          : NO
GLC Delay Value Index           : 1
YC Delay Value Index            : 1
Secondary2 PhaseShift           : 0
Secondary2 Mux Select           : 4

…
…
…

Primary Clock frequency 33.333
Primary Clock Phase Shift 0.000
Primary Clock Output Delay from CLKA 0.180

Secondary1 Clock frequency 66.667
Secondary1 Clock Phase Shift 0.000
Secondary1 Clock Global Output Delay from CLKA 0.180
Secondary1 Clock Core Output Delay from CLKA 0.625

Secondary2 Clock frequency 100.000
Secondary2 Clock Phase Shift 0.000
Secondary2 Clock Global Output Delay from CLKA 0.180
```

Below is an example Verilog HDL description of a legal PLL core configuration generated by SmartGen:

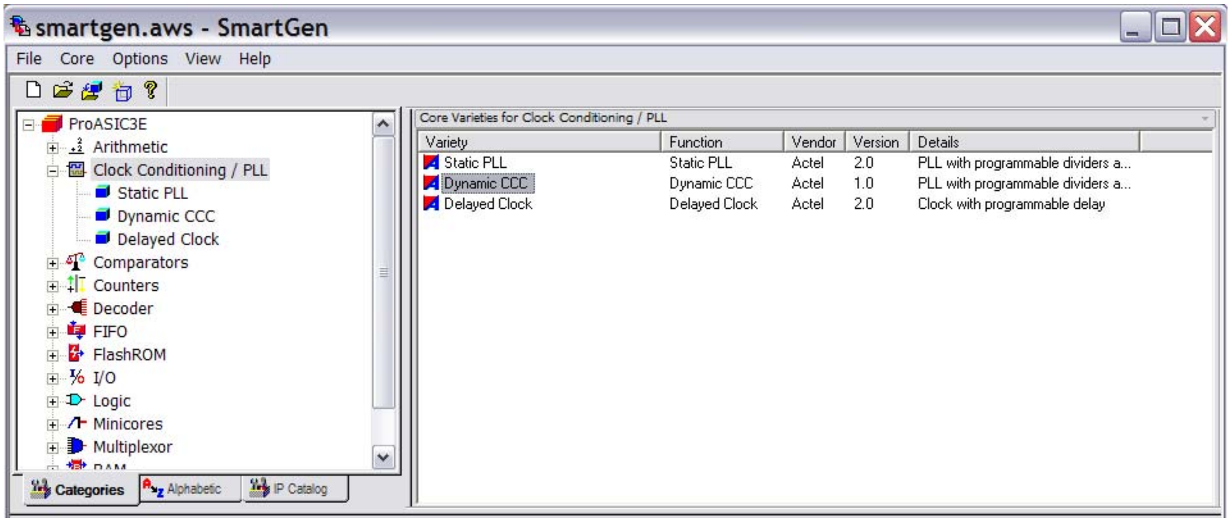
```
module test_pll(POWERDOWN,CLKA,LOCK,GLA);
input POWERDOWN, CLKA;
output  LOCK, GLA;
```

```
    wire VCC, GND;

    VCC VCC_1_net(.Y(VCC));
    GND GND_1_net(.Y(GND));
    PLL Core(.CLKA(CLKA), .EXTFB(GND), .POWERDOWN(POWERDOWN),
        .GLA(GLA), .LOCK(LOCK), .GLB(), .YB(), .GLC(), .YC(),
        .OADIV0(GND), .OADIV1(GND), .OADIV2(GND), .OADIV3(GND),
        .OADIV4(GND), .OAMUX0(GND), .OAMUX1(GND), .OAMUX2(VCC),
        .DLYGLA0(GND), .DLYGLA1(GND), .DLYGLA2(GND), .DLYGLA3(GND)
        , .DLYGLA4(GND), .OBDIV0(GND), .OBDIV1(GND), .OBDIV2(GND),
        .OBDIV3(GND), .OBDIV4(GND), .OBMUX0(GND), .OBMUX1(GND),
        .OBMUX2(GND), .DLYYB0(GND), .DLYYB1(GND), .DLYYB2(GND),
        .DLYYB3(GND), .DLYYB4(GND), .DLYGLB0(GND), .DLYGLB1(GND),
        .DLYGLB2(GND), .DLYGLB3(GND), .DLYGLB4(GND), .OCDIV0(GND),
        .OCDIV1(GND), .OCDIV2(GND), .OCDIV3(GND), .OCDIV4(GND),
        .OCMUX0(GND), .OCMUX1(GND), .OCMUX2(GND), .DLYYC0(GND),
        .DLYYC1(GND), .DLYYC2(GND), .DLYYC3(GND), .DLYYC4(GND),
        .DLYGLC0(GND), .DLYGLC1(GND), .DLYGLC2(GND), .DLYGLC3(GND)
        , .DLYGLC4(GND), .FINDIV0(VCC), .FINDIV1(GND), .FINDIV2(
        VCC), .FINDIV3(GND), .FINDIV4(GND), .FINDIV5(GND),
        .FINDIV6(GND), .FBDIV0(VCC), .FBDIV1(GND), .FBDIV2(VCC),
        .FBDIV3(GND), .FBDIV4(GND), .FBDIV5(GND), .FBDIV6(GND),
        .FBDLY0(GND), .FBDLY1(GND), .FBDLY2(GND), .FBDLY3(GND),
        .FBDLY4(GND), .FBSEL0(VCC), .FBSEL1(GND), .XDLYSEL(GND),
        .VCOSEL0(GND), .VCOSEL1(GND), .VCOSEL2(GND));
    defparam Core.VCOFREQUENCY = 33.000;
endmodule
```

The "PLL Configuration Bits Description" section on page 92 provides descriptions of the PLL configuration bits for completeness. The configuration bits are shown as busses only for purposes of illustration. They will actually be broken up into individual pins in compilation libraries and all simulation models. For example, the FBSEL[1:0] bus will actually appear as pins FBSEL1 and FBSEL0. The setting of these select lines for the static PLL configuration is performed by the software and is completely transparent to the user.

Dynamic PLL Configuration

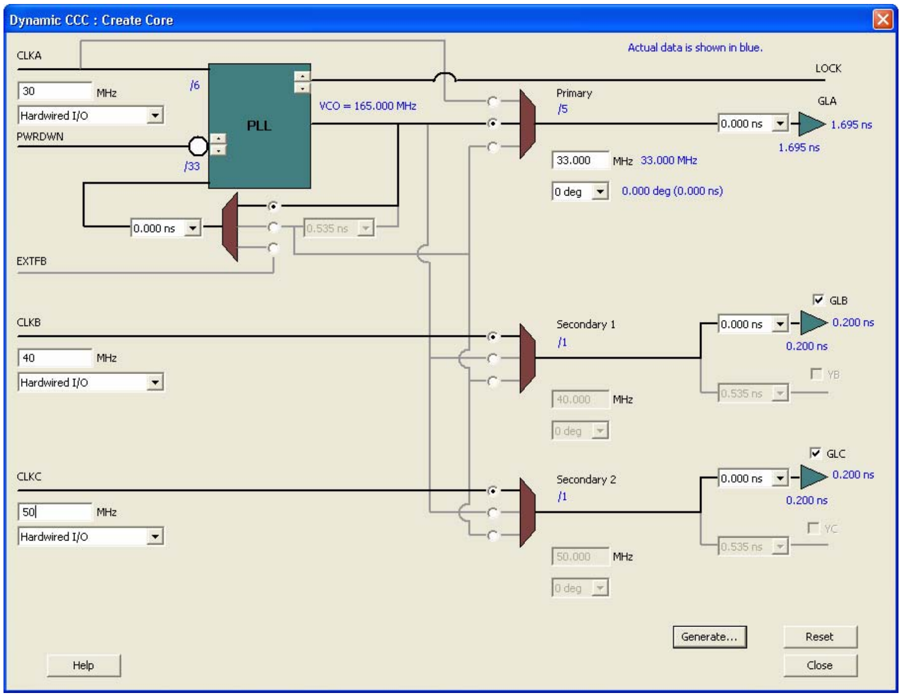
To generate a dynamically reconfigurable CCC, the user should select **Dynamic CCC** in the configuration section of the SmartGen GUI (Figure 4-26). This will generate both the CCC core and the configuration shift register / control bit MUX.



Variety	Function	Vendor	Version	Details
Static PLL	Static PLL	Actel	2.0	PLL with programmable dividers a...
Dynamic CCC	Dynamic CCC	Actel	1.0	PLL with programmable dividers a...
Delayed Clock	Delayed Clock	Actel	2.0	Clock with programmable delay

Figure 4-26 **• SmartGen GUI**

Even if dynamic configuration is selected in SmartGen, the user must still specify the static configuration data for the CCC (Figure 4-27). The specified static configuration is used whenever the MODE signal is set to LOW and the CCC is required to function in the static mode. The static configuration data can be used as the default behavior of the CCC where required.


Dynamic CCC : Create Core

Actual data is shown in blue.

CLKA: 30 MHz, Hardwired I/O

PWRDWN

PLL: /6, /33, VCO = 165.000 MHz

Feedback delay: 0.000 ns; 0.535 ns

EXTFB

LOCK

Primary /5: 33.000 MHz 33.000 MHz; 0 deg 0.000 deg (0.000 ns)

GLA: 0.000 ns, 1.695 ns, 1.695 ns

CLKB: 40 MHz, Hardwired I/O

Secondary 1 /1: 40.000 MHz; 0 deg

☑ GLB: 0.000 ns, 0.200 ns, 0.200 ns

☐ YB: 0.535 ns

CLKC: 50 MHz, Hardwired I/O

Secondary 2 /1: 50.000 MHz; 0 deg

☑ GLC: 0.000 ns, 0.200 ns, 0.200 ns

☐ YC: 0.535 ns

Figure 4-27 **• Dynamic CCC Configuration in SmartGen**

When SmartGen is used to define the configuration that will be shifted in via the serial interface, SmartGen prints out the values of the 81 configuration bits. For ease of use, several configuration bits are automatically inferred by SmartGen when the dynamic PLL core is generated; however, <71:73> (STATASEL, STATBSEL, STATCSEL) and <77:79> (DYNASEL, DYNBSEL, DYNCSEL) depend on the input clock source of the corresponding CCC. Users must first run Layout in Designer to determine the exact setting for these ports. After Layout is complete, generate the "CCC_Configuration" report by choosing **Tools** > **Reports** > **CCC_Configuration** in the Designer software. Refer to "PLL Configuration Bits Description" on page 92 for descriptions of the PLL configuration bits. For simulation purposes, bits <71:73> and <78:80> are "don't care." Therefore, it is strongly suggested that SmartGen be used to generate the correct configuration bit settings for the dynamic PLL core.

After setting all the required parameters, users can generate one or more PLL configurations with HDL or EDIF descriptions by clicking the **Generate** button. SmartGen gives the option of saving session results and messages in a log file:

```
****************
Macro Parameters
****************

Name                                : dyn_pll_hardio
Family                              : ProASIC3E
Output Format                       : VERILOG
Type                                : Dynamic CCC
Input Freq(MHz)                     : 30.000
CLKA Source                         : Hardwired I/O
Feedback Delay Value Index          : 1
Feedback Mux Select                 : 1
XDLY Mux Select                     : No
Primary Freq(MHz)                   : 33.000
Primary PhaseShift                  : 0
Primary Delay Value Index           : 1
Primary Mux Select                  : 4
Secondary1 Freq(MHz)                : 40.000
Use GLB                             : YES
Use YB                              : NO
GLB Delay Value Index               : 1
YB Delay Value Index                : 1
Secondary1 PhaseShift               : 0
Secondary1 Mux Select               : 0
Secondary1 Input Freq(MHz)          : 40.000
CLKB Source                         : Hardwired I/O
Secondary2 Freq(MHz)                : 50.000
Use GLC                             : YES
Use YC                              : NO
GLC Delay Value Index               : 1
YC Delay Value Index                : 1
Secondary2 PhaseShift               : 0
Secondary2 Mux Select               : 0
Secondary2 Input Freq(MHz)          : 50.000
CLKC Source                         : Hardwired I/O

Configuration Bits:
FINDIV[6:0]     0000101
FBDIV[6:0]      0100000
OADIV[4:0]      00100
OBDIV[4:0]      00000
OCDIV[4:0]      00000
OAMUX[2:0]      100
OBMUX[2:0]      000
OCMUX[2:0]      000
FBSEL[1:0]      01
FBDLY[4:0]      00000
XDLYSEL         0
DLYGLA[4:0]     00000
DLYGLB[4:0]     00000
```

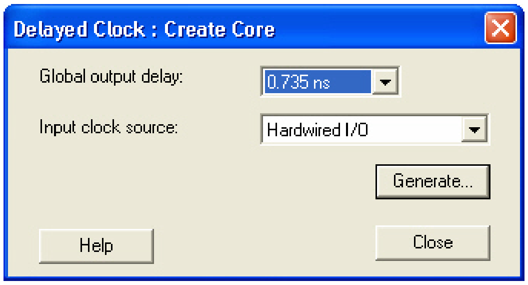
```
DLYGLC[4:0]     00000
DLYYB[4:0]      00000
DLYYC[4:0]      00000
VCOSEL[2:0]     100

Primary Clock Frequency 33.000
Primary Clock Phase Shift 0.000
Primary Clock Output Delay from CLKA 1.695

Secondary1 Clock Frequency 40.000
Secondary1 Clock Phase Shift 0.000
Secondary1 Clock Global Output Delay from CLKB 0.200

Secondary2 Clock Frequency 50.000
Secondary2 Clock Phase Shift 0.000
Secondary2 Clock Global Output Delay from CLKC 0.200

######################################
# Dynamic Stream Data
######################################
--------------------------------------
|NAME    |SDIN     |VALUE   |TYPE     |
--------------------------------------
|FINDIV  |[6:0]    |0000101 |EDIT     |
|FBDIV   |[13:7]   |0100000 |EDIT     |
|OADIV   |[18:14]  |00100   |EDIT     |
|OBDIV   |[23:19]  |00000   |EDIT     |
|OCDIV   |[28:24]  |00000   |EDIT     |
|OAMUX   |[31:29]  |100     |EDIT     |
|OBMUX   |[34:32]  |000     |EDIT     |
|OCMUX   |[37:35]  |000     |EDIT     |
|FBSEL   |[39:38]  |01      |EDIT     |
|FBDLY   |[44:40]  |00000   |EDIT     |
|XDLYSEL |[45]     |0       |EDIT     |
|DLYGLA  |[50:46]  |00000   |EDIT     |
|DLYGLB  |[55:51]  |00000   |EDIT     |
|DLYGLC  |[60:56]  |00000   |EDIT     |
|DLYYB   |[65:61]  |00000   |EDIT     |
|DLYYC   |[70:66]  |00000   |EDIT     |
|STATASEL|[71]     |X       |MASKED   |
|STATBSEL|[72]     |X       |MASKED   |
|STATCSEL|[73]     |X       |MASKED   |
|VCOSEL  |[76:74]  |100     |EDIT     |
|DYNASEL |[77]     |X       |MASKED   |
|DYNBSEL |[78]     |X       |MASKED   |
|DYNCSEL |[79]     |X       |MASKED   |
|RESETEN |[80]     |1       |READONLY |
```

Below is the resultant Verilog HDL description of a legal dynamic PLL core configuration generated by SmartGen:

```
module dyn_pll_macro(POWERDOWN, CLKA, LOCK, GLA, GLB, GLC, SDIN, SCLK, SSHIFT, SUPDATE,
  MODE, SDOUT, CLKB, CLKC);

input POWERDOWN, CLKA;
output  LOCK, GLA, GLB, GLC;
input  SDIN, SCLK, SSHIFT, SUPDATE, MODE;
output  SDOUT;
input  CLKB, CLKC;

  wire VCC, GND;

  VCC VCC_1_net(.Y(VCC));
  GND GND_1_net(.Y(GND));
  DYNCCC Core(.CLKA(CLKA), .EXTFB(GND), .POWERDOWN(POWERDOWN), .GLA(GLA), .LOCK(LOCK),
    .CLKB(CLKB), .GLB(GLB), .YB(), .CLKC(CLKC), .GLC(GLC), .YC(), .SDIN(SDIN),
    .SCLK(SCLK), .SSHIFT(SSHIFT), .SUPDATE(SUPDATE), .MODE(MODE), .SDOUT(SDOUT),
    .OADIV0(GND), .OADIV1(GND), .OADIV2(VCC), .OADIV3(GND), .OADIV4(GND), .OAMUX0(GND),
    .OAMUX1(GND), .OAMUX2(VCC), .DLYGLA0(GND), .DLYGLA1(GND), .DLYGLA2(GND),
    .DLYGLA3(GND), .DLYGLA4(GND), .OBDIV0(GND), .OBDIV1(GND), .OBDIV2(GND),
    .OBDIV3(GND), .OBDIV4(GND), .OBMUX0(GND), .OBMUX1(GND), .OBMUX2(GND), .DLYYB0(GND),
    .DLYYB1(GND), .DLYYB2(GND), .DLYYB3(GND), .DLYYB4(GND), .DLYGLB0(GND),
    .DLYGLB1(GND), .DLYGLB2(GND), .DLYGLB3(GND), .DLYGLB4(GND), .OCDIV0(GND),
    .OCDIV1(GND), .OCDIV2(GND), .OCDIV3(GND), .OCDIV4(GND), .OCMUX0(GND), .OCMUX1(GND),
    .OCMUX2(GND), .DLYYC0(GND), .DLYYC1(GND), .DLYYC2(GND), .DLYYC3(GND), .DLYYC4(GND),
    .DLYGLC0(GND), .DLYGLC1(GND), .DLYGLC2(GND), .DLYGLC3(GND), .DLYGLC4(GND),
    .FINDIV0(VCC), .FINDIV1(GND), .FINDIV2(VCC), .FINDIV3(GND), .FINDIV4(GND),
    .FINDIV5(GND), .FINDIV6(GND), .FBDIV0(GND), .FBDIV1(GND), .FBDIV2(GND),
    .FBDIV3(GND), .FBDIV4(GND), .FBDIV5(VCC), .FBDIV6(GND), .FBDLY0(GND), .FBDLY1(GND),
    .FBDLY2(GND), .FBDLY3(GND), .FBDLY4(GND), .FBSEL0(VCC), .FBSEL1(GND),
    .XDLYSEL(GND), .VCOSEL0(GND), .VCOSEL1(GND), .VCOSEL2(VCC));
  defparam Core.VCOFREQUENCY = 165.000;

endmodule
```

Delayed Clock Configuration

The CLKDLY macro can be generated with the desired delay and input clock source (Hardwired I/O, External I/O, or Core Logic), as in Figure 4-28.



Delayed Clock : Create Core

Global output delay: 0.735 ns

Input clock source: Hardwired I/O

***Figure 4-28* • Delayed Clock Configuration Dialog Box**

After setting all the required parameters, users can generate one or more PLL configurations with HDL or EDIF descriptions by clicking the **Generate** button. SmartGen gives the option of saving session results and messages in a log file:

```
****************
Macro Parameters
****************

Name                            : delay_macro
Family                          : ProASIC3
Output Format                   : Verilog
Type                            : Delayed Clock
Delay Index                     : 2
CLKA Source                     : Hardwired I/O

Total Clock Delay = 0.935 ns.

The resultant CLKDLY macro Verilog netlist is as follows:

module delay_macro(GL,CLK);

output GL;
input  CLK;
```

```
  wire VCC, GND;

  VCC VCC_1_net(.Y(VCC));
  GND GND_1_net(.Y(GND));
  CLKDLY Inst1(.CLK(CLK), .GL(GL), .DLYGL0(VCC), .DLYGL1(GND), .DLYGL2(VCC),
    .DLYGL3(GND), .DLYGL4(GND));

endmodule
```

Detailed Usage Information

Clock Frequency Synthesis

Deriving clocks of various frequencies from a single reference clock is known as frequency synthesis. The PLL has an input frequency range from 1.5 to 350 MHz. This frequency is automatically divided down to a range between 1.5 MHz and 5.5 MHz by input dividers (not shown in Figure 4-19 on page 86) between PLL macro inputs and PLL phase detector inputs. The VCO output is capable of an output range from 24 to 350 MHz. With dividers before the input to the PLL core and following the VCO outputs, the VCO output frequency can be divided to provide the final frequency range from 0.75 to 350 MHz. Using SmartGen, the dividers are automatically set to achieve the closest possible matches to the specified output frequencies.

Users should be cautious when selecting the desired PLL input and output frequencies and the I/O buffer standard used to connect to the PLL input and output clocks. Depending on the I/O standards used for the PLL input and output clocks, the I/O frequencies have different maximum limits. Refer to the family datasheets for specifications of maximum I/O frequencies for supported I/O standards. Desired PLL input or output frequencies will not be achieved if the selected frequencies are higher than the maximum I/O frequencies allowed by the selected I/O standards. Users should be careful when selecting the I/O standards used for PLL input and output clocks. Performing post-layout simulation can help detect this type of error, which will be identified with pulse width violation errors. Users are strongly encouraged to perform post-layout simulation to ensure the I/O standard used can provide the desired PLL input or output frequencies. Users can also choose to cascade PLLs together to achieve the high frequencies needed for their applications. Details of cascading PLLs are discussed in the "Cascading CCCs" section on page 111.

In SmartGen, the actual generated frequency (under typical operating conditions) will be displayed beside the requested output frequency value. This provides the ability to determine the exact frequency that can be generated by SmartGen, in real time. The log file generated by SmartGen is a useful tool in determining how closely the requested clock frequencies match the user specifications. For example, assume a user specifies 101 MHz as one of the secondary output frequencies. If the best output frequency that could be achieved were 100 MHz, the log file generated by SmartGen would indicate the actual generated frequency.

Simulation Verification

The integration of the generated PLL and CLKDLY modules is similar to any VHDL component or Verilog module instantiation in a larger design; i.e., there is no special requirement that users need to take into account to successfully synthesize their designs.

For simulation purposes, users need to refer to the VITAL or Verilog library that includes the functional description and associated timing parameters. Refer to the Software Tools section of the Microsemi SoC Products Group website to obtain the family simulation libraries. If Designer is installed, these libraries are stored in the following locations:

<Designer_Installation_Directory>\lib\vtl\95\proasic3.vhd

<Designer_Installation_Directory>\lib\vtl\95\proasic3e.vhd

<Designer_Installation_Directory>\lib\vlog\proasic3.v

<Designer_Installation_Directory>\lib\vlog\proasic3e.v

For Libero users, there is no need to compile the simulation libraries, as they are conveniently pre-compiled in the Model*Sim*® Microsemi simulation tool.

The following is an example of a PLL configuration utilizing the clock frequency synthesis and clock delay adjustment features. The steps include generating the PLL core with SmartGen, performing simulation for verification with Model*Sim*, and performing static timing analysis with SmartTime in Designer.

Parameters of the example PLL configuration:

Input Frequency – 20 MHz

Primary Output Requirement – 20 MHz with clock advancement of 3.02 ns

Secondary 1 Output Requirement – 40 MHz with clock delay of 2.515 ns

Figure 4-29 shows the SmartGen settings. Notice that the overall delays are calculated automatically, allowing the user to adjust the delay elements appropriately to obtain the desired delays.

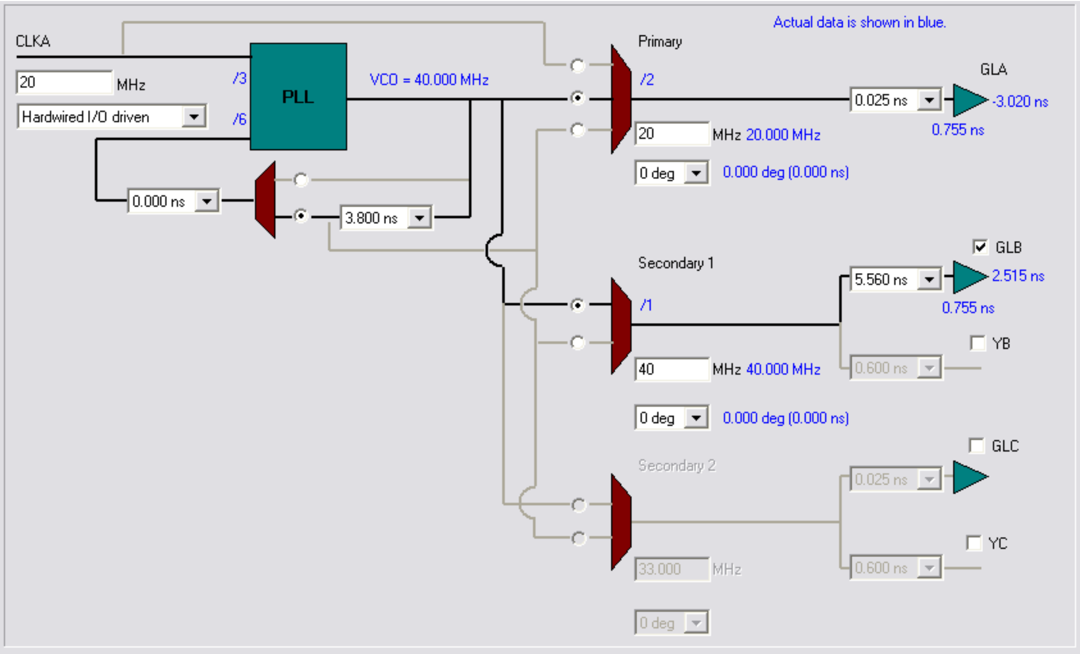
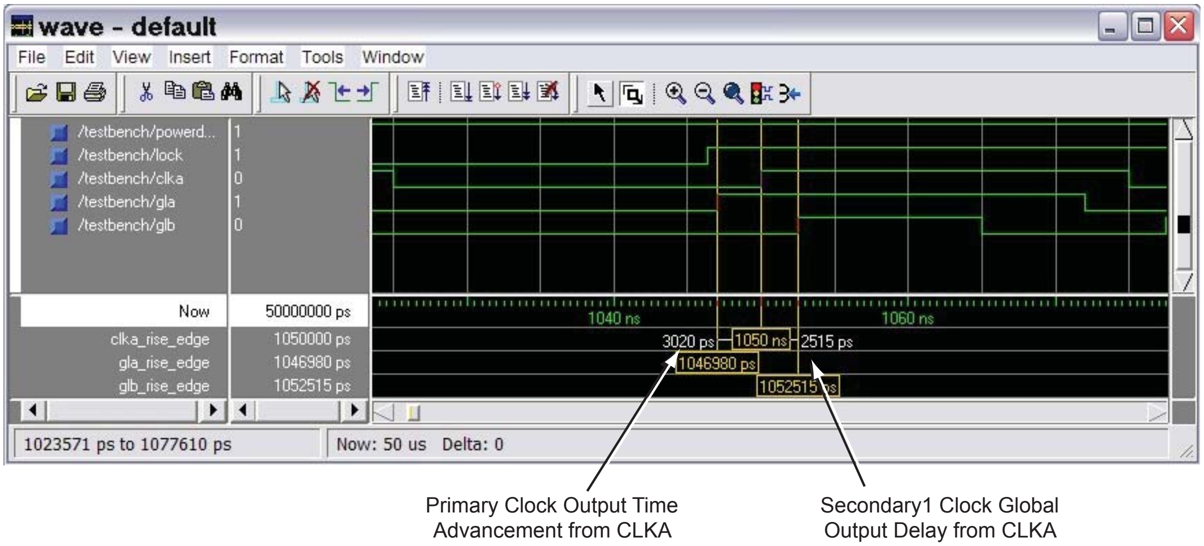


Figure 4-29 • **SmartGen Settings**

After confirming the correct settings, generate a structural netlist of the PLL and verify PLL core settings by checking the log file:

```
Name                             : test_pll_delays
Family                           : ProASIC3E
Output Format                    : VHDL
Type                             : Static PLL
Input Freq(MHz)                  : 20.000
CLKA Source                      : Hardwired I/O
Feedback Delay Value Index       : 21
Feedback Mux Select              : 2
XDLY Mux Select                  : No
Primary Freq(MHz)                : 20.000
Primary PhaseShift               : 0
Primary Delay Value Index        : 1
Primary Mux Select               : 4
Secondary1 Freq(MHz)             : 40.000
Use GLB                          : YES
Use YB                           : NO
…
…
…
Primary Clock frequency 20.000
Primary Clock Phase Shift 0.000
Primary Clock Output Delay from CLKA -3.020

Secondary1 Clock frequency 40.000
Secondary1 Clock Phase Shift 0.000
Secondary1 Clock Global Output Delay from CLKA 2.515
```

Next, perform simulation in Model*Sim* to verify the correct delays. Figure 4-30 shows the simulation results. The delay values match those reported in the SmartGen PLL Wizard.



/testbench/powerd... 1
/testbench/lock 1
/testbench/clka 0
/testbench/gla 1
/testbench/glb 0

Now 50000000 ps
clka_rise_edge 1050000 ps
gla_rise_edge 1046980 ps
glb_rise_edge 1052515 ps

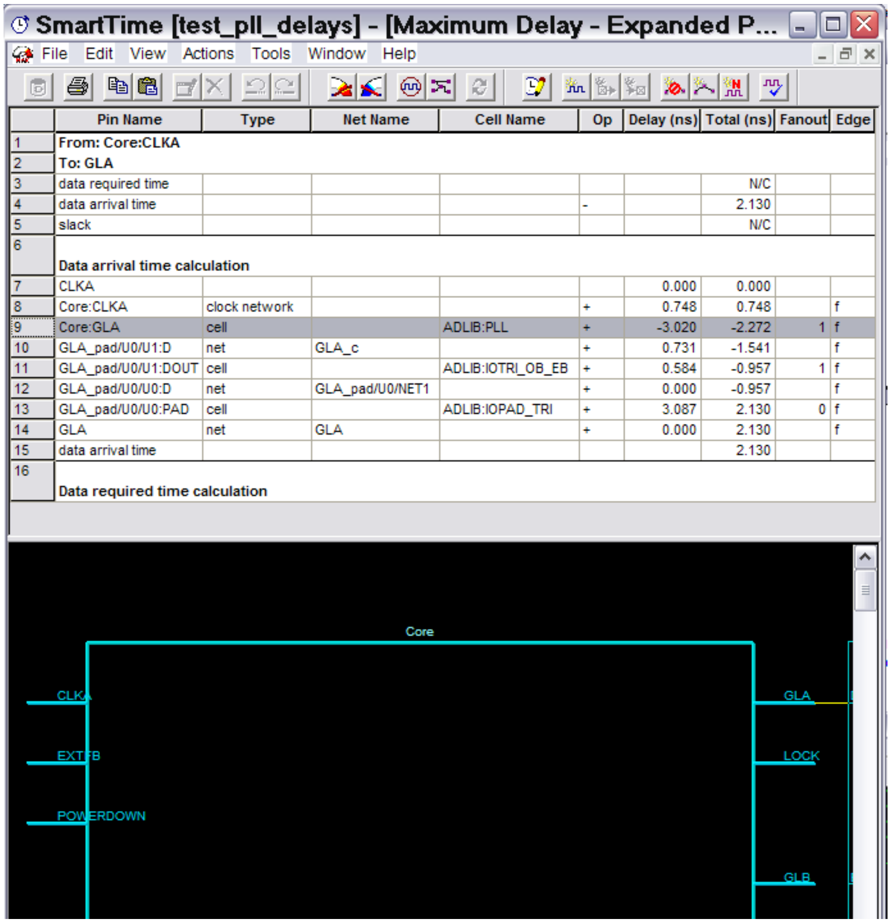
1023571 ps to 1077610 ps
Now: 50 us Delta: 0

Primary Clock Output Time Advancement from CLKA

Secondary1 Clock Global Output Delay from CLKA

***Figure 4-30* • Model*Sim* Simulation Results**

The timing can also be analyzed using SmartTime in Designer. The user should import the synthesized netlist to Designer, perform Compile and Layout, and then invoke SmartTime. Go to **Tools** > **Options** and change the maximum delay operating conditions to **Typical Case**. Then expand the Clock-to-Out paths of GLA and GLB and the individual components of the path delays are shown. The path of GLA is shown in Figure 4-31 on page 109 displaying the same delay value.


SmartTime [test_pll_delays] - [Maximum Delay - Expanded P...
File Edit View Actions Tools Window Help
Core
CLKA
EXTFB
POWERDOWN
GLA
LOCK
GLB

	Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
1	**From: Core:CLKA**								
2	**To: GLA**								
3	data required time						N/C		
4	data arrival time				-		2.130		
5	slack						N/C		
6	**Data arrival time calculation**								
7	CLKA					0.000	0.000		
8	Core:CLKA	clock network			+	0.748	0.748		f
9	Core:GLA	cell		ADLIB:PLL	+	-3.020	-2.272	1	f
10	GLA_pad/U0/U1:D	net	GLA_c		+	0.731	-1.541		f
11	GLA_pad/U0/U1:DOUT	cell		ADLIB:IOTRI_OB_EB	+	0.584	-0.957	1	f
12	GLA_pad/U0/U0:D	net	GLA_pad/U0/NET1		+	0.000	-0.957		f
13	GLA_pad/U0/U0:PAD	cell		ADLIB:IOPAD_TRI	+	3.087	2.130	0	f
14	GLA	net	GLA		+	0.000	2.130		f
15	data arrival time						2.130		
16	**Data required time calculation**								

Figure 4-31 **• Static Timing Analysis Using SmartTime**

Place-and-Route Stage Considerations

Several considerations must be noted to properly place the CCC macros for layout.

For CCCs with clock inputs configured with the Hardwired I/O–Driven option:

- PLL macros must have the clock input pad coming from one of the GmA* locations.
- CLKDLY macros must have the clock input pad coming from one of the Global I/Os.

If a PLL with a Hardwired I/O input is used at a CCC location and a Hardwired I/O–Driven CLKDLY macro is used at the same CCC location, the clock input of the CLKDLY macro must be chosen from one of the GmB* or GmC* pin locations. If the PLL is not used or is an External I/O–Driven or Core Logic–Driven PLL, the clock input of the CLKDLY macro can be sourced from the GmA*, GmB*, or GmC* pin locations.

For CCCs with clock inputs configured with the External I/O–Driven option, the clock input pad can be assigned to any regular I/O location (IO******** pins). Note that since global I/O pins can also be used as regular I/Os, regardless of CCC function (CLKDLY or PLL), clock inputs can also be placed in any of these I/O locations.

By default, the Designer layout engine will place global nets in the design at one of the six chip globals. When the number of globals in the design is greater than six, the Designer layout engine will automatically assign additional globals to the quadrant global networks of the low power flash devices. If the user wishes to decide which global signals should be assigned to chip globals (six available) and which to the quadrant globals (three per quadrant for a total of 12 available), the assignment can be achieved with PinEditor, ChipPlanner, or by importing a placement constraint file. Layout will fail if the global assignments are not allocated properly. See the "Physical Constraints for Quadrant Clocks" section for information on assigning global signals to the quadrant clock networks.

Promoted global signals will be instantiated with CLKINT macros to drive these signals onto the global network. This is automatically done by Designer when the Auto-Promotion option is selected. If the user wishes to assign the signals to the quadrant globals instead of the default chip globals, this can done by using ChipPlanner, by declaring a physical design constraint (PDC), or by importing a PDC file.

Physical Constraints for Quadrant Clocks

If it is necessary to promote global clocks (CLKBUF, CLKINT, PLL, CLKDLY) to quadrant clocks, the user can define PDCs to execute the promotion. PDCs can be created using PDC commands (pre-compile) or the MultiView Navigator (MVN) interface (post-compile). The advantage of using the PDC flow over the MVN flow is that the Compile stage is able to automatically promote any regular net to a global net before assigning it to a quadrant. There are three options to place a quadrant clock using PDC commands:

- Place a clock core (not hardwired to an I/O) into a quadrant clock location.
- Place a clock core (hardwired to an I/O) into an I/O location (set_io) or an I/O module location (set_location) that drives a quadrant clock location.
- Assign a net driven by a regular net or a clock net to a quadrant clock using the following command:

  `assign_local_clock -net <net name> -type quadrant <quadrant clock region>`

  where

  `<net name>` is the name of the net assigned to the local user clock region.

  `<quadrant clock region>` defines which quadrant the net should be assigned to. Quadrant clock regions are defined as UL (upper left), UR (upper right), LL (lower left), and LR (lower right).

Note: If the net is a regular net, the software inserts a CLKINT buffer on the net.

For example:

```
assign_local_clock -net localReset -type quadrant UR
```

Keep in mind the following when placing quadrant clocks using MultiView Navigator:

Hardwired I/O–Driven CCCs

- Find the associated clock input port under the Ports tab, and place the input port at one of the Gmn* locations using PinEditor or I/O Attribute Editor, as shown in Figure 4-32.

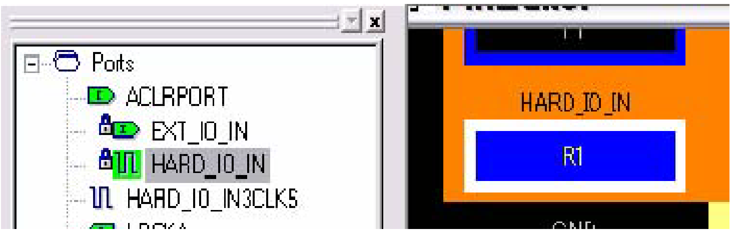
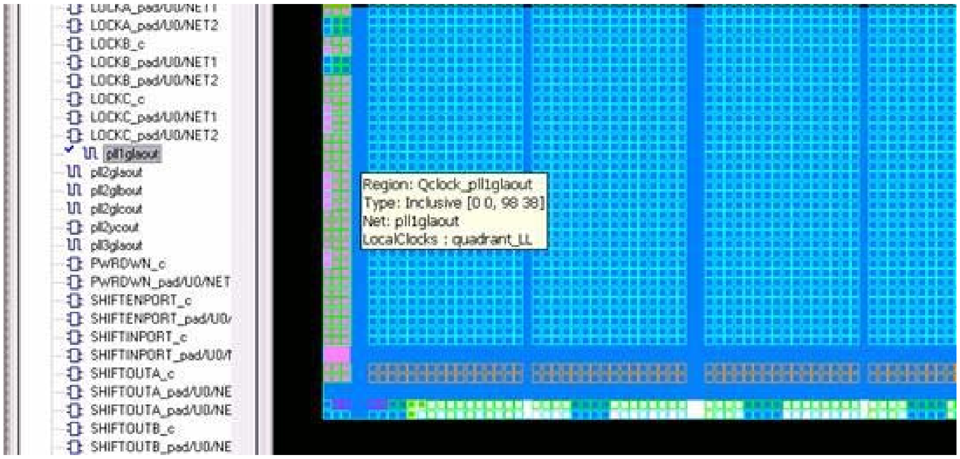


Figure 4-32 **• Port Assignment for a CCC with Hardwired I/O Clock Input**

- Use quadrant global region assignments by finding the clock net associated with the CCC macro under the Nets tab and creating a quadrant global region for the net, as shown in Figure 4-33.



```
LOCKA_pad/U0/NET1
LOCKA_pad/U0/NET2
LOCKB_c
LOCKB_pad/U0/NET1
LOCKB_pad/U0/NET2
LOCKC_c
LOCKC_pad/U0/NET1
LOCKC_pad/U0/NET2
pll1glaout
pll2glaout
pll2glbout
pll2glcout
pll2ycout
pll3glaout
PWRDWN_c
PWRDWN_pad/U0/NET
SHIFTENPORT_c
SHIFTENPORT_pad/U0/
SHIFTINPORT_c
SHIFTINPORT_pad/U0/
SHIFTOUTA_c
SHIFTOUTA_pad/U0/NE
SHIFTOUTA_pad/U0/NE
SHIFTOUTB_c
SHIFTOUTB_pad/U0/NE
```

```
Region: Qclock_pll1glaout
Type: Inclusive [0 0, 98 38]
Net: pll1glaout
LocalClocks : quadrant_LL
```

Figure 4-33 • **Quadrant Clock Assignment for a Global Net**

External I/O–Driven CCCs

The above-mentioned recommendation for proper layout techniques will ensure the correct assignment. It is possible that, especially with External I/O–Driven CCC macros, placement of the CCC macro in a desired location may not be achieved. For example, assigning an input port of an External I/O–Driven CCC near a particular CCC location does not guarantee global assignments to the desired location. This is because the clock inputs of External I/O–Driven CCCs can be assigned to any I/O location; therefore, it is possible that the CCC connected to the clock input will be routed to a location other than the one closest to the I/O location, depending on resource availability and placement constraints.

Clock Placer

The clock placer is a placement engine for low power flash devices that places global signals on the chip global and quadrant global networks. Based on the clock assignment constraints for the chip global and quadrant global clocks, it will try to satisfy all constraints, as well as creating quadrant clock regions when necessary. If the clock placer fails to create the quadrant clock regions for the global signals, it will report an error and stop Layout.

The user must ensure that the constraints set to promote clock signals to quadrant global networks are valid.

Cascading CCCs

The CCCs in low power flash devices can be cascaded. Cascading CCCs can help achieve more accurate PLL output frequency results than those achievable with a single CCC. In addition, this technique is useful when the user application requires the output clock of the PLL to be a multiple of the reference clock by an integer greater than the maximum feedback divider value of the PLL (divide by 128) to achieve the desired frequency.

For example, the user application may require a 280 MHz output clock using a 2 MHz input reference clock, as shown in Figure 4-34 on page 112.

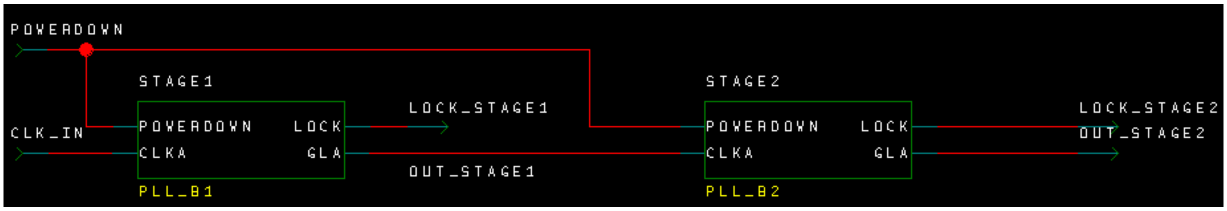


***Figure 4-34* • Cascade PLL Configuration**

Using internal feedback, we know from EQ 4-1 on page 88 that the maximum achievable output frequency from the primary output is

$$f_{GLA} = f_{CLKA} \times m / (n \times u) = 2\text{ MHz} \times 128 / (1 \times 1) = 256\text{ MHz}$$

EQ 4-5

Figure 4-35 shows the settings of the initial PLL. When configuring the initial PLL, specify the input to be either Hardwired I/O–Driven or External I/O–Driven. This generates a netlist with the initial PLL routed from an I/O. Do not specify the input to be Core Logic–Driven, as this prohibits the connection from the I/O pin to the input of the PLL.

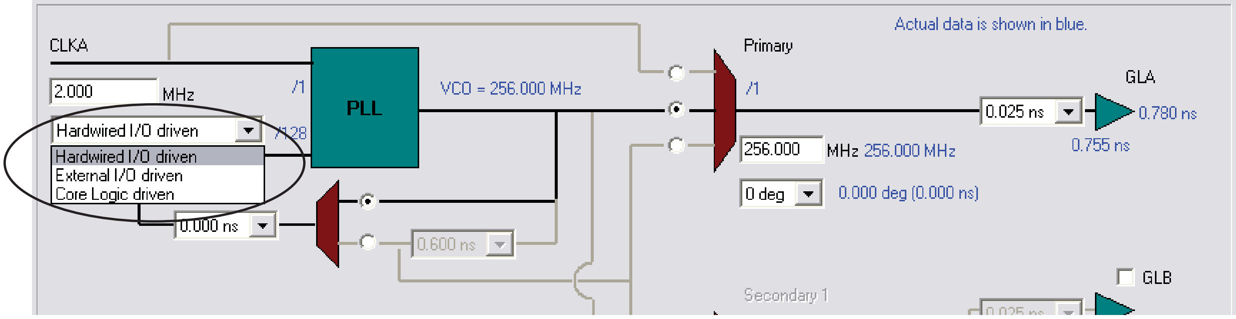


***Figure 4-35* • First-Stage PLL Showing Input of 2 MHz and Output of 256 MHz**

A second PLL can be connected serially to achieve the required frequency. EQ 4-1 on page 88 to EQ 4-3 on page 88 are extended as follows:

$$f_{GLA2} = f_{GLA} \times m_2 / (n_2 \times u_2) = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times u_1 \times n_2 \times u_2) \text{ – Primary PLL Output Clock}$$

EQ 4-6

$$f_{GLB2} = f_{YB2} = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times n_2 \times v_1 \times v_2) \text{ – Secondary 1 PLL Output Clock(s)}$$

EQ 4-7

$$f_{GLC2} = f_{YC2} = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times n_2 \times w_1 \times w2) \text{ – Secondary 2 PLL Output Clock(s)}$$

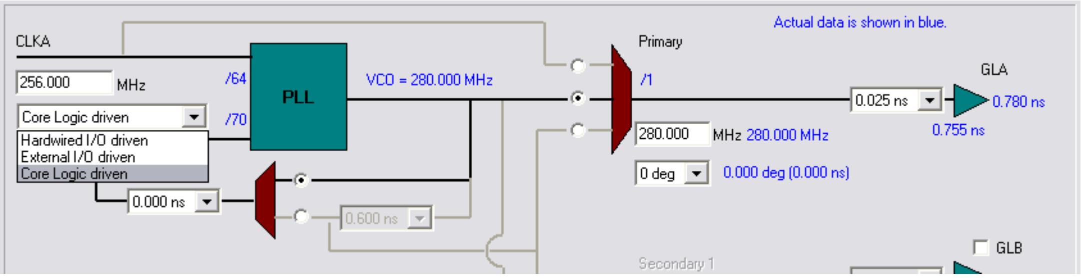
EQ 4-8

In the example, the final output frequency (f_{output}) from the primary output of the second PLL will be as follows (EQ 4-9):

$$f_{output} = f_{GLA2} = f_{GLA} \times m_2 / (n_2 \times u_2) = 256\text{ MHz} \times 70 / (64 \times 1) = 280\text{ MHz}$$

EQ 4-9

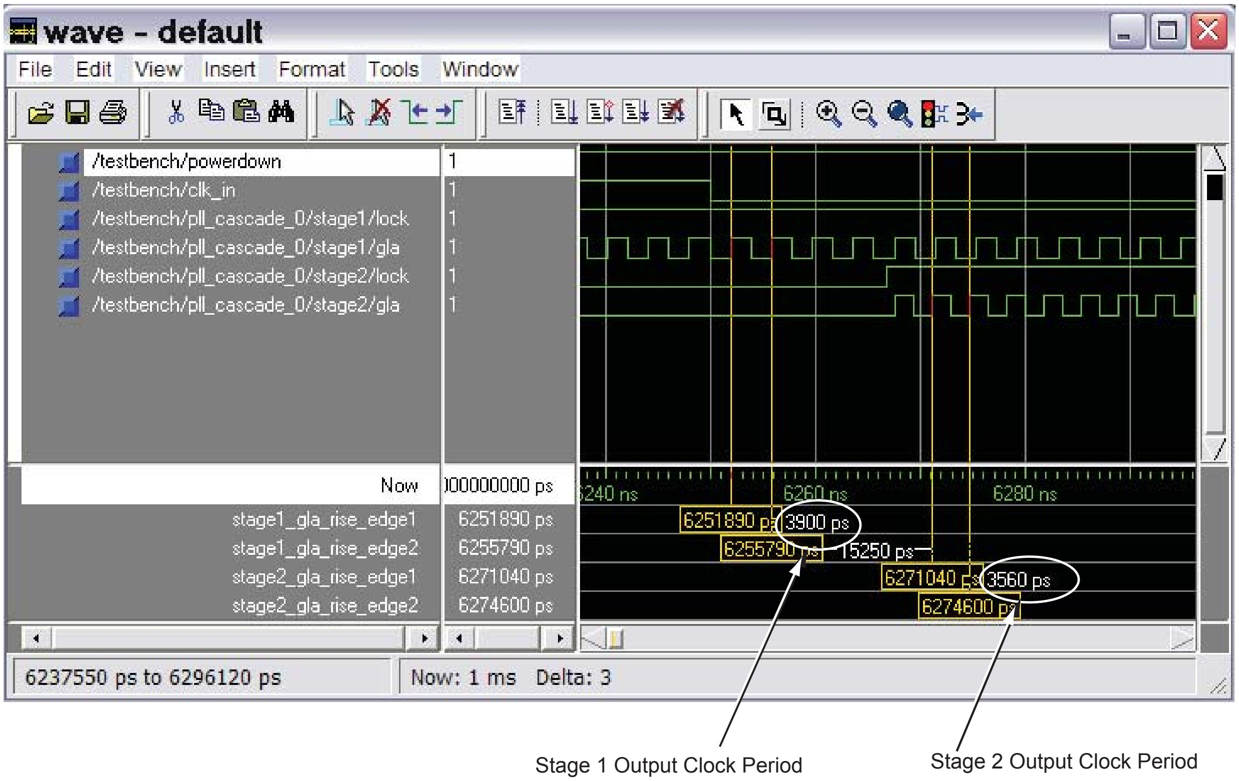
Figure 4-36 on page 113 shows the settings of the second PLL. When configuring the second PLL (or any subsequent-stage PLLs), specify the input to be Core Logic–Driven. This generates a netlist with the second PLL routed internally from the core. Do not specify the input to be Hardwired I/O–Driven or External I/O–Driven, as these options prohibit the connection from the output of the first PLL to the input of the second PLL.



CLKA
256.000 MHz
Core Logic driven
Hardwired I/O driven
External I/O driven
Core Logic driven
0.000 ns
/64
/70
PLL
VCO = 280.000 MHz
0.600 ns
Actual data is shown in blue.
Primary
/1
280.000 MHz 280.000 MHz
0 deg 0.000 deg (0.000 ns)
0.025 ns
GLA
0.780 ns
0.755 ns
□ GLB
Secondary 1

Figure 4-36 • **Second-Stage PLL Showing Input of 256 MHz from First Stage and Final Output of 280 MHz**

Figure 4-37 shows the simulation results, where the first PLL’s output period is 3.9 ns (~256 MHz), and the stage 2 (final) output period is 3.56 ns (~280 MHz).



wave - default

```
/testbench/powerdown                  1
/testbench/clk_in                     1
/testbench/pll_cascade_0/stage1/lock  1
/testbench/pll_cascade_0/stage1/gla   1
/testbench/pll_cascade_0/stage2/lock  1
/testbench/pll_cascade_0/stage2/gla   1

Now                 )00000000 ps
stage1_gla_rise_edge1  6251890 ps
stage1_gla_rise_edge2  6255790 ps
stage2_gla_rise_edge1  6271040 ps
stage2_gla_rise_edge2  6274600 ps

6237550 ps to 6296120 ps    Now: 1 ms  Delta: 3
```

Figure 4-37 • **Model*Sim* Simulation Results**

Recommended Board-Level Considerations

The power to the PLL core is supplied by VCCPLA/B/C/D/E/F (VCCPLx), and the associated ground connections are supplied by VCOMPLA/B/C/D/E/F (VCOMPLx). When the PLLs are not used, the Designer place-and-route tool automatically disables the unused PLLs to lower power consumption. The user should tie unused VCCPLx and VCOMPLx pins to ground. Optionally, the PLL can be turned on/off during normal device operation via the POWERDOWN port (see Table 4-3 on page 70).

PLL Power Supply Decoupling Scheme

The PLL core is designed to tolerate noise levels on the PLL power supply as specified in the datasheets. When operated within the noise limits, the PLL will meet the output peak-to-peak jitter specifications specified in the datasheets. User applications should always ensure the PLL power supply is powered from a noise-free or low-noise power source.

However, in situations where the PLL power supply noise level is higher than the tolerable limits, various decoupling schemes can be designed to suppress noise to the PLL power supply. An example is provided in Figure 4-38. The VCCPLx and VCOMPLx pins correspond to the PLL analog power supply and ground.

Microsemi strongly recommends that two ceramic capacitors (10 nF in parallel with 100 nF) be placed close to the power pins (less than 1 inch away). A third generic 10 µF electrolytic capacitor is recommended for low-frequency noise and should be placed farther away due to its large physical size. Microsemi recommends that a 6.8 µH inductor be placed between the supply source and the capacitors to filter out any low-/medium- and high-frequency noise. In addition, the PCB layers should be controlled so the VCCPLx and VCOMPLx planes have the minimum separation possible, thus generating a good-quality RF capacitor.

For more recommendations, refer to the *Board-Level Considerations* application note.

Recommended 100 nF capacitor:

- Producer BC Components, type X7R, 100 nF, 16 V
- BC Components part number: 0603B104K160BT
- Digi-Key part number: BC1254CT-ND
- Digi-Key part number: BC1254TR-ND

Recommended 10 nF capacitor:

- Surface-mount ceramic capacitor
- Producer BC Components, type X7R, 10 nF, 50 V
- BC Components part number: 0603B103K500BT
- Digi-Key part number: BC1252CT-ND
- Digi-Key part number: BC1252TR-ND

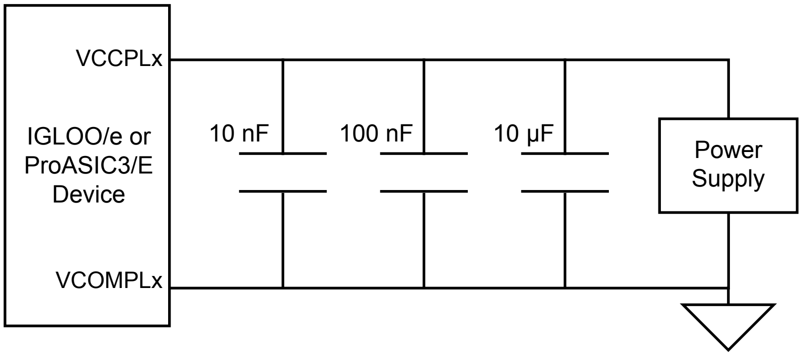


***Figure 4-38* • Decoupling Scheme for One PLL (should be replicated for each PLL used)**

Conclusion

The advanced CCCs of the IGLOO and ProASIC3 devices are ideal for applications requiring precise clock management. They integrate easily with the internal low-skew clock networks and provide flexible frequency synthesis, clock deskewing, and/or time-shifting operations.

Related Documents

Application Notes

Board-Level Considerations

http://www.microsemi.com/soc/documents/ALL_AC276_AN.pdf

Datasheets

Fusion Family of Mixed Signal FPGAs

http://www.microsemi.com/soc/documents/Fusion_DS.pdf

User's Guides

IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide

http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	The "Implementing EXTFB in ProASIC3/E Devices" section is new (SAR 36647).	72
	Table 4-7 • Delay Values in Libero SoC Software per Device Family was added to the "Clock Delay Adjustment" section (SAR 22709).	88
	The "Phase Adjustment" section was rewritten to explain better why the visual CCC shows both the actual phase and the actual delay that is equivalent to this phase shift (SAR 29647).	89
	The hyperlink for the *Board-Level Considerations* application note was corrected (SAR 36663)	114, 115
December 2011	Figure 4-20 • PLL Block Diagram, Figure 4-22 • CCC Block Control Bits – Graphical Representation of Assignments, and Table 4-12 • MUXA, MUXB, MUXC were revised to change the phase shift assignments for PLLs 4 through 7 (SAR 33791).	87, 91, 95
June 2011	The description for RESETEN in Table 4-8 • Configuration Bit Descriptions for the CCC Blocks was revised. The phrase "and should not be modified via dynamic configuration" was deleted because RESETEN is read only (SAR 25949).	92
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A

Date	Changes	Page
v1.4 (December 2008)	The"CCC Support in Microsemi's Flash Devices" section was updated to include IGLOO nano and ProASIC3 nano devices.	65
	Figure 4-2 • CCC Options: Global Buffers with No Programmable Delay was revised to add the CLKBIBUF macro.	66
	The description of the reference clock was revised in Table 4-2 • Input and Output Description of the CLKDLY Macro.	67
	Figure 4-7 • Clock Input Sources (30 k gates devices and below) is new. Figure 4-8 • Clock Input Sources Including CLKBUF, CLKBUF_LVDS/LVPECL, and CLKINT (60 k gates devices and above) applies to 60 k gate devices and above.	74
	The "IGLOO and ProASIC3" section was updated to include information for IGLOO nano devices.	75
	A note regarding Fusion CCCs was added to Figure 4-9 • Illustration of Hardwired I/O (global input pins) Usage for IGLOO and ProASIC3 devices 60 k Gates and Larger and the name of the figure was changed from Figure 4-8 • Illustration of Hardwired I/O (global input pins) Usage. Figure 4-10 • Illustration of Hardwired I/O (global input pins) Usage for IGLOO and ProASIC3 devices 30 k Gates and Smaller is new.	76
	Table 4-5 • Number of CCCs by Device Size and Package was updated to include IGLOO nano and ProASIC3 nano devices. Entries were added to note differences for the CS81, CS121, and CS201 packages.	80
	The "Clock Conditioning Circuits without Integrated PLLs" section was rewritten.	81
	The "IGLOO and ProASIC3 CCC Locations" section was updated for nano devices.	83
	Figure 4-13 • CCC Locations in the 15 k and 30 k Gate Devices was deleted.	4-20
v1.3 (October 2008)	This document was updated to include Fusion and RT ProASIC3 device information. Please review the document very carefully.	N/A
	The "CCC Support in Microsemi's Flash Devices" section was updated.	65
	In the "Global Buffer with Programmable Delay" section, the following sentence was changed from: "In this case, the I/O must be placed in one of the dedicated global I/O locations." To "In this case, the software will automatically place the dedicated global I/O in the appropriate locations."	66
	Figure 4-4 • CCC Options: Global Buffers with PLL was updated to include OADIVRST and OADIVHALF.	69
	In Figure 4-6 • CCC with PLL Block "fixed delay" was changed to "programmable delay".	69
	Table 4-3 • Input and Output Signals of the PLL Block was updated to include OADIVRST and OADIVHALF descriptions.	70
	Table 4-8 • Configuration Bit Descriptions for the CCC Blocks was updated to include configuration bits 88 to 81. Note 2 is new. In addition, the description for bit <76:74> was updated.	92
	Table 4-16 • Fusion Dynamic CCC Clock Source Selection and Table 4-17 • Fusion Dynamic CCC NGMUX Configuration are new.	96
	Table 4-18 • Fusion Dynamic CCC Division by Half Configuration and Table 4-19 • Configuration Bit <76:75> / VCOSEL<2:1> Selection for All Families are new.	97

Date	Changes	Page
v1.2 (June 2008)	The following changes were made to the family descriptions in Figure 4-1 • Overview of the CCCs Offered in Fusion, IGLOO, and ProASIC3: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	63
v1.1 (March 2008)	Table 4-1 • Flash-Based FPGAs and the associated text were updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	65
	The "Global Input Selections" section was updated to include 15 k gate devices as supported I/O types for globals, for CCC only.	73
	Table 4-5 • Number of CCCs by Device Size and Package was revised to include ProASIC3L, IGLOO PLUS, A3P015, AGL015, AGLP030, AGLP060, and AGLP125.	80
	The "IGLOO and ProASIC3 CCC Locations" section was revised to include 15 k gate devices in the exception statements, as they do not contain PLLs.	83
v1.0 (January 2008)	Information about unlocking the PLL was removed from the "Dynamic PLL Configuration" section.	89
	In the "Dynamic PLL Configuration" section, information was added about running Layout and determining the exact setting of the ports.	102
	In Table 4-8 • Configuration Bit Descriptions for the CCC Blocks, the following bits were updated to delete "transport to the user" and reference the footnote at the bottom of the table: 79 to 71.	92

5 – FlashROM in Microsemi’s Low Power Flash Devices

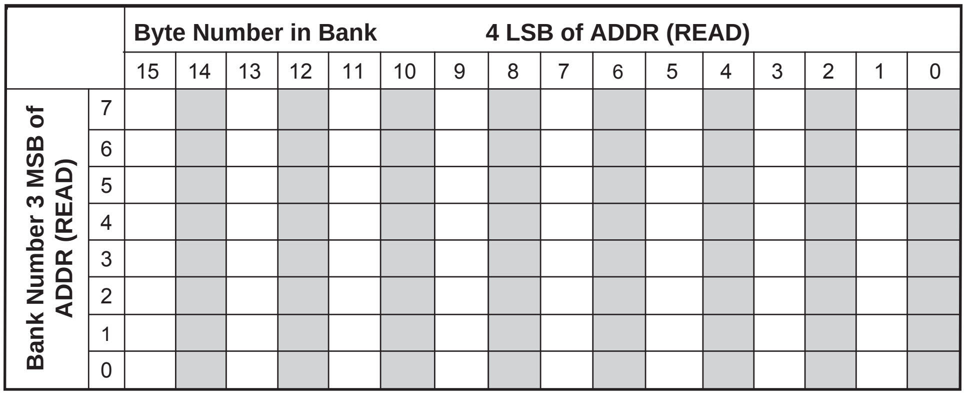
Introduction

The Fusion, IGLOO, and ProASIC3 families of low power flash-based devices have a dedicated nonvolatile FlashROM memory of 1,024 bits, which provides a unique feature in the FPGA market. The FlashROM can be read, modified, and written using the JTAG (or UJTAG) interface. It can be read but not modified from the FPGA core. Only low power flash devices contain on-chip user nonvolatile memory (NVM).

Architecture of User Nonvolatile FlashROM

Low power flash devices have 1 kbit of user-accessible nonvolatile flash memory on-chip that can be read from the FPGA core fabric. The FlashROM is arranged in eight banks of 128 bits (16 bytes) during programming. The 128 bits in each bank are addressable as 16 bytes during the read-back of the FlashROM from the FPGA core. Figure 5-1 shows the FlashROM logical structure.

The FlashROM can only be programmed via the IEEE 1532 JTAG port. It cannot be programmed directly from the FPGA core. When programming, each of the eight 128-bit banks can be selectively reprogrammed. The FlashROM can only be reprogrammed on a bank boundary. Programming involves an automatic, on-chip bank erase prior to reprogramming the bank. The FlashROM supports synchronous read. The address is latched on the rising edge of the clock, and the new output data is stable after the falling edge of the same clock cycle. For more information, refer to the timing diagrams in the DC and Switching Characteristics chapter of the appropriate datasheet. The FlashROM can be read on byte boundaries. The upper three bits of the FlashROM address from the FPGA core define the bank being accessed. The lower four bits of the FlashROM address from the FPGA core define which of the 16 bytes in the bank is being accessed.



		Byte Number in Bank 4 LSB of ADDR (READ)															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Bank Number 3 MSB of ADDR (READ)	7																
	6																
	5																
	4																
	3																
	2																
	1																
	0																

Figure 5-1 • **FlashROM Architecture**

FlashROM Support in Flash-Based Devices

The flash FPGAs listed in Table 5-1 support the FlashROM feature and the functions described in this document.

Table 5-1 **• Flash-Based FPGAs**

Series	Family[*]	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 5-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 5-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

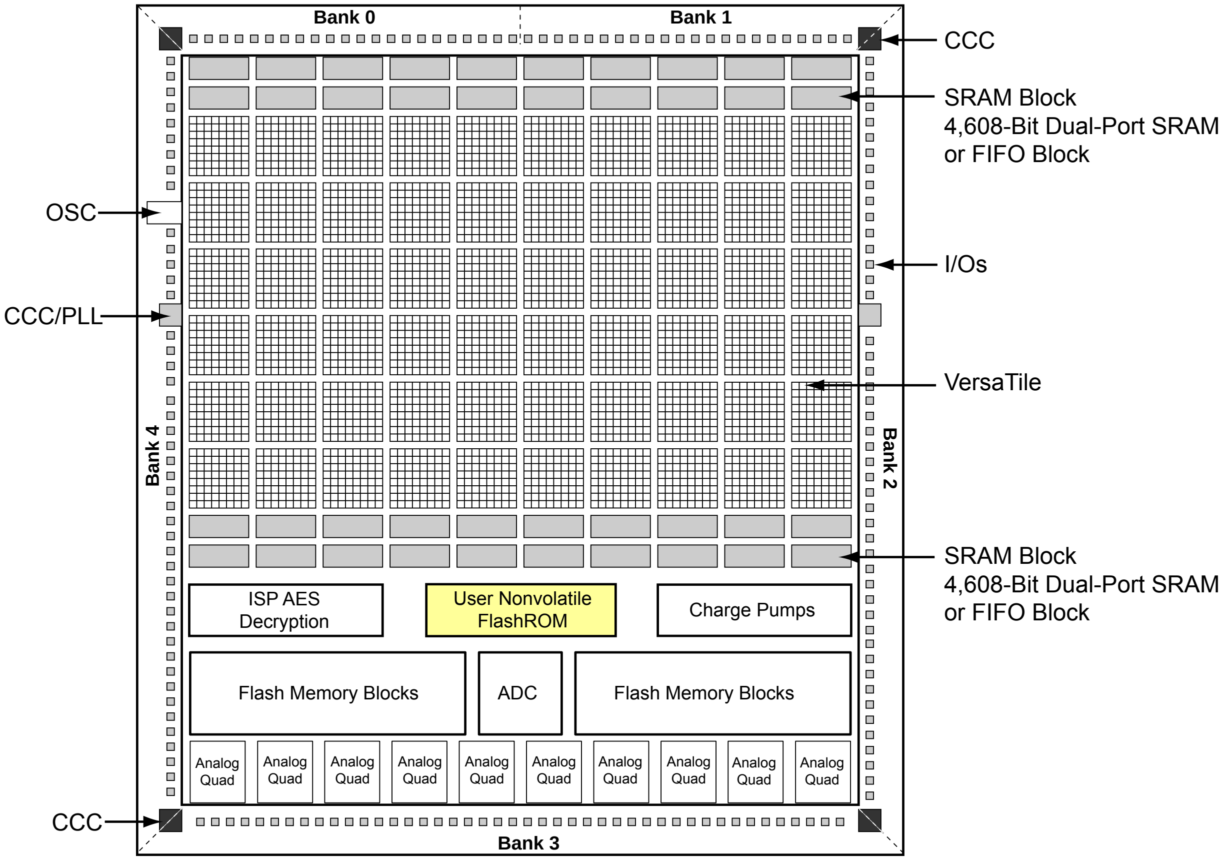


Figure 5-2 • **Fusion Device Architecture Overview (AFS600)**

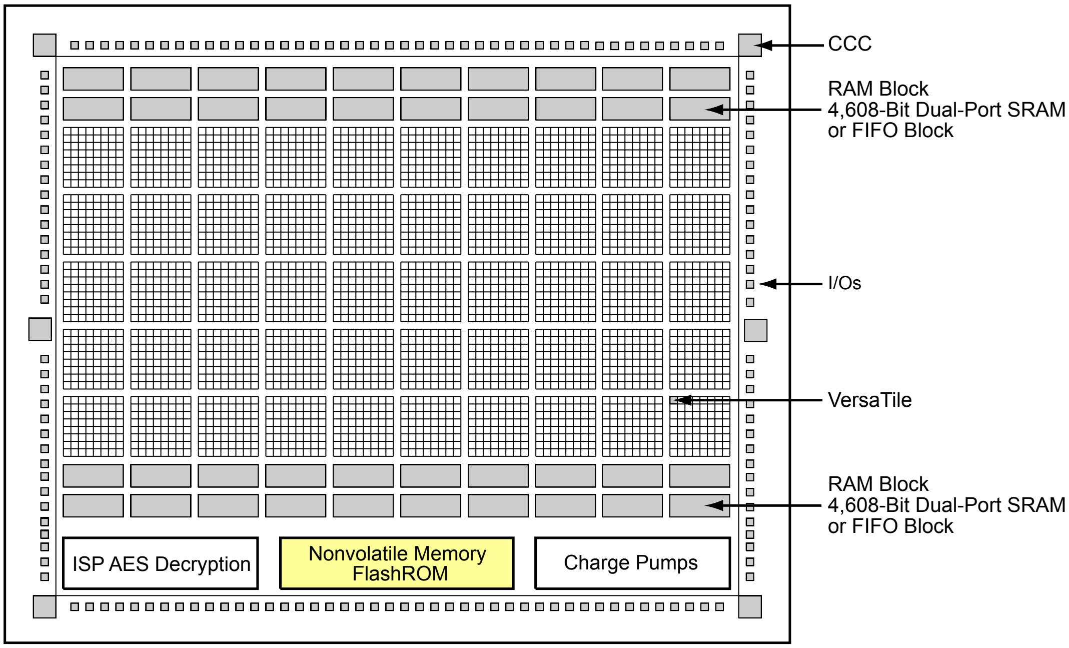


Figure 5-3 • **ProASIC3 and IGLOO Device Architecture**

FlashROM Applications

The SmartGen core generator is used to configure FlashROM content. You can configure each page independently. SmartGen enables you to create and modify regions within a page; these regions can be 1 to 16 bytes long (Figure 5-4).

		Byte Number in Page															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Page Number	7																
	6																
	5																
	4																
	3																
	2																
	1																
	0																

***Figure 5-4* • FlashROM Configuration**

The FlashROM content can be changed independently of the FPGA core content. It can be easily accessed and programmed via JTAG, depending on the security settings of the device. The SmartGen core generator enables each region to be independently updated (described in the "Programming and Accessing FlashROM" section on page 124). This enables you to change the FlashROM content on a per-part basis while keeping some regions "constant" for all parts. These features allow the FlashROM to be used in diverse system applications. Consider the following possible uses of FlashROM:

- Internet protocol (IP) addressing (wireless or fixed)
- System calibration settings
- Restoring configuration after unpredictable system power-down
- Device serialization and/or inventory control
- Subscription-based business models (e.g., set-top boxes)
- Secure key storage
- Asset management tracking
- Date stamping
- Version management

FlashROM Security

Low power flash devices have an on-chip Advanced Encryption Standard (AES) decryption core, combined with an enhanced version of the Microsemi flash-based lock technology (FlashLock®). Together, they provide unmatched levels of security in a programmable logic device. This security applies to both the FPGA core and FlashROM content. These devices use the 128-bit AES (Rijndael) algorithm to encrypt programming files for secure transmission to the on-chip AES decryption core. The same algorithm is then used to decrypt the programming file. This key size provides approximately 3.4 × 10^{38} possible 128-bit keys. A computing system that could find a DES key in a second would take approximately 149 trillion years to crack a 128-bit AES key. The 128-bit FlashLock feature in low power flash devices works via a FlashLock security Pass Key mechanism, where the user locks or unlocks the device with a user-defined key. Refer to the "Security in Low Power Flash Devices" section on page 323.

If the device is locked with certain security settings, functions such as device read, write, and erase are disabled. This unique feature helps to protect against invasive and noninvasive attacks. Without the correct Pass Key, access to the FPGA is denied. To gain access to the FPGA, the device first must be unlocked using the correct Pass Key. During programming of the FlashROM or the FPGA core, you can generate the security header programming file, which is used to program the AES key and/or FlashLock Pass Key. The security header programming file can also be generated independently of the FlashROM and FPGA core content. The FlashLock Pass Key is not stored in the FlashROM.

Low power flash devices with AES-based security allow for secure remote field updates over public networks such as the Internet, and ensure that valuable intellectual property (IP) remains out of the hands of IP thieves. Figure 5-5 shows this flow diagram.

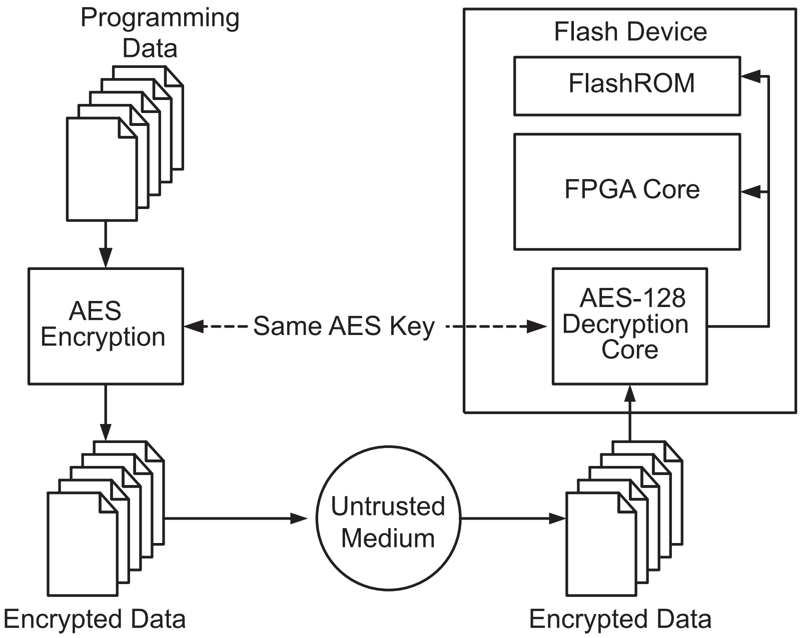


***Figure 5-5* • Programming FlashROM Using AES**

Programming and Accessing FlashROM

The FlashROM content can only be programmed via JTAG, but it can be read back selectively through the JTAG programming interface, the UJTAG interface, or via direct FPGA core addressing. The pages of the FlashROM can be made secure to prevent read-back via JTAG. In that case, read-back on these secured pages is only possible by the FPGA core fabric or via UJTAG.

A 7-bit address from the FPGA core defines which of the eight pages (three MSBs) is being read, and which of the 16 bytes within the selected page (four LSBs) are being read. The FlashROM content can be read on a random basis; the access time is 10 ns for a device supporting commercial specifications. The FPGA core will be powered down during writing of the FlashROM content. FPGA power-down during FlashROM programming is managed on-chip, and FPGA core functionality is not available during programming of the FlashROM. Table 5-2 summarizes various FlashROM access scenarios.

***Table 5-2* • FlashROM Read/Write Capabilities by Access Mode**

Access Mode	FlashROM Read	FlashROM Write
JTAG	Yes	Yes
UJTAG	Yes	No
FPGA core	Yes	No

Figure 5-6 shows the accessing of the FlashROM using the UJTAG macro. This is similar to FPGA core access, where the 7-bit address defines which of the eight pages (three MSBs) is being read and which of the 16 bytes within the selected page (four LSBs) are being read. Refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 377 for details on using the UJTAG macro to read the FlashROM.

Figure 5-7 on page 125 and Figure 5-8 on page 125 show the FlashROM access from the JTAG port. The FlashROM content can be read on a random basis. The three-bit address defines which page is being read or updated.

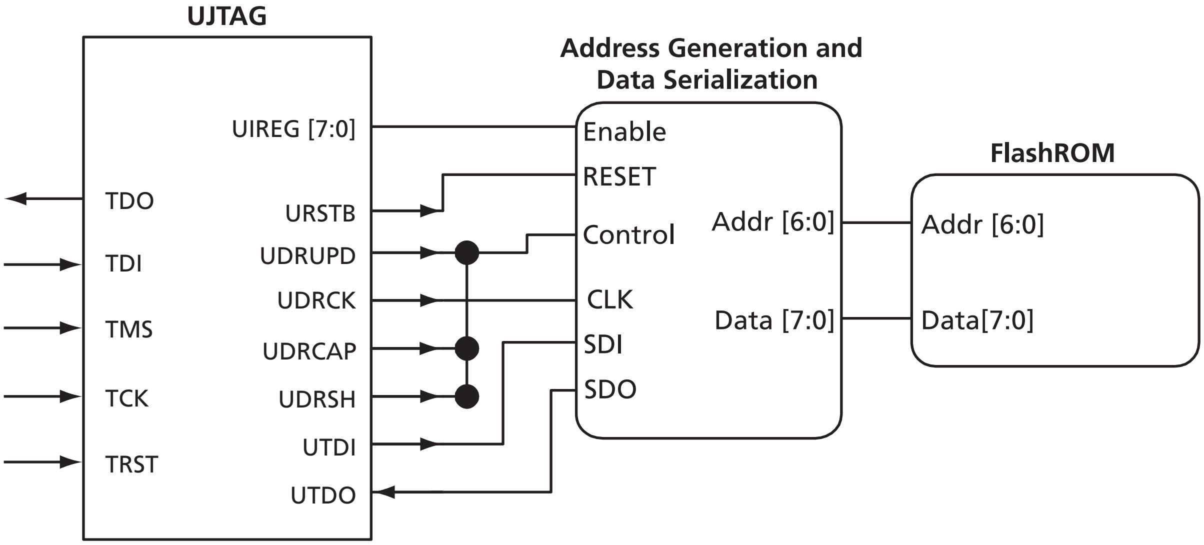


***Figure 5-6* • Block Diagram of Using UJTAG to Read FlashROM Contents**

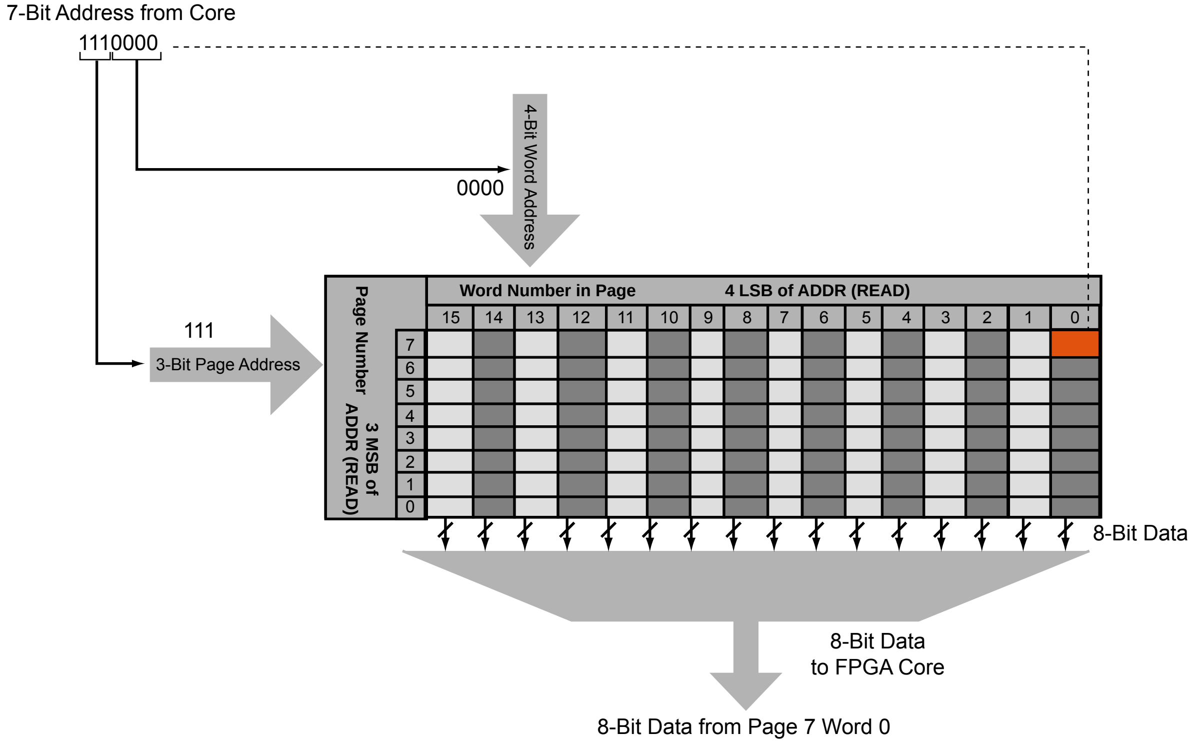


Figure 5-7 **• Accessing FlashROM Using FPGA Core**

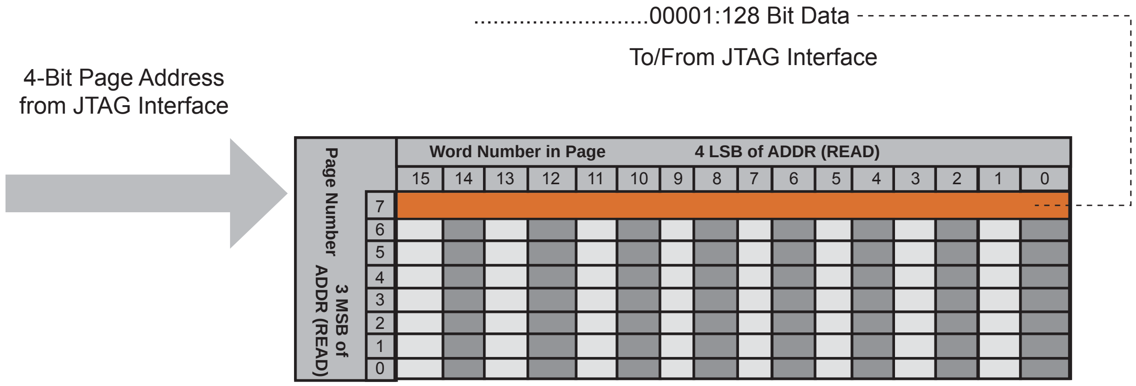


Figure 5-8 **• Accessing FlashROM Using JTAG Port**

FlashROM Design Flow

The Microsemi Libero System-on-Chip (SoC) software has extensive FlashROM support, including FlashROM generation, instantiation, simulation, and programming. Figure 5-9 shows the user flow diagram. In the design flow, there are three main steps:

1. FlashROM generation and instantiation in the design
2. Simulation of FlashROM design
3. Programming file generation for FlashROM design

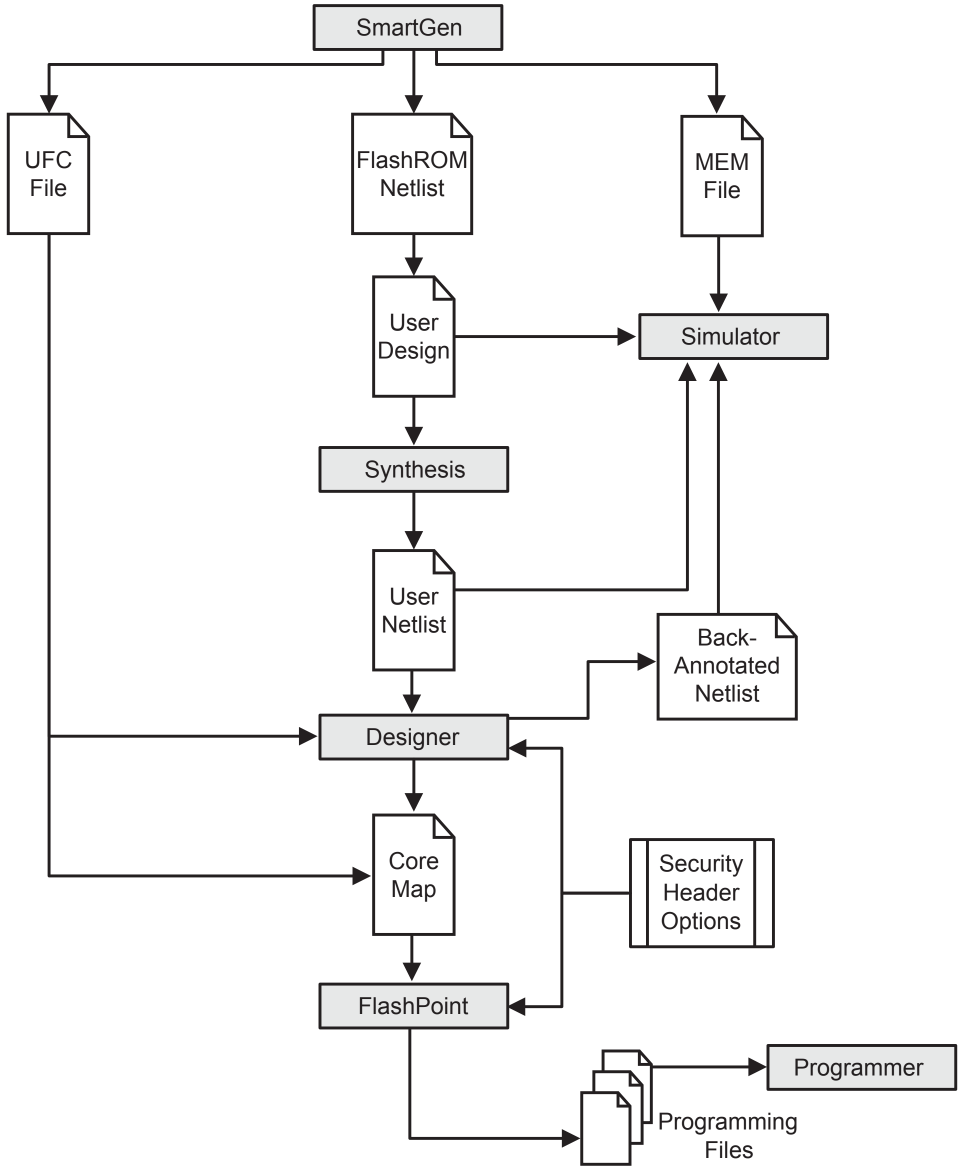
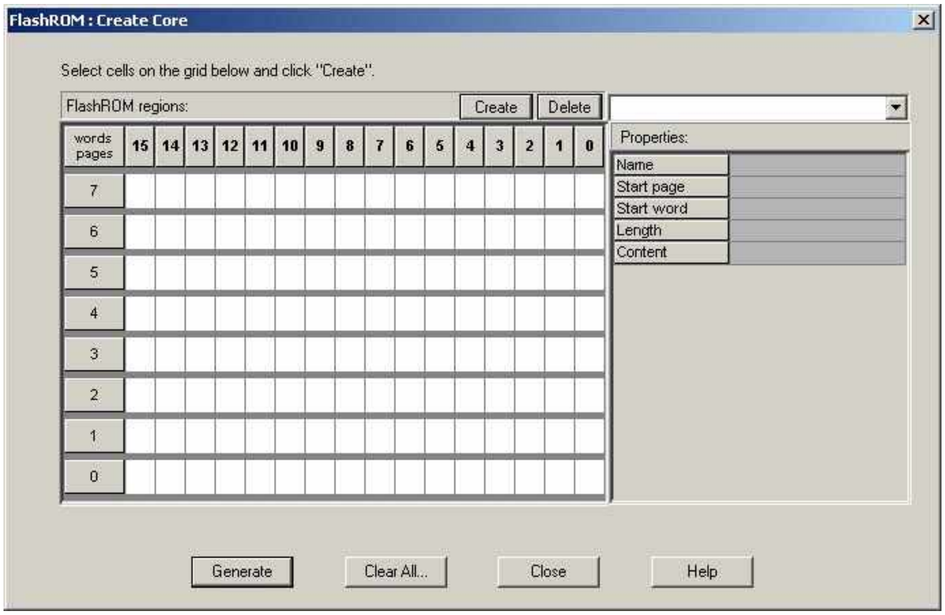


Figure 5-9 • **FlashROM Design Flow**

FlashROM Generation and Instantiation in the Design

The SmartGen core generator, available in Libero SoC and Designer, is the only tool that can be used to generate the FlashROM content. SmartGen has several user-friendly features to help generate the FlashROM contents. Instead of selecting each byte and assigning values, you can create a region within a page, modify the region, and assign properties to that region. The FlashROM user interface, shown in Figure 5-10, includes the configuration grid, existing regions list, and properties field. The properties field specifies the region-specific information and defines the data used for that region. You can assign values to the following properties:

1. Static Fixed Data—Enables you to fix the data so it cannot be changed during programming time. This option is useful when you have fixed data stored in this region, which is required for the operation of the design in the FPGA. Key storage is one example.
2. Static Modifiable Data—Select this option when the data in a particular region is expected to be static data (such as a version number, which remains the same for a long duration but could conceivably change in the future). This option enables you to avoid changing the value every time you enter new data.
3. Read from File—This provides the full flexibility of FlashROM usage to the customer. If you have a customized algorithm for generating the FlashROM data, you can specify this setting. You can then generate a text file with data for as many devices as you wish to program, and load that into the FlashPoint programming file generation software to get programming files that include all the data. SmartGen will optionally pass the location of the file where the data is stored if the file is specified in SmartGen. Each text file has only one type of data format (binary, decimal, hex, or ASCII text). The length of each data file must be shorter than or equal to the selected region length. If the data is shorter than the selected region length, the most significant bits will be padded with 0s. For multiple text files for multiple regions, the first lines are for the first device. In SmartGen, **Load Sim. Value From File** allows you to load the first device data in the MEM file for simulation.
4. Auto Increment/Decrement—This scenario is useful when you specify the contents of FlashROM for a large number of devices in a series. You can specify the step value for the serial number and a maximum value for inventory control. During programming file generation, the actual number of devices to be programmed is specified and a start value is fed to the software.



FlashROM : Create Core

Select cells on the grid below and click "Create".

Figure 5-10 **• SmartGen GUI of the FlashROM**

SmartGen allows you to generate the FlashROM netlist in VHDL, Verilog, or EDIF format. After the FlashROM netlist is generated, the core can be instantiated in the main design like other SmartGen cores. Note that the macro library name for FlashROM is UFROM. The following is a sample FlashROM VHDL netlist that can be instantiated in the main design:

```
library ieee;
use ieee.std_logic_1164.all;
library fusion;

entity FROM_a is
  port( ADDR : in std_logic_vector(6 downto 0); DOUT : out std_logic_vector(7 downto 0));
end FROM_a;

architecture DEF_ARCH of  FROM_a is

  component UFROM
    generic (MEMORYFILE:string);
    port(DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7 : out std_logic;
      ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, ADDR5, ADDR6 : in std_logic := 'U') ;
  end component;

  component GND
    port( Y : out std_logic);
  end component;

signal U_7_PIN2 : std_logic ;

begin

  GND_1_net : GND port map(Y => U_7_PIN2);
  UFROM0 : UFROM
  generic map(MEMORYFILE => "FROM_a.mem")
  port map(DO0 => DOUT(0), DO1 => DOUT(1), DO2 => DOUT(2), DO3 => DOUT(3), DO4 => DOUT(4),
    DO5 => DOUT(5), DO6 => DOUT(6), DO7 => DOUT(7), ADDR0 => ADDR(0), ADDR1 => ADDR(1),
    ADDR2 => ADDR(2), ADDR3 => ADDR(3), ADDR4 => ADDR(4), ADDR5 => ADDR(5),
    ADDR6 => ADDR(6));

end DEF_ARCH;
```

SmartGen generates the following files along with the netlist. These are located in the SmartGen folder for the Libero SoC project.

1. MEM (Memory Initialization) file
2. UFC (User Flash Configuration) file
3. Log file

The MEM file is used for simulation, as explained in the "Simulation of FlashROM Design" section on page 129. The UFC file, generated by SmartGen, has the FlashROM configuration for single or multiple devices and is used during STAPL generation. It contains the region properties and simulation values. Note that any changes in the MEM file will not be reflected in the UFC file. Do not modify the UFC to change FlashROM content. Instead, use the SmartGen GUI to modify the FlashROM content. See the "Programming File Generation for FlashROM Design" section on page 129 for a description of how the UFC file is used during the programming file generation. The log file has information regarding the file type and file location.

Simulation of FlashROM Design

The MEM file has 128 rows of 8 bits, each representing the contents of the FlashROM used for simulation. For example, the first row represents page 0, byte 0; the next row is page 0, byte 1; and so the pattern continues. Note that the three MSBs of the address define the page number, and the four LSBs define the byte number. So, if you send address 0000100 to FlashROM, this corresponds to the page 0 and byte 4 location, which is the fifth row in the MEM file. SmartGen defaults to 0s for any unspecified location of the FlashROM. Besides using the MEM file generated by SmartGen, you can create a binary file with 128 rows of 8 bits each and use this as a MEM file. Microsemi recommends that you use different file names if you plan to generate multiple MEM files. During simulation, Libero SoC passes the MEM file used as the generic file in the netlist, along with the design files and testbench. If you want to use different MEM files during simulation, you need to modify the generic file reference in the netlist.

```
....................
UFROM0: UFROM
--generic map(MEMORYFILE => "F:\Appsnotes\FROM\test_designs\testa\smartgen\FROM_a.mem")
--generic map(MEMORYFILE => "F:\Appsnotes\FROM\test_designs\testa\smartgen\FROM_b.mem")
.......................
```

The VITAL and Verilog simulation models accept the generics passed by the netlist, read the MEM file, and perform simulation with the data in the file.

Programming File Generation for FlashROM Design

FlashPoint is the programming software used to generate the programming files for flash devices. Depending on the applications, you can use the FlashPoint software to generate a STAPL file with different FlashROM contents. In each case, optional AES decryption is available. To generate a STAPL file that contains the same FPGA core content and different FlashROM contents, the FlashPoint software needs an Array Map file for the core and UFC file(s) for the FlashROM. This final STAPL file represents the combination of the logic of the FPGA core and FlashROM content.

FlashPoint generates the STAPL files you can use to program the desired FlashROM page and/or FPGA core of the FPGA device contents. FlashPoint supports the encryption of the FlashROM content and/or FPGA Array configuration data. In the case of using the FlashROM for device serialization, a sequence of unique FlashROM contents will be generated. When generating a programming file with multiple unique FlashROM contents, you can specify in FlashPoint whether to include all FlashROM content in a single STAPL file or generate a different STAPL file for each FlashROM (Figure 5-11). The programming software (FlashPro) handles the single STAPL file that contains the FlashROM content from multiple devices. It enables you to program the FlashROM content into a series of devices sequentially (Figure 5-11). See the *FlashPro User's Guide* for information on serial programming.

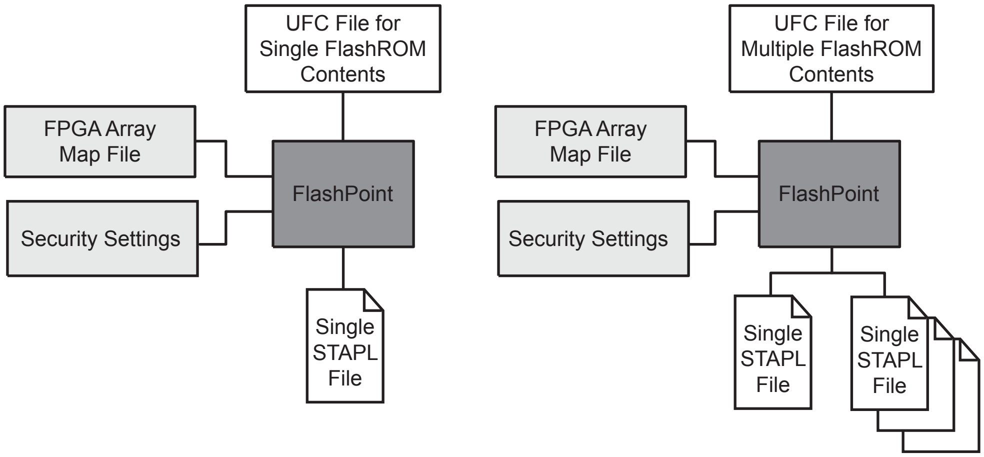
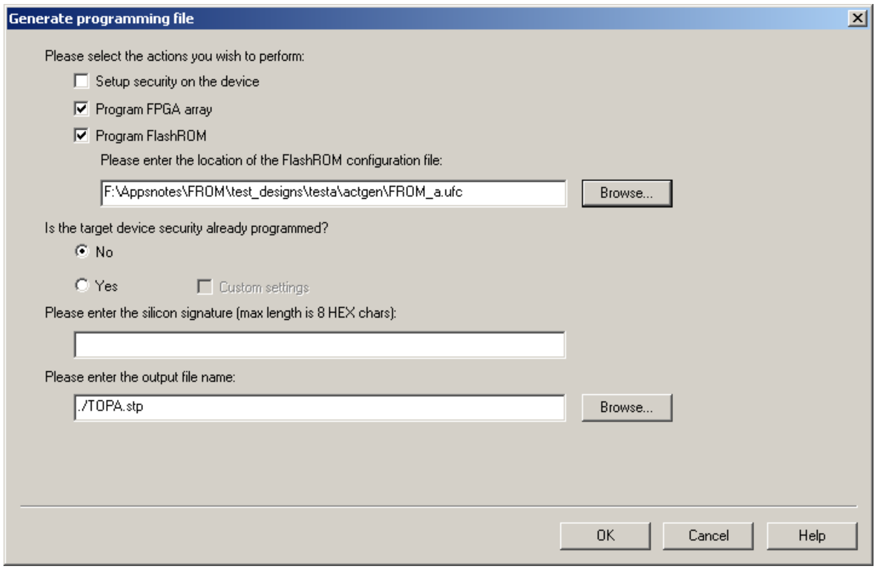


***Figure 5-11* • Single or Multiple Programming File Generation**

Figure 5-12 shows the programming file generator, which enables different STAPL file generation methods. When you select **Program FlashROM** and choose the UFC file, the FlashROM Settings window appears, as shown in Figure 5-13. In this window, you can select the FlashROM page you want to program and the data value for the configured regions. This enables you to use a different page for different programming files.



Generate programming file

Please select the actions you wish to perform:

☐ Setup security on the device

☑ Program FPGA array

☑ Program FlashROM

Please enter the location of the FlashROM configuration file:

F:\Appsnotes\FROM\test_designs\testa\actgen\FROM_a.ufc [Browse...]

Is the target device security already programmed?

● No

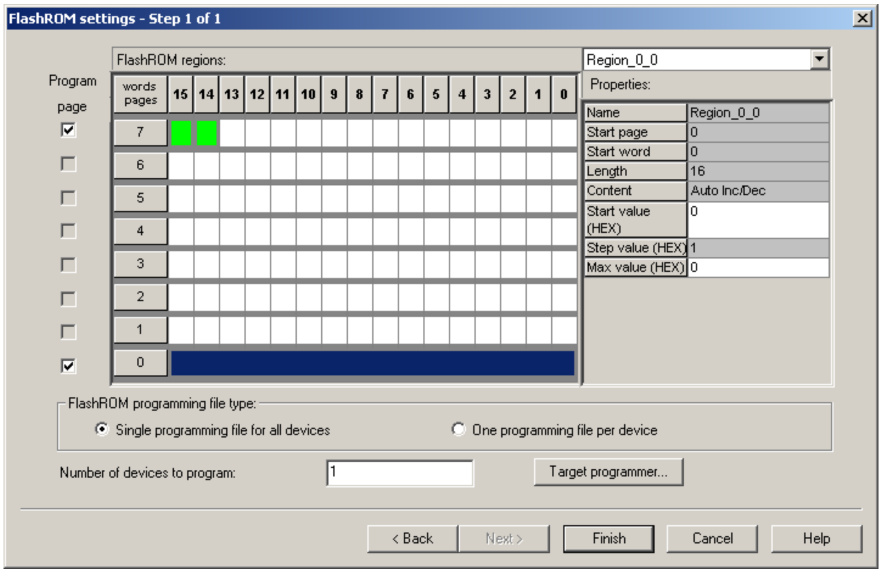
○ Yes ☐ Custom settings

Please enter the silicon signature (max length is 8 HEX chars):

Please enter the output file name:

./TOPA.stp [Browse...]

Figure 5-12 • **Programming File Generator**



FlashROM settings - Step 1 of 1

FlashROM regions:

Program page	words / pages	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
☑	7	🟩	🟩														
☐	6																
☐	5																
☐	4																
☐	3																
☐	2																
☐	1																
☑	0	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦

Region_0_0

Properties:

Name	Region_0_0
Start page	0
Start word	0
Length	16
Content	Auto Inc/Dec
Start value (HEX)	0
Step value (HEX)	1
Max value (HEX)	0

FlashROM programming file type:

● Single programming file for all devices ○ One programming file per device

Number of devices to program: 1 [Target programmer...]

Figure 5-13 • **Setting FlashROM during Programming File Generation**

The programming hardware and software can load the FlashROM with the appropriate STAPL file. Programming software handles the single STAPL file that contains multiple FlashROM contents for multiple devices, and programs the FlashROM in sequential order (e.g., for device serialization). This feature is supported in the programming software. After programming with the STAPL file, you can run DEVICE_INFO to check the FlashROM content.

DEVICE_INFO displays the FlashROM content, serial number, Design Name, and checksum, as shown below:

```
EXPORT IDCODE[32] = 123261CF
EXPORT SILSIG[32] = 00000000
User information :
CHECKSUM: 61A0
Design Name:        TOP
Programming Method: STAPL
Algorithm Version: 1
Programmer: UNKNOWN
=========================================
FlashROM Information :
EXPORT Region_7_0[128] = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
=========================================
Security Setting :
Encrypted FlashROM Programming Enabled.
Encrypted FPGA Array Programming Enabled.
=========================================
```

The Libero SoC file manager recognizes the UFC and MEM files and displays them in the appropriate view. Libero SoC also recognizes the multiple programming files if you choose the option to generate multiple files for multiple FlashROM contents in Designer. These features enable a user-friendly flow for the FlashROM generation and programming in Libero SoC.

Custom Serialization Using FlashROM

You can use FlashROM for device serialization or inventory control by using the Auto Inc region or Read From File region. FlashPoint will automatically generate the serial number sequence for the Auto Inc region with the **Start Value**, **Max Value**, and **Step Value** provided. If you have a unique serial number generation scheme that you prefer, the Read From File region allows you to import the file with your serial number scheme programmed into the region. See the *FlashPro User's Guide* for custom serialization file format information.

The following steps describe how to perform device serialization or inventory control using FlashROM:

1. Generate FlashROM using SmartGen. From the Properties section in the FlashROM Settings dialog box, select the **Auto Inc** or **Read From File** region. For the Auto Inc region, specify the desired step value. You will not be able to modify this value in the FlashPoint software.
2. Go through the regular design flow and finish place-and-route.
3. Select **Programming File in Designer** and open **Generate Programming File** (Figure 5-12 on page 130).
4. Click **Program FlashROM**, browse to the UFC file, and click **Next**. The FlashROM Settings window appears, as shown in Figure 5-13 on page 130.
5. Select the FlashROM page you want to program and the data value for the configured regions. The STAPL file generated will contain only the data that targets the selected FlashROM page.
6. Modify properties for the serialization.
   – For the Auto Inc region, specify the **Start** and **Max** values.
   – For the Read From File region, select the file name of the custom serialization file.
7. Select the FlashROM programming file type you want to generate from the two options below:
   – Single programming file for all devices: generates one programming file with all FlashROM values.
   – One programming file per device: generates a separate programming file for each FlashROM value.
8. Enter the number of devices you want to program and generate the required programming file.
9. Open the programming software and load the programming file. The programming software, FlashPro3 and Silicon Sculptor II, supports the device serialization feature. If, for some reason, the device fails to program a part during serialization, the software allows you to reuse or skip the serial data. Refer to the *FlashPro User's Guide* for details.

Conclusion

The Fusion, IGLOO, and ProASIC3 families are the only FPGAs that offer on-chip FlashROM support. This document presents information on the FlashROM architecture, possible applications, programming, access through the JTAG and UJTAG interface, and integration into your design. In addition, the Libero tool set enables easy creation and modification of the FlashROM content.

The nonvolatile FlashROM block in the FPGA can be customized, enabling multiple applications.

Additionally, the security offered by the low power flash devices keeps both the contents of FlashROM and the FPGA design safe from system over-builders, system cloners, and IP thieves.

Related Documents

User's Guides

FlashPro User's Guide

http://www.microsemi.com/documents/FlashPro_UG.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 5-1 • Flash-Based FPGAs.	120
v1.3 (October 2008)	The "FlashROM Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	120
	Figure 5-2 • Fusion Device Architecture Overview (AFS600) was replaced. Figure 5-5 • Programming FlashROM Using AES was revised to change "Fusion" to "Flash Device."	121, 123
	The *FlashPoint User's Guide* was removed from the "User's Guides" section, as its content is now part of the *FlashPro User's Guide*.	132
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 5-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	120
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	N/A

6 – SRAM and FIFO Memories in Microsemi's Low Power Flash Devices

Introduction

As design complexity grows, greater demands are placed upon an FPGA's embedded memory. Fusion, IGLOO, and ProASIC3 devices provide the flexibility of true dual-port and two-port SRAM blocks. The embedded memory, along with built-in, dedicated FIFO control logic, can be used to create cascading RAM blocks and FIFOs without using additional logic gates.

IGLOO, IGLOO PLUS, and ProASIC3L FPGAs contain an additional feature that allows the device to be put in a low power mode called Flash*Freeze. In this mode, the core draws minimal power (on the order of 2 to 127 µW) and still retains values on the embedded SRAM/FIFO and registers. Flash*Freeze technology allows the user to switch to Active mode on demand, thus simplifying power management and the use of SRAM/FIFOs.

Device Architecture

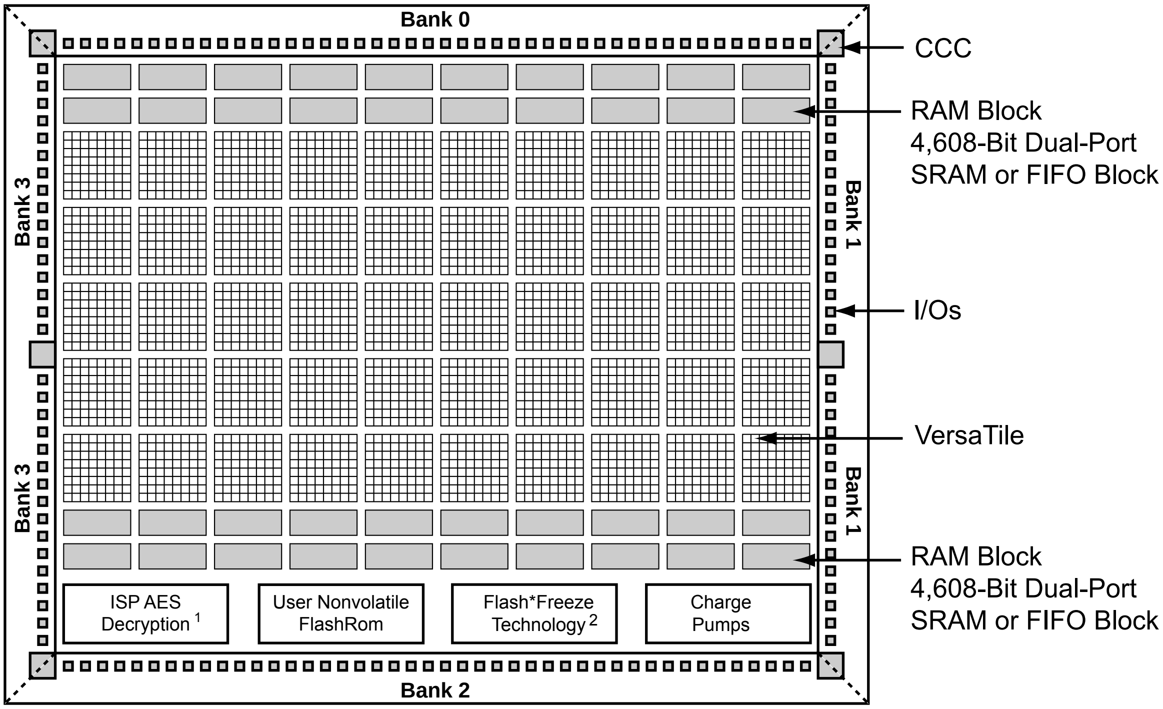
The low power flash devices feature up to 504 kbits of RAM in 4,608-bit blocks (Figure 6-1 on page 134 and Figure 6-2 on page 135). The total embedded SRAM for each device can be found in the datasheets. These memory blocks are arranged along the top and bottom of the device to allow better access from the core and I/O (in some devices, they are only available on the north side of the device). Every RAM block has a flexible, hardwired, embedded FIFO controller, enabling the user to implement efficient FIFOs without sacrificing user gates.

In the IGLOO and ProASIC3 families of devices, the following memories are supported:

- 30 k gate devices and smaller do not support SRAM and FIFO.
- 60 k and 125 k gate devices support memories on the north side of the device only.
- 250 k devices and larger support memories on the north and south sides of the device.

In Fusion devices, the following memories are supported:

- AFS090 and AFS250 support memories on the north side of the device only.
- AFS600 and AFS1500 support memories on the north and south sides of the device.



Notes:

1. *AES decryption not supported in 30 k gate devices and smaller.*
2. *Flash*Freeze is supported in all IGLOO devices and the ProASIC3L devices.*

Figure 6-1 **• IGLOO and ProASIC3 Device Architecture Overview**

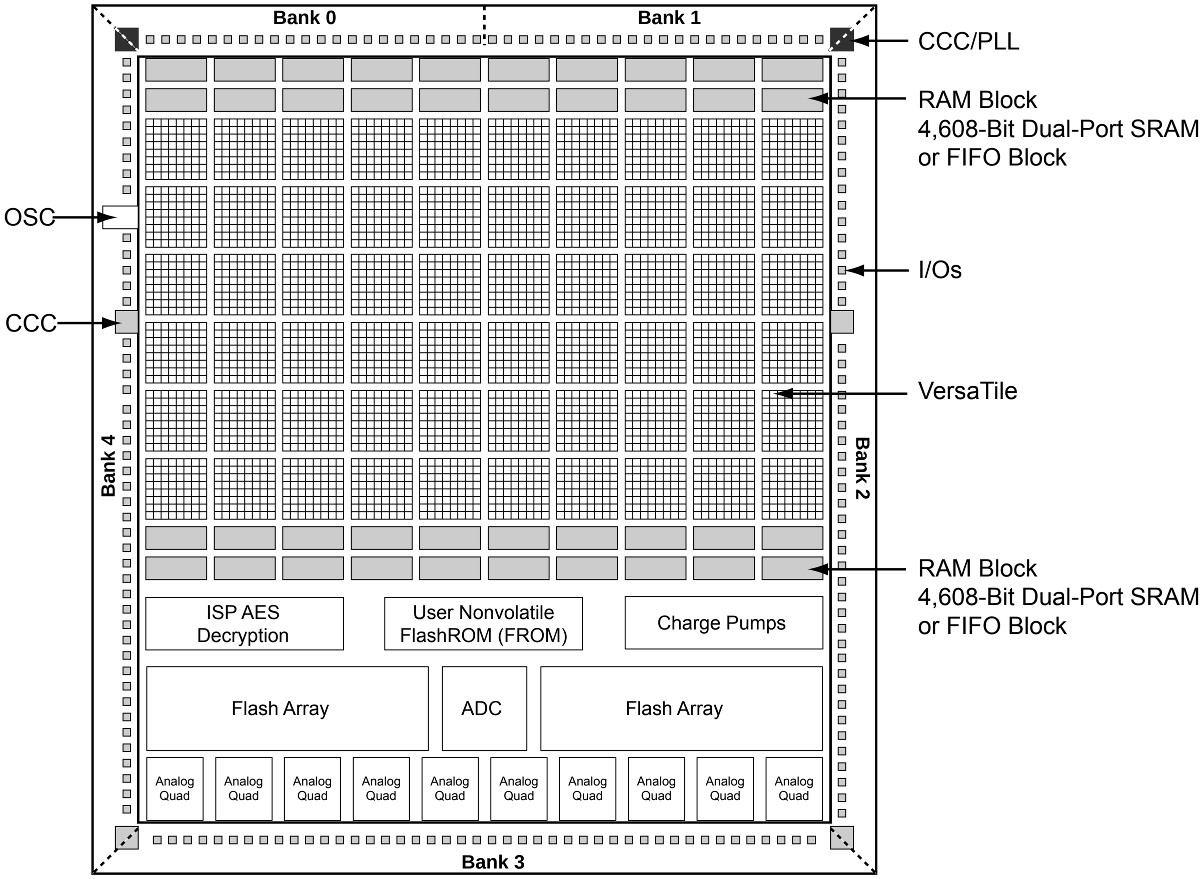


Figure 6-2 • **Fusion Device Architecture Overview (AFS600)**

SRAM/FIFO Support in Flash-Based Devices

The flash FPGAs listed in Table 6-1 support SRAM and FIFO blocks and the functions described in this document.

***Table 6-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 6-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 6-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

SRAM and FIFO Architecture

To meet the needs of high-performance designs, the memory blocks operate strictly in synchronous mode for both read and write operations. The read and write clocks are completely independent, and each can operate at any desired frequency up to 250 MHz.

- 4k×1, 2k×2, 1k×4, 512×9 (dual-port RAM—2 read / 2 write or 1 read / 1 write)
- 512×9, 256×18 (2-port RAM—1 read / 1 write)
- Sync write, sync pipelined / nonpipelined read

Automotive ProASIC3 devices support single-port SRAM capabilities or dual-port SRAM only under specific conditions. Dual-port mode is supported if the clocks to the two SRAM ports are the same and 180° out of phase (i.e., the port A clock is the inverse of the port B clock). The Libero SoC software macro libraries support a dual-port macro only. For use of this macro as a single-port SRAM, the inputs and clock of one port should be tied off (grounded) to prevent errors during design compile. For use in dual-port mode, the same clock with an inversion between the two clock pins of the macro should be used in the design to prevent errors during compile.

The memory block includes dedicated FIFO control logic to generate internal addresses and external flag logic (FULL, EMPTY, AFULL, AEMPTY).

Simultaneous dual-port read/write and write/write operations at the same address are allowed when certain timing requirements are met.

During RAM operation, addresses are sourced by the user logic, and the FIFO controller is ignored. In FIFO mode, the internal addresses are generated by the FIFO controller and routed to the RAM array by internal MUXes.

The low power flash device architecture enables the read and write sizes of RAMs to be organized independently, allowing for bus conversion. For example, the write size can be set to 256×18 and the read size to 512×9.

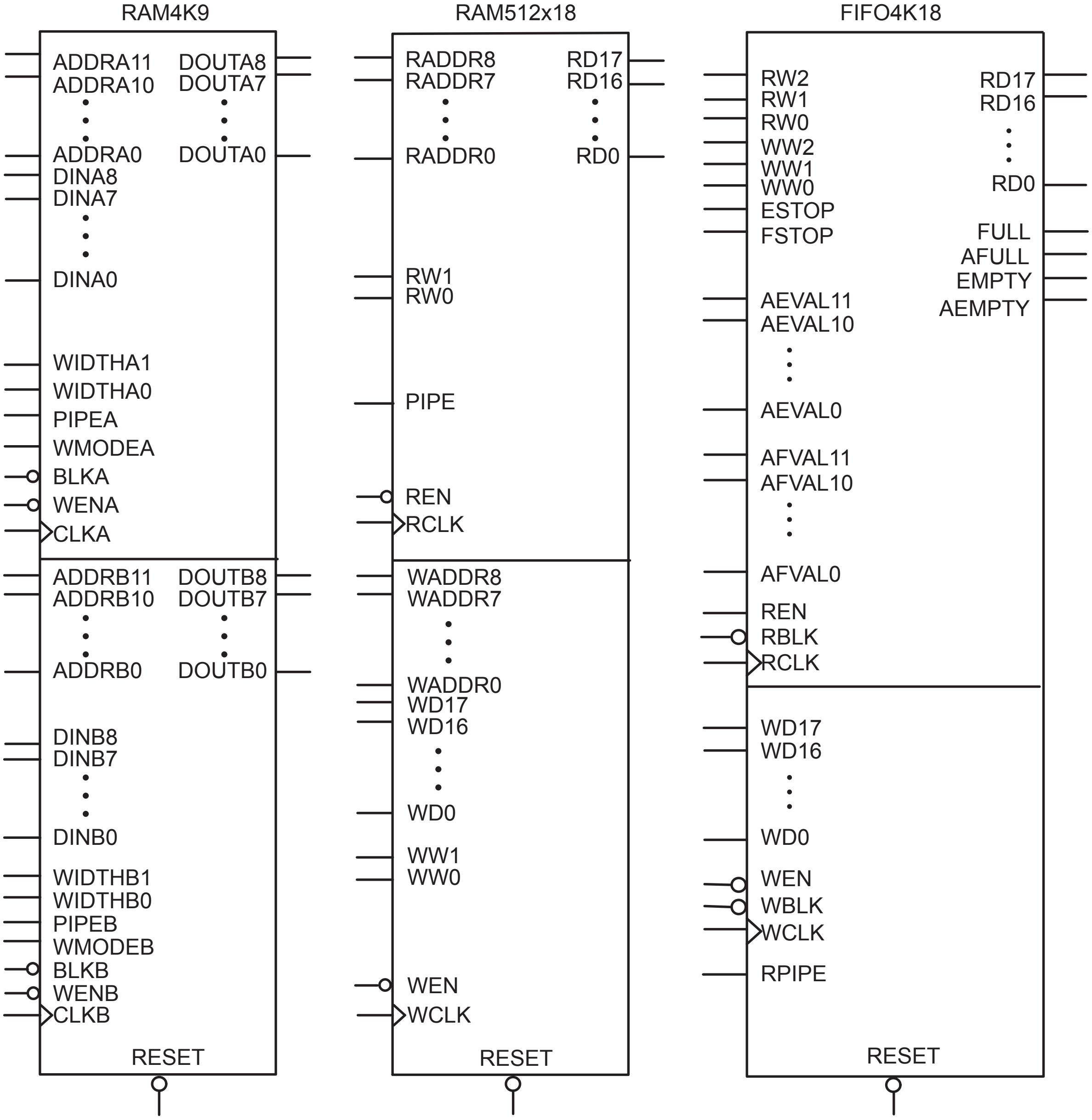
Both the write width and read width for the RAM blocks can be specified independently with the WW (write width) and RW (read width) pins. The different D×W configurations are 256×18, 512×9, 1k×4, 2k×2, and 4k×1. When widths of one, two, or four are selected, the ninth bit is unused. For example, when writing nine-bit values and reading four-bit values, only the first four bits and the second four bits of each nine-bit value are addressable for read operations. The ninth bit is not accessible.

Conversely, when writing four-bit values and reading nine-bit values, the ninth bit of a read operation will be undefined. The RAM blocks employ little-endian byte order for read and write operations.

Memory Blocks and Macros

Memory blocks can be configured with many different aspect ratios, but are generically supported in the macro libraries as one of two memory elements: RAM4K9 or RAM512X18. The RAM4K9 is configured as a true dual-port memory block, and the RAM512X18 is configured as a two-port memory block. Dual-port memory allows the RAM to both read from and write to either port independently. Two-port memory allows the RAM to read from one port and write to the other using a common clock or independent read and write clocks. If needed, the RAM4K9 blocks can be configured as two-port memory blocks. The memory block can be configured as a FIFO by combining the basic memory block with dedicated FIFO controller logic. The FIFO macro is named FIFO4KX18 (Figure 6-3 on page 138).

Clocks for the RAM blocks can be driven by the VersaNet (global resources) or by regular nets. When using local clock segments, the clock segment region that encompasses the RAM blocks can drive the RAMs. In the dual-port configuration (RAM4K9), each memory block port can be driven by either rising-edge or falling-edge clocks. Each port can be driven by clocks with different edges. Though only a rising-edge clock can drive the physical block itself, the Microsemi Designer software will automatically bubble-push the inversion to properly implement the falling-edge trigger for the RAM block.



Notes:

1. *Automotive ProASIC3 devices restrict RAM4K9 to a single port or to dual ports with the same clock 180° out of phase (inverted) between clock pins. In single-port mode, inputs to port B should be tied to ground to prevent errors during compile. This warning applies only to automotive ProASIC3 parts of certain revisions and earlier. Contact Technical Support at soc_tech@microsemi.com for information on the revision number for a particular lot and date code.*
2. *For FIFO4K18, the same clock 180° out of phase (inverted) between clock pins should be used.*

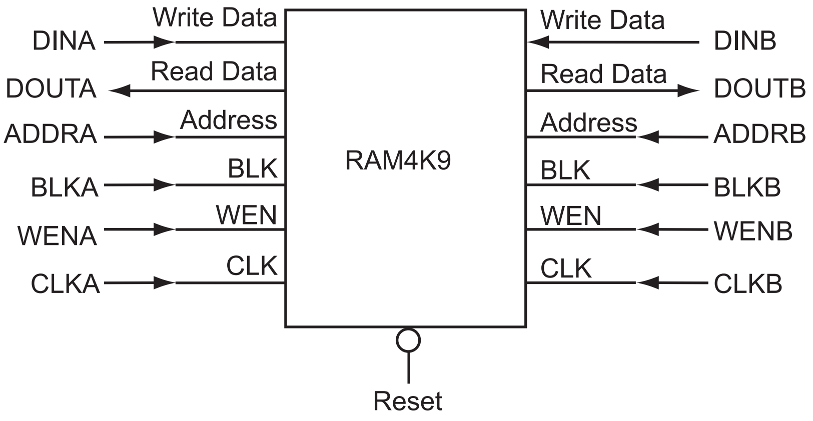
***Figure 6-3* • Supported Basic RAM Macros**

SRAM Features

RAM4K9 Macro

RAM4K9 is the dual-port configuration of the RAM block (Figure 6-4). The RAM4K9 nomenclature refers to both the deepest possible configuration and the widest possible configuration the dual-port RAM block can assume, and does not denote a possible memory aspect ratio. The RAM block can be configured to the following aspect ratios: 4,096×1, 2,048×2, 1,024×4, and 512×9. RAM4K9 is fully synchronous and has the following features:

- Two ports that allow fully independent reads and writes at different frequencies
- Selectable pipelined or nonpipelined read
- Active-low block enables for each port
- Toggle control between read and write mode for each port
- Active-low asynchronous reset
- Pass-through write data or hold existing data on output. In pass-through mode, the data written to the write port will immediately appear on the read port.
- Designer software will automatically facilitate falling-edge clocks by bubble-pushing the inversion to previous stages.



Note: For timing diagrams of the RAM signals, refer to the appropriate family datasheet.

***Figure 6-4* • RAM4K9 Simplified Configuration**

Signal Descriptions for RAM4K9

Note: Automotive ProASIC3 devices support single-port SRAM capabilities, or dual-port SRAM only under specific conditions. Dual-port mode is supported if the clocks to the two SRAM ports are the same and 180° out of phase (i.e., the port A clock is the inverse of the port B clock). Since Libero SoC macro libraries support a dual-port macro only, certain modifications must be made. These are detailed below.

The following signals are used to configure the RAM4K9 memory element:

WIDTHA and WIDTHB

These signals enable the RAM to be configured in one of four allowable aspect ratios (Table 6-2 on page 140).

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, WIDTHB should be tied to ground.

Table 6-2 • **Allowable Aspect Ratio Settings for WIDTHA[1:0]**

WIDTHA[1:0]	WIDTHB[1:0]	D×W
00	00	4k×1
01	01	2k×2
10	10	1k×4
11	11	512×9

Note: The aspect ratio settings are constant and cannot be changed on the fly.

BLKA and BLKB

These signals are active-low and will enable the respective ports when asserted. When a BLKx signal is deasserted, that port's outputs hold the previous value.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, BLKB should be tied to ground.

WENA and WENB

These signals switch the RAM between read and write modes for the respective ports. A LOW on these signals indicates a write operation, and a HIGH indicates a read.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, WENB should be tied to ground.

CLKA and CLKB

These are the clock signals for the synchronous read and write operations. These can be driven independently or with the same driver.

Note: For Automotive ProASIC3 devices, dual-port mode is supported if the clocks to the two SRAM ports are the same and 180° out of phase (i.e., the port A clock is the inverse of the port B clock). For use of this macro as a single-port SRAM, the inputs and clock of one port should be tied off (grounded) to prevent errors during design compile.

PIPEA and PIPEB

These signals are used to specify pipelined read on the output. A LOW on PIPEA or PIPEB indicates a nonpipelined read, and the data appears on the corresponding output in the same clock cycle. A HIGH indicates a pipelined read, and data appears on the corresponding output in the next clock cycle.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, PIPEB should be tied to ground. For use in dual-port mode, the same clock with an inversion between the two clock pins of the macro should be used in the design to prevent errors during compile.

WMODEA and WMODEB

These signals are used to configure the behavior of the output when the RAM is in write mode. A LOW on these signals makes the output retain data from the previous read. A HIGH indicates pass-through behavior, wherein the data being written will appear immediately on the output. This signal is overridden when the RAM is being read.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, WMODEB should be tied to ground.

RESET

This active-low signal resets the control logic, forces the output hold state registers to zero, disables reads and writes from the SRAM block, and clears the data hold registers when asserted. It does not reset the contents of the memory array.

While the RESET signal is active, read and write operations are disabled. As with any asynchronous reset signal, care must be taken not to assert it too close to the edges of active read and write clocks.

ADDRA and ADDRB

These are used as read or write addresses, and they are 12 bits wide. When a depth of less than 4 k is specified, the unused high-order bits must be grounded (Table 6-3 on page 141).

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, ADDRB should be tied to ground.

***Table 6-3* • Address Pins Unused/Used for Various Supported Bus Widths**

D×W	ADDRx	
	Unused	**Used**
4k×1	None	[11:0]
2k×2	[11]	[10:0]
1k×4	[11:10]	[9:0]
512×9	[11:9]	[8:0]

Note: The "x" in ADDRx implies A or B.

DINA and DINB

These are the input data signals, and they are nine bits wide. Not all nine bits are valid in all configurations. When a data width less than nine is specified, unused high-order signals must be grounded (Table 6-4).

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, DINB should be tied to ground.

DOUTA and DOUTB

These are the nine-bit output data signals. Not all nine bits are valid in all configurations. As with DINA and DINB, high-order bits may not be used (Table 6-4). The output data on unused pins is undefined.

***Table 6-4* • Unused/Used Input and Output Data Pins for Various Supported Bus Widths**

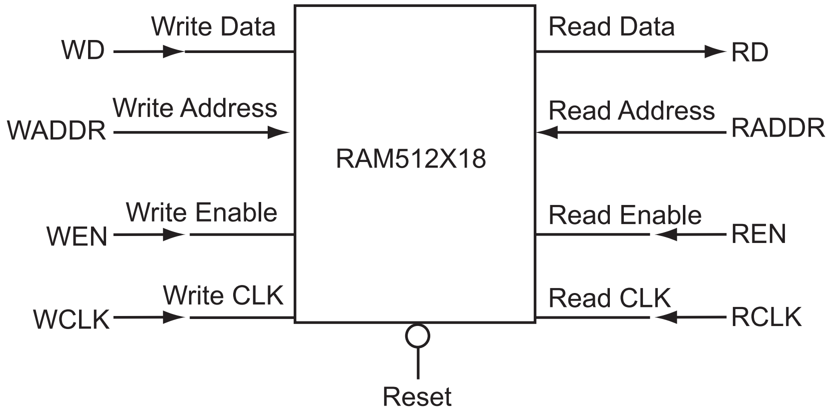
D×W	DINx/DOUTx	
	Unused	**Used**
4k×1	[8:1]	[0]
2k×2	[8:2]	[1:0]
1k×4	[8:4]	[3:0]
512×9	None	[8:0]

Note: The "x" in DINx or DOUTx implies A or B.

RAM512X18 Macro

RAM512X18 is the two-port configuration of the same RAM block (Figure 6-5 on page 142). Like the RAM4K9 nomenclature, the RAM512X18 nomenclature refers to both the deepest possible configuration and the widest possible configuration the two-port RAM block can assume. In two-port mode, the RAM block can be configured to either the 512×9 aspect ratio or the 256×18 aspect ratio. RAM512X18 is also fully synchronous and has the following features:

- Dedicated read and write ports
- Active-low read and write enables
- Selectable pipelined or nonpipelined read
- Active-low asynchronous reset
- Designer software will automatically facilitate falling-edge clocks by bubble-pushing the inversion to previous stages.



Note: For timing diagrams of the RAM signals, refer to the appropriate family datasheet.

***Figure 6-5* • 512X18 Two-Port RAM Block Diagram**

Signal Descriptions for RAM512X18

RAM512X18 has slightly different behavior from RAM4K9, as it has dedicated read and write ports.

WW and RW

These signals enable the RAM to be configured in one of the two allowable aspect ratios (Table 6-5).

***Table 6-5* • Aspect Ratio Settings for WW[1:0]**

WW[1:0]	RW[1:0]	D×W
01	01	512×9
10	10	256×18
00, 11	00, 11	Reserved

WD and RD

These are the input and output data signals, and they are 18 bits wide. When a 512×9 aspect ratio is used for write, WD[17:9] are unused and must be grounded. If this aspect ratio is used for read, RD[17:9] are undefined.

WADDR and RADDR

These are read and write addresses, and they are nine bits wide. When the 256×18 aspect ratio is used for write or read, WADDR[8] and RADDR[8] are unused and must be grounded.

WCLK and RCLK

These signals are the write and read clocks, respectively. They can be clocked on the rising or falling edge of WCLK and RCLK.

WEN and REN

These signals are the write and read enables, respectively. They are both active-low by default. These signals can be configured as active-high.

RESET

This active-low signal resets the control logic, forces the output hold state registers to zero, disables reads and writes from the SRAM block, and clears the data hold registers when asserted. It does not reset the contents of the memory array.

While the RESET signal is active, read and write operations are disabled. As with any asynchronous reset signal, care must be taken not to assert it too close to the edges of active read and write clocks.

PIPE

This signal is used to specify pipelined read on the output. A LOW on PIPE indicates a nonpipelined read, and the data appears on the output in the same clock cycle. A HIGH indicates a pipelined read, and data appears on the output in the next clock cycle.

SRAM Usage

The following descriptions refer to the usage of both RAM4K9 and RAM512X18.

Clocking

The dual-port SRAM blocks are only clocked on the rising edge. SmartGen allows falling-edge-triggered clocks by adding inverters to the netlist, hence achieving dual-port SRAM blocks that are clocked on either edge (rising or falling). For dual-port SRAM, each port can be clocked on either edge and by separate clocks by port. Note that for Automotive ProASIC3, the same clock, with an inversion between the two clock pins of the macro, should be used in design to prevent errors during compile.

Low power flash devices support inversion (bubble-pushing) throughout the FPGA architecture, including the clock input to the SRAM modules. Inversions added to the SRAM clock pin on the design schematic or in the HDL code will be automatically accounted for during design compile without incurring additional delay in the clock path.

The two-port SRAM can be clocked on the rising or falling edge of WCLK and RCLK.

If negative-edge RAM and FIFO clocking is selected for memory macros, clock edge inversion management (bubble-pushing) is automatically used within the development tools, without performance penalty.

Modes of Operation

There are two read modes and one write mode:

- Read Nonpipelined (synchronous—1 clock edge): In the standard read mode, new data is driven onto the RD bus in the same clock cycle following RA and REN valid. The read address is registered on the read port clock active edge, and data appears at RD after the RAM access time. Setting PIPE to OFF enables this mode.
- Read Pipelined (synchronous—2 clock edges): The pipelined mode incurs an additional clock delay from address to data but enables operation at a much higher frequency. The read address is registered on the read port active clock edge, and the read data is registered and appears at RD after the second read clock edge. Setting PIPE to ON enables this mode.
- Write (synchronous—1 clock edge): On the write clock active edge, the write data is written into the SRAM at the write address when WEN is HIGH. The setup times of the write address, write enables, and write data are minimal with respect to the write clock.

RAM Initialization

Each SRAM block can be individually initialized on power-up by means of the JTAG port using the UJTAG mechanism. The shift register for a target block can be selected and loaded with the proper bit configuration to enable serial loading. The 4,608 bits of data can be loaded in a single operation.

FIFO Features

The FIFO4KX18 macro is created by merging the RAM block with dedicated FIFO logic (Figure 6-6 on page 144). Since the FIFO logic can only be used in conjunction with the memory block, there is no separate FIFO controller macro. As with the RAM blocks, the FIFO4KX18 nomenclature does not refer to a possible aspect ratio, but rather to the deepest possible data depth and the widest possible data width. FIFO4KX18 can be configured into the following aspect ratios: 4,096×1, 2,048×2, 1,024×4, 512×9, and 256×18. In addition to being fully synchronous, the FIFO4KX18 also has the following features:

- Four FIFO flags: Empty, Full, Almost-Empty, and Almost-Full
- Empty flag is synchronized to the read clock
- Full flag is synchronized to the write clock
- Both Almost-Empty and Almost-Full flags have programmable thresholds
- Active-low asynchronous reset
- Active-low block enable
- Active-low write enable
- Active-high read enable
- Ability to configure the FIFO to either stop counting after the empty or full states are reached or to allow the FIFO counters to continue

- Designer software will automatically facilitate falling-edge clocks by bubble-pushing the inversion to previous stages.

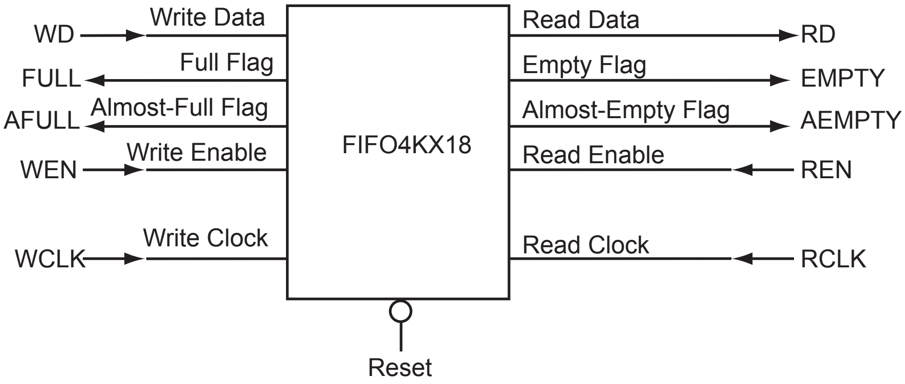


Figure 6-6 **• FIFO4KX18 Block Diagram**

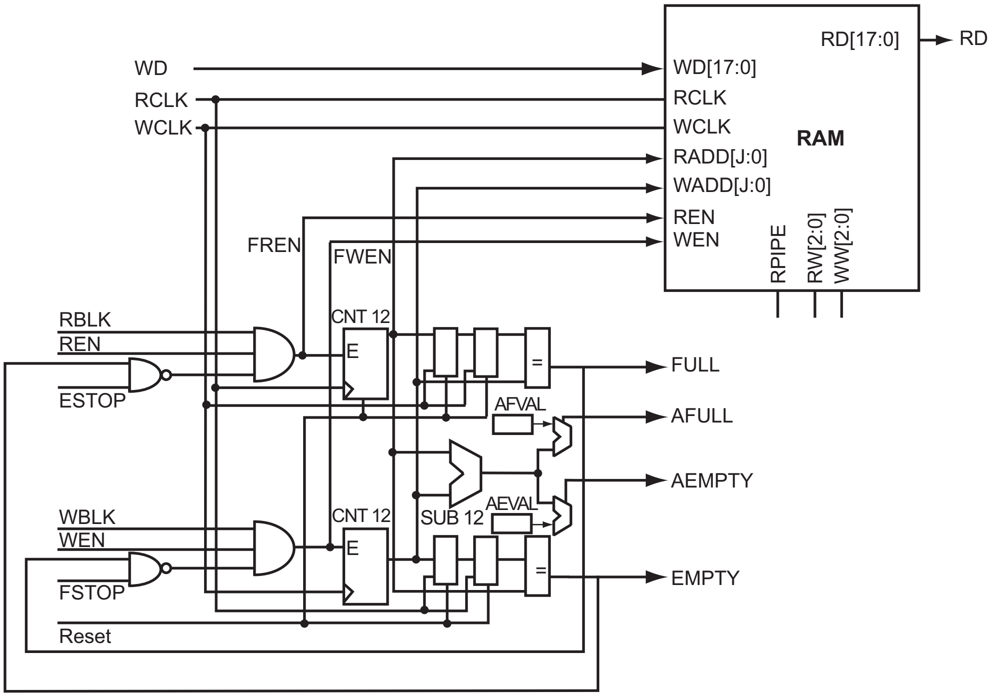


Figure 6-7 **• RAM Block with Embedded FIFO Controller**

The FIFOs maintain a separate read and write address. Whenever the difference between the write address and the read address is greater than or equal to the almost-full value (AFVAL), the Almost-Full flag is asserted. Similarly, the Almost-Empty flag is asserted whenever the difference between the write address and read address is less than or equal to the almost-empty value (AEVAL).

Due to synchronization between the read and write clocks, the Empty flag will deassert after the second read clock edge from the point that the write enable asserts. However, since the Empty flag is synchronized to the read clock, it will assert after the read clock reads the last data in the FIFO. Also, since the Full flag is dependent on the actual hardware configuration, it will assert when the actual physical implementation of the FIFO is full.

For example, when a user configures a 128×18 FIFO, the actual physical implementation will be a 256×18 FIFO element. Since the actual implementation is 256×18, the Full flag will not trigger until the

256×18 FIFO is full, even though a 128×18 FIFO was requested. For this example, the Almost-Full flag can be used instead of the Full flag to signal when the 128th data word is reached.

To accommodate different aspect ratios, the almost-full and almost-empty values are expressed in terms of data bits instead of data words. SmartGen translates the user's input, expressed in data words, into data bits internally. SmartGen allows the user to select the thresholds for the Almost-Empty and Almost-Full flags in terms of either the read data words or the write data words, and makes the appropriate conversions for each flag.

After the empty or full states are reached, the FIFO can be configured so the FIFO counters either stop or continue counting. For timing numbers, refer to the appropriate family datasheet.

Signal Descriptions for FIFO4K18

The following signals are used to configure the FIFO4K18 memory element:

WW and RW

These signals enable the FIFO to be configured in one of the five allowable aspect ratios (Table 6-6).

Table 6-6 **• Aspect Ratio Settings for WW[2:0]**

WW[2:0]	RW[2:0]	D×W
000	000	4k×1
001	001	2k×2
010	010	1k×4
011	011	512×9
100	100	256×18
101, 110, 111	101, 110, 111	Reserved

WBLK and RBLK

These signals are active-low and will enable the respective ports when LOW. When the RBLK signal is HIGH, that port's outputs hold the previous value.

WEN and REN

Read and write enables. WEN is active-low and REN is active-high by default. These signals can be configured as active-high or -low.

WCLK and RCLK

These are the clock signals for the synchronous read and write operations. These can be driven independently or with the same driver.

Note: **For the Automotive ProASIC3 FIFO4K18, for the same clock, 180° out of phase (inverted) between clock pins should be used.**

RPIPE

This signal is used to specify pipelined read on the output. A LOW on RPIPE indicates a nonpipelined read, and the data appears on the output in the same clock cycle. A HIGH indicates a pipelined read, and data appears on the output in the next clock cycle.

RESET

This active-low signal resets the control logic and forces the output hold state registers to zero when asserted. It does not reset the contents of the memory array (Table 6-7 on page 146).

While the RESET signal is active, read and write operations are disabled. As with any asynchronous RESET signal, care must be taken not to assert it too close to the edges of active read and write clocks.

WD

This is the input data bus and is 18 bits wide. Not all 18 bits are valid in all configurations. When a data width less than 18 is specified, unused higher-order signals must be grounded (Table 6-7 on page 146).

RD

This is the output data bus and is 18 bits wide. Not all 18 bits are valid in all configurations. Like the WD bus, high-order bits become unusable if the data width is less than 18. The output data on unused pins is undefined (Table 6-7).

***Table 6-7* • Input Data Signal Usage for Different Aspect Ratios**

D×W	WD/RD Unused
4k×1	WD[17:1], RD[17:1]
2k×2	WD[17:2], RD[17:2]
1k×4	WD[17:4], RD[17:4]
512×9	WD[17:9], RD[17:9]
256×18	–

ESTOP, FSTOP

ESTOP is used to stop the FIFO read counter from further counting once the FIFO is empty (i.e., the EMPTY flag goes HIGH). A HIGH on this signal inhibits the counting.

FSTOP is used to stop the FIFO write counter from further counting once the FIFO is full (i.e., the FULL flag goes HIGH). A HIGH on this signal inhibits the counting.

For more information on these signals, refer to the "ESTOP and FSTOP Usage" section.

FULL, EMPTY

When the FIFO is full and no more data can be written, the FULL flag asserts HIGH. The FULL flag is synchronous to WCLK to inhibit writing immediately upon detection of a full condition and to prevent overflows. Since the write address is compared to a resynchronized (and thus time-delayed) version of the read address, the FULL flag will remain asserted until two WCLK active edges after a read operation eliminates the full condition.

When the FIFO is empty and no more data can be read, the EMPTY flag asserts HIGH. The EMPTY flag is synchronous to RCLK to inhibit reading immediately upon detection of an empty condition and to prevent underflows. Since the read address is compared to a resynchronized (and thus time-delayed) version of the write address, the EMPTY flag will remain asserted until two RCLK active edges after a write operation removes the empty condition.

For more information on these signals, refer to the "FIFO Flag Usage Considerations" section on page 147.

AFULL, AEMPTY

These are programmable flags and will be asserted on the threshold specified by AFVAL and AEVAL, respectively.

When the number of words stored in the FIFO reaches the amount specified by AEVAL while reading, the AEMPTY output will go HIGH. Likewise, when the number of words stored in the FIFO reaches the amount specified by AFVAL while writing, the AFULL output will go HIGH.

AFVAL, AEVAL

The AEVAL and AFVAL pins are used to specify the almost-empty and almost-full threshold values. They are 12-bit signals. For more information on these signals, refer to the "FIFO Flag Usage Considerations" section on page 147.

FIFO Usage

ESTOP and FSTOP Usage

The ESTOP pin is used to stop the read counter from counting any further once the FIFO is empty (i.e., the EMPTY flag goes HIGH). Likewise, the FSTOP pin is used to stop the write counter from counting any further once the FIFO is full (i.e., the FULL flag goes HIGH).

The FIFO counters in the device start the count at zero, reach the maximum depth for the configuration (e.g., 511 for a 512×9 configuration), and then restart at zero. An example application for ESTOP, where the read counter keeps counting, would be writing to the FIFO once and reading the same content over and over without doing another write.

FIFO Flag Usage Considerations

The AEVAL and AFVAL pins are used to specify the 12-bit AEMPTY and AFULL threshold values. The FIFO contains separate 12-bit write address (WADDR) and read address (RADDR) counters. WADDR is incremented every time a write operation is performed, and RADDR is incremented every time a read operation is performed. Whenever the difference between WADDR and RADDR is greater than or equal to AFVAL, the AFULL output is asserted. Likewise, whenever the difference between WADDR and RADDR is less than or equal to AEVAL, the AEMPTY output is asserted. To handle different read and write aspect ratios, AFVAL and AEVAL are expressed in terms of total data bits instead of total data words. When users specify AFVAL and AEVAL in terms of read or write words, the SmartGen tool translates them into bit addresses and configures these signals automatically. SmartGen configures the AFULL flag to assert when the write address exceeds the read address by at least a predefined value. In a 2k×8 FIFO, for example, a value of 1,500 for AFVAL means that the AFULL flag will be asserted after a write when the difference between the write address and the read address reaches 1,500 (there have been at least 1,500 more writes than reads). It will stay asserted until the difference between the write and read addresses drops below 1,500.

The AEMPTY flag is asserted when the difference between the write address and the read address is less than a predefined value. In the example above, a value of 200 for AEVAL means that the AEMPTY flag will be asserted when a read causes the difference between the write address and the read address to drop to 200. It will stay asserted until that difference rises above 200. Note that the FIFO can be configured with different read and write widths; in this case, the AFVAL setting is based on the number of write data entries, and the AEVAL setting is based on the number of read data entries. For aspect ratios of 512×9 and 256×18, only 4,096 bits can be addressed by the 12 bits of AFVAL and AEVAL. The number of words must be multiplied by 8 and 16 instead of 9 and 18. The SmartGen tool automatically uses the proper values. To avoid halfwords being written or read, which could happen if different read and write aspect ratios were specified, the FIFO will assert FULL or EMPTY as soon as at least one word cannot be written or read. For example, if a two-bit word is written and a four-bit word is being read, the FIFO will remain in the empty state when the first word is written. This occurs even if the FIFO is not completely empty, because in this case, a complete word cannot be read. The same is applicable in the full state. If a four-bit word is written and a two-bit word is read, the FIFO is full and one word is read. The FULL flag will remain asserted because a complete word cannot be written at this point.

Variable Aspect Ratio and Cascading

Variable aspect ratio and cascading allow users to configure the memory in the width and depth required. The memory block can be configured as a FIFO by combining the basic memory block with dedicated FIFO controller logic. The FIFO macro is named FIFO4KX18. Low power flash device RAM can be configured as 1, 2, 4, 9, or 18 bits wide. By cascading the memory blocks, any multiple of those widths can be created. The RAM blocks can be from 256 to 4,096 bits deep, depending on the aspect ratio, and the blocks can also be cascaded to create deeper areas. Refer to the aspect ratios available for each macro cell in the "SRAM Features" section on page 139. The largest continuous configurable memory area is equal to half the total memory available on the device, because the RAM is separated into two groups, one on each side of the device.

The SmartGen core generator will automatically configure and cascade both RAM and FIFO blocks. Cascading is accomplished using dedicated memory logic and does not consume user gates for depths up to 4,096 bits deep and widths up to 18, depending on the configuration. Deeper memory will utilize some user gates to multiplex the outputs.

Generated RAM and FIFO macros can be created as either structural VHDL or Verilog for easy instantiation into the design. Users of Libero SoC can create a symbol for the macro and incorporate it into a design schematic.

Table 6-10 on page 149 shows the number of memory blocks required for each of the supported depth and width memory configurations, and for each depth and width combination. For example, a 256-bit deep by 32-bit wide two-port RAM would consist of two 256×18 RAM blocks. The first 18 bits would be stored in the first RAM block, and the remaining 14 bits would be implemented in the other 256×18 RAM block. This second RAM block would have four bits of unused storage. Similarly, a dual-port memory block that is 8,192 bits deep and 8 bits wide would be implemented using 16 memory blocks. The dual-port memory would be configured in a 4,096×1 aspect ratio. These blocks would then be cascaded two deep to achieve 8,192 bits of depth, and eight wide to achieve the eight bits of width.

Table 6-8 and Table 6-9 show the maximum potential width and depth configuration for each device. Note that 15 k and 30 k gate devices do not support RAM or FIFO.

***Table 6-8* • Memory Availability per IGLOO and ProASIC3 Device**

Device		RAM Blocks	Maximum Potential Width[1]		Maximum Potential Depth[2]	
IGLOO IGLOO nano IGLOO PLUS	**ProASIC3 ProASIC3 nano ProASIC3L**		**Depth**	**Width**	**Depth**	**Width**
AGL060 AGLN060 AGLP060	A3P060 A3PN060	4	256	72 (4×18)	16,384 (4,096×4)	1
AGL125 AGLN125 AGLP125	A3P125 A3PN125	8	256	144 (8×18)	32,768 (4,094×8)	1
AGL250 AGLN250	A3P250/L A3PN250	8	256	144 (8×18)	32,768 (4,096×8)	1
AGL400	A3P400	12	256	216 (12×18)	49,152 (4,096×12)	1
AGL600	A3P600/L	24	256	432 (24×18)	98,304 (4,096×24)	1
AGL1000	A3P1000/L	32	256	576 (32×18)	131,072 (4,096×32)	1
AGLE600	A3PE600	24	256	432 (24×18)	98,304 (4,096×24)	1
	A3PE1500	60	256	1,080 (60×18)	245,760 (4,096×60)	1
AGLE3000	A3PE3000/L	112	256	2,016 (112×18)	458,752 (4,096×112)	1

Notes:

1. *Maximum potential width uses the two-port configuration.*
2. *Maximum potential depth uses the dual-port configuration.*

***Table 6-9* • Memory Availability per Fusion Device**

Device	RAM Blocks	Maximum Potential Width[1]		Maximum Potential Depth[2]	
		Depth	**Width**	**Depth**	**Width**
AFS090	6	256	108 (6×18)	24,576 (4,094×6)	1
AFS250	8	256	144 (8×18)	32,768 (4,094×8)	1
AFS600	24	256	432 (24×18)	98,304 (4,096×24)	1
AFS1500	60	256	1,080 (60×18)	245,760 (4,096×60)	1

Notes:

1. *Maximum potential width uses the two-port configuration.*
2. *Maximum potential depth uses the dual-port configuration.*

Table 6-10 • **RAM and FIFO Memory Block Consumption**

			Depth									
			256		512	1,024	2,048	4,096	8,192	16,384	32,768	65,536
			Two-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port
Width	1	Number Block	1	1	1	1	1	1	2	4	8	16 × 1
		Configuration	Any	Any	Any	1,024 × 4	2,048 × 2	4,096 × 1	2 × (4,096 × 1) Cascade Deep	4 × (4,096 × 1) Cascade Deep	8 × (4,096 × 1) Cascade Deep	16 × (4,096 × 1) Cascade Deep
	2	Number Block	1	1	1	1	1	2	4	8	16	32
		Configuration	Any	Any	Any	1,024×4	2,048 × 2	2 × (4,096 × 1) Cascaded Wide	4 × (4,096 × 1) Cascaded 2 Deep and 2 Wide	8 × (4,096 × 1) Cascaded 4 Deep and 2 Wide	16 × (4,096 × 1) Cascaded 8 Deep and 2 Wide	32 × (4,096 × 1) Cascaded 16 Deep and 2 Wide
	4	Number Block	1	1	1	1	2	4	8	16	32	64
		Configuration	Any	Any	Any	1,024 × 4	2 × (2,048 × 2) Cascaded Wide	4 × (4,096 × 1) Cascaded Wide	4 × (4,096 × 1) Cascaded 2 Deep and 4 Wide	16 × (4,096 × 1) Cascaded 4 Deep and 4 Wide	32 × (4,096 × 1) Cascaded 8 Deep and 4 Wide	64 × (4,096 × 1) Cascaded 16 Deep and 4 Wide
	8	Number Block	1	1	1	2	4	8	16	32	64	
		Configuration	Any	Any	Any	2 × (1,024 × 4) Cascaded Wide	4 × (2,048 × 2) Cascaded Wide	8 × (4,096 × 1) Cascaded Wide	16 × (4,096 × 1) Cascaded 2 Deep and 8 Wide	32 × (4,096 × 1) Cascaded 4 Deep and 8 Wide	64 × (4,096 × 1) Cascaded 8 Deep and 8 Wide	
	9	Number Block	1	1	1	2	4	8	16	32		
		Configuration	Any	Any	Any	2 × (512 × 9) Cascaded Deep	4 × (512 × 9) Cascaded Deep	8 × (512 × 9) Cascaded Deep	16 × (512 × 9) Cascaded Deep	32 × (512 × 9) Cascaded Deep		
	16	Number Block	1	1	1	4	8	16	32	64		
		Configuration	256 × 18	256 × 18	256 × 18	4 × (1,024 × 4) Cascaded Wide	8 × (2,048 × 2) Cascaded Wide	16 × (4,096 × 1) Cascaded Wide	32 × (4,096 × 1) Cascaded 2 Deep and 16 Wide	32 × (4,096 × 1) Cascaded 4 Deep and 16 Wide		
	18	Number Block	1	2	2	4	8	18	32			
		Configuration	256 × 8	2 × (512 × 9) Cascaded Wide	2 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded 2 Deep and 2 Wide	8 × (512 × 9) Cascaded 4 Deep and 2 Wide	16 × (512 × 9) Cascaded 8 Deep and 2 Wide	16 × (512 × 9) Cascaded 16 Deep and 2 Wide			
	32	Number Block	2	4	4	8	16	32	64			
		Configuration	2 × (256 × 18) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded Wide	8 × (1,024 × 4) Cascaded Wide	16 × (2,048 × 2) Cascaded Wide	32 × (4,096 × 1) Cascaded Wide	64 × (4,096 × 1) Cascaded 2 Deep and 32 Wide			
	36	Number Block	2	4	4	8	16	32				
		Configuration	2 × (256 × 18) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded 2 Deep and 4 Wide	16 × (512 × 9) Cascaded 4 Deep and 4 Wide	16 × (512 × 9) Cascaded 8 Deep and 4 Wide				
	64	Number Block	4	8	8	16	32	64				
		Configuration	4 × (256 × 18) Cascaded Wide	8 × (512 × 9) Cascaded Wide	8 × (512 × 9) Cascaded Wide	16 × (1,024 × 4) Cascaded Wide	32 × (2,048 × 2) Cascaded Wide	64 × (4,096 × 1) Cascaded Wide				
	72	Number Block	4	8	8	16	32					
		Configuration	4 × (256 × 18) Cascaded Wide	8 × (512 × 9) Cascaded Wide	8 × (512 × 9) Cascaded Wide	16 × (512 × 9) Cascaded Wide	16 × (512 × 9) Cascaded 4 Deep and 8 Wide					

Note: *Memory configurations represented by grayed cells are not supported.*

Initializing the RAM/FIFO

The SRAM blocks can be initialized with data to use as a lookup table (LUT). Data initialization can be accomplished either by loading the data through the design logic or through the UJTAG interface. The UJTAG macro is used to allow access from the JTAG port to the internal logic in the device. By sending the appropriate initialization string to the JTAG Test Access Port (TAP) Controller, the designer can put the JTAG circuitry into a mode that allows the user to shift data into the array logic through the JTAG port using the UJTAG macro. For a more detailed explanation of the UJTAG macro, refer to the "FlashROM in Microsemi's Low Power Flash Devices" section on page 119.

A user interface is required to receive the user command, initialization data, and clock from the UJTAG macro. The interface must synchronize and load the data into the correct RAM block of the design. The main outputs of the user interface block are the following:

- Memory block chip select: Selects a memory block for initialization. The chip selects signals for each memory block that can be generated from different user-defined pockets or simple logic, such as a ring counter (see below).
- Memory block write address: Identifies the address of the memory cell that needs to be initialized.
- Memory block write data: The interface block receives the data serially from the UTDI port of the UJTAG macro and loads it in parallel into the write data ports of the memory blocks.
- Memory block write clock: Drives the WCLK of the memory block and synchronizes the write data, write address, and chip select signals.

Figure 6-8 shows the user interface between UJTAG and the memory blocks.

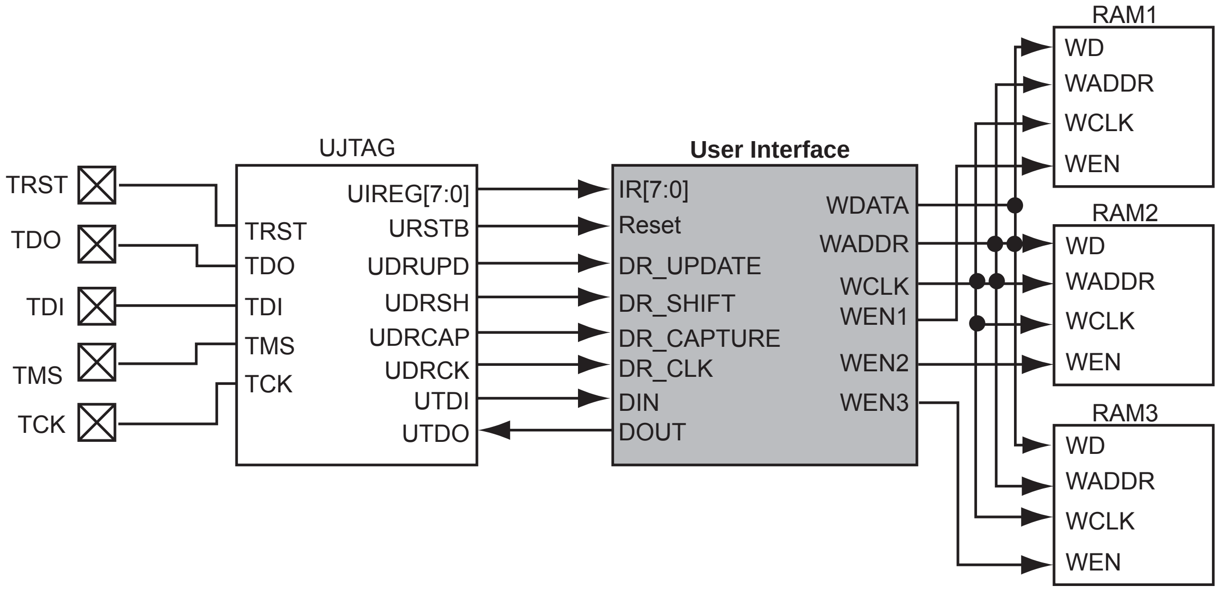


***Figure 6-8* • Interfacing TAP Ports and SRAM Blocks**

An important component of the interface between the UJTAG macro and the RAM blocks is a serial-in/parallel-out shift register. The width of the shift register should equal the data width of the RAM blocks. The RAM data arrives serially from the UTDI output of the UJTAG macro. The data must be shifted into a shift register clocked by the JTAG clock (provided at the UDRCK output of the UJTAG macro).

Then, after the shift register is fully loaded, the data must be transferred to the write data port of the RAM block. To synchronize the loading of the write data with the write address and write clock, the output of the shift register can be pipelined before driving the RAM block.

The write address can be generated in different ways. It can be imported through the TAP using a different instruction opcode and another shift register, or generated internally using a simple counter. Using a counter to generate the address bits and sweep through the address range of the RAM blocks is recommended, since it reduces the complexity of the user interface block and the board-level JTAG driver.

Moreover, using an internal counter for address generation speeds up the initialization procedure, since the user only needs to import the data through the JTAG port.

The designer may use different methods to select among the multiple RAM blocks. Using counters along with demultiplexers is one approach to set the write enable signals. Basically, the number of RAM blocks needing initialization determines the most efficient approach. For example, if all the blocks are initialized with the same data, one enable signal is enough to activate the write procedure for all of them at the same time. Another alternative is to use different opcodes to initialize each memory block. For a small number of RAM blocks, using counters is an optimal choice. For example, a ring counter can be used to select from multiple RAM blocks. The clock driver of this counter needs to be controlled by the address generation process.

Once the addressing of one block is finished, a clock pulse is sent to the (ring) counter to select the next memory block.

Figure 6-9 illustrates a simple block diagram of an interface block between UJTAG and RAM blocks.

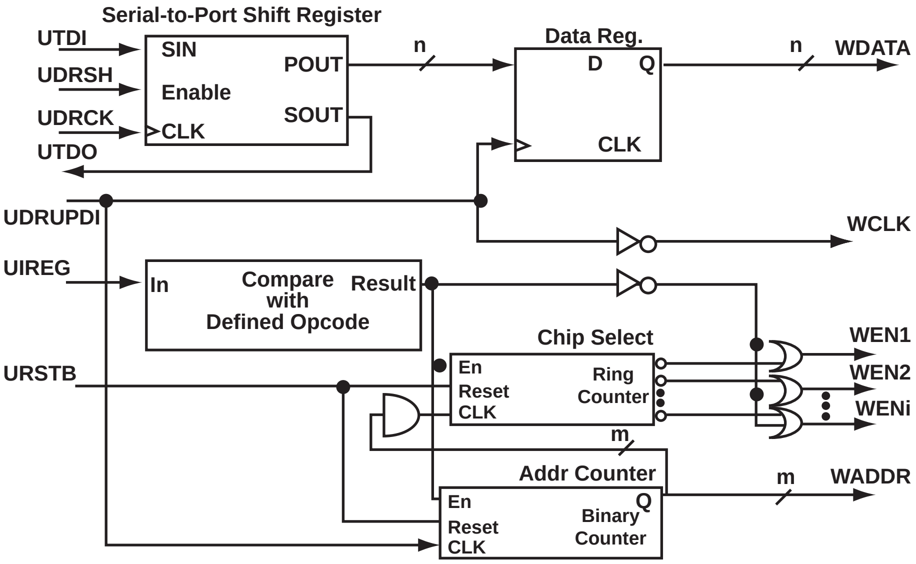


Figure 6-9 • **Block Diagram of a Sample User Interface**

In the circuit shown in Figure 6-9, the shift register is enabled by the UDRSH output of the UJTAG macro. The counters and chip select outputs are controlled by the value of the TAP Instruction Register. The comparison block compares the UIREG value with the "start initialization" opcode value (defined by the user). If the result is true, the counters start to generate addresses and activate the WEN inputs of appropriate RAM blocks.

The UDRUPD output of the UJTAG macro, also shown in Figure 6-9, is used for generating the write clock (WCLK) and synchronizing the data register and address counter with WCLK. UDRUPD is HIGH when the TAP Controller is in the Data Register Update state, which is an indication of completing the loading of one data word. Once the TAP Controller goes into the Data Register Update state, the UDRUPD output of the UJTAG macro goes HIGH. Therefore, the pipeline register and the address counter place the proper data and address on the outputs of the interface block. Meanwhile, WCLK is defined as the inverted UDRUPD. This will provide enough time (equal to the UDRUPD HIGH time) for the data and address to be placed at the proper ports of the RAM block before the rising edge of WCLK. The inverter is not required if the RAM blocks are clocked at the falling edge of the write clock. An example of this is described in the "Example of RAM Initialization" section on page 152.

Example of RAM Initialization

This section of the document presents a sample design in which a 4×4 RAM block is being initialized through the JTAG port. A test feature has been implemented in the design to read back the contents of the RAM after initialization to verify the procedure.

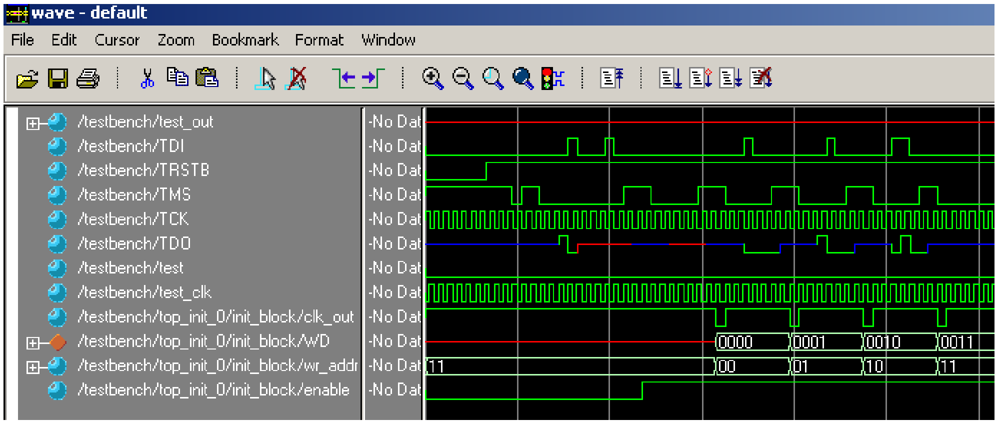
The interface block of this example performs two major functions: initialization of the RAM block and running a test procedure to read back the contents. The clock output of the interface is either the write clock (for initialization) or the read clock (for reading back the contents). The Verilog code for the interface block is included in the "Sample Verilog Code" section on page 153.

For simulation purposes, users can declare the input ports of the UJTAG macro for easier assignment in the testbench. However, the UJTAG input ports should not be declared on the top level during synthesis. If the input ports of the UJTAG are declared during synthesis, the synthesis tool will instantiate input buffers on these ports. The input buffers on the ports will cause Compile to fail in Designer.

Figure 6-10 shows the simulation results for the initialization step of the example design.

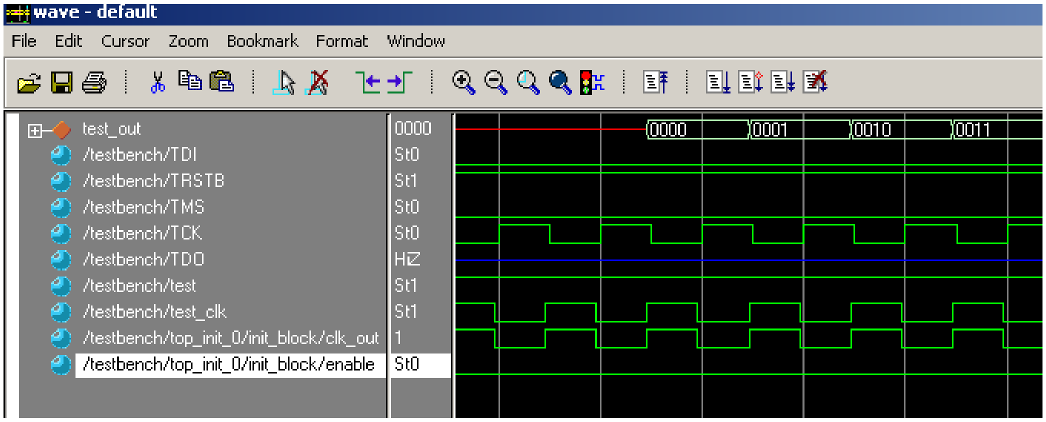
The CLK_OUT signal, which is the clock output of the interface block, is the inverted DR_UPDATE output of the UJTAG macro. It is clear that it gives sufficient time (while the TAP Controller is in the Data Register Update state) for the write address and data to become stable before loading them into the RAM block.

Figure 6-11 presents the test procedure of the example. The data read back from the memory block matches the written data, thus verifying the design functionality.



/testbench/test_out -No Dat
/testbench/TDI -No Dat
/testbench/TRSTB -No Dat
/testbench/TMS -No Dat
/testbench/TCK -No Dat
/testbench/TDO -No Dat
/testbench/test -No Dat
/testbench/test_clk -No Dat
/testbench/top_init_0/init_block/clk_out -No Dat
/testbench/top_init_0/init_block/WD -No Dat
/testbench/top_init_0/init_block/wr_addr -No Dat
/testbench/top_init_0/init_block/enable -No Dat

Figure 6-10 • Simulation of Initialization Step



test_out 0000
/testbench/TDI St0
/testbench/TRSTB St1
/testbench/TMS St0
/testbench/TCK St0
/testbench/TDO HiZ
/testbench/test St1
/testbench/test_clk St1
/testbench/top_init_0/init_block/clk_out 1
/testbench/top_init_0/init_block/enable St0

Figure 6-11 • Simulation of the Test Procedure of the Example

The ROM emulation application is based on RAM block initialization. If the user's main design has access only to the read ports of the RAM block (RADDR, RD, RCLK, and REN), and the contents of the RAM are already initialized through the TAP, then the memory blocks will emulate ROM functionality for the core design. In this case, the write ports of the RAM blocks are accessed only by the user interface block, and the interface is activated only by the TAP Instruction Register contents.

Users should note that the contents of the RAM blocks are lost in the absence of applied power. However, the 1 kbit of flash memory, FlashROM, in low power flash devices can be used to retain data after power is removed from the device. Refer to the "SRAM and FIFO Memories in Microsemi's Low Power Flash Devices" section on page 133 for more information.

Sample Verilog Code

Interface Block

```
`define Initialize_start 8'h22 //INITIALIZATION START COMMAND VALUE
`define Initialize_stop 8'h23 //INITIALIZATION START COMMAND VALUE

module interface(IR, rst_n, data_shift, clk_in, data_update, din_ser, dout_ser, test,
  test_out,test_clk,clk_out,wr_en,rd_en,write_word,read_word,rd_addr, wr_addr);

input [7:0] IR;
input [3:0] read_word; //RAM DATA READ BACK
input rst_n, data_shift, clk_in, data_update, din_ser; //INITIALIZATION SIGNALS
input test, test_clk; //TEST PROCEDURE CLOCK AND COMMAND INPUT
output [3:0] test_out; //READ DATA
output [3:0] write_word; //WRITE DATA
output [1:0] rd_addr; //READ ADDRESS
output [1:0] wr_addr; //WRITE ADDRESS
output dout_ser; //TDO DRIVER
output clk_out, wr_en, rd_en;

wire [3:0] write_word;
wire [1:0] rd_addr;
wire [1:0] wr_addr;
wire [3:0] Q_out;
wire enable, test_active;

reg clk_out;

//SELECT CLOCK FOR INITIALIZATION OR READBACK TEST
always @(enable or test_clk or data_update)
begin
  case ({test_active})
    1 : clk_out = test_clk ;
    0 : clk_out = !data_update;
    default : clk_out = 1'b1;
  endcase
end

assign test_active = test && (IR == 8'h23);
assign enable = (IR == 8'h22);
assign wr_en = !enable;
assign rd_en = !test_active;
assign test_out = read_word;
assign dout_ser = Q_out[3];

//4-bit SIN/POUT SHIFT REGISTER
shift_reg data_shift_reg (.Shiften(data_shift), .Shiftin(din_ser), .Clock(clk_in),
  .Q(Q_out));

//4-bit PIPELINE REGISTER
D_pipeline pipeline_reg (.Data(Q_out), .Clock(data_update), .Q(write_word));
```

```
//
addr_counter counter_1 (.Clock(data_update), .Q(wr_addr), .Aset(rst_n),
  .Enable(enable));
addr_counter counter_2 (.Clock(test_clk), .Q(rd_addr), .Aset(rst_n),
  .Enable( test_active));

endmodule
```

Interface Block / UJTAG Wrapper

This example is a sample wrapper, which connects the interface block to the UJTAG and the memory blocks.

```
// WRAPPER
module top_init (TDI, TRSTB, TMS, TCK, TDO, test, test_clk, test_ out);

input TDI, TRSTB, TMS, TCK;
output TDO;
input test, test_clk;
output [3:0] test_out;

wire [7:0] IR;
wire reset, DR_shift, DR_cap, init_clk, DR_update, data_in, data_out;
wire clk_out, wen, ren;
wire [3:0] word_in, word_out;
wire [1:0] write_addr, read_addr;

UJTAG UJTAG_U1 (.UIREG0(IR[0]), .UIREG1(IR[1]), .UIREG2(IR[2]), .UIREG3(IR[3]),
  .UIREG4(IR[4]), .UIREG5(IR[5]), .UIREG6(IR[6]), .UIREG7(IR[7]), .URSTB(reset),
  .UDRSH(DR_shift), .UDRCAP(DR_cap), .UDRCK(init_clk), .UDRUPD(DR_update),
  .UT-DI(data_in), .TDI(TDI), .TMS(TMS), .TCK(TCK), .TRSTB(TRSTB), .TDO(TDO),
  .UT-DO(data_out));
mem_block RAM_block (.DO(word_out), .RCLOCK(clk_out), .WCLOCK(clk_out), .DI(word_in),
  .WRB(wen), .RDB(ren), .WAD-DR(write_addr), .RADDR(read_addr));
interface init_block (.IR(IR), .rst_n(reset), .data_shift(DR_shift), .clk_in(init_clk),
  .data_update(DR_update), .din_ser(data_in), .dout_ser(data_out), .test(test),
  .test_out(test_out), .test_clk(test_clk), .clk_out(clk_out), .wr_en(wen),
  .rd_en(ren), .write_word(word_in), .read_word(word_out), .rd_addr(read_addr),
  .wr_addr(write_addr));

endmodule
```

Address Counter

```
module addr_counter (Clock, Q, Aset, Enable);

input Clock;
output [1:0] Q;
input Aset;
input Enable;

reg [1:0] Qaux;

always @(posedge Clock or negedge Aset)
begin
  if (!Aset) Qaux <= 2'b11;
  else if (Enable) Qaux <= Qaux + 1;
end

assign Q = Qaux;

endmodule
```

Pipeline Register

```
module D_pipeline (Data, Clock, Q);

input [3:0] Data;
input Clock;
output [3:0] Q;

reg [3:0] Q;

always @ (posedge Clock) Q <= Data;

endmodule
```

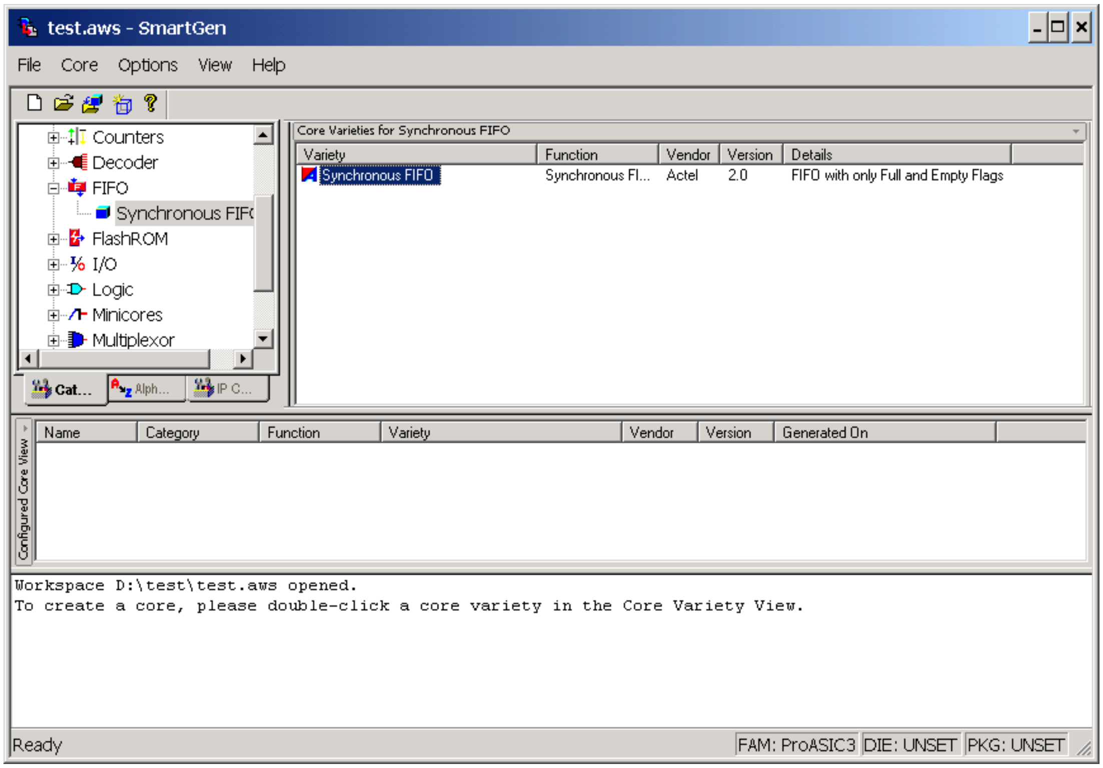
4x4 RAM Block (created by SmartGen Core Generator)

```
module mem_block(DI,DO,WADDR,RADDR,WRB,RDB,WCLOCK,RCLOCK);

input [3:0] DI;
output [3:0] DO;
input [1:0] WADDR, RADDR;
input WRB, RDB, WCLOCK, RCLOCK;

wire WEBP, WEAP, VCC, GND;

VCC VCC_1_net(.Y(VCC));
GND GND_1_net(.Y(GND));
INV WEBUBBLEB(.A(WRB), .Y(WEBP));
RAM4K9 RAMBLOCK0(.ADDRA11(GND), .ADDRA10(GND), .ADDRA9(GND), .ADDRA8(GND),
  .ADDRA7(GND), .ADDRA6(GND), .ADDRA5(GND), .ADDRA4(GND), .ADDRA3(GND), .ADDRA2(GND),
  .ADDRA1(RADDR[1]), .ADDRA0(RADDR[0]), .ADDRB11(GND), .ADDRB10(GND), .ADDRB9(GND),
  .ADDRB8(GND), .ADDRB7(GND), .ADDRB6(GND), .ADDRB5(GND), .ADDRB4(GND), .ADDRB3(GND),
  .ADDRB2(GND), .ADDRB1(WADDR[1]), .ADDRB0(WADDR[0]), .DINA8(GND), .DINA7(GND),
  .DINA6(GND), .DINA5(GND), .DINA4(GND), .DINA3(GND), .DINA2(GND), .DINA1(GND),
  .DINA0(GND), .DINB8(GND), .DINB7(GND), .DINB6(GND), .DINB5(GND), .DINB4(GND),
  .DINB3(DI[3]), .DINB2(DI[2]), .DINB1(DI[1]), .DINB0(DI[0]), .WIDTHA0(GND),
  .WIDTHA1(VCC), .WIDTHB0(GND), .WIDTHB1(VCC), .PIPEA(GND), .PIPEB(GND),
  .WMODEA(GND), .WMODEB(GND), .BLKA(WEAP), .BLKB(WEBP), .WENA(VCC), .WENB(GND),
  .CLKA(RCLOCK), .CLKB(WCLOCK), .RESET(VCC), .DOUTA8(), .DOUTA7(), .DOUTA6(),
  .DOUTA5(), .DOUTA4(), .DOUTA3(DO[3]), .DOUTA2(DO[2]), .DOUTA1(DO[1]),
  .DOUTA0(DO[0]), .DOUTB8(), .DOUTB7(), .DOUTB6(), .DOUTB5(), .DOUTB4(), .DOUTB3(),
  .DOUTB2(), .DOUTB1(), .DOUTB0());
INV WEBUBBLEA(.A(RDB), .Y(WEAP));

endmodule
```

Software Support

The SmartGen core generator is the easiest way to select and configure the memory blocks (Figure 6-12). SmartGen automatically selects the proper memory block type and aspect ratio, and cascades the memory blocks based on the user's selection. SmartGen also configures any additional signals that may require tie-off.

SmartGen will attempt to use the minimum number of blocks required to implement the desired memory. When cascading, SmartGen will configure the memory for width before configuring for depth. For example, if the user requests a 256×8 FIFO, SmartGen will use a 512×9 FIFO configuration, not 256×18.



Core Varieties for Synchronous FIFO

Variety	Function	Vendor	Version	Details
Synchronous FIFO	Synchronous Fl...	Actel	2.0	FIFO with only Full and Empty Flags

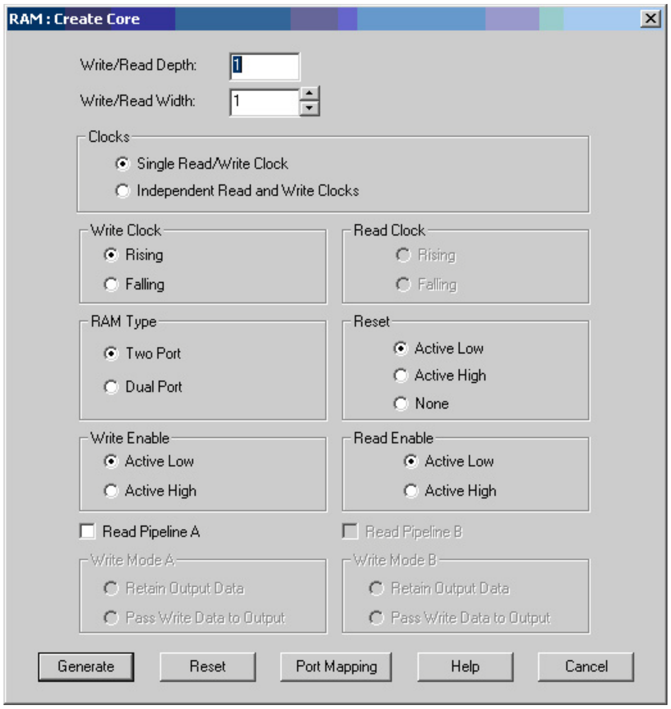
Name	Category	Function	Variety	Vendor	Version	Generated On

```
Workspace D:\test\test.aws opened.
To create a core, please double-click a core variety in the Core Variety View.
```

FAM: ProASIC3 DIE: UNSET PKG: UNSET

***Figure 6-12* • SmartGen Core Generator Interface**

SmartGen enables the user to configure the desired RAM element to use either a single clock for read and write, or two independent clocks for read and write. The user can select the type of RAM as well as the width/depth and several other parameters (Figure 6-13).



RAM : Create Core

Write/Read Depth: 1
Write/Read Width: 1

Clocks
● Single Read/Write Clock
○ Independent Read and Write Clocks

Write Clock
● Rising
○ Falling

Read Clock
○ Rising
○ Falling

RAM Type
● Two Port
○ Dual Port

Reset
● Active Low
○ Active High
○ None

Write Enable
● Active Low
○ Active High

Read Enable
● Active Low
○ Active High

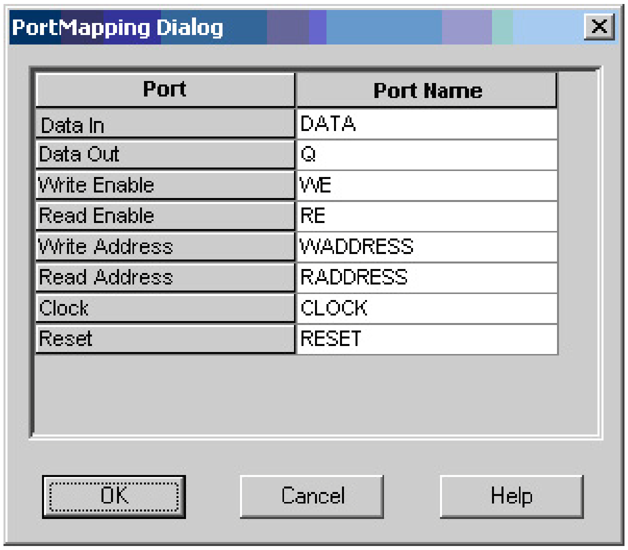
□ Read Pipeline A
□ Read Pipeline B

Write Mode A
○ Retain Output Data
○ Pass Write Data to Output

Write Mode B
○ Retain Output Data
○ Pass Write Data to Output

***Figure 6-13* • SmartGen Memory Configuration Interface**

SmartGen also has a Port Mapping option that allows the user to specify the names of the ports generated in the memory block (Figure 6-14).



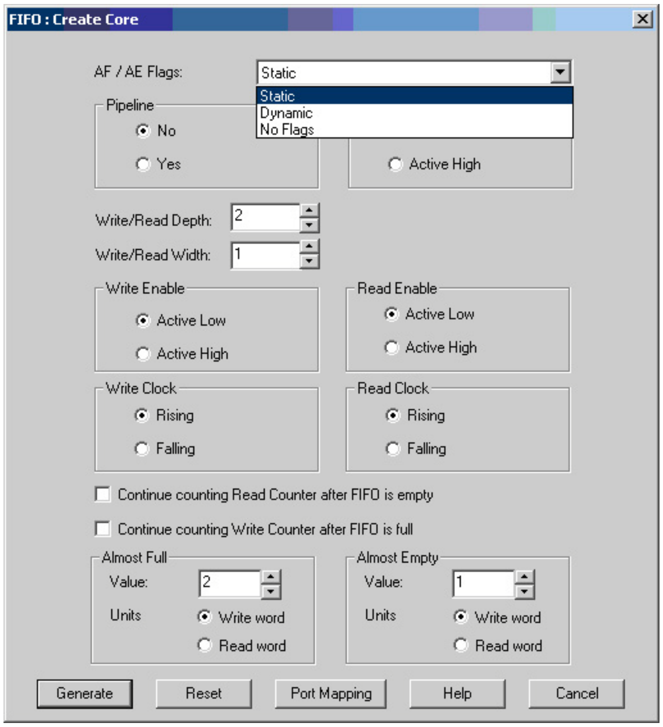
PortMapping Dialog

Port	Port Name
Data In	DATA
Data Out	Q
Write Enable	WE
Read Enable	RE
Write Address	WADDRESS
Read Address	RADDRESS
Clock	CLOCK
Reset	RESET

***Figure 6-14* • Port Mapping Interface for SmartGen-Generated Memory**

SmartGen also configures the FIFO according to user specifications. Users can select no flags, static flags, or dynamic flags. Static flag settings are configured using configuration flash and cannot be altered without reprogramming the device. Dynamic flag settings are determined by register values and can be altered without reprogramming the device by reloading the register values either from the design or through the UJTAG interface described in the "Initializing the RAM/FIFO" section on page 150.

SmartGen can also configure the FIFO to continue counting after the FIFO is full. In this configuration, the FIFO write counter will wrap after the counter is full and continue to write data. With the FIFO configured to continue to read after the FIFO is empty, the read counter will also wrap and re-read data that was previously read. This mode can be used to continually read back repeating data patterns stored in the FIFO (Figure 6-15).



FIFO : Create Core

AF / AE Flags: Static

Pipeline: ● No ○ Yes

○ Active High

Write/Read Depth: 2

Write/Read Width: 1

Write Enable: ● Active Low ○ Active High

Read Enable: ● Active Low ○ Active High

Write Clock: ● Rising ○ Falling

Read Clock: ● Rising ○ Falling

□ Continue counting Read Counter after FIFO is empty

□ Continue counting Write Counter after FIFO is full

Almost Full: Value: 2 Units ● Write word ○ Read word

Almost Empty: Value: 1 Units ● Write word ○ Read word

Figure 6-15 • **SmartGen FIFO Configuration Interface**

FIFOs configured using SmartGen can also make use of the port mapping feature to configure the names of the ports.

Limitations

Users should be aware of the following limitations when configuring SRAM blocks for low power flash devices:

- SmartGen does not track the target device in a family, so it cannot determine if a configured memory block will fit in the target device.
- Dual-port RAMs with different read and write aspect ratios are not supported.
- Cascaded memory blocks can only use a maximum of 64 blocks of RAM.
- The Full flag of the FIFO is sensitive to the maximum depth of the actual physical FIFO block, not the depth requested in the SmartGen interface.

Conclusion

Fusion, IGLOO, and ProASIC3 devices provide users with extremely flexible SRAM blocks for most design needs, with the ability to choose between an easy-to-use dual-port memory or a wide-word two-port memory. Used with the built-in FIFO controllers, these memory blocks also serve as highly efficient FIFOs that do not consume user gates when implemented. The SmartGen core generator provides a fast and easy way to configure these memory elements for use in designs.

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	The note connected with Figure 6-3 • Supported Basic RAM Macros, regarding RAM4K9, was revised to explain that it applies only to part numbers of certain revisions and earlier (SAR 29574).	138
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.5 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 6-1 • Flash-Based FPGAs.	136
	IGLOO nano and ProASIC3 nano devices were added to Figure 6-8 • Interfacing TAP Ports and SRAM Blocks.	150
v1.4 (October 2008)	The "SRAM/FIFO Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	136
	The "SRAM and FIFO Architecture" section was modified to remove "IGLOO and ProASIC3E" from the description of what the memory block includes, as this statement applies to all memory blocks.	137
	Wording in the "Clocking" section was revised to change "IGLOO and ProASIC3 devices support inversion" to "Low power flash devices support inversion." The reference to IGLOO and ProASIC3 development tools in the last paragraph of the section was changed to refer to development tools in general.	143
	The "ESTOP and FSTOP Usage" section was updated to refer to FIFO counters in devices in general rather than only IGLOO and ProASIC3E devices.	146
v1.3 (August 2008)	The note was removed from Figure 6-7 • RAM Block with Embedded FIFO Controller and placed in the WCLK and RCLK description.	144
	The "WCLK and RCLK" description was revised.	145
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 6-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	136
v1.1 (March 2008)	The "Introduction" section was updated to include the IGLOO PLUS family.	133
	The "Device Architecture" section was updated to state that 15 k gate devices do not support SRAM and FIFO.	133
	The first note in Figure 6-1 • IGLOO and ProASIC3 Device Architecture Overview was updated to include mention of 15 k gate devices, and IGLOO PLUS was added to the second note.	135

Date	Changes	Page
v1.1 (continued)	Table 6-1 • Flash-Based FPGAs and associated text were updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	136
	The text introducing Table 6-8 • Memory Availability per IGLOO and ProASIC3 Device was updated to replace "A3P030 and AGL030" with "15 k and 30 k gate devices." Table 6-8 • Memory Availability per IGLOO and ProASIC3 Device was updated to remove AGL400 and AGLE1500 and include IGLOO PLUS and ProASIC3L devices.	148

7 – I/O Structures in IGLOO and ProASIC3 Devices

Introduction

Low power flash devices feature a flexible I/O structure, supporting a range of mixed voltages (1.2 V, 1.5 V, 1.8 V, 2.5 V, and 3.3 V) through bank-selectable voltages. IGLOO,® ProASIC3®L, and ProASIC3 families support Standard, Standard Plus, and Advanced I/Os.

Users designing I/O solutions are faced with a number of implementation decisions and configuration choices that can directly impact the efficiency and effectiveness of their final design. The flexible I/O structure, supporting a wide variety of voltages and I/O standards, enables users to meet the growing challenges of their many diverse applications. Libero SoC software provides an easy way to implement I/Os that will result in robust I/O design.

This document first describes the two different I/O types in terms of the standards and features they support. It then explains the individual features and how to implement them in Libero SoC.

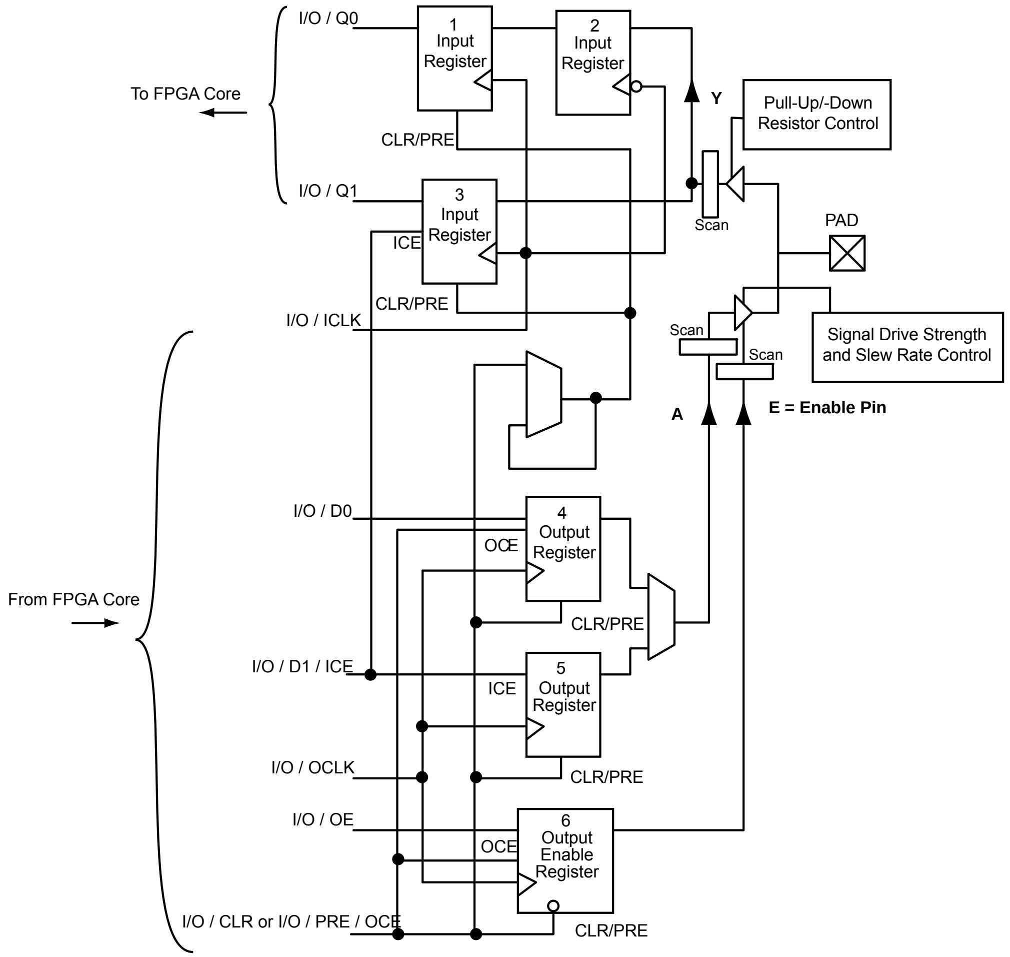


Figure 7-1 • **DDR Configured I/O Block Logical Representation**

Low Power Flash Device I/O Support

The low power flash FPGAs listed in Table 7-1 support I/Os and the functions described in this document.

Table 7-1 **• Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 7-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 7-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Advanced I/Os—IGLOO, ProASIC3L, and ProASIC3

Table 7-2 and Table 7-3 show the voltages and compatible I/O standards for the IGLOO, ProASIC3L, and ProASIC3 families.

I/Os provide programmable slew rates (except 30 K gate devices), drive strengths, and weak pull-up and pull-down circuits. 3.3 V PCI and 3.3 V PCI-X can be configured to be 5 V–tolerant. See the "5 V Input Tolerance" section on page 180 for possible implementations of 5 V tolerance.

All I/Os are in a known state during power-up, and any power-up sequence is allowed without current impact. Refer to the "I/O Power-Up and Supply Voltage Thresholds for Power-On Reset (Commercial and Industrial)" section in the datasheet for more information. During power-up, before reaching activation levels, the I/O input and output buffers are disabled while the weak pull-up is enabled. Activation levels are described in the datasheet.

Table 7-2 • **Supported I/O Standards**

IGLOO	**AGL015**	**AGL030**	**AGL060**	**AGL125**	**AGL250**		**AGL600**	**AGL1000**
ProASIC3	**A3P015**	**A3P030**	**A3P060**	**A3P125**	**A3P250/ A3P250L**	**A3P400**	**A3P600/ A3P600L**	**A3P1000/ A3P1000L**
Single-Ended								
LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V / 1.8 V / 1.5 V / 1.2 V LVCMOS 2.5 V / 5.0 V	✓	✓	✓	✓	✓	✓	✓	✓
3.3 V PCI/PCI-X	–	–	✓	✓	✓	✓	✓	✓
Differential								
LVPECL, LVDS, B-LVDS, M-LVDS	–	–	–	–	✓	✓	✓	✓

I/O Banks and I/O Standards Compatibility

I/Os are grouped into I/O voltage banks.

Each I/O voltage bank has dedicated I/O supply and ground voltages (VMV/GNDQ for input buffers and VCCI/GND for output buffers). This isolation is necessary to minimize simultaneous switching noise from the input and output (SSI and SSO). The switching noise (ground bounce and power bounce) is generated by the output buffers and transferred into input buffer circuits, and vice versa. Because of these dedicated supplies, only I/Os with compatible standards can be assigned to the same I/O voltage bank. Table 7-3 shows the required voltage compatibility values for each of these voltages.

There are four I/O banks on the 250K gate through 1M gate devices.

There are two I/O banks on the 30K, 60K, and 125K gate devices.

I/O standards are compatible if their VCCI and VMV values are identical. VMV and GNDQ are "quiet" input power supply pins and are not used on 30K gate devices (Table 7-3).

Table 7-3 • **VCCI Voltages and Compatible IGLOO and ProASIC3 Standards**

VCCI and VMV (typical)	**Compatible Standards**
3.3 V	LVTTL/LVCMOS 3.3, PCI 3.3, PCI-X 3.3 LVPECL
2.5 V	LVCMOS 2.5, LVCMOS 2.5/5.0, LVDS, B-LVDS, M-LVDS
1.8 V	LVCMOS 1.8
1.5 V	LVCMOS 1.5
1.2 V	LVCMOS 1.2

I/O Banks

Advanced I/Os are divided into multiple technology banks. Each device has two to four banks, and the number of banks is device-dependent as described above. The bank types have different characteristics, such as drive strength, the I/O standards supported, and timing and power differences.

There are three types of banks: Advanced I/O banks, Standard Plus I/O banks, and Standard I/O banks.

Advanced I/O banks offer single-ended and differential capabilities. These banks are available on the east and west sides of 250K, 400K, 600K, and 1M gate devices.

Standard Plus I/O banks offer LVTTL/LVCMOS and PCI single-ended I/O standards. These banks are available on the north and south sides of 250K, 400K, 600K, and 1M gate devices as well as all sides of 125K and 60K devices.

Standard I/O banks offer LVTTL/LVCMOS single-ended I/O standards. These banks are available on all sides of 30K gate devices.

Table 7-4 shows the I/O bank types, devices and bank locations supported, drive strength, slew rate control, and supported standards.

All inputs and disabled outputs are voltage-tolerant up to 3.3 V.

For more information about I/O and global assignments to I/O banks in a device, refer to the specific pin table for the device in the packaging section of the datasheet and the "User I/O Naming Convention" section on page 192.

***Table 7-4* • IGLOO and ProASIC3 Bank Type Definitions and Differences**

			I/O Standards Supported		
I/O Bank Type	**Device and Bank Location**	**Drive Strength**	**LVTTL/ LVCMOS**	**PCI/PCI-X**	**LVPECL, LVDS, B-LVDS, M-LVDS**
Standard	30 k gate devices (all banks)	Refer to Table 7-14 on page 189	✓	Not Supported	Not Supported
Standard Plus	60 k and 125 k gate devices (all banks)	Refer to Table 7-15 on page 189	✓	✓	Not Supported
	North and south banks of 250 k and 1 M gate devices	Refer to Table 7-15 on page 189	✓	✓	Not Supported
Advanced	East and west banks of 250 k and 1 M gate devices	Refer to Table 7-16 on page 189	✓	✓	✓

Features Supported on Every I/O

Table 7-5 lists all features supported by transmitter/receiver for single-ended and differential I/Os. Table 7-6 on page 166 lists the performance of each I/O technology.

***Table 7-5* • I/O Features**

Feature	Description
All I/O	• High performance (Table 7-6 on page 166) • Electrostatic discharge (ESD) protection • I/O register combining option
Single-Ended Transmitter Features	• Hot-swap: – 30K gate devices: hot-swap in every mode – All other IGLOO and ProASIC3 devices: no hot-swap • Output slew rate: 2 slew rates (except 30K gate devices) • Weak pull-up and pull-down resistors • Output drive: 3 drive strengths • Programmable output loading • Skew between output buffer enable/disable time: 2 ns delay on rising edge and 0 ns delay on falling edge (see the "Selectable Skew between Output Buffer Enable and Disable Times" section on page 185 for more information) • LVTTL/LVCMOS 3.3 V outputs compatible with 5 V TTL inputs
Single-Ended Receiver Features	• 5 V–input–tolerant receiver (Table 7-12 on page 179) • Separate ground plane for GNDQ pin and power plane for VMV pin are used for input buffer to reduce output-induced noise.
Differential Receiver Features—250K through 1M Gate Devices	• Separate ground plane for GNDQ pin and power plane for VMV pin are used for input buffer to reduce output-induced noise.
CMOS-Style LVDS, B-LVDS, M-LVDS, or LVPECL Transmitter	• Two I/Os and external resistors are used to provide a CMOS-style LVDS, DDR LVDS, B-LVDS, and M-LVDS/LVPECL transmitter solution. • High slew rate • Weak pull-up and pull-down resistors • Programmable output loading

Table 7-6 • **Maximum I/O Frequency for Single-Ended and Differential I/Os in All Banks in IGLOO and ProASIC Devices (maximum drive strength and high slew selected)**

	Maximum Performance		
Specification	**ProASIC3**	**IGLOO V2 or V5 Devices, 1.5 V DC Core Supply Voltage**	**IGLOO V2, 1.2 V DC Core Supply Voltage**
LVTTL/LVCMOS 3.3 V	200 MHz	180 MHz	TBD
LVCMOS 2.5 V	250 MHz	230 MHz	TBD
LVCMOS 1.8 V	200 MHz	180 MHz	TBD
LVCMOS 1.5 V	130 MHz	120 MHz	TBD
PCI	200 MHz	180 MHz	TBD
PCI-X	200 MHz	180 MHz	TBD
LVDS	350 MHz	300 MHz	TBD
LVPECL	350 MHz	300 MHz	TBD

I/O Architecture

I/O Tile

The I/O tile provides a flexible, programmable structure for implementing a large number of I/O standards. In addition, the registers available in the I/O tile can be used to support high-performance register inputs and outputs, with register enable if desired (Figure 7-2). The registers can also be used to support the JESD-79C Double Data Rate (DDR) standard within the I/O structure (see the "DDR for Microsemi’s Low Power Flash Devices" section on page 219 for more information). In addition, the registers available in the I/O tile can be used to support high-performance register inputs and outputs, with register enable if desired (Figure 7-2).

As depicted in Figure 7-2, all I/O registers share one CLR port. The output register and output enable register share one CLK port.

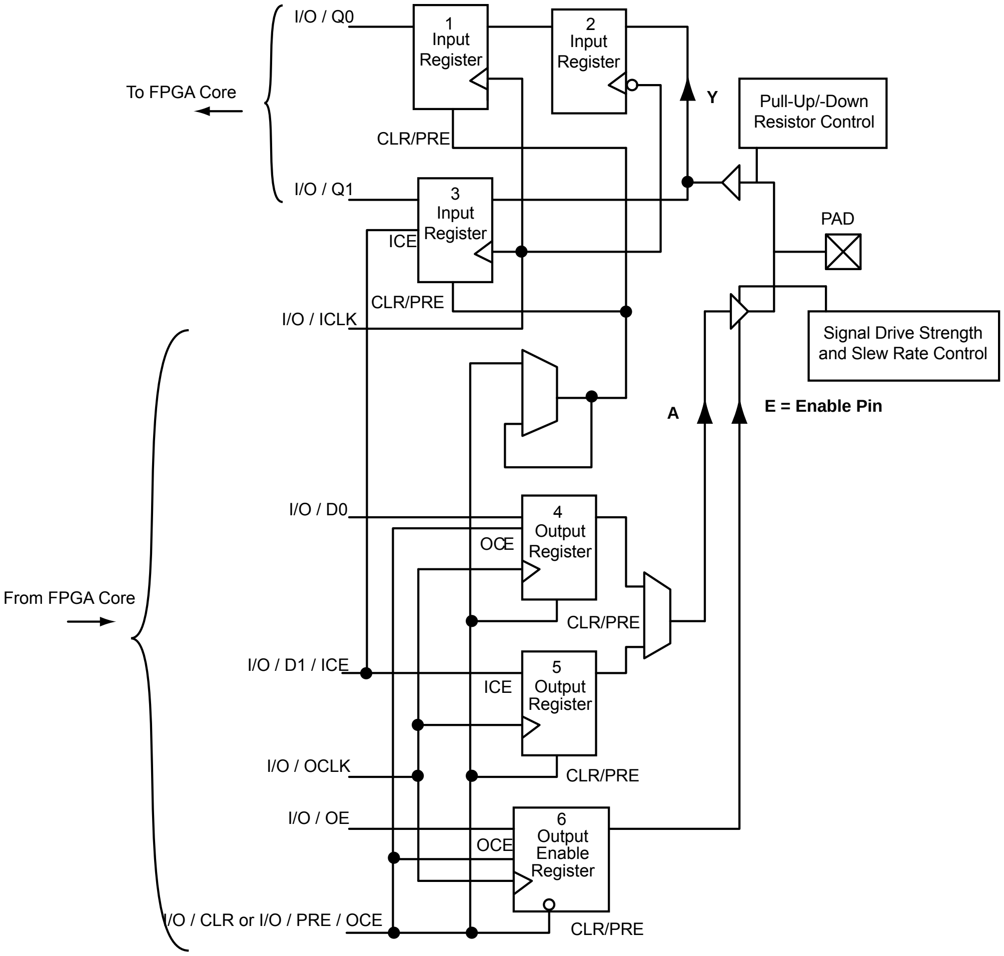


Figure 7-2 • **DDR Configured I/O Block Logical Representation**

I/O Bank Structure

Low power flash device I/Os are divided into multiple technology banks. The number of banks is device-dependent. The IGLOOe, ProASIC3EL, and ProASIC3E devices have eight banks (two per side); and IGLOO, ProASIC3L, and ProASIC3 devices have two to four banks. Each bank has its own V**CCI** power supply pin. Multiple I/O standards can co-exist within a single I/O bank.

In IGLOOe, ProASIC3EL, and ProASIC3E devices, each I/O bank is subdivided into VREF minibanks. These are used by voltage-referenced I/Os. VREF minibanks contain 8 to 18 I/Os. All I/Os in a given minibank share a common VREF line (only one VREF pin is needed per VREF minibank). Therefore, if an I/O in a VREF minibank is configured as a VREF pin, the remaining I/Os in that minibank will be able to use the voltage assigned to that pin. If the location of the VREF pin is selected manually in the software, the user must satisfy VREF rules (refer to the "I/O Software Control in Low Power Flash Devices" section on page 199). If the user does not pick the VRE$_F$ pin manually, the software automatically assigns it.

Figure 7-3 is a snapshot of a section of the I/O ring, showing the basic elements of an I/O tile, as viewed from the Designer place-and-route tool's MultiView Navigator (MVN).

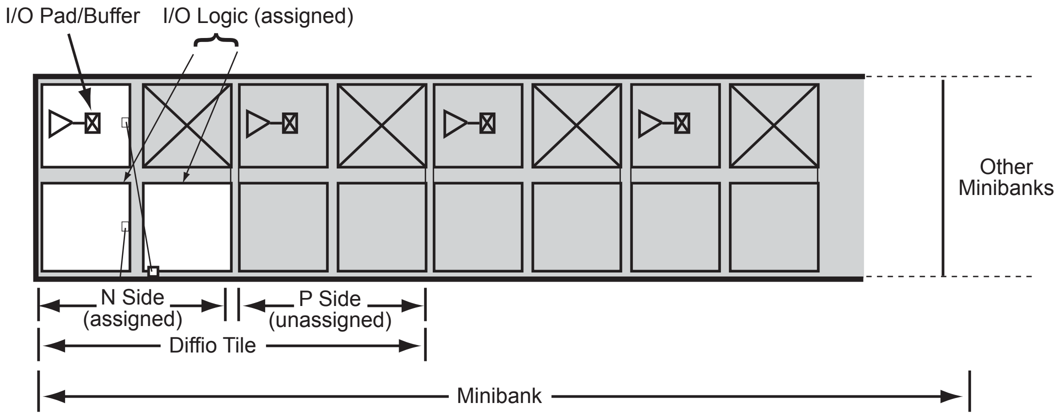


Figure 7-3 **• Snapshot of an I/O Tile**

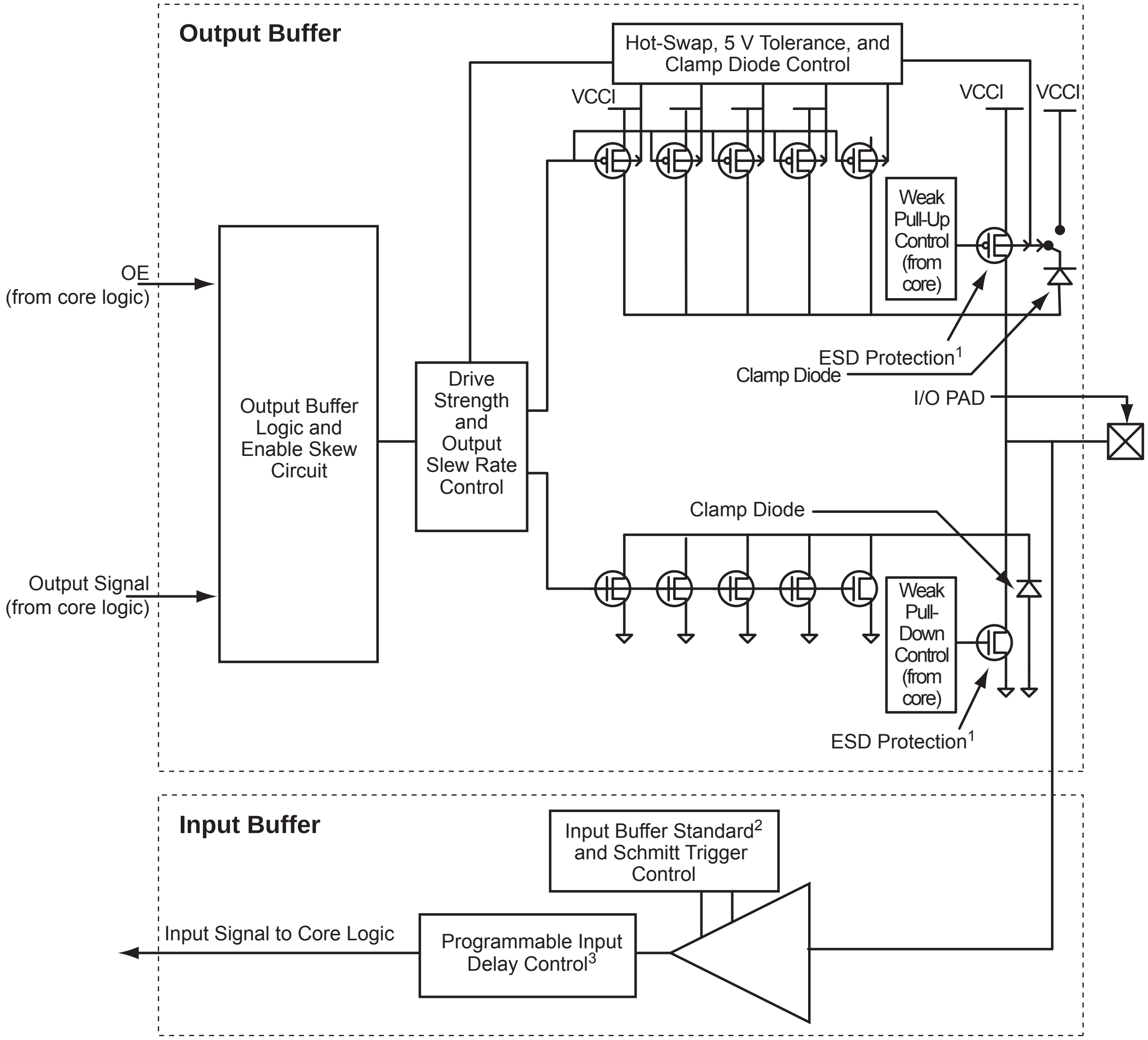
Low power flash device I/Os are implemented using two tile types: I/O and differential I/O (diffio).

The diffio tile is built up using two I/O tiles, which form an I/O pair (P side and N side). These I/O pairs are used according to differential I/O standards. Both the P and N sides of the diffio tile include an I/O buffer and two I/O logic blocks (auxiliary and main logic).

Every minibank (E devices only) is built up from multiple diffio tiles. The number of the minibank depends on the different-size dies. Refer to the "I/O Architecture" section on page 167 for an illustration of the minibank structure.

Figure 7-4 on page 169 shows a simplified diagram of the I/O buffer circuitry. The Output Enable signal (OE) enables the output buffer to pass the signal from the core logic to the pin. The output buffer contains ESD protection circuitry, an n-channel transistor that shunts all ESD surges (up to the limit of the device ESD specification) to GND. This transistor also serves as an output pull-down resistor.

Each output buffer also contains programmable slew rate, drive strength, programmable power-up state (pull-up/-down resistor), hot-swap, 5 V tolerance, and clamp diode control circuitry. Multiple flash switches (not shown in Figure 7-4 on page 169) are programmed by user selections in the software to activate different I/O features.


Output Buffer
Hot-Swap, 5 V Tolerance, and Clamp Diode Control
VCCI
VCCI
VCCI
Weak Pull-Up Control (from core)
OE (from core logic)
Output Buffer Logic and Enable Skew Circuit
Drive Strength and Output Slew Rate Control
ESD Protection1
Clamp Diode
I/O PAD
Clamp Diode
Output Signal (from core logic)
Weak Pull-Down Control (from core)
ESD Protection1
Input Buffer
Input Buffer Standard2 and Schmitt Trigger Control
Input Signal to Core Logic
Programmable Input Delay Control3

Notes:

1. *All NMOS transistors connected to the I/O pad serve as ESD protection.*
2. *See Table 7-2 on page 163 for available I/O standards.*
3. *Programmable input delay is applicable only to ProASIC3EL and RT ProASIC3 devices.*

***Figure 7-4* • Simplified I/O Buffer Circuitry**

I/O Registers

Each I/O module contains several input, output, and enable registers. Refer to Figure 7-4 for a simplified representation of the I/O block. The number of input registers is selected by a set of switches (not shown in Figure 7-2 on page 167) between registers to implement single-ended or differential data transmission to and from the FPGA core. The Designer software sets these switches for the user. A common CLR/PRE signal is employed by all I/O registers when I/O register combining is used. Input Register 2 does not have a CLR/PRE pin, as this register is used for DDR implementation. The I/O register combining must satisfy certain rules.

I/O Standards

Single-Ended Standards

These I/O standards use a push-pull CMOS output stage with a voltage referenced to system ground to designate logical states. The input buffer configuration, output drive, and I/O supply voltage (VCCI) vary among the I/O standards (Figure 7-5).

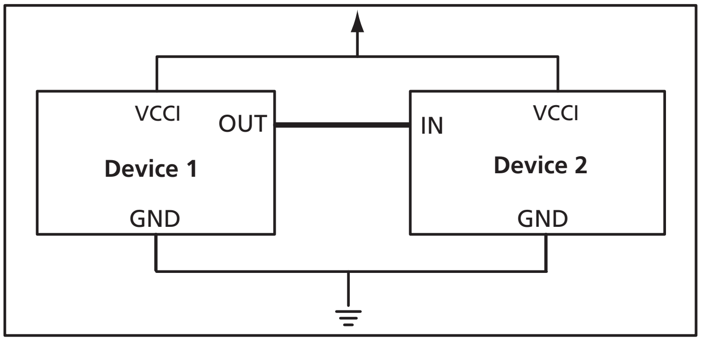


Figure 7-5 **• Single-Ended I/O Standard Topology**

The advantage of these standards is that a common ground can be used for multiple I/Os. This simplifies board layout and reduces system cost. Their low-edge-rate (*dv*/*dt*) data transmission causes less electromagnetic interference (EMI) on the board. However, they are not suitable for high-frequency (>200 MHz) switching due to noise impact and higher power consumption.

LVTTL (Low-Voltage TTL)

This is a general-purpose standard (EIA/JESD8-B) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. The LVTTL output buffer can have up to six different programmable drive strengths. The default drive strength is 12 mA. VCCI is 3.3 V. Refer to "I/O Programmable Features" on page 174 for details.

LVCMOS (Low-Voltage CMOS)

The low power flash devices provide four different kinds of LVCMOS: LVCMOS 3.3 V, LVCMOS 2.5 V, LVCMOS 1.8 V, and LVCMOS 1.5 V. LVCMOS 3.3 V is an extension of the LVCMOS standard (JESD8-B–compliant) used for general-purpose 3.3 V applications.

LVCMOS 2.5 V is an extension of the LVCMOS standard (JESD8-5–compliant) used for general-purpose 2.5 V applications.

There is yet another standard supported by IGLOO and ProASIC3 devices (except A3P030): LVCMOS 2.5/5.0 V. This standard is similar to LVCMOS 2.5 V, with the exception that it can support up to 3.3 V on the input side (2.5 V output drive).

LVCMOS 1.8 V is an extension of the LVCMOS standard (JESD8-7–compliant) used for general-purpose 1.8 V applications. LVCMOS 1.5 V is an extension of the LVCMOS standard (JESD8-11–compliant) used for general-purpose 1.5 V applications.

The VCCI values for these standards are 3.3 V, 2.5 V, 1.8 V, and 1.5 V, respectively. Like LVTTL, the output buffer has up to seven different programmable drive strengths (2, 4, 6, 8, 12, 16, and 24 mA). Refer to "I/O Programmable Features" on page 174 for details.

3.3 V PCI (Peripheral Component Interface)

This standard specifies support for both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5 V–compliant for low power flash devices. It does not have programmable drive strength.

3.3 V PCI-X (Peripheral Component Interface Extended)

An enhanced version of the PCI specification, 3.3 V PCI-X can support higher average bandwidths; it increases the speed that data can move within a computer from 66 MHz to 133 MHz. It is backward-compatible, which means devices can operate at conventional PCI frequencies (33 MHz and 66 MHz). PCI-X is more fault-tolerant than PCI. It also does not have programmable drive strength.

Voltage-Referenced Standards

I/Os using these standards are referenced to an external reference voltage (V_{REF}) and are supported on E devices only.

HSTL Class I and II (High-Speed Transceiver Logic)

These are general-purpose, high-speed 1.5 V bus standards (EIA/JESD 8-6) for signaling between integrated circuits. The signaling range is 0 V to 1.5 V, and signals can be either single-ended or differential. HSTL requires a differential amplifier input buffer and a push-pull output buffer. The reference voltage (VREF) is 0.75 V. These standards are used in the memory bus interface with data switching capability of up to 400 MHz. The other advantages of these standards are low power and fewer EMI concerns.

HSTL has four classes, of which low power flash devices support Class I and II. These classes are defined by standard EIA/JESD 8-6 from the Electronic Industries Alliance (EIA):

- Class I – Unterminated or symmetrically parallel-terminated
- Class II – Series-terminated
- Class III – Asymmetrically parallel-terminated
- Class IV – Asymmetrically double-parallel-terminated

SSTL2 Class I and II (Stub Series Terminated Logic 2.5 V)

These are general-purpose 2.5 V memory bus standards (JESD 8-9) for driving transmission lines, designed specifically for driving the DDR SDRAM modules used in computer memory. SSTL2 requires a differential amplifier input buffer and a push-pull output buffer. The reference voltage (VREF) is 1.25 V.

SSTL3 Class I and II (Stub Series Terminated Logic 3.3 V)

These are general-purpose 3.3 V memory bus standards (JESD 8-8) for driving transmission lines. SSTL3 requires a differential amplifier input buffer and a push-pull output buffer. The reference voltage (VREF) is 1.5 V.

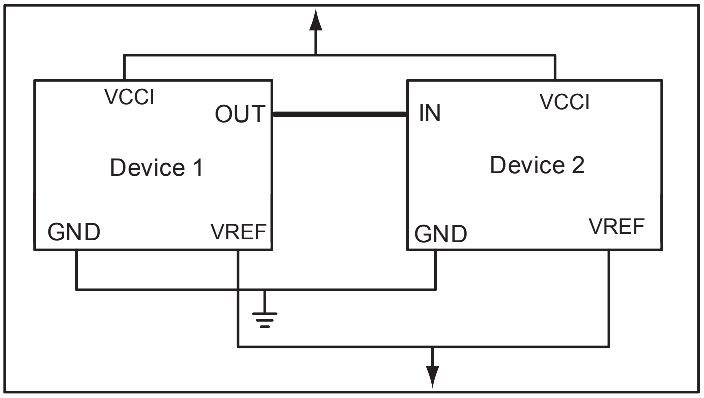


Figure 7-6 • **SSTL and HSTL Topology**

GTL 2.5 V (Gunning Transceiver Logic 2.5 V)

This is a low power standard (JESD 8.3) for electrical signals used in CMOS circuits that allows for low electromagnetic interference at high transfer speeds. It has a voltage swing between 0.4 V and 1.2 V and typically operates at speeds of between 20 and 40 MHz. VCCI must be connected to 2.5 V. The reference voltage (VREF) is 0.8 V.

GTL 3.3 V (Gunning Transceiver Logic 3.3 V)

This is the same as GTL 2.5 V above, except VCCI must be connected to 3.3 V.

GTL+ (Gunning Transceiver Logic Plus)

This is an enhanced version of GTL that has defined slew rates and higher voltage levels. It requires a differential amplifier input buffer and an open-drain output buffer. Even though the output is open-drain, VCCI must be connected to either 2.5 V or 3.3 V. The reference voltage (VREF) is 1 V.

Differential Standards

These standards require two I/Os per signal (called a "signal pair"). Logic values are determined by the potential difference between the lines, not with respect to ground. This is why differential drivers and receivers have much better noise immunity than single-ended standards. The differential interface standards offer higher performance and lower power consumption than their single-ended counterparts. Two I/O pins are used for each data transfer channel. Both differential standards require resistor termination.

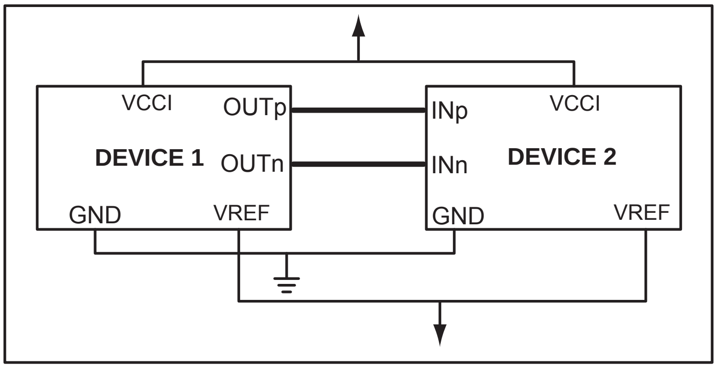


Figure 7-7 • **Differential Topology**

LVPECL (Low-Voltage Positive Emitter Coupled Logic)

LVPECL requires that one data bit be carried through two signal lines; therefore, two pins are needed per input or output. It also requires external resistor termination. The voltage swing between the two signal lines is approximately 850 mV. When the power supply is +3.3 V, it is commonly referred to as Low-Voltage PECL (LVPECL). Refer to the device datasheet for the full implementation of the LVPECL transmitter and receiver.

LVDS (Low-Voltage Differential Signal)

LVDS is a moderate-speed differential signaling system, in which the transmitter generates two different voltages that are compared at the receiver. LVDS uses a differential driver connected to a terminated receiver through a constant-impedance transmission line. It requires that one data bit be carried through two signal lines; therefore, the user will need two pins per input or output. It also requires external resistor termination. The voltage swing between the two signal lines is approximately 350 mV. VCCI is 2.5 V. Low power flash devices contain dedicated circuitry supporting a high-speed LVDS standard that has its own user specification. Refer to the device datasheet for the full implementation of the LVDS transmitter and receiver.

B-LVDS/M-LVDS

Bus LVDS (B-LVDS) refers to bus interface circuits based on LVDS technology. Multipoint LVDS (M-LVDS) specifications extend the LVDS standard to high-performance multipoint bus applications. Multidrop and multipoint bus configurations may contain any combination of drivers, receivers, and transceivers. Microsemi LVDS drivers provide the higher drive current required by B-LVDS and M-LVDS to accommodate the loading. The driver requires series terminations for better signal quality and to control voltage swing. Termination is also required at both ends of the bus, since the driver can be located anywhere on the bus. These configurations can be implemented using TRIBUF_LVDS and BIBUF_LVDS macros along with appropriate terminations. Multipoint designs using Microsemi LVDS macros can achieve up to 200 MHz with a maximum of 20 loads. A sample application is given in Figure 7-8. The input and output buffer delays are available in the LVDS sections in the datasheet.

Example: For a bus consisting of 20 equidistant loads, the terminations given in EQ 1 provide the required differential voltage, in worst-case industrial operating conditions, at the farthest receiver:

$$R_S = 60\ \Omega,\ R_T = 70\ \Omega,\ \text{given } Z_O = 50\ \Omega\ (2")\ \text{and } Z_{stub} = 50\ \Omega\ (\sim 1.5"). \qquad \textit{EQ 1}$$

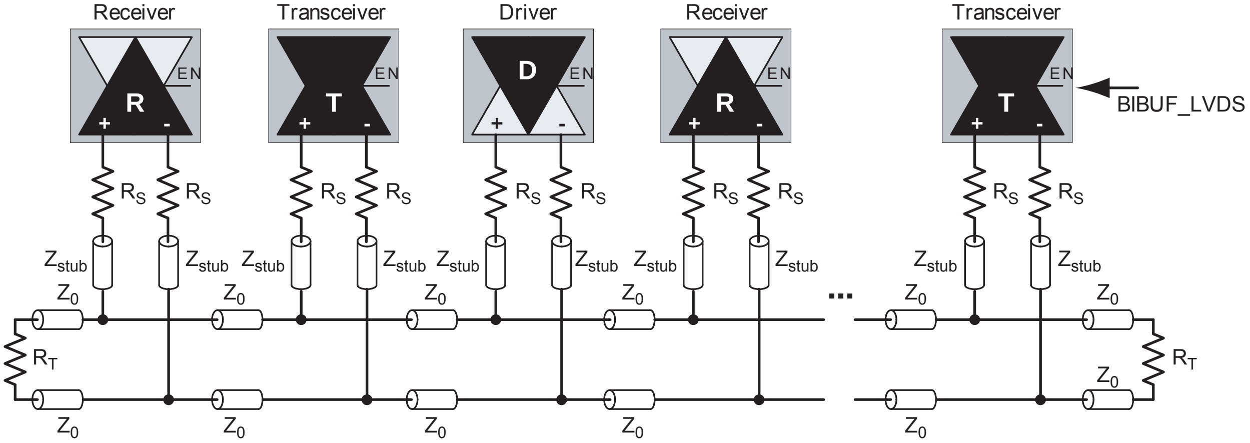


Figure 7-8 • **A B-LVDS/M-LVDS Multipoint Application Using LVDS I/O Buffers**

I/O Features

Low power flash devices support multiple I/O features that make board design easier. For example, an I/O feature like Schmitt Trigger in the ProASIC3E input buffer saves the board space that would be used by an external Schmitt trigger for a slow or noisy input signal. These features are also programmable for each I/O, which in turn gives flexibility in interfacing with other components. The following is a detailed description of all available features in low power flash devices.

I/O Programmable Features

Low power flash devices offer many flexible I/O features to support a wide variety of board designs. Some of the features are programmable, with a range for selection. Table 7-7 lists programmable I/O features and their ranges.

Table 7-7 **• Programmable I/O Features (user control via I/O Attribute Editor)**

Feature[1]	Description	Range
Slew Control	Output slew rate	HIGH, LOW
Output Drive (mA)	Output drive strength	2, 4, 6, 8, 12, 16, 24
Skew Control	Output tristate enable delay option	ON, OFF
Resistor Pull	Resistor pull circuit	Up, Down, None
Input Delay[2]	Input delay	OFF, 0–7
Schmitt Trigger	Schmitt trigger for input only	ON, OFF

Notes:

1. *Limitations of these features with respect to different devices are discussed in later sections.*
2. *Programmable input delay is applicable only to ProASIC3EL and RT ProASIC3 devices.*

Hot-Swap Support

A pull-up clamp diode must not be present in the I/O circuitry if the hot-swap feature is used. The 3.3 V PCI standard requires a pull-up clamp diode on the I/O, so it cannot be selected if hot-swap capability is required. The A3P030 device does not support 3.3 V PCI, so it is the only device in the ProASIC3 family that supports the hot-swap feature. All devices in the ProASIC3E family are hot-swappable. All standards except LVCMOS 2.5/5.0 V and 3.3 V PCI/PCI-X support the hot-swap feature.

The hot-swap feature appears as a read-only check box in the I/O Attribute Editor that shows whether an I/O is hot-swappable or not. Refer to the *"Power-Up/-Down Behavior of Low Power Flash Devices" section on page 387* for details on hot-swapping.

Hot-swapping (also called hot-plugging) is the operation of hot insertion or hot removal of a card in a powered-up system. The levels of hot-swap support and examples of related applications are described in Table 7-8 on page 175 to Table 7-11 on page 176. The I/Os also need to be configured in hot-insertion mode if hot-plugging compliance is required. The AGL030 and A3P030 devices have an I/O structure that allows the support of Level 3 and Level 4 hot-swap with only two levels of staging.

Table 7-8 • **Hot-Swap Level 1**

Description	Cold-swap
Power Applied to Device	No
Bus State	–
Card Ground Connection	–
Device Circuitry Connected to Bus Pins	–
Example Application	System and card with Microsemi FPGA chip are powered down, and the card is plugged into the system. Then the power supplies are turned on for the system but not for the FPGA on the card.
Compliance of IGLOO and ProASIC3 Devices	30 k gate devices: Compliant Other IGLOO/ProASIC3 devices: Compliant if bus switch used to isolate FPGA I/Os from rest of system IGLOOe/ProASIC3E devices: Compliant I/Os can but do not have to be set to hot-insertion mode.

Table 7-9 • **Hot-Swap Level 2**

Description	Hot-swap while reset
Power Applied to Device	Yes
Bus State	Held in reset state
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	–
Example Application	In the PCI hot-plug specification, reset control circuitry isolates the card busses until the card supplies are at their nominal operating levels and stable.
Compliance of IGLOO and ProASIC3 Devices	30 k gate devices, all IGLOOe/ProASIC3E devices: Compliant I/Os can but do not have to be set to hot-insertion mode. Other IGLOO/ProASIC3 devices: Compliant

Table 7-10 • **Hot-Swap Level 3**

Description	Hot-swap while bus idle
Power Applied to Device	Yes
Bus State	Held idle (no ongoing I/O processes during insertion/removal)
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	Must remain glitch-free during power-up or power-down
Example Application	Board bus shared with card bus is "frozen," and there is no toggling activity on the bus. It is critical that the logic states set on the bus signal not be disturbed during card insertion/removal.
Compliance of IGLOO and ProASIC3 Devices	30K gate devices, all IGLOOe/ProASIC3E devices: Compliant with two levels of staging (first: GND; second: all other pins) Other IGLOO/ProASIC3 devices: Compliant: Option A – Two levels of staging (first: GND; second: all other pins) together with bus switch on the I/Os Option B – Three levels of staging (first: GND; second: supplies; third: all other pins)

Table 7-11 • **Hot-Swap Level 4**

Description	Hot-swap on an active bus
Power Applied to Device	Yes
Bus State	Bus may have active I/O processes ongoing, but device being inserted or removed must be idle.
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	Must remain glitch-free during power-up or power-down
Example Application	There is activity on the system bus, and it is critical that the logic states set on the bus signal not be disturbed during card insertion/removal.
Compliance of IGLOO and ProASIC3 Devices	30K gate devices, all IGLOOe/ProASIC3E devices: Compliant with two levels of staging (first: GND; second: all other pins) Other IGLOO/ProASIC3 devices: Compliant: Option A – Two levels of staging (first: GND; second: all other pins) together with bus switch on the I/Os Option B – Three levels of staging (first: GND; second: supplies; third: all other pins)

IGLOO and ProASIC3

For boards and cards with three levels of staging, card power supplies must have time to reach their final values before the I/Os are connected. Pay attention to the sizing of power supply decoupling capacitors on the card to ensure that the power supplies are not overloaded with capacitance.

Cards with three levels of staging should have the following sequence:

- Grounds
- Powers
- I/Os and other pins

For Level 3 and Level 4 compliance with the 30K gate device, cards with two levels of staging should have the following sequence:

- Grounds
- Powers, I/Os, and other pins

Cold-Sparing Support

Cold-sparing refers to the ability of a device to leave system data undisturbed when the system is powered up, while the component itself is powered down, or when power supplies are floating.

The resistor value is calculated based on the decoupling capacitance on a given power supply. The RC constant should be greater than 3 µs.

To remove resistor current during operation, it is suggested that the resistor be disconnected (e.g., with an NMOS switch) from the power supply after the supply has reached its final value. Refer to the "Power-Up/-Down Behavior of Low Power Flash Devices" section on page 387 for details on cold-sparing.

Cold-sparing means that a subsystem with no power applied (usually a circuit board) is electrically connected to the system that is in operation. This means that all input buffers of the subsystem must present very high input impedance with no power applied so as not to disturb the operating portion of the system.

The 30 k gate devices fully support cold-sparing, since the I/O clamp diode is always off (see Table 7-12 on page 179). If the 30 k gate device is used in applications requiring cold-sparing, a discharge path from the power supply to ground should be provided. This can be done with a discharge resistor or a switched resistor. This is necessary because the 30K gate devices do not have built-in I/O clamp diodes.

For other IGLOO and ProASIC3 devices, since the I/O clamp diode is always active, cold-sparing can be accomplished either by employing a bus switch to isolate the device I/Os from the rest of the system or by driving each I/O pin to 0 V. If the resistor is chosen, the resistor value must be calculated based on decoupling capacitance on a given power supply on the board (this decoupling capacitance is in parallel with the resistor). The RC time constant should ensure full discharge of supplies before cold-sparing functionality is required. The resistor is necessary to ensure that the power pins are discharged to ground every time there is an interruption of power to the device.

IGLOOe and ProASIC3E devices support cold-sparing for all I/O configurations. Standards, such as PCI, that require I/O clamp diodes can also achieve cold-sparing compliance, since clamp diodes get disconnected internally when the supplies are at 0 V.

When targeting low power applications, I/O cold-sparing may add additional current if a pin is configured with either a pull-up or pull-down resistor and driven in the opposite direction. A small static current is induced on each I/O pin when the pin is driven to a voltage opposite to the weak pull resistor. The current is equal to the voltage drop across the input pin divided by the pull resistor. Refer to the "Detailed I/O DC Characteristics" section of the appropriate family datasheet for the specific pull resistor value for the corresponding I/O standard.

For example, assuming an LVTTL 3.3 V input pin is configured with a weak pull-up resistor, a current will flow through the pull-up resistor if the input pin is driven LOW. For LVTTL 3.3 V, the pull-up resistor is ~45 kΩ, and the resulting current is equal to 3.3 V / 45 kΩ = 73 µA for the I/O pin. This is true also when a weak pull-down is chosen and the input pin is driven HIGH. This current can be avoided by driving the input LOW when a weak pull-down resistor is used and driving it HIGH when a weak pull-up resistor is used.

This current draw can occur in the following cases:

- In Active and Static modes:
  - Input buffers with pull-up, driven Low
  - Input buffers with pull-down, driven High
  - Bidirectional buffers with pull-up, driven Low
  - Bidirectional buffers with pull-down, driven High
  - Output buffers with pull-up, driven Low
  - Output buffers with pull-down, driven High
  - Tristate buffers with pull-up, driven Low
  - Tristate buffers with pull-down, driven High
- In Flash*Freeze mode:
  - Input buffers with pull-up, driven Low
  - Input buffers with pull-down, driven High
  - Bidirectional buffers with pull-up, driven Low
  - Bidirectional buffers with pull-down, driven High

Electrostatic Discharge Protection

Low power flash devices are tested per JEDEC Standard JESD22-A114-B.

These devices contain clamp diodes at every I/O, global, and power pad. Clamp diodes protect all device pads against damage from ESD as well as from excessive voltage transients.

All IGLOO and ProASIC3 devices are tested to the Human Body Model (HBM) and the Charged Device Model (CDM).

Each I/O has two clamp diodes. One diode has its positive (P) side connected to the pad and its negative (N) side connected to VCCI. The second diode has its P side connected to GND and its N side connected to the pad. During operation, these diodes are normally biased in the off state, except when transient voltage is significantly above VCCI or below GND levels.

In 30K gate devices, the first diode is always off. In other devices, the clamp diode is always on and cannot be switched off.

By selecting the appropriate I/O configuration, the diode is turned on or off. Refer to Table 7-12 on page 179 for more information about the I/O standards and the clamp diode.

The second diode is always connected to the pad, regardless of the I/O configuration selected.

***Table 7-12* • I/O Hot-Swap and 5 V Input Tolerance Capabilities in IGLOO and ProASIC3 Devices**

	Clamp Diode [1]		Hot Insertion		5 V Input Tolerance [2]		
I/O Assignment	**AGL030 and A3P030**	**Other IGLOO and ProASIC3 Devices**	**AGL015 and AGL030**	**Other IGLOO Devices and All ProASIC3**	**AGL030 and A3P030**	**Other IGLOO and ProASIC3 Devices**	**Input and Output Buffer**
3.3 V LVTTL/LVCMOS	No	Yes	Yes	No	Yes [2]	Yes [2]	Enabled/Disabled
3.3 V PCI, 3.3 V PCI-X	N/A	Yes	N/A	No	N/A	Yes [2]	Enabled/Disabled
LVCMOS 2.5 V [5]	No	Yes	Yes	No	Yes [2]	Yes [4]	Enabled/Disabled
LVCMOS 2.5 V/5.0 V [6]	N/A	Yes	N/A	No	N/A	Yes [4]	Enabled/Disabled
LVCMOS 1.8 V	No	Yes	Yes	No	No	No	Enabled/Disabled
LVCMOS 1.5 V	No	Yes	Yes	No	No	No	Enabled/Disabled
Differential, LVDS/ B-LVDS/M-LVDS/LVPECL	N/A	Yes	N/A	No	N/A	No	Enabled/Disabled

Notes:

1. *The clamp diode is always off for the AGL030 and A3P030 device and always active for other IGLOO and ProASIC3 devices.*
2. *Can be implemented with an external IDT bus switch, resistor divider, or Zener with resistor.*
3. *Refer to Table 7-8 on page 175 to Table 7-11 on page 176 for device-compliant information.*
4. *Can be implemented with an external resistor and an internal clamp diode.*
5. *The LVCMOS 2.5 V I/O standard is supported by the 30 k gate devices only; select the LVCMOS25 macro.*
6. *The LVCMOS 2.5 V / 5.0 V I/O standard is supported by all IGLOO and ProASIC3 devices except 30K gate devices; select the LVCMOS5 macro.*

5 V Input and Output Tolerance

IGLOO and ProASIC3 devices are both 5 V-input– and 5 V–output–tolerant if certain I/O standards are selected. Table 7-5 on page 165 shows the I/O standards that support 5 V input tolerance. Only 3.3 V LVTTL/LVCMOS standards support 5 V output tolerance. Refer to the appropriate family datasheet for the detailed description and configuration information.

This feature is not shown in the I/O Attribute Editor.

5 V Input Tolerance

I/Os can support 5 V input tolerance when LVTTL 3.3 V, LVCMOS 3.3 V, LVCMOS 2.5 V, and LVCMOS 2.5 V / 5.0 V configurations are used (see Table 7-12 on page 179). There are four recommended solutions for achieving 5 V receiver tolerance (see Figure 7-9 on page 181 to Figure 7-12 on page 183 for details of board and macro setups). All the solutions meet a common requirement of limiting the voltage at the input to 3.6 V or less. In fact, the I/O absolute maximum voltage rating is 3.6 V, and any voltage above 3.6 V may cause long-term gate oxide failures.

Solution 1

The board-level design must ensure that the reflected waveform at the pad does not exceed the limits provided in the recommended operating conditions in the datasheet. This is a requirement to ensure long-term reliability.

This scheme will also work for a 3.3 V PCI/PCI-X configuration, but the internal diode should not be used for clamping, and the voltage must be limited by the two external resistors as explained below. Relying on the diode clamping would create an excessive pad DC voltage of 3.3 V + 0.7 V = 4 V.

This solution requires two board resistors, as demonstrated in Figure 7-9 on page 181. Here are some examples of possible resistor values (based on a simplified simulation model with no line effects and 10 Ω transmitter output resistance, where Rtx_out_high = (VCCI – VOH) / I_{OH} and Rtx_out_low = VOL / I_{OL}).

Example 1 (high speed, high current):

Rtx_out_high = Rtx_out_low = 10 Ω

R1 = 36 Ω (±5%), P(r1)min = 0.069 Ω

R2 = 82 Ω (±5%), P(r2)min = 0.158 Ω

Imax_tx = 5.5 V / (82 × 0.95 + 36 × 0.95 + 10) = 45.04 mA

t_{RISE} = t_{FALL} = 0.85 ns at C_pad_load = 10 pF (includes up to 25% safety margin)

t_{RISE} = t_{FALL} = 4 ns at C_pad_load = 50 pF (includes up to 25% safety margin)

Example 2 (low–medium speed, medium current):

Rtx_out_high = Rtx_out_low = 10 Ω

R1 = 220 Ω (±5%), P(r1)min = 0.018 Ω

R2 = 390 Ω (±5%), P(r2)min = 0.032 Ω

Imax_tx = 5.5 V / (220 × 0.95 + 390 × 0.95 + 10) = 9.17 mA

t_{RISE} = t_{FALL} = 4 ns at C_pad_load = 10 pF (includes up to 25% safety margin)

t_{RISE} = t_{FALL} = 20 ns at C_pad_load = 50 pF (includes up to 25% safety margin)

Other values of resistors are also allowed as long as the resistors are sized appropriately to limit the voltage at the receiving end to 2.5 V < Vin (rx) < 3.6 V when the transmitter sends a logic 1. This range of Vin_dc(rx) must be assured for any combination of transmitter supply (5 V ± 0.5 V), transmitter output resistance, and board resistor tolerances.

Temporary overshoots are allowed according to the overshoot and undershoot table in the datasheet.

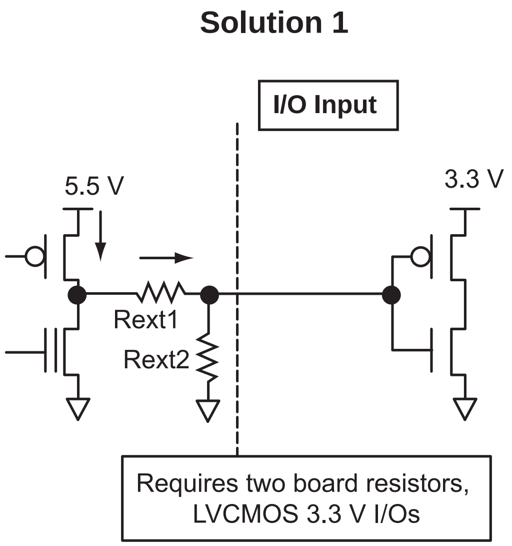


Figure 7-9 • **Solution 1**

Solution 2

The board-level design must ensure that the reflected waveform at the pad does not exceed the voltage overshoot/undershoot limits provided in the datasheet. This is a requirement to ensure long-term reliability.

This scheme will also work for a 3.3 V PCI/PCI-X configuration, but the internal diode should not be used for clamping, and the voltage must be limited by the external resistors and Zener, as shown in Figure 7-10. Relying on the diode clamping would create an excessive pad DC voltage of 3.3 V + 0.7 V = 4 V.

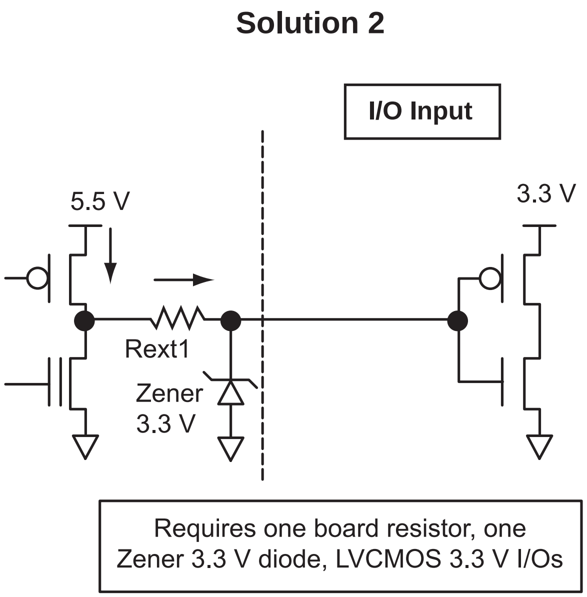

Solution 2
I/O Input
5.5 V
3.3 V
Rext1
Zener
3.3 V
Requires one board resistor, one Zener 3.3 V diode, LVCMOS 3.3 V I/Os

Figure 7-10 • **Solution 2**

Solution 3

The board-level design must ensure that the reflected waveform at the pad does not exceed the voltage overshoot/undershoot limits provided in the datasheet. This is a requirement to ensure long-term reliability.

This scheme will also work for a 3.3 V PCI/PCI-X configuration, but the internal diode should not be used for clamping, and the voltage must be limited by the bus switch, as shown in Figure 7-11. Relying on the diode clamping would create an excessive pad DC voltage of 3.3 V + 0.7 V = 4 V.

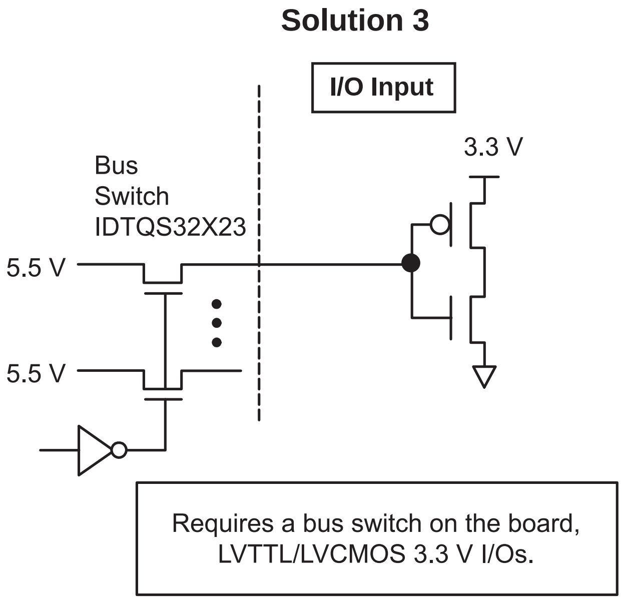


Figure 7-11 • **Solution 3**

Solution 4

The board-level design must ensure that the reflected waveform at the pad does not exceed the voltage overshoot/undershoot limits provided in the datasheet. This is a requirement to ensure long-term reliability.

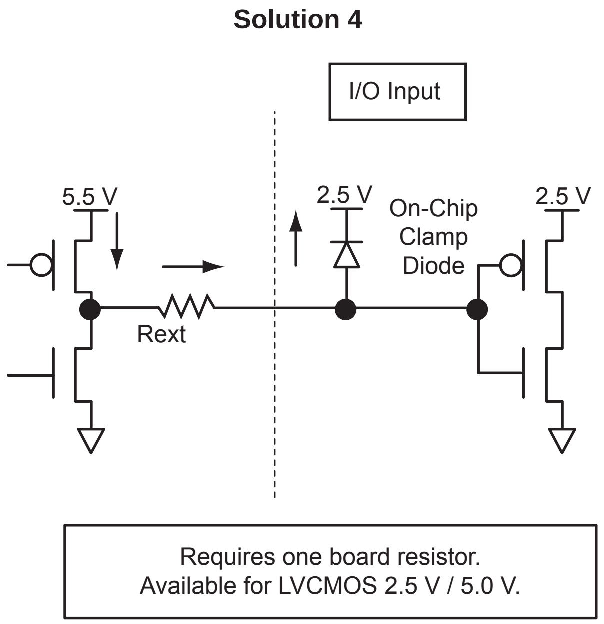

Solution 4
I/O Input
5.5 V
2.5 V
On-Chip
Clamp
Diode
2.5 V
Rext
Requires one board resistor.
Available for LVCMOS 2.5 V / 5.0 V.

Figure 7-12 • **Solution 4**

***Table 7-13* • Comparison Table for 5 V–Compliant Receiver Solutions**

Solution	Board Components	Speed	Current Limitations
1	Two resistors	Low to High[1]	Limited by transmitter's drive strength
2	Resistor and Zener 3.3 V	Medium	Limited by transmitter's drive strength
3	Bus switch	High	N/A

4	Minimum resistor value[2,3,4,5]
	R = 47 Ω at T_J = 70°C
	R = 150 Ω at T_J = 85°C
	R = 420 Ω at T_J = 100°C
		Medium	Maximum diode current at 100% duty cycle, signal constantly at 1
			52.7 mA at T_J = 70°C / 10-year lifetime
			16.5 mA at T_J = 85°C / 10-year lifetime
			5.9 mA at T_J = 100°C / 10-year lifetime
			For duty cycles other than 100%, the currents can be increased by a factor of 1 / (duty cycle).
			Example: 20% duty cycle at 70°C
			Maximum current = (1 / 0.2) × 52.7 mA = 5 × 52.7 mA = 263.5 mA

Notes:

1. *Speed and current consumption increase as the board resistance values decrease.*
2. *Resistor values ensure I/O diode long-term reliability.*
3. *At 70°C, customers could still use 420 Ω on every I/O.*
4. *At 85°C, a 5 V solution on every other I/O is permitted, since the resistance is lower (150 Ω) and the current is higher. Also, the designer can still use 420 Ω and use the solution on every I/O.*
5. *At 100°C, the 5 V solution on every I/O is permitted, since 420 Ω are used to limit the current to 5.9 mA.*

5 V Output Tolerance

IGLOO and ProASIC3 I/Os must be set to 3.3 V LVTTL or 3.3 V LVCMOS mode to reliably drive 5 V TTL receivers. It is also critical that there be NO external I/O pull-up resistor to 5 V, since this resistor would pull the I/O pad voltage beyond the 3.6 V absolute maximum value and consequently cause damage to the I/O.

When set to 3.3 V LVTTL or 3.3 V LVCMOS mode, the I/Os can directly drive signals into 5 V TTL receivers. In fact, VOL = 0.4 V and VOH = 2.4 V in both 3.3 V LVTTL and 3.3 V LVCMOS modes exceeds the VIL = 0.8 V and VIH = 2 V level requirements of 5 V TTL receivers. Therefore, level 1 and level 0 will be recognized correctly by 5 V TTL receivers.

Schmitt Trigger

A Schmitt trigger is a buffer used to convert a slow or noisy input signal into a clean one before passing it to the FPGA. Using Schmitt trigger buffers guarantees a fast, noise-free input signal to the FPGA.

The Schmitt trigger is available for the LVTTL, LVCMOS, and 3.3 V PCI I/O standards.

This feature can be implemented by using a Physical Design Constraints (PDC) command (Table 7-5 on page 165) or by selecting a check box in the I/O Attribute Editor in Designer. The check box is cleared by default.

Selectable Skew between Output Buffer Enable and Disable Times

Low power flash devices have a configurable skew block in the output buffer circuitry that can be enabled to delay output buffer assertion without affecting deassertion time. Since this skew block is only available for the OE signal, the feature can be used in tristate and bidirectional buffers. A typical 1.2 ns delay is added to the OE signal to prevent potential bus contention. Refer to the appropriate family datasheet for detailed timing diagrams and descriptions.

The skew feature is available for all I/O standards.

This feature can be implemented by using a PDC command (Table 7-5 on page 165) or by selecting a check box in the I/O Attribute Editor in Designer. The check box is cleared by default.

The configurable skew block is used to delay output buffer assertion (enable) without affecting deassertion (disable) time.

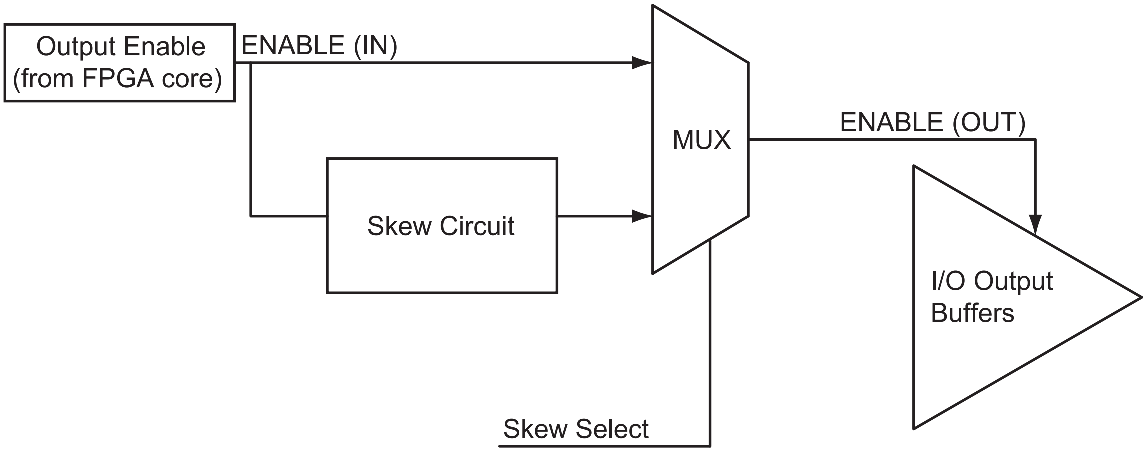


***Figure 7-13* • Block Diagram of Output Enable Path**

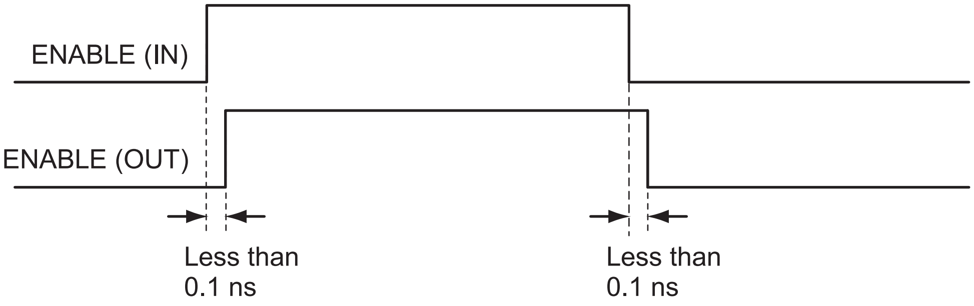


***Figure 7-14* • Timing Diagram (option 1: bypasses skew circuit)**

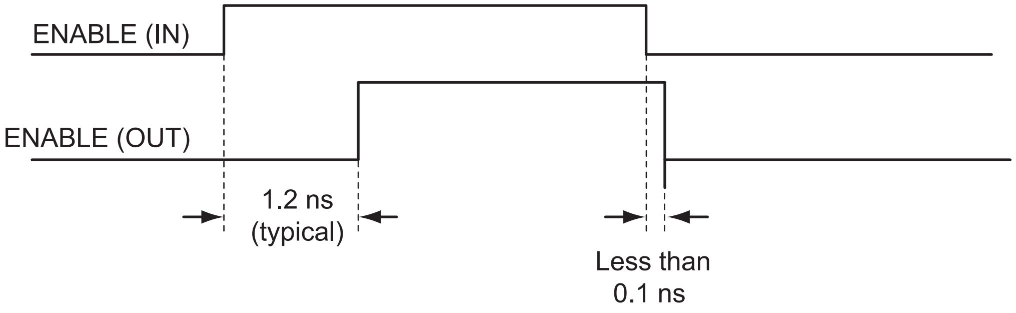


Figure 7-15 **• Timing Diagram (option 2: enables skew circuit)**

At the system level, the skew circuit can be used in applications where transmission activities on bidirectional data lines need to be coordinated. This circuit, when selected, provides a timing margin that can prevent bus contention and subsequent data loss and/or transmitter over-stress due to transmitter-to-transmitter current shorts. Figure 7-16 presents an example of the skew circuit implementation in a bidirectional communication system. Figure 7-17 on page 187 shows how bus contention is created, and Figure 7-18 on page 187 shows how it can be avoided with the skew circuit.

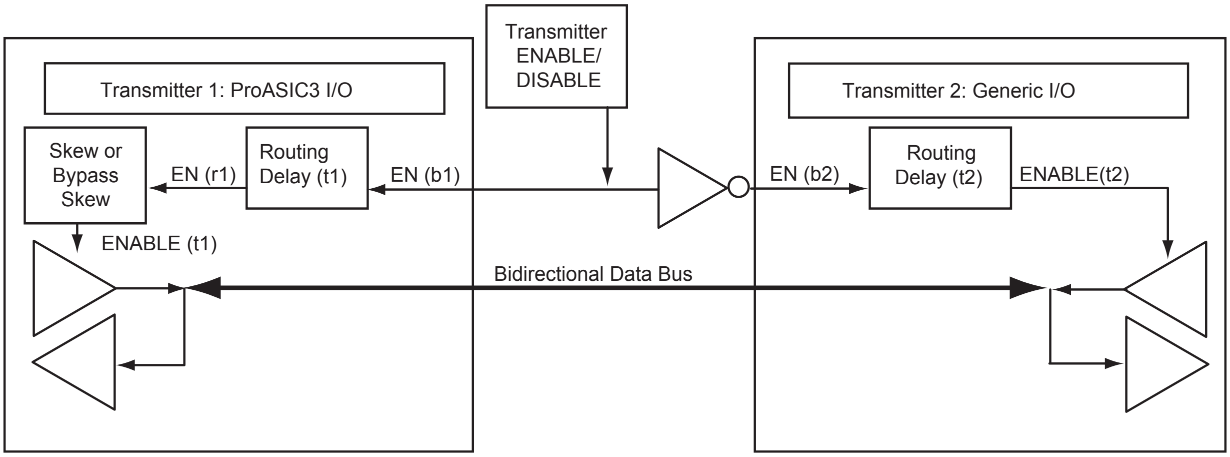


Figure 7-16 **• Example of Implementation of Skew Circuits in Bidirectional Transmission Systems Using IGLOO or ProASIC3 Devices**

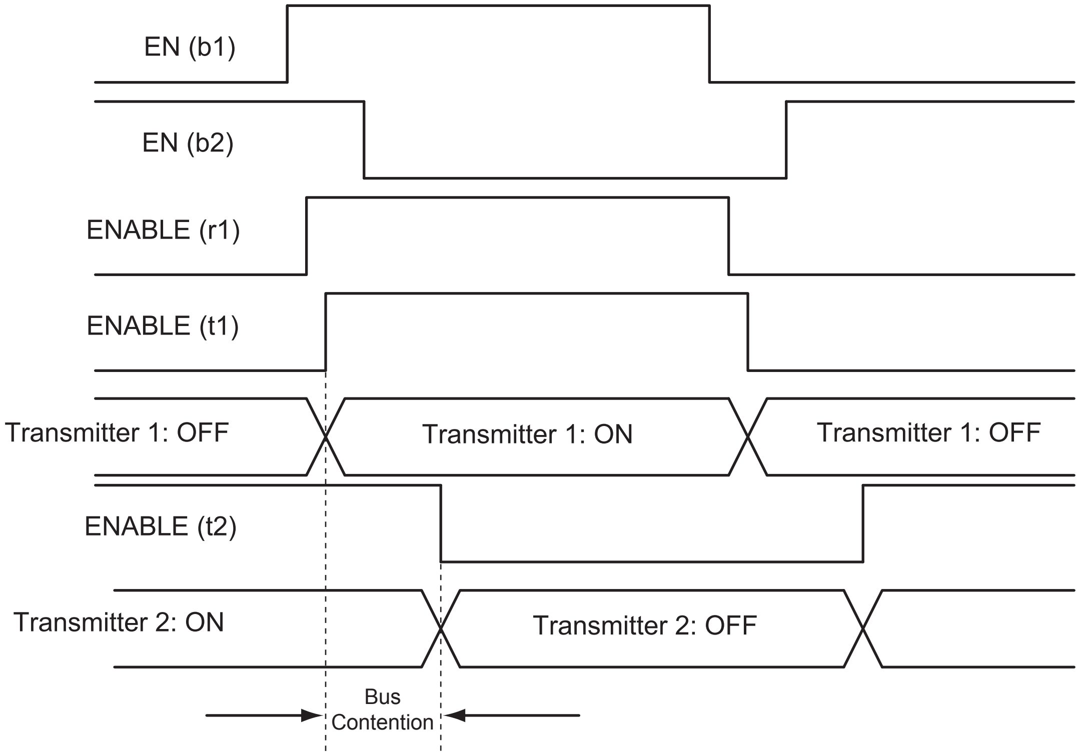


Figure 7-17 • **Timing Diagram (bypasses skew circuit)**

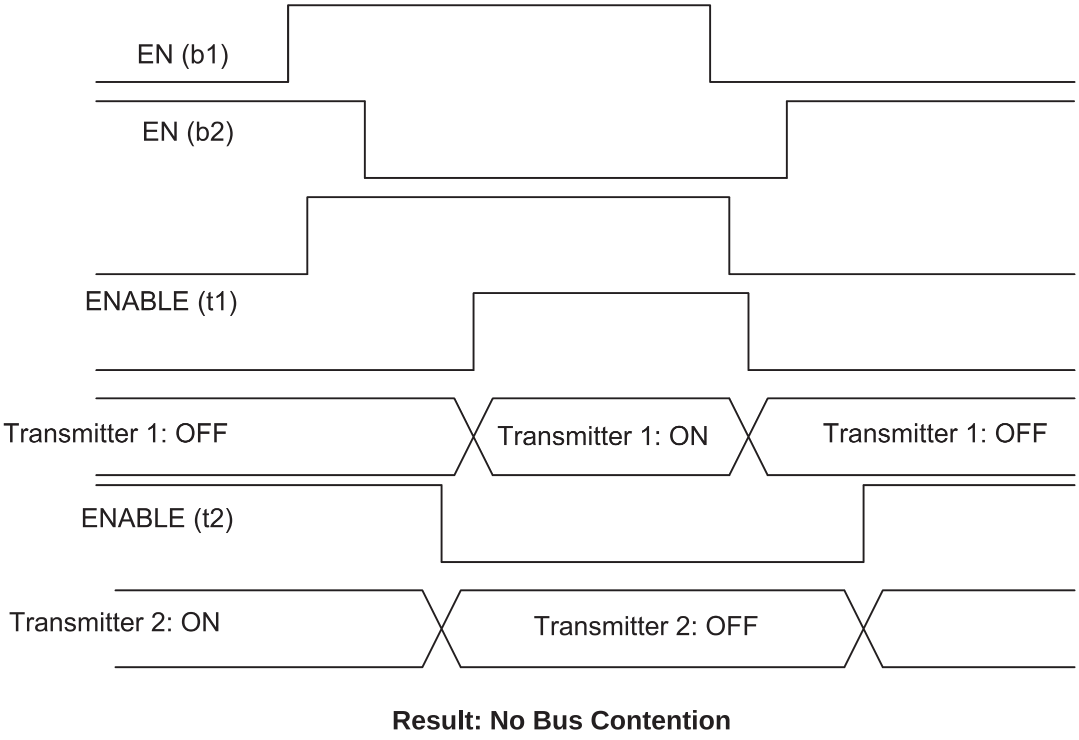


Figure 7-18 • **Timing Diagram (with skew circuit selected)**

I/O Register Combining

Every I/O has several embedded registers in the I/O tile that are close to the I/O pads. Rather than using the internal register from the core, the user has the option of using these registers for faster clock-to-out timing, and external hold and setup. When combining these registers at the I/O buffer, some architectural rules must be met. Provided these rules are met, the user can enable register combining globally during Compile (as shown in the "Compiling the Design" section on page 209).

This feature is supported by all I/O standards.

Rules for Registered I/O Function

1. The fanout between an I/O pin (D, Y, or E) and a register must be equal to one for combining to be considered on that pin.
2. All registers (Input, Output, and Output Enable) connected to an I/O must share the same clear or preset function:
   - If one of the registers has a CLR pin, all the other registers that are candidates for combining in the I/O must have a CLR pin.
   - If one of the registers has a PRE pin, all the other registers that are candidates for combining in the I/O must have a PRE pin.
   - If one of the registers has neither a CLR nor a PRE pin, all the other registers that are candidates for combining must have neither a CLR nor a PRE pin.
   - If the clear or preset pins are present, they must have the same polarity.
   - If the clear or preset pins are present, they must be driven by the same signal (net).
3. Registers connected to an I/O on the Output and Output Enable pins must have the same clock and enable function:
   - Both the Output and Output Enable registers must have an E pin (clock enable), or none at all.
   - If the E pins are present, they must have the same polarity. The CLK pins must also have the same polarity.

In some cases, the user may want registers to be combined with the input of a bibuf while maintaining the output as-is. This can be achieved by using PDC commands as follows:

```
set_io <signal name> -REGISTER yes ------register will combine
set_preserve <signal name> ----register will not combine
```

Weak Pull-Up and Weak Pull-Down Resistors

IGLOO and ProASIC3 devices support optional weak pull-up and pull-down resistors on each I/O pin. When the I/O is pulled up, it is connected to the VCCI of its corresponding I/O bank. When it is pulled down, it is connected to GND. Refer to the datasheet for more information.

For low power applications, configuration of the pull-up or pull-down of the I/O can be used to set the I/O to a known state while the device is in Flash*Freeze mode. Refer to the "Flash*Freeze Technology and Low Power Modes in IGLOO and ProASIC3L Devices" chapter of the *IGLOO FPGA Fabric User's Guide* or *ProASIC3L FPGA Fabric User's Guide* for more information.

The Flash*Freeze (FF) pin cannot be configured with a weak pull-down or pull-up I/O attribute, as the signal needs to be driven at all times.

Output Slew Rate Control

The slew rate is the amount of time an input signal takes to get from logic Low to logic High or vice versa.

It is commonly defined as the propagation delay between 10% and 90% of the signal's voltage swing. Slew rate control is available for the output buffers of low power flash devices. The output buffer has a programmable slew rate for both HIGH-to-LOW and LOW-to-HIGH transitions. Slew rate control is available for LVTTL, LVCMOS, and PCI-X I/O standards. The other I/O standards have a preset slew value.

The slew rate can be implemented by using a PDC command (Table 7-5 on page 165), setting it "High" or "Low" in the I/O Attribute Editor in Designer, or instantiating a special I/O macro. The default slew rate value is "High."

IGLOO and ProASIC3 devices support output slew rate control: high and low. Microsemi recommends the high slew rate option to minimize the propagation delay. This high-speed option may introduce noise into the system if appropriate signal integrity measures are not adopted. Selecting a low slew rate reduces this kind of noise but adds some delays in the system. Low slew rate is recommended when bus transients are expected.

Output Drive

The output buffers of IGLOO and ProASIC3 devices can provide multiple drive strengths to meet signal integrity requirements. The LVTTL and LVCMOS (except 1.2 V LVCMOS) standards have selectable drive strengths. Other standards have a preset value.

Drive strength should also be selected according to the design requirements and noise immunity of the system.

The output slew rate and multiple drive strength controls are available in LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V, LVCMOS 2.5 V / 5.0 V input, LVCMOS 1.8 V, and LVCMOS 1.5 V. All other I/O standards have a high output slew rate by default.

For 30 k gate devices, refer to Table 7-14. For other ProASIC3 and IGLOO devices, refer to Table 7-15 through Table 7-16 on page 189 for more information about the slew rate and drive strength specification. Refer to Table 7-4 on page 164 for I/O bank type definitions.

There will be a difference in timing between the Standard Plus I/O banks and the Advanced I/O banks. Refer to the I/O timing tables in the datasheet for the standards supported by each device.

***Table 7-14* • IGLOO and ProASIC3 Output Drive and Slew for Standard I/O Bank Type (for 30 k gate devices)**

I/O Standards	2 mA	4 mA	6 mA	8 mA	Slew	
LVTTL/LVCMOS 3.3 V	✓	✓	✓	✓	High	Low
LVCMOS 2.5 V	✓	✓	✓	✓	High	Low
LVCMOS 1.8 V	✓	✓	–	–	High	Low
LVCMOS 1.5 V	✓	–	–	–	High	Low

***Table 7-15* • IGLOO and ProASIC3 Output Drive and Slew for Standard Plus I/O Bank Type**

I/O Standards	2 mA	4 mA	6 mA	8 mA	12 mA	16 mA	Slew	
LVTTL	✓	✓	✓	✓	✓	✓	High	Low
LVCMOS 3.3 V	✓	✓	✓	✓	✓	✓	High	Low
LVCMOS 2.5 V	✓	✓ *	✓	✓ *	✓	–	High	Low
LVCMOS 1.8 V	✓	✓	✓	✓	–	–	High	Low
LVCMOS 1.5 V	✓	✓	–	–	–	–	High	Low

*Note: *Not available in Automotive devices.*

***Table 7-16* • IGLOO and ProASIC3 Output Drive and Slew for Advanced I/O Bank Type**

I/O Standards	2 mA	4 mA	6 mA	8 mA	12 mA	16 mA	24 mA	Slew	
LVTTL	✓	✓	✓	✓	✓	✓	✓	High	Low
LVCMOS 3.3 V	✓	✓	✓	✓	✓	✓	✓	High	Low
LVCMOS 2.5 V	✓	✓ *	✓	✓ *	✓	✓	✓	High	Low
LVCMOS 2.5/5.0 V	✓	✓ *	✓	✓ *	✓	✓	✓	High	Low
LVCMOS 1.8 V	✓	✓	✓	✓	✓	✓	–	High	Low
LVCMOS 1.5 V	✓	✓	✓	✓	✓	–	–	High	Low

Note: Not available in Automotive devices.

Simultaneously Switching Outputs (SSOs) and Printed Circuit Board Layout

Each I/O voltage bank has a separate ground and power plane for input and output circuits (VMV/GNDQ for input buffers and VCCI/GND for output buffers). This isolation is necessary to minimize simultaneous switching noise from the input and output (SSI and SSO). The switching noise (ground bounce and power bounce) is generated by the output buffers and transferred into input buffer circuits, and vice versa.

Since voltage bounce originates on the package inductance, the VMV and VCCI supplies have separate package pin assignments. For the same reason, GND and GNDQ also have separate pin assignments.

The VMV and VCCI pins must be shorted to each other on the board. Also, the GND and GNDQ pins must be shorted to each other on the board. This will prevent unwanted current draw from the power supply.

SSOs can cause signal integrity problems on adjacent signals that are not part of the SSO bus. Both inductive and capacitive coupling parasitics of bond wires inside packages and of traces on PCBs will transfer noise from SSO busses onto signals adjacent to those busses. Additionally, SSOs can produce ground bounce noise and VCCI dip noise. These two noise types are caused by rapidly changing currents through GND and VCCI package pin inductances during switching activities (EQ 2 and EQ 3).

$$\text{Ground bounce noise voltage} = L(\text{GND}) \times di/dt$$

EQ 2

$$\text{VCCI dip noise voltage} = L(\text{VCCI}) \times di/dt$$

EQ 3

Any group of four or more input pins switching on the same clock edge is considered an SSO bus. The shielding should be done both on the board and inside the package unless otherwise described.

In-package shielding can be achieved in several ways; the required shielding will vary depending on whether pins next to the SSO bus are LVTTL/LVCMOS inputs, LVTTL/LVCMOS outputs, or GTL/SSTL/HSTL/LVDS/LVPECL inputs and outputs. Board traces in the vicinity of the SSO bus have to be adequately shielded from mutual coupling and inductive noise that can be generated by the SSO bus. Also, noise generated by the SSO bus needs to be reduced inside the package.

PCBs perform an important function in feeding stable supply voltages to the IC and, at the same time, maintaining signal integrity between devices.

Key issues that need to be considered are as follows:

- Power and ground plane design and decoupling network design
- Transmission line reflections and terminations

For extensive data per package on the SSO and PCB issues, refer to the "ProASIC3/E SSO and Pin Placement and Guidelines" chapter of the *ProASIC3 FPGA Fabric User's Guide*.

I/O Software Support

In Microsemi's Libero software, default settings have been defined for the various I/O standards supported. Changes can be made to the default settings via the use of attributes; however, not all I/O attributes are applicable for all I/O standards. Table 7-17 list the valid I/O attributes that can be manipulated by the user for each I/O standard.

Single-ended I/O standards in low power flash devices support up to five different drive strengths.

***Table 7-17* • IGLOO and ProASIC3 I/O Attributes vs. I/O Standard Applications**

I/O Standard	SLEW (output only)	OUT_DRIVE (output only)	SKEW (all macros with OE)	RES_PULL	OUT_LOAD (output only)	COMBINE_REGISTER
LVTTL/LVCMOS 3.3 V	✓	✓	✓	✓	✓	✓
LVCMOS 2.5 V	✓	✓	✓	✓	✓	✓
LVCMOS 2.5/5.0 V	✓	✓	✓	✓	✓	✓
LVCMOS 1.8 V	✓	✓	✓	✓	✓	✓
LVCMOS 1.5 V	✓	✓	✓	✓	✓	✓
PCI (3.3 V)			✓		✓	✓
PCI-X (3.3 V)	✓		✓		✓	✓
LVDS, B-LVDS, M-LVDS			✓			✓
LVPECL						✓

Note: Applies to all 30 k gate devices.

Table 7-18 lists the default values for the above selectable I/O attributes as well as those that are preset for that I/O standard. See Table 7-14 on page 189 to Table 7-16 on page 189 for SLEW and OUT_DRIVE settings.

***Table 7-18* • IGLOO and ProASIC3 I/O Default Attributes**

I/O Standards	SLEW (output only)	OUT_DRIVE (output only)	SKEW (tribuf and bibuf only)	RES_PULL	OUT_LOAD (output only)	COMBINE_REGISTER
LVTTL/LVCMOS 3.3 V	See Table 7-14 on page 189 to Table 7-16 on page 189.	See Table 7-14 on page 189 to Table 7-16 on page 189.	Off	None	35 pF	–
LVCMOS 2.5 V			Off	None	35 pF	–
LVCMOS 2.5/5.0 V			Off	None	35 pF	–
LVCMOS 1.8 V			Off	None	35 pF	–
LVCMOS 1.5 V			Off	None	35 pF	–
PCI (3.3 V)			Off	None	10 pF	–
PCI-X (3.3 V)			Off	None	10 pF	–
LVDS, B-LVDS, M-LVDS			Off	None	0 pF	–
LVPECL			Off	None	0 pF	–

User I/O Naming Convention

IGLOO and ProASIC3

Due to the comprehensive and flexible nature of IGLOO and ProASIC3 device user I/Os, a naming scheme is used to show the details of each I/O (Figure 7-19 on page 193 and Figure 7-20 on page 193). The name identifies to which I/O bank it belongs, as well as pairing and pin polarity for differential I/Os.

I/O Nomenclature = FF/Gmn/IOuxwBy

Gmn is only used for I/Os that also have CCC access—i.e., global pins.

FF = Indicates the I/O dedicated for the Flash*Freeze mode activation pin in IGLOO and ProASIC3L devices only

G = Global

m = Global pin location associated with each CCC on the device: A (northwest corner), B (northeast corner), C (east middle), D (southeast corner), E (southwest corner), and F (west middle)

n = Global input MUX and pin number of the associated Global location m—either A0, A1, A2, B0, B1, B2, C0, C1, or C2. Refer to the "Global Resources in Low Power Flash Devices" section on page 33 for information about the three input pins per clock source MUX at CCC location m.

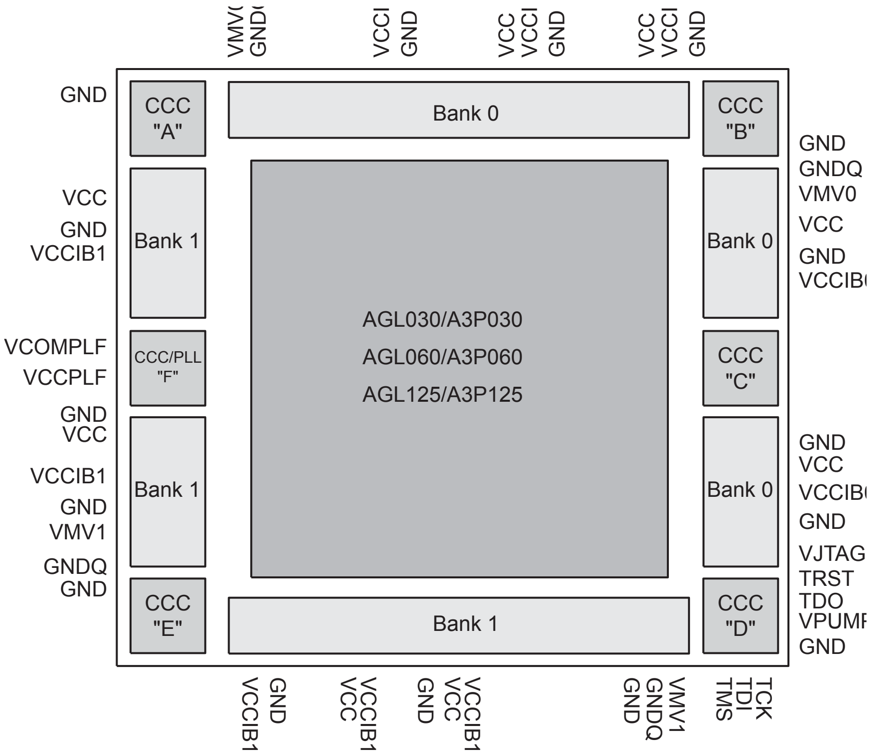
u = I/O pair number in the bank, starting at 00 from the northwest I/O bank and proceeding in a clockwise direction

x = P or U (Positive), N or V (Negative) for differential pairs, or R (Regular—single-ended) for the I/Os that support single-ended and voltage-referenced I/O standards only. U (Positive) or V (Negative)—for LVDS, DDR LVDS, B-LVDS, and M-LVDS only—restricts the I/O differential pair from being selected as an LVPECL pair.

w = D (Differential Pair), P (Pair), or S (Single-Ended). D (Differential Pair) if both members of the pair are bonded out to adjacent pins or are separated only by one GND or NC pin; P (Pair) if both members of the pair are bonded out but do not meet the adjacency requirement; or S (Single-Ended) if the I/O pair is not bonded out. For Differential Pairs (D), adjacency for ball grid packages means only vertical or horizontal. Diagonal adjacency does not meet the requirements for a true differential pair.

B = Bank

y = Bank number (0–3). The Bank number starts at 0 from the northwest I/O bank and proceeds in a clockwise direction.



Note: The 30 k gate devices do not support a PLL (V_{COMPLF} and V_{CCPLF} pins).

Figure 7-19 **• Naming Conventions of IGLOO and ProASIC3 Devices with Two I/O Banks – Top View**

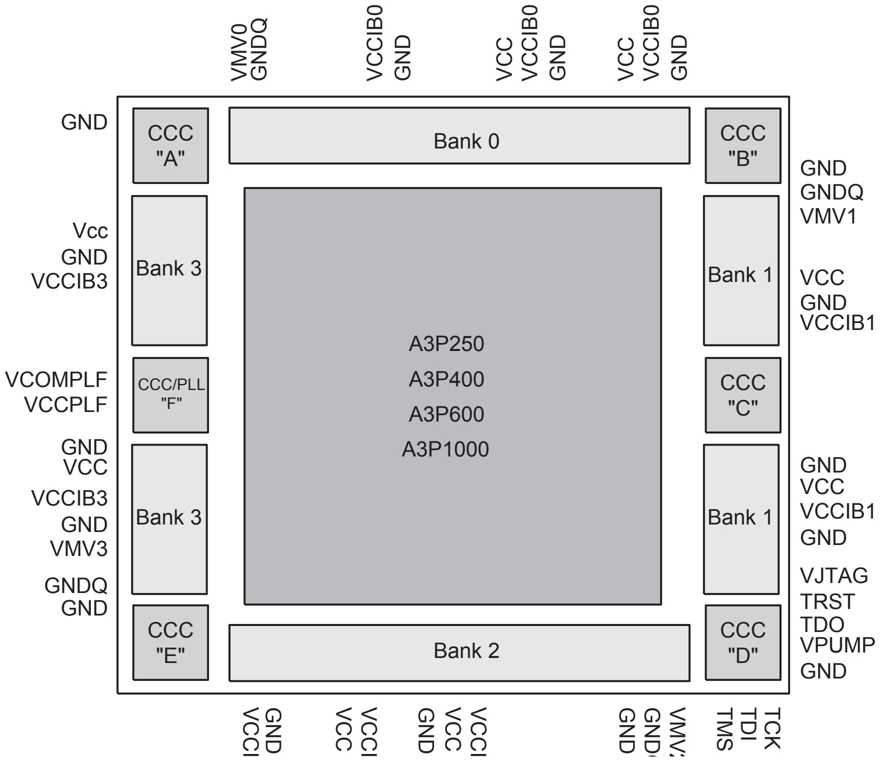


Figure 7-20 **• Naming Conventions of IGLOO and ProASIC3 Devices with Four I/O Banks – Top View**

Board-Level Considerations

Low power flash devices have robust I/O features that can help in reducing board-level components. The devices offer single-chip solutions, which makes the board layout simpler and more immune to signal integrity issues. Although, in many cases, these devices resolve board-level issues, special attention should always be given to overall signal integrity. This section covers important board-level considerations to facilitate optimum device performance.

Termination

Proper termination of all signals is essential for good signal quality. Nonterminated signals, especially clock signals, can cause malfunctioning of the device.

For general termination guidelines, refer to the *Board-Level Considerations* application note for Microsemi FPGAs. Also refer to the "Pin Descriptions" chapter of the appropriate datasheet for termination requirements for specific pins.

Low power flash I/Os are equipped with on-chip pull-up/-down resistors. The user can enable these resistors by instantiating them either in the top level of the design (refer to the *IGLOO, Fusion, and ProASIC3 Macro Library Guide* for the available I/O macros with pull-up/-down) or in the I/O Attribute Editor in Designer if generic input or output buffers are instantiated in the top level. Unused I/O pins are configured as inputs with pull-up resistors.

As mentioned earlier, low power flash devices have multiple programmable drive strengths, and the user can eliminate unwanted overshoot and undershoot by adjusting the drive strengths.

Power-Up Behavior

Low power flash devices are power-up/-down friendly; i.e., no particular sequencing is required for power-up and power-down. This eliminates extra board components for power-up sequencing, such as a power-up sequencer.

During power-up, all I/Os are tristated, irrespective of I/O macro type (input buffers, output buffers, I/O buffers with weak pull-ups or weak pull-downs, etc.). Once I/Os become activated, they are set to the user-selected I/O macros. Refer to the "Power-Up/-Down Behavior of Low Power Flash Devices" section on page 387 for details.

Drive Strength

Low power flash devices have up to seven programmable output drive strengths. The user can select the drive strength of a particular output in the I/O Attribute Editor or can instantiate a specialized I/O macro, such as OUTBUF_S_12 (slew = low, out_drive = 12 mA).

The maximum available drive strength is 24 mA per I/O. Though no I/O should be forced to source or sink more than 24 mA indefinitely, I/Os may handle a higher amount of current (refer to the device IBIS model for maximum source/sink current) during signal transition (AC current). Every device package has its own power dissipation limit; hence, power calculation must be performed accurately to determine how much current can be tolerated per I/O within that limit.

I/O Interfacing

Low power flash devices are 5 V–input– and 5 V–output–tolerant if certain I/O standards are selected (refer to the "5 V Input and Output Tolerance" section on page 180). Along with other low-voltage I/O macros, this 5 V tolerance makes these devices suitable for many types of board component interfacing.

Table 7-19 shows some high-level interfacing examples using low power flash devices.

***Table 7-19* • High-Level Interface Examples**

	Clock		**I/O**			
Interface	**Type**	**Frequency**	**Type**	**Signals In**	**Signals Out**	**Data I/O**
GM	Src Sync	125 MHz	LVTTL	8	8	125 Mbps
TBI	Src Sync	125 MHz	LVTTL	10	10	125 Mbps
XSBI	Src Sync	644 MHz	LVDS	16	16	644 Mbps
XGMI	Src Sync DDR	156 MHz	HSTL1	32	32	312 Mbps
FlexBus 3	Sys Sync	104 MHz	LVTTL	≤ 32	≤ 32	≤ 104
Pos-PHY3/SPI-3	Sys Sync	104	LVTTL	8, 16, 32	8, 16, 32	≤ 104 Mbps
FlexBus 4/SPI-4.1	Src Sync	200 MHz	HSTL1	16,64	16,64	200 Mbps
Pos-PHY4/SPI-4.2	Src Sync DDR	≥ 311 MHz	LVDS	16	16	≥ 622 Mbps
SFI-4.1	Src Sync	622 MHz	LVDS	16	16	622 Mbps
CSIX L1	Sys Sync	≤ 250 MHz	HSTL1	32,64,96,128	32,64,96,128	≤ 250 Mbps
Hyper Transport	Sys Sync DDR	≤ 800 MHz	LVDS	2,4,8,16	2,4,8,16	≤ 1.6 Gbps
Rapid I/O Parallel	Sys Sync DDR	250 MHz – 1 GHz	LVDS	8,16	8,16	≤ 2 Gbps
Star Fabric	CDR		LVDS	4	4	622 Mbps

Note: Sys Sync = System Synchronous Clocking, Src Sync = Source Synchronous Clocking, and CDR = Clock and Data Recovery.

Conclusion

IGLOO and ProASIC3 support for multiple I/O standards minimizes board-level components and makes possible a wide variety of applications. The Microsemi Designer software, integrated with Libero SoC, presents a clear visual display of I/O assignments, allowing users to verify I/O and board-level design requirements before programming the device. The IGLOO and ProASIC3 device I/O features and functionalities ensure board designers can produce low-cost and low power FPGA applications fulfilling the complexities of contemporary design needs.

Related Documents

Application Notes

Board-Level Considerations

http://www.microsemi.com/soc/documents/ALL_AC276_AN.pdf

User's Guides

Libero SoC User's Guide

http://www.microsemi.com.soc/documents/libero_ug.pdf

IGLOO, Fusion, and ProASIC3 Macro Library Guide

http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

SmartGen Core Reference Guide

http://www.microsemi.com/soc/documents/genguide_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the document.

Date	Change	Page
August 2012	Figure 7-1 • DDR Configured I/O Block Logical Representation and Figure 7-2 • DDR Configured I/O Block Logical Representation were revised to indicate that resets on registers 1, 3, 4, and 5 are active high rather than active low. The title of the figures was revised from "I/O Block Logical Representation" (SAR 38215).	161, 167
	AGL015 and A3P015 were added to Table 7-2 • Supported I/O Standards. 1.2 V was added under single-ended I/O standards. LVCMOS 1.2 was added to Table 7-3 • VCCI Voltages and Compatible IGLOO and ProASIC3 Standards (SAR 38096).	163
	Figure 7-4 • Simplified I/O Buffer Circuitry and Table 7-7 • Programmable I/O Features (user control via I/O Attribute Editor) were modified to indicate that programmable input delay control is applicable only to ProASIC3EL and RT ProASIC3 devices (SAR 39666).	169, 174
	The following sentence is incorrect and was removed from the "LVCMOS (Low-Voltage CMOS)" section (SAR 40191): LVCMOS 2.5 V for the 30 k gate devices has a clamp diode to VCCI, but for all other devices there is no clamp diode.	170
	The hyperlink for the *Board-Level Considerations* application note was corrected (SAR 36663).	194, 196
June 2011	Figure 7-1 • DDR Configured I/O Block Logical Representation and Figure 7-2 • DDR Configured I/O Block Logical Representation were revised so that the I/O_CLR and I/O_OCLK nets are no longer joined in front of Input Register 3 but instead on the branch of the CLR/PRE signal (SAR 26052).	161, 167
	Table 7-1 • Flash-Based FPGAs was revised to remove RT ProASIC3 and add Military ProASIC3/EL in its place (SAR 31824, 31825).	162
	The "Advanced I/Os—IGLOO, ProASIC3L, and ProASIC3" section was revised. Formerly it stated, "3.3 V PCI and 3.3 V PCI-X are 5 V–tolerant." This sentence now reads, "3.3 V PCI and 3.3 V PCI-X can be configured to be 5 V–tolerant" (SAR 20983).	163

Date	Change	Page
June 2011 (continued)	The following sentence was removed from the "LVCMOS (Low-Voltage CMOS)" section (SAR 22634): "All these versions use a 3.3 V–tolerant CMOS input buffer and a push-pull output buffer."	170
	Hot-insertion was changed to "No" for other IGLOO and all ProASIC3 devices in Table 7-12 • I/O Hot-Swap and 5 V Input Tolerance Capabilities in IGLOO and ProASIC3 Devices (SAR 24526).	179
	The "Electrostatic Discharge Protection" section was revised to remove references to tolerances (refer to the *Reliability Report* for tolerances). The Machine Model (MM) is not supported and was deleted from this section (SAR 24385).	178
	The "I/O Interfacing" section was revised to state that low power flash devices are 5 V–input– and 5 V–output–tolerant if certain I/O standards are selected, removing "without adding any extra circuitry," which was incorrect (SAR 21404).	194
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	The terminology in the "Low Power Flash Device I/O Support" section was revised.	162
v1.3 (October 2008)	The "Low Power Flash Device I/O Support" section was revised to include new families and make the information more concise.	162
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 7-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	162
v1.1 (March 2008)	Originally, this document contained information on all IGLOO and ProASIC3 families. With the addition of new families and to highlight the differences between the features, the document has been separated into 3 documents: This document contains information specific to IGLOO, ProASIC3, and ProASIC3L. "I/O Structures in IGLOOe and ProASIC3E Devices" in the *ProASIC3E FPGA Fabric User's Guide* contains information specific to IGLOOe, ProASIC3E, and ProASIC3EL I/O features. "I/O Structures in IGLOO PLUS Devices" in the *IGLOO PLUS FPGA Fabric User's Guide* contains information specific to IGLOO PLUS I/O features.	N/A

8 – I/O Software Control in Low Power Flash Devices

Fusion, IGLOO, and ProASIC3 I/Os provide more design flexibility, allowing the user to control specific features by enabling certain I/O standards. Some features are selectable only for certain I/O standards, whereas others are available for all I/O standards. For example, slew control is not supported by differential I/O standards. Conversely, I/O register combining is supported by all I/O standards. For detailed information about which I/O standards and features are available on each device and each I/O type, refer to the I/O Structures section of the handbook for the device you are using.

Figure 8-1 shows the various points in the software design flow where a user can provide input or control of the I/O selection and parameters. A detailed description is provided throughout this document.

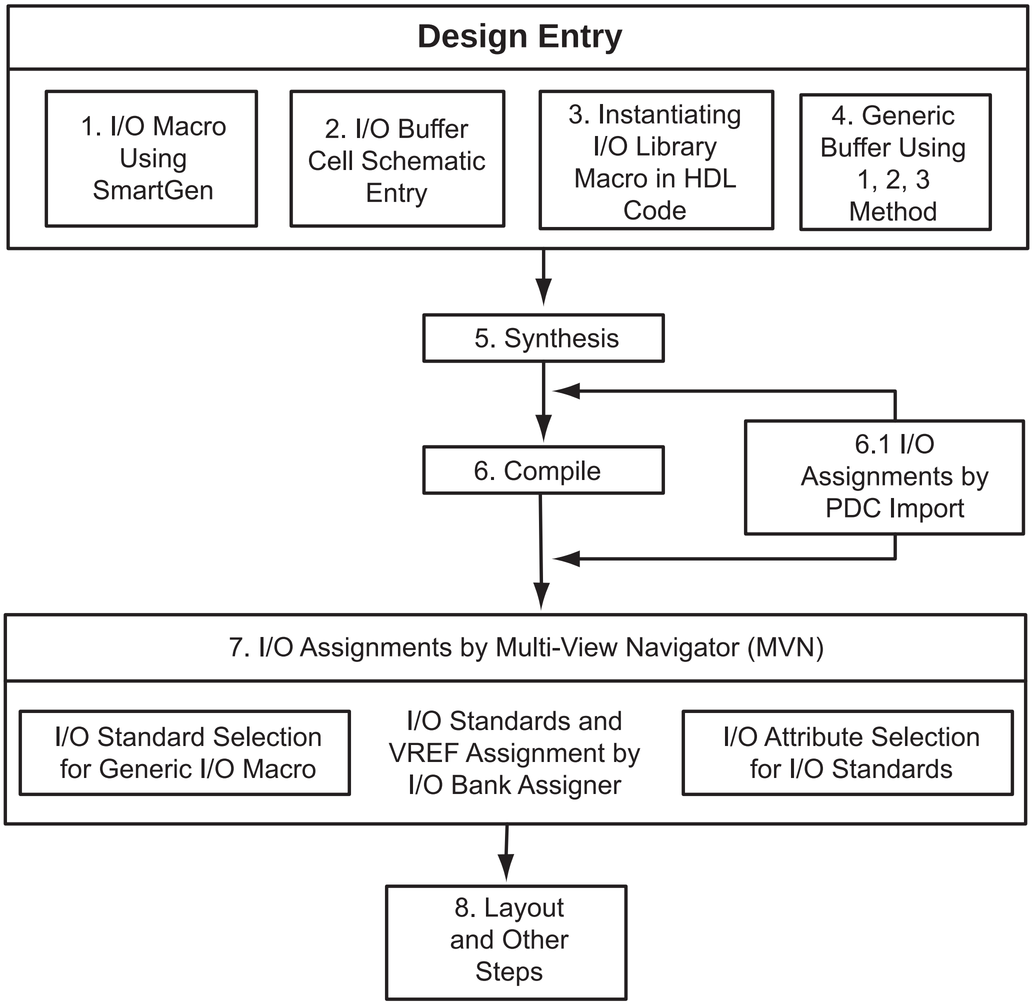

Design Entry
1. I/O Macro Using SmartGen
2. I/O Buffer Cell Schematic Entry
3. Instantiating I/O Library Macro in HDL Code
4. Generic Buffer Using 1, 2, 3 Method
5. Synthesis
6. Compile
6.1 I/O Assignments by PDC Import
7. I/O Assignments by Multi-View Navigator (MVN)
I/O Standard Selection for Generic I/O Macro
I/O Standards and VREF Assignment by I/O Bank Assigner
I/O Attribute Selection for I/O Standards
8. Layout and Other Steps

Figure 8-1 **• User I/O Assignment Flow Chart**

Flash FPGAs I/O Support

The flash FPGAs listed in Table 8-1 support I/Os and the functions described in this document.

***Table 8-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 8-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 8-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Software-Controlled I/O Attributes

Users may modify these programmable I/O attributes using the I/O Attribute Editor. Modifying an I/O attribute may result in a change of state in Designer. Table 8-2 details which steps have to be re-run as a function of modified I/O attribute.

Table 8-2 **• Designer State (resulting from I/O attribute modification)**

I/O Attribute	Designer States[1]				
	Compile	**Layout**	**Fuse**	**Timing**	**Power**
Slew Control[2]	No	No	Yes	Yes	Yes
Output Drive (mA)	No	No	Yes	Yes	Yes
Skew Control	No	No	Yes	Yes	Yes
Resistor Pull	No	No	Yes	Yes	Yes
Input Delay	No	No	Yes	Yes	Yes
Schmitt Trigger	No	No	Yes	Yes	Yes
OUT_LOAD	No	No	No	Yes	Yes
COMBINE_REGISTER	Yes	Yes	N/A	N/A	N/A

Notes:

1. *No = Remains the same, Yes = Re-run the step, N/A = Not applicable*
2. *Skew control does not apply to IGLOO nano, IGLOO PLUS, and ProASIC3 nano devices.*
3. *Programmable input delay is applicable only for ProASIC3E, ProASIC3EL, RT ProASIC3, and IGLOOe devices.*

Implementing I/Os in Microsemi Software

Microsemi Libero SoC software is integrated with design entry tools such as the SmartGen macro builder, the ViewDraw schematic entry tool, and an HDL editor. It is also integrated with the synthesis and Designer tools. In this section, all necessary steps to implement the I/Os are discussed.

Design Entry

There are three ways to implement I/Os in a design:

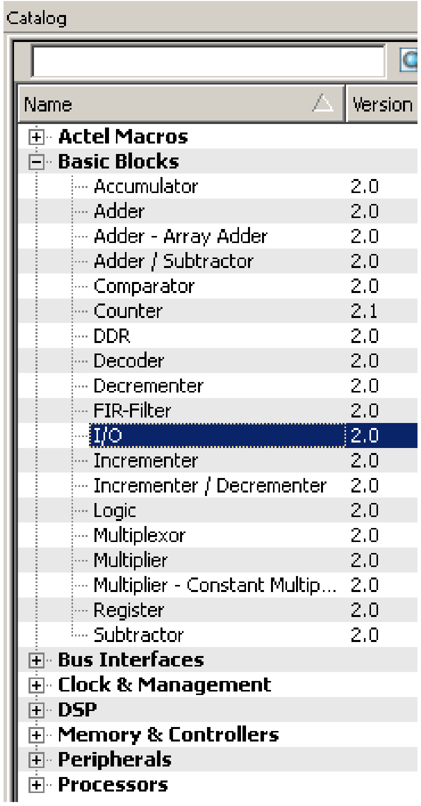
1. Use the SmartGen macro builder to configure I/Os by generating specific I/O library macros and then instantiating them in top-level code. This is especially useful when creating I/O bus structures.
2. Use an I/O buffer cell in a schematic design.
3. Manually instantiate specific I/O macros in the top-level code.

If technology-specific macros, such as INBUF_LVCMOS33 and OUTBUF_PCI, are used in the HDL code or schematic, the user will not be able to change the I/O standard later on in Designer. If generic I/O macros are used, such as INBUF, OUTBUF, TRIBUF, CLKBUF, and BIBUF, the user can change the I/O standard using the Designer I/O Attribute Editor tool.

Using SmartGen for I/O Configuration

The SmartGen tool in Libero SoC provides a GUI-based method of configuring the I/O attributes. The user can select certain I/O attributes while configuring the I/O macro in SmartGen. The steps to configure an I/O macro with specific I/O attributes are as follows:

1. Open Libero SoC.
2. On the left-hand side of the Catalog View, select **I/O**, as shown in Figure 8-2.

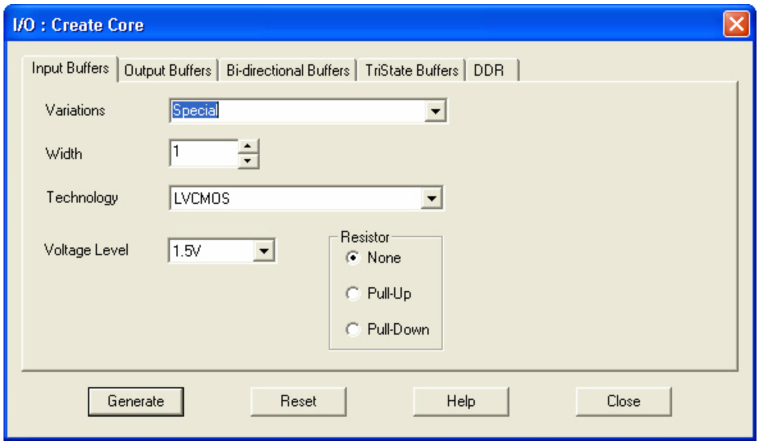


Catalog

Name	Version
⊞ **Actel Macros**	
⊟ **Basic Blocks**	
Accumulator	2.0
Adder	2.0
Adder - Array Adder	2.0
Adder / Subtractor	2.0
Comparator	2.0
Counter	2.1
DDR	2.0
Decoder	2.0
Decrementer	2.0
FIR-Filter	2.0
I/O	2.0
Incrementer	2.0
Incrementer / Decrementer	2.0
Logic	2.0
Multiplexor	2.0
Multiplier	2.0
Multiplier - Constant Multip...	2.0
Register	2.0
Subtractor	2.0
⊞ **Bus Interfaces**	
⊞ **Clock & Management**	
⊞ **DSP**	
⊞ **Memory & Controllers**	
⊞ **Peripherals**	
⊞ **Processors**	

Figure 8-2 • **SmartGen Catalog**

3. Double-click I/O to open the Create Core window, which is shown in Figure 8-3).



Variations: Special

Width: 1

Technology: LVCMOS

Voltage Level: 1.5V

Resistor: ● None ○ Pull-Up ○ Pull-Down

Figure 8-3 **• I/O Create Core Window**

As seen in Figure 8-3, there are five tabs to configure the I/O macro: Input Buffers, Output Buffers, Bidirectional Buffers, Tristate Buffers, and DDR.

Input Buffers

There are two variations: Regular and Special.

If the **Regular** variation is selected, only the Width (1 to 128) needs to be entered. The default value for Width is 1.

The **Special** variation has Width, Technology, Voltage Level, and Resistor Pull-Up/-Down options (see Figure 8-3). All the I/O standards and supply voltages (V_{CCI}) supported for the device family are available for selection.

Output Buffers

There are two variations: Regular and Special.

If the **Regular** variation is selected, only the Width (1 to 128) needs to be entered. The default value for Width is 1.

The **Special** variation has Width, Technology, Output Drive, and Slew Rate options.

Bidirectional Buffers

There are two variations: Regular and Special.

The **Regular** variation has Enable Polarity (Active High, Active Low) in addition to the Width option.

The **Special** variation has Width, Technology, Output Drive, Slew Rate, and Resistor Pull-Up/-Down options.

Tristate Buffers

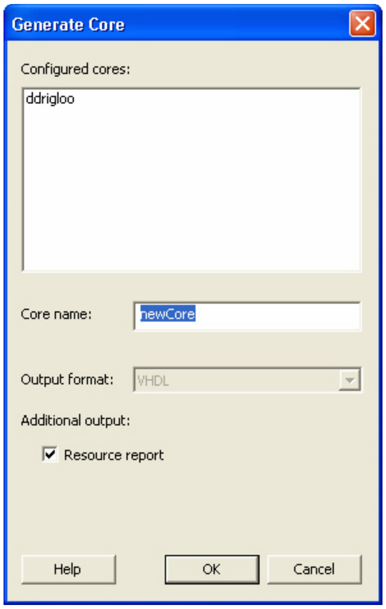
Same as Bidirectional Buffers.

DDR

There are eight variations: DDR with Regular Input Buffers, Special Input Buffers, Regular Output Buffers, Special Output Buffers, Regular Tristate Buffers, Special Tristate Buffers, Regular Bidirectional Buffers, and Special Bidirectional Buffers.

These variations resemble the options of the previous I/O macro. For example, the Special Input Buffers variation has Width, Technology, Voltage Level, and Resistor Pull-Up/-Down options. DDR is not available on IGLOO PLUS devices.

4. Once the desired configuration is selected, click the **Generate** button. The Generate Core window opens (Figure 8-4).
5. Enter a name for the macro. Click **OK**. The core will be generated and saved to the appropriate location within the project files (Figure 8-5 on page 205).



ddrigloo

newCore

☑ Resource report

Figure 8-4 • **Generate Core Window**

6. Instantiate the I/O macro in the top-level code.

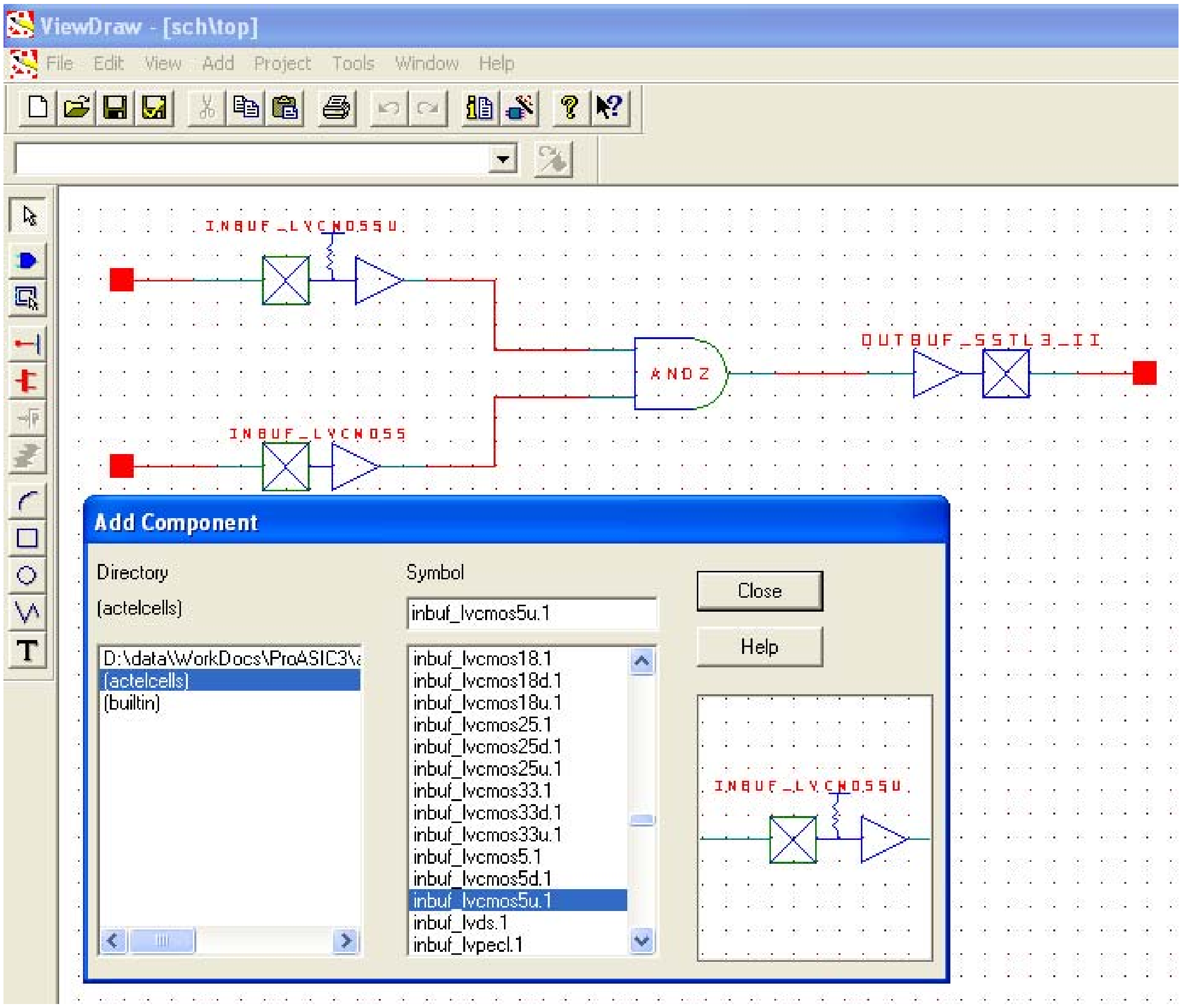
   The user must instantiate the DDR_REG or DDR_OUT macro in the design. Use SmartGen to generate both these macros and then instantiate them in your top level. To combine the DDR macros with the I/O, the following rules must be met:

Rules for the DDR I/O Function

- The fanout between an I/O pin (D or Y) and a DDR (DDR_REG or DDR_OUT) macro must be equal to one for the combining to happen on that pin.
- If a DDR_REG macro and a DDR_OUT macro are combined on the same bidirectional I/O, they must share the same clear signal.
- Registers will not be combined in an I/O in the presence of DDR combining on the same I/O.

Using the I/O Buffer Schematic Cell

Libero SoC software includes the ViewDraw schematic entry tool. Using ViewDraw, the user can insert any supported I/O buffer cell in the top-level schematic. Figure 8-5 shows a top-level schematic with different I/O buffer cells. When synthesized, the netlist will contain the same I/O macro.



Add Component

Directory
(actelcells)

D:\data\WorkDocs\ProASIC3\a
(actelcells)
(builtin)

Symbol
inbuf_lvcmos5u.1

inbuf_lvcmos18.1
inbuf_lvcmos18d.1
inbuf_lvcmos18u.1
inbuf_lvcmos25.1
inbuf_lvcmos25d.1
inbuf_lvcmos25u.1
inbuf_lvcmos33.1
inbuf_lvcmos33d.1
inbuf_lvcmos33u.1
inbuf_lvcmos5.1
inbuf_lvcmos5d.1
inbuf_lvcmos5u.1
inbuf_lvds.1
inbuf_lvpecl.1

***Figure 8-5* • I/O Buffer Schematic Cell Usage**

Instantiating in HDL code

All the supported I/O macros can be instantiated in the top-level HDL code (refer to the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide* for a detailed list of all I/O macros). The following is an example:

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3e;

entity TOP is
  port(IN2, IN1 : in std_logic; OUT1 : out std_logic);
end TOP;

architecture DEF_ARCH of TOP is

  component INBUF_LVCMOS5U
    port(PAD : in std_logic := 'U'; Y : out std_logic);
  end component;

  component INBUF_LVCMOS5
    port(PAD : in std_logic := 'U'; Y : out std_logic);
  end component;

  component OUTBUF_SSTL3_II
    port(D : in std_logic := 'U'; PAD : out std_logic);
  end component;

  Other component …..

signal x, y, z…….other signals : std_logic;

begin

  I1 : INBUF_LVCMOS5U
    port map(PAD => IN1, Y =>x);
  I2 : INBUF_LVCMOS5
    port map(PAD => IN2, Y => y);
  I3 : OUTBUF_SSTL3_II
    port map(D => z, PAD => OUT1);

  other port mapping…

end DEF_ARCH;
```

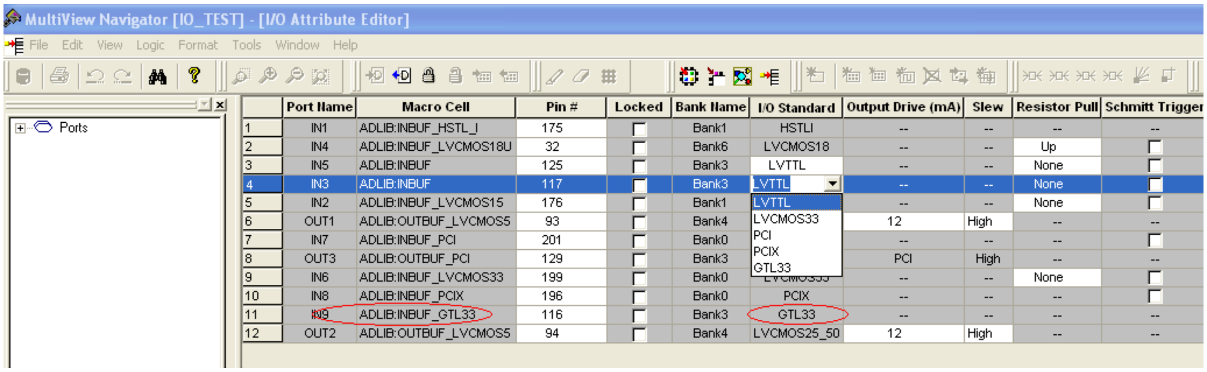
Synthesizing the Design

Libero SoC integrates with the Synplify® synthesis tool. Other synthesis tools can also be used with Libero SoC. Refer to the *Libero SoC User's Guide* or Libero online help for details on how to set up the Libero tool profile with synthesis tools from other vendors.

During synthesis, the following rules apply:

- Generic macros:
  - Users can instantiate generic INBUF, OUTBUF, TRIBUF, and BIBUF macros.
  - Synthesis will automatically infer generic I/O macros.
  - The default I/O technology for these macros is LVTTL.
  - Users will need to use the I/O Attribute Editor in Designer to change the default I/O standard if needed (see Figure 8-6 on page 207).
- Technology-specific I/O macros:
  - Technology-specific I/O macros, such as INBUF_LVCMO25 and OUTBUF_GTL25, can be instantiated in the design. Synthesis will infer these I/O macros in the netlist.

- – The I/O standard of technology-specific I/O macros cannot be changed in the I/O Attribute Editor (see Figure 8-6).
- – The user MUST instantiate differential I/O macros (LVDS/LVPECL) in the design. This is the only way to use these standards in the design (IGLOO nano and ProASIC3 nano devices do not support differential inputs).
- – To implement the DDR I/O function, the user must instantiate a DDR_REG or DDR_OUT macro. This is the only way to use a DDR macro in the design.



	Port Name	Macro Cell	Pin #	Locked	Bank Name	I/O Standard	Output Drive (mA)	Slew	Resistor Pull	Schmitt Trigger
1	IN1	ADLIB:INBUF_HSTL_I	175	☐	Bank1	HSTLI	--	--	--	--
2	IN4	ADLIB:INBUF_LVCMOS18U	32	☐	Bank6	LVCMOS18	--	--	Up	☐
3	IN5	ADLIB:INBUF	125	☐	Bank3	LVTTL	--	--	None	☐
4	IN3	ADLIB:INBUF	117	☐	Bank3	LVTTL	--	--	None	☐
5	IN2	ADLIB:INBUF_LVCMOS15	176	☐	Bank1		--	--	None	☐
6	OUT1	ADLIB:OUTBUF_LVCMOS5	93	☐	Bank4		12	High	--	--
7	IN7	ADLIB:INBUF_PCI	201	☐	Bank0		--	--	--	☐
8	OUT3	ADLIB:OUTBUF_PCI	129	☐	Bank3		PCI	High	--	--
9	IN6	ADLIB:INBUF_LVCMOS33	199	☐	Bank0	LVCMOS33	--	--	None	☐
10	IN8	ADLIB:INBUF_PCIX	196	☐	Bank0	PCIX	--	--	--	☐
11	IN9	ADLIB:INBUF_GTL33	116	☐	Bank3	GTL33	--	--	--	--
12	OUT2	ADLIB:OUTBUF_LVCMOS5	94	☐	Bank4	LVCMOS25_50	12	High	--	--

***Figure 8-6* • Assigning a Different I/O Standard to the Generic I/O Macro**

Performing Place-and-Route on the Design

The netlist created by the synthesis tool should now be imported into Designer and compiled. During Compile, the user can specify the I/O placement and attributes by importing the PDC file. The user can also specify the I/O placement and attributes using ChipPlanner and the I/O Attribute Editor under MVN.

Defining I/O Assignments in the PDC File

A PDC file is a Tcl script file specifying physical constraints. This file can be imported to and exported from Designer.

Table 8-3 shows I/O assignment constraints supported in the PDC file.

***Table 8-3* • PDC I/O Constraints**

Command	Action	Example	Comment
I/O Banks Setting Constraints			

set_iobank	Sets the I/O supply voltage, V_{CCI}, and the input reference voltage, V_{REF}, for the specified I/O bank.	`set_iobank bankname [-vcci vcci_voltage] [-vref vref_voltage]`
		`set_iobank Bank7 -vcci 1.50 -vref 0.75`
			Must use in case of mixed I/O voltage (V_{CCI}) design

set_vref	Assigns a V_{REF} pin to a bank.	`set_vref -bank [bankname] [pinnum]`
		`set_vref -bank Bank0 685 704 723 742 761`
			Must use if voltage-referenced I/Os are used

set_vref_defaults	Sets the default V_{REF} pins for the specified bank. This command is ignored if the bank does not need a V_{REF} pin.	`set_vref_defaults bankname`
		`set_vref_defaults bank2`	

Note: Refer to the Libero SoC User’s Guide for detailed rules on PDC naming and syntax conventions.

Table 8-3 • **PDC I/O Constraints (continued)**

Command	Action	Example	Comment
I/O Attribute Constraint			

set_io	Sets the attributes of an I/O	`set_io portname`
		`[-pinname value]`
		`[-fixed value]`
		`[-iostd value]`
		`[-out_drive value]`
		`[-slew value]`
		`[-res_pull value]`
		`[-schmitt_trigger value]`
		`[-in_delay value]`
		`[-skew value]`
		`[-out_load value]`
		`[-register value]`
		
		`set_io IN2 -pinname 28`
		`-fixed yes -iostd LVCMOS15`
		`-out_drive 12 -slew high`
		`-RES_PULL None`
		`-SCHMITT_TRIGGER Off`
		`-IN_DELAY Off –skew off`
		`-REGISTER No`
			If the I/O macro is generic (e.g., INBUF) or technology-specific (INBUF_LVCMOS25), then all I/O attributes can be assigned using this constraint.
			If the netlist has an I/O macro that specifies one of its attributes, that attribute cannot be changed using this constraint, though other attributes can be changed.
			Example: OUTBUF_S_24 (low slew, output drive 24 mA)
			Slew and output drive cannot be changed.

I/O Region Placement Constraints			

define_region	Defines either a rectangular region or a rectilinear region	`define_region`
		`-name [region_name]`
		`-type [region_type] x1 y1 x2 y2`
		
		`define_region -name test`
		`-type inclusive 0 15 2 29`
			If any number of I/Os must be assigned to a particular I/O region, such a region can be created with this constraint.

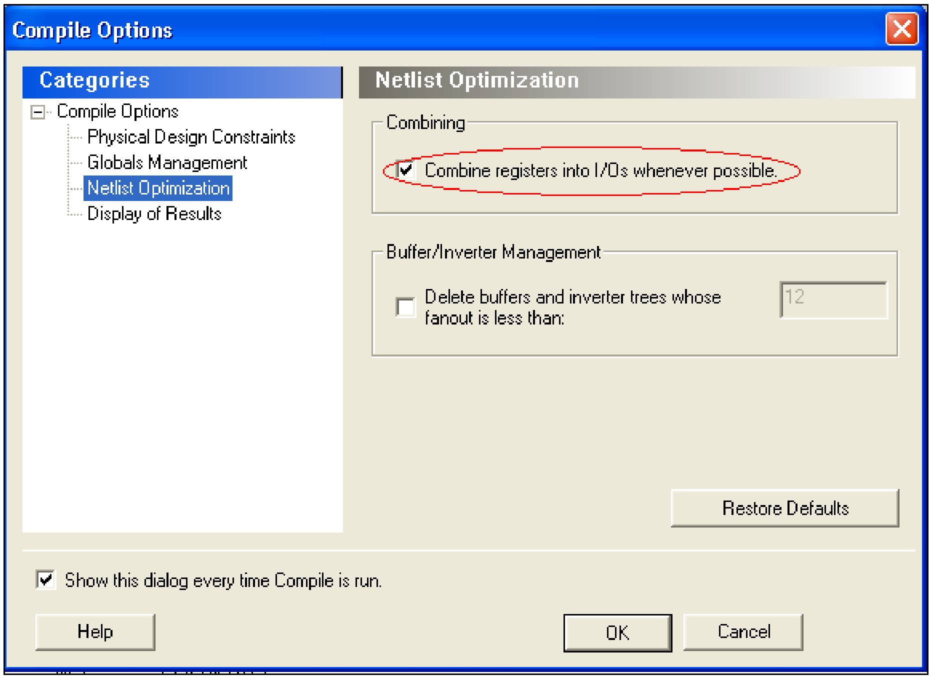
assign_region	Assigns a set of macros to a specified region	`assign_region [region name]`
		`[macro_name...]`
		
		`assign_region test U12`
			This constraint assigns I/O macros to the I/O regions. When assigning an I/O macro, PDC naming conventions must be followed if the macro name contains special characters; e.g., if the macro name is \\$1I19\\, the correct use of escape characters is \\\\\$1I19\\\\.

Note: Refer to the Libero SoC User's Guide *for detailed rules on PDC naming and syntax conventions.*

Compiling the Design

During Compile, a PDC I/O constraint file can be imported along with the netlist file. If only the netlist file is compiled, certain I/O assignments need to be completed before proceeding to Layout. All constraints that can be entered in PDC can also be entered using ChipPlanner, I/O Attribute Editor, and PinEditor.

There are certain rules that must be followed in implementing I/O register combining and the I/O DDR macro (refer to the I/O Registers section of the handbook for the device that you are using and the "DDR" section on page 204 for details). Provided these rules are met, the user can enable or disable I/O register combining by using the PDC command `set_io portname -register yes|no` in the I/O Attribute Editor or selecting a check box in the Compile Options dialog box (see Figure 8-7). The Compile Options dialog box appears when the design is compiled for the first time. It can also be accessed by choosing **Options** > **Compile** during successive runs. I/O register combining is off by default. The PDC command overrides the setting in the Compile Options dialog box.



☑ Combine registers into I/Os whenever possible.

□ Delete buffers and inverter trees whose fanout is less than: 12

☑ Show this dialog every time Compile is run.

Figure 8-7 • **Setting Register Combining During Compile**

Understanding the Compile Report

The I/O bank report is generated during Compile and displayed in the log window. This report lists the I/O assignments necessary before Layout can proceed.

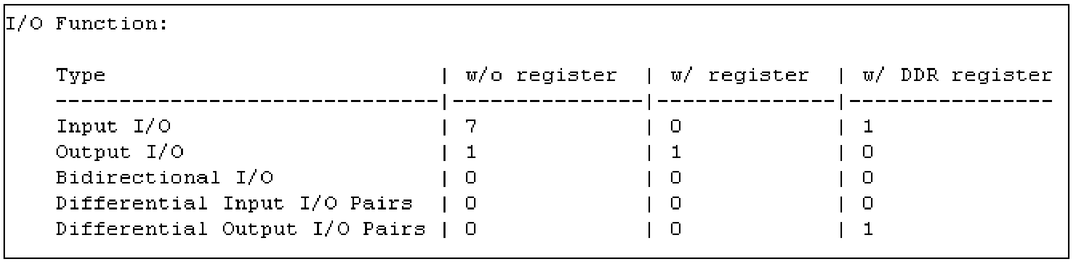
When Designer is started, the I/O Bank Assigner tool is run automatically if the Layout command is executed. The I/O Bank Assigner takes care of the necessary I/O assignments. However, these assignments can also be made manually with MVN or by importing the PDC file. Refer to the "Assigning Technologies and VREF to I/O Banks" section on page 212 for further description.

The I/O bank report can also be extracted from Designer by choosing **Tools** > **Report** and setting the Report Type to **IOBank**.

This report has the following tables: I/O Function, I/O Technology, I/O Bank Resource Usage, and I/O Voltage Usage. This report is useful if the user wants to do I/O assignments manually.

I/O Function

Figure 8-8 shows an example of the I/O Function table included in the I/O bank report:



```
I/O Function:

    Type                           | w/o register  | w/ register   | w/ DDR register
    -------------------------------|---------------|---------------|----------------
    Input I/O                      | 7             | 0             | 1
    Output I/O                     | 1             | 1             | 0
    Bidirectional I/O              | 0             | 0             | 0
    Differential Input I/O Pairs   | 0             | 0             | 0
    Differential Output I/O Pairs  | 0             | 0             | 1
```

***Figure 8-8* • I/O Function Table**

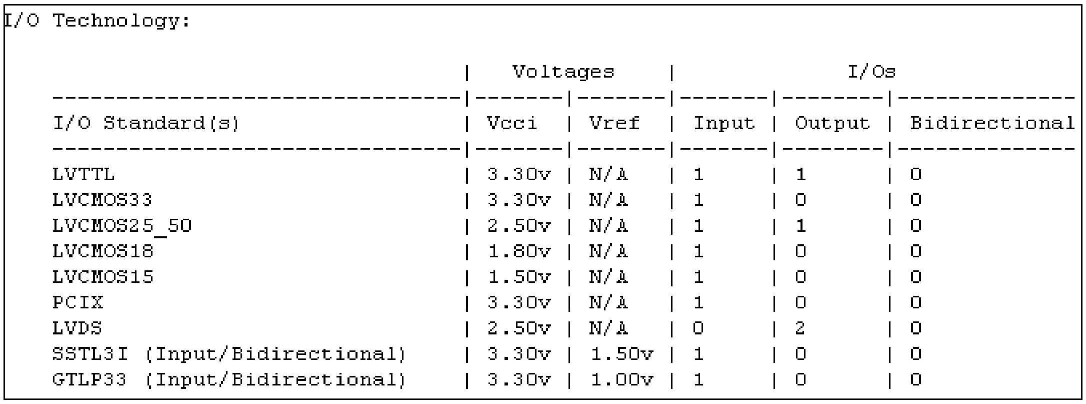
This table lists the number of input I/Os, output I/Os, bidirectional I/Os, and differential input and output I/O pairs that use I/O and DDR registers.

Note: IGLOO nano and ProASIC3 nano devices do not support differential inputs.

Certain rules must be met to implement registered and DDR I/O functions (refer to the I/O Structures section of the handbook for the device you are using and the "DDR" section on page 204).

I/O Technology

The I/O Technology table (shown in Figure 8-9) gives the values of VCCI and VREF (reference voltage) for all the I/O standards used in the design. The user should assign these voltages appropriately.

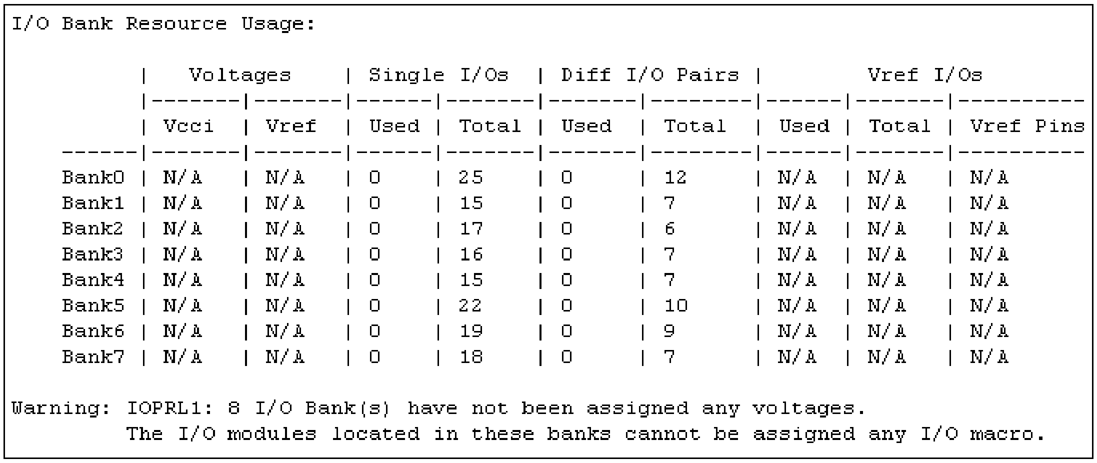


```
I/O Technology:

                                    |    Voltages     |             I/Os
    --------------------------------|-------|-------|--------|---------|--------------
    I/O Standard(s)                 | Vcci  | Vref  | Input  | Output  | Bidirectional
    --------------------------------|-------|-------|--------|---------|--------------
    LVTTL                           | 3.30v | N/A   | 1      | 1       | 0
    LVCMOS33                        | 3.30v | N/A   | 1      | 0       | 0
    LVCMOS25_50                     | 2.50v | N/A   | 1      | 1       | 0
    LVCMOS18                        | 1.80v | N/A   | 1      | 0       | 0
    LVCMOS15                        | 1.50v | N/A   | 1      | 0       | 0
    PCIX                            | 3.30v | N/A   | 1      | 0       | 0
    LVDS                            | 2.50v | N/A   | 0      | 2       | 0
    SSTL3I (Input/Bidirectional)    | 3.30v | 1.50v | 1      | 0       | 0
    GTLP33 (Input/Bidirectional)    | 3.30v | 1.00v | 1      | 0       | 0
```

***Figure 8-9* • I/O Technology Table**

I/O Bank Resource Usage

This is an important portion of the report. The user must meet the requirements stated in this table. Figure 8-10 shows the I/O Bank Resource Usage table included in the I/O bank report:



```
I/O Bank Resource Usage:

          |   Voltages     | Single I/Os   | Diff I/O Pairs |         Vref I/Os
          |-------|-------|------|-------|-------|--------|------|-------|----------
          | Vcci  | Vref  | Used | Total | Used  | Total  | Used | Total | Vref Pins
    ------|-------|-------|------|-------|-------|--------|------|-------|----------
    Bank0 | N/A   | N/A   | 0    | 25    | 0     | 12     | N/A  | N/A   | N/A
    Bank1 | N/A   | N/A   | 0    | 15    | 0     | 7      | N/A  | N/A   | N/A
    Bank2 | N/A   | N/A   | 0    | 17    | 0     | 6      | N/A  | N/A   | N/A
    Bank3 | N/A   | N/A   | 0    | 16    | 0     | 7      | N/A  | N/A   | N/A
    Bank4 | N/A   | N/A   | 0    | 15    | 0     | 7      | N/A  | N/A   | N/A
    Bank5 | N/A   | N/A   | 0    | 22    | 0     | 10     | N/A  | N/A   | N/A
    Bank6 | N/A   | N/A   | 0    | 19    | 0     | 9      | N/A  | N/A   | N/A
    Bank7 | N/A   | N/A   | 0    | 18    | 0     | 7      | N/A  | N/A   | N/A

Warning: IOPRL1: 8 I/O Bank(s) have not been assigned any voltages.
         The I/O modules located in these banks cannot be assigned any I/O macro.
```

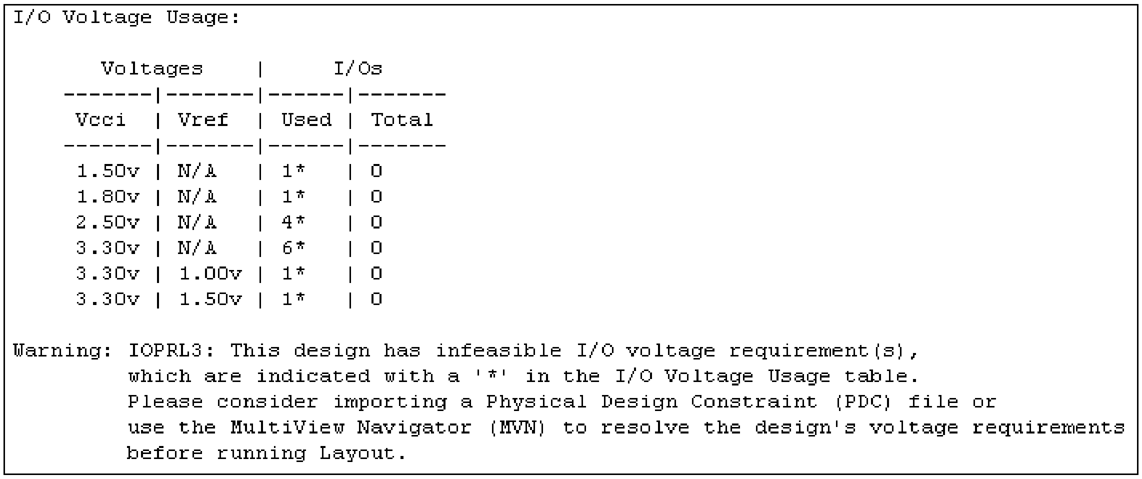
Figure 8-10 **• I/O Bank Resource Usage Table**

The example in Figure 8-10 shows that none of the I/O macros is assigned to the bank because more than one VCCI is detected.

I/O Voltage Usage

The I/O Voltage Usage table provides the number of VREF (E devices only) and V_{CCI} assignments required in the design. If the user decides to make I/O assignments manually (PDC or MVN), the issues listed in this table must be resolved before proceeding to Layout. As stated earlier, VREF assignments must be made if there are any voltage-referenced I/Os.

Figure 8-11 shows the I/O Voltage Usage table included in the I/O bank report.



```
I/O Voltage Usage:

       Voltages    |     I/Os
    -------|-------|------|-------
     Vcci  | Vref  | Used | Total
    -------|-------|------|-------
     1.50v | N/A   | 1*   | 0
     1.80v | N/A   | 1*   | 0
     2.50v | N/A   | 4*   | 0
     3.30v | N/A   | 6*   | 0
     3.30v | 1.00v | 1*   | 0
     3.30v | 1.50v | 1*   | 0

Warning: IOPRL3: This design has infeasible I/O voltage requirement(s),
         which are indicated with a '*' in the I/O Voltage Usage table.
         Please consider importing a Physical Design Constraint (PDC) file or
         use the MultiView Navigator (MVN) to resolve the design's voltage requirements
         before running Layout.
```

Figure 8-11 **• I/O Voltage Usage Table**

The table in Figure 8-11 indicates that there are two voltage-referenced I/Os used in the design. Even though both of the voltage-referenced I/O technologies have the same VCCI voltage, their VREF voltages are different. As a result, two I/O banks are needed to assign the VCCI and VREF voltages.

In addition, there are six single-ended I/Os used that have the **same VCCI voltage. Since two banks are already assigned with the same VCCI voltage and there are enough u**nused bonded I/Os in those banks, the user does not need to assign the same VCCI voltage to another bank. The user needs to assign the other three VCCI voltages to three more banks.

Assigning Technologies and VREF to I/O Banks

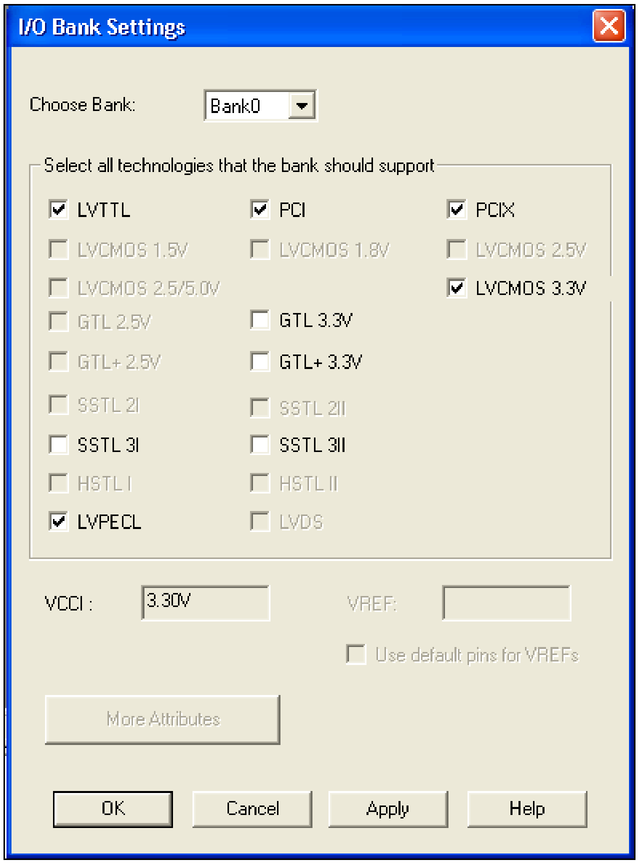
Low power flash devices offer a wide variety of I/O standards, including voltage-referenced standards. Before proceeding to Layout, each bank must have the required VCCI voltage assigned for the corresponding I/O technologies used for that bank. The voltage-referenced standards require the use of a reference voltage (VREF). This assignment can be done manually or automatically. The following sections describe this in detail.

Manually Assigning Technologies to I/O Banks

The user can import the PDC at this point and resolve this requirement. The PDC command is

```
set_iobank [bank name] –vcci [vcci value]
```

Another method is to use the I/O Bank Settings dialog box (**MVN** > **Edit** > **I/O Bank Settings**) to set up the V_{CCI} voltage for the bank (Figure 8-12).



I/O Bank Settings

Choose Bank: Bank0

Select all technologies that the bank should support

☑ LVTTL ☑ PCI ☑ PCIX
☐ LVCMOS 1.5V ☐ LVCMOS 1.8V ☐ LVCMOS 2.5V
☐ LVCMOS 2.5/5.0V ☑ LVCMOS 3.3V
☐ GTL 2.5V ☐ GTL 3.3V
☐ GTL+ 2.5V ☐ GTL+ 3.3V
☐ SSTL 2I ☐ SSTL 2II
☐ SSTL 3I ☐ SSTL 3II
☐ HSTL I ☐ HSTL II
☑ LVPECL ☐ LVDS

VCCI : 3.30V

VREF:

☐ Use default pins for VREFs

Figure 8-12 • **Setting VCCI for a Bank**

The procedure is as follows:

1. Select the bank to which you want VCCI to be assigned from the **Choose Bank** list.
2. Select the I/O standards for that bank. If you select any standard, the tool will automatically show all compatible standards that have a common VCCI voltage requirement.
3. Click **Apply**.
4. Repeat steps 1–3 to assign VCCI voltages to other banks. Refer to Figure 8-11 on page 211 to find out how many I/O banks are needed for VCCI bank assignment.

Manually Assigning VREF Pins

Voltage-referenced inputs require an input reference voltage (VREF). The user must assign VREF pins before running Layout. Before assigning a VREF pin, the user must set a VREF technology for the bank to which the pin belongs.

VREF Rules for the Implementation of Voltage-Referenced I/O Standards

The VREF rules are as follows:

1. Any I/O (except JTAG I/Os) can be used as a V_{REF} pin.
2. One V_{REF} pin can support up to 15 I/Os. It is recommended, but not required, that eight of them be on one side and seven on the other side (in other words, all 15 can still be on one side of VREF).
3. SSTL3 (I) and (II): Up to 40 I/Os per north or south bank in any position
4. LVPECL / GTL+ 3.3 V / GTL 3.3 V: Up to 48 I/Os per north or south bank in any position (not applicable for IGLOO nano and ProASIC3 nano devices)
5. SSTL2 (I) and (II) / GTL+ 2.5 V / GTL 2.5 V: Up to 72 I/Os per north or south bank in any position
6. VREF minibanks partition rule: Each I/O bank is physically partitioned into VREF minibanks. The VREF pins within a VREF minibank are interconnected internally, and consequently, only one VREF voltage can be used within each VREF minibank. If a bank does not require a VREF signal, the VREF pins of that bank are available as user I/Os.
7. The first VREF minibank includes all I/Os starting from one end of the bank to the first power triple and eight more I/Os after the power triple. Therefore, the first VREF minibank may contain (0 + 8), (2 + 8), (4 + 8), (6 + 8), or (8 + 8) I/Os.

   The second VREF minibank is adjacent to the first VREF minibank and contains eight I/Os, a power triple, and eight more I/Os after the triple. An analogous rule applies to all other VREF minibanks but the last.

   The last VREF minibank is adjacent to the previous one but contains eight I/Os, a power triple, and all I/Os left at the end of the bank. This bank may also contain (8 + 0), (8 + 2), (8 + 4), (8 + 6), or (8 + 8) available I/Os.

   Example:

   4 I/Os → Triple → 8 I/Os, 8 I/Os → Triple → 8 I/Os, 8 I/Os → Triple → 2 I/Os

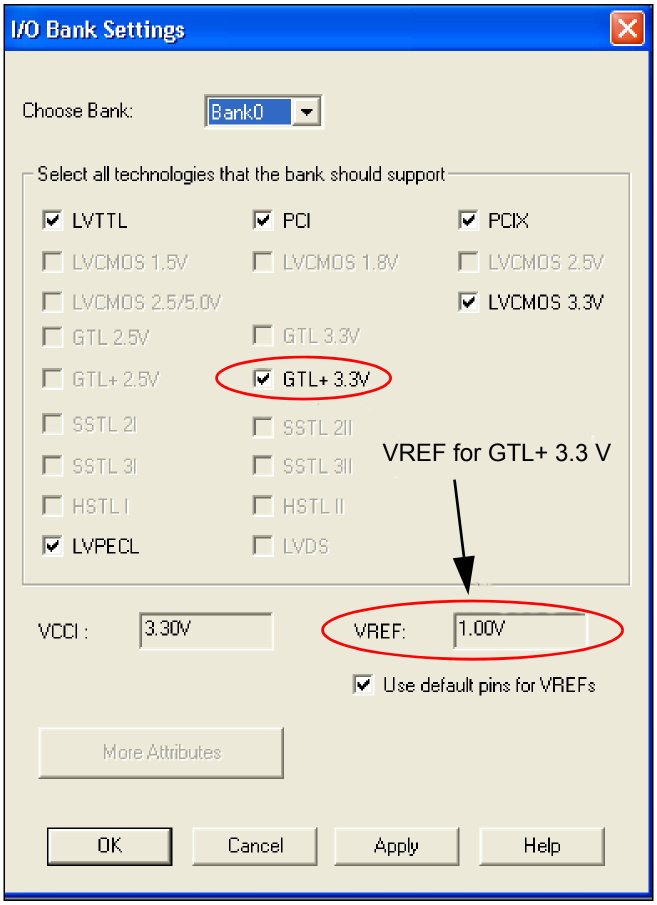
   That is, minibank A = (4 + 8) I/Os, minibank B = (8 + 8) I/Os, minibank C = (8 + 2) I/Os.
8. Only minibanks that contain input or bidirectional I/Os require a VREF. A VREF is not needed for minibanks composed of output or tristated I/Os.

Assigning the VREF Voltage to a Bank

When importing the PDC file, the VREF voltage can be assigned to the I/O bank. The PDC command is as follows:

```
set_iobank –vref [value]
```

Another method for assigning VREF is by using **MVN** > **Edit** > **I/O Bank Settings** (Figure 8-13 on page 214).



Choose Bank: Bank0

Select all technologies that the bank should support

☑ LVTTL ☑ PCI ☑ PCIX

□ LVCMOS 1.5V □ LVCMOS 1.8V □ LVCMOS 2.5V

□ LVCMOS 2.5/5.0V ☑ LVCMOS 3.3V

□ GTL 2.5V □ GTL 3.3V

□ GTL+ 2.5V ☑ GTL+ 3.3V

□ SSTL 2I □ SSTL 2II

□ SSTL 3I □ SSTL 3II

□ HSTL I □ HSTL II

☑ LVPECL □ LVDS

VCCI : 3.30V

VREF: 1.00V

☑ Use default pins for VREFs

VREF for GTL+ 3.3 V

Figure 8-13 **• Selecting VREF Voltage for the I/O Bank**

Assigning VREF Pins for a Bank

The user can use default pins for VREF. In this case, select the **Use default pins for VREFs** check box (Figure 8-13). This option guarantees full VREF coverage of the bank. The equivalent PDC command is as follows:

```
set_vref_default [bank name]
```

To be able to choose VREF pins, adequate VREF pins must be created to allow legal placement of the compatible voltage-referenced I/Os.

To assign VREF pins manually, the PDC command is as follows:

```
set_vref –bank [bank name] [package pin numbers]
```

For ChipPlanner/PinEditor to show the range of a VREF pin, perform the following steps:

1. Assign VCCI to a bank using **MVN** > **Edit** > **I/O Bank Settings**.
2. Open **ChipPlanner**. Zoom in on an I/O package pin in that bank.
3. Highlight the pin and then right-click. Choose **Use Pin for VREF**.

4. Right-click and then choose **Highlight VREF range**. All the pins covered by that VREF pin will be highlighted (Figure 8-14).

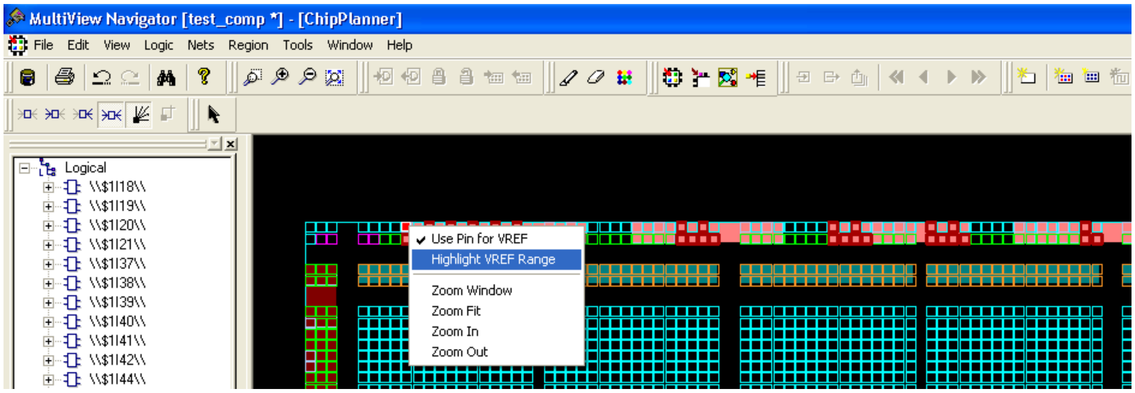


***Figure 8-14* • VREF Range**

Using PinEditor or ChipPlanner, VREF pins can also be assigned (Figure 8-15).

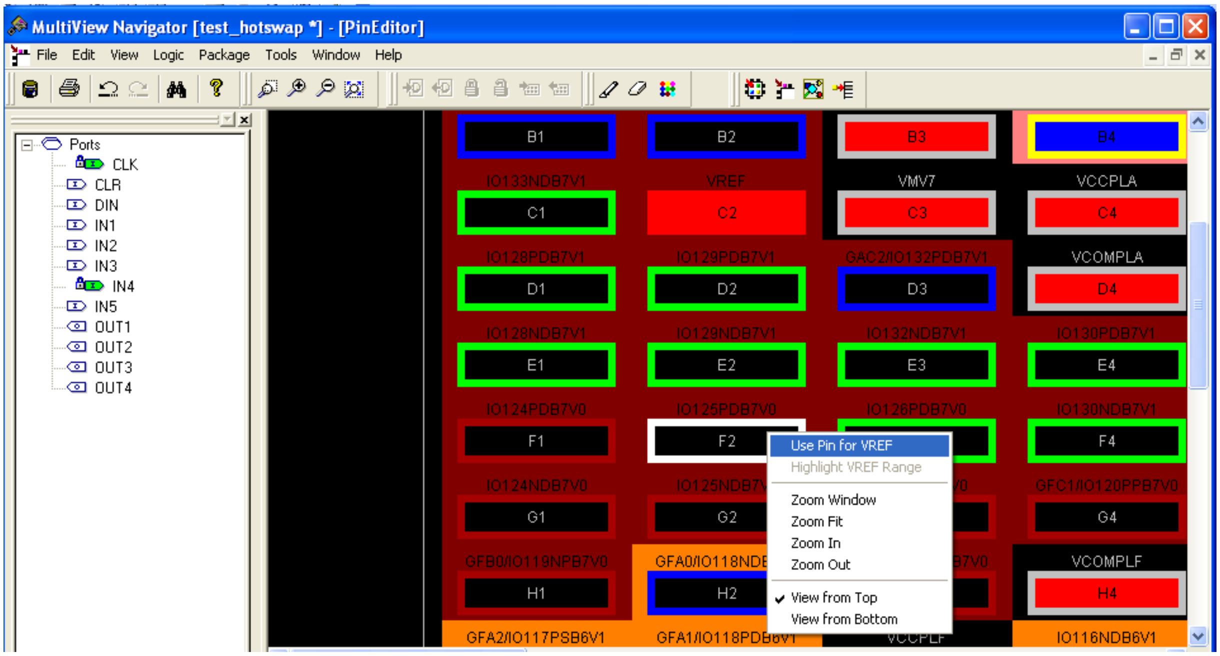


***Figure 8-15* • Assigning VREF from PinEditor**

To unassign a VREF pin:

1. Select the pin to unassign.
2. Right-click and choose **Use Pin for VREF.** The check mark next to the command disappears. The VREF pin is now a regular pin.

Resetting the pin may result in unassigning I/O cores, even if they are locked. In this case, a warning message appears so you can cancel the operation.

After you assign the VREF pins, right-click a VREF pin and choose **Highlight VREF Range** to see how many I/Os are covered by that pin. To unhighlight the range, choose **Unhighlight All** from the **Edit** menu.

Automatically Assigning Technologies to I/O Banks

The I/O Bank Assigner (IOBA) tool runs automatically when you run Layout. You can also use this tool from within the MultiView Navigator (Figure 8-17). The IOBA tool automatically assigns technologies and VREF pins (if required) to every I/O bank that does not currently have any technologies assigned to it. This tool is available when at least one I/O bank is unassigned.

To automatically assign technologies to I/O banks, choose I/O Bank Assigner from the **Tools** menu (or click the I/O Bank Assigner's toolbar button, shown in Figure 8-16).

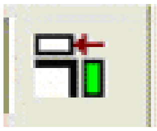
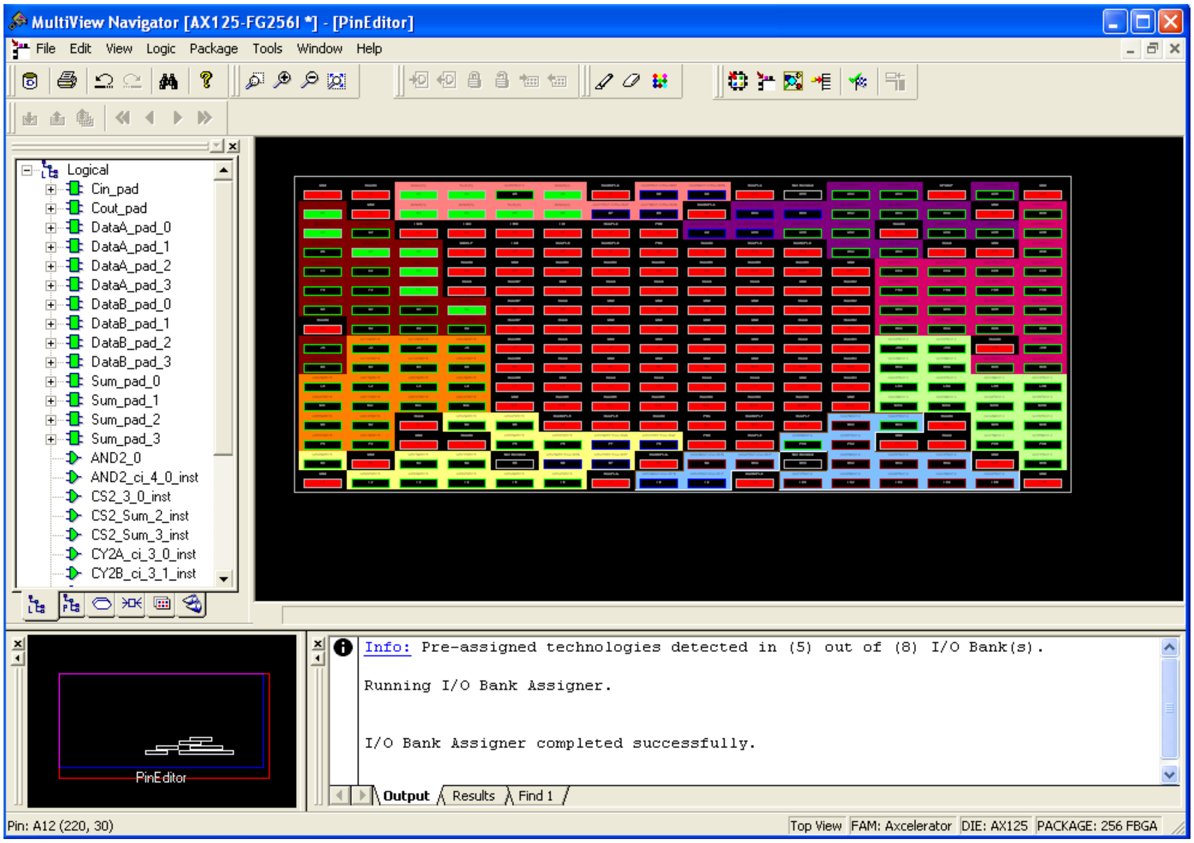


Figure 8-16 **• I/O Bank Assigner's Toolbar Button**

Messages will appear in the Output window informing you when the automatic I/O bank assignment begins and ends. If the assignment is successful, the message "I/O Bank Assigner completed successfully" appears in the Output window, as shown in Figure 8-17.



```
Info: Pre-assigned technologies detected in (5) out of (8) I/O Bank(s).

Running I/O Bank Assigner.

I/O Bank Assigner completed successfully.
```

Figure 8-17 **• I/O Bank Assigner Displays Messages in Output Window**

If the assignment is not successful, an error message appears in the Output window.

To undo the I/O bank assignments, choose **Undo** from the **Edit** menu. Undo removes the I/O technologies assigned by the IOBA. It does not remove the I/O technologies previously assigned.

To redo the changes undone by the Undo command, choose **Redo** from the **Edit** menu.

To clear I/O bank assignments made before using the Undo command, manually unassign or reassign I/O technologies to banks. To do so, choose **I/O Bank Settings** from the **Edit** menu to display the I/O Bank Settings dialog box.

Conclusion

Fusion, IGLOO, and ProASIC3 support for multiple I/O standards minimizes board-level components and makes possible a wide variety of applications. The Microsemi Designer software, integrated with Libero SoC, presents a clear visual display of I/O assignments, allowing users to verify I/O and board-level design requirements before programming the device. The device I/O features and functionalities ensure board designers can produce low-cost and low power FPGA applications fulfilling the complexities of contemporary design needs.

Related Documents

User's Guides

Libero SoC User's Guide

http://www.microsemi.com/soc/documents/libero_ug.pdf

IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide

http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

SmartGen Core Reference Guide

http://www.microsemi.com/soc/documents/genguide_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the document.

Date	Changes	Page
August 2012	The notes in Table 8-2 • Designer State (resulting from I/O attribute modification) were revised to clarify which device families support programmable input delay (SAR 39666).	201
June 2011	Figure 8-2 • SmartGen Catalog was updated (SAR 24310). Figure 8-3 • Expanded I/O Section and the step associated with it were deleted to reflect changes in the software.	202
	The following rule was added to the "VREF Rules for the Implementation of Voltage-Referenced I/O Standards" section: Only minibanks that contain input or bidirectional I/Os require a VREF. A VREF is not needed for minibanks composed of output or tristated I/Os (SAR 24310).	213
July 2010	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 8-1 • Flash-Based FPGAs.	200
	The notes for Table 8-2 • Designer State (resulting from I/O attribute modification) were revised to indicate that skew control and input delay do not apply to nano devices.	201
v1.3 (October 2008)	The "Flash FPGAs I/O Support" section was revised to include new families and make the information more concise.	200
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 8-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	200
v1.1 (March 2008)	This document was previously part of the *I/O Structures in IGLOO and ProASIC3 Devices* document. The content was separated and made into a new document.	N/A
	Table 8-2 • Designer State (resulting from I/O attribute modification) was updated to include note 2 for IGLOO PLUS.	201

9 – DDR for Microsemi's Low Power Flash Devices

Introduction

The I/Os in Fusion, IGLOO, and ProASIC3 devices support Double Data Rate (DDR) mode. In this mode, new data is present on every transition (or clock edge) of the clock signal. This mode doubles the data transfer rate compared with Single Data Rate (SDR) mode, where new data is present on one transition (or clock edge) of the clock signal. Low power flash devices have DDR circuitry built into the I/O tiles. I/Os are configured to be DDR receivers or transmitters by instantiating the appropriate special macros (examples shown in Figure 9-4 on page 224 and Figure 9-5 on page 225) and buffers (DDR_OUT or DDR_REG) in the RTL design. This document discusses the options the user can choose to configure the I/Os in this mode and how to instantiate them in the design.

Double Data Rate (DDR) Architecture

Low power flash devices support 350 MHz DDR inputs and outputs. In DDR mode, new data is present on every transition of the clock signal. Clock and data lines have identical bandwidths and signal integrity requirements, making them very efficient for implementing very high-speed systems. High-speed DDR interfaces can be implemented using LVDS (not applicable for IGLOO nano and ProASIC3 nano devices). In IGLOOe, ProASIC3E, AFS600, and AFS1500 devices, DDR interfaces can also be implemented using the HSTL, SSTL, and LVPECL I/O standards. The DDR feature is primarily implemented in the FPGA core periphery and is not tied to a specific I/O technology or limited to any I/O standard.

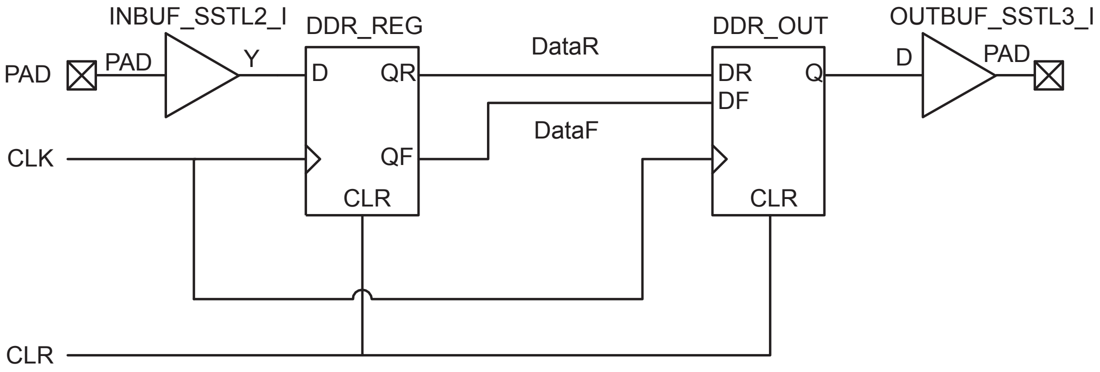


Figure 9-1 • **DDR Support in Low Power Flash Devices**

DDR Support in Flash-Based Devices

The flash FPGAs listed in Table 9-1 support the DDR feature and the functions described in this document.

Table 9-1 **• Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 9-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 9-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

I/O Cell Architecture

Low power flash devices support DDR in the I/O cells in four different modes: Input, Output, Tristate, and Bidirectional pins. For each mode, different I/O standards are supported, with most I/O standards having special sub-options. For the ProASIC3 nano and IGLOO nano devices, DDR is supported only in the 60 k, 125 k, and 250 k logic densities. Refer to Table 9-2 for a sample of the available I/O options. Additional I/O options can be found in the relevant family datasheet.

***Table 9-2* • DDR I/O Options**

DDR Register Type	I/O Type	I/O Standard	Sub-Options	Comments
Receive Register	Input	Normal	None	3.3 V TTL (default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
			Pull-Up	None (default)
		PCI/PCI-X	None	
		GTL/GTL+	Voltage	2.5 V, 3.3 V (3.3 V default)
		HSTL	Class	I / II (I default)
		SSTL2/SSTL3	Class	I / II (I default)
		LVPECL	None	
		LVDS	None	
Transmit Register	Output	Normal	None	3.3 V TTL (default)
		LVTTL	Output Drive	2, 4, 6, 8, 12, 16, 24, 36 mA (8 mA default)
			Slew Rate	Low/high (high default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
		PCI/PCI-X	None	
		GTL/GTL+	Voltage	1.8 V, 2.5 V, 3.3 V (3.3 V default)
		HSTL	Class	I / II (I default)
		SSTL2/SSTL3	Class	I / II (I default)
		LVPECL*	None	
		LVDS*	None	

*Note: *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

Table 9-2 • **DDR I/O Options (continued)**

DDR Register Type	I/O Type	I/O Standard	Sub-Options	Comments
Transmit Register (continued)	Tristate Buffer	Normal	Enable Polarity	Low/high (low default)
		LVTTL	Output Drive	2, 4, 6, 8, 12,16, 24, 36 mA (8 mA default)
			Slew Rate	Low/high (high default)
			Enable Polarity	Low/high (low default)
			Pull-Up/-Down	None (default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
			Output Drive	2, 4, 6, 8, 12, 16, 24, 36 mA (8 mA default)
			Slew Rate	Low/high (high default)
			Enable Polarity	Low/high (low default)
			Pull-Up/-Down	None (default)
		PCI/PCI-X	Enable Polarity	Low/high (low default)
		GTL/GTL+	Voltage	1.8 V, 2.5 V, 3.3 V (3.3 V default)
			Enable Polarity	Low/high (low default)
		HSTL	Class	I / II (I default)
			Enable Polarity	Low/high (low default)
		SSTL2/SSTL3	Class	I / II (I default)
			Enable Polarity	Low/high (low default)
	Bidirectional Buffer	Normal	Enable Polarity	Low/high (low default)
		LVTTL	Output Drive	2, 4, 6, 8, 12, 16, 24, 36 mA (8 mA default)
			Slew Rate	Low/high (high default)
			Enable Polarity	Low/high (low default)
			Pull-Up/-Down	None (default)
		LVCMOS	Voltage	1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default)
			Enable Polarity	Low/high (low default)
			Pull-Up	None (default)
		PCI/PCI-X	None	
			Enable Polarity	Low/high (low default)
		GTL/GTL+	Voltage	1.8 V, 2.5 V, 3.3 V (3.3 V default)
			Enable Polarity	Low/high (low default)
		HSTL	Class	I / II (I default)
			Enable Polarity	Low/high (low default)
		SSTL2/SSTL3	Class	I / II (I default)
			Enable Polarity	Low/high (low default)

*Note: *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

Input Support for DDR

The basic structure to support a DDR input is shown in Figure 9-2. Three input registers are used to capture incoming data, which is presented to the core on each rising edge of the I/O register clock. Each I/O tile supports DDR inputs.

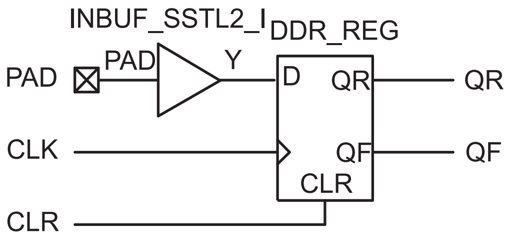


Figure 9-2 **• DDR Input Register Support in Low Power Flash Devices**

Output Support for DDR

The basic DDR output structure is shown in Figure 9-1 on page 219. New data is presented to the output every half clock cycle.

Note: DDR macros and I/O registers do not require additional routing. The combiner automatically recognizes the DDR macro and pushes its registers to the I/O register area at the edge of the chip. The routing delay from the I/O registers to the I/O buffers is already taken into account in the DDR macro.

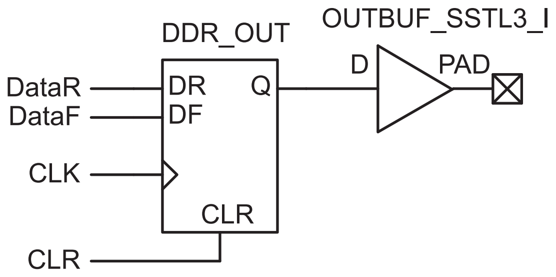
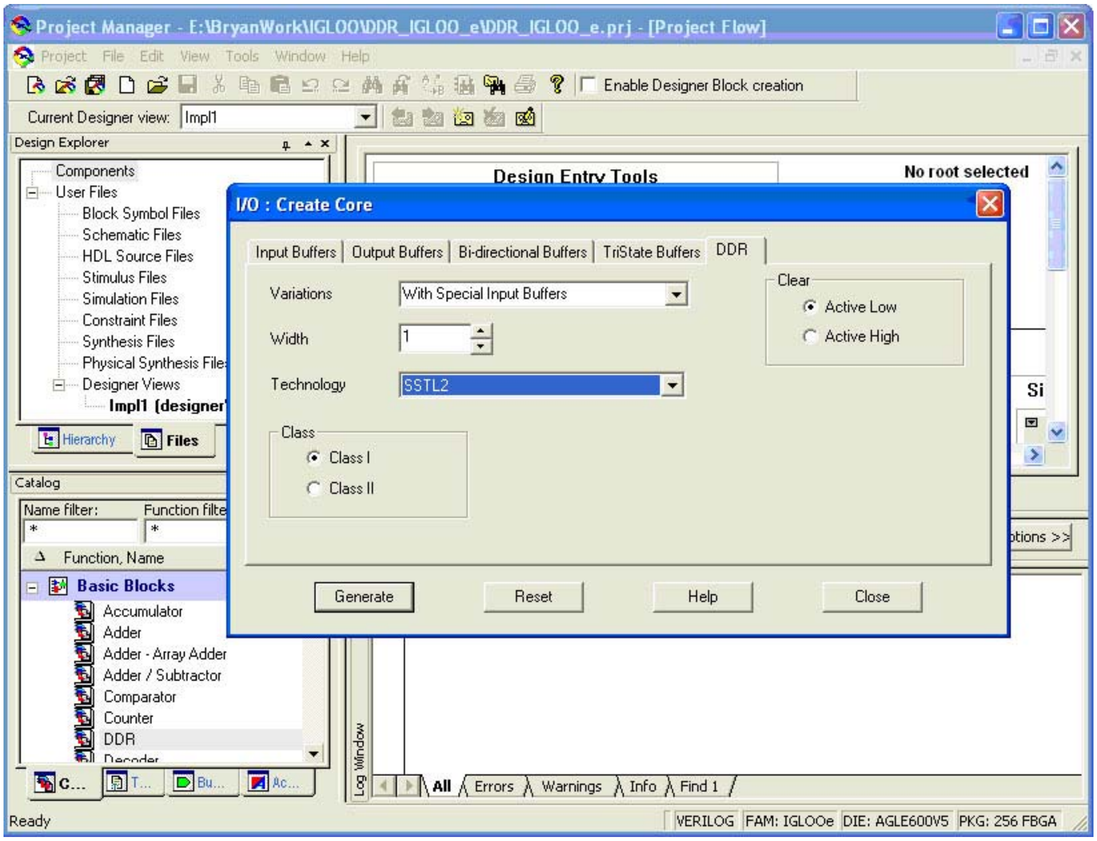


Figure 9-3 **• DDR Output Register (SSTL3 Class I)**

Instantiating DDR Registers

Using SmartGen is the simplest way to generate the appropriate RTL files for use in the design. Figure 9-4 shows an example of using SmartGen to generate a DDR SSTL2 Class I input register. SmartGen provides the capability to generate all of the DDR I/O cells as described. The user, through the graphical user interface, can select from among the many supported I/O standards. The output formats supported are Verilog, VHDL, and EDIF.

Figure 9-5 on page 225 through Figure 9-8 on page 228 show the I/O cell configured for DDR using SSTL2 Class I technology. For each I/O standard, the I/O pad is buffered by a special primitive that indicates the I/O standard type.



I/O : Create Core

Input Buffers | Output Buffers | Bi-directional Buffers | TriState Buffers | DDR

Variations: With Special Input Buffers

Width: 1

Technology: SSTL2

Class: ● Class I ○ Class II

Clear: ● Active Low ○ Active High

Generate | Reset | Help | Close

Figure 9-4 **• Example of Using SmartGen to Generate a DDR SSTL2 Class I Input Register**

DDR Input Register

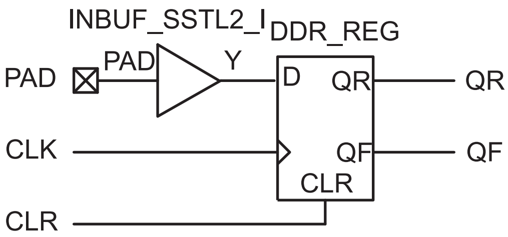


Figure 9-5 • **DDR Input Register (SSTL2 Class I)**

The corresponding structural representations, as generated by SmartGen, are shown below:

Verilog

```
module DDR_InBuf_SSTL2_I(PAD,CLR,CLK,QR,QF);

input   PAD, CLR, CLK;
output  QR, QF;

wire Y;

  INBUF_SSTL2_I INBUF_SSTL2_I_0_inst(.PAD(PAD),.Y(Y));
  DDR_REG DDR_REG_0_inst(.D(Y),.CLK(CLK),.CLR(CLR),.QR(QR),.QF(QF));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
--The correct library will be inserted automatically by SmartGen
library proasic3; use proasic3.all;
--library fusion; use fusion.all;
--library igloo; use igloo.all;

entity DDR_InBuf_SSTL2_I is
  port(PAD, CLR, CLK : in std_logic;  QR, QF : out std_logic) ;
end DDR_InBuf_SSTL2_I;

architecture DEF_ARCH of  DDR_InBuf_SSTL2_I is

  component INBUF_SSTL2_I
    port(PAD : in std_logic := 'U'; Y : out std_logic) ;
  end component;

  component DDR_REG
    port(D, CLK, CLR : in std_logic := 'U'; QR, QF : out std_logic) ;
  end component;

signal Y : std_logic ;

begin

  INBUF_SSTL2_I_0_inst : INBUF_SSTL2_I
  port map(PAD => PAD, Y => Y);
  DDR_REG_0_inst : DDR_REG
  port map(D => Y, CLK => CLK, CLR => CLR, QR => QR, QF => QF);

end DEF_ARCH;
```

DDR Output Register

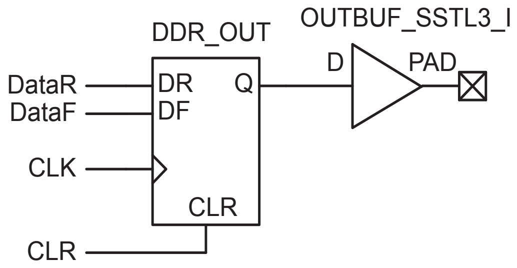


Figure 9-6 • **DDR Output Register (SSTL3 Class I)**

Verilog

```
module DDR_OutBuf_SSTL3_I(DataR,DataF,CLR,CLK,PAD);

input   DataR, DataF, CLR, CLK;
output  PAD;

wire Q, VCC;

  VCC VCC_1_net(.Y(VCC));
  DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
  OUTBUF_SSTL3_I OUTBUF_SSTL3_I_0_inst(.D(Q),.PAD(PAD));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3; use proasic3.all;

entity DDR_OutBuf_SSTL3_I is
  port(DataR, DataF, CLR, CLK : in std_logic;  PAD : out std_logic) ;
end DDR_OutBuf_SSTL3_I;

architecture DEF_ARCH of  DDR_OutBuf_SSTL3_I is

  component DDR_OUT
    port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
  end component;

  component OUTBUF_SSTL3_I
    port(D : in std_logic := 'U'; PAD : out std_logic) ;
  end component;

  component VCC
    port( Y : out std_logic);
  end component;

signal Q, VCC_1_net : std_logic ;

begin

  VCC_2_net : VCC port map(Y => VCC_1_net);
  DDR_OUT_0_inst : DDR_OUT
  port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
  OUTBUF_SSTL3_I_0_inst : OUTBUF_SSTL3_I
  port map(D => Q, PAD => PAD);

end DEF_ARCH;
```

DDR Tristate Output Register

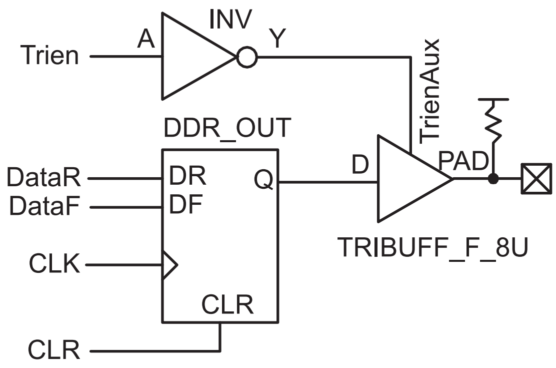


Figure 9-7 • **DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTL)**

Verilog

```
module DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp(DataR, DataF, CLR, CLK, Trien,
  PAD);

input   DataR, DataF, CLR, CLK, Trien;
output  PAD;

wire TrienAux, Q;

  INV Inv_Tri(.A(Trien),.Y(TrienAux));
  DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
  TRIBUFF_F_8U TRIBUFF_F_8U_0_inst(.D(Q),.E(TrienAux),.PAD(PAD));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3; use proasic3.all;

entity DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is
  port(DataR, DataF, CLR, CLK, Trien : in std_logic;  PAD : out std_logic) ;
end DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp;

architecture DEF_ARCH of DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is

  component INV
    port(A : in std_logic := 'U'; Y : out std_logic) ;
  end component;

  component DDR_OUT
    port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
  end component;

  component TRIBUFF_F_8U
    port(D, E : in std_logic := 'U'; PAD : out std_logic) ;
  end component;

signal TrienAux, Q : std_logic ;

begin

  Inv_Tri : INV
  port map(A => Trien, Y => TrienAux);
  DDR_OUT_0_inst : DDR_OUT
  port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
  TRIBUFF_F_8U_0_inst : TRIBUFF_F_8U
  port map(D => Q, E => TrienAux, PAD => PAD);

end DEF_ARCH;
```

DDR Bidirectional Buffer

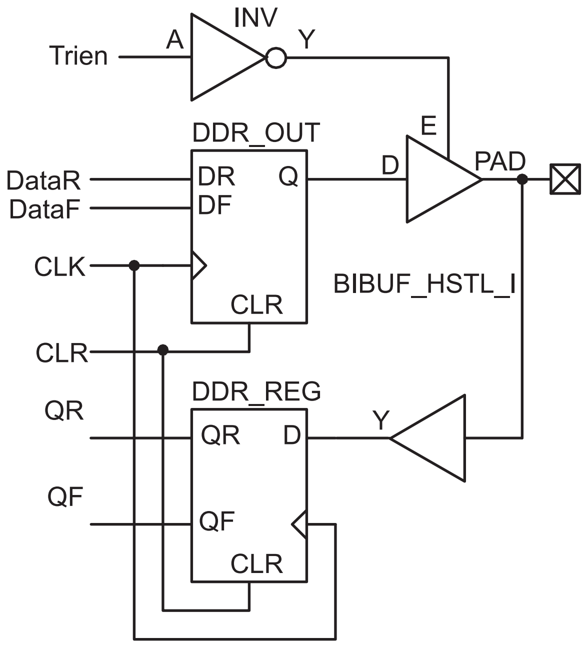


Figure 9-8 • **DDR Bidirectional Buffer, LOW Output Enable (HSTL Class II)**

Verilog

```
module DDR_BiDir_HSTL_I_LowEnb(DataR,DataF,CLR,CLK,Trien,QR,QF,PAD);

input   DataR, DataF, CLR, CLK, Trien;
output  QR, QF;
inout   PAD;

wire TrienAux, D, Q;

  INV Inv_Tri(.A(Trien), .Y(TrienAux));
  DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
  DDR_REG DDR_REG_0_inst(.D(D),.CLK(CLK),.CLR(CLR),.QR(QR),.QF(QF));
  BIBUF_HSTL_I BIBUF_HSTL_I_0_inst(.PAD(PAD),.D(Q),.E(TrienAux),.Y(D));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3; use proasic3.all;

entity DDR_BiDir_HSTL_I_LowEnb is
  port(DataR, DataF, CLR, CLK, Trien : in std_logic; QR, QF : out std_logic;
    PAD : inout std_logic) ;
end DDR_BiDir_HSTL_I_LowEnb;

architecture DEF_ARCH of  DDR_BiDir_HSTL_I_LowEnb is

  component INV
    port(A : in std_logic := 'U'; Y : out std_logic) ;
  end component;

  component DDR_OUT
    port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
  end component;

  component DDR_REG
    port(D, CLK, CLR : in std_logic := 'U'; QR, QF : out std_logic) ;
  end component;

  component BIBUF_HSTL_I
    port(PAD : inout std_logic := 'U'; D, E : in std_logic := 'U'; Y : out std_logic) ;
  end component;

signal TrienAux, D, Q : std_logic ;

begin

  Inv_Tri : INV
  port map(A => Trien, Y => TrienAux);
  DDR_OUT_0_inst : DDR_OUT
  port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
  DDR_REG_0_inst : DDR_REG
  port map(D => D, CLK => CLK, CLR => CLR, QR => QR, QF => QF);
  BIBUF_HSTL_I_0_inst : BIBUF_HSTL_I
  port map(PAD => PAD, D => Q, E => TrienAux, Y => D);

end DEF_ARCH;
```

Design Example

Figure 9-9 shows a simple example of a design using both DDR input and DDR output registers. The user can copy the HDL code in Libero SoC software and go through the design flow. Figure 9-10 and Figure 9-11 on page 231 show the netlist and ChipPlanner views of the ddr_test design. Diagrams may vary slightly for different families.

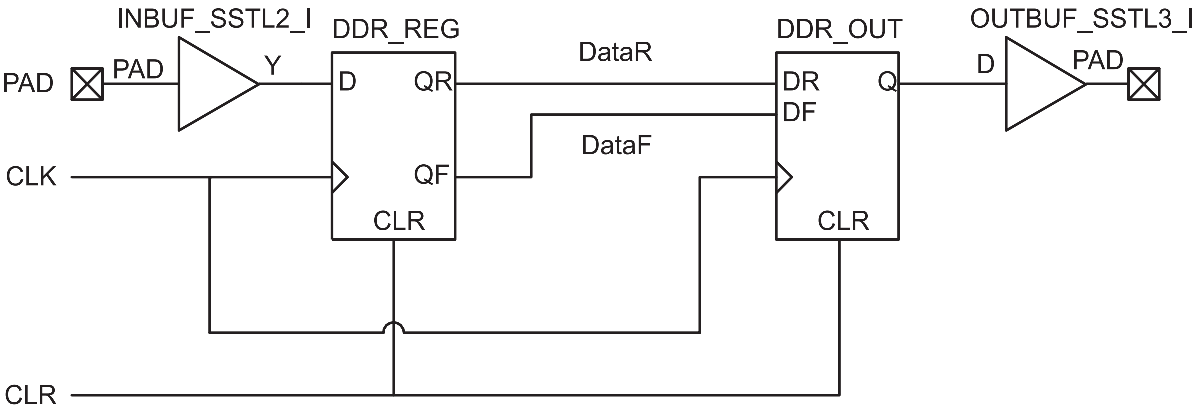


Figure 9-9 • **Design Example**

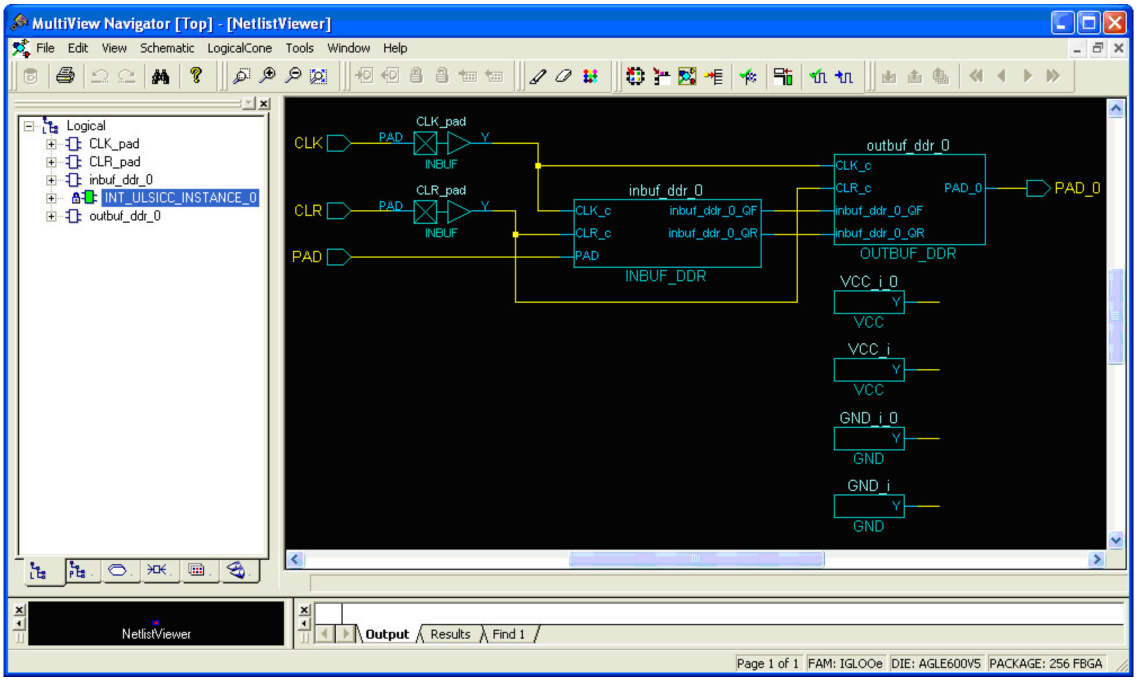


Figure 9-10 • **DDR Test Design as Seen by NetlistViewer for IGLOO/e Devices**

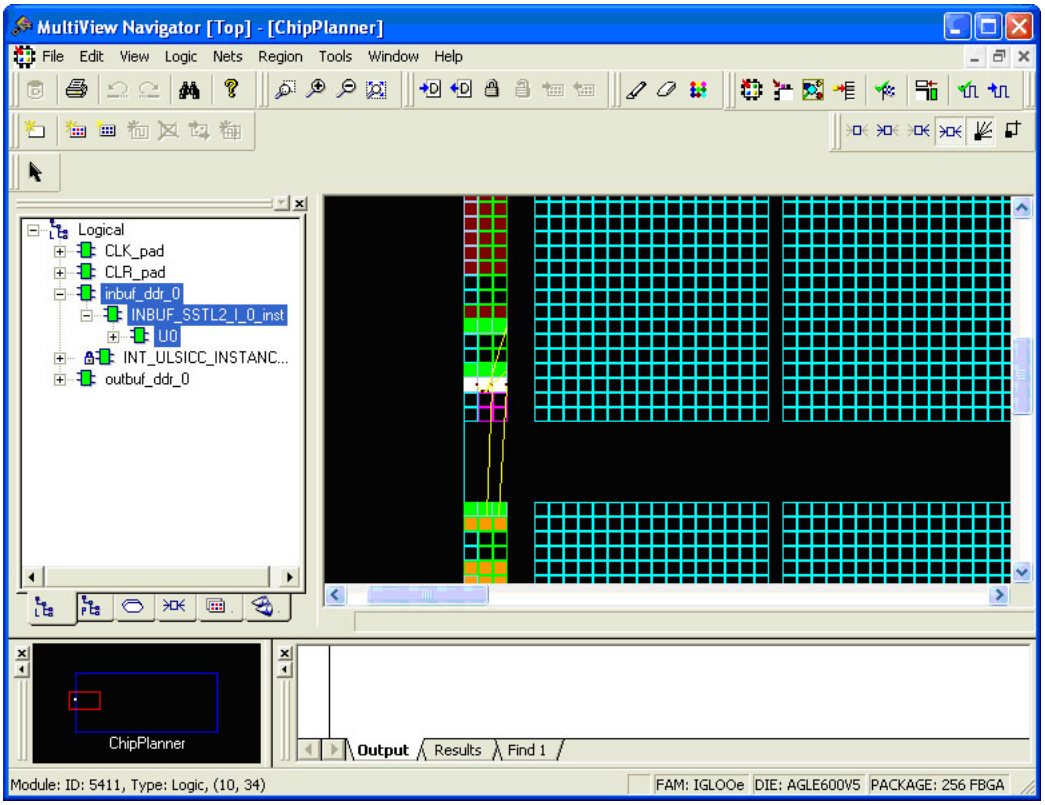


Figure 9-11 • **DDR Input/Output Cells as Seen by ChipPlanner for IGLOO/e Devices**

Verilog

```
module Inbuf_ddr(PAD,CLR,CLK,QR,QF);

input PAD, CLR, CLK;
output  QR, QF;

wire Y;

  DDR_REG DDR_REG_0_inst(.D(Y), .CLK(CLK), .CLR(CLR), .QR(QR), .QF(QF));
  INBUF INBUF_0_inst(.PAD(PAD), .Y(Y));

endmodule

module Outbuf_ddr(DataR,DataF,CLR,CLK,PAD);

input DataR, DataF, CLR, CLK;
output  PAD;

wire Q, VCC;

  VCC VCC_1_net(.Y(VCC));
  DDR_OUT DDR_OUT_0_inst(.DR(DataR), .DF(DataF), .CLK(CLK), .CLR(CLR), .Q(Q));
  OUTBUF OUTBUF_0_inst(.D(Q), .PAD(PAD));

endmodule
```

```
module ddr_test(DIN, CLK, CLR, DOUT);

input  DIN, CLK, CLR;
output DOUT;

  Inbuf_ddr Inbuf_ddr (.PAD(DIN), .CLR(clr), .CLK(clk), .QR(qr), .QF(qf));
  Outbuf_ddr Outbuf_ddr (.DataR(qr),.DataF(qf), .CLR(clr), .CLK(clk),.PAD(DOUT));

  INBUF INBUF_CLR (.PAD(CLR), .Y(clr));
  INBUF INBUF_CLK (.PAD(CLK), .Y(clk));

endmodule
```

Simulation Consideration

Microsemi DDR simulation models use inertial delay modeling by default (versus transport delay modeling). As such, pulses that are shorter than the actual gate delays should be avoided, as they will not be seen by the simulator and may be an issue in post-routed simulations. The user must be aware of the default delay modeling and must set the correct delay model in the simulator as needed.

Conclusion

Fusion, IGLOO, and ProASIC3 devices support a wide range of DDR applications with different I/O standards and include built-in DDR macros. The powerful capabilities provided by SmartGen and its GUI can simplify the process of including DDR macros in designs and minimize design errors. Additional considerations should be taken into account by the designer in design floorplanning and placement of I/O flip-flops to minimize datapath skew and to help improve system timing margins. Other system-related issues to consider include PLL and clock partitioning.

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 9-1 • Flash-Based FPGAs.	220
	The "I/O Cell Architecture" section was updated with information applicable to nano devices.	221
	The output buffer (OUTBUF_SSTL3_I) input was changed to D, instead of Q, in Figure 9-1 • DDR Support in Low Power Flash Devices, Figure 9-3 • DDR Output Register (SSTL3 Class I), Figure 9-6 • DDR Output Register (SSTL3 Class I), Figure 9-7 • DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTL), and the output from the DDR_OUT macro was connected to the input of the TRIBUFF macro in Figure 9-7 • DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTL).	219, 223, 226, 227
v1.3 (October 2008)	The "Double Data Rate (DDR) Architecture" section was updated to include mention of the AFS600 and AFS1500 devices.	219
	The "DDR Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	220
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 9-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	220
v1.1 (March 2008)	The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	220

10 – Migrating Designs in ProASIC3 Devices from Higher-Density to Mid-Density Devices

Introduction

The purpose of this document is to assist in migrating designs in ProASIC®3 A3P1000, A3P600, and A3P400 devices from higher-density to mid-density devices. There are three possible migration paths:

- A3P1000 to A3P600
- A3P1000 to A3P400
- A3P600 to A3P400

Since one of the key factors is pin compatibility, this document addresses pin compatibility for all available packages common to the A3P1000, A3P600, and A3P400 devices.

Design Migration

ProASIC3 family devices are architecturally compatible with each other. However, designers must pay attention to a few key areas when migrating a design. The specific issues discussed throughout this application note are as follows:

- "Design and Device Evaluation"
- "Device and Package Compatibility" on page 236
- "Migration and Implementation Methodologies" on page 237
- "I/O Banks and Standards" on page 238
- "Power Supply and Board-Level Considerations" on page 238
- "Pin Migration and Compatibility" on page 239

Design and Device Evaluation

When migrating a design, the primary task should be to compare the available resources between the two devices. The designer should evaluate effective gate count, RAM size, I/O banks, and number of I/Os (Table 10-1). In addition, necessary design timing analysis and simulations should be validated when porting designs to new ProASIC3 derivatives.

***Table 10-1* • Device Information**

	A3P1000	A3P600	A3P400
System Gates	1 M	600 k	400 k
Tiles (D-flip-flop)	24,576	13,824	9,126
RAM (kbits)	144	108	54
RAM Blocks (4,608 bits)	32	24	12
I/O Banks (+ JTAG)	4	4	4
Maximum User I/Os per Package			
PQ208	154/35	154/35	151/34
FG144	97/25	97/25	97/25
FG256	177/44	177/43	178/38
FG484	300/74	235/60	194/38

Note: Maximum user I/O is listed as single-ended/double-ended.

Device and Package Compatibility

ProASIC3 devices and packaging were designed to allow considerable footprint compatibility for smoother migration.

Common and Convertible I/Os between A3P400, A3P600, and A3P1000 Devices

Table 10-2 shows the number of I/Os that are common between any two of these devices, as well as the number of I/Os that will require conversion per the suggested design migration rules given in the "Migration and Implementation Methodologies" section on page 237.

Table 10-2 **• Common and Convertible I/Os**

	A3P1000 A3P600		A3P1000 A3P400		A3P600 A3P400	
Package	**Common I/Os**	**Convertible I/Os**	**Common I/Os**	**Convertible I/Os**	**Common I/Os**	**Convertible I/Os**
PQ208	154	0	154	5	151	5
FG144	97	0	97	0	98	0
FG256	178	31	177	59	177	33
FG484	236	96	192	75	236	166

Migration and Implementation Methodologies

Table 10-3 lists some possible migration combinations and the recommended implementation rules for compatible design conversions from higher-density to lower-density devices. The "Pin Migration and Compatibility" section on page 239 contains tables that list the required rules for different pin combinations. If “Rule x” is mentioned for a pin combination, that combination requires the implementation methodology given in Table 10-3. Note that many combinations of high-density/low-density pins do not require these rules; the pins have complete type compatibility. These pins are marked in the pin tables with “None.”

***Table 10-3* • Migration Rules from Higher-Density to Mid-Density Devices**

Migration Rule	Issue		Implementation Methodology
	Higher Density	Lower Density	
1	I/O or global I/O	NC	Leave this pin floating OR program I/Os as unused (software cannot program NC to usable I/O).
2	Single-ended I/O	Global I/O	Instantiate the I/O buffer as a global single-ended I/O.
3	Global I/O	Single-ended I/O	Use the physical design constraint (PDC) to promote the single-ended I/O to a global pin. There is an additional delay that affects the setup time on the board. Or, do not use this pin as a global input on the higher-density device.
4	VCC or VCCIB(x) [1,3]	NC	Leave pin connected to board VCC or VCCIBx plane.
5	VCCIB(x) [1]	VCCIB(y) [2]	Make sure the two bank voltage levels are the same. Tie the pin to the board's corresponding VCCIBx/VMVx plane.
6	VMV(x) [1]	VMV(y) [2]	Make sure the two bank voltage levels are the same. Tie pin to the board's corresponding VCCIBx/VMVx plane.
7	VMV(x) [2]	I/O or global I/O	Leave the pin tied to the board VCCIBx/VMVx plane. Instantiate the I/Os as tristate buffers with OE = 0 and no weak pull-ups/-downs.
8	GNDQ	Global I/O	Leave both pins tied to board GNDQ plane. Instantiate the I/O as tristate buffer with OE = 0 and no weak pull-ups/-downs.
9	GNDQ	NC	GNDQ and NC need to be connected to GND.
10	NC	VCC or VCCIB(x) [1,3]	Leave pin connected to board VCC or VCCIBx plane.

Notes:

1. *(x) = 1, 2, 3, or 4*
2. *(y) = 1, 2, 3, or 4*
3. *Refer to the "I/O Structures in IGLOO and ProASIC3 Devices" chapter of the IGLOO FPGA Fabric User's Guide or ProASIC3 FPGA Fabric User's Guide for I/O naming conventions.*

I/O Banks and Standards

ProASIC3 I/Os are partitioned into multiple I/O voltage banks. The number of banks is device-dependent. There are four I/O banks in A3P1000, A3P600, and A3P400 devices.

Package VCCIBx pins are routed through the corresponding banks of the devices.

The banks have dedicated supplies; therefore, only I/Os with compatible voltage standards can be assigned to the same I/O voltage bank.

Power Supply and Board-Level Considerations

I/O power supply requirements are one of the key aspects to consider for design migration. Since the migration is within the ProASIC3 family, there is no issue with respect to the core voltage, VCC. Pins that must be appropriately connected are VCCIBx (bank supply voltage to I/O output buffer and I/O logic), VMVx (quiet I/O supply voltage), GNDQ (quiet GND), and GND. An important function of GNDQ and VMVx is to decouple simultaneous switching noise for outputs (SSOs) to enhance signal integrity and improve noise immunity.

The following are the *key rules* of migration for the above-mentioned pins:

- VMVx and VCCIBx must be at the same voltage level for a given bank.
- VCCIBx pins and VMVx pins in unused banks must be connected to GND.
- Unused I/Os are automatically disabled by software.

A specific power-supply sequence at power-up is not required.

Any incorrect connection during the migration may affect overall dynamic or inrush power consumption and might even result in device malfunction.

Additionally, the I/O naming convention in ProASIC3 devices has significant embedded information (i.e., pin location, bank number, signal type, polarity, and clock conditioning). For a detailed explanation, refer to the “User I/O Naming Convention” section in the "I/O Structures in IGLOO and ProASIC3 Devices" *chapter of the IGLOO FPGA Fabric User’s Guide or ProASIC3 FPGA Fabric User’s Guide*. This datasheet also contains additional information on power issues.

Pin Migration and Compatibility

PQ208 Package

Table 10-4 • **Pin Compatibility and Migration Table for the PQ208 Package**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
1	GND	GND	GND	None	None	None
2	GAA2/IO225PDB3	GAA2/IO170PDB3	GAA2/IO155UDB3	None	None	None
3	IO225NDB3	IO170NDB3	IO155VDB3	None	None	None
4	GAB2/IO224PDB3	GAB2/IO169PDB3	GAB2/IO154UDB3	None	None	None
5	IO224NDB3	IO169NDB3	IO154VDB3	None	None	None
6	GAC2/IO223PDB3	GAC2/IO168PDB3	GAC2/IO153UDB3	None	None	None
7	IO223NDB3	IO168NDB3	IO153VDB3	None	None	None
8	IO222PDB3	IO167PDB3	IO152UDB3	None	None	None
9	IO222NDB3	IO167NDB3	IO152VDB3	None	None	None
10	IO220PDB3	IO166PDB3	IO151UDB3	None	None	None
11	IO220NDB3	IO166NDB3	IO151VDB3	None	None	None
12	IO218PDB3	IO165PDB3	IO150PDB3	None	None	None
13	IO218NDB3	IO165NDB3	IO150NDB3	None	None	None
14	IO216PDB3	IO164PDB3	IO149PDB3	None	None	None
15	IO216NDB3	IO164NDB3	IO149NDB3	None	None	None
16	VCC	VCC	VCC	None	None	None
17	GND	GND	GND	None	None	None
18	VCCIB3	VCCIB3	VCCIB3	None	None	None
19	IO212PDB3	IO163PDB3	IO148PDB3	None	None	None
20	IO212NDB3	IO163NDB3	IO148NDB3	None	None	None
21	GFC1/IO209PDB3	GFC1/IO161PDB3	GFC1/IO147PDB3	None	None	None
22	GFC0/IO209NDB3	GFC0/IO161NDB3	GFC0/IO147NDB3	None	None	None
23	GFB1/IO208PDB3	GFB1/IO160PDB3	GFB1/IO146PDB3	None	None	None
24	GFB0/IO208NDB3	GFB0/IO160NDB3	GFB0/IO146NDB3	None	None	None
25	VCOMPLF	VCOMPLF	VCOMPLF	None	None	None
26	GFA0/IO207NPB3	GFA0/IO159NPB3	GFA0/IO145NPB3	None	None	None
27	VCCPLF	VCCPLF	VCCPLF	None	None	None
28	GFA1/IO207PPB3	GFA1/IO159PPB3	GFA1/IO145PPB3	None	None	None
29	GND	GND	GND	None	None	None

***Table 10-4* • Pin Compatibility and Migration Table for the PQ208 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
30	GFA2/IO206PDB3	GFA2/IO158PDB3	GFA2/IO144PDB3	None	None	None
31	IO206NDB3	IO158NDB3	IO144NDB3	None	None	None
32	GFB2/IO205PDB3	GFB2/IO157PDB3	GFB2/IO143PDB3	None	None	None
33	IO205NDB3	IO157NDB3	IO143NDB3	None	None	None
34	GFC2/IO204PDB3	GFC2/IO156PDB3	GFC2/IO142PDB3	None	None	None
35	IO204NDB3	IO156NDB3	IO142NDB3	None	None	None
36	VCC	VCC	NC	None	**Rule 4**	**Rule 4**
37	IO199PDB3	IO147PDB3	IO141PSB3	None	None	None
38	IO199NDB3	IO147NDB3	IO140PDB3	None	None	None
39	IO197PSB3	IO146PSB3	IO140NDB3	None	None	None
40	VCCIB3	VCCIB3	VCCIB3	None	None	None
41	GND	GND	GND	None	None	None
42	IO191PDB3	IO145PDB3	IO138PDB3	None	None	None
43	IO191NDB3	IO145NDB3	IO138NDB3	None	None	None
44	GEC1/IO190PDB3	GEC1/IO144PDB3	GEC1/IO137PDB3	None	None	None
45	GEC0/IO190NDB3	GEC0/IO144NDB3	GEC0/IO137NDB3	None	None	None
46	GEB1/IO189PDB3	GEB1/IO143PDB3	GEB1/IO136PDB3	None	None	None
47	GEB0/IO189NDB3	GEB0/IO143NDB3	GEB0/IO136NDB3	None	None	None
48	GEA1/IO188PDB3	GEA1/IO142PDB3	GEA1/IO135PDB3	None	None	None
49	GEA0/IO188NDB3	GEA0/IO142NDB3	GEA0/IO135NDB3	None	None	None
50	VMV3	VMV3	VMV3	None	None	None
51	GNDQ	GNDQ	GNDQ	None	None	None
52	GND	GND	GND	None	None	None
53	VMV2	VMV2	VMV2	None	None	None
54	GEA2/IO187RSB2	GEA2/IO141RSB2	NC	None	**Rule 1**	**Rule 1**
55	GEB2/IO186RSB2	GEB2/IO140RSB2	GEA2/IO134RSB2	None	None	None
56	GEC2/IO185RSB2	GEC2/IO139RSB2	GEB2/IO133RSB2	None	None	None
57	IO184RSB2	IO138RSB2	GEC2/IO132RSB2	None	**Rule 2**	**Rule 2**
58	IO183RSB2	IO137RSB2	IO131RSB2	None	None	None
59	IO182RSB2	IO136RSB2	IO130RSB2	None	None	None
60	IO181RSB2	IO135RSB2	IO129RSB2	None	None	None
61	IO180RSB2	IO134RSB2	IO128RSB2	None	None	None
62	VCCIB2	VCCIB2	VCCIB2	None	None	None

Table 10-4 **• Pin Compatibility and Migration Table for the PQ208 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
63	IO178RSB2	IO133RSB2	IO125RSB2	None	None	None
64	IO176RSB2	IO131RSB2	IO123RSB2	None	None	None
65	GND	GND	GND	None	None	None
66	IO174RSB2	IO129RSB2	IO121RSB2	None	None	None
67	IO172RSB2	IO127RSB2	IO119RSB2	None	None	None
68	IO170RSB2	IO125RSB2	IO117RSB2	None	None	None
69	IO168RSB2	IO123RSB2	IO115RSB2	None	None	None
70	IO166RSB2	IO121RSB2	IO113RSB2	None	None	None
71	VCC	VCC	VCC	None	None	None
72	VCCIB2	VCCIB2	VCCIB2	None	None	None
73	IO162RSB2	IO118RSB2	IO112RSB2	None	None	None
74	IO160RSB2	IO117RSB2	IO111RSB2	None	None	None
75	IO158RSB2	IO116RSB2	IO110RSB2	None	None	None
76	IO156RSB2	IO115RSB2	IO109RSB2	None	None	None
77	IO154RSB2	IO114RSB2	IO108RSB2	None	None	None
78	IO152RSB2	IO113RSB2	IO107RSB2	None	None	None
79	IO150RSB2	IO112RSB2	IO106RSB2	None	None	None
80	IO148RSB2	IO110RSB2	IO104RSB2	None	None	None
81	GND	GND	GND	None	None	None
82	IO143RSB2	IO109RSB2	IO102RSB2	None	None	None
83	IO141RSB2	IO108RSB2	IO101RSB2	None	None	None
84	IO139RSB2	IO107RSB2	IO100RSB2	None	None	None
85	IO137RSB2	IO106RSB2	IO99RSB2	None	None	None
86	IO135RSB2	IO105RSB2	IO98RSB2	None	None	None
87	IO133RSB2	IO104RSB2	IO97RSB2	None	None	None
88	VCC	VCC	VCC	None	None	None
89	VCCIB2	VCCIB2	VCCIB2	None	None	None
90	IO128RSB2	IO102RSB2	IO94RSB2	None	None	None
91	IO126RSB2	IO100RSB2	IO92RSB2	None	None	None
92	IO124RSB2	IO98RSB2	IO90RSB2	None	None	None
93	IO122RSB2	IO96RSB2	IO88RSB2	None	None	None
94	IO120RSB2	IO94RSB2	IO86RSB2	None	None	None
95	IO118RSB2	IO90RSB2	IO84RSB2	None	None	None

***Table 10-4* • Pin Compatibility and Migration Table for the PQ208 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
96	GDC2/IO116RSB2	GDC2/IO89RSB2	GDC2/IO82RSB2	None	None	None
97	GND	GND	GND	None	None	None
98	GDB2/IO115RSB2	GDB2/IO88RSB2	GDB2/IO81RSB2	None	None	None
99	GDA2/IO114RSB2	GDA2/IO87RSB2	GDA2/IO80RSB2	None	None	None
100	GNDQ	GNDQ	GNDQ	None	None	None
101	TCK	TCK	TCK	None	None	None
102	TDI	TDI	TDI	None	None	None
103	TMS	TMS	TMS	None	None	None
104	VMV2	VMV2	VMV2	None	None	None
105	GND	GND	GND	None	None	None
106	VPUMP	VPUMP	VPUMP	None	None	None
107	GNDQ	GNDQ	NC	None	**Rule 8**	**Rule 8**
108	TDO	TDO	TDO	None	None	None
109	TRST	TRST	TRST	None	None	None
110	VJTAG	VJTAG	VJTAG	None	None	None
111	GDA0/IO113NDB1	GDA0/IO86NDB1	GDA0/IO79VDB1	None	None	None
112	GDA1/IO113PDB1	GDA1/IO86PDB1	GDA1/IO79UDB1	None	None	None
113	GDB0/IO112NDB1	GDB0/IO85NDB1	GDB0/IO78VDB1	None	None	None
114	GDB1/IO112PDB1	GDB1/IO85PDB1	GDB1/IO78UDB1	None	None	None
115	GDC0/IO111NDB1	GDC0/IO84NDB1	GDC0/IO77VDB1	None	None	None
116	GDC1/IO111PDB1	GDC1/IO84PDB1	GDC1/IO77UDB1	None	None	None
117	IO109NDB1	IO82NDB1	IO76VDB1	None	None	None
118	IO109PDB1	IO82PDB1	IO76UDB1	None	None	None
119	IO106NDB1	IO80NDB1	IO75NDB1	None	None	None
120	IO106PDB1	IO80PDB1	IO75PDB1	None	None	None
121	IO104PSB1	IO79PSB1	IO74RSB1	None	None	None
122	GND	GND	GND	None	None	None
123	VCCIB1	VCCIB1	VCCIB1	None	None	None
124	IO99NDB1	IO75NDB1	NC	None	**Rule 1**	**Rule 1**
125	IO99PDB1	IO75PDB1	NC	None	**Rule 1**	**Rule 1**
126	NC	NC	VCC	None	**Rule 10**	**Rule 10**
127	IO96NDB1	IO73NDB1	IO72NDB1	None	None	None
128	GCC2/IO96PDB1	GCC2/IO73PDB1	GCC2/IO72PDB1	None	None	None

Table 10-4 • **Pin Compatibility and Migration Table for the PQ208 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
129	GCB2/IO95PSB1	GCB2/IO72PSB1	GCB2/IO71PSB1	None	None	None
130	GND	GND	GND	None	None	None
131	GCA2/IO94PSB1	GCA2/IO71PSB1	GCA2/IO70PSB1	None	None	None
132	GCA1/IO93PDB1	GCA1/IO70PDB1	GCA1/IO69PDB1	None	None	None
133	GCA0/IO93NDB1	GCA0/IO70NDB1	GCA0/IO69NDB1	None	None	None
134	GCB0/IO92NDB1	GCB0/IO69NDB1	GCB0/IO68NDB1	None	None	None
135	GCB1/IO92PDB1	GCB1/IO69PDB1	GCB1/IO68PDB1	None	None	None
136	GCC0/IO91NDB1	GCC0/IO68NDB1	GCC0/IO67NDB1	None	None	None
137	GCC1/IO91PDB1	GCC1/IO68PDB1	GCC1/IO67PDB1	None	None	None
138	IO88NDB1	IO66NDB1	IO66NDB1	None	None	None
139	IO88PDB1	IO66PDB1	IO66PDB1	None	None	None
140	VCCIB1	VCCIB1	VCCIB1	None	None	None
141	GND	GND	GND	None	None	None
142	VCC	VCC	VCC	None	None	None
143	IO86PSB1	IO65PSB1	IO65RSB1	None	None	None
144	IO84NDB1	IO64NDB1	IO64NDB1	None	None	None
145	IO84PDB1	IO64PDB1	IO64PDB1	None	None	None
146	IO82NDB1	IO63NDB1	IO63NDB1	None	None	None
147	IO82PDB1	IO63PDB1	IO63PDB1	None	None	None
148	IO80NDB1	IO62NDB1	IO62NDB1	None	None	None
149	GBC2/IO80PDB1	GBC2/IO62PDB1	GBC2/IO62PDB1	None	None	None
150	IO79NDB1	IO61NDB1	IO61NDB1	None	None	None
151	GBB2/IO79PDB1	GBB2/IO61PDB1	GBB2/IO61PDB1	None	None	None
152	IO78NDB1	IO60NDB1	IO60NDB1	None	None	None
153	GBA2/IO78PDB1	GBA2/IO60PDB1	GBA2/IO60PDB1	None	None	None
154	VMV1	VMV1	VMV1	None	None	None
155	GNDQ	GNDQ	GNDQ	None	None	None
156	GND	GND	GND	None	None	None
157	VMV0	VMV0	VMV0	None	None	None
158	GBA1/IO77RSB0	GBA1/IO59RSB0	GBA1/IO59RSB0	None	None	None
159	GBA0/IO76RSB0	GBA0/IO58RSB0	GBA0/IO58RSB0	None	None	None
160	GBB1/IO75RSB0	GBB1/IO57RSB0	GBB1/IO57RSB0	None	None	None
161	GBB0/IO74RSB0	GBB0/IO56RSB0	GBB0/IO56RSB0	None	None	None

Table 10-4 **• Pin Compatibility and Migration Table for the PQ208 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
162	GND	GND	GND	None	None	None
163	GBC1/IO73RSB0	GBC1/IO55RSB0	GBC1/IO55RSB0	None	None	None
164	GBC0/IO72RSB0	GBC0/IO54RSB0	GBC0/IO54RSB0	None	None	None
165	IO70RSB0	IO52RSB0	IO52RSB0	None	None	None
166	IO67RSB0	IO50RSB0	IO49RSB0	None	None	None
167	IO63RSB0	IO48RSB0	IO46RSB0	None	None	None
168	IO60RSB0	IO46RSB0	IO43RSB0	None	None	None
169	IO57RSB0	IO44RSB0	IO40RSB0	None	None	None
170	VCCIB0	VCCIB0	VCCIB0	None	None	None
171	VCC	VCC	VCC	None	None	None
172	IO54RSB0	IO36RSB0	IO36RSB0	None	None	None
173	IO51RSB0	IO35RSB0	IO35RSB0	None	None	None
174	IO48RSB0	IO34RSB0	IO34RSB0	None	None	None
175	IO45RSB0	IO33RSB0	IO33RSB0	None	None	None
176	IO42RSB0	IO32RSB0	IO32RSB0	None	None	None
177	IO40RSB0	IO31RSB0	IO31RSB0	None	None	None
178	GND	GND	GND	None	None	None
179	IO38RSB0	IO29RSB0	IO29RSB0	None	None	None
180	IO35RSB0	IO28RSB0	IO28RSB0	None	None	None
181	IO33RSB0	IO27RSB0	IO27RSB0	None	None	None
182	IO31RSB0	IO26RSB0	IO26RSB0	None	None	None
183	IO29RSB0	IO25RSB0	IO25RSB0	None	None	None
184	IO27RSB0	IO24RSB0	IO24RSB0	None	None	None
185	IO25RSB0	IO23RSB0	IO23RSB0	None	None	None
186	VCCIB0	VCCIB0	VCCIB0	None	None	None
187	VCC	VCC	VCC	None	None	None
188	IO22RSB0	IO20RSB0	IO21RSB0	None	None	None
189	IO20RSB0	IO19RSB0	IO20RSB0	None	None	None
190	IO18RSB0	IO18RSB0	IO19RSB0	None	None	None
191	IO16RSB0	IO17RSB0	IO18RSB0	None	None	None
192	IO15RSB0	IO16RSB0	IO17RSB0	None	None	None
193	IO14RSB0	IO14RSB0	IO16RSB0	None	None	None
194	IO13RSB0	IO12RSB0	IO15RSB0	None	None	None

Table 10-4 **• Pin Compatibility and Migration Table for the PQ208 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
195	GND	GND	GND	None	None	None
196	IO12RSB0	IO10RSB0	IO13RSB0	None	None	None
197	IO11RSB0	IO09RSB0	IO11RSB0	None	None	None
198	IO10RSB0	IO08RSB0	IO09RSB0	None	None	None
199	IO09RSB0	IO07RSB0	IO07RSB0	None	None	None
200	VCCIB0	VCCIB0	VCCIB0	None	None	None
201	GAC1/IO05RSB0	GAC1/IO05RSB0	GAC1/IO05RSB0	None	None	None
202	GAC0/IO04RSB0	GAC0/IO04RSB0	GAC0/IO04RSB0	None	None	None
203	GAB1/IO03RSB0	GAB1/IO03RSB0	GAB1/IO03RSB0	None	None	None
204	GAB0/IO02RSB0	GAB0/IO02RSB0	GAB0/IO02RSB0	None	None	None
205	GAA1/IO01RSB0	GAA1/IO01RSB0	GAA1/IO01RSB0	None	None	None
206	GAA0/IO00RSB0	GAA0/IO00RSB0	GAA0/IO00RSB0	None	None	None
207	GNDQ	GNDQ	GNDQ	None	None	None
208	VMV0	VMV0	VMV0	None	None	None

FG144 Package

***Table 10-5* • Pin Compatibility and Migration Table for the FG144 Package**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
A1	GNDQ	GNDQ	GNDQ	None	None	None
A2	VMV0	VMV0	VMV0	None	None	None
A3	GAB0/IO02RSB0	GAB0/IO02RSB0	GAB0/IO02RSB0	None	None	None
A4	GAB1/IO03RSB0	GAB1/IO03RSB0	GAB1/IO03RSB0	None	None	None
A5	IO10RSB0	IO10RSB0	IO16RSB0	None	None	None
A6	GND	GND	GND	None	None	None
A7	IO44RSB0	IO44RSB0	IO30RSB0	None	None	None
A8	VCC	VCC	VCC	None	None	None
A9	IO69RSB0	IO69RSB0	IO34RSB0	None	None	None
A10	GBA0/IO76RSB0	GBA0/IO76RSB0	GBA0/IO58RSB0	None	None	None
A11	GBA1/IO77RSB0	GBA1/IO77RSB0	GBA1/IO59RSB0	None	None	None
A12	GNDQ	GNDQ	GNDQ	None	None	None
B1	GAB2/IO224PDB3	GAB2/IO224PDB3	GAB2/IO154UDB3	None	None	None
B2	GND	GND	GND	None	None	None
B3	GAA0/IO00RSB0	GAA0/IO00RSB0	GAA0/IO00RSB0	None	None	None
B4	GAA1/IO01RSB0	GAA1/IO01RSB0	GAA1/IO01RSB0	None	None	None
B5	IO13RSB0	IO13RSB0	IO14RSB0	None	None	None
B6	IO26RSB0	IO26RSB0	IO19RSB0	None	None	None
B7	IO35RSB0	IO35RSB0	IO23RSB0	None	None	None
B8	IO60RSB0	IO60RSB0	IO31RSB0	None	None	None
B9	GBB0/IO74RSB0	GBB0/IO74RSB0	GBB0/IO56RSB0	None	None	None
B10	GBB1/IO75RSB0	GBB1/IO75RSB0	GBB1/IO57RSB0	None	None	None
B11	GND	GND	GND	None	None	None
B12	VMV1	VMV1	VMV1	None	None	None
C1	IO224NDB3	IO224NDB3	IO154VDB3	None	None	None
C2	GFA2/IO206PPB3	GFA2/IO206PPB3	GFA2/IO144PPB3	None	None	None
C3	GAC2/IO223PDB3	GAC2/IO223PDB3	GAC2/IO153UDB3	None	None	None
C4	VCC	VCC	VCC	None	None	None
C5	IO16RSB0	IO16RSB0	IO12RSB0	None	None	None
C6	IO29RSB0	IO29RSB0	IO17RSB0	None	None	None
C7	IO32RSB0	IO32RSB0	IO25RSB0	None	None	None

Table 10-5 • **Pin Compatibility and Migration Table for the FG144 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
C8	IO63RSB0	IO63RSB0	IO32RSB0	None	None	None
C9	IO66RSB0	IO66RSB0	IO53RSB0	None	None	None
C10	GBA2/IO78PDB1	GBA2/IO78PDB1	GBA2/IO60PDB1	None	None	None
C11	IO78NDB1	IO78NDB1	IO60NDB1	None	None	None
C12	GBC2/IO80PPB1	GBC2/IO80PPB1	GBC2/IO62PPB1	None	None	None
D1	IO213PDB3	IO213PDB3	IO149NDB3	None	None	None
D2	IO213NDB3	IO213NDB3	IO149PDB3	None	None	None
D3	IO223NDB3	IO223NDB3	IO153VDB3	None	None	None
D4	GAA2/IO225PPB3	GAA2/IO225PPB3	GAA2/IO155UPB3	None	None	None
D5	GAC0/IO04RSB0	GAC0/IO04RSB0	GAC0/IO04RSB0	None	None	None
D6	GAC1/IO05RSB0	GAC1/IO05RSB0	GAC1/IO05RSB0	None	None	None
D7	GBC0/IO72RSB0	GBC0/IO72RSB0	GBC0/IO54RSB0	None	None	None
D8	GBC1/IO73RSB0	GBC1/IO73RSB0	GBC1/IO55RSB0	None	None	None
D9	GBB2/IO79PDB1	GBB2/IO79PDB1	GBB2/IO61PDB1	None	None	None
D10	IO79NDB1	IO79NDB1	IO61NDB1	None	None	None
D11	IO80NPB1	IO80NPB1	IO62NPB1	None	None	None
D12	GCB1/IO92PPB1	GCB1/IO92PPB1	GCB1/IO68PPB1	None	None	None
E1	VCC	VCC	VCC	None	None	None
E2	GFC0/IO209NDB3	GFC0/IO209NDB3	GFC0/IO147NDB3	None	None	None
E3	GFC1/IO209PDB3	GFC1/IO209PDB3	GFC1/IO147PDB3	None	None	None
E4	VCCIB3	VCCIB3	VCCIB3	None	None	None
E5	IO225NPB3	IO225NPB3	IO155VPB3	None	None	None
E6	VCCIB0	VCCIB0	VCCIB0	None	None	None
E7	VCCIB0	VCCIB0	VCCIB0	None	None	None
E8	GCC1/IO91PDB1	GCC1/IO91PDB1	GCC1/IO67PDB1	None	None	None
E9	VCCIB1	VCCIB1	VCCIB1	None	None	None
E10	VCC	VCC	VCC	None	None	None
E11	GCA0/IO93NDB1	GCA0/IO93NDB1	GCA0/IO69NDB1	None	None	None
E12	IO94NDB1	IO94NDB1	IO70NDB1	None	None	None
F1	GFB0/IO208NPB3	GFB0/IO208NPB3	GFB0/IO146NPB3	None	None	None
F2	VCOMPLF	VCOMPLF	VCOMPLF	None	None	None
F3	GFB1/IO208PPB3	GFB1/IO208PPB3	GFB1/IO146PPB3	None	None	None
F4	IO206NPB3	IO206NPB3	IO144NPB3	None	None	None

Table 10-5 **• Pin Compatibility and Migration Table for the FG144 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
F5	GND	GND	GND	None	None	None
F6	GND	GND	GND	None	None	None
F7	GND	GND	GND	None	None	None
F8	GCC0/IO91NDB1	GCC0/IO91NDB1	GCC0/IO67NDB1	None	None	None
F9	GCB0/IO92NPB1	GCB0/IO92NPB1	GCB0/IO68NPB1	None	None	None
F10	GND	GND	GND	None	None	None
F11	GCA1/IO93PDB1	GCA1/IO93PDB1	GCA1/IO69PDB1	None	None	None
F12	GCA2/IO94PDB1	GCA2/IO94PDB1	GCA2/IO70PDB1	None	None	None
G1	GFA1/IO207PPB3	GFA1/IO207PPB3	GFA1/IO145PPB3	None	None	None
G2	GND	GND	GND	None	None	None
G3	VCCPLF	VCCPLF	VCCPLF	None	None	None
G4	GFA0/IO207NPB3	GFA0/IO207NPB3	GFA0/IO145NPB3	None	None	None
G5	GND	GND	GND	None	None	None
G6	GND	GND	GND	None	None	None
G7	GND	GND	GND	None	None	None
G8	GDC1/IO111PPB1	GDC1/IO111PPB1	GDC1/IO77UPB1	None	None	None
G9	IO96NDB1	IO96NDB1	IO72NDB1	None	None	None
G10	GCC2/IO96PDB1	GCC2/IO96PDB1	GCC2/IO72PDB1	None	None	None
G11	IO95NDB1	IO95NDB1	IO71NDB1	None	None	None
G12	GCB2/IO95PDB1	GCB2/IO95PDB1	GCB2/IO71PDB1	None	None	None
H1	VCC	VCC	VCC	None	None	None
H2	GFB2/IO205PDB3	GFB2/IO205PDB3	GFB2/IO143PDB3	None	None	None
H3	GFC2/IO204PSB3	GFC2/IO204PSB3	GFC2/IO142PSB3	None	None	None
H4	GEC1/IO190PDB3	GEC1/IO190PDB3	GEC1/IO137PDB3	None	None	None
H5	VCC	VCC	VCC	None	None	None
H6	IO105PDB1	IO105PDB1	IO75PDB1	None	None	None
H7	IO105NDB1	IO105NDB1	IO75NDB1	None	None	None
H8	GDB2/IO115RSB2	GDB2/IO115RSB2	GDB2/IO81RSB2	None	None	None
H9	GDC0/IO111NPB1	GDC0/IO111NPB1	GDC0/IO77VPB1	None	None	None
H10	VCCIB1	VCCIB1	VCCIB1	None	None	None
H11	IO101PSB1	IO101PSB1	IO73PSB1	None	None	None
H12	VCC	VCC	VCC	None	None	None
J1	GEB1/IO189PDB3	GEB1/IO189PDB3	GEB1/IO136PDB3	None	None	None

Table 10-5 • **Pin Compatibility and Migration Table for the FG144 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
J2	IO205NDB3	IO205NDB3	IO143NDB3	None	None	None
J3	VCCIB3	VCCIB3	VCCIB3	None	None	None
J4	GEC0/IO190NDB3	GEC0/IO190NDB3	GEC0/IO137NDB3	None	None	None
J5	IO160RSB2	IO160RSB2	IO125RSB2	None	None	None
J6	IO157RSB2	IO157RSB2	IO116RSB2	None	None	None
J7	VCC	VCC	VCC	None	None	None
J8	TCK	TCK	TCK	None	None	None
J9	GDA2/IO114RSB2	GDA2/IO114RSB2	GDA2/IO80RSB2	None	None	None
J10	TDO	TDO	TDO	None	None	None
J11	GDA1/IO113PDB1	GDA1/IO113PDB1	GDA1/IO79UDB1	None	None	None
J12	GDB1/IO112PDB1	GDB1/IO112PDB1	GDB1/IO78UDB1	None	None	None
K1	GEB0/IO189NDB3	GEB0/IO189NDB3	GEB0/IO136NDB3	None	None	None
K2	GEA1/IO188PDB3	GEA1/IO188PDB3	GEA1/IO135PDB3	None	None	None
K3	GEA0/IO188NDB3	GEA0/IO188NDB3	GEA0/IO135NDB3	None	None	None
K4	GEA2/IO187RSB2	GEA2/IO187RSB2	GEA2/IO134RSB2	None	None	None
K5	IO169RSB2	IO169RSB2	IO127RSB2	None	None	None
K6	IO152RSB2	IO152RSB2	IO121RSB2	None	None	None
K7	GND	GND	GND	None	None	None
K8	IO117RSB2	IO117RSB2	IO104RSB2	None	None	None
K9	GDC2/IO116RSB2	GDC2/IO116RSB2	GDC2/IO82RSB2	None	None	None
K10	GND	GND	GND	None	None	None
K11	GDA0/IO113NDB1	GDA0/IO113NDB1	GDA0/IO79VDB1	None	None	None
K12	GDB0/IO112NDB1	GDB0/IO112NDB1	GDB0/IO78VDB1	None	None	None
L1	GND	GND	GND	None	None	None
L2	VMV3	VMV3	VMV3	None	None	None
L3	GEB2/IO186RSB2	GEB2/IO186RSB2	GEB2/IO133RSB2	None	None	None
L4	IO172RSB2	IO172RSB2	IO128RSB2	None	None	None
L5	VCCIB2	VCCIB2	VCCIB2	None	None	None
L6	IO153RSB2	IO153RSB2	IO119RSB2	None	None	None
L7	IO144RSB2	IO144RSB2	IO114RSB2	None	None	None
L8	IO140RSB2	IO140RSB2	IO110RSB2	None	None	None
L9	TMS	TMS	TMS	None	None	None
L10	VJTAG	VJTAG	VJTAG	None	None	None

Table 10-5 **• Pin Compatibility and Migration Table for the FG144 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
L11	VMV2	VMV2	VMV2	None	None	None
L12	TRST	TRST	TRST	None	None	None
M1	GNDQ	GNDQ	GNDQ	None	None	None
M2	GEC2/IO185RSB2	GEC2/IO185RSB2	GEC2/IO132RSB2	None	None	None
M3	IO173RSB2	IO173RSB2	IO129RSB2	None	None	None
M4	IO168RSB2	IO168RSB2	IO126RSB2	None	None	None
M5	IO161RSB2	IO161RSB2	IO124RSB2	None	None	None
M6	IO156RSB2	IO156RSB2	IO122RSB2	None	None	None
M7	IO145RSB2	IO145RSB2	IO117RSB2	None	None	None
M8	IO141RSB2	IO141RSB2	IO115RSB2	None	None	None
M9	TDI	TDI	TDI	None	None	None
M10	VCCIB2	VCCIB2	VCCIB2	None	None	None
M11	VPUMP	VPUMP	VPUMP	None	None	None
M12	GNDQ	GNDQ	GNDQ	None	None	None

FG256 Package

Table 10-6 • **Pin Compatibility and Migration Table for the FG256 Package**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
A1	GND	GND	GND	None	None	None
A2	GAA0/IO00RSB0	GAA0/IO00RSB0	GAA0/IO00RSB0	None	None	None
A3	GAA1/IO01RSB0	GAA1/IO01RSB0	GAA1/IO01RSB0	None	None	None
A4	GAB0/IO02RSB0	GAB0/IO02RSB0	GAB0/IO02RSB0	None	None	None
A5	IO16RSB0	IO11RSB0	IO16RSB0	None	None	None
A6	IO22RSB0	IO16RSB0	IO17RSB0	None	None	None
A7	IO28RSB0	IO18RSB0	IO22RSB0	None	None	None
A8	IO35RSB0	IO28RSB0	IO28RSB0	None	None	None
A9	IO45RSB0	IO34RSB0	IO34RSB0	None	None	None
A10	IO50RSB0	IO37RSB0	IO37RSB0	None	None	None
A11	IO55RSB0	IO41RSB0	IO41RSB0	None	None	None
A12	IO61RSB0	IO43RSB0	IO43RSB0	None	None	None
A13	GBB1/IO75RSB0	GBB1/IO57RSB0	GBB1/IO57RSB0	None	None	None
A14	GBA0/IO76RSB0	GBA0/IO58RSB0	GBA0/IO58RSB0	None	None	None
A15	GBA1/IO77RSB0	GBA1/IO59RSB0	GBA1/IO59RSB0	None	None	None
A16	GND	GND	GND	None	None	None
B1	GAB2/IO224PDB3	GAB2/IO173PDB3	GAB2/IO154UDB3	None	None	None
B2	GAA2/IO225PDB3	GAA2/IO174PDB3	GAA2/IO155UDB3	None	None	None
B3	GNDQ	GNDQ	IO12RSB0	None	**Rule 8**	**Rule 8**
B4	GAB1/IO03RSB0	GAB1/IO03RSB0	GAB1/IO03RSB0	None	None	None
B5	IO17RSB0	IO13RSB0	IO13RSB0	None	None	None
B6	IO21RSB0	IO14RSB0	IO14RSB0	None	None	None
B7	IO27RSB0	IO21RSB0	IO21RSB0	None	None	None
B8	IO34RSB0	IO27RSB0	IO27RSB0	None	None	None
B9	IO44RSB0	IO32RSB0	IO32RSB0	None	None	None
B10	IO51RSB0	IO38RSB0	IO38RSB0	None	None	None
B11	IO57RSB0	IO42RSB0	IO42RSB0	None	None	None
B12	GBC1/IO73RSB0	GBC1/IO55RSB0	GBC1/IO55RSB0	None	None	None
B13	GBB0/IO74RSB0	GBB0/IO56RSB0	GBB0/IO56RSB0	None	None	None
B14	IO71RSB0	IO52RSB0	IO44RSB0	None	None	None
B15	GBA2/IO78PDB1	GBA2/IO60PDB1	GBA2/IO60PDB1	None	None	None

***Table 10-6* • Pin Compatibility and Migration Table for the FG256 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
B16	IO81PDB1	IO60NDB1	IO60NDB1	None	None	None
C1	IO224NDB3	IO173NDB3	IO154VDB3	None	None	None
C2	IO225NDB3	IO174NDB3	IO155VDB3	None	None	None
C3	VMV3	VMV3	IO11RSB0	None	**Rule 7**	**Rule 7**
C4	IO11RSB0	IO07RSB0	IO07RSB0	None	None	None
C5	GAC0/IO04RSB0	GAC0/IO04RSB0	GAC0/IO04RSB0	None	None	None
C6	GAC1/IO05RSB0	GAC1/IO05RSB0	GAC1/IO05RSB0	None	None	None
C7	IO25RSB0	IO20RSB0	IO20RSB0	None	None	None
C8	IO36RSB0	IO24RSB0	IO24RSB0	None	None	None
C9	IO42RSB0	IO33RSB0	IO33RSB0	None	None	None
C10	IO49RSB0	IO39RSB0	IO39RSB0	None	None	None
C11	IO56RSB0	IO44RSB0	IO45RSB0	None	None	None
C12	GBC0/IO72RSB0	GBC0/IO54RSB0	GBC0/IO54RSB0	None	None	None
C13	IO62RSB0	IO51RSB0	IO48RSB0	None	None	None
C14	VMV0	VMV0	VMV0	None	None	None
C15	IO78NDB1	IO61NPB1	IO61NPB1	None	None	None
C16	IO81NDB1	IO63PDB1	IO63PDB1	None	None	None
D1	IO222NDB3	IO171NDB3	IO151VDB3	None	None	None
D2	IO222PDB3	IO171PDB3	IO151UDB3	None	None	None
D3	GAC2/IO223PDB3	GAC2/IO172PDB3	GAC2/IO153UDB3	None	None	None
D4	IO223NDB3	IO06RSB0	IO06RSB0	None	None	None
D5	GNDQ	GNDQ	GNDQ	None	None	None
D6	IO23RSB0	IO10RSB0	IO10RSB0	None	None	None
D7	IO29RSB0	IO19RSB0	IO19RSB0	None	None	None
D8	IO33RSB0	IO26RSB0	IO26RSB0	None	None	None
D9	IO46RSB0	IO30RSB0	IO30RSB0	None	None	None
D10	IO52RSB0	IO40RSB0	IO40RSB0	None	None	None
D11	IO60RSB0	IO45RSB0	IO46RSB0	None	None	None
D12	GNDQ	GNDQ	GNDQ	None	None	None
D13	IO80NDB1	IO50RSB0	IO47RSB0	None	None	None
D14	GBB2/IO79PDB1	GBB2/IO61PPB1	GBB2/IO61PPB1	None	None	None
D15	IO79NDB1	IO53RSB0	IO53RSB0	None	None	None
D16	IO82NSB1	IO63NDB1	IO63NDB1	None	None	None

Table 10-6 • **Pin Compatibility and Migration Table for the FG256 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
E1	IO217PDB3	IO166PDB3	IO150PDB3	None	None	None
E2	IO218PDB3	IO167NPB3	IO08RSB0	None	None	None
E3	IO221NDB3	IO172NDB3	IO153VDB3	None	None	None
E4	IO221PDB3	IO169NDB3	IO152VDB3	None	None	None
E5	VMV0	VMV0	VMV0	None	None	None
E6	VCCIB0	VCCIB0	VCCIB0	None	None	None
E7	VCCIB0	VCCIB0	VCCIB0	None	None	None
E8	IO38RSB0	IO25RSB0	IO25RSB0	None	None	None
E9	IO47RSB0	IO31RSB0	IO31RSB0	None	None	None
E10	VCCIB0	VCCIB0	VCCIB0	None	None	None
E11	VCCIB0	VCCIB0	VCCIB0	None	None	None
E12	VMV1	VMV1	VMV1	None	None	None
E13	GBC2/IO80PDB1	GBC2/IO62PDB1	GBC2/IO62PDB1	None	None	None
E14	IO83PPB1	IO67PPB1	IO65RSB1	None	None	None
E15	IO86PPB1	IO64PPB1	IO52RSB0	None	None	None
E16	IO87PDB1	IO66PDB1	IO66PDB1	None	None	None
F1	IO217NDB3	IO166NDB3	IO150NDB3	None	None	None
F2	IO218NDB3	IO168NPB3	IO149NPB3	None	None	None
F3	IO216PDB3	IO167PPB3	IO09RSB0	None	None	None
F4	IO216NDB3	IO169PDB3	IO152UDB3	None	None	None
F5	VCCIB3	VCCIB3	VCCIB3	None	None	None
F6	GND	GND	GND	None	None	None
F7	VCC	VCC	VCC	None	None	None
F8	VCC	VCC	VCC	None	None	None
F9	VCC	VCC	VCC	None	None	None
F10	VCC	VCC	VCC	None	None	None
F11	GND	GND	GND	None	None	None
F12	VCCIB1	VCCIB1	VCCIB1	None	None	None
F13	IO83NPB1	IO62NDB1	IO62NDB1	None	None	None
F14	IO86NPB1	IO64NPB1	IO49RSB0	None	None	None
F15	IO90PPB1	IO65PPB1	IO64PPB1	None	None	None
F16	IO87NDB1	IO66NDB1	IO66NDB1	None	None	None
G1	IO210PSB3	IO165NDB3	IO148NDB3	None	None	None

Table 10-6 • **Pin Compatibility and Migration Table for the FG256 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
G2	IO213NDB3	IO165PDB3	IO148PDB3	None	None	None
G3	IO213PDB3	IO168PPB3	IO149PPB3	None	None	None
G4	GFC1/IO209PPB3	GFC1/IO164PPB3	GFC1/IO147PPB3	None	None	None
G5	VCCIB3	VCCIB3	VCCIB3	None	None	None
G6	VCC	VCC	VCC	None	None	None
G7	GND	GND	GND	None	None	None
G8	GND	GND	GND	None	None	None
G9	GND	GND	GND	None	None	None
G10	GND	GND	GND	None	None	None
G11	VCC	VCC	VCC	None	None	None
G12	VCCIB1	VCCIB1	VCCIB1	None	None	None
G13	GCC1/IO91PPB1	GCC1/IO69PPB1	GCC1/IO67PPB1	None	None	None
G14	IO90NPB1	IO65NPB1	IO64NPB1	None	None	None
G15	IO88PDB1	IO75PDB1	IO73PDB1	None	None	None
G16	IO88NDB1	IO75NDB1	IO73NDB1	None	None	None
H1	GFB0/IO208NPB3	GFB0/IO163NPB3	GFB0/IO146NPB3	None	None	None
H2	GFA0/IO207NDB3	GFA0/IO162NDB3	GFA0/IO145NDB3	None	None	None
H3	GFB1/IO208PPB3	GFB1/IO163PPB3	GFB1/IO146PPB3	None	None	None
H4	VCOMPLF	VCOMPLF	VCOMPLF	None	None	None
H5	GFC0/IO209NPB3	GFC0/IO164NPB3	GFC0/IO147NPB3	None	None	None
H6	VCC	VCC	VCC	None	None	None
H7	GND	GND	GND	None	None	None
H8	GND	GND	GND	None	None	None
H9	GND	GND	GND	None	None	None
H10	GND	GND	GND	None	None	None
H11	VCC	VCC	VCC	None	None	None
H12	GCC0/IO91NPB1	GCC0/IO69NPB1	GCC0/IO67NPB1	None	None	None
H13	GCB1/IO92PPB1	GCB1/IO70PPB1	GCB1/IO68PPB1	None	None	None
H14	GCA0/IO93NPB1	GCA0/IO71NPB1	GCA0/IO69NPB1	None	None	None
H15	IO96NPB1	IO67NPB1	NC	None	**Rule 1**	**Rule 1**
H16	GCB0/IO92NPB1	GCB0/IO70NPB1	GCB0/IO68NPB1	None	None	None
J1	GFA2/IO206PSB3	GFA2/IO161PPB3	GFA2/IO144PPB3	None	None	None
J2	GFA1/IO207PDB3	GFA1/IO162PDB3	GFA1/IO145PDB3	None	None	None

***Table 10-6* • Pin Compatibility and Migration Table for the FG256 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
J3	VCCPLF	VCCPLF	VCCPLF	None	None	None
J4	IO205NDB3	IO160NDB3	IO143NDB3	None	None	None
J5	GFB2/IO205PDB3	GFB2/IO160PDB3	GFB2/IO143PDB3	None	None	None
J6	VCC	VCC	VCC	None	None	None
J7	GND	GND	GND	None	None	None
J8	GND	GND	GND	None	None	None
J9	GND	GND	GND	None	None	None
J10	GND	GND	GND	None	None	None
J11	VCC	VCC	VCC	None	None	None
J12	GCB2/IO95PPB1	GCB2/IO73PPB1	GCB2/IO71PPB1	None	None	None
J13	GCA1/IO93PPB1	GCA1/IO71PPB1	GCA1/IO69PPB1	None	None	None
J14	GCC2/IO96PPB1	GCC2/IO74PPB1	GCC2/IO72PPB1	None	None	None
J15	IO100PPB1	IO80PPB1	NC	None	**Rule 1**	**Rule 1**
J16	GCA2/IO94PSB1	GCA2/IO72PDB1	GCA2/IO70PDB1	None	None	None
K1	GFC2/IO204PDB3	GFC2/IO159PDB3	GFC2/IO142PDB3	None	None	None
K2	IO204NDB3	IO161NPB3	IO144NPB3	None	None	None
K3	IO203NDB3	IO156PPB3	IO141PPB3	None	None	None
K4	IO203PDB3	IO129RSB2	IO120RSB2	None	None	None
K5	VCCIB3	VCCIB3	VCCIB3	None	None	None
K6	VCC	VCC	VCC	None	None	None
K7	GND	GND	GND	None	None	None
K8	GND	GND	GND	None	None	None
K9	GND	GND	GND	None	None	None
K10	GND	GND	GND	None	None	None
K11	VCC	VCC	VCC	None	None	None
K12	VCCIB1	VCCIB1	VCCIB1	None	None	None
K13	IO95NPB1	IO73NPB1	IO71NPB1	None	None	None
K14	IO100NPB1	IO80NPB1	IO74RSB1	None	None	None
K15	IO102NDB1	IO74NPB1	IO72NPB1	None	None	None
K16	IO102PDB1	IO72NDB1	IO70NDB1	None	None	None
L1	IO202NDB3	IO159NDB3	IO142NDB3	None	None	None
L2	IO202PDB3	IO156NPB3	IO141NPB3	None	None	None
L3	IO196PPB3	IO151PPB3	IO125RSB2	None	None	None

Table 10-6 • Pin Compatibility and Migration Table for the FG256 Package (continued)

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
L4	IO193PPB3	IO158PSB3	IO139RSB3	None	None	None
L5	VCCIB3	VCCIB3	VCCIB3	None	None	None
L6	GND	GND	GND	None	None	None
L7	VCC	VCC	VCC	None	None	None
L8	VCC	VCC	VCC	None	None	None
L9	VCC	VCC	VCC	None	None	None
L10	VCC	VCC	VCC	None	None	None
L11	GND	GND	GND	None	None	None
L12	VCCIB1	VCCIB1	VCCIB1	None	None	None
L13	GDB0/IO112NPB1	GDB0/IO87NPB1	GDB0/IO78VPB1	None	None	None
L14	IO106NDB1	IO85NDB1	IO76VDB1	None	None	None
L15	IO106PDB1	IO85PDB1	IO76UDB1	None	None	None
L16	IO107PDB1	IO84PDB1	IO75PDB1	None	None	None
M1	IO197NSB3	IO150PDB3	IO140PDB3	None	None	None
M2	IO196NPB3	IO151NPB3	IO130RSB2	None	None	None
M3	IO193NPB3	IO147NPB3	IO138NPB3	None	None	None
M4	GEC0/IO190NPB3	GEC0/IO146NPB3	GEC0/IO137NPB3	None	None	None
M5	VMV3	VMV3	VMV3	None	None	None
M6	VCCIB2	VCCIB2	VCCIB2	None	None	None
M7	VCCIB2	VCCIB2	VCCIB2	None	None	None
M8	IO147RSB2	IO117RSB2	IO108RSB2	None	None	None
M9	IO136RSB2	IO110RSB2	IO101RSB2	None	None	None
M10	VCCIB2	VCCIB2	VCCIB2	None	None	None
M11	VCCIB2	VCCIB2	VCCIB2	None	None	None
M12	VMV2	VMV2	VMV2	None	None	None
M13	IO110NDB1	IO94RSB2	IO83RSB2	None	None	None
M14	GDB1/IO112PPB1	GDB1/IO87PPB1	GDB1/IO78UPB1	None	None	None
M15	GDC1/IO111PDB1	GDC1/IO86PDB1	GDC1/IO77UDB1	None	None	None
M16	IO107NDB1	IO84NDB1	IO75NDB1	None	None	None
N1	IO194PSB3	IO150NDB3	IO140NDB3	None	None	None
N2	IO192PPB3	IO147PPB3	IO138PPB3	None	None	None
N3	GEC1/IO190PPB3	GEC1/IO146PPB3	GEC1/IO137PPB3	None	None	None
N4	IO192NPB3	IO140RSB2	IO131RSB2	None	None	None

***Table 10-6* • Pin Compatibility and Migration Table for the FG256 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
N5	GNDQ	GNDQ	GNDQ	None	None	None
N6	GEA2/IO187RSB2	GEA2/IO143RSB2	GEA2/IO134RSB2	None	None	None
N7	IO161RSB2	IO126RSB2	IO117RSB2	None	None	None
N8	IO155RSB2	IO120RSB2	IO111RSB2	None	None	None
N9	IO141RSB2	IO108RSB2	IO99RSB2	None	None	None
N10	IO129RSB2	IO103RSB2	IO94RSB2	None	None	None
N11	IO124RSB2	IO99RSB2	IO87RSB2	None	None	None
N12	GNDQ	GNDQ	GNDQ	None	None	None
N13	IO110PDB1	IO92RSB2	IO93RSB2	None	None	None
N14	VJTAG	VJTAG	VJTAG	None	None	None
N15	GDC0/IO111NDB1	GDC0/IO86NDB1	GDC0/IO77VDB1	None	None	None
N16	GDA1/IO113PDB1	GDA1/IO88PDB1	GDA1/IO79UDB1	None	None	None
P1	GEB1/IO189PDB3	GEB1/IO145PDB3	GEB1/IO136PDB3	None	None	None
P2	GEB0/IO189NDB3	GEB0/IO145NDB3	GEB0/IO136NDB3	None	None	None
P3	VMV2	VMV2	VMV2	None	None	None
P4	IO179RSB2	IO138RSB2	IO129RSB2	None	None	None
P5	IO171RSB2	IO136RSB2	IO128RSB2	None	None	None
P6	IO165RSB2	IO131RSB2	IO122RSB2	None	None	None
P7	IO159RSB2	IO124RSB2	IO115RSB2	None	None	None
P8	IO151RSB2	IO119RSB2	IO110RSB2	None	None	None
P9	IO137RSB2	IO107RSB2	IO98RSB2	None	None	None
P10	IO134RSB2	IO104RSB2	IO95RSB2	None	None	None
P11	IO128RSB2	IO97RSB2	IO88RSB2	None	None	None
P12	VMV1	VMV1	IO84RSB2	None	**Rule 7**	**Rule 7**
P13	TCK	TCK	TCK	None	None	None
P14	VPUMP	VPUMP	VPUMP	None	None	None
P15	TRST	TRST	TRST	None	None	None
P16	GDA0/IO113NDB1	GDA0/IO88NDB1	GDA0/IO79VDB1	None	None	None
R1	GEA1/IO188PDB3	GEA1/IO144PDB3	GEA1/IO135PDB3	None	None	None
R2	GEA0/IO188NDB3	GEA0/IO144NDB3	GEA0/IO135NDB3	None	None	None
R3	IO184RSB2	IO139RSB2	IO127RSB2	None	None	None
R4	GEC2/IO185RSB2	GEC2/IO141RSB2	GEC2/IO132RSB2	None	None	None
R5	IO168RSB2	IO132RSB2	IO123RSB2	None	None	None

Table 10-6 • **Pin Compatibility and Migration Table for the FG256 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
R6	IO163RSB2	IO127RSB2	IO118RSB2	None	None	None
R7	IO157RSB2	IO121RSB2	IO112RSB2	None	None	None
R8	IO149RSB2	IO114RSB2	IO106RSB2	None	None	None
R9	IO143RSB2	IO109RSB2	IO100RSB2	None	None	None
R10	IO138RSB2	IO105RSB2	IO96RSB2	None	None	None
R11	IO131RSB2	IO98RSB2	IO89RSB2	None	None	None
R12	IO125RSB2	IO96RSB2	IO85RSB2	None	None	None
R13	GDB2/IO115RSB2	GDB2/IO90RSB2	GDB2/IO81RSB2	None	None	None
R14	TDI	TDI	TDI	None	None	None
R15	GNDQ	GNDQ	NC	None	**Rule 8**	**Rule 8**
R16	TDO	TDO	TDO	None	None	None
T1	GND	GND	GND	None	None	None
T2	IO183RSB2	IO137RSB2	IO126RSB2	None	None	None
T3	GEB2/IO186RSB2	GEB2/IO142RSB2	GEB2/IO133RSB2	None	None	None
T4	IO172RSB2	IO134RSB2	IO124RSB2	None	None	None
T5	IO170RSB2	IO125RSB2	IO116RSB2	None	None	None
T6	IO164RSB2	IO123RSB2	IO113RSB2	None	None	None
T7	IO158RSB2	IO118RSB2	IO107RSB2	None	None	None
T8	IO153RSB2	IO115RSB2	IO105RSB2	None	None	None
T9	IO142RSB2	IO111RSB2	IO102RSB2	None	None	None
T10	IO135RSB2	IO106RSB2	IO97RSB2	None	None	None
T11	IO130RSB2	IO102RSB2	IO92RSB2	None	None	None
T12	GDC2/IO116RSB2	GDC2/IO91RSB2	GDC2/IO82RSB2	None	None	None
T13	IO120RSB2	IO93RSB2	IO86RSB2	None	None	None
T14	GDA2/IO114RSB2	GDA2/IO89RSB2	GDA2/IO80RSB2	None	None	None
T15	TMS	TMS	TMS	None	None	None
T16	GND	GND	GND	None	None	None

FG484 Package

Table 10-7 • **Pin Compatibility and Migration Table for the FG484 Package**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
A1	GND	GND	GND	None	None	None
A2	GND	GND	GND	None	None	None
A3	V_{CCI}B0	V_{CCI}B0	V_{CCI}B0	None	None	None
A4	IO07RSB0	NC	NC	**Rule 1**	**Rule 1**	None
A5	IO09RSB0	NC	NC	**Rule 1**	**Rule 1**	None
A6	IO13RSB0	IO09RSB0	IO15RSB0	None	None	None
A7	IO18RSB0	IO15RSB0	IO18RSB0	None	None	None
A8	IO20RSB0	NC	NC	**Rule 1**	**Rule 1**	None
A9	IO26RSB0	NC	NC	**Rule 1**	**Rule 1**	None
A10	IO32RSB0	IO22RSB0	IO23RSB0	None	None	None
A11	IO40RSB0	IO23RSB0	IO29RSB0	None	None	None
A12	IO41RSB0	IO29RSB0	IO35RSB0	None	None	None
A13	IO53RSB0	IO35RSB0	IO36RSB0	None	None	None
A14	IO59RSB0	NC	NC	**Rule 1**	**Rule 1**	None
A15	IO64RSB0	NC	NC	**Rule 1**	**Rule 1**	None
A16	IO65RSB0	IO46RSB0	IO50RSB0	None	None	None
A17	IO67RSB0	IO48RSB0	IO51RSB0	None	None	None
A18	IO69RSB0	NC	NC	**Rule 1**	**Rule 1**	None
A19	NC	NC	NC	None	None	None
A20	VCCIB0	VCCIB0	VCCIB0	None	None	None
A21	GND	GND	GND	None	None	None
A22	GND	GND	GND	None	None	None
AA1	GND	GND	GND	None	None	None
AA2	VCCIB3	VCCIB3	VCCIB3	None	None	None
AA3	NC	NC	NC	None	None	None
AA4	IO181RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA5	IO178RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA6	IO175RSB2	IO135RSB2	NC	None	**Rule 1**	**Rule 1**
AA7	IO169RSB2	IO133RSB2	NC	None	**Rule 1**	**Rule 1**
AA8	IO166RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA9	IO160RSB2	NC	NC	**Rule 1**	**Rule 1**	None

***Table 10-7* • Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
AA10	IO152RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA11	IO146RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA12	IO139RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA13	IO133RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA14	NC	NC	NC	None	None	None
AA15	NC	NC	NC	None	None	None
AA16	IO122RSB2	IO101RSB2	NC	None	**Rule 1**	**Rule 1**
AA17	IO119RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA18	IO117RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AA19	NC	NC	NC	None	None	None
AA20	NC	NC	NC	None	None	None
AA21	VCCIB1	VCCIB1	VCCIB1	None	None	None
AA22	GND	GND	GND	None	None	None
AB1	GND	GND	GND	None	None	None
AB2	GND	GND	GND	None	None	None
AB3	VCCIB2	VCCIB2	VCCIB2	None	None	None
AB4	IO180RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AB5	IO176RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AB6	IO173RSB2	IO130RSB2	IO121RSB2	None	None	None
AB7	IO167RSB2	IO128RSB2	IO119RSB2	None	None	None
AB8	IO162RSB2	IO122RSB2	IO114RSB2	None	None	None
AB9	IO156RSB2	IO116RSB2	IO109RSB2	None	None	None
AB10	IO150RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AB11	IO145RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AB12	IO144RSB2	IO113RSB2	IO104RSB2	None	None	None
AB13	IO132RSB2	IO112RSB2	IO103RSB2	None	None	None
AB14	IO127RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AB15	IO126RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AB16	IO123RSB2	IO100RSB2	IO91RSB2	None	None	None
AB17	IO121RSB2	IO95RSB2	IO90RSB2	None	None	None
AB18	IO118RSB2	NC	NC	**Rule 1**	**Rule 1**	None
AB19	NC	NC	NC	None	None	None
AB20	VCCIB2	VCCIB2	VCCIB2	None	None	None

Table 10-7 **• Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
AB21	GND	GND	GND	None	None	None
AB22	GND	GND	GND	None	None	None
B1	GND	GND	GND	None	None	None
B2	VCCIB3	VCCIB3	VCCIB3	None	None	None
B3	NC	NC	NC	None	None	None
B4	IO06RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B5	IO08RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B6	IO12RSB0	IO08RSB0	NC	None	**Rule 1**	**Rule 1**
B7	IO15RSB0	IO12RSB0	NC	None	**Rule 1**	**Rule 1**
B8	IO19RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B9	IO24RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B10	IO31RSB0	IO17RSB0	NC	None	**Rule 1**	**Rule 1**
B11	IO39RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B12	IO48RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B13	IO54RSB0	IO36RSB0	NC	None	**Rule 1**	**Rule 1**
B14	IO58RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B15	IO63RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B16	IO66RSB0	IO47RSB0	NC	None	**Rule 1**	**Rule 1**
B17	IO68RSB0	IO49RSB0	NC	None	**Rule 1**	**Rule 1**
B18	IO70RSB0	NC	NC	**Rule 1**	**Rule 1**	None
B19	NC	NC	NC	None	None	None
B20	NC	NC	NC	None	None	None
B21	VCCIB1	VCCIB1	VCCIB1	None	None	None
B22	GND	GND	GND	None	None	None
C1	VCCIB3	VCCIB3	VCCIB3	None	None	None
C2	IO220PDB3	NC	NC	**Rule 1**	**Rule 1**	None
C3	NC	NC	NC	None	None	None
C4	NC	NC	NC	None	None	None
C5	GND	GND	GND	None	None	None
C6	IO10RSB0	NC	NC	**Rule 1**	**Rule 1**	None
C7	IO14RSB0	NC	NC	**Rule 1**	**Rule 1**	None
C8	VCC	VCC	VCC	None	None	None
C9	VCC	VCC	VCC	None	None	None

***Table 10-7* • Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
C10	IO30RSB0	NC	NC	**Rule 1**	**Rule 1**	None
C11	IO37RSB0	NC	NC	**Rule 1**	**Rule 1**	None
C12	IO43RSB0	NC	NC	**Rule 1**	**Rule 1**	None
C13	NC	NC	NC	None	None	None
C14	VCC	VCC	VCC	None	None	None
C15	VCC	VCC	VCC	None	None	None
C16	NC	NC	NC	None	None	None
C17	NC	NC	NC	None	None	None
C18	GND	GND	GND	None	None	None
C19	NC	NC	NC	None	None	None
C20	NC	NC	NC	None	None	None
C21	NC	NC	NC	None	None	None
C22	VCCIB1	VCCIB1	VCCIB1	None	None	None
D1	IO219PDB3	NC	NC	**Rule 1**	**Rule 1**	None
D2	IO220NDB3	NC	NC	**Rule 1**	**Rule 1**	None
D3	NC	NC	NC	None	None	None
D4	GND	GND	GND	None	None	None
D5	GAA0/IO00RSB0	GAA0/IO00RSB0	GAA0/IO00RSB0	None	None	None
D6	GAA1/IO01RSB0	GAA1/IO01RSB0	GAA1/IO01RSB0	None	None	None
D7	GAB0/IO02RSB0	GAB0/IO02RSB0	GAB0/IO02RSB0	None	None	None
D8	IO16RSB0	IO11RSB0	IO16RSB0	None	None	None
D9	IO22RSB0	IO16RSB0	IO17RSB0	None	None	None
D10	IO28RSB0	IO18RSB0	IO22RSB0	None	None	None
D11	IO35RSB0	IO28RSB0	IO28RSB0	None	None	None
D12	IO45RSB0	IO34RSB0	IO34RSB0	None	None	None
D13	IO50RSB0	IO37RSB0	IO37RSB0	None	None	None
D14	IO55RSB0	IO41RSB0	IO41RSB0	None	None	None
D15	IO61RSB0	IO43RSB0	IO43RSB0	None	None	None
D16	GBB1/IO75RSB0	GBB1/IO57RSB0	GBB1/IO57RSB0	None	None	None
D17	GBA0/IO76RSB0	GBA0/IO58RSB0	GBA0/IO58RSB0	None	None	None
D18	GBA1/IO77RSB0	GBA1/IO59RSB0	GBA1/IO59RSB0	None	None	None
D19	GND	GND	GND	None	None	None
D20	NC	NC	NC	None	None	None

***Table 10-7* • Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
D21	NC	NC	NC	None	None	None
D22	NC	NC	NC	None	None	None
E1	IO219NDB3	NC	NC	**Rule 1**	**Rule 1**	None
E2	NC	NC	NC	None	None	None
E3	GND	GND	GND	None	None	None
E4	GAB2/IO224PDB3	GAB2/IO173PDB3	GAB2/IO154UDB3	None	None	None
E5	GAA2/IO225PDB3	GAA2/IO174PDB3	GAA2/IO155UDB3	None	None	None
E6	GNDQ	GNDQ	IO12RSB0	None	**Rule 8**	**Rule 8**
E7	GAB1/IO03RSB0	GAB1/IO03RSB0	GAB1/IO03RSB0	None	None	None
E8	IO17RSB0	IO13RSB0	IO13RSB0	None	None	None
E9	IO21RSB0	IO14RSB0	IO14RSB0	None	None	None
E10	IO27RSB0	IO21RSB0	IO21RSB0	None	None	None
E11	IO34RSB0	IO27RSB0	IO27RSB0	None	None	None
E12	IO44RSB0	IO32RSB0	IO32RSB0	None	None	None
E13	IO51RSB0	IO38RSB0	IO38RSB0	None	None	None
E14	IO57RSB0	IO42RSB0	IO42RSB0	None	None	None
E15	GBC1/IO73RSB0	GBC1/IO55RSB0	GBC1/IO55RSB0	None	None	None
E16	GBB0/IO74RSB0	GBB0/IO56RSB0	GBB0/IO56RSB0	None	None	None
E17	IO71RSB0	IO52RSB0	IO44RSB0	None	None	None
E18	GBA2/IO78PDB1	GBA2/IO60PDB1	GBA2/IO60PDB1	None	None	None
E19	IO81PDB1	IO60NDB1	IO60NDB1	None	None	None
E20	GND	GND	GND	None	None	None
E21	NC	NC	NC	None	None	None
E22	IO84PDB1	NC	NC	**Rule 1**	**Rule 1**	None
F1	NC	NC	NC	None	None	None
F2	IO215PDB3	NC	NC	**Rule 1**	**Rule 1**	None
F3	IO215NDB3	NC	NC	**Rule 1**	**Rule 1**	None
F4	IO224NDB3	IO173NDB3	IO154VDB3	None	None	None
F5	IO225NDB3	IO174NDB3	IO155VDB3	None	None	None
F6	VMV3	VMV3	IO11RSB0	None	**Rule 7**	**Rule 7**
F7	IO11RSB0	IO07RSB0	IO07RSB0	None	None	None
F8	GAC0/IO04RSB0	GAC0/IO04RSB0	GAC0/IO04RSB0	None	None	None
F9	GAC1/IO05RSB0	GAC1/IO05RSB0	GAC1/IO05RSB0	None	None	None

Table 10-7 • **Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
F10	IO25RSB0	IO20RSB0	IO20RSB0	None	None	None
F11	IO36RSB0	IO24RSB0	IO24RSB0	None	None	None
F12	IO42RSB0	IO33RSB0	IO33RSB0	None	None	None
F13	IO49RSB0	IO39RSB0	IO39RSB0	None	None	None
F14	IO56RSB0	IO44RSB0	IO45RSB0	None	None	None
F15	GBC0/IO72RSB0	GBC0/IO54RSB0	GBC0/IO54RSB0	None	None	None
F16	IO62RSB0	IO51RSB0	IO48RSB0	None	None	None
F17	VMV0	VMV0	VMV0	None	None	None
F18	IO78NDB1	IO61NPB1	IO61NPB1	None	None	None
F19	IO81NDB1	IO63PDB1	IO63PDB1	None	None	None
F20	IO82PPB1	NC	NC	**Rule 1**	**Rule 1**	None
F21	NC	NC	NC	None	None	None
F22	IO84NDB1	NC	NC	**Rule 1**	**Rule 1**	None
G1	IO214NDB3	IO170NDB3	NC	None	**Rule 1**	**Rule 1**
G2	IO214PDB3	IO170PDB3	NC	None	**Rule 1**	**Rule 1**
G3	NC	NC	NC	None	None	None
G4	IO222NDB3	IO171NDB3	IO151VDB3	None	None	None
G5	IO222PDB3	IO171PDB3	IO151UDB3	None	None	None
G6	GAC2/IO223PDB3	GAC2/IO172PDB3	GAC2/IO153UDB3	None	None	None
G7	IO223NDB3	IO06RSB0	IO06RSB0	None	None	None
G8	GNDQ	GNDQ	GNDQ	None	None	None
G9	IO23RSB0	IO10RSB0	IO10RSB0	None	None	None
G10	IO29RSB0	IO19RSB0	IO19RSB0	None	None	None
G11	IO33RSB0	IO26RSB0	IO26RSB0	None	None	None
G12	IO46RSB0	IO30RSB0	IO30RSB0	None	None	None
G13	IO52RSB0	IO40RSB0	IO40RSB0	None	None	None
G14	IO60RSB0	IO45RSB0	IO46RSB0	None	None	None
G15	GNDQ	GNDQ	GNDQ	None	None	None
G16	IO80NDB1	IO50RSB0	IO47RSB0	None	None	None
G17	GBB2/IO79PDB1	GBB2/IO61PPB1	GBB2/IO61PPB1	None	None	None
G18	IO79NDB1	IO53RSB0	IO53RSB0	None	None	None
G19	IO82NPB1	IO63NDB1	IO63NDB1	None	None	None
G20	IO85PDB1	NC	NC	**Rule 1**	**Rule 1**	None

Table 10-7 **• Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
G21	IO85NDB1	NC	NC	**Rule 1**	**Rule 1**	None
G22	NC	NC	NC	None	None	None
H1	NC	NC	NC	None	None	None
H2	NC	NC	NC	None	None	None
H3	VCC	VCC	VCC	None	None	None
H4	IO217PDB3	IO166PDB3	IO150PDB3	None	None	None
H5	IO218PDB3	IO167NPB3	IO08RSB0	None	None	None
H6	IO221NDB3	IO172NDB3	IO153VDB3	None	None	None
H7	IO221PDB3	IO169NDB3	IO152VDB3	None	None	None
H8	VMV0	VMV0	VMV0	None	None	None
H9	VCCIB0	VCCIB0	VCCIB0	None	None	None
H10	VCCIB0	VCCIB0	VCCIB0	None	None	None
H11	IO38RSB0	IO25RSB0	IO25RSB0	None	None	None
H12	IO47RSB0	IO31RSB0	IO31RSB0	None	None	None
H13	VCCIB0	VCCIB0	VCCIB0	None	None	None
H14	VCCIB0	VCCIB0	VCCIB0	None	None	None
H15	VMV1	VMV1	VMV1	None	None	None
H16	GBC2/IO80PDB1	GBC2/IO62PDB1	GBC2/IO62PDB1	None	None	None
H17	IO83PPB1	IO67PPB1	IO65RSB1	None	None	None
H18	IO86PPB1	IO64PPB1	IO52RSB0	None	None	None
H19	IO87PDB1	IO66PDB1	IO66PDB1	None	None	None
H20	VCC	VCC	VCC	None	None	None
H21	NC	NC	NC	None	None	None
H22	NC	NC	NC	None	None	None
J1	IO212NDB3	NC	NC	**Rule 1**	**Rule 1**	None
J2	IO212PDB3	NC	NC	**Rule 1**	**Rule 1**	None
J3	NC	NC	NC	None	None	None
J4	IO217NDB3	IO166NDB3	IO150NDB3	None	None	None
J5	IO218NDB3	IO168NPB3	IO149NPB3	None	None	None
J6	IO216PDB3	IO167PPB3	IO09RSB0	None	None	None
J7	IO216NDB3	IO169PDB3	IO152UDB3	None	None	None
J8	VCCIB3	VCCIB3	VCCIB3	None	None	None
J9	GND	GND	GND	None	None	None

Table 10-7 **• Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
J10	VCC	VCC	VCC	None	None	None
J11	VCC	VCC	VCC	None	None	None
J12	VCC	VCC	VCC	None	None	None
J13	VCC	VCC	VCC	None	None	None
J14	GND	GND	GND	None	None	None
J15	VCCIB1	VCCIB1	VCCIB1	None	None	None
J16	IO83NPB1	IO62NDB1	IO62NDB1	None	None	None
J17	IO86NPB1	IO64NPB1	IO49RSB0	None	None	None
J18	IO90PPB1	IO65PPB1	IO64PPB1	None	None	None
J19	IO87NDB1	IO66NDB1	IO66NDB1	None	None	None
J20	NC	NC	NC	None	None	None
J21	IO89PDB1	IO68PDB1	NC	None	**Rule 1**	**Rule 1**
J22	IO89NDB1	IO68NDB1	NC	None	**Rule 1**	**Rule 1**
K1	IO211PDB3	IO157PDB3	NC	None	**Rule 1**	**Rule 1**
K2	IO211NDB3	IO157NDB3	NC	None	**Rule 1**	**Rule 1**
K3	NC	NC	NC	None	None	None
K4	IO210PPB3	IO165NDB3	IO148NDB3	None	None	None
K5	IO213NDB3	IO165PDB3	IO148PDB3	None	None	None
K6	IO213PDB3	IO168PPB3	IO149PPB3	None	None	None
K7	GFC1/IO209PPB3	GFC1/IO164PPB3	GFC1/IO147PPB3	None	None	None
K8	VCCIB3	VCCIB3	VCCIB3	None	None	None
K9	V_{CC}	V_{CC}	V_{CC}	None	None	None
K10	GND	GND	GND	None	None	None
K11	GND	GND	GND	None	None	None
K12	GND	GND	GND	None	None	None
K13	GND	GND	GND	None	None	None
K14	VCC	VCC	VCC	None	None	None
K15	VCCIB1	VCCIB1	VCCIB1	None	None	None
K16	GCC1/IO91PPB1	GCC1/IO69PPB1	GCC1/IO67PPB1	None	None	None
K17	IO90NPB1	IO65NPB1	IO64NPB1	None	None	None
K18	IO88PDB1	IO75PDB1	IO73PDB1	None	None	None
K19	IO88NDB1	IO75NDB1	IO73NDB1	None	None	None
K20	IO94NPB1	NC	NC	**Rule 1**	**Rule 1**	None

***Table 10-7* • Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
K21	IO98NDB1	IO76NDB1	NC	None	**Rule 1**	**Rule 1**
K22	IO98PDB1	IO76PDB1	NC	None	**Rule 1**	**Rule 1**
L1	NC	NC	NC	None	None	None
L2	IO200PDB3	IO155PDB3	NC	None	**Rule 1**	**Rule 1**
L3	IO210NPB3	NC	NC	**Rule 1**	**Rule 1**	None
L4	GFB0/IO208NPB3	GFB0/IO163NPB3	GFB0/IO146NPB3	None	None	None
L5	GFA0/IO207NDB3	GFA0/IO162NDB3	GFA0/IO145NDB3	None	None	None
L6	GFB1/IO208PPB3	GFB1/IO163PPB3	GFB1/IO146PPB3	None	None	None
L7	VCOMPLF	VCOMPLF	VCOMPLF	None	None	None
L8	GFC0/IO209NPB3	GFC0/IO164NPB3	GFC0/IO147NPB3	None	None	None
L9	VCC	VCC	VCC	None	None	None
L10	GND	GND	GND	None	None	None
L11	GND	GND	GND	None	None	None
L12	GND	GND	GND	None	None	None
L13	GND	GND	GND	None	None	None
L14	V_{CC}	V_{CC}	V_{CC}	None	None	None
L15	GCC0/IO91NPB1	GCC0/IO69NPB1	GCC0/IO67NPB1	None	None	None
L16	GCB1/IO92PPB1	GCB1/IO70PPB1	GCB1/IO68PPB1	None	None	None
L17	GCA0/IO93NPB1	GCA0/IO71NPB1	GCA0/IO69NPB1	None	None	None
L18	IO96NPB1	IO67NPB1	NC	None	**Rule 1**	**Rule 1**
L19	GCB0/IO92NPB1	GCB0/IO70NPB1	GCB0/IO68NPB1	None	None	None
L20	IO97PDB1	IO77PDB1	NC	None	**Rule 1**	**Rule 1**
L21	IO97NDB1	IO77NDB1	NC	None	**Rule 1**	**Rule 1**
L22	IO99NPB1	IO78NPB1	NC	None	**Rule 1**	**Rule 1**
M1	NC	NC	NC	None	None	None
M2	IO200NDB3	IO155NDB3	NC	None	**Rule 1**	**Rule 1**
M3	IO206NDB3	IO158NPB3	NC	None	**Rule 1**	**Rule 1**
M4	GFA2/IO206PDB3	GFA2/IO161PPB3	GFA2/IO144PPB3	None	None	None
M5	GFA1/IO207PDB3	GFA1/IO162PDB3	GFA1/IO145PDB3	None	None	None
M6	VCCPLF	VCCPLF	VCCPLF	None	None	None
M7	IO205NDB3	IO160NDB3	IO143NDB3	None	None	None
M8	GFB2/IO205PDB3	GFB2/IO160PDB3	GFB2/IO143PDB3	None	None	None
M9	VCC	VCC	VCC	None	None	None

Table 10-7 • **Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
M10	GND	GND	GND	None	None	None
M11	GND	GND	GND	None	None	None
M12	GND	GND	GND	None	None	None
M13	GND	GND	GND	None	None	None
M14	V_{CC}	V_{CC}	V_{CC}	None	None	None
M15	GCB2/IO95PPB1	GCB2/IO73PPB1	GCB2/IO71PPB1	None	None	None
M16	GCA1/IO93PPB1	GCA1/IO71PPB1	GCA1/IO69PPB1	None	None	None
M17	GCC2/IO96PPB1	GCC2/IO74PPB1	GCC2/IO72PPB1	None	None	None
M18	IO100PPB1	IO80PPB1	NC	None	**Rule 1**	**Rule 1**
M19	GCA2/IO94PPB1	GCA2/IO72PDB1	GCA2/IO70PDB1	None	None	None
M20	IO101PPB1	IO79PPB1	NC	None	**Rule 1**	**Rule 1**
M21	IO99PPB1	IO78PPB1	NC	None	**Rule 1**	**Rule 1**
M22	NC	NC	NC	None	None	None
N1	IO201NDB3	IO154NDB3	NC	None	**Rule 1**	**Rule 1**
N2	IO201PDB3	IO154PDB3	NC	None	**Rule 1**	**Rule 1**
N3	NC	NC	NC	None	None	None
N4	GFC2/IO204PDB3	GFC2/IO159PDB3	GFC2/IO142PDB3	None	None	None
N5	IO204NDB3	IO161NPB3	IO144NPB3	None	None	None
N6	IO203NDB3	IO156PPB3	IO141PPB3	None	None	None
N7	IO203PDB3	IO129RSB2	IO120RSB2	None	None	None
N8	VCCIB3	VCCIB3	VCCIB3	None	None	None
N9	V_{CC}	V_{CC}	V_{CC}	None	None	None
N10	GND	GND	GND	None	None	None
N11	GND	GND	GND	None	None	None
N12	GND	GND	GND	None	None	None
N13	GND	GND	GND	None	None	None
N14	VCC	VCC	VCC	None	None	None
N15	VCCIB1	VCCIB1	VCCIB1	None	None	None
N16	IO95NPB1	IO73NPB1	IO71NPB1	None	None	None
N17	IO100NPB1	IO80NPB1	IO74RSB1	None	None	None
N18	IO102NDB1	IO74NPB1	IO72NPB1	None	None	None
N19	IO102PDB1	IO72NDB1	IO70NDB1	None	None	None
N20	NC	NC	NC	None	None	None

Table 10-7 **• Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
N21	IO101NPB1	IO79NPB1	NC	None	**Rule 1**	**Rule 1**
N22	IO103PDB1	NC	NC	**Rule 1**	**Rule 1**	None
P1	NC	NC	NC	None	None	None
P2	IO199PDB3	IO153PDB3	NC	None	**Rule 1**	**Rule 1**
P3	IO199NDB3	IO153NDB3	NC	None	**Rule 1**	**Rule 1**
P4	IO202NDB3	IO159NDB3	IO142NDB3	None	None	None
P5	IO202PDB3	IO156NPB3	IO141NPB3	None	None	None
P6	IO196PPB3	IO151PPB3	IO125RSB2	None	None	None
P7	IO193PPB3	IO158PPB3	IO139RSB3	None	None	None
P8	VCCIB3	VCCIB3	VCCIB3	None	None	None
P9	GND	GND	GND	None	None	None
P10	VCC	VCC	VCC	None	None	None
P11	VCC	VCC	VCC	None	None	None
P12	VCC	V_{CC}	VCC	None	None	None
P13	VCC	VCC	VCC	None	None	None
P14	GND	GND	GND	None	None	None
P15	VCCIB1	VCCIB1	VCCIB1	None	None	None
P16	GDB0/IO112NPB1	GDB0/IO87NPB1	GDB0/IO78VPB1	None	None	None
P17	IO106NDB1	IO85NDB1	IO76VDB1	None	None	None
P18	IO106PDB1	IO85PDB1	IO76UDB1	None	None	None
P19	IO107PDB1	IO84PDB1	IO75PDB1	None	None	None
P20	NC	NC	NC	None	None	None
P21	IO104PDB1	IO81PDB1	NC	None	**Rule 1**	**Rule 1**
P22	IO103NDB1	NC	NC	**Rule 1**	**Rule 1**	None
R1	NC	NC	NC	None	None	None
R2	IO197PPB3	NC	NC	**Rule 1**	**Rule 1**	None
R3	VCC	VCC	VCC	None	None	None
R4	IO197NPB3	IO150PDB3	IO140PDB3	None	None	None
R5	IO196NPB3	IO151NPB3	IO130RSB2	None	None	None
R6	IO193NPB3	IO147NPB3	IO138NPB3	None	None	None
R7	GEC0/IO190NPB3	GEC0/IO146NPB3	GEC0/IO137NPB3	None	None	None
R8	VMV3	VMV3	VMV3	None	None	None
R9	VCCIB2	VCCIB2	VCCIB2	None	None	None

***Table 10-7* • Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
R10	VCCIB2	VCCIB2	VCCIB2	None	None	None
R11	IO147RSB2	IO117RSB2	IO108RSB2	None	None	None
R12	IO136RSB2	IO110RSB2	IO101RSB2	None	None	None
R13	VCCIB2	VCCIB2	VCCIB2	None	None	None
R14	VCCIB2	VCCIB2	VCCIB2	None	None	None
R15	VMV2	VMV2	VMV2	None	None	None
R16	IO110NDB1	IO94RSB2	IO83RSB2	None	None	None
R17	GDB1/IO112PPB1	GDB1/IO87PPB1	GDB1/IO78UPB1	None	None	None
R18	GDC1/IO111PDB1	GDC1/IO86PDB1	GDC1/IO77UDB1	None	None	None
R19	IO107NDB1	IO84NDB1	IO75NDB1	None	None	None
R20	VCC	VCC	VCC	None	None	None
R21	IO104NDB1	IO81NDB1	NC	None	**Rule 1**	**Rule 1**
R22	IO105PDB1	IO82PDB1	NC	None	**Rule 1**	**Rule 1**
T1	IO198PDB3	IO152PDB3	NC	None	**Rule 1**	**Rule 1**
T2	IO198NDB3	IO152NDB3	NC	None	**Rule 1**	**Rule 1**
T3	NC	NC	NC	None	None	None
T4	IO194PPB3	IO150NDB3	IO140NDB3	None	None	None
T5	IO192PPB3	IO147PPB3	IO138PPB3	None	None	None
T6	GEC1/IO190PPB3	GEC1/IO146PPB3	GEC1/IO137PPB3	None	None	None
T7	IO192NPB3	IO140RSB2	IO131RSB2	None	None	None
T8	GNDQ	GNDQ	GNDQ	None	None	None
T9	GEA2/IO187RSB2	GEA2/IO143RSB2	GEA2/IO134RSB2	None	None	None
T10	IO161RSB2	IO126RSB2	IO117RSB2	None	None	None
T11	IO155RSB2	IO120RSB2	IO111RSB2	None	None	None
T12	IO141RSB2	IO108RSB2	IO99RSB2	None	None	None
T13	IO129RSB2	IO103RSB2	IO94RSB2	None	None	None
T14	IO124RSB2	IO99RSB2	IO87RSB2	None	None	None
T15	GNDQ	GNDQ	GNDQ	None	None	None
T16	IO110PDB1	IO92RSB2	IO93RSB2	None	None	None
T17	VJTAG	VJTAG	VJTAG	None	None	None
T18	GDC0/IO111NDB1	GDC0/IO86NDB1	GDC0/IO77VDB1	None	None	None
T19	GDA1/IO113PDB1	GDA1/IO88PDB1	GDA1/IO79UDB1	None	None	None
T20	NC	NC	NC	None	None	None

Table 10-7 • **Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
T21	IO108PDB1	IO83PDB1	NC	None	**Rule 1**	**Rule 1**
T22	IO105NDB1	IO82NDB1	NC	None	**Rule 1**	**Rule 1**
U1	IO195PDB3	IO149PDB3	NC	None	**Rule 1**	**Rule 1**
U2	IO195NDB3	IO149NDB3	NC	None	**Rule 1**	**Rule 1**
U3	IO194NPB3	NC	NC	**Rule 1**	**Rule 1**	None
U4	GEB1/IO189PDB3	GEB1/IO145PDB3	GEB1/IO136PDB3	None	None	None
U5	GEB0/IO189NDB3	GEB0/IO145NDB3	GEB0/IO136NDB3	None	None	None
U6	VMV2	VMV2	VMV2	None	None	None
U7	IO179RSB2	IO138RSB2	IO129RSB2	None	None	None
U8	IO171RSB2	IO136RSB2	IO128RSB2	None	None	None
U9	IO165RSB2	IO131RSB2	IO122RSB2	None	None	None
U10	IO159RSB2	IO124RSB2	IO115RSB2	None	None	None
U11	IO151RSB2	IO119RSB2	IO110RSB2	None	None	None
U12	IO137RSB2	IO107RSB2	IO98RSB2	None	None	None
U13	IO134RSB2	IO104RSB2	IO95RSB2	None	None	None
U14	IO128RSB2	IO97RSB2	IO88RSB2	None	None	None
U15	VMV1	VMV1	IO84RSB2	None	**Rule 7**	**Rule 7**
U16	TCK	TCK	TCK	None	None	None
U17	VPUMP	VPUMP	VPUMP	None	None	None
U18	TRST	TRST	TRST	None	None	None
U19	GDA0/IO113NDB1	GDA0/IO88NDB1	GDA0/IO79VDB1	None	None	None
U20	NC	NC	NC	None	None	None
U21	IO108NDB1	IO83NDB1	NC	None	**Rule 1**	**Rule 1**
U22	IO109PDB1	NC	NC	**Rule 1**	**Rule 1**	None
V1	NC	NC	NC	None	None	None
V2	NC	NC	NC	None	None	None
V3	GND	GND	GND	None	None	None
V4	GEA1/IO188PDB3	GEA1/IO144PDB3	GEA1/IO135PDB3	None	None	None
V5	GEA0/IO188NDB3	GEA0/IO144NDB3	GEA0/IO135NDB3	None	None	None
V6	IO184RSB2	IO139RSB2	IO127RSB2	None	None	None
V7	GEC2/IO185RSB2	GEC2/IO141RSB2	GEC2/IO132RSB2	None	None	None
V8	IO168RSB2	IO132RSB2	IO123RSB2	None	None	None
V9	IO163RSB2	IO127RSB2	IO118RSB2	None	None	None

Table 10-7 **• Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
V10	IO157RSB2	IO121RSB2	IO112RSB2	None	None	None
V11	IO149RSB2	IO114RSB2	IO106RSB2	None	None	None
V12	IO143RSB2	IO109RSB2	IO100RSB2	None	None	None
V13	IO138RSB2	IO105RSB2	IO96RSB2	None	None	None
V14	IO131RSB2	IO98RSB2	IO89RSB2	None	None	None
V15	IO125RSB2	IO96RSB2	IO85RSB2	None	None	None
V16	GDB2/IO115RSB2	GDB2/IO90RSB2	GDB2/IO81RSB2	None	None	None
V17	TDI	TDI	TDI	None	None	None
V18	GNDQ	GNDQ	NC	None	**Rule 8**	**Rule 8**
V19	TDO	TDO	TDO	None	None	None
V20	GND	GND	GND	None	None	None
V21	NC	NC	NC	None	None	None
V22	IO109NDB1	NC	NC	**Rule 1**	**Rule 1**	None
W1	NC	NC	NC	None	None	None
W2	IO191PDB3	IO148PDB3	NC	None	**Rule 1**	**Rule 1**
W3	NC	NC	NC	None	None	None
W4	GND	GND	GND	None	None	None
W5	IO183RSB2	IO137RSB2	IO126RSB2	None	None	None
W6	GEB2/IO186RSB2	GEB2/IO142RSB2	GEB2/IO133RSB2	None	None	None
W7	IO172RSB2	IO134RSB2	IO124RSB2	None	None	None
W8	IO170RSB2	IO125RSB2	IO116RSB2	None	None	None
W9	IO164RSB2	IO123RSB2	IO113RSB2	None	None	None
W10	IO158RSB2	IO118RSB2	IO107RSB2	None	None	None
W11	IO153RSB2	IO115RSB2	IO105RSB2	None	None	None
W12	IO142RSB2	IO111RSB2	IO102RSB2	None	None	None
W13	IO135RSB2	IO106RSB2	IO97RSB2	None	None	None
W14	IO130RSB2	IO102RSB2	IO92RSB2	None	None	None
W15	GDC2/IO116RSB2	GDC2/IO91RSB2	GDC2/IO82RSB2	None	None	None
W16	IO120RSB2	IO93RSB2	IO86RSB2	None	None	None
W17	GDA2/IO114RSB2	GDA2/IO89RSB2	GDA2/IO80RSB2	None	None	None
W18	TMS	TMS	TMS	None	None	None
W19	GND	GND	GND	None	None	None
W20	NC	NC	NC	None	None	None

***Table 10-7* • Pin Compatibility and Migration Table for the FG484 Package (continued)**

Pin Number	A3P1000 Function	A3P600 Function	A3P400 Function	Migration Rule between A3P1000 and A3P600	Migration Rule between A3P1000 and A3P400	Migration Rule between A3P600 and A3P400
W21	NC	NC	NC	None	None	None
W22	NC	NC	NC	None	None	None
Y1	VCCIB3	VCCIB3	VCCIB3	None	None	None
Y2	IO191NDB3	IO148NDB3	NC	None	**Rule 1**	**Rule 1**
Y3	NC	NC	NC	None	None	None
Y4	IO182RSB2	NC	NC	**Rule 1**	**Rule 1**	None
Y5	GND	GND	GND	None	None	None
Y6	IO177RSB2	NC	NC	**Rule 1**	**Rule 1**	None
Y7	IO174RSB2	NC	NC	**Rule 1**	**Rule 1**	None
Y8	VCC	VCC	VCC	None	None	None
Y9	VCC	VCC	VCC	None	None	None
Y10	IO154RSB2	NC	NC	**Rule 1**	**Rule 1**	None
Y11	IO148RSB2	NC	NC	**Rule 1**	**Rule 1**	None
Y12	IO140RSB2	NC	NC	**Rule 1**	**Rule 1**	None
Y13	NC	NC	NC	None	None	None
Y14	VCC	VCC	VCC	None	None	None
Y15	VCC	VCC	VCC	None	None	None
Y16	NC	NC	NC	None	None	None
Y17	NC	NC	NC	None	None	None
Y18	GND	GND	GND	None	None	None
Y19	NC	NC	NC	None	None	None
Y20	NC	NC	NC	None	None	None
Y21	NC	NC	NC	None	None	None
Y22	VCCIB1	VCCIB1	VCCIB1	None	None	None

Conclusion

This application note describes design migration among ProASIC3 family devices with an emphasis on package pin compatibility. The ProASIC3 family of devices shares numerous common architectural features. During a system migration, care should be taken regarding the architectural features of each core. Additionally, a key requirement is running functional simulation before and after the migration, using Microsemi tools. Microsemi will be updating this document with additional packages in the future.

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	Rule 10 was added to Table 10-3 • Migration Rules from Higher-Density to Mid-Density Devices and listed for the rule to follow on pin 126 of thePQ208 package in Table 10-4 • Pin Compatibility and Migration Table for the PQ208 Package (SAR 24369).	237
v1.1 (March 2008)	The part number for this document was changed from 51700094-016-0 to 51700094-017-1.	N/A
v1.0 (January 2008)	In Table 10-4 • Pin Compatibility and Migration Table for the PQ208 Package, pin 107 was updated to include Rule 8.	239
51900143-1/10.07	The title of the document and the "Introduction" section were updated to clarify the topics covered in the application note.	235
	The "Design Migration" section was updated to match and cross-reference the sections of the document in sequence.	235
	The title of Table 10-1 • Device Information was updated.	235
	Table 10-3 • Migration Rules from Higher-Density to Mid-Density Devices and Table 4 were combined and a table note was added to refer to the datasheet for I/O naming conventions.	237
	In the "Power Supply and Board-Level Considerations" section, the following bullet was removed: "Since each bank independently supports 1.5 V to 3.3 V, I/Os must be connected to the V_{CCI}Bx of their own banks."	238
	The "Related Documents" section was added.	40

11 – Migrating Designs from A3P250 to Lower-Logic-Density Devices

Introduction

The purpose of this document is to assist you in migrating designs from a high-density ProASIC®3 device (A3P250) to lower-density devices (A3P125, A3P060, and A3P030). Since one of the key factors is pin compatibility for a given package among the devices within the family, the primary focus of this document will be to address the pin compatibility issue.

Design Migration

ProASIC3 family devices are architecturally compatible with each other. However, customers must pay attention to a few key areas when migrating a design. The specific issues discussed throughout this application note are as follows:

- "Design and Device Evaluation"
- "Device and Package Compatibility" on page 276
- "Migration and Implementation Methodologies" on page 276
- "I/O Banks and Standards" on page 277
- "Power Supply Considerations" on page 278
- "Pin Migration and Compatibility" on page 279

Design and Device Evaluation

When migrating a design, the primary task should be to compare the available resources between two devices. You need to evaluate effective gate count, RAM size, I/O banks, and the number of I/Os. In addition, when porting designs to new ProASIC3 derivatives, timing analysis and simulations should also be validated. Table 11-1 gives a summary of device resources for the A3P250 device and its smaller migration targets.

***Table 11-1* • Device Information**

	A3P250	A3P125	A3P060	A3P030
System Gates	250 k	125 k	60 k	30 k
Tiles (D-flip-flops)	6,144	3,072	1,536	768
RAM (kbits)	36	36	18	–
RAM Blocks (4,608 bits)	8	8	4	–
I/O Banks (+ JTAG)	4	2	2	2
User I/Os per Package:				
VQ100	68/13	71	71	77
QN132	87/19	84	80	81
TQ144		100	91	
FG144	97/24	97	96	
PQ208	151/34	133		
FG256	157/38			

Note: User I/O is given as X (single-ended) or X/Y (single-ended/double-ended).

Device and Package Compatibility

ProASIC3 devices and packaging were designed to allow considerable footprint compatibility for smoother migration.

Common and Convertible I/Os among A3P030, A3P060, A3P125, and A3P250

Table 11-2 shows the number of I/Os that are common between any two of the above four devices. In addition, the table indicates the number of I/Os that require the necessary conversion (convertible I/Os) using suggested design migration rules in the "Migration and Implementation Methodologies" section.

***Table 11-2* • Common and Convertible I/Os**

Package	A3P250 A3P125		A3P250 A3P060		A3P250 A3P030		A3P125 A3P060		A3P125 A3P030		A3P060 A3P030	
	Common I/Os	Convertible I/Os	Common I/Os	Convertible I/Os	Common I/Os	Convertible I/Os	Common I/Os	Convertible I/Os	Common I/Os	Convertible I/Os	Common I/Os	Convertible I/Os
VQ100	68	7	67	5	69	46	69	4	72	46	71	46
QN132	84	13	80	25	64	72	80	14	66	70	61	75
FG144	97	–	96	–	N/A	N/A	96	–	N/A	N/A	N/A	N/A
PQ208	134	18	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
TQ144	N/A	N/A	N/A	N/A	N/A	N/A	90	19	N/A	N/A	N/A	N/A

Migration and Implementation Methodologies

Table 11-3 on page 277 lists some possible migration combinations and the recommended implementation rules for compatible design conversions from higher-density to lower-density devices. The "Pin Migration and Compatibility" section on page 279 contains tables that list the required rules for different pin combinations. If "Rule x" is mentioned for a pin combination, that combination requires the implementation methodology given in Table 11-3 on page 277. Note that many combinations of high-density/low-density pins require none of these rules; the pins have complete type compatibility. These pins are marked in the pin tables with "None."

Table 11-3 • **Migration Rules from Higher-Density Device to Lower-Density Device**

Migration Rule	**Issue**		
	Higher Density	**Lower Density**	**Implementation Methodology**
1	I/O or Global I/O	NC	Leave this pin floating OR program the I/O as unused (software cannot program NC to usable I/O).
	NC	I/O or Global I/O	
2	I/O	Global I/O	Instantiate the global I/O as an I/O buffer (works as a single-ended I/O).
3	Global I/O	I/O	Use the PDC constraint to promote the single-ended I/O to a global pin. There will be some additional delay.
4	VCC or VCCI	NC	The pin can remain connected to the board's VCC, VCCI, VMV, VCOMPLF, or GNDQ plane, as applicable.
	GNDQ	NC	
	VMV	NC	
	VCOMPLF	NC	
5	VCCIB(x)[1]	VCCI B(y)[2]	Make sure the two bank voltage levels are same. Tie the pin to the board's corresponding VCCI/VMV plane.
	VMV(x)[1]	VMV(y)[2]	
6	VMV0	I/O or Global I/O	Leave the pin connected to the board's VCC, VCCI, VMV, VCOMPLF, or GNDQ plane, as applicable. Instantiate the I/O as a tristate buffer with OE = 0 and no weak pull-ups/-downs.
	GNDQ	I/O	
	GNDQ	Global I/O	
	I/O	VCC	
	VCC	I/O	
	VCOMPLF	I/O	
	VCCPLF	I/O	
7	I/O bank 2	I/O bank 0	Bank 2 of the larger device must be at the same voltage level as bank 0 of the smaller device, which effectively limits the larger device to a single voltage.

Notes:

1. *"x" is a bank number designator and can be 0–3 for ProASIC3 and 0–7 for ProASIC3E.*
2. *"y" is a bank number designator and can be 0–3 for ProASIC3 and 0–7 for ProASIC3E.*

I/O Banks and Standards

ProASIC3 I/Os are partitioned into multiple I/O voltage banks. The number of banks is device-dependent. There are four I/O banks in the A3P250 device and two I/O banks in the A3P030, A3P060, and A3P125 devices.

Package pins routed to banks 0 and 1 in the A3P250 device are routed to bank 0 in the A3P030, A3P060, and A3P125 devices, and banks 2 and 3 in the A3P250 device are routed to bank 1 in the A3P030, A3P060, and A3P125 devices.

The banks have dedicated supplies; therefore, only I/Os with compatible voltage standards can be assigned to the same I/O voltage bank.

Note that the A3P250 device supports double-ended I/Os; however, the A3P030, A3P060, and A3P125 devices do not support double-ended I/Os.

Power Supply Considerations

I/O power supply requirements are very important for design migration. Since the migration is within the ProASIC3 family, there is no issue with respect to the core voltage VCC. Pins that must be appropriately connected are VCCIBx (bank supply voltage to I/O output buffer and I/O logic), VMVx (quiet I/O supply voltage), GNDQ (quiet GND), and GND. GNDQ and VMVx are important to decouple simultaneous switching noise (SSO) for I/Os—enhancing signal integrity and improving noise immunity.

The key rules of migration for the above-mentioned pins are as follows:

- VMV and VCCI values of the higher-density device in a given bank must correspond to the same VMV and VCCI values in the smaller device's migrating bank.
- Since banks 0 and 1 are connected to bank 0 in the smaller device—and banks 2 and 3 are connected to bank 1 in the smaller device—this implies that banks 0 and 1 in the A3P250 device must have identical VMV and identical VCCI. Similarly, the VMV and VCCI voltages in banks 2 and 3 of the A3P250 device must be identical.
- VCCIBx pins in unused banks and VMV pins in unused banks must be connected to GND.
- Unused I/Os should be left alone, since the software automatically configures them as inputs with pull-ups.

Any inappropriate connection during the migration may affect overall dynamic or inrush power consumption and might even result in device malfunction.

Additionally, the I/O naming convention in ProASIC3 devices has significant embedded information (e.g., pin location, bank number, signal type, polarity, and clock conditioning). For a detailed explanation, refer to the "User I/O Naming Convention" section on page 192. For additional information on power issues, refer to the relevant datasheet.

Pin Migration and Compatibility

VQ100 Package

***Table 11-4* • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging**

Pin No.	**A3P030 Function**	**A3P060 Function**	**A3P125 Function**	**A3P250 Function**	**Migration Rule between A3P125 and A3P060**	**Migration Rule between A3P250 and A3P060**	**Migration Rule between A3P250 and A3P125**	**Migration Rule between A3P060 and A3P030**	**Migration Rule between A3P125 and A3P030**	**Migration Rule between A3P250 and A3P030**
1	GND	GND	GND	GND	None	None	None	None	None	None
2	IO82RSB1	**GAA2/IO51RSB1**	**GAA2/IO67RSB1**	**GAA2/IO118UDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
3	IO81RSB1	IO52RSB1	IO68RSB1	IO118VDB3	None	None	None	None	None	None
4	**IO80RSB1**	**GAB2/IO53RSB1**	**GAB2/IO69RSB1**	**GAB2/IO117UDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
5	IO79RSB1	IO95RSB1	IO132RSB1	IO117VDB3	None	None	None	None	None	None
6	**IO78RSB1**	**GAC2/IO94RSB1**	**GAC2/IO131RSB1**	**GAC2/IO116UDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
7	IO77RSB1	IO93RSB1	IO130RSB1	IO116VDB3	None	None	None	None	None	None
8	**IO76RSB1**	**IO92RSB1**	**IO129RSB1**	**IO112PSB3**	None	None	None	None	None	None
9	GND	GND	GND	GND	None	None	None	None	None	None
10	IO75RSB1	GFB1/IO87RSB1	GFB1/IO124RSB1	GFB1/IO109PDB3	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
11	IO74RSB1	GFB0/IO86RSB1	GFB0/IO123RSB1	GFB0/IO109NDB3	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
12	**GEC0/IO73RSB1**	**VCOMPLF**	**VCOMPLF**	**VCOMPLF**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
13	**GEA0/IO72RSB1**	**GFA0/IO85RSB1**	**GFA0/IO122RSB1**	**GFA0/IO108NPB3**	None	None	None	None	None	None
14	**GEB0/IO71RSB1**	**VCCPLF**	**VCCPLF**	**VCCPLF**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
15	**IO70RSB1**	**GFA1/IO84RSB1**	**GFA1/IO121RSB1**	**GFA1/IO108PPB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
16	**IO69RSB1**	**GFA2/IO83RSB1**	**GFA2/IO120RSB1**	**GFA2/IO107PSB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
17	VCC	VCC	VCC	VCC	None	None	None	None	None	None
18	**VCCIB1**	**VCCIB1**	**VCCIB1**	**VCCIB3**	None	None	None	None	None	None
19	**IO68RSB1**	**GEC1/IO77RSB1**	**GEC0/IO111RSB1**	**GFC2/IO105PSB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
20	**IO67RSB1**	**GEB1/IO75RSB1**	**GEB1/IO110RSB1**	**GEC1/IO100PDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-4 • **Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125	Migration Rule between A3P060 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P250 and A3P030
21	**IO66RSB1**	**GEB0/IO74RSB1**	**GEB0/IO109RSB1**	**GEC0/IO100NDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
22	**IO65RSB1**	**GEA1/IO73RSB1**	**GEA1/IO108RSB1**	**GEA1/IO98PDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
23	**IO64RSB1**	**GEA0/IO72RSB1**	**GEA0/IO107RSB1**	**GEA0/IO98NDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
24	**IO63RSB1**	**VMV1**	**VMV1**	**VMV3**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
25	**IO62RSB1**	**GNDQ**	**GNDQ**	**GNDQ**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
26	**IO61RSB1**	**GEA2/IO71RSB1**	**GEA2/IO106RSB1**	**GEA2/IO97RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
27	**IO60RSB1**	**GEB2/IO70RSB1**	**GEB2/IO105RSB1**	**GEB2/IO96RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
28	**IO59RSB1**	**GEC2/IO69RSB1**	**GEC2/IO104RSB1**	**GEC2/IO95RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
29	IO58RSB1	IO68RSB1	IO102RSB1	IO93RSB2	None	None	None	None	None	None
30	IO57RSB1	IO67RSB1	IO100RSB1	IO92RSB2	None	None	None	None	None	None
31	IO56RSB1	IO66RSB1	IO99RSB1	IO91RSB2	None	None	None	None	None	None
32	IO55RSB1	IO65RSB1	IO97RSB1	IO90RSB2	None	None	None	None	None	None
33	IO54RSB1	IO64RSB1	IO96RSB1	IO88RSB2	None	None	None	None	None	None
34	IO53RSB1	IO63RSB1	IO95RSB1	IO86RSB2	None	None	None	None	None	None
35	IO52RSB1	IO62RSB1	IO94RSB1	IO85RSB2	None	None	None	None	None	None
36	IO51RSB1	IO61RSB1	IO93RSB1	IO84RSB2	None	None	None	None	None	None
37	VCC	V_{CC}	VCC	VCC	None	None	None	None	None	None
38	GND	GND	GND	GND	None	None	None	None	None	None
39	VCCIB1	VCCIB1	VCCIB1	VCCIB2	None	None	None	None	None	None
40	IO49RSB1	IO60RSB1	IO87RSB1	IO77RSB2	None	None	None	None	None	None
41	IO47RSB1	IO59RSB1	IO84RSB1	IO74RSB2	None	None	None	None	None	None
42	IO46RSB1	IO58RSB1	IO81RSB1	IO71RSB2	None	None	None	None	None	None
43	**IO45RSB1**	**IO57RSB1**	**IO75RSB1**	**GDC2/IO63RSB2**	None	**Rule 2**	**Rule 3**	None	None	**Rule 3**
44	**IO44RSB1**	**GDC2/IO56RSB1**	**GDC2/IO72RSB1**	**GDB2/IO62RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

***Table 11-4* • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125	Migration Rule between A3P060 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P250 and A3P030
45	**IO43RSB1**	**GDB2/ IO55RSB1**	**GDB2/ IO71RSB1**	**GDA2/ IO61RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
46	**IO42RSB1**	**GDA2/ IO54RSB1**	**GDA2/ IO70RSB1**	**GNDQ**	None	**Rule 6**	**Rule 6**	**Rule 3**	**Rule 3**	**Rule 6**
47	TCK	TCK	TCK	TCK	None	None	None	None	None	None
48	TDI	TDI	TDI	TDI	None	None	None	None	None	None
49	TMS	TMS	TMS	TMS	None	None	None	None	None	None
50	**Not Bonded**	**VMV1**	**VMV1**	**VMV2**	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
51	GND	GND	GND	GND	None	None	None	None	None	None
52	VPUMP	VPUMP	VPUMP	VPUMP	None	None	None	None	None	None
53	Not Bonded	NC	NC	NC	None	None	None	None	None	None
54	TDO	TDO	TDO	TDO	None	None	None	None	None	None
55	TRST	TRST	TRST	TRST	None	None	None	None	None	None
56	VJTAG	VJTAG	VJTAG	VJTAG	None	None	None	None	None	None
57	**IO41RSB0**	**GDA1/ IO49RSB0**	**GDA1/ IO65RSB0**	**GDA1/ IO60USB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
58	**IO40RSB0**	**GDC0/ IO46RSB0**	**GDC0/ IO62RSB0**	**GDC0/ IO58VDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
59	**IO39RSB0**	**GDC1/ IO45RSB0**	**GDC1/ IO61RSB0**	**GDC1/ IO58UDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
60	**IO38RSB0**	**GCC2/ IO43RSB0**	**GCC2/ IO59RSB0**	**IO52NDB1**	None	**Rule 2**	**Rule 2**	**Rule 3**	**Rule 3**	**None**
61	**IO37RSB0**	**GCB2/ IO42RSB0**	**GCB2/ IO58RSB0**	**GCB2/ IO52PDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
62	**IO36RSB0**	**GCA0/ IO40RSB0**	**GCA0/ IO56RSB0**	**GCA1/ IO50PDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
63	GDB0/IO34RSB0	**GCA1/ IO39RSB0**	**GCA1/ IO55RSB0**	**GCA0/ IO50NDB1**	None	None	None	None	None	None
64	GDA0/IO33RSB0	GCC0/ IO36RSB0	GCC0/ IO52RSB0	GCC0/ IO48NDB1	None	None	None	None	None	None
65	GDC0/IO32RSB0	GCC1/ IO35RSB0	GCC1/ IO51RSB0	GCC1/ IO48PDB1	None	None	None	None	None	None
66	VCCIB0	VCCIB0	VCCIB0	VCCIB1	None	None	None	None	None	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-4 **• Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125	Migration Rule between A3P060 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P250 and A3P030
67	GND	GND	GND	GND	None	None	None	None	None	None
68	VCC	VCC	VCC	VCC	None	None	None	None	None	None
69	IO31RSB0	IO31RSB0	IO47RSB0	IO43NDB1	None	None	None	None	None	None
70	**IO30RSB0**	**GBC2/ IO29RSB0**	**GBC2/ IO45RSB0**	**GBC2/ IO43PDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
71	**IO29RSB0**	**GBB2/ IO27RSB0**	**GBB2/ IO43RSB0**	**GBB2/ IO42PSB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
72	IO28RSB0	IO26RSB0	IO42RSB0	IO41NDB1	None	None	None	None	None	None
73	**IO27RSB0**	**GBA2/ IO25RSB0**	**GBA2/ IO41RSB0**	**GBA2/ IO41PDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
74	**IO26RSB0**	**VMV0**	**VMV0**	**VMV1**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
75	**IO25RSB0**	**GNDQ**	**GNDQ**	**GNDQ**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
76	**IO24RSB0**	**GBA1/ IO24RSB0**	**GBA1/ IO40RSB0**	**GBA1/ IO40RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
77	**IO23RSB0**	**GBA0/ IO23RSB0**	**GBA0/ IO39RSB0**	**GBA0/ IO39RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
78	**IO22RSB0**	**GBB1/ IO22RSB0**	**GBB1/ IO38RSB0**	**GBB1/ IO38RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
79	**IO21RSB0**	**GBB0/ IO21RSB0**	**GBB0/ IO37RSB0**	**GBB0/ IO37RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
80	**IO20RSB0**	**GBC1/ IO20RSB0**	**GBC1/ IO36RSB0**	**GBC1/ IO36RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
81	**IO19RSB0**	**GBC0/ IO19RSB0**	**GBC0/ IO35RSB0**	**GBC0/ IO35RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
82	IO18RSB0	IO18RSB0	IO32RSB0	IO29RSB0	None	None	None	None	None	None
83	IO17RSB0	IO17RSB0	IO28RSB0	IO27RSB0	None	None	None	None	None	None
84	IO16RSB0	IO15RSB0	IO25RSB0	IO25RSB0	None	None	None	None	None	None
85	IO15RSB0	IO13RSB0	IO22RSB0	IO23RSB0	None	None	None	None	None	None
86	IO14RSB0	IO11RSB0	IO19RSB0	IO21RSB0	None	None	None	None	None	None
87	VCCIB0	VCCIB0	VCCIB0	VCCIB0	None	None	None	None	None	None
88	GND	GND	GND	GND	None	None	None	None	None	None
89	VCC	VCC	VCC	VCC	None	None	None	None	None	None
90	IO12RSB0	IO10RSB0	IO15RSB0	IO15RSB0	None	None	None	None	None	None
91	IO10RSB0	IO09RSB0	IO13RSB0	IO13RSB0	None	None	None	None	None	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-4 • **Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125	Migration Rule between A3P060 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P250 and A3P030
92	IO08RSB0	IO08RSB0	IO11RSB0	IO11RSB0	None	None	None	None	None	None
93	**IO07RSB0**	**GAC1/ IO07RSB0**	**IO09RSB0**	**GAC1/ IO05RSB0**	**Rule 2**	None	**Rule 3**	**Rule 3**	None	**Rule 3**
94	**IO06RSB0**	**GAC0/ IO06RSB0**	**IO07RSB0**	**GAC0/ IO04RSB0**	**Rule 2**	None	**Rule 3**	**Rule 3**	None	**Rule 3**
95	**IO05RSB0**	**GAB1/ IO05RSB0**	**GAC1/ IO05RSB0**	**GAB1/ IO03RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
96	**IO04RSB0**	**GAB0/ IO04RSB0**	**GAC0/ IO04RSB0**	**GAB0/ IO02RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
97	**IO03RSB0**	**GAA1/ IO03RSB0**	**GAB1/ IO03RSB0**	**GAA1/ IO01RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
98	**IO02RSB0**	**GAA0/ IO02RSB0**	**GAB0/ IO02RSB0**	**GAA0/ IO00RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
99	**IO01RSB0**	**IO01RSB0**	**GAA1/ IO01RSB0**	**GNDQ**	**Rule 3**	**Rule 6**	**Rule 6**	None	**Rule 3**	**Rule 6**
100	**IO00RSB0**	**IO00RSB0**	**GAA0/ IO00RSB0**	**VMV0**	**Rule 3**	**Rule 6**	**Rule 6**	None	**Rule 3**	**Rule 6**

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

QFN132 Package

***Table 11-5* • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P060 and A3P030
A1	**IO01RSB1**	**GAB2/ IO00RSB1**	**GAB2/ IO69RSB1**	**GAB2/ IO117UPB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A2	IO81RSB1	IO93RSB1	IO130RSB1	IO117VPB3	None	None	None	None	None	None
A3	**NC**	**VCCIB1**	**VCCIB1**	**VCCIB3**	**Rule 5**	**None**	**Rule 5**	**Rule 4**	**Rule 4**	**Rule 4**
A4	IO80RSB1	GFC1/ IO89RSB1	GFC1/ IO126RSB1	GFC1/ IO110PDB3	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A5	GEC0/ IO77RSB1	**GFB0/ IO86RSB1**	**GFB0/ IO123RSB1**	**GFB0/ IO109NPB3**	None	None	None	None	None	None
A6	NC	**VCCPLF**	**VCCPLF**	**VCCPLF**	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
A7	GEB0/ IO75RSB1	**GFA1/ IO84RSB1**	**GFA1/ IO121RSB1**	**GFA1/ IO108PPB3**	None	None	None	None	None	None
A8	IO73RSB1	**GFC2/ IO81RSB1**	**GFC2/ IO118RSB1**	**GFC2/ IO105PPB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A9	**NC**	**IO78RSB1**	**IO115RSB1**	**IO103NDB3**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
A10	VCC	VCC	VCC	VCC	None	None	None	None	None	None
A11	**IO71RSB1**	**GEB1/ IO75RSB1**	**GEB1/ IO110RSB1**	**GEA1/ IO98PPB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A12	**IO68RSB1**	**GEA0/ IO72RSB1**	**GEA0/ IO107RSB1**	**GEA0/ IO98NPB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A13	**IO63RSB1**	**GEC2/ IO69RSB1**	**GEC2/ IO104RSB1**	**GEC2/ IO95RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A14	IO60RSB1	IO65RSB1	IO100RSB1	IO91RSB2	None	None	None	None	None	None
A15	NC	VCC	VCC	VCC	None	None	None	None	None	None
A16	IO59RSB1	IO64RSB1	IO99RSB1	IO90RSB2	None	None	None	None	None	None
A17	IO57RSB1	IO63RSB1	IO96RSB1	IO87RSB2	None	None	None	None	None	None
A18	VCC	**IO62RSB1**	**IO94RSB1**	**IO85RSB2**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
A19	IO54RSB1	IO61RSB1	IO91RSB1	IO82RSB2	None	None	None	None	None	None
A20	IO52RSB1	IO58RSB1	IO85RSB1	IO76RSB2	None	None	None	None	None	None
A21	**IO49RSB1**	**GDB2/ IO55RSB1**	**IO79RSB1**	**IO70RSB2**	None	**Rule 2**	**Rule 2**	**None**	**None**	**Rule 3**
A22	**IO48RSB1**	**NC**	VCC	VCC	None	**Rule 4**	**Rule 4**	**Rule 6**	**Rule 6**	**Rule 1**

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *D pins are corner pins per the package specification.*
3. *"None" implies that the pins can be connected without any change.*

Table 11-5 **• Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P060 and A3P030
A23	**IO47RSB1**	**GDA2/IO54RSB1**	**GDB2/IO71RSB1**	**GDB2/IO62RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A24	TDI	TDI	TDI	TDI	None	None	None	None	None	None
A25	TRST	TRST	TRST	TRST	None	None	None	None	None	None
A26	IO44RSB0	**GDC1/IO48RSB0**	**GDC1/IO61RSB0**	**GDC1/IO58UDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A27	NC	VCC	VCC	VCC	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
A28	IO43RSB0	**IO47RSB0**	**IO60RSB0**	**IO54NDB1**	None	None	None	None	None	None
A29	IO42RSB0	**GCC2/IO46RSB0**	**GCC2/IO59RSB0**	**IO52NDB1**	None	None	**Rule 2**	None	**Rule 3**	**Rule 3**
A30	IO40RSB0	**GCA2/IO44RSB0**	**GCA2/IO57RSB0**	**GCA2/IO51PPB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A31	IO39RSB0	**GCA0/IO43RSB0**	**GCA0/IO56RSB0**	**GCA0/IO50NPB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A32	GDC0/IO36RSB0	GCB1/IO40RSB0	GCB1/IO53RSB0	GCB1/IO49PDB1	None	None	None	None	None	None
A33	**NC**	**IO36RSB0**	**IO49RSB0**	**IO47NSB1**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
A34	VCC	VCC	VCC	VCC	None	None	None	None	None	None
A35	IO34RSB0	IO31RSB0	IO44RSB0	IO41NPB1	None	None	None	None	None	None
A36	**IO31RSB0**	**GBA2/IO28RSB0**	**GBA2/IO41RSB0**	**GBA2/IO41PPB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A37	**IO26RSB0**	**GBB1/IO25RSB0**	**GBB1/IO38RSB0**	**GBB1/IO38RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A38	**IO23RSB0**	**GBC0/IO22RSB0**	**GBC0/IO35RSB0**	**GBC0/IO35RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A39	**NC**	VCCIB0	VCCIB0	VCCIB0	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
A40	IO22RSB0	IO21RSB0	IO28RSB0	IO28RSB0	None	None	None	None	None	None
A41	IO20RSB0	IO18RSB0	IO22RSB0	IO22RSB0	None	None	None	None	None	None
A42	IO18RSB0	IO15RSB0	IO18RSB0	IO18RSB0	None	None	None	None	None	None
A43	VCC	**IO14RSB0**	**IO14RSB0**	**IO14RSB0**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
A44	IO15RSB0	IO11RSB0	IO11RSB0	IO11RSB0	None	None	None	None	None	None
A45	IO12RSB0	**GAB1/IO08RSB0**	**IO07RSB0**	**IO07RSB0**	None	**Rule 2**	**Rule 2**	None	None	**Rule 3**

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *D pins are corner pins per the package specification.*
3. *"None" implies that the pins can be connected without any change.*

Table 11-5 **• Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P060 and A3P030
A46	IO10RSB0	**NC**	VCC	VCC	None	**Rule 4**	**Rule 4**	**Rule 6**	**Rule 6**	**Rule 1**
A47	IO09RSB0	**GAB0/IO07RSB0**	**GAC1/IO05RSB0**	**GAC1/IO05RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
A48	IO06RSB0	**IO04RSB0**	**GAB0/IO02RSB0**	**GAB0/IO02RSB0**	None	**Rule 3**	**Rule 3**	**Rule 3**	**Rule 3**	**None**
B1	IO02RSB1	IO01RSB1	IO68RSB1	IO118VDB3	None	None	None	None	None	None
B2	IO82RSB1	**GAC2/IO94RSB1**	**GAC2/IO131RSB1**	**GAC2/IO116UDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B3	GND	GND	GND	GND	None	None	None	None	None	None
B4	IO79RSB1	GFC0/IO88RSB1	GFC0/IO125RSB1	GFC0/IO110NDB3	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B5	**NC**	VCOMPLF	VCOMPLF	VCOMPLF	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
B6	GND	GND	GND	GND	None	None	None	None	None	None
B7	**IO74RSB1**	**GFB2/IO82RSB1**	**GFB2/IO119RSB1**	**GFB2/IO106PSB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B8	**NC**	**IO79RSB1**	**IO116RSB1**	**IO103PDB3**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
B9	GND	GND	GND	GND	None	None	None	None	None	None
B10	IO70RSB1	**GEB0/IO74RSB1**	**GEB0/IO109RSB1**	**GEB0/IO99NDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B11	IO67RSB1	**VMV1**	**VMV1**	**VMV3**	**Rule 6**	**None**	**Rule 5**	**Rule 6**	**Rule 6**	**Rule 6**
B12	IO64RSB1	**GEB2/IO70RSB1**	**GEB2/IO105RSB1**	**GEB2/IO96RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B13	IO61RSB1	IO67RSB1	IO101RSB1	IO92RSB2	None	None	None	None	None	None
B14	GND	GND	GND	GND	None	None	None	None	None	None
B15	IO58RSB1	**NC**	**IO98RSB1**	**IO89RSB2**	None	**Rule 1**	**Rule 1**	None	None	**Rule 1**
B16	IO56RSB1	**NC**	**IO95RSB1**	**IO86RSB2**	None	**Rule 1**	**Rule 1**	None	None	**Rule 1**
B17	GND	GND	GND	GND	None	None	None	None	None	None
B18	IO53RSB1	IO59RSB1	IO87RSB1	IO78RSB2	None	None	None	None	None	None
B19	IO50RSB1	**GDC2/IO56RSB1**	**IO81RSB1**	**IO72RSB2**	None	**Rule 2**	**Rule 2**	None	None	**Rule 3**
B20	GND	GND	GND	GND	None	None	None	None	None	None
B21	IO46RSB1	**GNDQ**	**GNDQ**	**GNDQ**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *D pins are corner pins per the package specification.*
3. *"None" implies that the pins can be connected without any change.*

Table 11-5 **• Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P060 and A3P030
B22	TMS	TMS	TMS	TMS	None	None	None	None	None	None
B23	TDO	TDO	TDO	TDO	None	None	None	None	None	None
B24	IO45RSB0	**GDC0/ IO49RSB0**	**GDC0/ IO62RSB0**	**GDC0/ IO58VDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B25	GND	**GND**	**GND**	**GND**	None	None	None	None	None	None
B26	NC	NC	NC	IO54PDB1	Rule 1	None	Rule 1	Rule 1	None	None
B27	IO41RSB0	**GCB2/ IO45RSB0**	**GCB2/ IO58RSB0**	**GCB2/ IO52PDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B28	GND	GND	GND	GND	None	None	None	None	None	None
B29	GDA0/ IO37RSB0	GCB0/ IO41RSB0	GCB0/ IO54RSB0	GCB0/ IO49NDB1	None	None	None	None	None	None
B30	NC	**GCC1/ IO38RSB0**	**GCC1/ IO51RSB0**	**GCC1/ IO48PDB1**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
B31	GND	GND	GND	GND	None	None	None	None	None	None
B32	IO33RSB0	**GBB2/ IO30RSB0**	**GBB2/ IO43RSB0**	**GBB2/ IO42PDB1**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B33	IO30RSB0	**VMV0**	**VMV0**	**VMV1**	**Rule 5**	**None**	**Rule 5**	**Rule 6**	**Rule 6**	**Rule 6**
B34	IO27RSB0	**GBA0/ IO26RSB0**	**GBA0/ IO39RSB0**	**GBA0/ IO39RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B35	IO24RSB0	**GBC1/ IO23RSB0**	**GBC1/ IO36RSB0**	**GBC1/ IO36RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B36	GND	GND	GND	GND	None	None	None	None	None	None
B37	IO21RSB0	IO20RSB0	IO26RSB0	IO26RSB0	None	None	None	None	None	None
B38	IO19RSB0	IO17RSB0	IO21RSB0	IO21RSB0	None	None	None	None	None	None
B39	GND	GND	GND	GND	None	None	None	None	None	None
B40	IO16RSB0	IO12RSB0	IO13RSB0	IO13RSB0	None	None	None	None	None	None
B41	IO13RSB0	**GAC0/ IO09RSB0**	**IO08RSB0**	**IO08RSB0**	None	**Rule 2**	**Rule 2**	None	None	**Rule 3**
B42	GND	GND	GND	GND	None	None	None	None	None	None
B43	IO08RSB0	**GAA1/ IO06RSB0**	**GAC0/ IO04RSB0**	**GAC0/ IO04RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
B44	IO05RSB0	**GNDQ**	**GNDQ**	**GNDQ**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *D pins are corner pins per the package specification.*
3. *"None" implies that the pins can be connected without any change.*

***Table 11-5* • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P060 and A3P030
C1	IO03RSB1	**GAA2/ IO02RSB1**	**GAA2/ IO67RSB1**	**GAA2/ IO118UDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C2	IO00RSB1	IO95RSB1	IO132RSB1	IO116VDB3	None	None	None	None	None	None
C3	NC	VCC	VCC	VCC	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
C4	IO78RSB1	GFB1/ IO87RSB1	GFB1/ IO124RSB1	GFB1/ IO109PPB3	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C5	GEA0/ IO76RSB1	**GFA0/ IO85RSB1**	**GFA0/ IO122RSB1**	**GFA0/ IO108NPB3**	None	None	None	None	None	None
C6	NC	**GFA2/ IO83RSB1**	**GFA2/ IO120RSB1**	**GFA2/ IO107PSB3**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
C7	NC	**IO80RSB1**	**IO117RSB1**	**IO105NPB3**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
C8	VCCIB1	VCCIB1	VCCIB1	VCCIB3	**Rule 5**	None	**Rule 5**	**Rule 5**	**Rule 5**	**Rule 5**
C9	IO69RSB1	**GEA1/ IO73RSB1**	**GEA1/ IO108RSB1**	**GEB1/ IO99PDB3**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C10	IO66RSB1	**GNDQ**	**GNDQ**	**GNDQ**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
C11	IO65RSB1	**GEA2/ IO71RSB1**	**GEA2/ IO106RSB1**	**GEA2/ IO97RSB2**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C12	IO62RSB1	IO68RSB1	IO103RSB1	IO94RSB2	None	None	None	None	None	None
C13	NC	VCCIB1	VCCIB1	VCCIB2	**Rule 5**	None	**Rule 5**	**Rule 4**	**Rule 4**	**Rule 4**
C14	NC	**NC**	**IO97RSB1**	**IO88RSB2**	None	**Rule 1**	**Rule 1**	**Rule 1**	**Rule 1**	**None**
C15	IO55RSB1	**NC**	**IO93RSB1**	**IO84RSB2**	None	**Rule 1**	**Rule 1**	None	None	**Rule 1**
C16	V_{CCI}B1	**IO60RSB1**	**IO89RSB1**	**IO80RSB2**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
C17	IO51RSB1	IO57RSB1	IO83RSB1	IO74RSB2	None	None	None	None	None	None
C18	NC	**NC**	VCCIB1	VCCIB2	**Rule 5**	**Rule 4**	**Rule 4**	**Rule 4**	**Rule 4**	**Rule 4**
C19	TCK	TCK	TCK	TCK	None	None	None	None	None	None
C20	NC	**VMV1**	**VMV1**	**VMV2**	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
C21	VPUMP	VPUMP	VPUMP	VPUMP	None	None	None	None	None	None
C22	VJTAG	VJTAG	VJTAG	VJTAG	None	None	None	None	None	None
C23	NC	VCCIB0	VCCIB0	VCCIB1	**Rule 5**	None	**Rule 5**	**Rule 4**	**Rule 4**	**Rule 4**
C24	NC	**NC**	**NC**	**IO53NSB1**	**Rule 1**	None	**Rule 1**	**Rule 1**	None	None
C25	NC	**NC**	**NC**	**IO51NPB1**	**Rule 1**	None	**Rule 1**	**Rule 1**	None	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *D pins are corner pins per the package specification.*
3. *"None" implies that the pins can be connected without any change.*

Table 11-5 • **Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging (continued)**

Pin No.	A3P030 Function	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P030	Migration Rule between A3P125 and A3P030	Migration Rule between A3P060 and A3P030
C26	GDB0/IO38RSB0	GCA1/IO42RSB0	GCA1/IO55RSB0	GCA1/IO50PPB1	None	None	None	None	None	None
C27	NC	**GCC0/IO39RSB0**	**GCC0/IO52RSB0**	**GCC0/IO48NDB1**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
C28	VCCIB0	VCCIB0	VCCIB0	VCCIB1	**Rule 5**	**None**	**Rule 5**	**Rule 5**	**Rule 5**	**None**
C29	IO32RSB0	IO29RSB0	IO42RSB0	IO42NDB1	None	None	None	None	None	None
C30	IO29RSB0	**GNDQ**	**GNDQ**	**GNDQ**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
C31	IO28RSB0	**GBA1/IO27RSB0**	**GBA1/IO40RSB0**	**GBA1/IO40RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C32	IO25RSB0	**GBB0/IO24RSB0**	**GBB0/IO37RSB0**	**GBB0/IO37RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C33	NC	VCC	VCC	VCC	None	None	None	**Rule 4**	**Rule 4**	**Rule 4**
C34	NC	**IO19RSB0**	**IO24RSB0**	**IO24RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C35	VCCIB0	**IO16RSB0**	**IO19RSB0**	**IO19RSB0**	None	None	None	**Rule 1**	**Rule 1**	**Rule 1**
C36	IO17RSB0	IO13RSB0	IO16RSB0	IO16RSB0	None	None	None	None	None	None
C37	IO14RSB0	**GAC1/IO10RSB0**	**IO10RSB0**	**IO10RSB0**	None	**Rule 2**	**Rule 2**	None	None	**Rule 3**
C38	IO11RSB0	**NC**	VCCIB0	VCCIB0	**Rule 5**	**Rule 4**	**Rule 4**	**Rule 6**	**Rule 6**	**Rule 1**
C39	IO07RSB0	**GAA0/IO05RSB0**	**GAB1/IO03RSB0**	**GAB1/IO03RSB0**	None	None	None	**Rule 3**	**Rule 3**	**Rule 3**
C40	IO04RSB0	**VMV0**	**VMV0**	**VMV0**	None	None	None	**Rule 6**	**Rule 6**	**Rule 6**
D1	GND	GND	GND	GND	None	None	None	None	None	None
D2	GND	GND	GND	GND	None	None	None	None	None	None
D3	GND	GND	GND	GND	None	None	None	None	None	None
D4	GND	GND	GND	GND	None	None	None	None	None	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *D pins are corner pins per the package specification.*
3. *"None" implies that the pins can be connected without any change.*

TQ144 Package

Table 11-6 • **Pin Compatibility and Migration Table for ProASIC3 A3P060 and A3P125 with TQ144 Packaging**

Pin Number	A3P060 Function	A3P125 Function	Migration Rule between A3P125 and A3P060
1	GAA2/IO51RSB1	GAA2/IO67RSB1	None
2	IO52RSB1	IO68RSB1	None
3	GAB2/IO53RSB1	GAB2/IO69RSB1	None
4	IO95RSB1	IO132RSB1	None
5	GAC2/IO94RSB1	GAC2/IO131RSB1	None
6	IO93RSB1	IO130RSB1	None
7	IO92RSB1	IO129RSB1	None
8	IO91RSB1	IO128RSB1	None
9	VCC	VCC	None
10	GND	GND	None
11	VCCIB1	VCCIB1	None
12	IO90RSB1	IO127RSB1	None
13	GFC1/IO89RSB1	GFC1/IO126RSB1	None
14	GFC0/IO88RSB1	GFC0/IO125RSB1	None
15	GFB1/IO87RSB1	GFB1/IO124RSB1	None
16	GFB0/IO86RSB1	GFB0/IO123RSB1	None
17	VCOMPLF	VCOMPLF	None
18	GFA0/IO85RSB1	GFA0/IO122RSB1	None
19	VCCPLF	VCCPLF	None
20	GFA1/IO84RSB1	GFA1/IO121RSB1	None
21	GFA2/IO83RSB1	GFA2/IO120RSB1	None
22	GFB2/IO82RSB1	GFB2/IO119RSB1	None
23	GFC2/IO81RSB1	GFC2/IO118RSB1	None
24	IO80RSB1	IO117RSB1	None
25	IO79RSB1	IO116RSB1	None
26	IO78RSB1	IO115RSB1	None
27	GND	GND	None
28	VCCIB1	VCCIB1	None
29	GEC1/IO77RSB1	GEC1/IO112RSB1	None
30	GEC0/IO76RSB1	GEC0/IO111RSB1	None
31	GEB1/IO75RSB1	GEB1/IO110RSB1	None
32	GEB0/IO74RSB1	GEB0/IO109RSB1	None
33	GEA1/IO73RSB1	GEA1/IO108RSB1	None
34	GEA0/IO72RSB1	GEA0/IO107RSB1	None
35	VMV1	VMV1	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-6 **• Pin Compatibility and Migration Table for ProASIC3 A3P060 and A3P125 with TQ144 Packaging (continued)**

Pin Number	A3P060 Function	A3P125 Function	Migration Rule between A3P125 and A3P060
36	GNDQ	GNDQ	None
37	NC	NC	None
38	GEA2/IO71RSB1	GEA2/IO106RSB1	None
39	GEB2/IO70RSB1	GEB2/IO105RSB1	None
40	GEC2/IO69RSB1	GEC2/IO104RSB1	None
41	IO68RSB1	IO103RSB1	None
42	IO67RSB1	IO102RSB1	None
43	IO66RSB1	IO101RSB1	None
44	IO65RSB1	IO100RSB1	None
45	VCC	VCC	None
46	GND	GND	None
47	VCCIB1	VCCIB1	None
48	NC	IO99RSB1	Rule 1
49	IO64RSB1	IO97RSB1	None
50	NC	IO95RSB1	Rule 1
51	IO63RSB1	IO93RSB1	None
52	NC	IO92RSB1	Rule 1
53	IO62RSB1	IO90RSB1	None
54	NC	IO88RSB1	Rule 1
55	IO61RSB1	IO86RSB1	None
56	NC	IO84RSB1	Rule 1
57	NC	IO83RSB1	Rule 1
58	IO60RSB1	IO82RSB1	None
59	IO59RSB1	IO81RSB1	None
60	IO58RSB1	IO80RSB1	None
61	IO57RSB1	IO79RSB1	None
62	NC	VCC	Rule 4
63	GND	GND	None
64	NC	VCCIB1	Rule 4
65	GDC2/IO56RSB1	GDC2/IO72RSB1	None
66	GDB2/IO55RSB1	GDB2/IO71RSB1	None
67	GDA2/IO54RSB1	GDA2/IO70RSB1	None
68	GNDQ	GNDQ	None
69	TCK	TCK	None
70	TDI	TDI	None
71	TMS	TMS	None
72	VMV1	VMV1	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-6 • Pin Compatibility and Migration Table for ProASIC3 A3P060 and A3P125 with TQ144 Packaging (continued)

Pin Number	A3P060 Function	A3P125 Function	Migration Rule between A3P125 and A3P060
73	VPUMP	VPUMP	None
74	NC	NC	None
75	TDO	TDO	None
76	TRST	TRST	None
77	VJTAG	VJTAG	None
78	GDA0/IO50RSB0	GDA0/IO66RSB0	None
79	GDB0/IO48RSB0	GDB0/IO64RSB0	None
80	GDB1/IO47RSB0	GDB1/IO63RSB0	None
81	VCCIB0	VCCIB0	None
82	GND	GND	None
83	IO44RSB0	IO60RSB0	None
84	GCC2/IO43RSB0	GCC2/IO59RSB0	None
85	GCB2/IO42RSB0	GCB2/IO58RSB0	None
86	GCA2/IO41RSB0	GCA2/IO57RSB0	None
87	GCA0/IO40RSB0	GCA0/IO56RSB0	None
88	GCA1/IO39RSB0	GCA1/IO55RSB0	None
89	GCB0/IO38RSB0	GCB0/IO54RSB0	None
90	GCB1/IO37RSB0	GCB1/IO53RSB0	None
91	GCC0/IO36RSB0	GCC0/IO52RSB0	None
92	GCC1/IO35RSB0	GCC1/IO51RSB0	None
93	IO34RSB0	IO50RSB0	None
94	IO33RSB0	IO49RSB0	None
95	NC	NC	None
96	NC	NC	None
97	NC	NC	None
98	VCCIB0	VCCIB0	None
99	GND	GND	None
100	VCC	VCC	None
101	IO30RSB0	IO47RSB0	None
102	GBC2/IO29RSB0	GBC2/IO45RSB0	None
103	IO28RSB0	IO44RSB0	None
104	GBB2/IO27RSB0	GBB2/IO43RSB0	None
105	IO26RSB0	IO42RSB0	None
106	GBA2/IO25RSB0	GBA2/IO41RSB0	None
107	VMV0	VMV0	None
108	GNDQ	GNDQ	None
109	NC	GBA1/IO40RSB0	Rule 1

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-6 • **Pin Compatibility and Migration Table for ProASIC3 A3P060 and A3P125 with TQ144 Packaging (continued)**

Pin Number	A3P060 Function	A3P125 Function	Migration Rule between A3P125 and A3P060
110	NC	GBA0/IO39RSB0	Rule 1
111	GBA1/IO24RSB0	GBB1/IO38RSB0	None
112	GBA0/IO23RSB0	GBB0/IO37RSB0	None
113	GBB1/IO22RSB0	GBC1/IO36RSB0	None
114	GBB0/IO21RSB0	GBC0/IO35RSB0	None
115	GBC1/IO20RSB0	IO34RSB0	Rule 2
116	GBC0/IO19RSB0	IO33RSB0	Rule 2
117	VCCIB0	VCCIB0	None
118	GND	GND	None
119	VCC	VCC	None
120	IO18RSB0	IO29RSB0	None
121	IO17RSB0	IO28RSB0	None
122	IO16RSB0	IO27RSB0	None
123	IO15RSB0	IO25RSB0	None
124	IO14RSB0	IO23RSB0	None
125	IO13RSB0	IO21RSB0	None
126	IO12RSB0	IO19RSB0	None
127	IO11RSB0	IO17RSB0	None
128	NC	IO16RSB0	Rule 1
129	IO10RSB0	IO14RSB0	None
130	IO09RSB0	IO12RSB0	None
131	IO08RSB0	IO10RSB0	None
132	GAC1/IO07RSB0	IO08RSB0	Rule 2
133	GAC0/IO06RSB0	IO06RSB0	Rule 2
134	NC	VCCIB0	Rule 4
135	GND	GND	None
136	NC	V_{CC}	Rule 4
137	GAB1/IO05RSB0	GAC1/IO05RSB0	None
138	GAB0/IO04RSB0	GAC0/IO04RSB0	None
139	GAA1/IO03RSB0	GAB1/IO03RSB0	None
140	GAA0/IO02RSB0	GAB0/IO02RSB0	None
141	IO01RSB0	GAA1/IO01RSB0	Rule 3
142	IO00RSB0	GAA0/IO00RSB0	Rule 3
143	GNDQ	GNDQ	None
144	VMV0	VMV0	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

PQ208 Package

***Table 11-7* • Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging**

Pin Number	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125
1	GND	GND	None
2	GAA2/IO67RSB1	GAA2/IO118UDB3	None
3	IO68RSB1	IO118VDB3	None
4	GAB2/IO69RSB1	GAB2/IO117UDB3	None
5	IO132RSB1	IO117VDB3	None
6	GAC2/IO131RSB1	GAC2/IO116UDB3	None
7	NC	IO116VDB3	Rule 1
8	NC	IO115UDB3	Rule 1
9	IO130RSB1	IO115VDB3	None
10	IO129RSB1	IO114UDB3	None
11	NC	IO114VDB3	Rule 1
12	IO128RSB1	IO113PDB3	None
13	NC	IO113NDB3	Rule 1
14	NC	IO112PDB3	Rule 1
15	NC	IO112NDB3	Rule 1
16	VCC	VCC	None
17	GND	GND	None
18	VCCIB1	VCCIB3	Rule 5
19	IO127RSB1	IO111PDB3	None
20	NC	IO111NDB3	Rule 1
21	GFC1/IO126RSB1	GFC1/IO110PDB3	None
22	GFC0/IO125RSB1	GFC0/IO110NDB3	None
23	GFB1/IO124RSB1	GFB1/IO109PDB3	None
24	GFB0/IO123RSB1	GFB0/IO109NDB3	None
25	VCOMPLF	VCOMPLF	None
26	GFA0/IO122RSB1	GFA0/IO108NPB3	None
27	VCCPLF	VCCPLF	None
28	GFA1/IO121RSB1	GFA1/IO108PPB3	None
29	GND	GND	None
30	GFA2/IO120RSB1	GFA2/IO107PDB3	None
31	NC	IO107NDB3	Rule 1
32	GFB2/IO119RSB1	GFB2/IO106PDB3	None
33	NC	IO106NDB3	Rule 1

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-7 • **Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging (continued)**

Pin Number	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125
34	GFC2/IO118RSB1	GFC2/IO105PDB3	None
35	IO117RSB1	IO105NDB3	None
36	NC	NC	None
37	IO116RSB1	IO104PDB3	None
38	IO115RSB1	IO104NDB3	None
39	NC	IO103PSB3	Rule 1
40	VCCIB1	VCCIB3	Rule 5
41	GND	GND	None
42	IO114RSB1	IO101PDB3	None
43	IO113RSB1	IO101NDB3	None
44	GEC1/IO112RSB1	GEC1/IO100PDB3	None
45	GEC0/IO111RSB1	GEC0/IO100NDB3	None
46	GEB1/IO110RSB1	GEB1/IO99PDB3	None
47	GEB0/IO109RSB1	GEB0/IO99NDB3	None
48	GEA1/IO108RSB1	GEA1/IO98PDB3	None
49	GEA0/IO107RSB1	GEA0/IO98NDB3	None
50	VMV1	VMV3	Rule 5
51	GNDQ	GNDQ	None
52	GND	GND	None
53	NC	NC	None
54	NC	NC	None
55	GEA2/IO106RSB1	GEA2/IO97RSB2	None
56	GEB2/IO105RSB1	GEB2/IO96RSB2	None
57	GEC2/IO104RSB1	GEC2/IO95RSB2	None
58	IO103RSB1	IO94RSB2	None
59	IO102RSB1	IO93RSB2	None
60	IO101RSB1	IO92RSB2	None
61	IO100RSB1	IO91RSB2	None
62	VCCIB1	VCCIB2	Rule 5
63	IO99RSB1	IO90RSB2	None
64	IO98RSB1	IO89RSB2	None
65	GND	GND	None
66	IO97RSB1	IO88RSB2	None
67	IO96RSB1	IO87RSB2	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

***Table 11-7* • Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging (continued)**

Pin Number	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125
68	IO95RSB1	IO86RSB2	None
69	IO94RSB1	IO85RSB2	None
70	IO93RSB1	IO84RSB2	None
71	VCC	VCC	None
72	VCCIB1	VCCIB2	Rule 5
73	IO92RSB1	IO83RSB2	None
74	IO91RSB1	IO82RSB2	None
75	IO90RSB1	IO81RSB2	None
76	IO89RSB1	IO80RSB2	None
77	IO88RSB1	IO79RSB2	None
78	IO87RSB1	IO78RSB2	None
79	IO86RSB1	IO77RSB2	None
80	IO85RSB1	IO76RSB2	None
81	GND	GND	None
82	IO84RSB1	IO75RSB2	None
83	IO83RSB1	IO74RSB2	None
84	IO82RSB1	IO73RSB2	None
85	IO81RSB1	IO72RSB2	None
86	IO80RSB1	IO71RSB2	None
87	IO79RSB1	IO70RSB2	None
88	VCC	VCC	None
89	VCCIB1	VCCIB2	Rule 5
90	IO78RSB1	IO69RSB2	None
91	IO77RSB1	IO68RSB2	None
92	IO76RSB1	IO67RSB2	None
93	IO75RSB1	IO66RSB2	None
94	IO74RSB1	IO65RSB2	None
95	IO73RSB1	IO64RSB2	None
96	GDC2/IO72RSB1	GDC2/IO63RSB2	None
97	GND	GND	None
98	GDB2/IO71RSB1	GDB2/IO62RSB2	None
99	GDA2/IO70RSB1	GDA2/IO61RSB2	None
100	GNDQ	GNDQ	None
101	TCK	TCK	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-7 • **Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging (continued)**

Pin Number	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125
102	TDI	TDI	None
103	TMS	TMS	None
104	VMV1	VMV2	Rule 5
105	GND	GND	None
106	VPUMP	VPUMP	None
107	NC	NC	None
108	TDO	TDO	None
109	TRST	TRST	None
110	VJTAG	VJTAG	None
111	GDA0/IO66RSB0	GDA0/IO60VDB1	None
112	GDA1/IO65RSB0	GDA1/IO60UDB1	None
113	GDB0/IO64RSB0	GDB0/IO59VDB1	None
114	GDB1/IO63RSB0	GDB1/IO59UDB1	None
115	GDC0/IO62RSB0	GDC0/IO58VDB1	None
116	GDC1/IO61RSB0	GDC1/IO58UDB1	None
117	NC	IO57VDB1	Rule 1
118	NC	IO57UDB1	Rule 1
119	NC	IO56NDB1	Rule 1
120	NC	IO56PDB1	Rule 1
121	NC	IO55RSB1	Rule 1
122	GND	GND	None
123	VCCIB0	VCCIB1	Rule 5
124	NC	NC	None
125	NC	NC	None
126	VCC	VCC	None
127	IO60RSB0	IO53NDB1	None
128	GCC2/IO59RSB0	GCC2/IO53PDB1	None
129	GCB2/IO58RSB0	GCB2/IO52PSB1	None
130	GND	GND	None
131	GCA2/IO57RSB0	GCA2/IO51PSB1	None
132	GCA0/IO56RSB0	GCA1/IO50PDB1	None
133	GCA1/IO55RSB0	GCA0/IO50NDB1	None
134	GCB0/IO54RSB0	GCB0/IO49NDB1	None
135	GCB1/IO53RSB0	GCB1/IO49PDB1	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-7 • **Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging (continued)**

Pin Number	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125
136	GCC0/IO52RSB0	GCC0/IO48NDB1	None
137	GCC1/IO51RSB0	GCC1/IO48PDB1	None
138	IO50RSB0	IO47NDB1	None
139	IO49RSB0	IO47PDB1	None
140	VCCIB0	VCCIB1	Rule 5
141	GND	GND	None
142	VCC	VCC	None
143	IO48RSB0	IO46RSB1	None
144	IO47RSB0	IO45NDB1	None
145	IO46RSB0	IO45PDB1	None
146	NC	IO44NDB1	Rule 1
147	NC	IO44PDB1	Rule 1
148	NC	IO43NDB1	Rule 1
149	GBC2/IO45RSB0	GBC2/IO43PDB1	None
150	IO44RSB0	IO42NDB1	None
151	GBB2/IO43RSB0	GBB2/IO42PDB1	None
152	IO42RSB0	IO41NDB1	None
153	GBA2/IO41RSB0	GBA2/IO41PDB1	None
154	VMV0	VMV1	Rule 5
155	GNDQ	GNDQ	None
156	GND	GND	None
157	NC	NC	None
158	GBA1/IO40RSB0	GBA1/IO40RSB0	None
159	GBA0/IO39RSB0	GBA0/IO39RSB0	None
160	GBB1/IO38RSB0	GBB1/IO38RSB0	None
161	GBB0/IO37RSB0	GBB0/IO37RSB0	None
162	GND	GND	None
163	GBC1/IO36RSB0	GBC1/IO36RSB0	None
164	GBC0/IO35RSB0	GBC0/IO35RSB0	None
165	IO34RSB0	IO34RSB0	None
166	IO33RSB0	IO33RSB0	None
167	IO32RSB0	IO32RSB0	None
168	IO31RSB0	IO31RSB0	None
169	IO30RSB0	IO30RSB0	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-7 **• Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging (continued)**

Pin Number	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125
170	VCCIB0	VCCIB0	None
171	VCC	VCC	None
172	IO29RSB0	IO29RSB0	None
173	IO28RSB0	IO28RSB0	None
174	IO27RSB0	IO27RSB0	None
175	IO26RSB0	IO26RSB0	None
176	IO25RSB0	IO25RSB0	None
177	IO24RSB0	IO24RSB0	None
178	GND	GND	None
179	IO23RSB0	IO23RSB0	None
180	IO22RSB0	IO22RSB0	None
181	IO21RSB0	IO21RSB0	None
182	IO20RSB0	IO20RSB0	None
183	IO19RSB0	IO19RSB0	None
184	IO18RSB0	IO18RSB0	None
185	IO17RSB0	IO17RSB0	None
186	VCCIB0	VCCIB0	None
187	VCC	VCC	None
188	IO16RSB0	IO16RSB0	None
189	IO15RSB0	IO15RSB0	None
190	IO14RSB0	IO14RSB0	None
191	IO13RSB0	IO13RSB0	None
192	IO12RSB0	IO12RSB0	None
193	IO11RSB0	IO11RSB0	None
194	IO10RSB0	IO10RSB0	None
195	GND	GND	None
196	IO09RSB0	IO09RSB0	None
197	IO08RSB0	IO08RSB0	None
198	IO07RSB0	IO07RSB0	None
199	IO06RSB0	IO06RSB0	None
200	VCCIB0	VCCIB0	None
201	GAC1/IO05RSB0	GAC1/IO05RSB0	None
202	GAC0/IO04RSB0	GAC0/IO04RSB0	None
203	GAB1/IO03RSB0	GAB1/IO03RSB0	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-7 • **Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging (continued)**

Pin Number	A3P125 Function	A3P250 Function	Migration Rule between A3P250 and A3P125
204	GAB0/IO02RSB0	GAB0/IO02RSB0	None
205	GAA1/IO01RSB0	GAA1/IO01RSB0	None
206	GAA0/IO00RSB0	GAA0/IO00RSB0	None
207	GNDQ	GNDQ	None
208	VMV0	VMV0	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

FG144 Package

Table 11-8 • **Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging**

Pin No.	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125
A1	GNDQ	GNDQ	GNDQ	None	None	None
A2	VMV0	VMV0	VMV0	None	None	None
A3	GAB0/IO04RSB0	GAB0/IO02RSB0	GAB0/IO02RSB0	None	None	None
A4	GAB1/IO05RSB0	GAB1/IO03RSB0	GAB1/IO03RSB0	None	None	None
A5	IO08RSB0	IO11RSB0	IO16RSB0	None	None	None
A6	GND	GND	GND	None	None	None
A7	IO11RSB0	IO18RSB0	IO29RSB0	None	None	None
A8	VCC	VCC	VCC	None	None	None
A9	IO16RSB0	IO25RSB0	IO33RSB0	None	None	None
A10	GBA0/IO23RSB0	GBA0/IO39RSB0	GBA0/IO39RSB0	None	None	None
A11	GBA1/IO24RSB0	GBA1/IO40RSB0	GBA1/IO40RSB0	None	None	None
A12	GNDQ	GNDQ	GNDQ	None	None	None
B1	GAB2/IO53RSB1	GAB2/IO69RSB1	GAB2/IO117UDB3	None	None	None
B2	GND	GND	GND	None	None	None
B3	GAA0/IO02RSB0	GAA0/IO00RSB0	GAA0/IO00RSB0	None	None	None
B4	GAA1/IO03RSB0	GAA1/IO01RSB0	GAA1/IO01RSB0	None	None	None
B5	IO00RSB0	IO08RSB0	IO14RSB0	None	None	None
B6	IO10RSB0	IO14RSB0	IO19RSB0	None	None	None
B7	IO12RSB0	IO19RSB0	IO22RSB0	None	None	None
B8	IO14RSB0	IO22RSB0	IO30RSB0	None	None	None
B9	GBB0/IO21RSB0	GBB0/IO37RSB0	GBB0/IO37RSB0	None	None	None
B10	GBB1/IO22RSB0	GBB1/IO38RSB0	GBB1/IO38RSB0	None	None	None
B11	GND	GND	GND	None	None	None
B12	VMV0	VMV0	VMV1	None	None	None
C1	IO95RSB1	IO132RSB1	IO117VDB3	None	None	None
C2	GFA2/IO83RSB1	GFA2/IO120RSB1	GFA2/IO107PPB3	None	None	None
C3	GAC2/IO94RSB1	GAC2/IO131RSB1	GAC2/IO116UDB3	None	None	None
C4	VCC	VCC	VCC	None	None	None
C5	IO01RSB0	IO10RSB0	IO12RSB0	None	None	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-8 • Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging (continued)

Pin No.	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125
C6	IO09RSB0	IO12RSB0	IO17RSB0	None	None	None
C7	IO13RSB0	IO21RSB0	IO24RSB0	None	None	None
C8	IO15RSB0	IO24RSB0	IO31RSB0	None	None	None
C9	IO17RSB0	IO27RSB0	IO34RSB0	None	None	None
C10	GBA2/IO25RSB0	GBA2/IO41RSB0	GBA2/IO41PDB1	None	None	None
C11	IO26RSB0	IO42RSB0	IO41NDB1	None	None	None
C12	GBC2/IO29RSB0	GBC2/IO45RSB0	GBC2/IO43PPB1	None	None	None
D1	IO91RSB1	IO128RSB1	IO112NDB3	None	None	None
D2	IO92RSB1	IO129RSB1	IO112PDB3	None	None	None
D3	IO93RSB1	IO130RSB1	IO116VDB3	None	None	None
D4	GAA2/IO51RSB1	GAA2/IO67RSB1	GAA2/IO118UPB3	None	None	None
D5	GAC0/IO06RSB0	GAC0/IO04RSB0	GAC0/IO04RSB0	None	None	None
D6	GAC1/IO07RSB0	GAC1/IO05RSB0	GAC1/IO05RSB0	None	None	None
D7	GBC0/IO19RSB0	GBC0/IO35RSB0	GBC0/IO35RSB0	None	None	None
D8	GBC1/IO20RSB0	GBC1/IO36RSB0	GBC1/IO36RSB0	None	None	None
D9	GBB2/IO27RSB0	GBB2/IO43RSB0	GBB2/IO42PDB1	None	None	None
D10	IO18RSB0	IO28RSB0	IO42NDB1	None	None	None
D11	IO28RSB0	IO44RSB0	IO43NPB1	None	None	None
D12	GCB1/IO37RSB0	GCB1/IO53RSB0	GCB1/IO49PPB1	None	None	None
E1	VCC	VCC	VCC	None	None	None
E2	GFC0/IO88RSB1	GFC0/IO125RSB1	GFC0/IO110NDB3	None	None	None
E3	GFC1/IO89RSB1	GFC1/IO126RSB1	GFC1/IO110PDB3	None	None	None
E4	VCCIB1	VCCIB1	VCCIB3	None	Rule 5	Rule 5
E5	IO52RSB1	IO68RSB1	IO118VPB3	None	None	None
E6	VCCIB0	VCCIB0	VCCIB0	None	None	None
E7	VCCIB0	VCCIB0	VCCIB0	None	None	None
E8	GCC1/IO35RSB0	GCC1/IO51RSB0	GCC1/IO48PDB1	None	None	None
E9	VCCIB0	VCCIB0	VCCIB1	None	Rule 5	Rule 5
E10	VCC	VCC	VCC	None	None	None
E11	GCA0/IO40RSB0	GCA0/IO56RSB0	GCA0/IO50NDB1	None	None	None
E12	IO30RSB0	IO46RSB0	IO51NDB1	None	None	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-8 • **Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging (continued)**

Pin No.	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125
F1	GFB0/IO86RSB1	GFB0/IO123RSB1	GFB0/IO109NPB3	None	None	None
F2	VCOMPLF	VCOMPLF	VCOMPLF	None	None	None
F3	GFB1/IO87RSB1	GFB1/IO124RSB1	GFB1/IO109PPB3	None	None	None
F4	IO90RSB1	IO127RSB1	IO107NPB3	None	None	None
F5	GND	GND	GND	None	None	None
F6	GND	GND	GND	None	None	None
F7	GND	GND	GND	None	None	None
F8	GCC0/IO36RSB0	GCC0/IO52RSB0	GCC0/IO48NDB1	None	None	None
F9	GCB0/IO38RSB0	GCB0/IO54RSB0	GCB0/IO49NPB1	None	None	None
F10	GND	GND	GND	None	None	None
F11	GCA1/IO39RSB0	GCA1/IO55RSB0	GCA1/IO50PDB1	None	None	None
F12	GCA2/IO41RSB0	GCA2/IO57RSB0	GCA2/IO51PDB1	None	None	None
G1	GFA1/IO84RSB1	GFA1/IO121RSB1	GFA1/IO108PPB3	None	None	None
G2	GND	GND	GND	None	None	None
G3	VCCPLF	VCCPLF	VCCPLF	None	None	None
G4	GFA0/IO85RSB1	GFA0/IO122RSB1	GFA0/IO108NPB3	None	None	None
G5	GND	GND	GND	None	None	None
G6	GND	GND	GND	None	None	None
G7	GND	GND	GND	None	None	None
G8	GDC1/IO45RSB0	GDC1/IO61RSB0	GDC1/IO58UPB1	None	None	None
G9	IO32RSB0	IO48RSB0	IO53NDB1	None	None	None
G10	GCC2/IO43RSB0	GCC2/IO59RSB0	GCC2/IO53PDB1	None	None	None
G11	IO31RSB0	IO47RSB0	IO52NDB1	None	None	None
G12	GCB2/IO42RSB0	GCB2/IO58RSB0	GCB2/IO52PDB1	None	None	None
H1	VCC	VCC	VCC	None	None	None
H2	GFB2/IO82RSB1	GFB2/IO119RSB1	GFB2/IO106PDB3	None	None	None
H3	GFC2/IO81RSB1	GFC2/IO118RSB1	GFC2/IO105PSB3	None	None	None
H4	GEC1/IO77RSB1	GEC1/IO112RSB1	GEC1/IO100PDB3	None	None	None
H5	VCC	VCC	VCC	None	None	None
H6	IO34RSB0	IO50RSB0	IO79RSB2	None	Rule 7	Rule 7
H7	IO44RSB0	IO60RSB0	IO65RSB2	None	Rule 7	Rule 7

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-8 • **Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging (continued)**

Pin No.	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125
H8	GDB2/IO55RSB1	GDB2/IO71RSB1	GDB2/IO62RSB2	None	None	None
H9	GDC0/IO46RSB0	GDC0/IO62RSB0	GDC0/IO58VPB1	None	None	None
H10	VCCIB0	VCCIB0	VCCIB1	None	Rule 5	Rule 5
H11	IO33RSB0	IO49RSB0	IO54PSB1	None	None	None
H12	VCC	VCC	VCC	None	None	None
J1	GEB1/IO75RSB1	GEB1/IO110RSB1	GEB1/IO99PDB3	None	None	None
J2	IO78RSB1	IO115RSB1	IO106NDB3	None	None	None
J3	VCCIB1	VCCIB1	VCCIB3	None	Rule 5	Rule 5
J4	GEC0/IO76RSB1	GEC0/IO111RSB1	GEC0/IO100NDB3	None	None	None
J5	IO79RSB1	IO116RSB1	IO88RSB2	None	None	None
J6	IO80RSB1	IO117RSB1	IO81RSB2	None	None	None
J7	VCC	VCC	VCC	None	None	None
J8	TCK	TCK	TCK	None	None	None
J9	GDA2/IO54RSB1	GDA2/IO70RSB1	GDA2/IO61RSB2	None	None	None
J10	TDO	TDO	TDO	None	None	None
J11	GDA1/IO49RSB0	GDA1/IO65RSB0	GDA1/IO60UDB1	None	None	None
J12	GDB1/IO47RSB0	GDB1/IO63RSB0	GDB1/IO59UDB1	None	None	None
K1	GEB0/IO74RSB1	GEB0/IO109RSB1	GEB0/IO99NDB3	None	None	None
K2	GEA1/IO73RSB1	GEA1/IO108RSB1	GEA1/IO98PDB3	None	None	None
K3	GEA0/IO72RSB1	GEA0/IO107RSB1	GEA0/IO98NDB3	None	None	None
K4	GEA2/IO71RSB1	GEA2/IO106RSB1	GEA2/IO97RSB2	None	None	None
K5	IO65RSB1	IO100RSB1	IO90RSB2	None	None	None
K6	IO64RSB1	IO98RSB1	IO84RSB2	None	None	None
K7	GND	GND	GND	None	None	None
K8	IO57RSB1	IO73RSB1	IO66RSB2	None	None	None
K9	GDC2/IO56RSB1	GDC2/IO72RSB1	GDC2/IO63RSB2	None	None	None
K10	GND	GND	GND	None	None	None
K11	GDA0/IO50RSB0	GDA0/IO66RSB0	GDA0/IO60VDB1	None	None	None
K12	GDB0/IO48RSB0	GDB0/IO64RSB0	GDB0/IO59VDB1	None	None	None
L1	GND	GND	GND	None	None	None
L2	VMV1	VMV1	VMV3	None	Rule 5	Rule 5

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Table 11-8 • **Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging (continued)**

Pin No.	A3P060 Function	A3P125 Function	A3P250 Function	Migration Rule between A3P125 and A3P060	Migration Rule between A3P250 and A3P060	Migration Rule between A3P250 and A3P125
L3	GEB2/IO70RSB1	GEB2/IO105RSB1	GEB2/IO96RSB2	None	None	None
L4	IO67RSB1	IO102RSB1	IO91RSB2	None	None	None
L5	VCCIB1	VCCIB1	VCCIB2	None	Rule 5	Rule 5
L6	IO62RSB1	IO95RSB1	IO82RSB2	None	None	None
L7	IO59RSB1	IO85RSB1	IO80RSB2	None	None	None
L8	IO58RSB1	IO74RSB1	IO72RSB2	None	None	None
L9	TMS	TMS	TMS	None	None	None
L10	VJTAG	VJTAG	VJTAG	None	None	None
L11	VMV1	VMV1	VMV2	None	Rule 5	Rule 5
L12	TRST	TRST	TRST	None	None	None
M1	GNDQ	GNDQ	GNDQ	None	None	None
M2	GEC2/IO69RSB1	GEC2/IO104RSB1	GEC2/IO95RSB2	None	None	None
M3	IO68RSB1	IO103RSB1	IO92RSB2	None	None	None
M4	IO66RSB1	IO101RSB1	IO89RSB2	None	None	None
M5	IO63RSB1	IO97RSB1	IO87RSB2	None	None	None
M6	IO61RSB1	IO94RSB1	IO85RSB2	None	None	None
M7	IO60RSB1	IO86RSB1	IO78RSB2	None	None	None
M8	NC	IO75RSB1	IO76RSB2	Rule 1	Rule 1	None
M9	TDI	TDI	TDI	None	None	None
M10	VCCIB1	VCCIB1	VCCIB2	None	Rule 5	Rule 5
M11	VPUMP	VPUMP	VPUMP	None	None	None
M12	GNDQ	GNDQ	GNDQ	None	None	None

Notes:

1. *See Table 11-3 on page 277 for the high-density/low-density pin combination guidelines.*
2. *"None" implies that the pins can be connected without any change.*

Conclusion

This application note describes the design migration among ProASIC3 family devices with an emphasis on package pin compatibility. Devices in the ProASIC3 family share numerous common architectural features. However, not all architectural features of family members are identical; use the datasheet to identify differences. Additionally, a key requirement is running functional simulation before and after the migration using Microsemi tools. Microsemi will continue to update this document with additional packages.

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	Rule 7 was added to Table 11-3 • Migration Rules from Higher-Density Device to Lower-Density Device and noted for pins H6 and H7 in Table 11-8 • Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging. This rule limits the device to a single voltage on this migration path (SAR 29451).	277, 301
June 2011	The A3P030 function values in Table 11-4 • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging were updated. Some of the values did not match the current pinout for VQ100 and have been corrected (SAR 29640).	279
	Incorrect references to Rules 7, 8, 9, and 10 in migration tables were removed and replaced with a reference to Rule 6. Affected pins are listed below (SAR 28348). Table 11-4 • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging: Pin 46 – Between A3P250 and A3P060, between A3P250 and A3P125 Pin 99 – Between A3P250 and A3P060, between A3P250 and A3P125 Pin 100 – Between A3P250 and A3P060, between A3P250 and A3P125 Table 11-5 • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging Pin B21 – Between A3P060 and A3P030 Pin B33 – Between A3P060 and A3P030	279 284
	The references to Rule 6 for pin B4 in Table 11-8 • Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging were changed to "None" (SARs 22561).	301
v1.0 (January 2008)	In Table 11-1 • Device Information, A3P250 device information was updated.	275
	In Table 11-2 • Common and Convertible I/Os, I/O counts for the VQ100 package were updated.	276
	In Table 11-4 • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging, the A3P030 pin function information was updated. Several rules changed throughout the table.	279

Date	Changes	Page
51900135-2/12.06	Table 11-1 • Device Information was updated to include A3P030 device information.	275
	Table 11-2 • Common and Convertible I/Os was updated.	276
	The "Migration and Implementation Methodologies" section was updated.	276
	Table 11-3 • Migration Rules from Higher-Density Device to Lower-Density Device was updated.	277
	In Table 11-4 • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with VQ100 Packaging, the A3P030 pin information was updated. Several rules changed throughout the table.	279
	In Table 11-5 • Pin Compatibility and Migration Table for A3P030, A3P060, A3P125, and A3P250 with QFN132 Packaging, all pin data was updated. Several rules changed throughout the table.	284
	In Table 11-7 • Pin Compatibility and Migration Table for ProASIC3 A3P125 and A3P250 with PQ208 Packaging, the rules were updated for the following pins: 50, 104, and 154.	294
	In Table 11-8 • Pin Compatibility and Migration Table for ProASIC3 A3P060, A3P125, and A3P250 with FG144 Packaging, the rules were updated for the following pins: L2 and L11.	301
51900135-1/12.06	QN132 information was added to Table 11-1 • Device Information.	275
	QN132 information was added to Table 11-2 • Common and Convertible I/Os.	276
	Table 11-6 • Pin Compatibility and Migration Table for ProASIC3 A3P060 and A3P125 with TQ144 Packaging is new.	290

12 – Programming Flash Devices

Introduction

This document provides an overview of the various programming options available for the Microsemi flash families. The electronic version of this document includes active links to all programming resources, which are available at http://www.microsemi.com/soc/products/hardware/default.aspx. For Microsemi antifuse devices, refer to the *Programming Antifuse Devices* document.

Summary of Programming Support

FlashPro4 and FlashPro3 are high-performance in-system programming (ISP) tools targeted at the latest generation of low power flash devices offered by the SmartFusion,® Fusion, IGLOO,® and ProASIC®3 families, including ARM-enabled devices. FlashPro4 and FlashPro3 offer extremely high performance through the use of USB 2.0, are high-speed compliant for full use of the 480 Mbps bandwidth, and can program ProASIC3 devices in under 30 seconds. Powered exclusively via USB, FlashPro4 and FlashPro3 provide a VPUMP voltage of 3.3 V for programming these devices.

FlashPro4 replaced FlashPro3 in 2010. FlashPro4 supports SmartFusion, Fusion, ProASIC3,and IGLOO devices as well as future generation flash devices. FlashPro4 also adds 1.2 V programming for IGLOO nano V2 devices. FlashPro4 is compatible with FlashPro3; however it adds a programming mode (PROG_MODE) signal to the previously unused pin 4 of the JTAG connector. The PROG_MODE goes high during programming and can be used to turn on a 1.5 V external supply for those devices that require 1.5 V for programming. If both FlashPro3 and FlashPro4 programmers are used for programming the same boards, pin 4 of the JTAG connector must not be connected to anything on the board because FlashPro4 uses pin 4 for PROG_MODE.

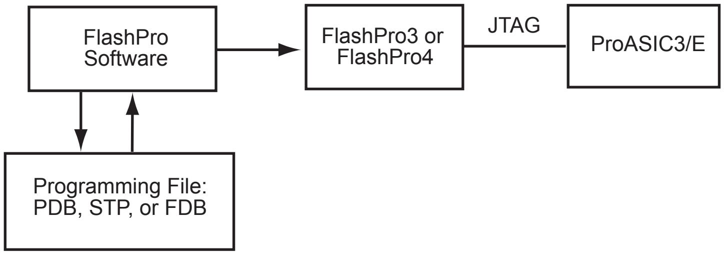


Figure 12-1 • **FlashPro Programming Setup**

Programming Support in Flash Devices

The flash FPGAs listed in Table 12-1 support flash in-system programming and the functions described in this document.

***Table 12-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution, supporting 1.2 V to 1.5 V core voltage with Flash*Freeze technology
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V core voltage with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
SmartFusion	SmartFusion	Mixed-signal FPGA integrating FPGA fabric, programmable microcontroller subsystem (MSS), including programmable analog and ARM® Cortex™-M3 hard processor and flash memory in a monolithic device
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device
ProASIC	ProASIC	First generation ProASIC devices
	ProASIC^{PLUS}	Second generation ProASIC devices

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 12-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 12-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

General Flash Programming Information

Programming Basics

When choosing a programming solution, there are a number of options available. This section provides a brief overview of those options. The next sections provide more detail on those options as they apply to Microsemi FPGAs.

Reprogrammable or One-Time-Programmable (OTP)

Depending on the technology chosen, devices may be reprogrammable or one-time-programmable. As the name implies, a reprogrammable device can be programmed many times. Generally, the contents of such a device will be completely overwritten when it is reprogrammed. All Microsemi flash devices are reprogrammable.

An OTP device is programmable one time only. Once programmed, no more changes can be made to the contents. Microsemi flash devices provide the option of disabling the reprogrammability for security purposes. This combines the convenience of reprogrammability during design verification with the security of an OTP technology for highly sensitive designs.

Device Programmer or In-System Programming

There are two fundamental ways to program an FPGA: using a device programmer or, if the technology permits, using in-system programming. A device programmer is a piece of equipment in a lab or on the production floor that is used for programming FPGA devices. The devices are placed into a socket mounted in a programming adapter module, and the appropriate electrical interface is applied. The programmed device can then be placed on the board. A typical programmer, used during development, programs a single device at a time and is referred to as a single-site engineering programmer.

With ISP, the device is already mounted onto the system printed circuit board when programming occurs. Typically, ISD programming is performed via a JTAG interface on the FPGA. The JTAG pins can be controlled either by an on-board resource, such as a microprocessor, or by an off-board programmer through a header connection. Once mounted, it can be programmed repeatedly and erased. If the application requires it, the system can be designed to reprogram itself using a microprocessor, without the use of any external programmer.

If multiple devices need to be programmed with the same program, various multi-site programming hardware is available in order to program many devices in parallel. Microsemi In House Programming is also available for this purpose.

Programming Features for Microsemi Devices

Flash Devices

The flash devices supplied by Microsemi are reprogrammable by either a generic device programmer or ISP. Microsemi supports ISP using JTAG, which is supported by the FlashPro4 and FlashPro3, FlashPro Lite, Silicon Sculptor 3, and Silicon Sculptor II programmers.

Levels of ISP support vary depending on the device chosen:

- All SmartFusion, Fusion, IGLOO, and ProASIC3 devices support ISP.
- IGLOO, IGLOOe, IGLOO nano V5, and IGLOO PLUS devices can be programmed in-system when the device is using a 1.5 V supply voltage to the FPGA core.
- IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only) or 1.5 V. IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V.

Types of Programming for Flash Devices

The number of devices to be programmed will influence the optimal programming methodology. Those available are listed below:

- In-system programming
  - Using a programmer
  - Using a microprocessor or microcontroller
- Device programmers
  - Single-site programmers
  - Multi-site programmers, batch programmers, or gang programmers
  - Automated production (robotic) programmers
- Volume programming services
  - Microsemi in-house programming
  - Programming centers

In-System Programming

Device Type Supported: Flash

ISP refers to programming the FPGA after it has been mounted on the system printed circuit board. The FPGA may be preprogrammed and later reprogrammed using ISP.

The advantage of using ISP is the ability to update the FPGA design many times without any changes to the board. This eliminates the requirement of using a socket for the FPGA, saving cost and improving reliability. It also reduces programming hardware expenses, as the ISP methodology is die-/package-independent.

There are two methods of in-system programming: external and internal.

- Programmer ISP—Refer to the "In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" section on page 349 for more information.

  Using an external programmer and a cable, the device can be programmed through a header on the system board. In Microsemi SoC Products Group documentation, this is referred to as external ISP. Microsemi provides FlashPro4, FlashPro3, FlashPro Lite, or Silicon Sculptor 3 to perform external ISP. Note that Silicon Sculptor II and Silicon Sculptor 3 can only provide ISP for ProASIC and ProASIC^{PLUS}® families, not for SmartFusion, Fusion, IGLOO, or ProASIC3. Silicon Sculptor II and Silicon Sculptor 3 can be used for programming ProASIC and ProASIC^{PLUS} devices by using an adapter module (part number SMPA-ISP-ACTEL-3).

  - Advantages: Allows local control of programming and data files for maximum security. The programming algorithms and hardware are available from Microsemi. The only hardware required on the board is a programming header.
  - Limitations: A negligible board space requirement for the programming header and JTAG signal routing
- Microprocessor ISP—Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

  Using a microprocessor and an external or internal memory, you can store the program in memory and use the microprocessor to perform the programming. In Microsemi documentation, this is referred to as internal ISP. Both the code for the programming algorithm and the FPGA programming file must be stored in memory on the board. Programming voltages must also be generated on the board.

  - Advantages: The programming code is stored in the system memory. An external programmer is not required during programming.
  - Limitations: This is the approach that requires the most design work, since some way of getting and/or storing the data is needed; a system interface to the device must be designed; and the low-level API to the programming firmware must be written and linked into the code provided by Microsemi. While there are benefits to this methodology, serious thought and planning should go into the decision.

Device Programmers

Single Device Programmer

Single device programmers are used to program a device before it is mounted on the system board.

The advantage of using device programmers is that no programming hardware is required on the system board. Therefore, no additional components or board space are required.

Adapter modules are purchased with single device programmers to support the FPGA packages used. The FPGA is placed in the adapter module and the programming software is run from a PC. Microsemi supplies the programming software for all of the Microsemi programmers. The software allows for the selection of the correct die/package and programming files. It will then program and verify the device.

- Single-site programmers

  A single-site programmer programs one device at a time. Microsemi offers Silicon Sculptor 3, built by BP Microsystems, as a single-site programmer. Silicon Sculptor 3 and associated software are available only from Microsemi.

  – Advantages: Lower cost than multi-site programmers. No additional overhead for programming on the system board. Allows local control of programming and data files for maximum security. Allows on-demand programming on-site.

  – Limitations: Only programs one device at a time.

- Multi-site programmers

Often referred to as batch or gang programmers, multi-site programmers can program multiple devices at the same time using the same programming file. This is often used for large volume programming and by programming houses. The sites often have independent processors and memory enabling the sites to operate concurrently, meaning each site may start programming the same file independently. This enables the operator to change one device while the other sites continue programming, which increases throughput. Multiple adapter modules for the same package are required when using a multi-site programmer. Silicon Sculptor I, II, and 3 programmers can be cascaded to program multiple devices in a chain. Multi-site programmers, such as the BP2610 and BP2710, can also be purchased from BP Microsystems. When using BP Microsystems multi-site programmers, users must use programming adapter modules available only from Microsemi. Visit the Microsemi SoC Products Group website to view the part numbers of the desired adapter module:

http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx.

Also when using BP Microsystems programmers, customers must use Microsemi programming software to ensure the best programming result will occur.

  – Advantages: Provides the capability of programming multiple devices at the same time. No additional overhead for programming on the system board. Allows local control of programming and data files for maximum security.

  – Limitations: More expensive than a single-site programmer

- Automated production (robotic) programmers

Automated production programmers are based on multi-site programmers. They consist of a large input tray holding multiple parts and a robotic arm to select and place parts into appropriate programming sockets automatically. When the programming of the parts is complete, the parts are removed and placed in a finished tray. The automated programmers are often used in volume programming houses to program parts for which the programming time is small. BP Microsystems part number BP4710, BP4610, BP3710 MK2, and BP3610 are available for this purpose. Auto programmers cannot be used to program RTAX-S devices.

Where an auto-programmer is used, the appropriate open-top adapter module from BP Microsystems must be used.

Volume Programming Services

Device Type Supported: Flash and Antifuse

Once the design is stable for applications with large production volumes, preprogrammed devices can be purchased. Table 12-2 describes the volume programming services.

Table 12-2 **• Volume Programming Services**

Programmer	Vendor	Availability
In-House Programming	Microsemi	Contact Microsemi Sales
Distributor Programming Centers	Memec Unique	Contact Distribution
Independent Programming Centers	Various	Contact Vendor

Advantages: As programming is outsourced, this solution is easier to implement than creating a substantial in-house programming capability. As programming houses specialize in large-volume programming, this is often the most cost-effective solution.

Limitations: There are some logistical issues with the use of a programming service provider, such as the transfer of programming files and the approval of First Articles. By definition, the programming file must be released to a third-party programming house. Nondisclosure agreements (NDAs) can be signed to help ensure data protection; however, for extremely security-conscious designs, this may not be an option.

- Microsemi In-House Programming

  When purchasing Microsemi devices in volume, IHP can be requested as part of the purchase. If this option is chosen, there is a small cost adder for each device programmed. Each device is marked with a special mark to distinguish it from blank parts. Programming files for the design will be sent to Microsemi. Sample parts with the design programmed, First Articles, will be returned for customer approval. Once approval of First Articles has been received, Microsemi will proceed with programming the remainder of the order. To request Microsemi IHP, contact your local Microsemi representative.
- Distributor Programming Centers

  If purchases are made through a distributor, many distributors will provide programming for their customers. Consult with your preferred distributor about this option.

Programming Solutions

Details for the available programmers can be found in the programmer user's guides listed in the "Related Documents" section on page 319.

All the programmers except FlashPro4, FlashPro3, FlashPro Lite, and FlashPro require adapter modules, which are designed to support device packages. All modules are listed on the Microsemi SoC Products Group website at http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx. They are not listed in this document, since this list is updated frequently with new package options and any upgrades required to improve programming yield or support new families.

***Table 12-3* • Programming Solutions**

Programmer	Vendor	ISP	Single Device	Multi-Device	Availability
FlashPro4	Microsemi	Only	Yes	Yes[1]	Available
FlashPro3	Microsemi	Only	Yes	Yes[1]	Available
FlashPro Lite[2]	Microsemi	Only	Yes	Yes[1]	Available
FlashPro	Microsemi	Only	Yes	Yes[1]	Discontinued
Silicon Sculptor 3	Microsemi	Yes[3]	Yes	Cascade option (up to two)	Available
Silicon Sculptor II	Microsemi	Yes[3]	Yes	Cascade option (up to two)	Available
Silicon Sculptor	Microsemi	Yes	Yes	Cascade option (up to four)	Discontinued
Sculptor 6X	Microsemi	No	Yes	Yes	Discontinued
BP MicroProgrammers	BP Microsystems	No	Yes	Yes	Contact BP Microsystems at www.bpmicro.com

Notes:

1. *Multiple devices can be connected in the same JTAG chain for programming.*
2. *If FlashPro Lite is used for programming, the programmer derives all of its power from the target pc board's VDD supply. The FlashPro Lite's VPP and VPN power supplies use the target pc board's VDD as a power source. The target pc board must supply power to both the VDDP and VDD power pins of the ProASIC<u>PLUS</u> device in addition to supplying VDD to the FlashPro Lite. The target pc board needs to provide at least 500 mA of current to the FlashPro Lite VDD connection for programming.*
3. *Silicon Sculptor II and Silicon Sculptor 3 can only provide ISP for ProASIC and ProASIC<u>PLUS</u> families, not for Fusion, IGLOO, or ProASIC3 devices.*

Programmer Ordering Codes

The products shown in Table 12-4 can be ordered through Microsemi sales and will be shipped directly from Microsemi. Products can also be ordered from Microsemi distributors, but will still be shipped directly from Microsemi. Table 12-4 includes ordering codes for the full kit, as well as codes for replacement items and any related hardware. Some additional products can be purchased from external suppliers for use with the programmers. Ordering codes for adapter modules used with Silicon Sculptor are available at http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx.

***Table 12-4* • Programming Ordering Codes**

Description	Vendor	Ordering Code	Comment
FlashPro4 ISP programmer	Microsemi	FLASHPRO 4	Uses a 2×5, RA male header connector
FlashPro Lite ISP programmer	Microsemi	FLASHPRO LITE	Supports small programming header or large header through header converter (not included)
Silicon Sculptor 3	Microsemi	SILICON-SCULPTOR 3	USB 2.0 high-speed production programmer
Silicon Sculptor II	Microsemi	SILICON-SCULPTOR II	Requires add-on adapter modules to support devices
Silicon Sculptor ISP module	Microsemi	SMPA-ISP-ACTEL-3-KIT	Ships with both large and small header support
ISP cable for small header	Microsemi	ISP-CABLE-S	Supplied with SMPA-ISP-ACTEL-3-KIT
ISP cable for large header	Microsemi	PA-ISP-CABLE	Supplied with SMPA-ISP-ACTEL-3-KIT

Programmer Device Support

Refer to www.microsemi.com/soc for the current information on programmer and device support.

Certified Programming Solutions

The Microsemi-certified programmers for flash devices are FlashPro4, FlashPro3, FlashPro Lite, FlashPro, Silicon Sculptor II, Silicon Sculptor 3, and any programmer that is built by BP Microsystems. All other programmers are considered noncertified programmers.

- FlashPro4, FlashPro3, FlashPro Lite, FlashPro

  The Microsemi family of FlashPro device programmers provides in-system programming in an easy-to-use, compact system that supports all flash families. Whether programming a board containing a single device or multiple devices connected in a chain, the Microsemi line of FlashPro programmers enables fast programming and reprogramming. Programming with the FlashPro series of programmers saves board space and money as it eliminates the need for sockets on the board. There are no built-in algorithms, so there is no delay between product release and programming support. The FlashPro programmer is no longer available.
- Silicon Sculptor 3, Silicon Sculptor II

  Silicon Sculptor 3 and Silicon Sculptor II are robust, compact, single-device programmers with standalone software for the PC. They are designed to enable concurrent programming of multiple units from the same PC with speeds equivalent to or faster than previous Microsemi programmers.
- Noncertified Programmers

  Microsemi does not test programming solutions from other vendors, and DOES NOT guarantee programming yield. Also, Microsemi will not perform any failure analysis on devices programmed on non-certified programmers. Please refer to the *Programming and Functional Failure Guidelines* document for more information.

- Programming Centers

  Microsemi programming hardware policy also applies to programming centers. Microsemi expects all programming centers to use certified programmers to program Microsemi devices. If a programming center uses noncertified programmers to program Microsemi devices, the "Noncertified Programmers" policy applies.

Important Programming Guidelines

Preprogramming Setup

Before programming, several steps are required to ensure an optimal programming yield.

Use Proper Handling and Electrostatic Discharge (ESD) Precautions

Microsemi FPGAs are sensitive electronic devices that are susceptible to damage from ESD and other types of mishandling. For more information about ESD, refer to the *Quality and Reliability Guide,* beginning with page 41.

Use the Latest Version of the Designer Software to Generate Your Programming File (recommended)

The files used to program Microsemi flash devices (*.bit, *.stp, *.pdb) contain important information about the switches that will be programmed in the FPGA. Find the latest version and corresponding release notes at http://www.microsemi.com/soc/download/software/designer/. Also, programming files must always be zipped during file transfer to avoid the possibility of file corruption.

Use the Latest Version of the Programming Software

The programming software is frequently updated to accommodate yield enhancements in FPGA manufacturing. These updates ensure maximum programming yield and minimum programming times. Before programming, always check the version of software being used to ensure it is the most recent. Depending on the programming software, refer to one of the following:

- FlashPro: http://www.microsemi.com/soc/download/program_debug/flashpro/
- Silicon Sculptor: http://www.microsemi.com/soc/download/program_debug/ss/

Use the Most Recent Adapter Module with Silicon Sculptor

Occasionally, Microsemi makes modifications to the adapter modules to improve programming yields and programming times. To identify the latest version of each module before programming, visit http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx.

Perform Routine Hardware Self-Diagnostic Test

- Adapter modules must be regularly cleaned. Adapter modules need to be inserted carefully into the programmer to make sure the DIN connectors (pins at the back side) are not damaged.
- FlashPro

  The self-test is only applicable when programming with FlashPro and FlashPro3 programmers. It is not supported with FlashPro4 or FlashPro Lite. To run the self-diagnostic test, follow the instructions given in the "Performing a Self-Test" section of http://www.microsemi.com/soc/documents/FlashPro_UG.pdf.
- Silicon Sculptor

  The self-diagnostic test verifies correct operation of the pin drivers, power supply, CPU, memory, and adapter module. This test should be performed with an adapter module installed and before every programming session. At minimum, the test must be executed every week. To perform self-diagnostic testing using the Silicon Sculptor software, perform the following steps, depending on the operating system:

  – DOS: From anywhere in the software, type **ALT + D**.

  – Windows: Click **Device** > choose **Actel Diagnostic** > select the **Test** tab > click **OK**.

  Silicon Sculptor programmers must be verified annually for calibration. Refer to the *Silicon Sculptor Verification of Calibration Work Instruction* document on the website.

Signal Integrity While Using ISP

For ISP of flash devices, customers are expected to follow the board-level guidelines provided on the Microsemi SoC Products Group website. These guidelines are discussed in the datasheets and application notes (refer to the “Related Documents” section of the datasheet for application note links). Customers are also expected to troubleshoot board-level signal integrity issues by measuring voltages and taking oscilloscope plots.

Programming Failure Allowances

Microsemi has strict policies regarding programming failure allowances. Please refer to *Programming and Functional Failure Guidelines* on the Microsemi SoC Products Group website for details.

Contacting the Customer Support Group

Highly skilled engineers staff the Customer Applications Center from 7:00 A.M. to 6:00 P.M., Pacific time, Monday through Friday. You can contact the center by one of the following methods:

Electronic Mail

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. Microsemi monitors the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and contact information for efficient processing of your request. The technical support email address is soc_tech@microsemi.com.

Telephone

Our Technical Support Hotline answers all calls. The center retrieves information, such as your name, company name, telephone number, and question. Once this is done, a case number is assigned. Then the center forwards the information to a queue where the first available applications engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific time, Monday through Friday.

The Customer Applications Center number is (800) 262-1060.

European customers can call +44 (0) 1256 305 600.

Related Documents

Below is a list of related documents, their location on the Microsemi SoC Products Group website, and a brief summary of each document.

Application Notes

Programming Antifuse Devices

http://www.microsemi.com/soc/documents/AntifuseProgram_AN.pdf

Implementation of Security in Actel's ProASIC and ProASIC^{PLUS} Flash-Based FPGAs

http://www.microsemi.com/soc/documents/Flash_Security_AN.pdf

User's Guides

FlashPro Programmers

FlashPro4,[1] FlashPro3, FlashPro Lite, and FlashPro[2]

http://www.microsemi.com/soc/products/hardware/program_debug/flashpro/default.aspx

FlashPro User's Guide

http://www.microsemi.com/soc/documents/FlashPro_UG.pdf

The FlashPro User's Guide includes hardware and software setup, self-test instructions, use instructions, and a troubleshooting / error message guide.

Silicon Sculptor 3 and Silicon Sculptor II

http://www.microsemi.com/soc/products/hardware/program_debug/ss/default.aspx

Other Documents

http://www.microsemi.com/soc/products/solutions/security/default.aspx#flashlock

The security resource center describes security in Microsemi Flash FPGAs.

Quality and Reliability Guide

http://www.microsemi.com/soc/documents/RelGuide.pdf

Programming and Functional Failure Guidelines

http://www.microsemi.com/soc/documents/FA_Policies_Guidelines_5-06-00002.pdf

[1] *FlashPro4 replaced FlashPro3 in Q1 2010.*

[2] *FlashPro is no longer available.*

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
July 2010	FlashPro4 is a replacement for FlashPro3 and has been added to this chapter. FlashPro is no longer available.	N/A
	The chapter was updated to include SmartFusion devices.	N/A
	The following were deleted: "Live at Power-Up (LAPU) or Boot PROM" section "Design Security" section Table 14-2 • Programming Features for Actel Devices and much of the text in the "Programming Features for Microsemi Devices" section "Programming Flash FPGAs" section "Return Material Authorization (RMA) Policies" section	N/A
	The "Device Programmers" section was revised.	313
	The Independent Programming Centers information was removed from the "Volume Programming Services" section.	314
	Table 12-3 • Programming Solutions was revised to add FlashPro4 and note that FlashPro is discontinued. A note was added for FlashPro Lite regarding power supply requirements.	315
	Most items were removed from Table 12-4 • Programming Ordering Codes, including FlashPro3 and FlashPro.	316
	The "Programmer Device Support" section was deleted and replaced with a reference to the Microsemi SoC Products Group website for the latest information.	316
	The "Certified Programming Solutions" section was revised to add FlashPro4 and remove Silicon Sculptor I and Silicon Sculptor 6X. Reference to *Programming and Functional Failure Guidelines* was added.	316
	The file type *.pdb was added to the "Use the Latest Version of the Designer Software to Generate Your Programming File (recommended)" section.	317
	Instructions on cleaning and careful insertion were added to the "Perform Routine Hardware Self-Diagnostic Test" section. Information was added regarding testing Silicon Sculptor programmers with an adapter module installed before every programming session verifying their calibration annually.	317
	The "Signal Integrity While Using ISP" section is new.	318
	The "Programming Failure Allowances" section was revised.	318

Date	Changes	Page
v1.3 (December 2008)	The "Programming Support in Flash Devices" section was updated to include IGLOO nano and ProASIC3 nano devices.	310
	The "Flash Devices" section was updated to include information for IGLOO nano devices. The following sentence was added: IGLOO PLUS devices can also be operated at any voltage between 1.2 V and 1.5 V; the Designer software allows 50 mV increments in the voltage.	311
	Table 12-4 · Programming Ordering Codes was updated to replace FP3-26PIN-ADAPTER with FP3-10PIN-ADAPTER-KIT.	316
	Table 14-6 · Programmer Device Support was updated to add IGLOO nano and ProASIC3 nano devices. AGL400 was added to the IGLOO portion of the table.	317
v1.2 (October 2008)	The "Programming Support in Flash Devices" section was revised to include new families and make the information more concise.	310
	Figure 12-1 · FlashPro Programming Setup and the "Programming Support in Flash Devices" section are new.	309, 310
	Table 14-6 · Programmer Device Support was updated to include A3PE600L with the other ProASIC3L devices, and the RT ProASIC3 family was added.	317
v1.1 (March 2008)	The "Flash Devices" section was updated to include the IGLOO PLUS family. The text, "Voltage switching is required in-system to switch from a 1.2 V core to 1.5 V core for programming," was revised to state, "Although the device can operate at 1.2 V core voltage, the device can only be reprogrammed when the core voltage is 1.5 V. Voltage switching is required in-system to switch from a 1.2 V supply (V_{CC}, V_{CCI}, and V_{JTAG}) to 1.5 V for programming."	311
	The ProASIC3L family was added to Table 14-6 · Programmer Device Support as a separate set of rows rather than combined with ProASIC3 and ProASIC3E devices. The IGLOO PLUS family was included, and AGL015 and A3P015 were added.	317

13 – Security in Low Power Flash Devices

Security in Programmable Logic

The need for security on FPGA programmable logic devices (PLDs) has never been greater than today. If the contents of the FPGA can be read by an external source, the intellectual property (IP) of the system is vulnerable to unauthorized copying. Fusion, IGLOO, and ProASIC3 devices contain state-of-the-art circuitry to make the flash-based devices secure during and after programming. Low power flash devices have a built-in 128-bit Advanced Encryption Standard (AES) decryption core (except for 30 k gate devices and smaller). The decryption core facilitates secure in-system programming (ISP) of the FPGA core array fabric, the FlashROM, and the Flash Memory Blocks (FBs) in Fusion devices. The FlashROM, Flash Blocks, and FPGA core fabric can be programmed independently of each other, allowing the FlashROM or Flash Blocks to be updated without the need for change to the FPGA core fabric.

Microsemi has incorporated the AES decryption core into the low power flash devices and has also included the Microsemi flash-based lock technology, FlashLock.® Together, they provide leading-edge security in a programmable logic device. Configuration data loaded into a device can be decrypted prior to being written to the FPGA core using the AES 128-bit block cipher standard. The AES encryption key is stored in on-chip, nonvolatile flash memory.

This document outlines the security features offered in low power flash devices, some applications and uses, as well as the different software settings for each application.

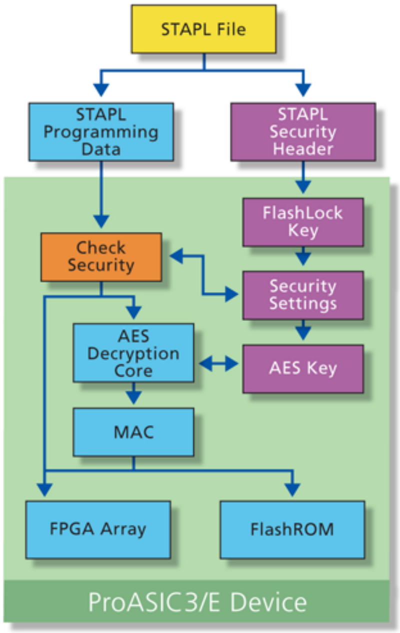


Figure 13-1 • **Overview on Security**

Security Support in Flash-Based Devices

The flash FPGAs listed in Table 13-1 support the security feature and the functions described in this document.

Table 13-1 **• Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM Cortex™-M1 soft processors, and flash memory into a monolithic device

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 13-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

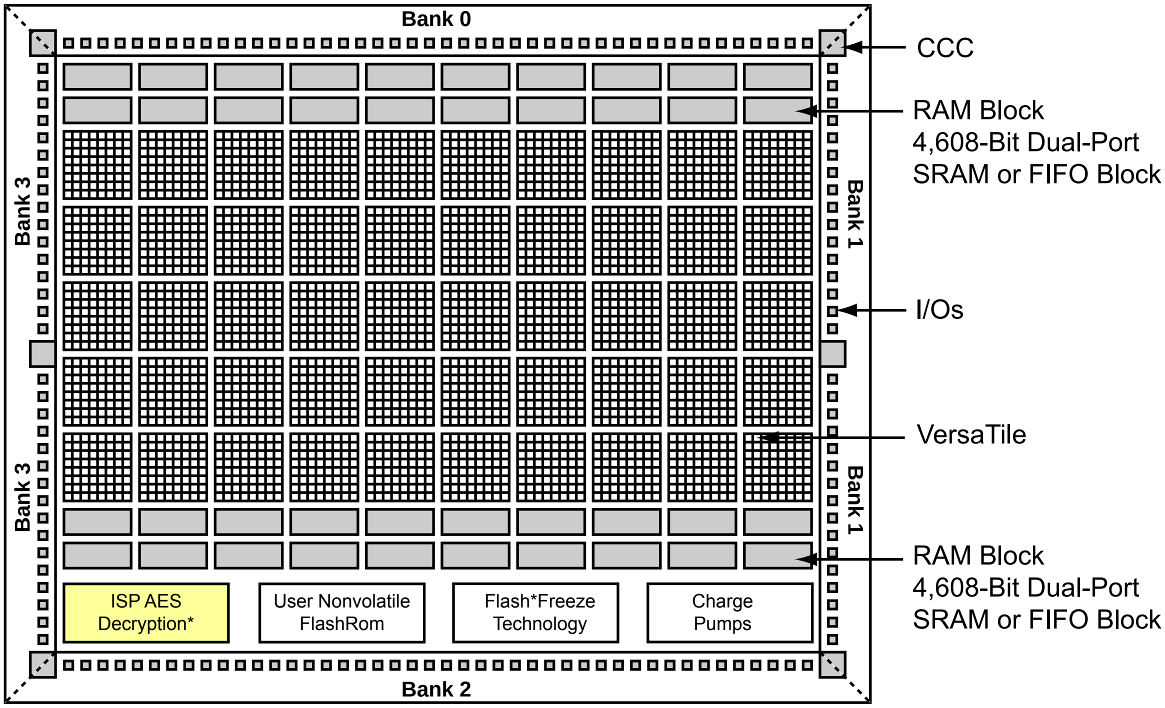
ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 13-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Security Architecture

Fusion, IGLOO, and ProASIC3 devices have been designed with the most comprehensive programming logic design security in the industry. In the architecture of these devices, security has been designed into the very fabric. The flash cells are located beneath seven metal layers, and the use of many device design and layout techniques makes invasive attacks difficult. Since device layers cannot be removed without disturbing the charge on the programmed (or erased) flash gates, devices cannot be easily deconstructed to decode the design. Low power flash devices are unique in being reprogrammable and having inherent resistance to both invasive and noninvasive attacks on valuable IP. Secure, remote ISP is now possible with AES encryption capability for the programming file during electronic transfer. Figure 13-2 shows a view of the AES decryption core inside an IGLOO device; Figure 13-3 on page 326 shows the AES decryption core inside a Fusion device. The AES core is used to decrypt the encrypted programming file when programming.



Note: **ISP AES Decryption is not supported by 30 k gate devices and smaller. For details of other architecture features by device, refer to the appropriate family datasheet.*

***Figure 13-2* • Block Representation of the AES Decryption Core in IGLOO and ProASIC3 Devices**

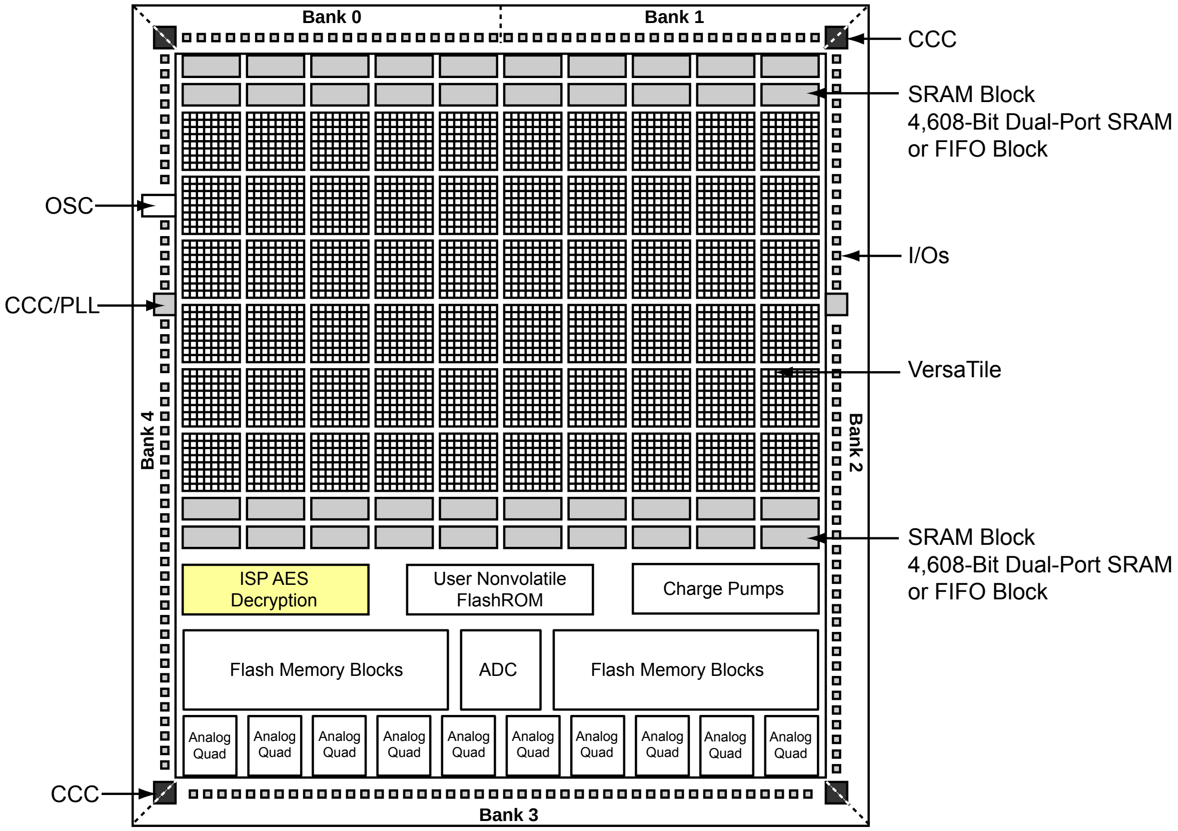


Figure 13-3 **• Block Representation of the AES Decryption Core in a Fusion AFS600 FPGA**

Security Features

IGLOO and ProASIC3 devices have two entities inside: FlashROM and the FPGA core fabric. Fusion devices contain three entities: FlashROM, FBs, and the FPGA core fabric. The parts can be programmed or updated independently with a STAPL programming file. The programming files can be AES-encrypted or plaintext. This allows maximum flexibility in providing security to the entire device. Refer to the "Programming Flash Devices" section on page 309 for information on the FlashROM structure.

Unlike SRAM-based FPGA devices, which require a separate boot PROM to store programming data, low power flash devices are nonvolatile, and the secured configuration data is stored in on-chip flash cells that are part of the FPGA fabric. Once programmed, this data is an inherent part of the FPGA array and does not need to be loaded at system power-up. SRAM-based FPGAs load the configuration bitstream upon power-up; therefore, the configuration is exposed and can be read easily.

The built-in FPGA core, FBs, and FlashROM support programming files encrypted with the 128-bit AES (FIPS-192) block ciphers. The AES key is stored in dedicated, on-chip flash memory and can be programmed before the device is shipped to other parties (allowing secure remote field updates).

Security in ARM-Enabled Low Power Flash Devices

There are slight differences between the regular flash devices and the ARM®-enabled flash devices, which have the M1 and M7 prefix.

The AES key is used by Microsemi and preprogrammed into the device to protect the ARM IP. As a result, the design is encrypted along with the ARM IP, according to the details below.

Cortex-M1 Device Security

Cortex-M1–enabled devices are shipped with the following security features:

- FPGA array enabled for AES-encrypted programming and verification
- FlashROM enabled for AES-encrypted Write and Verify
- Fusion Embedded Flash Memory enabled for AES-encrypted Write

AES Encryption of Programming Files

Low power flash devices employ AES as part of the security mechanism that prevents invasive and noninvasive attacks. The mechanism entails encrypting the programming file with AES encryption and then passing the programming file through the AES decryption core, which is embedded in the device. The file is decrypted there, and the device is successfully programmed. The AES master key is stored in on-chip nonvolatile memory (flash). The AES master key can be preloaded into parts in a secure programming environment (such as the Microsemi In-House Programming center), and then "blank" parts can be shipped to an untrusted programming or manufacturing center for final personalization with an AES-encrypted bitstream. Late-stage product changes or personalization can be implemented easily and securely by simply sending a STAPL file with AES-encrypted data. Secure remote field updates over public networks (such as the Internet) are possible by sending and programming a STAPL file with AES-encrypted data.

The AES key protects the programming data for file transfer into the device with 128-bit AES encryption. If AES encryption is used, the AES key is stored or preprogrammed into the device. To program, you must use an AES-encrypted file, and the encryption used on the file must match the encryption key already in the device.

The AES key is protected by a FlashLock security Pass Key that is also implemented in each device. The AES key is always protected by the FlashLock Key, and the AES-encrypted file does NOT contain the FlashLock Key. This FlashLock Pass Key technology is exclusive to the Microsemi flash-based device families. FlashLock Pass Key technology can also be implemented without the AES encryption option, providing a choice of different security levels.

In essence, security features can be categorized into the following three options:

- AES encryption with FlashLock Pass Key protection
- FlashLock protection only (no AES encryption)
- No protection

Each of the above options is explained in more detail in the following sections with application examples and software implementation options.

Advanced Encryption Standard

The 128-bit AES standard (FIPS-192) block cipher is the NIST (National Institute of Standards and Technology) replacement for DES (Data Encryption Standard FIPS46-2). AES has been designed to protect sensitive government information well into the 21st century. It replaces the aging DES, which NIST adopted in 1977 as a Federal Information Processing Standard used by federal agencies to protect sensitive, unclassified information. The 128-bit AES standard has 3.4×10^{38} possible 128-bit key variants, and it has been estimated that it would take 1,000 trillion years to crack 128-bit AES cipher text using exhaustive techniques. Keys are stored (securely) in low power flash devices in nonvolatile flash memory. All programming files sent to the device can be authenticated by the part prior to programming to ensure that bad programming data is not loaded into the part that may possibly damage it. All programming verification is performed on-chip, ensuring that the contents of low power flash devices remain secure.

Microsemi has implemented the 128-bit AES (Rijndael) algorithm in low power flash devices. With this key size, there are approximately 3.4×10^{38} possible 128-bit keys. DES has a 56-bit key size, which provides approximately 7.2×10^{16} possible keys. In their AES fact sheet, the National Institute of Standards and Technology uses the following hypothetical example to illustrate the theoretical security provided by AES. If one were to assume that a computing system existed that could recover a DES key in a second, it would take that same machine approximately 149 trillion years to crack a 128-bit AES key. NIST continues to make their point by stating the universe is believed to be less than 20 billion years old.[1]

The AES key is securely stored on-chip in dedicated low power flash device flash memory and cannot be read out. In the first step, the AES key is generated and programmed into the device (for example, at a secure or trusted programming site). The Microsemi Designer software tool provides AES key generation capability. After the key has been programmed into the device, the device will only correctly decrypt programming files that have been encrypted with the same key. If the individual programming file content is incorrect, a Message Authentication Control (MAC) mechanism inside the device will fail in authenticating the programming file. In other words, when an encrypted programming file is being loaded into a device that has a different programmed AES key, the MAC will prevent this incorrect data from being loaded, preventing possible device damage. See Figure 13-3 on page 326 and Figure 13-4 on page 328 for graphical representations of this process.

It is important to note that the user decides what level of protection will be implemented for the device. When AES protection is desired, the FlashLock Pass Key must be set. The AES key is a content protection mechanism, whereas the FlashLock Pass Key is a device protection mechanism. When the AES key is programmed into the device, the device still needs the Pass Key to protect the FPGA and FlashROM contents and the security settings, including the AES key. Using the FlashLock Pass Key prevents modification of the design contents by means of simply programming the device with a different AES key.

AES Decryption and MAC Authentication

Low power flash devices have a built-in 128-bit AES decryption core, which decrypts the encrypted programming file and performs a MAC check that authenticates the file prior to programming.

MAC authenticates the entire programming data stream. After AES decryption, the MAC checks the data to make sure it is valid programming data for the device. This can be done while the device is still operating. If the MAC validates the file, the device will be erased and programmed. If the MAC fails to validate, then the device will continue to operate uninterrupted.

This will ensure the following:

- Correct decryption of the encrypted programming file
- Prevention of erroneous or corrupted data being programmed during the programming file transfer
- Correct bitstream passed to the device for decryption

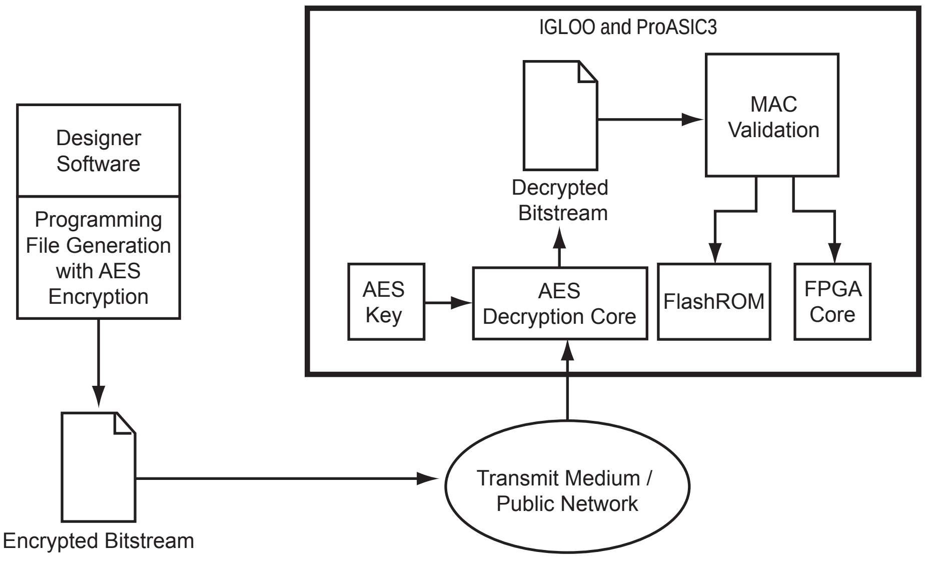


Figure 13-4 • **Example Application Scenario Using AES in IGLOO and ProASIC3 Devices**

[1] National Institute of Standards and Technology, "ADVANCED ENCRYPTION STANDARD (AES) Questions and Answers," 28 January 2002 (10 January 2005). See http://csrc.nist.gov/archive/aes/index1.html for more information.

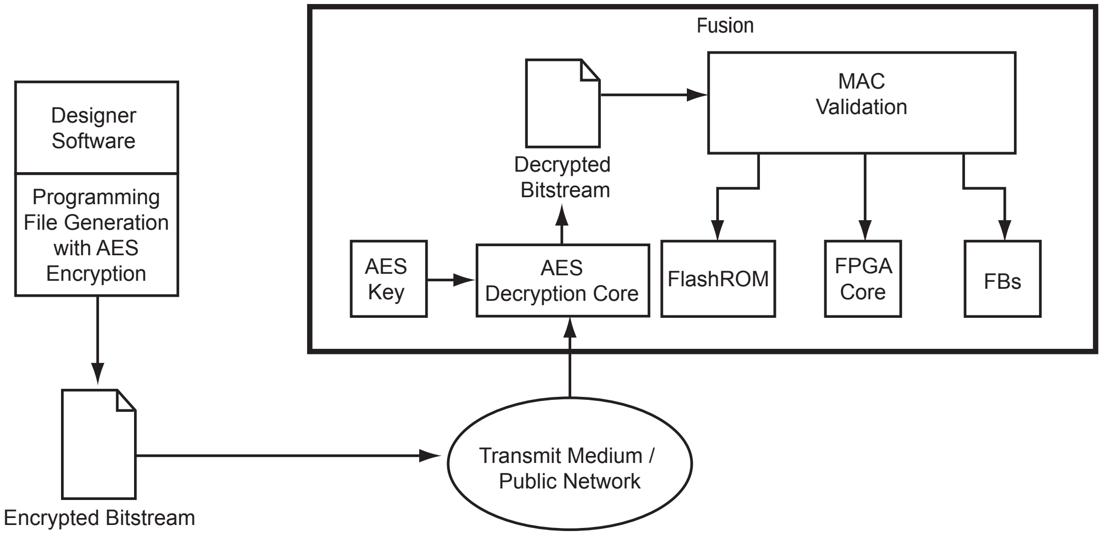


Figure 13-5 • **Example Application Scenario Using AES in Fusion Devices**

FlashLock

Additional Options for IGLOO and ProASIC3 Devices

The user also has the option of prohibiting Write operations to the FPGA array but allowing Verify operations on the FPGA array and/or Read operations on the FlashROM without the use of the FlashLock Pass Key. This option provides the user the freedom of verifying the FPGA array and/or reading the FlashROM contents after the device is programmed, without having to provide the FlashLock Pass Key. The user can incorporate AES encryption on the programming files to better enhance the level of security used.

Permanent Security Setting Options

In applications where a permanent lock is not desired, yet the security settings should not be modifiable, IGLOO and ProASIC3 devices can accommodate this requirement.

This application is particularly useful in cases where a device is located at a remote location and must be reprogrammed with a design or data update. Refer to the "Application 3: Nontrusted Environment—Field Updates/Upgrades" section on page 332 for further discussion and examples of how this can be achieved.

The user must be careful when considering the Permanent FlashLock or Permanent Security Settings option. Once the design is programmed with the permanent settings, it is not possible to reconfigure the security settings already employed on the device. Therefore, exercise careful consideration before programming permanent settings.

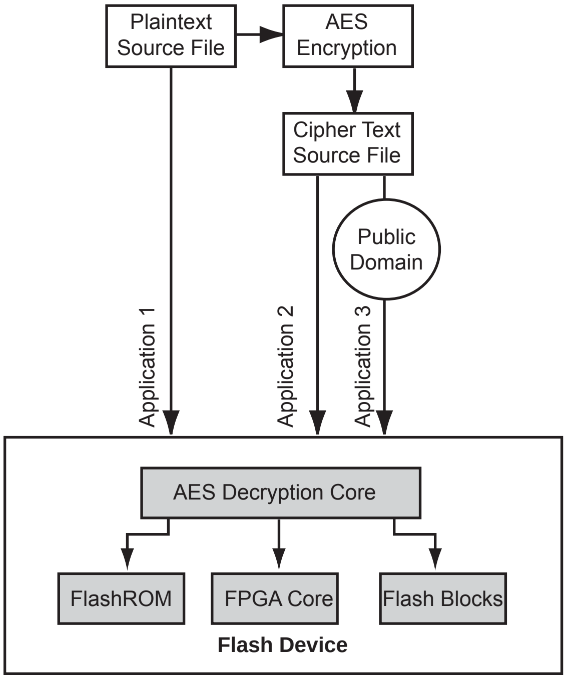
Permanent FlashLock

The purpose of the permanent lock feature is to provide the benefits of the highest level of security to IGLOO and ProASIC3 devices. If selected, the permanent FlashLock feature will create a permanent barrier, preventing any access to the contents of the device. This is achieved by permanently disabling Write and Verify access to the array, and Write and Read access to the FlashROM. After permanently locking the device, it has been effectively rendered one-time-programmable. This feature is useful if the intended applications do not require design or system updates to the device.

Security in Action

This section illustrates some applications of the security advantages of Microsemi's devices (Figure 13-6).

.



Note: Flash blocks are only used in Fusion devices

***Figure 13-6* • Security Options**

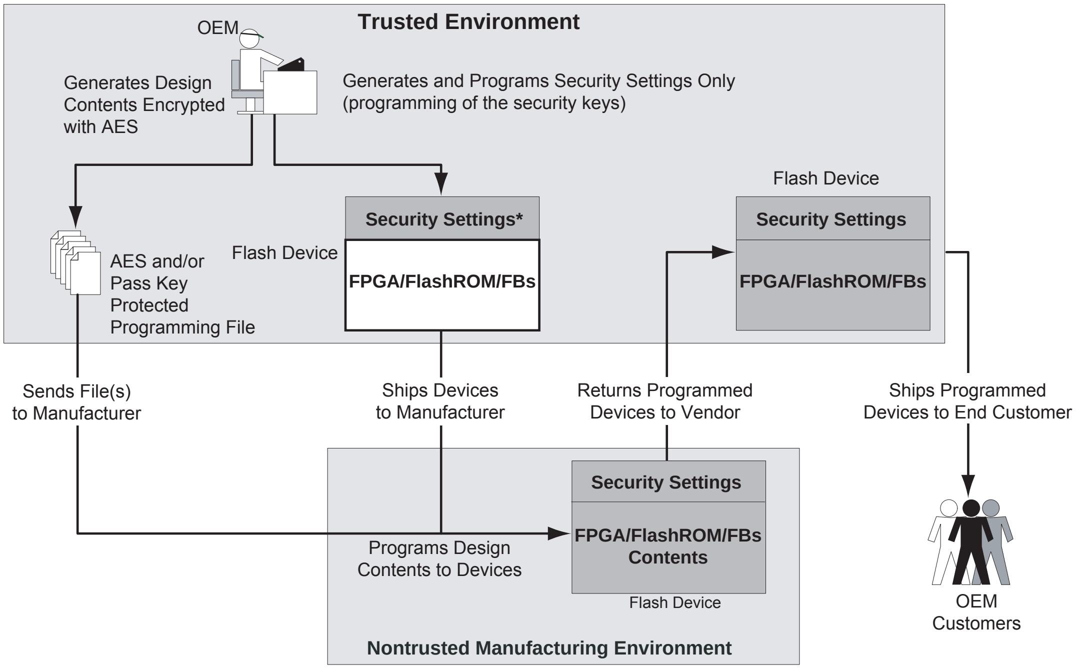
Application 1: Trusted Environment

As illustrated in Figure 13-7, this application allows the programming of devices at design locations where research and development take place. Therefore, encryption is not necessary and is optional to the user. This is often a secure way to protect the design, since the design program files are not sent elsewhere. In situations where production programming is not available at the design location, programming centers (such as Microsemi In-House Programming) provide a way of programming designs at an alternative, secure, and trusted location. In this scenario, the user generates a STAPL programming file from the Designer software in plaintext format, containing information on the entire design or the portion of the design to be programmed. The user can choose to employ the FlashLock Pass Key feature with the design. Once the design is programmed to unprogrammed devices, the design is protected by this FlashLock Pass Key. If no future programming is needed, the user can consider permanently securing the IGLOO and ProASIC3 device, as discussed in the "Permanent FlashLock" section on page 329.

Application 2: Nontrusted Environment—Unsecured Location

Often, programming of devices is not performed in the same location as actual design implementation, to reduce manufacturing cost. Overseas programming centers and contract manufacturers are examples of this scenario.

To achieve security in this case, the AES key and the FlashLock Pass Key can be initially programmed in-house (trusted environment). This is done by generating a programming file with only the security settings and no design contents. The design FPGA core, FlashROM, and (for Fusion) FB contents are generated in a separate programming file. This programming file must be set with the same AES key that was used to program to the device previously so the device will correctly decrypt this encrypted programming file. As a result, the encrypted design content programming file can be safely sent off-site to nontrusted programming locations for design programming. Figure 13-7 shows a more detailed flow for this application.


Trusted Environment
OEM
Generates Design Contents Encrypted with AES
Generates and Programs Security Settings Only (programming of the security keys)
Flash Device
Security Settings*
FPGA/FlashROM/FBs
AES and/or Pass Key Protected Programming File
Flash Device
Security Settings
FPGA/FlashROM/FBs
Sends File(s) to Manufacturer
Ships Devices to Manufacturer
Returns Programmed Devices to Vendor
Ships Programmed Devices to End Customer
Security Settings
FPGA/FlashROM/FBs Contents
Programs Design Contents to Devices
Flash Device
Nontrusted Manufacturing Environment
OEM Customers

Notes:

1. *Programmed portion indicated with dark gray.*
2. *Programming of FBs applies to Fusion only.*

Figure 13-7 **• Application 2: Device Programming in a Nontrusted Environment**

Application 3: Nontrusted Environment—Field Updates/Upgrades

Programming or reprogramming of devices may occur at remote locations. Reconfiguration of devices in consumer products/equipment through public networks is one example. Typically, the remote system is already programmed with particular design contents. When design update (FPGA array contents update) and/or data upgrade (FlashROM and/or FB contents upgrade) is necessary, an updated programming file with AES encryption can be generated, sent across public networks, and transmitted to the remote system. Reprogramming can then be done using this AES-encrypted programming file, providing easy and secure field upgrades. Low power flash devices support this secure ISP using AES. The detailed flow for this application is shown in Figure 13-8. Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

To prepare devices for this scenario, the user can initially generate a programming file with the available security setting options. This programming file is programmed into the devices before shipment. During the programming file generation step, the user has the option of making the security settings permanent or not. In situations where no changes to the security settings are necessary, the user can select this feature in the software to generate the programming file with permanent security settings. Microsemi recommends that the programming file use encryption with an AES key, especially when ISP is done via public domain.

For example, if the designer wants to use an AES key for the FPGA array and the FlashROM, **Permanent** needs to be chosen for this setting. At first, the user chooses the options to use an AES key for the FPGA array and the FlashROM, and then chooses **Permanently lock the security settings**. A unique AES key is chosen. Once this programming file is generated and programmed to the devices, the AES key is permanently stored in the on-chip memory, where it is secured safely. The devices are sent to distant locations for the intended application. When an update is needed, a new programming file must be generated. The programming file must use the same AES key for encryption; otherwise, the authentication will fail and the file will not be programmed in the device.

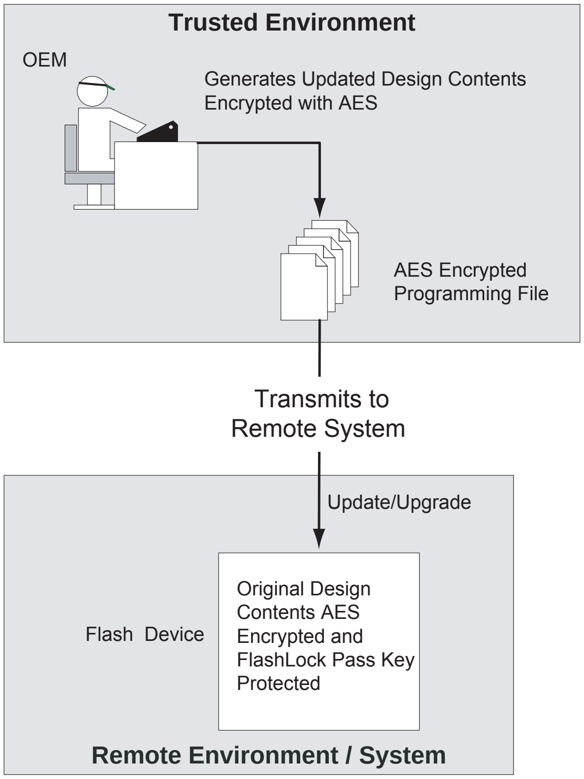

Trusted Environment
OEM
Generates Updated Design Contents Encrypted with AES
AES Encrypted Programming File
Transmits to Remote System
Update/Upgrade
Flash Device
Original Design Contents AES Encrypted and FlashLock Pass Key Protected
Remote Environment / System

Figure 13-8 • **Application 3: Nontrusted Environment—Field Updates/Upgrades**

FlashROM Security Use Models

Each of the subsequent sections describes in detail the available selections in Microsemi Designer as an aid to understanding security applications and generating appropriate programming files for those applications. Before proceeding, it is helpful to review Figure 13-7 on page 331, which gives a general overview of the programming file generation flow within the Designer software as well as what occurs during the device programming stage. Specific settings are discussed in the following sections.

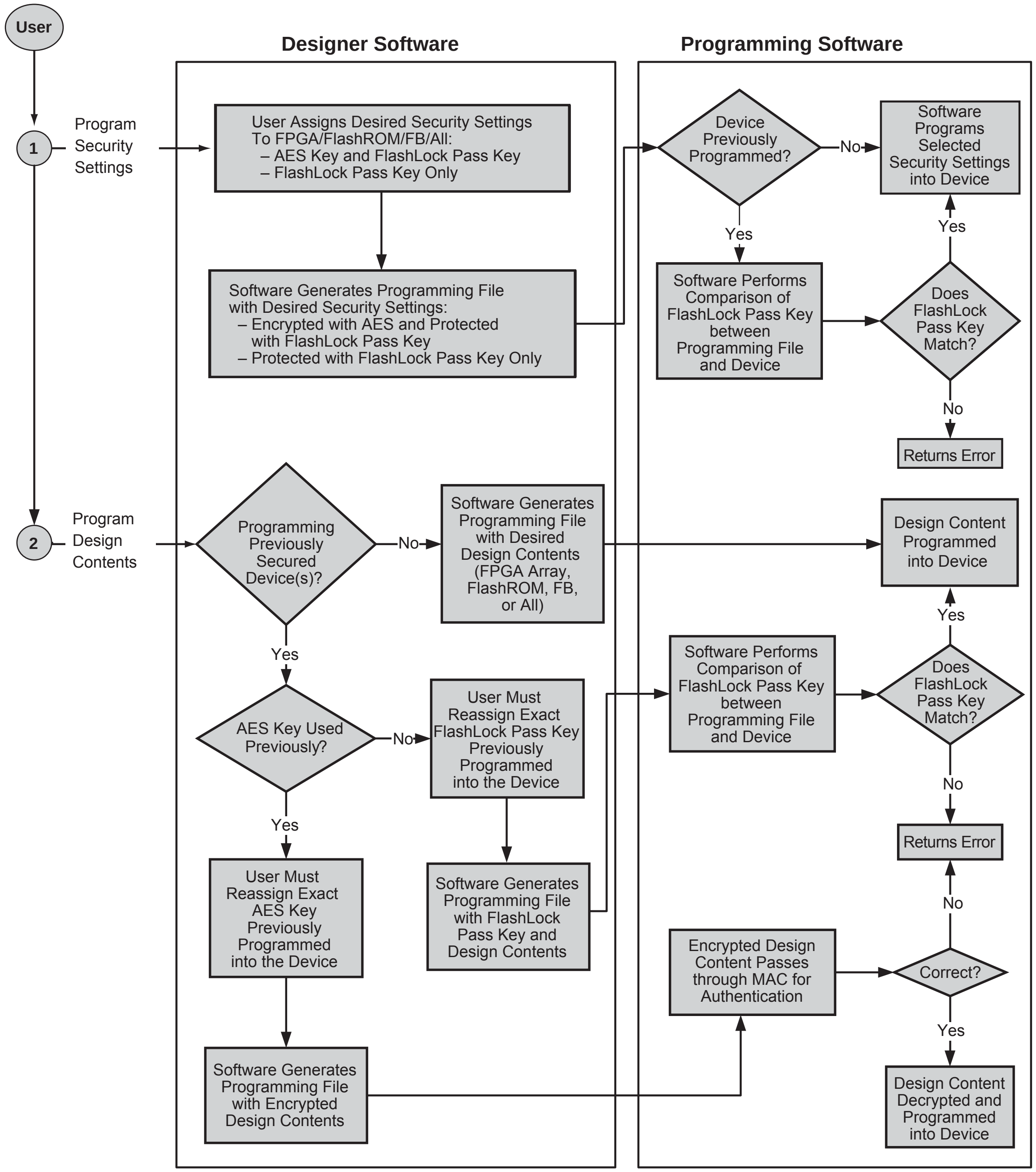
In Figure 13-7 on page 331, the flow consists of two sub-flows. Sub-flow 1 describes programming security settings to the device only, and sub-flow 2 describes programming the design contents only.

In Application 1, described in the "Application 1: Trusted Environment" section on page 331, the user does not need to generate separate files but can generate one programming file containing both security settings and design contents. Then programming of the security settings and design contents is done in one step. Both sub-flow 1 and sub-flow 2 are used.

In Application 2, described in the "Application 2: Nontrusted Environment—Unsecured Location" section on page 331, the trusted site should follow sub-flows 1 and 2 separately to generate two separate programming files. The programming file from sub-flow 1 will be used at the trusted site to program the device(s) first. The programming file from sub-flow 2 will be sent off-site for production programming.

In Application 3, described in the "Application 3: Nontrusted Environment—Field Updates/Upgrades" section on page 332, typically only sub-flow 2 will be used, because only updates to the design content portion are needed and no security settings need to be changed.

In the event that update of the security settings is necessary, see the "Reprogramming Devices" section on page 343 for details. For more information on programming low power flash devices, refer to the "In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" section on page 349.


User
Designer Software
Programming Software
1
Program Security Settings
User Assigns Desired Security Settings To FPGA/FlashROM/FB/All:
– AES Key and FlashLock Pass Key
– FlashLock Pass Key Only
Software Generates Programming File with Desired Security Settings:
– Encrypted with AES and Protected with FlashLock Pass Key
– Protected with FlashLock Pass Key Only
Device Previously Programmed?
No
Software Programs Selected Security Settings into Device
Yes
Software Performs Comparison of FlashLock Pass Key between Programming File and Device
Yes
Does FlashLock Pass Key Match?
No
Returns Error
2
Program Design Contents
Programming Previously Secured Device(s)?
No
Software Generates Programming File with Desired Design Contents (FPGA Array, FlashROM, FB, or All)
Design Content Programmed into Device
Yes
Yes
AES Key Used Previously?
No
User Must Reassign Exact FlashLock Pass Key Previously Programmed into the Device
Software Performs Comparison of FlashLock Pass Key between Programming File and Device
Does FlashLock Pass Key Match?
No
Returns Error
Yes
User Must Reassign Exact AES Key Previously Programmed into the Device
Software Generates Programming File with FlashLock Pass Key and Design Contents
No
Encrypted Design Content Passes through MAC for Authentication
Correct?
Yes
Software Generates Programming File with Encrypted Design Contents
Design Content Decrypted and Programmed into Device

Note: *If programming the Security Header only, just perform sub-flow 1.*
If programming design content only, just perform sub-flow 2.

***Figure 13-9* • Security Programming Flows**

Generating Programming Files

Generation of the Programming File in a Trusted Environment—Application 1

As discussed in the "Application 1: Trusted Environment" section on page 331, in a trusted environment, the user can choose to program the device with plaintext bitstream content. It is possible to use plaintext for programming even when the FlashLock Pass Key option has been selected. In this application, it is not necessary to employ AES encryption protection. For AES encryption settings, refer to the next sections.

The generated programming file will include the security setting (if selected) and the plaintext programming file content for the FPGA array, FlashROM, and/or FBs. These options are indicated in Table 13-2 and Table 13-3.

Table 13-2 **• IGLOO and ProASIC3 Plaintext Security Options, No AES**

Security Protection	FlashROM Only	FPGA Core Only	Both FlashROM and FPGA
No AES / no FlashLock	✓	✓	✓
FlashLock only	✓	✓	✓
AES and FlashLock	–	–	–

Table 13-3 **• Fusion Plaintext Security Options**

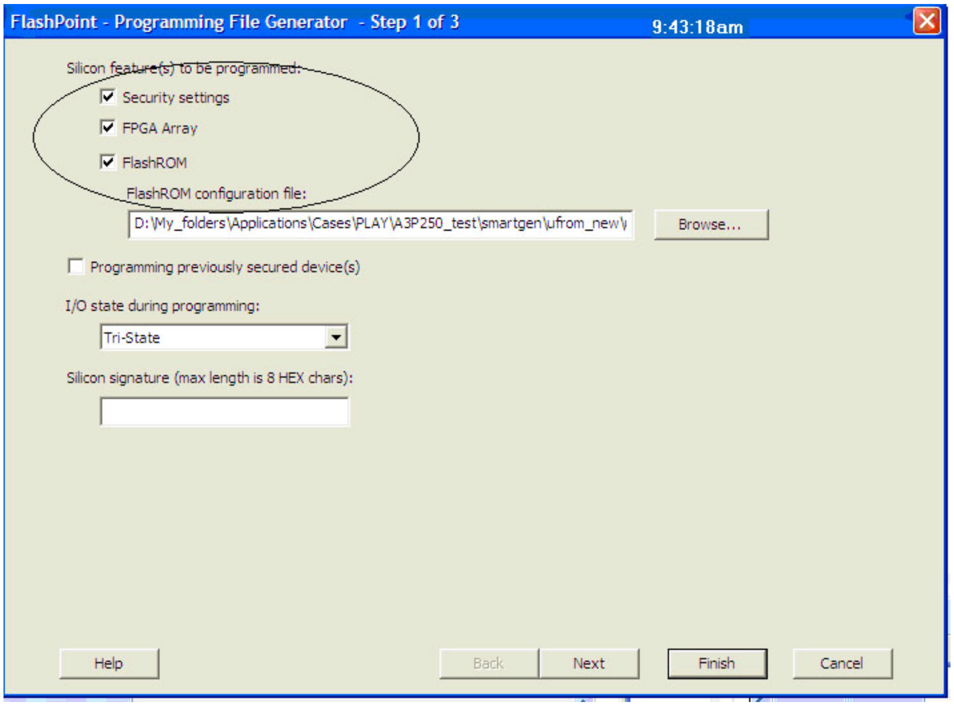
Security Protection	FlashROM Only	FPGA Core Only	FB Core Only	All
No AES / no FlashLock	✓	✓	✓	✓
FlashLock	✓	✓	✓	✓
AES and FlashLock	–	–	–	–

Note: For all instructions, the programming of Flash Blocks refers to Fusion only.

For this scenario, generate the programming file as follows:

1. Select the **Silicon features to be programmed** (Security Settings, FPGA Array, FlashROM, Flash Memory Blocks), as shown in Figure 13-10 on page 336 and Figure 13-11 on page 336. Click **Next**.

   If **Security Settings** is selected (i.e., the FlashLock security Pass Key feature), an additional dialog will be displayed to prompt you to select the security level setting. If no security setting is selected, you will be directed to Step 3.



Silicon feature(s) to be programmed:

☑ Security settings

☑ FPGA Array

☑ FlashROM

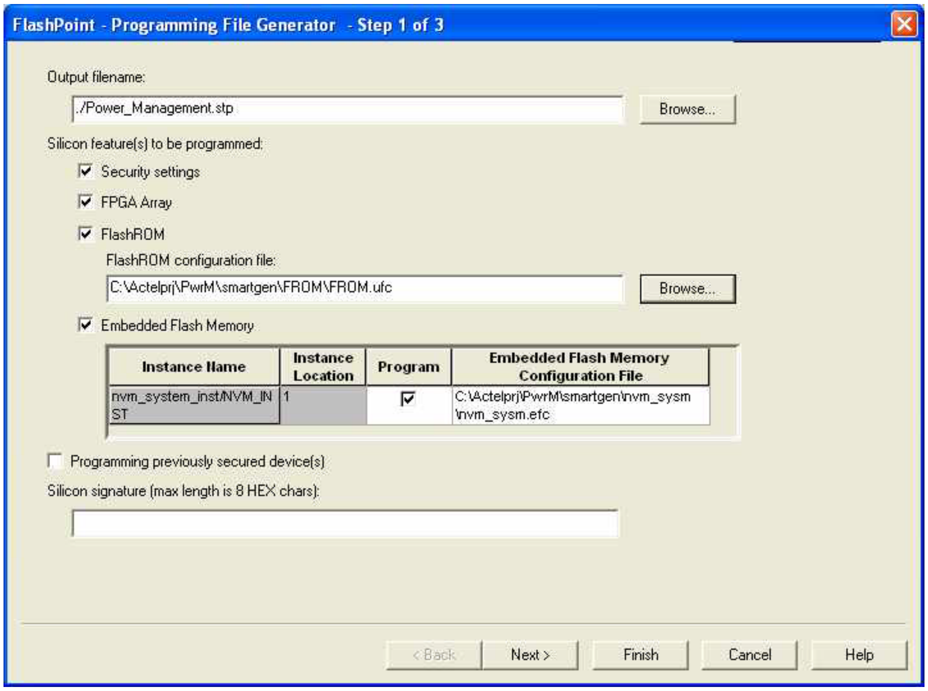
FlashROM configuration file: D:\My_folders\Applications\Cases\PLAY\A3P250_test\smartgen\ufrom_new\

☐ Programming previously secured device(s)

I/O state during programming: Tri-State

Silicon signature (max length is 8 HEX chars):

Figure 13-10 **• All Silicon Features Selected for IGLOO and ProASIC3 Devices**



Output filename: ./Power_Management.stp

Silicon feature(s) to be programmed:

☑ Security settings

☑ FPGA Array

☑ FlashROM

FlashROM configuration file: C:\Actelprj\PwrM\smartgen\FROM\FROM.ufc

☑ Embedded Flash Memory

Instance Name	Instance Location	Program	Embedded Flash Memory Configuration File
nvm_system_inst/NVM_INST	1	☑	C:\Actelprj\PwrM\smartgen\nvm_sysm\nvm_sysm.efc

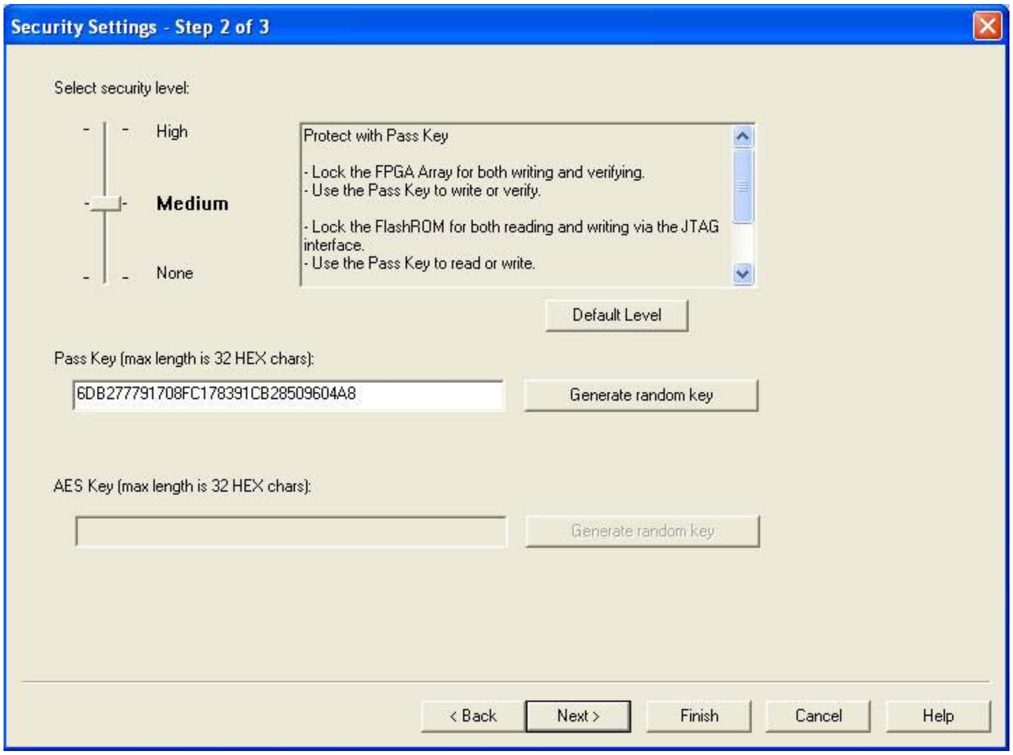
☐ Programming previously secured device(s)

Silicon signature (max length is 8 HEX chars):

Figure 13-11 **• All Silicon Features Selected for Fusion**

2. Choose the appropriate security level setting and enter a FlashLock Pass Key. The default is the **Medium** security level (Figure 13-12). Click **Next**.

   If you want to select different options for the FPGA and/or FlashROM, this can be set by clicking **Custom Level**. Refer to the "Advanced Options" section on page 344 for different custom security level options and descriptions of each.



Security Settings - Step 2 of 3

Select security level:

High

Medium

None

Protect with Pass Key

- Lock the FPGA Array for both writing and verifying.
- Use the Pass Key to write or verify.

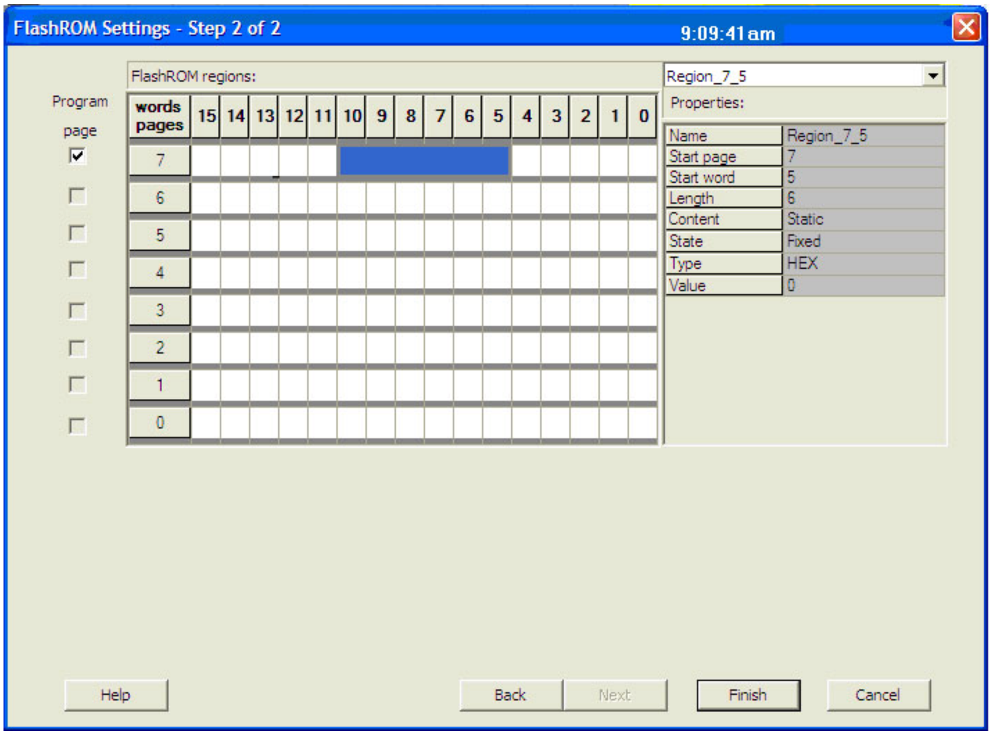
- Lock the FlashROM for both reading and writing via the JTAG interface.
- Use the Pass Key to read or write.

Pass Key (max length is 32 HEX chars): 6DB277791708FC178391CB28509604A8

AES Key (max length is 32 HEX chars):

Figure 13-12 • **Medium Security Level Selected for Low Power Flash Devices**

3. Choose the desired settings for the FlashROM configurations to be programmed (Figure 13-13). Click **Finish** to generate the STAPL programming file for the design.



Program page: ☑ 7, □ 6, □ 5, □ 4, □ 3, □ 2, □ 1, □ 0

Region_7_5

Properties:

Name	Region_7_5
Start page	7
Start word	5
Length	6
Content	Static
State	Fixed
Type	HEX
Value	0

Figure 13-13 • **FlashROM Configuration Settings for Low Power Flash Devices**

Generation of Security Header Programming File Only—Application 2

As mentioned in the "Application 2: Nontrusted Environment—Unsecured Location" section on page 331, the designer may employ FlashLock Pass Key protection or FlashLock Pass Key with AES encryption on the device before sending it to a nontrusted or unsecured location for device programming. To achieve this, the user needs to generate a programming file containing only the security settings desired (Security Header programming file).

Note: If AES encryption is configured, FlashLock Pass Key protection must also be configured.

The available security options are indicated in Table 13-4 and Table 13-5 on page 339.

Table 13-4 • **FlashLock Security Options for IGLOO and ProASIC3**

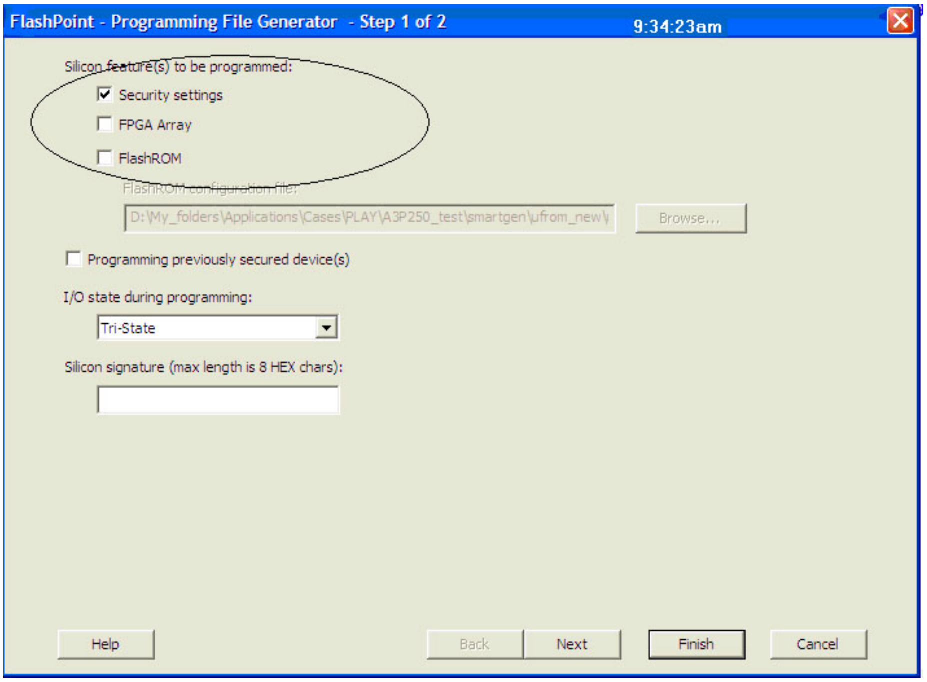
Security Option	FlashROM Only	FPGA Core Only	Both FlashROM and FPGA
No AES / no FlashLock	–	–	–
FlashLock only	✓	✓	✓
AES and FlashLock	✓	✓	✓

***Table 13-5* • FlashLock Security Options for Fusion**

Security Option	FlashROM Only	FPGA Core Only	FB Core Only	All
No AES / no FlashLock	–	–	–	–
FlashLock	✓	✓	✓	✓
AES and FlashLock	✓	✓	✓	✓

For this scenario, generate the programming file as follows:

1. Select only the **Security settings** option, as indicated in Figure 13-14 and Figure 13-15 on page 340. Click **Next**.



FlashPoint - Programming File Generator - Step 1 of 2

9:34:23am

Silicon feature(s) to be programmed:

☑ Security settings

□ FPGA Array

□ FlashROM

FlashROM configuration file:

D:\My_folders\Applications\Cases\PLAY\A3P250_test\smartgen\ufrom_new\

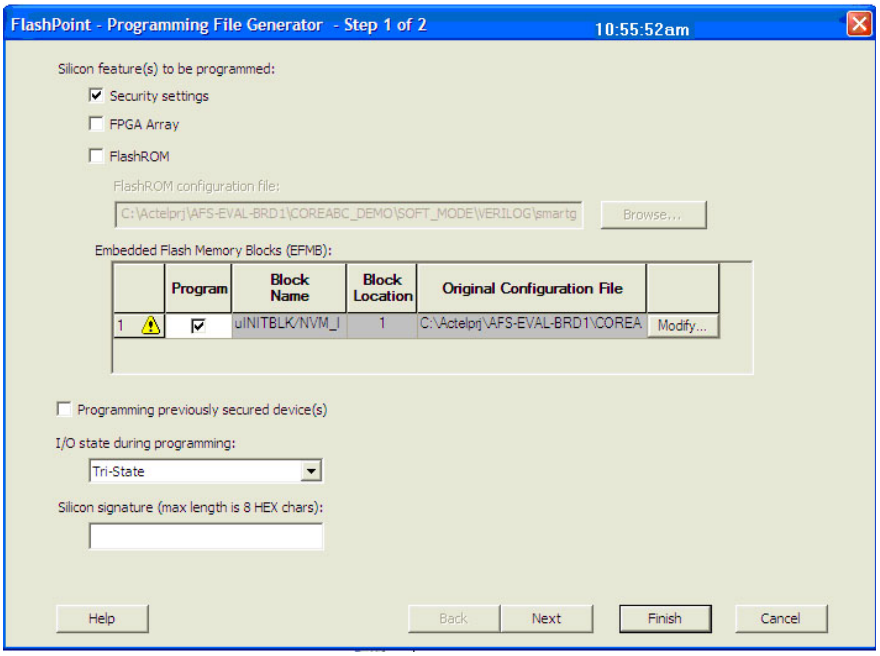
□ Programming previously secured device(s)

I/O state during programming:

Tri-State

Silicon signature (max length is 8 HEX chars):

***Figure 13-14* • Programming IGLOO and ProASIC3 Security Settings Only**



Silicon feature(s) to be programmed:

☑ Security settings

□ FPGA Array

□ FlashROM

FlashROM configuration file: C:\Actelprj\AFS-EVAL-BRD1\COREABC_DEMO\SOFT_MODE\VERILOG\smartg

Embedded Flash Memory Blocks (EFMB):

	Program	Block Name	Block Location	Original Configuration File	
1 ⚠	☑	uINITBLK/NVM_I	1	C:\Actelprj\AFS-EVAL-BRD1\COREA	Modify...

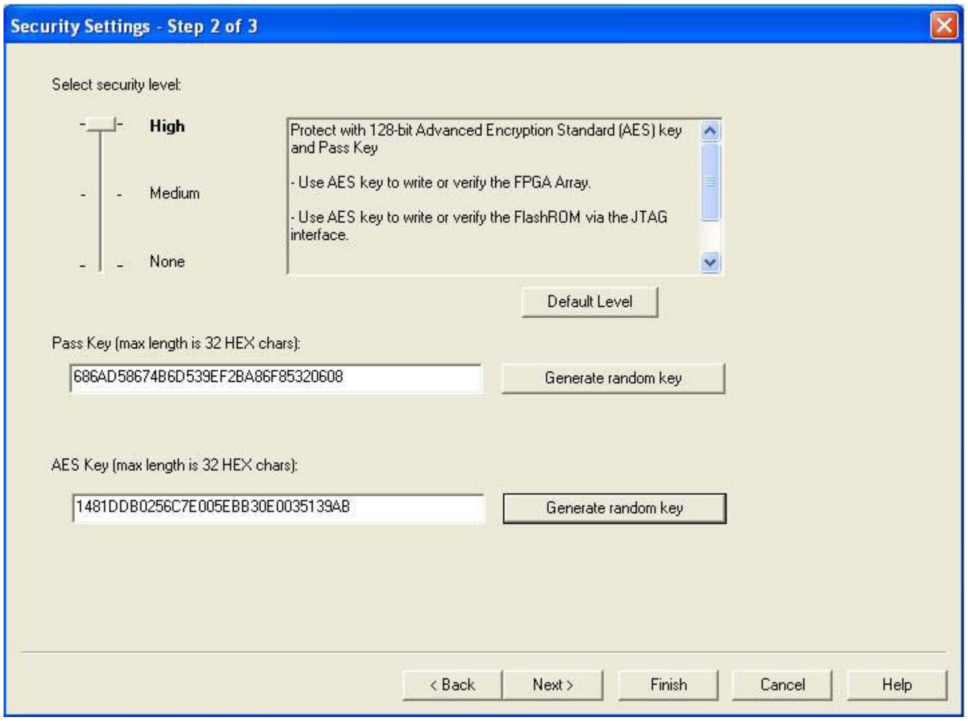
□ Programming previously secured device(s)

I/O state during programming: Tri-State

Silicon signature (max length is 8 HEX chars):

Figure 13-15 **• Programming Fusion Security Settings Only**

2. Choose the desired security level setting and enter the key(s).
   – The **High** security level employs FlashLock Pass Key with AES Key protection.
   – The **Medium** security level employs FlashLock Pass Key protection only.



Select security level: High

Protect with 128-bit Advanced Encryption Standard (AES) key and Pass Key

- Use AES key to write or verify the FPGA Array.

- Use AES key to write or verify the FlashROM via the JTAG interface.

Pass Key (max length is 32 HEX chars): 686AD58674B6D539EF2BA86F85320608

AES Key (max length is 32 HEX chars): 1481DDB0256C7E005EBB30E0035139AB

Figure 13-16 **• High Security Level to Implement FlashLock Pass Key and AES Key Protection**

Table 13-6 and Table 13-7 show all available options. If you want to implement custom levels, refer to the "Advanced Options" section on page 344 for information on each option and how to set it.

3. When done, click **Finish** to generate the Security Header programming file.

***Table 13-6* • All IGLOO and ProASIC3 Header File Security Options**

Security Option	FlashROM Only	FPGA Core Only	Both FlashROM and FPGA
No AES / no FlashLock	✓	✓	✓
FlashLock only	✓	✓	✓
AES and FlashLock	✓	✓	✓

Note: ✓ = options that may be used

***Table 13-7* • All Fusion Header File Security Options**

Security Option	FlashROM Only	FPGA Core Only	FB Core Only	All
No AES / No FlashLock	✓	✓	✓	✓
FlashLock	✓	✓	✓	✓
AES and FlashLock	✓	✓	✓	✓

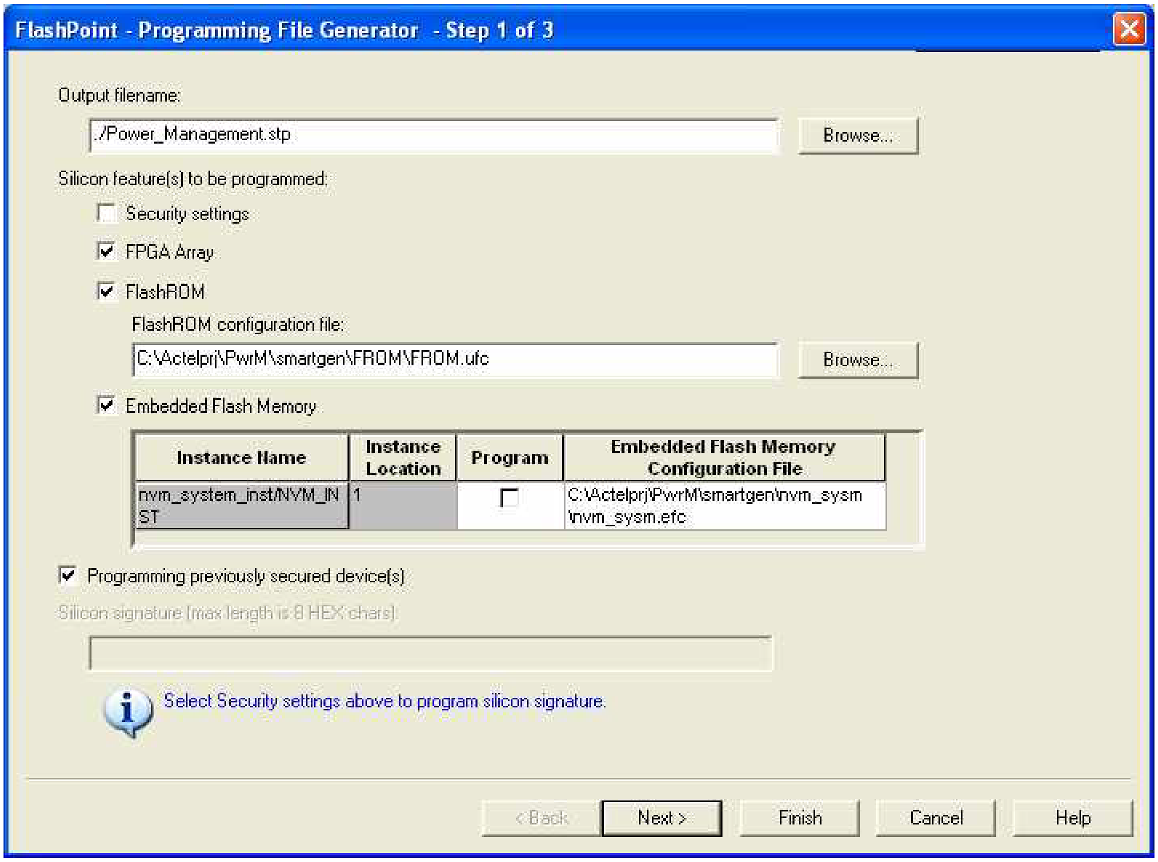
Generation of Programming Files with AES Encryption—Application 3

This section discusses how to generate design content programming files needed specifically at unsecured or remote locations to program devices with a Security Header (FlashLock Pass Key and AES key) already programmed ("Application 2: Nontrusted Environment—Unsecured Location" section on page 331 and "Application 3: Nontrusted Environment—Field Updates/Upgrades" section on page 332). In this case, the encrypted programming file must correspond to the AES key already programmed into the device. If AES encryption was previously selected to encrypt the FlashROM, FBs, and FPGA array, AES encryption must be set when generating the programming file for them. AES encryption can be applied to the FlashROM only, the FBs only, the FPGA array only, or all. The user must ensure both the FlashLock Pass Key and the AES key match those already programmed to the device(s), and all security settings must match what was previously programmed. Otherwise, the encryption and/or device unlocking will not be recognized when attempting to program the device with the programming file.

The generated programming file will be AES-encrypted.

In this scenario, generate the programming file as follows:

1. Deselect **Security settings** and select the portion of the device to be programmed (Figure 13-17 on page 342). Select **Programming previously secured device(s**). Click **Next**.



Output filename:

./Power_Management.stp

Silicon feature(s) to be programmed:

□ Security settings

☑ FPGA Array

☑ FlashROM

FlashROM configuration file:

C:\Actelprj\PwrM\smartgen\FROM\FROM.ufc

☑ Embedded Flash Memory

Instance Name	Instance Location	Program	Embedded Flash Memory Configuration File
nvm_system_inst/NVM_INST	1	□	C:\Actelprj\PwrM\smartgen\nvm_sysm\nvm_sysm.efc

☑ Programming previously secured device(s)

Silicon signature (max length is 8 HEX chars):

Select Security settings above to program silicon signature.

Note: *The settings in this figure are used to show the generation of an AES-encrypted programming file for the FPGA array, FlashROM, and FB contents. One or all locations may be selected for encryption.*

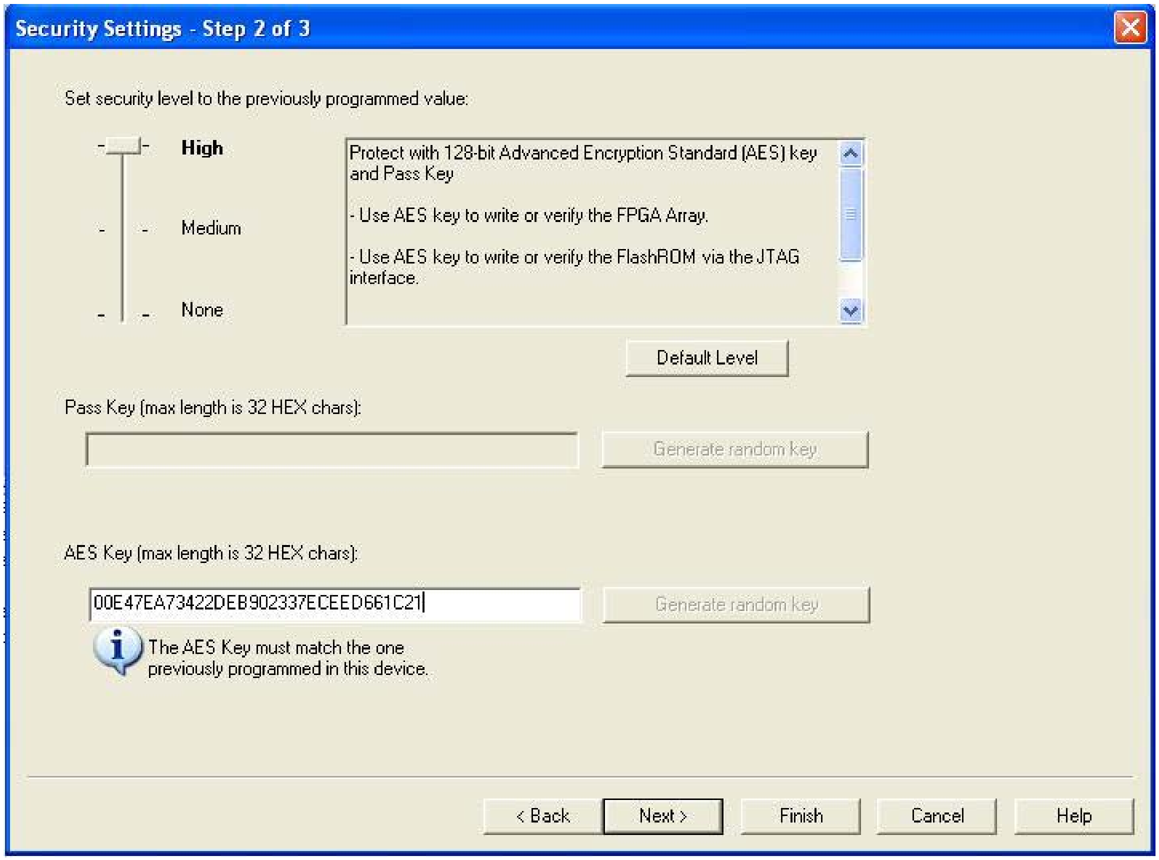
***Figure 13-17* • Settings to Program a Device Secured with FlashLock and using AES Encryption**

Choose the **High** security level to reprogram devices using both the FlashLock Pass Key and AES key protection (Figure 13-18 on page 343). Enter the AES key and click **Next**.

A device that has already been secured with FlashLock and has an AES key loaded must recognize the AES key to program the device and generate a valid bitstream in authentication. The FlashLock Key is only required to unlock the device and change the security settings.

This is what makes it possible to program in an untrusted environment. The AES key is protected inside the device by the FlashLock Key, so you can only program if you have the correct AES key. In fact, the AES key is not in the programming file either. It is the key used to encrypt the data in the file. The same key previously programmed with the FlashLock Key matches to decrypt the file.

An AES-encrypted file programmed to a device without FlashLock would not be secure, since without FlashLock to protect the AES key, someone could simply reprogram the AES key first, then program with any AES key desired or no AES key at all. This option is therefore not available in the software.



Security Settings - Step 2 of 3

Set security level to the previously programmed value:

Protect with 128-bit Advanced Encryption Standard (AES) key and Pass Key

- Use AES key to write or verify the FPGA Array.

- Use AES key to write or verify the FlashROM via the JTAG interface.

Pass Key (max length is 32 HEX chars):

AES Key (max length is 32 HEX chars):

00E47EA73422DEB902337ECEED661C21

The AES Key must match the one previously programmed in this device.

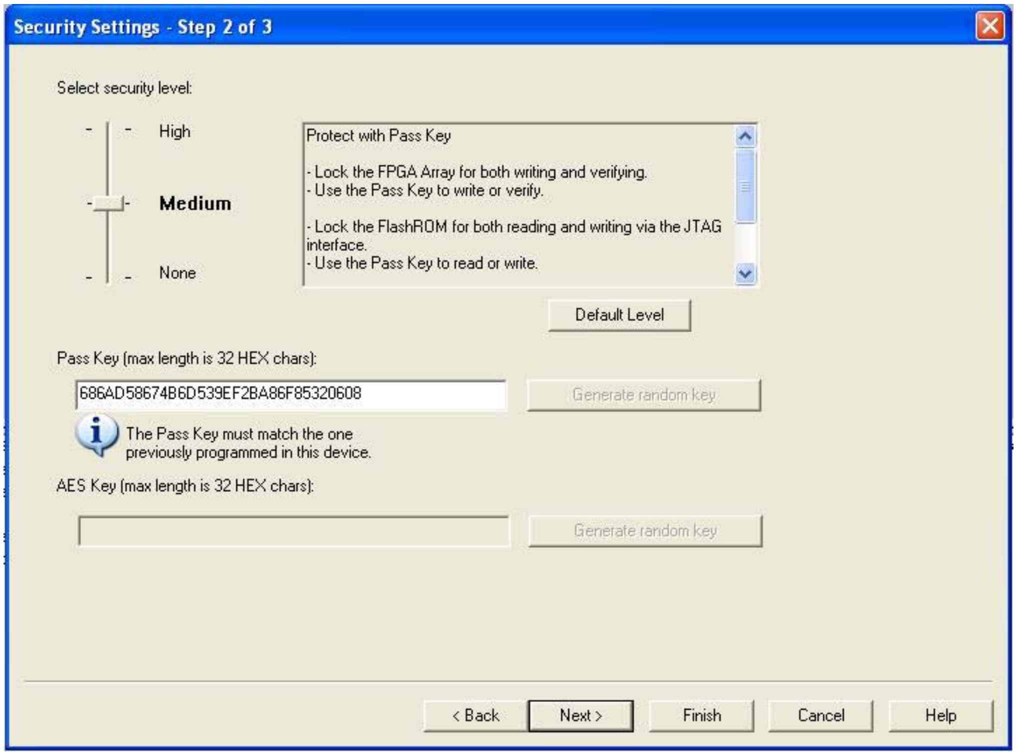
Figure 13-18 • **Security Level Set High to Reprogram Device with AES Key**

Programming with this file is intended for an unsecured environment. The AES key encrypts the programming file with the same AES key already used in the device and utilizes it to program the device.

Reprogramming Devices

Previously programmed devices can be reprogrammed using the steps in the "Generation of the Programming File in a Trusted Environment—Application 1" section on page 335 and "Generation of Security Header Programming File Only—Application 2" section on page 338. In the case where a FlashLock Pass Key has been programmed previously, the user must generate the new programming file with a FlashLock Pass Key that matches the one previously programmed into the device. The software will check the FlashLock Pass Key in the programming file against the FlashLock Pass Key in the device. The keys must match before the device can be unlocked to perform further programming with the new programming file.

Figure 13-10 on page 336 and Figure 13-11 on page 336 show the option **Programming previously secured device(s)**, which the user should select before proceeding. Upon going to the next step, the user will be notified that the same FlashLock Pass Key needs to be entered, as shown in Figure 13-19 on page 344.



Security Settings - Step 2 of 3

Select security level: Medium

Protect with Pass Key

- Lock the FPGA Array for both writing and verifying.
- Use the Pass Key to write or verify.

- Lock the FlashROM for both reading and writing via the JTAG interface.
- Use the Pass Key to read or write.

Pass Key (max length is 32 HEX chars): 686AD58674B6D539EF2BA86F85320608

The Pass Key must match the one previously programmed in this device.

AES Key (max length is 32 HEX chars):

***Figure 13-19* • FlashLock Pass Key, Previously Programmed Devices**

It is important to note that when the security settings need to be updated, the user also needs to select the **Security settings** check box in Step 1, as shown in Figure 13-10 on page 336 and Figure 13-11 on page 336, to modify the security settings. The user must consider the following:

- If only a new AES key is necessary, the user must re-enter the same Pass Key previously programmed into the device in Designer and then generate a programming file with the same Pass Key and a different AES key. This ensures the programming file can be used to access and program the device and the new AES key.
- If a new Pass Key is necessary, the user can generate a new programming file with a new Pass Key (with the same or a new AES key if desired). However, for programming, the user must first load the original programming file with the Pass Key that was previously used to unlock the device. Then the new programming file can be used to program the new security settings.

Advanced Options

As mentioned, there may be applications where more complicated security settings are required. The “Custom Security Levels” section in the *FlashPro User's Guide* describes different advanced options available to aid the user in obtaining the best available security settings.

Programming File Header Definition

In each STAPL programming file generated, there will be information about how the AES key and FlashLock Pass Key are configured. Table 13-8 shows the header definitions in STAPL programming files for different security levels.

Table 13-8 • **STAPL Programming File Header Definitions by Security Level**

Security Level	STAPL File Header Definition
No security (no FlashLock Pass Key or AES key)	`NOTE "SECURITY" "Disable";`
FlashLock Pass Key with no AES key	`NOTE "SECURITY" "KEYED ";`
FlashLock Pass Key with AES key	`NOTE "SECURITY" "KEYED ENCRYPT ";`
Permanent Security Settings option enabled	`NOTE "SECURITY" "PERMLOCK ENCRYPT ";`
AES-encrypted FPGA array (for programming updates)	`NOTE "SECURITY" "ENCRYPT CORE ";`
AES-encrypted FlashROM (for programming updates)	`NOTE "SECURITY" "ENCRYPT FROM ";`
AES-encrypted FPGA array and FlashROM (for programming updates)	`NOTE "SECURITY" "ENCRYPT FROM CORE ";`

Example File Headers

STAPL Files Generated with FlashLock Key and AES Key Containing Key Information

- FlashLock Key / AES key indicated in STAPL file header definition
- Intended ONLY for secured/trusted environment programming applications

```
=============================================
NOTE "CREATOR" "Designer Version: 6.1.1.108";
NOTE "DEVICE" "A3PE600";
NOTE "PACKAGE" "208 PQFP";
NOTE "DATE" "2005/04/08";
NOTE "STAPL_VERSION" "JESD71";
NOTE "IDCODE" "$123261CF";
NOTE "DESIGN" "counter32";
NOTE "CHECKSUM" "$EDB9";
NOTE "SAVE_DATA" "FRomStream";
NOTE "SECURITY" "KEYED ENCRYPT ";
NOTE "ALG_VERSION" "1";
NOTE "MAX_FREQ" "20000000";
NOTE "SILSIG" "$00000000";
NOTE "PASS_KEY" "$00123456789012345678901234567890";
NOTE "AES_KEY" "$ABCDEFABCDEFABCDEFABCDEFABCDEFAB";
==============================================
```

STAPL File with AES Encryption

- Does not contain AES key / FlashLock Key information
- Intended for transmission through web or service to unsecured locations for programming

```
=============================================
NOTE "CREATOR" "Designer Version: 6.1.1.108";
NOTE "DEVICE" "A3PE600";
NOTE "PACKAGE" "208 PQFP";
NOTE "DATE" "2005/04/08";
NOTE "STAPL_VERSION" "JESD71";
NOTE "IDCODE" "$123261CF";
NOTE "DESIGN" "counter32";
NOTE "CHECKSUM" "$EF57";
NOTE "SAVE_DATA" "FRomStream";
NOTE "SECURITY" "ENCRYPT FROM CORE ";
NOTE "ALG_VERSION" "1";
NOTE "MAX_FREQ" "20000000";
NOTE "SILSIG" "$00000000";
```

Conclusion

The new and enhanced security features offered in Fusion, IGLOO, and ProASIC3 devices provide state-of-the-art security to designs programmed into these flash-based devices. Microsemi low power flash devices employ the encryption standard used by NIST and the U.S. government—AES using the 128-bit Rijndael algorithm.

The combination of an on-chip AES decryption engine and FlashLock technology provides the highest level of security against invasive attacks and design theft, implementing the most robust and secure ISP solution. These security features protect IP within the FPGA and protect the system from cloning, wholesale “black box” copying of a design, invasive attacks, and explicit IP or data theft.

Glossary

Term	Explanation
Security Header programming file	Programming file used to program the FlashLock Pass Key and/or AES key into the device to secure the FPGA, FlashROM, and/or FBs.
AES (encryption) key	128-bit key defined by the user when the AES encryption option is set in the Microsemi Designer software when generating the programming file.
FlashLock Pass Key	128-bit key defined by the user when the FlashLock option is set in the Microsemi Designer software when generating the programming file. The FlashLock Key protects the security settings programmed to the device. Once a device is programmed with FlashLock, whatever settings were chosen at that time are secure.
FlashLock	The combined security features that protect the device content from attacks. These features are the following: • Flash technology that does not require an external bitstream to program the device • FlashLock Pass Key that secures device content by locking the security settings and preventing access to the device as defined by the user • AES key that allows secure, encrypted device reprogrammability

References

National Institute of Standards and Technology. “ADVANCED ENCRYPTION STANDARD (AES) Questions and Answers.” 28 January 2002 (10 January 2005).
See http://csrc.nist.gov/archive/aes/index1.html for more information.

Related Documents

User's Guides

FlashPro User's Guide

http://www.microsemi.com/soc/documents/flashpro_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.5 (August 2009)	The "CoreMP7 Device Security" section was removed from "Security in ARM-Enabled Low Power Flash Devices", since M7-enabled devices are no longer supported.	326
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 13-1 • Flash-Based FPGAs.	324
v1.3 (October 2008)	The "Security Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	324
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 13-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	324
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices.	N/A
	The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	324

14 – In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X

Introduction

Microsemi's low power flash devices are all in-system programmable. This document describes the general requirements for programming a device and specific requirements for the FlashPro4/3/3X programmers[1].

IGLOO, ProASIC3, SmartFusion, and Fusion devices offer a low power, single-chip, live-at-power-up solution with the ASIC advantages of security and low unit cost through nonvolatile flash technology. Each device contains 1 kbit of on-chip, user-accessible, nonvolatile FlashROM. The FlashROM can be used in diverse system applications such as Internet Protocol (IP) addressing, user system preference storage, device serialization, or subscription-based business models. IGLOO, ProASIC3, SmartFusion, and Fusion devices offer the best in-system programming (ISP) solution, FlashLock® security features, and AES-decryption-based ISP.

ISP Architecture

Low power flash devices support ISP via JTAG and require a single VPUMP voltage of 3.3 V during programming. In addition, programming via a microcontroller in a target system is also supported.

Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide.

Family-specific support:

- ProASIC3, ProASIC3E, SmartFusion, and Fusion devices support ISP.
- ProASIC3L devices operate using a 1.2 V core voltage; however, programming can be done only at 1.5 V. Voltage switching is required in-system to switch from a 1.2 V core to 1.5 V core for programming.
- IGLOO and IGLOOe V5 devices can be programmed in-system when the device is using a 1.5 V supply voltage to the FPGA core.
- IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only) or 1.5 V. IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V. Voltage switching is required in-system to switch from a 1.2 V supply (VCC,VCCI, and VJTAG) to 1.5 V for programming. The exception is that V2 devices can be programmed at 1.2 V VCC with FlashPro4.

IGLOO devices cannot be programmed in-system when the device is in Flash*Freeze mode. The device should exit Flash*Freeze mode and be in normal operation for programming to start. Programming operations in IGLOO devices can be achieved when the device is in normal operating mode and a 1.5 V core voltage is used.

JTAG 1532

IGLOO, ProASIC3, SmartFusion, and Fusion devices support the JTAG-based IEEE 1532 standard for ISP. To start JTAG operations, the IGLOO device must exit Flash*Freeze mode and be in normal operation before starting to send JTAG commands to the device. As part of this support, when a device is in an unprogrammed state, all user I/O pins are disabled. This is achieved by keeping the global IO_EN

[1] *FlashPro4 replaced FlashPro3/3X in 2010 and is backward compatible with FlashPro3/3X as long as there is no connection to pin 4 on the JTAG header on the board. On FlashPro3/3X, there is no connection to pin 4 on the JTAG header; however, pin 4 is used for programming mode (Prog_Mode) on FlashPro4. When converting from FlashPro3/3X to FlashPro4, users should make sure that JTAG connectors on system boards do not have any connection to pin 4. FlashPro3X supports discrete TCK toggling that is needed to support non-JTAG compliant devices in the chain. This feature is included in FlashPro4.*

signal deactivated, which also has the effect of disabling the input buffers. The SAMPLE/PRELOAD instruction captures the status of pads in parallel and shifts them out as new data is shifted in for loading into the Boundary Scan Register (BSR). When the device is in an unprogrammed state, the OE and output BSR will be undefined; however, the input BSR will be defined as long as it is connected and being used. For JTAG timing information on setup, hold, and fall times, refer to the *FlashPro User's Guide*.

ISP Support in Flash-Based Devices

The flash FPGAs listed in Table 14-1 support the ISP feature and the functions described in this document.

Table 14-1 **• Flash-Based FPGAs Supporting ISP**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
SmartFusion	SmartFusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable microcontroller subsystem (MSS) which includes programmable analog and an ARM® Cortex™-M3 hard processor and flash memory in a monolithic device
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device
ProASIC	ProASIC	First generation ProASIC devices
	ProASIC^{PLUS}	Second generation ProASIC devices

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 14-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 14-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Programming Voltage (VPUMP) and VJTAG

Low-power flash devices support on-chip charge pumps, and therefore require only a single 3.3 V programming voltage for the VPUMP pin during programming. When the device is not being programmed, the VPUMP pin can be left floating or can be tied (pulled up) to any voltage between 0 V and 3.6 V[1]. During programming, the target board or the FlashPro4/3/3X programmer can provide VPUMP. FlashPro4/3/3X is capable of supplying VPUMP to a single device. If more than one device is to be programmed using FlashPro4/3/3X on a given board, FlashPro4/3/3X should not be relied on to supply the VPUMP voltage. A FlashPro4/3/3X programmer is not capable of providing reliable VJTAG voltage. The board must supply VJTAG voltage to the device and the VJTAG pin of the programmer header must be connected to the device VJTAG pin. Microsemi recommends that VPUMP[2] and VJTAG power supplies be kept separate with independent filtering capacitors rather than supplying them from a common rail. Refer to the "Board-Level Considerations" section on page 359 for capacitor requirements.

Low power flash device I/Os support a bank-based, voltage-supply architecture that simultaneously supports multiple I/O voltage standards (Table 14-2). By isolating the JTAG power supply in a separate bank from the user I/Os, low power flash devices provide greater flexibility with supply selection and simplify power supply and printed circuit board (PCB) design. The JTAG pins can be run at any voltage from 1.5 V to 3.3 V (nominal). Microsemi recommends that TCK be tied to GND through a 200 ohm to 1 Kohm resistor. This prevents a possible totempole current on the input buffer stage. For TDI, TMS, and TRST pins, the devices provide an internal nominal 10 Kohm pull-up resistor. During programming, all I/O pins, except for JTAG interface pins, are tristated and weakly pulled up to VCCI. This isolates the part and prevents the signals from floating. The JTAG interface pins are driven by the FlashPro4/3/3X during programming, including the TRST pin, which is driven HIGH.

***Table 14-2* • Power Supplies**

Power Supply	Programming Mode	Current during Programming
VCC	1.2 V / 1.5 V	< 70 mA
VCCI	1.2 V / 1.5 V / 1.8 V / 2.5 V / 3.3 V (bank-selectable)	I/Os are weakly pulled up.
VJTAG	1.2 V / 1.5 V / 1.8 V / 2.5 V / 3.3 V	< 20 mA
VPUMP	3.15 V to 3.45 V	< 80 mA

Note: *All supply voltages should be at 1.5 V or higher, regardless of the setting during normal operation, except for IGLOO nano, where 1.2 V VCC and VJTAG programming is allowed.*

Nonvolatile Memory (NVM) Programming Voltage

SmartFusion and Fusion devices need stable VCCNVM/VCCENVM[3] (1.5 V power supply to the embedded nonvolatile memory blocks) and VCCOSC/VCCROSC[3] (3.3 V power supply to the integrated RC oscillator). The tolerance of VCCNVM/VCCENVM is ± 5% and VCCOSC/VCCROSC is ± 5%.

Unstable supply voltage on these pins can cause an NVM programming failure due to NVM page corruption. The NVM page can also be corrupted if the NVM reset pin has noise. This signal must be tied off properly.

Microsemi recommends installing the following capacitors[4] on the VCCNVM/VCCENVM and VCCOSC/VCCROSC pins:

- Add one bypass capacitor of 10 μF for each power supply plane followed by an array of decoupling capacitors of 0.1 μF.
- Add one 0.1 μF capacitor near each pin.

[1] *During sleep mode in IGLOO devices connect VPUMP to GND.*
[2] *VPUMP has to be quiet for successful programming. Therefore VPUMP must be separate and required capacitors must be installed close to the FPGA VPUMP pin.*
[3] *VCCROSC is for SmartFusion.*
[4] *The capacitors cannot guarantee reliable operation of the device if the board layout is not done properly.*

IEEE 1532 (JTAG) Interface

The supported industry-standard IEEE 1532 programming interface builds on the IEEE 1149.1 (JTAG) standard. IEEE 1532 defines the standardized process and methodology for ISP. Both silicon and software issues are addressed in IEEE 1532 to create a simplified ISP environment. Any IEEE 1532 compliant programmer can be used to program low power flash devices. Device serialization is not supported when using the IEEE1532 standard. Refer to the standard for detailed information about IEEE 1532.

Security

Unlike SRAM-based FPGAs that require loading at power-up from an external source such as a microcontroller or boot PROM, Microsemi nonvolatile devices are live at power-up, and there is no bitstream required to load the device when power is applied. The unique flash-based architecture prevents reverse engineering of the programmed code on the device, because the programmed data is stored in nonvolatile memory cells. Each nonvolatile memory cell is made up of small capacitors and any physical deconstruction of the device will disrupt stored electrical charges.

Each low power flash device has a built-in 128-bit Advanced Encryption Standard (AES) decryption core, except for the 30 k gate devices and smaller. Any FPGA core or FlashROM content loaded into the device can optionally be sent as encrypted bitstream and decrypted as it is loaded. This is particularly suitable for applications where device updates must be transmitted over an unsecured network such as the Internet. The embedded AES decryption core can prevent sensitive data from being intercepted (Figure 14-1 on page 353). A single 128-bit AES Key (32 hex characters) is used to encrypt FPGA core programming data and/or FlashROM programming data in the Microsemi tools. The low power flash devices also decrypt with a single 128-bit AES Key. In addition, low power flash devices support a Message Authentication Code (MAC) for authentication of the encrypted bitstream on-chip. This allows the encrypted bitstream to be authenticated and prevents erroneous data from being programmed into the device. The FPGA core, FlashROM, and Flash Memory Blocks (FBs), in Fusion only, can be updated independently using a programming file that is AES-encrypted (cipher text) or uses plain text.

Security in ARM-Enabled Low Power Flash Devices

There are slight differences between the regular flash device and the ARM-enabled flash devices, which have the M1 prefix.

The AES key is used by Microsemi and preprogrammed into the device to protect the ARM IP. As a result, the design will be encrypted along with the ARM IP, according to the details below.

Cortex-M1 and Cortex-M3 Device Security

Cortex-M1–enabled and Cortex-M3 devices are shipped with the following security features:

- FPGA array enabled for AES-encrypted programming and verification
- FlashROM enabled for AES-encrypted write and verify
- Embedded Flash Memory enabled for AES encrypted write

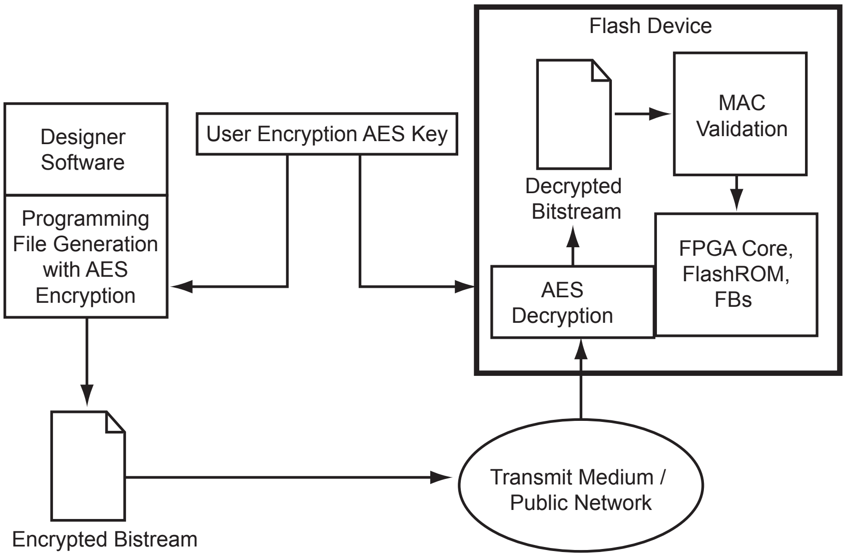


Figure 14-1 • **AES-128 Security Features**

Figure 14-2 shows different applications for ISP programming.

1. In a trusted programming environment, you can program the device using the unencrypted (plaintext) programming file.
2. You can program the AES Key in a trusted programming environment and finish the final programming in an untrusted environment using the AES-encrypted (cipher text) programming file.
3. For the remote ISP updating/reprogramming, the AES Key stored in the device enables the encrypted programming bitstream to be transmitted through the untrusted network connection.

Microsemi low power flash devices also provide the unique Microsemi FlashLock feature, which protects the Pass Key and AES Key. Unless the original FlashLock Pass Key is used to unlock the device, security settings cannot be modified. Microsemi does not support read-back of FPGA core-programmed data; however, the FlashROM contents can selectively be read back (or disabled) via the JTAG port based on the security settings established by the Microsemi Designer software. Refer to the "Security in Low Power Flash Devices" section on page 323 for more information.

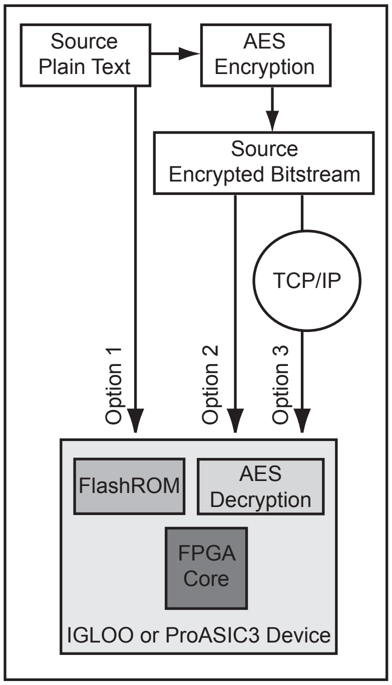


Figure 14-2 • **Different ISP Use Models**

FlashROM and Programming Files

Each low power flash device has 1 kbit of on-chip, nonvolatile flash memory that can be accessed from the FPGA core. This nonvolatile FlashROM is arranged in eight pages of 128 bits (Figure 14-3). Each page can be programmed independently, with or without the 128-bit AES encryption. The FlashROM can only be programmed via the IEEE 1532 JTAG port and cannot be programmed from the FPGA core. In addition, during programming of the FlashROM, the FPGA core is powered down automatically by the on-chip programming control logic.

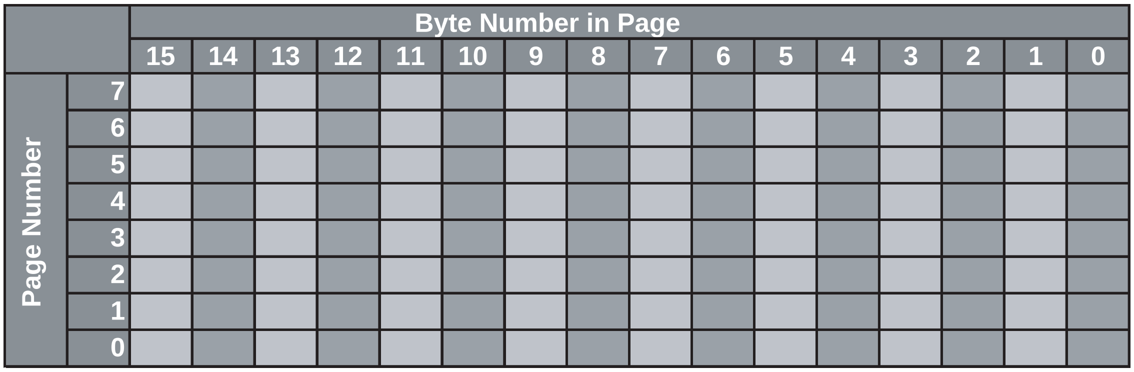


Figure 14-3 **• FlashROM Architecture**

When using FlashROM combined with AES, many subscription-based applications or device serialization applications are possible. The FROM configurator found in the Libero SoC Catalog supports easy management of the FlashROM contents, even over large numbers of devices. The FROM configurator can support FlashROM contents that contain the following:

- Static values
- Random numbers
- Values read from a file
- Independent updates of each page

In addition, auto-incrementing of fields is possible. In applications where the FlashROM content is different for each device, you have the option to generate a single STAPL file for all the devices or individual serialization files for each device. For more information on how to generate the FlashROM content for device serialization, refer to the "FlashROM in Microsemi's Low Power Flash Devices" section on page 119.

Libero SoC includes a unique tool to support the generation and management of FlashROM and FPGA programming files. This tool is called FlashPoint.

Depending on the applications, designers can use the FlashPoint software to generate a STAPL file with different contents. In each case, optional AES encryption and/or different security settings can be set.

In Designer, when you click the Programming File icon, FlashPoint launches, and you can generate STAPL file(s) with four different cases (Figure 14-4 on page 356). When the serialization feature is used during the configuration of FlashROM, you can generate a single STAPL file that will program all the devices or an individual STAPL file for each device.

The following cases present the FPGA core and FlashROM programming file combinations that can be used for different applications. In each case, you can set the optional security settings (FlashLock Pass Key and/or AES Key) depending on the application.

1. A single STAPL file or multiple STAPL files with multiple FlashROM contents and the FPGA core content. A single STAPL file will be generated if the device serialization feature is not used. You can program the whole FlashROM or selectively program individual pages.
2. A single STAPL file for the FPGA core content

3. A single STAPL file or multiple STAPL files with multiple FlashROM contents. A single STAPL file will be generated if the device serialization feature is not used. You can program the whole FlashROM or selectively program individual pages.
4. A single STAPL file to configure the security settings for the device, such as the AES Key and/or Pass Key.

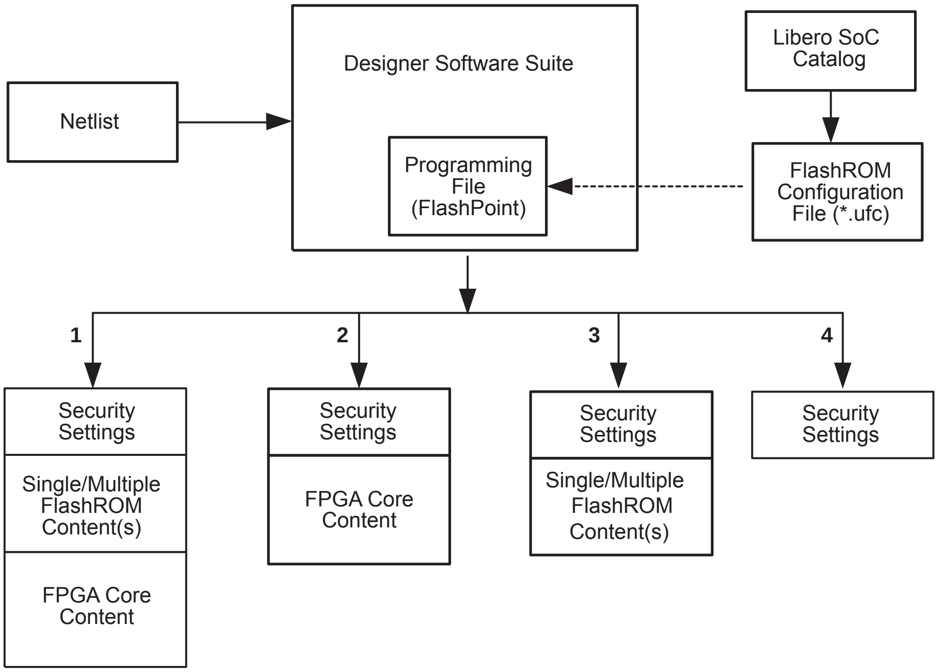


***Figure 14-4* • Flexible Programming File Generation for Different Applications**

Programming Solution

For device programming, any IEEE 1532–compliant programmer can be used; however, the FlashPro4/3/3X programmer must be used to control the low power flash device's rich security features and FlashROM programming options. The FlashPro4/3/3X programmer is a low-cost portable programmer for the Microsemi flash families. It can also be used with a powered USB hub for parallel programming. General specifications for the FlashPro4/3/3X programmer are as follows:

- Programming clock – TCK is used with a maximum frequency of 20 MHz, and the default frequency is 4 MHz.
- Programming file – STAPL
- Daisy chain – Supported. You can use the ChainBuilder software to build the programming file for the chain.
- Parallel programming – Supported. Multiple FlashPro4/3/3X programmers can be connected together using a powered USB hub or through the multiple USB ports on the PC.
- Power supply – The target board must provide VCC, VCCI, VPUMP, and VJTAG during programming. However, if there is only one device on the target board, the FlashPro4/3/3X programmer can generate the required VPUMP voltage from the USB port.

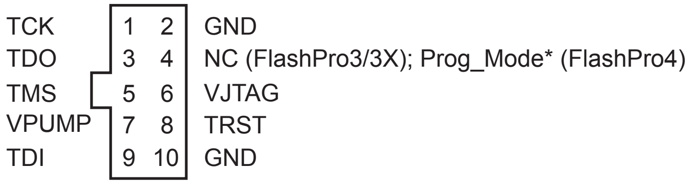
ISP Programming Header Information

The FlashPro4/3/3X programming cable connector can be connected with a 10-pin, 0.1"-pitch programming header. The recommended programming headers are manufactured by AMP (103310-1) and 3M (2510-6002UB). If you have limited board space, you can use a compact programming header manufactured by Samtec (FTSH-105-01-L-D-K). Using this compact programming header, you are required to order an additional header adapter manufactured by Microsemi SoC Products Group (FP3-10PIN-ADAPTER-KIT).

Existing ProASIC$^{\text{PLUS}}$ family customers who are using the Samtec Small Programming Header (FTSH-113-01-L-D-K) and are planning to migrate to IGLOO or ProASIC3 devices can also use FP3-10PIN-ADAPTER-KIT.

***Table 14-3* • Programming Header Ordering Codes**

Manufacturer	Part Number	Description
AMP	103310-1	10-pin, 0.1"-pitch cable header (right-angle PCB mount angle)
3M	2510-6002UB	10-pin, 0.1"-pitch cable header (straight PCB mount angle)
Samtec	FTSH-113-01-L-D-K	Small programming header supported by FlashPro and Silicon Sculptor
Samtec	FTSH-105-01-L-D-K	Compact programming header
Samtec	FFSD-05-D-06.00-01-N	10-pin cable with 50 mil pitch sockets; included in FP3-10PIN-ADAPTER-KIT.
Microsemi	FP3-10PIN-ADAPTER-KIT	Transition adapter kit to allow FP3 to be connected to a micro 10-pin header (50 mil pitch). Includes a 6 inch Samtec FFSD-05-D-06.00-01-N cable in the kit. The transition adapter board was previously offered as FP3-26PIN-ADAPTER and includes a 26-pin adapter for design transitions from ProASIC$^{\text{PLUS}}$ based boards to ProASIC3 based boards.



Note: **Prog_Mode on FlashPro4 is an output signal that goes High during device programming and returns to Low when programming is complete. This signal can be used to drive a system to provide a 1.5 V programming signal to IGLOO nano, ProASIC3L, and RT ProASIC3 devices that can run with 1.2 V core voltage but require 1.5 V for programming. IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only), but IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V.*

***Figure 14-5* • Programming Header (top view)**

Table 14-4 **• Programming Header Pin Numbers and Description**

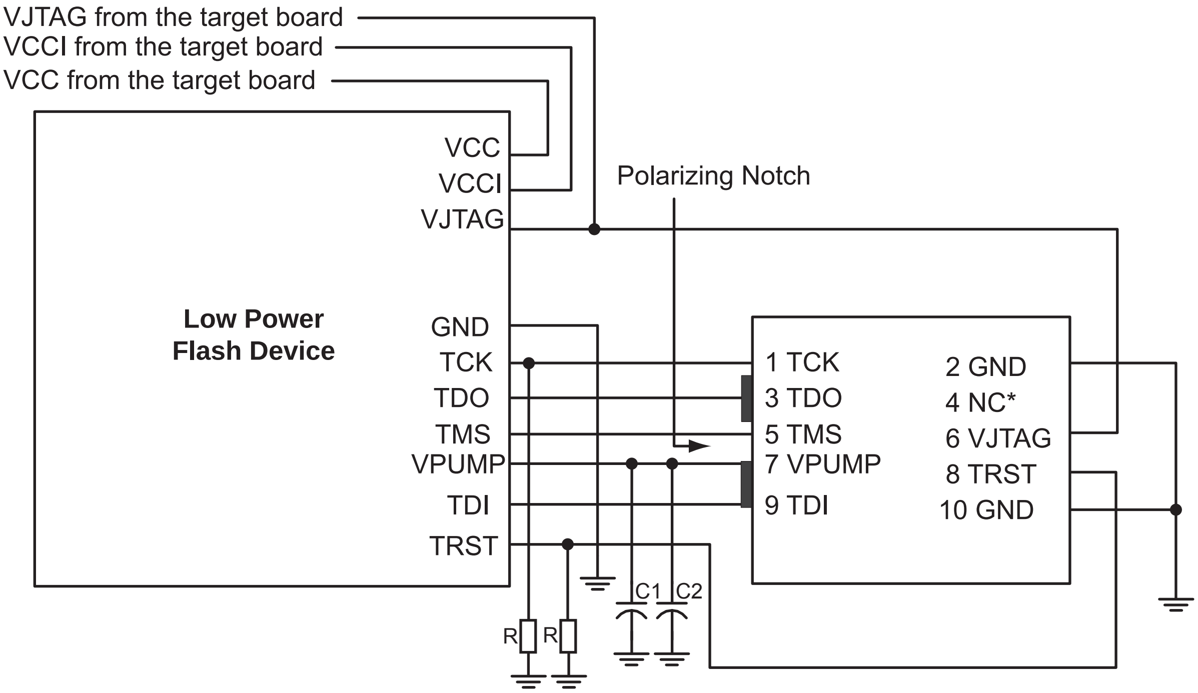
Pin	Signal	Source	Description
1	TCK	Programmer	JTAG Clock
2	GND[1]	–	Signal Reference
3	TDO	Target Board	Test Data Output
4	NC	–	No Connect (FlashPro3/3X); Prog_Mode (FlashPro4). See note associated with Figure 14-5 on page 357 regarding Prog_Mode on FlashPro4.
5	TMS	Programmer	Test Mode Select
6	VJTAG	Target Board	JTAG Supply Voltage
7	VPUMP[2]	Programmer/Target Board	Programming Supply Voltage
8	nTRST	Programmer	JTAG Test Reset (Hi-Z with 10 kΩ pull-down, HIGH, LOW, or toggling)
9	TDI	Programmer	Test Data Input
10	GND[1]	–	Signal Reference

Notes:

1. *Both GND pins must be connected.*
2. *FlashPro4/3/3X can provide VPUMP if there is only one device on the target board.*

Board-Level Considerations

A bypass capacitor is required from VPUMP to GND for all low power flash devices during programming. This bypass capacitor protects the devices from voltage spikes that may occur on the VPUMP supplies during the erase and programming cycles. Refer to the "Pin Descriptions and Packaging" chapter of the appropriate device datasheet for specific recommendations. For proper programming, 0.01 µF and 0.33 µF capacitors (both rated at 16 V) are to be connected in parallel across VPUMP and GND, and positioned as close to the FPGA pins as possible. The bypass capacitor must be placed within 2.5 cm of the device pins.



Note: **NC (FlashPro3/3X); Prog_Mode (FlashPro4). Prog_Mode on FlashPro4 is an output signal that goes High during device programming and returns to Low when programming is complete. This signal can be used to drive a system to provide a 1.5 V programming signal to IGLOO nano, ProASIC3L, and RT ProASIC3 devices that can run with 1.2 V core voltage but require 1.5 V for programming. IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only), but IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V.*

***Figure 14-6* • Board Layout and Programming Header Top View**

Troubleshooting Signal Integrity

Symptoms of a Signal Integrity Problem

A signal integrity problem can manifest itself in many ways. The problem may show up as extra or dropped bits during serial communication, changing the meaning of the communication. There is a normal variation of threshold voltage and frequency response between parts even from the same lot. Because of this, the effects of signal integrity may not always affect different devices on the same board in the same way. Sometimes, replacing a device appears to make signal integrity problems go away, but this is just masking the problem. Different parts on identical boards will exhibit the same problem sooner or later. It is important to fix signal integrity problems early. Unless the signal integrity problems are severe enough to completely block all communication between the device and the programmer, they may show up as subtle problems. Some of the FlashPro4/3/3X exit codes that are caused by signal integrity problems are listed below. Signal integrity problems are not the only possible cause of these errors, but this list is intended to show where problems can occur. FlashPro4/3/3X allows TCK to be lowered from 6 MHz down to 1 MHz to allow you to address some signal integrity problems that may occur with impedance mismatching at higher frequencies. Customers are expected to troubleshoot board-level signal integrity issues by measuring voltages and taking scope plots.

Scan Chain Failure

Normally, the FlashPro4/3/3X Scan Chain command expects to see 0x1 on the TDO pin. If the command reports reading 0x0 or 0x3, it is seeing the TDO pin stuck at 0 or 1. The only time the TDO pin comes out of tristate is when the JTAG TAP state machine is in the Shift-IR or Shift-DR state. If noise or reflections on the TCK or TMS lines have disrupted the correct state transitions, the device's TAP state controller might not be in one of these two states when the programmer tries to read the device. When this happens, the output is floating when it is read and does not match the expected data value. This can also be caused by a broken TDO net. Only a small amount of data is read from the device during the Scan Chain command, so marginal problems may not always show up during this command. Occasionally a faulty programmer can cause intermittent scan chain failures.

Exit 11

This error occurs during the verify stage of programming a device. After programming the design into the device, the device is verified to ensure it is programmed correctly. The verification is done by shifting the programming data into the device. An internal comparison is performed within the device to verify that all switches are programmed correctly. Noise induced by poor signal integrity can disrupt the writes and reads or the verification process and produce a verification error. While technically a verification error, the root cause is often related to signal integrity.

Refer to the *FlashPro User's Guide* for other error messages and solutions. For the most up-to-date known issues and solutions, refer to http://www.microsemi.com/soc/support.

Conclusion

IGLOO, ProASIC3, SmartFusion, and Fusion devices offer a low-cost, single-chip solution that is live at power-up through nonvolatile flash technology. The FlashLock Pass Key and 128-bit AES Key security features enable secure ISP in an untrusted environment. On-chip FlashROM enables a host of new applications, including device serialization, subscription-based applications, and IP addressing. Additionally, as the FlashROM is nonvolatile, all of these services can be provided without battery backup.

Related Documents

User's Guides

FlashPro User's Guide

http://www.microsemi.com/soc/documents/flashpro_ug.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	This chapter will now be published standalone as an application note in addition to being part of the IGLOO/ProASIC3/Fusion FPGA fabric user's guides (SAR 38769).	N/A
	The "ISP Programming Header Information" section was revised to update the description of FP3-10PIN-ADAPTER-KIT in Table 14-3 • Programming Header Ordering Codes, clarifying that it is the adapter kit used for ProASIC$^{\underline{PLUS}}$ based boards, and also for ProASIC3 based boards where a compact programming header is being used (SAR 36779).	357
June 2011	The VPUMP programming mode voltage was corrected in Table 14-2 • Power Supplies. The correct value is 3.15 V to 3.45 V (SAR 30668).	351
	The notes associated with Figure 14-5 • Programming Header (top view) and Figure 14-6 • Board Layout and Programming Header Top View were revised to make clear the fact that IGLOO nano V2 devices can be programmed at 1.2 V (SAR 30787).	357, 359
	Figure 14-6 • Board Layout and Programming Header Top View was revised to include resistors tying TCK and TRST to GND. Microsemi recommends tying off TCK and TRST to GND if JTAG is not used (SAR 22921). RT ProASIC3 was added to the list of device families.	359
	In the "ISP Programming Header Information" section, the kit for adapting ProASIC$^{\underline{PLUS}}$ devices was changed from FP3-10PIN-ADAPTER-KIT to FP3-26PIN-ADAPTER-KIT (SAR 20878).	357
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	References to FlashPro4 and FlashPro3X were added to this chapter, giving distinctions between them. References to SmartGen were deleted and replaced with Libero IDE Catalog.	N/A
	The "ISP Architecture" section was revised to indicate that V2 devices can be programmed at 1.2 V VCC with FlashPro4.	349
	SmartFusion was added to Table 14-1 • Flash-Based FPGAs Supporting ISP.	350
	The "Programming Voltage (VPUMP) and VJTAG" section was revised and 1.2 V was added to Table 14-2 • Power Supplies.	351
	The "Nonvolatile Memory (NVM) Programming Voltage" section is new.	351
	Cortex-M3 was added to the "Cortex-M1 and Cortex-M3 Device Security" section.	353
	In the "ISP Programming Header Information" section, the additional header adapter ordering number was changed from FP3-26PIN-ADAPTER to FP3-10PIN-ADAPTER-KIT, which contains 26-pin migration capability.	357
	The description of NC was updated in Figure 14-5 • Programming Header (top view), Table 14-4 • Programming Header Pin Numbers and Description and Figure 14-6 • Board Layout and Programming Header Top View.	357, 358
	The "Symptoms of a Signal Integrity Problem" section was revised to add that customers are expected to troubleshoot board-level signal integrity issues by measuring voltages and taking scope plots. "FlashPro4/3/3X allows TCK to be lowered from 6 MHz down to 1 MHz to allow you to address some signal integrity problems" formerly read, "from 24 MHz down to 1 MHz." "The Scan Chain command expects to see 0x2" was changed to 0x1.	359

Date	Changes	Page
July 2010 (continued)	The "Chain Integrity Test Error Analyze Chain Failure" section was renamed to the "Scan Chain Failure" section, and the Analyze Chain command was changed to Scan Chain. It was noted that occasionally a faulty programmer can cause scan chain failures.	360
v1.5 (August 2009)	The "CoreMP7 Device Security" section was removed from "Security in ARM-Enabled Low Power Flash Devices", since M7-enabled devices are no longer supported.	353
v1.4 (December 2008)	The "ISP Architecture" section was revised to include information about core voltage for IGLOO V2 and ProASIC3L devices, as well as 50 mV increments allowable in Designer software.	349
	IGLOO nano and ProASIC3 nano devices were added to Table 14-1 • Flash-Based FPGAs Supporting ISP.	350
	A second capacitor was added to Figure 14-6 • Board Layout and Programming Header Top View.	359
v1.3 (October 2008)	The "ISP Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	350
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 14-1 • Flash-Based FPGAs Supporting ISP: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	350
v1.1 (March 2008)	The "ISP Architecture" section was updated to included the IGLOO PLUS family in the discussion of family-specific support. The text, "When 1.2 V is used, the device can be reprogrammed in-system at 1.5 V only," was revised to state, "Although the device can operate at 1.2 V core voltage, the device can only be reprogrammed when all supplies (VCC, VCCI, and VJTAG) are at 1.5 V."	349
	The "ISP Support in Flash-Based Devices" section and Table 14-1 • Flash-Based FPGAs Supporting ISP were updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	350
	The "Security" section was updated to mention that 15 k gate devices do not have a built-in 128-bit decryption core.	352
	Table 14-2 • Power Supplies was revised to remove the Normal Operation column and add a table note stating, "All supply voltages should be at 1.5 V or higher, regardless of the setting during normal operation."	351
	The "ISP Programming Header Information" section was revised to change FP3-26PIN-ADAPTER to FP3-10PIN-ADAPTER-KIT. Table 14-3 • Programming Header Ordering Codes was updated with the same change, as well as adding the part number FFSD-05-D-06.00-01-N, a 10-pin cable with 50-mil-pitch sockets.	357
	The "Board-Level Considerations" section was updated to describe connecting two capacitors in parallel across VPUMP and GND for proper programming.	359
v1.0 (January 2008)	Information was added to the "Programming Voltage (VPUMP) and VJTAG" section about the JTAG interface pin.	351
51900055-2/7.06	ACTgen was changed to SmartGen.	N/A
	In Figure 14-6 • Board Layout and Programming Header Top View, the order of the text was changed to: VJTAG from the target board VCCI from the target board VCC from the target board	359

15 – Microprocessor Programming of Microsemi's Low Power Flash Devices

Introduction

The Fusion, IGLOO, and ProASIC3 families of flash FPGAs support in-system programming (ISP) with the use of a microprocessor. Flash-based FPGAs store their configuration information in the actual cells within the FPGA fabric. SRAM-based devices need an external configuration memory, and hybrid nonvolatile devices store the configuration in a flash memory inside the same package as the SRAM FPGA. Since the programming of a true flash FPGA is simpler, requiring only one stage, it makes sense that programming with a microprocessor in-system should be simpler than with other SRAM FPGAs. This reduces bill-of-materials costs and printed circuit board (PCB) area, and increases system reliability.

Nonvolatile flash technology also gives the low power flash devices the advantage of a secure, low power, live-at-power-up, and single-chip solution. Low power flash devices are reprogrammable and offer time-to-market benefits at an ASIC-level unit cost. These features enable engineers to create high-density systems using existing ASIC or FPGA design flows and tools.

This document is an introduction to microprocessor programming only. To explain the difference between the options available, user's guides for DirectC and STAPL provide more detail on implementing each style.

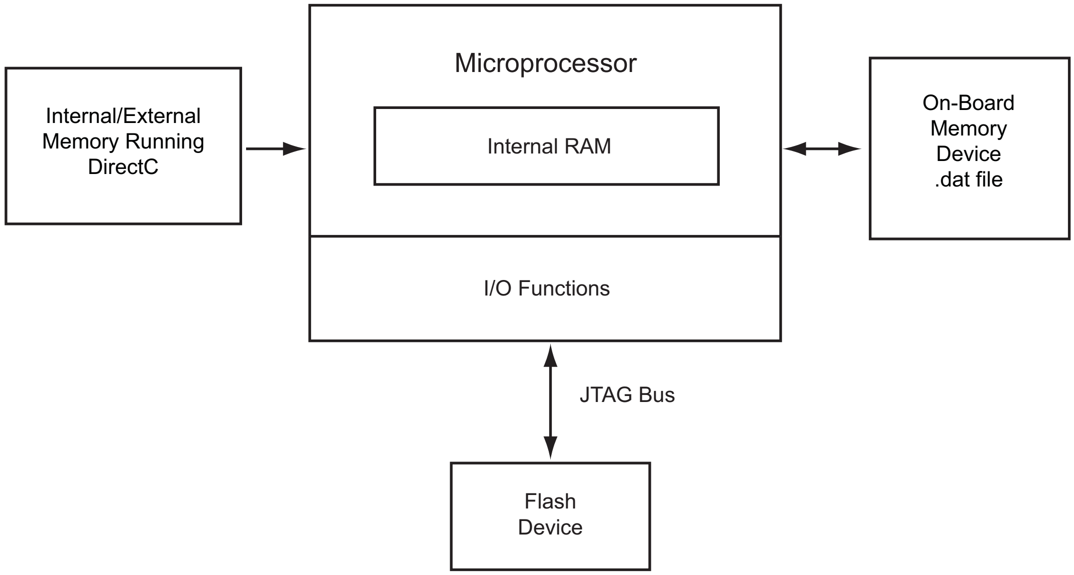

Internal/External Memory Running DirectC
Microprocessor
Internal RAM
I/O Functions
On-Board Memory Device .dat file
JTAG Bus
Flash Device

Figure 15-1 • **ISP Using Microprocessor**

Microprocessor Programming Support in Flash Devices

The flash-based FPGAs listed in Table 15-1 support programming with a microprocessor and the functions described in this document.

***Table 15-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

*Note: *The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 15-1. Where the information applies to only one device or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 15-1. Where the information applies to only one device or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Programming Algorithm

JTAG Interface

The low power flash families are fully compliant with the IEEE 1149.1 (JTAG) standard. They support all the mandatory boundary scan instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS) as well as six optional public instructions (USERCODE, IDCODE, HIGHZ, and CLAMP).

IEEE 1532

The low power flash families are also fully compliant with the IEEE 1532 programming standard. The IEEE 1532 standard adds programming instructions and associated data registers to devices that comply with the IEEE 1149.1 standard (JTAG). These instructions and registers extend the capabilities of the IEEE 1149.1 standard such that the Test Access Port (TAP) can be used for configuration activities. The IEEE 1532 standard greatly simplifies the programming algorithm, reducing the amount of time needed to implement microprocessor ISP.

Implementation Overview

To implement device programming with a microprocessor, the user should first download the C-based STAPL player or DirectC code from the Microsemi SoC Products Group website. Refer to the website for future updates regarding the STAPL player and DirectC code.

http://www.microsemi.com/soc/download/program_debug/stapl/default.aspx

http://www.microsemi.com/soc/download/program_debug/directc/default.aspx

Using the easy-to-follow user's guide, create the low-level application programming interface (API) to provide the necessary basic functions. These API functions act as the interface between the programming software and the actual hardware (Figure 15-2).

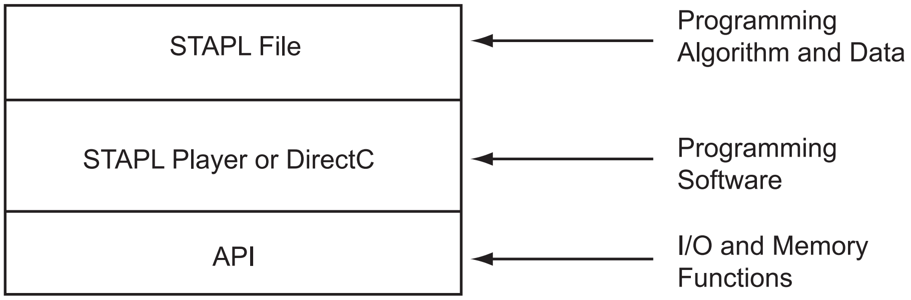


Figure 15-2 **• Device Programming Code Relationship**

The API is then linked with the STAPL player or DirectC and compiled using the microprocessor's compiler. Once the entire code is compiled, the user must download the resulting binary into the MCU system's program memory (such as ROM, EEPROM, or flash). The system is now ready for programming.

To program a design into the FPGA, the user creates a bitstream or STAPL file using the Microsemi Designer software, downloads it into the MCU system's volatile memory, and activates the stored programming binary file (Figure 15-3 on page 366). Once the programming is completed, the bitstream or STAPL file can be removed from the system, as the configuration profile is stored in the flash FPGA fabric and does not need to be reloaded at every system power-on.

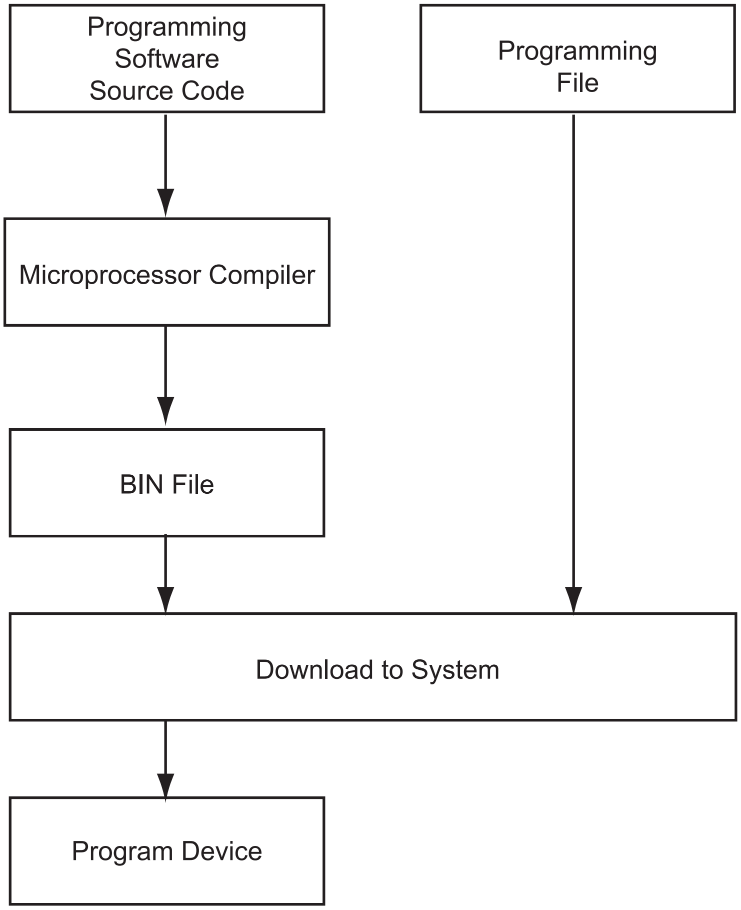


Figure 15-3 **• MCU FPGA Programming Model**

FlashROM

Microsemi low power flash devices have 1 kbit of user-accessible, nonvolatile, FlashROM on-chip. This nonvolatile FlashROM can be programmed along with the core or on its own using the standard IEEE 1532 JTAG programming interface.

The FlashROM is architected as eight pages of 128 bits. Each page can be individually programmed (erased and written). Additionally, on-chip AES security decryption can be used selectively to load data securely into the FlashROM (e.g., over public or private networks, such as the Internet). Refer to the "FlashROM in Microsemi's Low Power Flash Devices" section on page 119.

STAPL vs. DirectC

Programming the low power flash devices is performed using DirectC or the STAPL player. Both tools use the STAPL file as an input. DirectC is a compiled language, whereas STAPL is an interpreted language. Microprocessors will be able to load the FPGA using DirectC much more quickly than STAPL. This speed advantage becomes more apparent when lower clock speeds of 8- or 16-bit microprocessors are used. DirectC also requires less memory than STAPL, since the programming algorithm is directly implemented. STAPL does have one advantage over DirectC—the ability to upgrade. When a new programming algorithm is required, the STAPL user simply needs to regenerate a STAPL file using the latest version of the Designer software and download it to the system. The DirectC user must download the latest version of DirectC from Microsemi, compile everything, and download the result into the system (Figure 15-4).

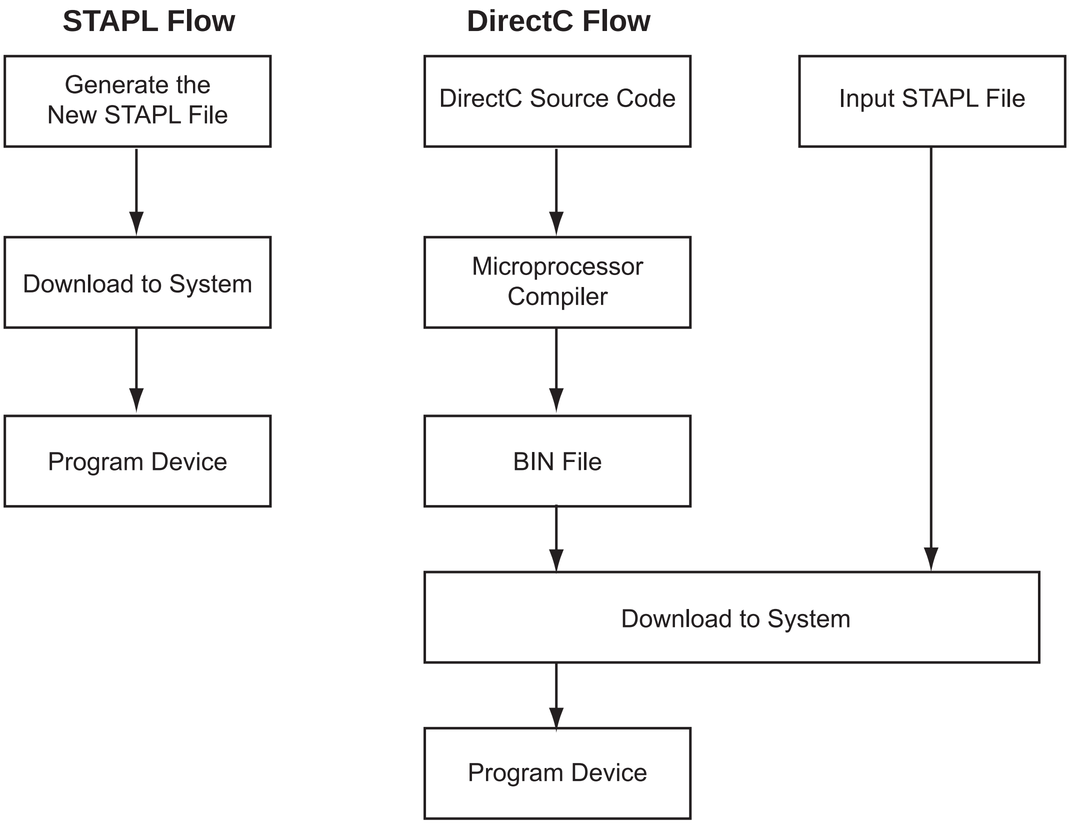


Figure 15-4 • **STAPL vs. DirectC**

Remote Upgrade via TCP/IP

Transmission Control Protocol (TCP) provides a reliable bitstream transfer service between two endpoints on a network. TCP depends on Internet Protocol (IP) to move packets around the network on its behalf. TCP protects against data loss, data corruption, packet reordering, and data duplication by adding checksums and sequence numbers to transmitted data and, on the receiving side, sending back packets and acknowledging the receipt of data.

The system containing the low power flash device can be assigned an IP address when deployed in the field. When the device requires an update (core or FlashROM), the programming instructions along with the new programming data (AES-encrypted cipher text) can be sent over the Internet to the target system via the TCP/IP protocol. Once the MCU receives the instruction and data, it can proceed with the FPGA update. Low power flash devices support Message Authentication Code (MAC), which can be used to validate data for the target device. More details are given in the "Message Authentication Code (MAC) Validation/Authentication" section.

Hardware Requirement

To facilitate the programming of the low power flash families, the system must have a microprocessor (with access to the device JTAG pins) to process the programming algorithm, memory to store the programming algorithm, programming data, and the necessary programming voltage. Refer to the relevant datasheet for programming voltages.

Security

Encrypted Programming

As an additional security measure, the devices are equipped with AES decryption. AES works in two steps. The first step is to program a key into the devices in a secure or trusted programming center (such as Microsemi SoC Products Group In-House Programming (IHP) center). The second step is to encrypt any programming files with the same encryption key. The encrypted programming file will only work with the devices that have the same key. The AES used in the low power flash families is the 128-bit AES decryption engine (Rijndael algorithm).

Message Authentication Code (MAC) Validation/Authentication

As part of the AES decryption flow, the devices are equipped with a MAC validation/authentication system. MAC is an authentication tag, also called a checksum, derived by applying an on-chip authentication scheme to a STAPL file as it is loaded into the FPGA. MACs are computed and verified with the same key so they can only be verified by the intended recipient. When the MCU system receives the AES-encrypted programming data (cipher text), it can validate the data by loading it into the FPGA and performing a MAC verification prior to loading the data, via a second programming pass, into the FPGA core cells. This prevents erroneous or corrupt data from getting into the FPGA.

Low power flash devices with AES and MAC are superior to devices with only DES or 3DES encryption. Because the MAC verifies the correctness of the data, the FPGA is protected from erroneous loading of invalid programming data that could damage a device (Figure 15-5 on page 369).

The AES with MAC enables field updates over public networks without fear of having the design stolen. An encrypted programming file can only work on devices with the correct key, rendering any stolen files useless to the thief. To learn more about the low power flash devices’ security features, refer to the "Security in Low Power Flash Devices" section on page 323.

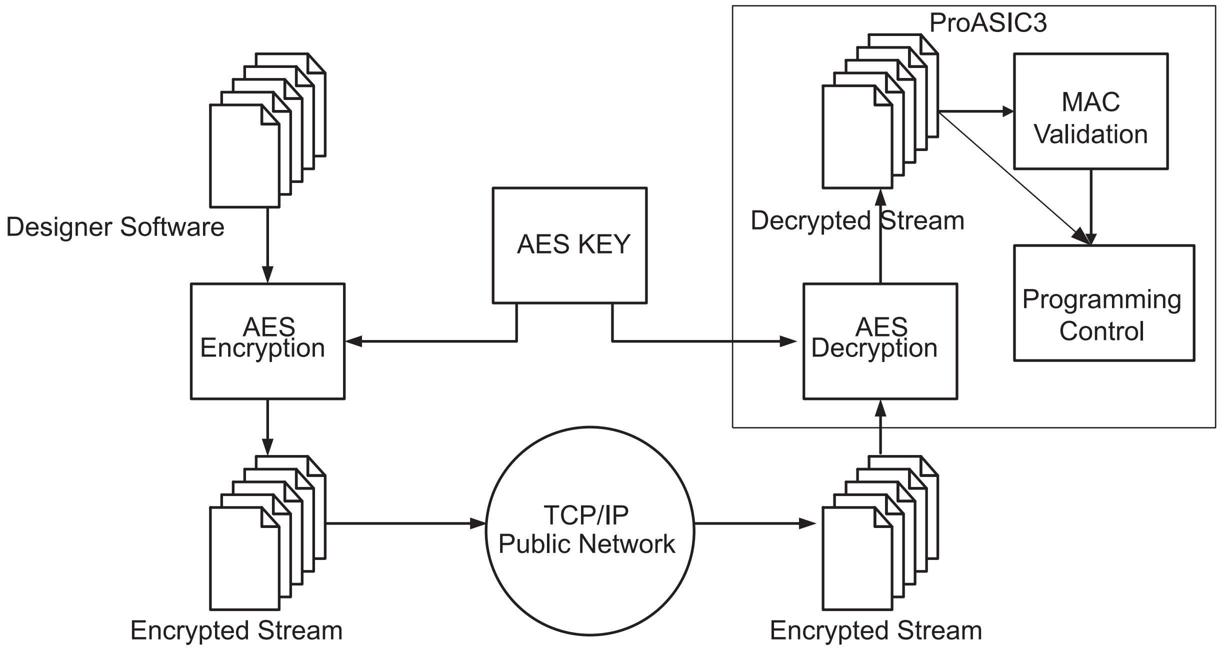


Figure 15-5 **• ProASIC3 Device Encryption Flow**

Conclusion

The Fusion, IGLOO, and ProASIC3 FPGAs are ideal for applications that require field upgrades. The single-chip devices save board space by eliminating the need for EEPROM. The built-in AES with MAC enables transmission of programming data over any network without fear of design theft. Fusion, IGLOO, and ProASIC3 FPGAs are IEEE 1532–compliant and support STAPL, making the target programming software easy to implement.

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
September 2012	The "Security" section was modified to clarify that Microsemi does not support read-back of FPGA core-programmed data (SAR 41235).	368
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 15-1 • Flash-Based FPGAs.	364
v1.3 (October 2008)	The "Microprocessor Programming Support in Flash Devices" section was revised to include new families and make the information more concise.	364
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 15-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	364
v1.1 (March 2008)	The "Microprocessor Programming Support in Flash Devices" section was updated to include information on the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	364

16 – Boundary Scan in Low Power Flash Devices

Boundary Scan

Low power flash devices are compatible with IEEE Standard 1149.1, which defines a hardware architecture and the set of mechanisms for boundary scan testing. JTAG operations are used during boundary scan testing.

The basic boundary scan logic circuit is composed of the TAP controller, test data registers, and instruction register (Figure 16-2 on page 374).

Low power flash devices support three types of test data registers: bypass, device identification, and boundary scan. The bypass register is selected when no other register needs to be accessed in a device. This speeds up test data transfer to other devices in a test data path. The 32-bit device identification register is a shift register with four fields (LSB, ID number, part number, and version). The boundary scan register observes and controls the state of each I/O pin. Each I/O cell has three boundary scan register cells, each with serial-in, serial-out, parallel-in, and parallel-out pins.

TAP Controller State Machine

The TAP controller is a 4-bit state machine (16 states) that operates as shown in Figure 16-1.

The 1s and 0s represent the values that must be present on TMS at a rising edge of TCK for the given state transition to occur. IR and DR indicate that the instruction register or the data register is operating in that state.

The TAP controller receives two control inputs (TMS and TCK) and generates control and clock signals for the rest of the test logic architecture. On power-up, the TAP controller enters the Test-Logic-Reset state. To guarantee a reset of the controller from any of the possible states, TMS must remain HIGH for five TCK cycles. The TRST pin can also be used to asynchronously place the TAP controller in the Test-Logic-Reset state.

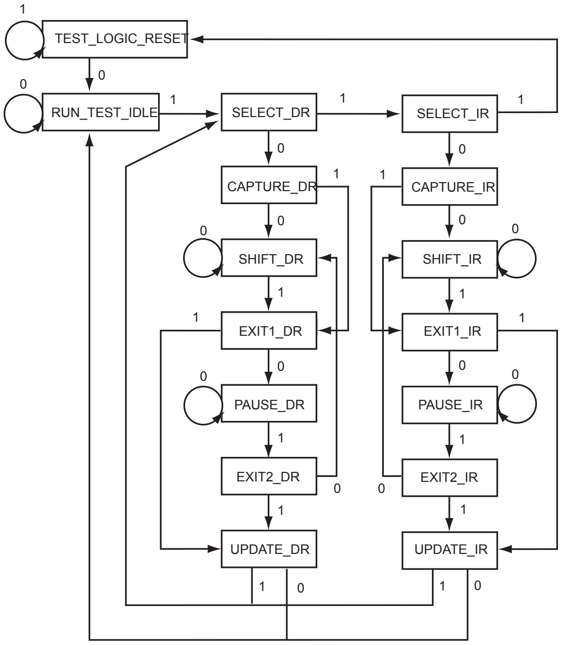


Figure 16-1 • **TAP Controller State Machine**

Microsemi's Flash Devices Support the JTAG Feature

The flash-based FPGAs listed in Table 16-1 support the JTAG feature and the functions described in this document.

***Table 16-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC®3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

*Note: *The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 16-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 16-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Boundary Scan Support in Low Power Devices

The information in this document applies to all Fusion, IGLOO, and ProASIC3 devices. For IGLOO, IGLOO PLUS, and ProASIC3L devices, the Flash*Freeze pin must be deasserted for successful boundary scan operations. Devices cannot enter JTAG mode directly from Flash*Freeze mode.

Boundary Scan Opcodes

Low power flash devices support all mandatory IEEE 1149.1 instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS) and the optional IDCODE instruction (Table 16-2).

***Table 16-2* • Boundary Scan Opcodes**

	Hex Opcode
EXTEST	00
HIGHZ	07
USERCODE	0E
SAMPLE/PRELOAD	01
IDCODE	0F
CLAMP	05
BYPASS	FF

Boundary Scan Chain

The serial pins are used to serially connect all the boundary scan register cells in a device into a boundary scan register chain (Figure 16-2 on page 374), which starts at the TDI pin and ends at the TDO pin. The parallel ports are connected to the internal core logic I/O tile and the input, output, and control ports of an I/O buffer to capture and load data into the register to control or observe the logic state of each I/O.

Each test section is accessed through the TAP, which has five associated pins: TCK (test clock input), TDI, TDO (test data input and output), TMS (test mode selector), and TRST (test reset input). TMS, TDI, and TRST are equipped with pull-up resistors to ensure proper operation when no input data is supplied to them. These pins are dedicated for boundary scan test usage. Refer to the "JTAG Pins" section in the "Pin Descriptions and Packaging" chapter of the appropriate device datasheet for pull-up/-down recommendations for TCK and TRST pins. Pull-down recommendations are also given in Table 16-3 on page 374

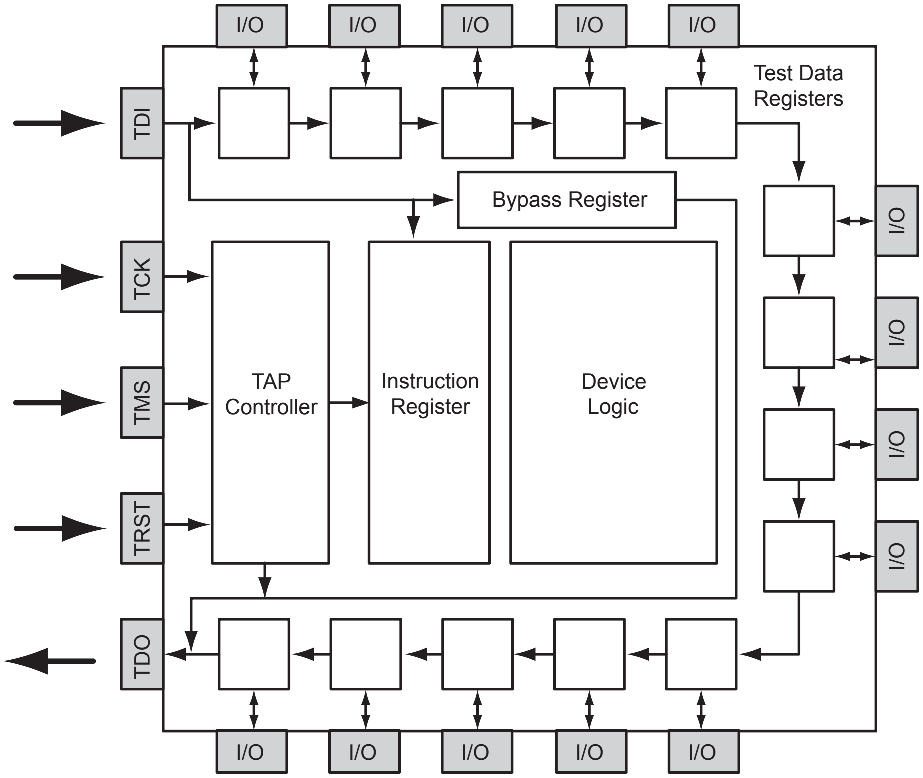


Figure 16-2 **• Boundary Scan Chain**

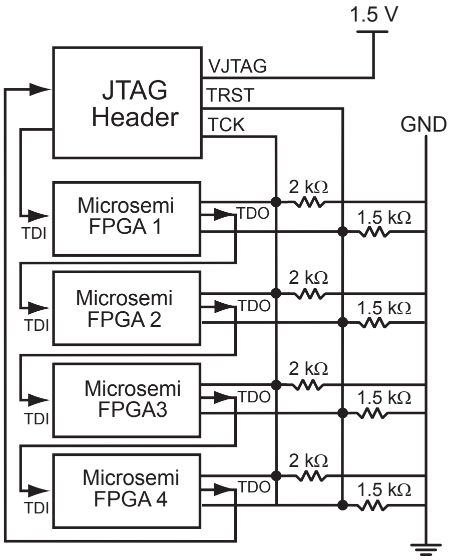
Board-Level Recommendations

Table 16-3 gives pull-down recommendations for the TRST and TCK pins.

Table 16-3 **• TRST and TCK Pull-Down Recommendations**

VJTAG	**Tie-Off Resistance***
VJTAG at 3.3 V	200 Ω to 1 kΩ
VJTAG at 2.5 V	200 Ω to 1 kΩ
VJTAG at 1.8 V	500 Ω to 1 kΩ
VJTAG at 1.5 V	500 Ω to 1 kΩ
VJTAG at 1.2 V	TBD

Note: Equivalent parallel resistance if more than one device is on JTAG chain (Figure 16-3)



Note: TCK is correctly wired with an equivalent tie-off resistance of 500 Ω, which satisfies the table for VJTAG of 1.5 V. The resistor values for TRST are not appropriate in this case, as the tie-off resistance of 375 Ω is below the recommended minimum for VJTAG = 1.5 V, but would be appropriate for a VJTAG setting of 2.5 V or 3.3 V.

***Figure 16-3* • Parallel Resistance on JTAG Chain of Devices**

Advanced Boundary Scan Register Settings

You will not be able to control the order in which I/Os are released from boundary scan control. Testing has produced cases where, depending on I/O placement and FPGA routing, a 5 ns glitch has been seen on exiting programming mode. The following setting is recommended to prevent such I/O glitches:

1. In the FlashPro software, configure the advanced BSR settings for **Specify I/O Settings During Programming**.
2. Set the input BSR cell to **Low** for the input I/O.

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	In the "Boundary Scan Chain" section, the reference made to the datasheet for pull-up/-down recommendations was changed to mention TCK and TRST pins rather than TDO and TCK pins. TDO is an output, so no pull resistor is needed (SAR 35937).	373
	The "Advanced Boundary Scan Register Settings" section is new (SAR 38432).	375
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	Table 16-3 • TRST and TCK Pull-Down Recommendations was revised to add VJTAG at 1.2 V.	374
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 16-1 • Flash-Based FPGAs.	372
v1.3 (October 2008)	The "Boundary Scan Support in Low Power Devices" section was revised to include new families and make the information more concise.	373
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 16-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	372
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices.	N/A
	The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	372

17 – UJTAG Applications in Microsemi's Low Power Flash Devices

Introduction

In Fusion, IGLOO, and ProASIC3 devices, there is bidirectional access from the JTAG port to the core VersaTiles during normal operation of the device (Figure 17-1). User JTAG (UJTAG) is the ability for the design to use the JTAG ports for access to the device for updates, etc. While regular JTAG is used, the UJTAG tiles, located at the southeast area of the die, are directly connected to the JTAG Test Access Port (TAP) Controller in normal operating mode. As a result, all the functional blocks of the device, such as Clock Conditioning Circuits (CCCs) with PLLs, SRAM blocks, embedded FlashROM, flash memory blocks, and I/O tiles, can be reached via the JTAG ports. The UJTAG functionality is available by instantiating the UJTAG macro directly in the source code of a design. Access to the FPGA core VersaTiles from the JTAG ports enables users to implement different applications using the TAP Controller (JTAG port). This document introduces the UJTAG tile functionality and discusses a few application examples. However, the possible applications are not limited to what is presented in this document. UJTAG can serve different purposes in many designs as an elementary or auxiliary part of the design. For detailed usage information, refer to the "Boundary Scan in Low Power Flash Devices" section on page 371.

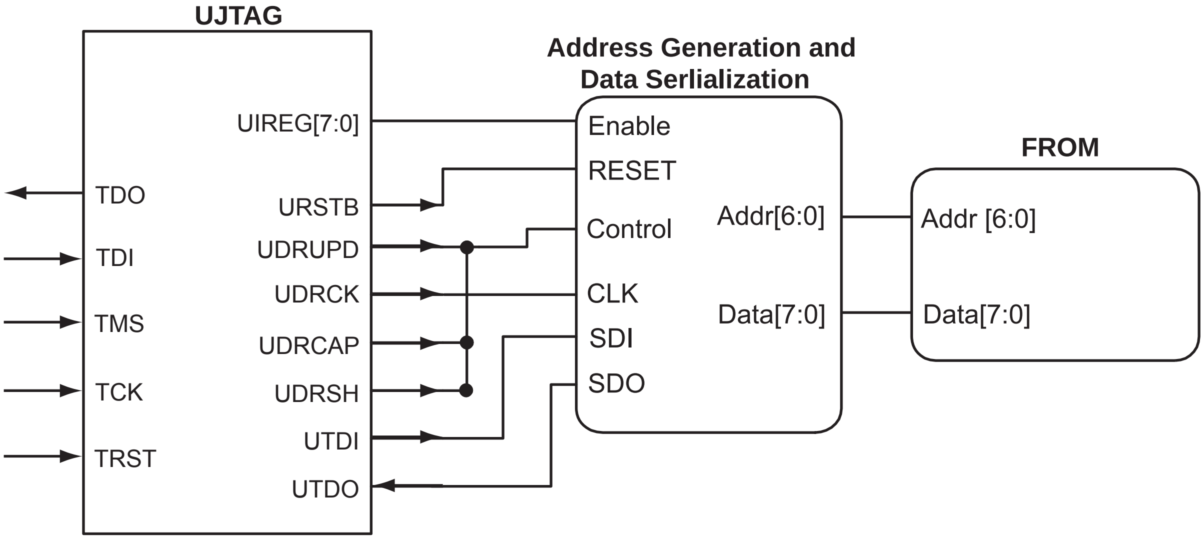


Figure 17-1 • **Block Diagram of Using UJTAG to Read FlashROM Contents**

UJTAG Support in Flash-Based Devices

The flash-based FPGAs listed in Table 17-1 support the UJTAG feature and the functions described in this document.

Table 17-1 • **Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

Note: **The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 17-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 17-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

UJTAG Macro

The UJTAG tiles can be instantiated in a design using the UJTAG macro from the Fusion, IGLOO, or ProASIC3 macro library. Note that "UJTAG" is a reserved name and cannot be used for any other user-defined blocks. A block symbol of the UJTAG tile macro is presented in Figure 17-2. In this figure, the ports on the left side of the block are connected to the JTAG TAP Controller, and the right-side ports are accessible by the FPGA core VersaTiles. The TDI, TMS, TDO, TCK, and TRST ports of UJTAG are only provided for design simulation purposes and should be treated as external signals in the design netlist. However, these ports must NOT be connected to any I/O buffer in the netlist. Figure 17-3 on page 380 illustrates the correct connection of the UJTAG macro to the user design netlist. Microsemi Designer software will automatically connect these ports to the TAP during place-and-route. Table 17-2 gives the port descriptions for the rest of the UJTAG ports:

Table 17-2 **• UJTAG Port Descriptions**

Port	Description
UIREG [7:0]	This 8-bit bus carries the contents of the JTAG Instruction Register of each device. Instruction Register values 16 to 127 are not reserved and can be employed as user-defined instructions.
URSTB	URSTB is an active-low signal and will be asserted when the TAP Controller is in Test-Logic-Reset mode. URSTB is asserted at power-up, and a power-on reset signal resets the TAP Controller. URSTB will stay asserted until an external TAP access changes the TAP Controller state.
UTDI	This port is directly connected to the TAP's TDI signal.
UTDO	This port is the user TDO output. Inputs to the UTDO port are sent to the TAP TDO output MUX when the IR address is in user range.
UDRSH	Active-high signal enabled in the ShiftDR TAP state
UDRCAP	Active-high signal enabled in the CaptureDR TAP state
UDRCK	This port is directly connected to the TAP's TCK signal.
UDRUPD	Active-high signal enabled in the UpdateDR TAP state

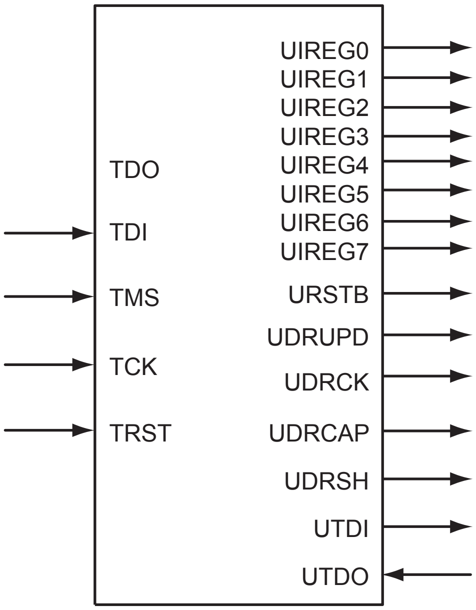
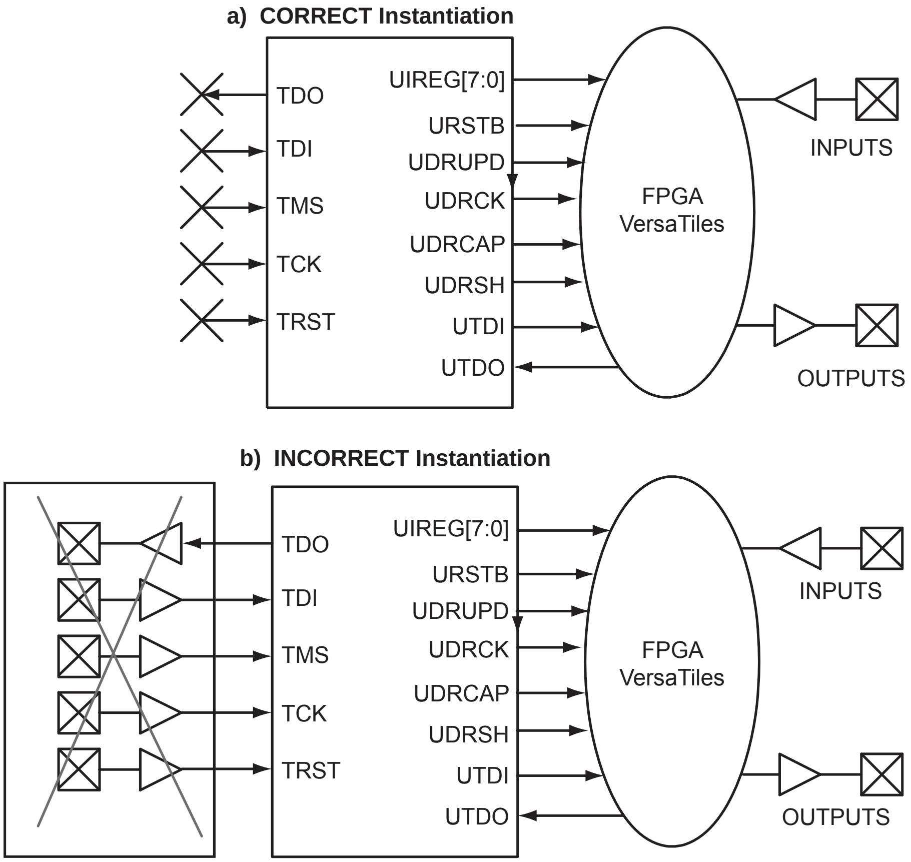


Figure 17-2 **• UJTAG Tile Block Symbol**



Note: Do not connect JTAG pins (TDO, TDI, TMS, TCK, or TRST) to I/Os in the design.

Figure 17-3 **• Connectivity Method of UJTAG Macro**

UJTAG Operation

There are a few basic functions of the UJTAG macro that users must understand before designing with it. The most important fundamental concept of the UJTAG design is its connection with the TAP Controller state machine.

TAP Controller State Machine

The 16 states of the TAP Controller state machine are shown in Figure 17-4 on page 381. The 1s and 0s, shown adjacent to the state transitions, represent the TMS values that must be present at the time of a rising TCK edge for a state transition to occur. In the states that include the letters "IR," the instruction register operates; in the states that contain the letters "DR," the test data register operates. The TAP Controller receives two control inputs, TMS and TCK, and generates control and clock signals for the rest of the test logic.

On power-up (or the assertion of TRST), the TAP Controller enters the Test-Logic-Reset state. To reset the controller from any other state, TMS must be held HIGH for at least five TCK cycles. After reset, the TAP state changes at the rising edge of TCK, based on the value of TMS.

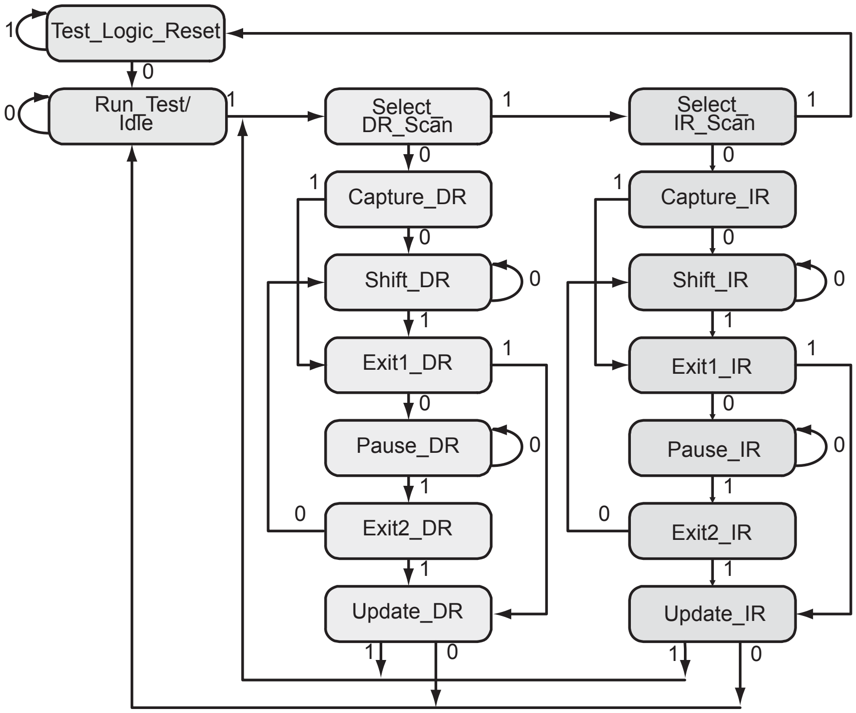


***Figure 17-4* • TAP Controller State Diagram**

UJTAG Port Usage

UIREG[7:0] hold the contents of the JTAG instruction register. The UIREG vector value is updated when the TAP Controller state machine enters the Update_IR state. Instructions 16 to 127 are user-defined and can be employed to encode multiple applications and commands within an application. Loading new instructions into the UIREG vector requires users to send appropriate logic to TMS to put the TAP Controller in a full IR cycle starting from the Select IR_Scan state and ending with the Update_IR state.

UTDI, UTDO, and UDRCK are directly connected to the JTAG TDI, TDO, and TCK ports, respectively. The TDI input can be used to provide either data (TAP Controller in the Shift_DR state) or the new contents of the instruction register (TAP Controller in the Shift_IR state).

UDRSH, UDRUPD, and UDRCAP are HIGH when the TAP Controller state machine is in the Shift_DR, Update_DR, and Capture_DR states, respectively. Therefore, they act as flags to indicate the stages of the data shift process. These flags are useful for applications in which blocks of data are shifted into the design from JTAG pins. For example, an active UDRSH can indicate that UTDI contains the data bitstream, and UDRUPD is a candidate for the end-of-data-stream flag.

As mentioned earlier, users should not connect the TDI, TDO, TCK, TMS, and TRST ports of the UJTAG macro to any port or net of the design netlist. The Designer software will automatically handle the port connection.

Typical UJTAG Applications

Bidirectional access to the JTAG port from VersaTiles—without putting the device into test mode—creates flexibility to implement many different applications. This section describes a few of these. All are based on importing/exporting data through the UJTAG tiles.

Clock Conditioning Circuitry—Dynamic Reconfiguration

In low power flash devices, CCCs, which include PLLs, can be configured dynamically through either an 81-bit embedded shift register or static flash programming switches. These 81 bits control all the characteristics of the CCC: routing MUX architectures, delay values, divider values, etc. Table 17-3 lists the 81 configuration bits in the CCC.

***Table 17-3* • Configuration Bits of Fusion, IGLOO, and ProASIC3 CCC Blocks**

Bit Number(s)	Control Function
80	RESET ENABLE
79	DYNCSEL
78	DYNBSEL
77	DYNASEL
<76:74>	VCOSEL [2:0]
73	STATCSEL
72	STATBSEL
71	STATASEL
<70:66>	DLYC [4:0]
<65:61>	DLYB {4:0]
<60:56>	DLYGLC [4:0]
<55:51>	DLYGLB [4:0]
<50:46>	DLYGLA [4:0]
45	XDLYSEL
<44:40>	FBDLY [4:0]
<39:38>	FBSEL
<37:35>	OCMUX [2:0]
<34:32>	OBMUX [2:0]
<31:29>	OAMUX [2:0]
<28:24>	OCDIV [4:0]
<23:19>	OBDIV [4:0]
<18:14>	OADIV [4:0]
<13:7>	FBDIV [6:0]
<6:0>	FINDIV [6:0]

The embedded 81-bit shift register (for the dynamic configuration of the CCC) is accessible to the VersaTiles, which, in turn, have access to the UJTAG tiles. Therefore, the CCC configuration shift register can receive and load the new configuration data stream from JTAG.

Dynamic reconfiguration eliminates the need to reprogram the device when reconfiguration of the CCC functional blocks is needed. The CCC configuration can be modified while the device continues to operate. Employing the UJTAG core requires the user to design a module to provide the configuration data and control the CCC configuration shift register. In essence, this is a user-designed TAP Controller requiring chip resources.

Similar reconfiguration capability exists in the ProASIC$^{\underline{PLUS}}$® family. The only difference is the number of shift register bits controlling the CCC (27 in ProASIC$^{\underline{PLUS}}$ and 81 in IGLOO, ProASIC3, and Fusion).

Fine Tuning

In some applications, design constants or parameters need to be modified after programming the original design. The tuning process can be done using the UJTAG tile without reprogramming the device with new values. If the parameters or constants of a design are stored in distributed registers or embedded SRAM blocks, the new values can be shifted onto the JTAG TAP Controller pins, replacing the old values. The UJTAG tile is used as the "bridge" for data transfer between the JTAG pins and the FPGA VersaTiles or SRAM logic. Figure 17-5 shows a flow chart example for fine-tuning application steps using the UJTAG tile.

In Figure 17-5, the TMS signal sets the TAP Controller state machine to the appropriate states. The flow mainly consists of two steps: a) shifting the defined instruction and b) shifting the new data. If the target parameter is constantly used in the design, the new data can be shifted into a temporary shift register from UTDI. The UDRSH output of UJTAG can be used as a shift-enable signal, and UDRCK is the shift clock to the shift register. Once the shift process is completed and the TAP Controller state is moved to the Update_DR state, the UDRUPD output of the UJTAG can latch the new parameter value from the temporary register into a permanent location. This avoids any interruption or malfunctioning during the serial shift of the new value.

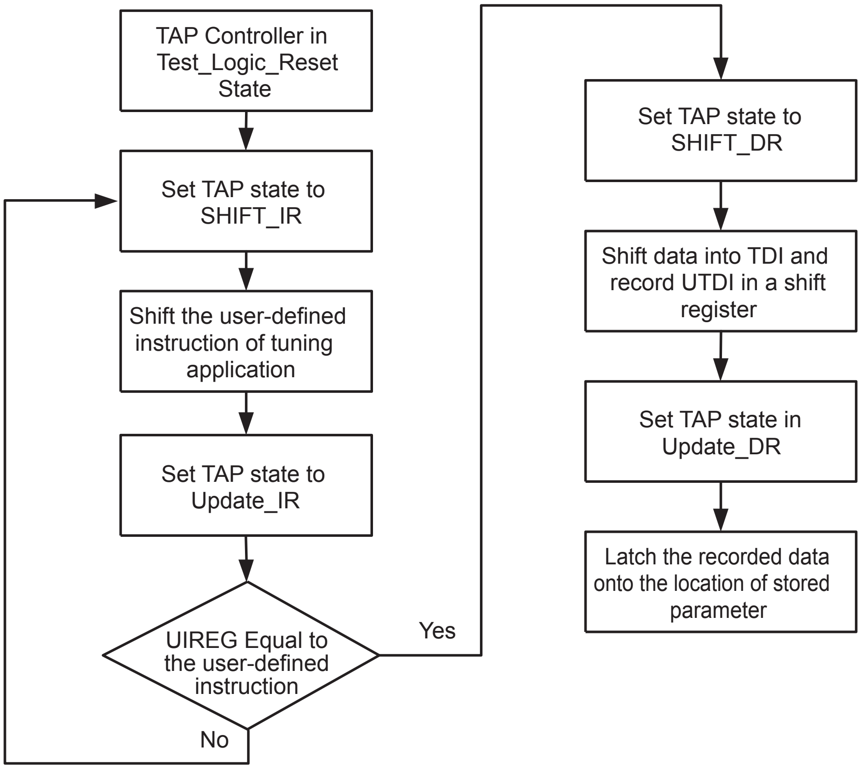

TAP Controller in Test_Logic_Reset State
Set TAP state to SHIFT_IR
Shift the user-defined instruction of tuning application
Set TAP state to Update_IR
UIREG Equal to the user-defined instruction
Yes
No
Set TAP state to SHIFT_DR
Shift data into TDI and record UTDI in a shift register
Set TAP state in Update_DR
Latch the recorded data onto the location of stored parameter

***Figure 17-5* • Flow Chart Example of Fine-Tuning an Application Using UJTAG**

Silicon Testing and Debugging

In many applications, the design needs to be tested, debugged, and verified on real silicon or in the final embedded application. To debug and test the functionality of designs, users may need to monitor some internal logic (or nets) during device operation. The approach of adding design test pins to monitor the critical internal signals has many disadvantages, such as limiting the number of user I/Os. Furthermore, adding external I/Os for test purposes may require additional or dedicated board area for testing and debugging.

The UJTAG tiles of low power flash devices offer a flexible and cost-effective solution for silicon test and debug applications. In this solution, the signals under test are shifted out to the TDO pin of the TAP Controller. The main advantage is that all the test signals are monitored from the TDO pin; no pins or additional board-level resources are required. Figure 17-6 illustrates this technique. Multiple test nets are brought into an internal MUX architecture. The selection of the MUX is done using the contents of the TAP Controller instruction register, where individual instructions (values from 16 to 127) correspond to different signals under test. The selected test signal can be synchronized with the rising or falling edge of TCK (optional) and sent out to UTDO to drive the TDO output of JTAG.

For flash devices, TDO (the output) is configured as low slew and the highest drive strength available in the technology and/or device. Here are some examples:

1. If the device is A3P1000 and VCCI is 3.3 V, TDO will be configured as LVTTL 3.3 V output, 24 mA, low slew.
2. If the device is AGLN020 and VCCI is 1.8 V, TDO will be configured as LVCMOS 1.8 V output, 4 mA, low slew.
3. If the device is AGLE300 and VCCI is 2.5 V, TDO will be configured as LVCMOS 2.5 V output, 24 mA, low slew.

The test and debug procedure is not limited to the example in Figure 17-5 on page 383. Users can customize the debug and test interface to make it appropriate for their applications. For example, multiple test signals can be registered and then sent out through UTDO, each at a different edge of TCK. In other words, *n* signals are sampled with an F_{TCK} / *n* sampling rate. The bandwidth of the information sent out to TDO is always proportional to the frequency of TCK.

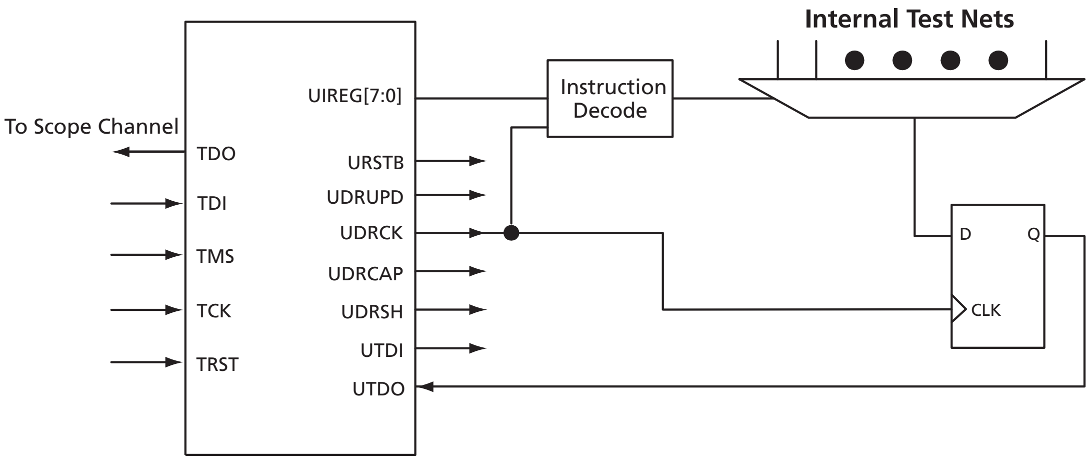


Figure 17-6 • **UJTAG Usage Example in Test and Debug Applications**

SRAM Initialization

Users can also initialize embedded SRAMs of the low power flash devices. The initialization of the embedded SRAM blocks of the design can be done using UJTAG tiles, where the initialization data is imported using the TAP Controller. Similar functionality is available in ProASIC$^{\underline{PLUS}}$ devices using JTAG. The guidelines for implementation and design examples are given in the *RAM Initialization and ROM Emulation in ProASIC$^{\underline{PLUS}}$ Devices* application note.

SRAMs are volatile by nature; data is lost in the absence of power. Therefore, the initialization process should be done at each power-up if necessary.

FlashROM Read-Back Using JTAG

The low power flash architecture contains a dedicated nonvolatile FlashROM block, which is formatted into eight 128-bit pages. For more information on FlashROM, refer to the "FlashROM in Microsemi's Low Power Flash Devices" section on page 119. The contents of FlashROM are available to the VersaTiles during normal operation through a read operation. As a result, the UJTAG macro can be used to provide the FlashROM contents to the JTAG port during normal operation. Figure 17-7 illustrates a simple block diagram of using UJTAG to read the contents of FlashROM during normal operation.

The FlashROM read address can be provided from outside the FPGA through the TDI input or can be generated internally using the core logic. In either case, data serialization logic is required (Figure 17-7) and should be designed using the VersaTile core logic. FlashROM contents are read asynchronously in parallel from the flash memory and shifted out in a synchronous serial format to TDO. Shifting the serial data out of the serialization block should be performed while the TAP is in UDRSH mode. The coordination between TCK and the data shift procedure can be done using the TAP state machine by monitoring UDRSH, UDRCAP, and UDRUPD.

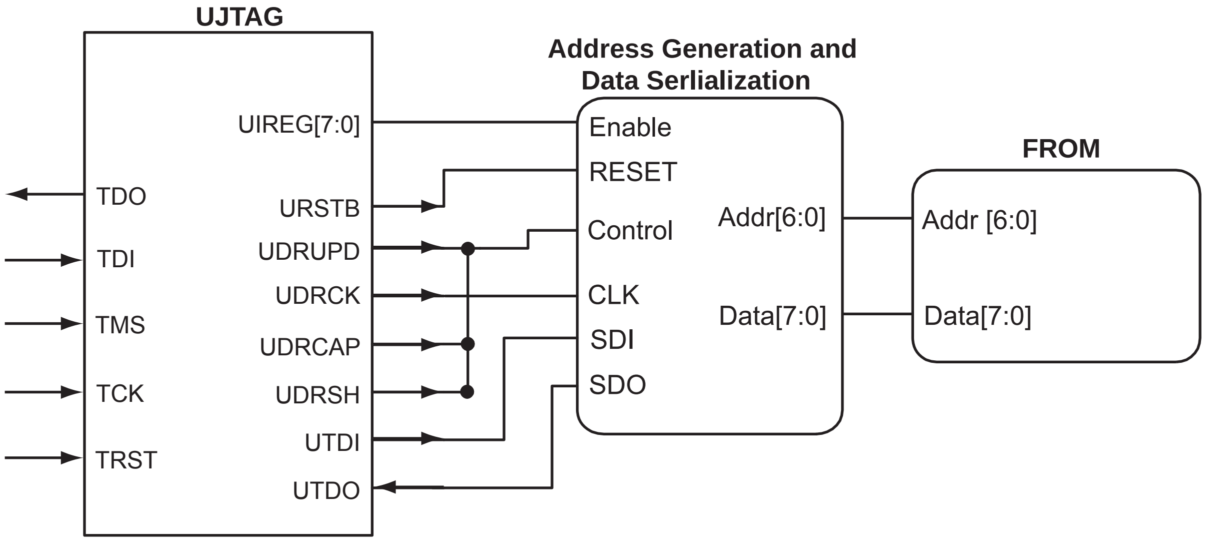


Figure 17-7 • **Block Diagram of Using UJTAG to Read FlashROM Contents**

Conclusion

Microsemi low power flash FPGAs offer many unique advantages, such as security, nonvolatility, reprogrammablity, and low power—all in a single chip. In addition, Fusion, IGLOO, and ProASIC3 devices provide access to the JTAG port from core VersaTiles while the device is in normal operating mode. A wide range of available user-defined JTAG opcodes allows users to implement various types of applications, exploiting this feature of these devices. The connection between the JTAG port and core tiles is implemented through an embedded and hardwired UJTAG tile. A UJTAG tile can be instantiated in designs using the UJTAG library cell. This document presents multiple examples of UJTAG applications, such as dynamic reconfiguration, silicon test and debug, fine-tuning of the design, and RAM initialization. Each of these applications offers many useful advantages.

Related Documents

Application Notes

RAM Initialization and ROM Emulation in ProASIC$^{\underline{PLUS}}$ Devices

http://www.microsemi.com/soc/documents/APA_RAM_Initd_AN.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
December 2011	Information on the drive strength and slew rate of TDO pins was added to the "Silicon Testing and Debugging" section (SAR 31749).	384
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 17-1 • Flash-Based FPGAs.	378
v1.3 (October 2008)	The "UJTAG Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	378
	The title of Table 17-3 • Configuration Bits of Fusion, IGLOO, and ProASIC3 CCC Blocks was revised to include Fusion.	382
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 17-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	378
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices.	N/A
	The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	378

18 – Power-Up/-Down Behavior of Low Power Flash Devices

Introduction

Microsemi's low power flash devices are flash-based FPGAs manufactured on a 0.13 µm process node. These devices offer a single-chip, reprogrammable solution and support Level 0 live at power-up (LAPU) due to their nonvolatile architecture.

Microsemi's low power flash FPGA families are optimized for logic area, I/O features, and performance. IGLOO® devices are optimized for power, making them the industry's lowest power programmable solution. IGLOO PLUS FPGAs offer enhanced I/O features beyond those of the IGLOO ultra-low power solution for I/O-intensive low power applications. IGLOO nano devices are the industry's lowest-power cost-effective solution. ProASIC3®L FPGAs balance low power with high performance. The ProASIC3 family is Microsemi's high-performance flash FPGA solution. ProASIC3 nano devices offer the lowest-cost solution with enhanced I/O capabilities.

Microsemi's low power flash devices exhibit very low transient current on each power supply during power-up. The peak value of the transient current depends on the device size, temperature, voltage levels, and power-up sequence.

The following devices can have inputs driven in while the device is not powered:

- IGLOO (AGL015 and AGL030)
- IGLOO nano (all devices)
- IGLOO PLUS (AGLP030, AGLP060, AGLP125)
- IGLOOe (AGLE600, AGLE3000)
- ProASIC3L (A3PE3000L)
- ProASIC3 (A3P015, A3P030)
- ProASIC3 nano (all devices)
- ProASIC3E (A3PE600, A3PE1500, A3PE3000)
- Military ProASIC3EL (A3PE600L, A3PE3000L, but not A3P1000)
- RT ProASIC3 (RT3PE600L, RT3PE3000L)

The driven I/Os do not pull up power planes, and the current draw is limited to very small leakage current, making them suitable for applications that require cold-sparing. These devices are hot-swappable, meaning they can be inserted in a live power system.[1]

[1] *For more details on the levels of hot-swap compatibility in Microsemi's low power flash devices, refer to the "Hot-Swap Support" section in the I/O Structures chapter of the FPGA fabric user's guide for the device you are using.*

Flash Devices Support Power-Up Behavior

The flash FPGAs listed in Table 18-1 support power-up behavior and the functions described in this document.

***Table 18-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications

*Note: *The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 18-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 18-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Power-Up/-Down Sequence and Transient Current

Microsemi's low power flash devices use the following main voltage pins during normal operation:[2]

- VCCPLX
- VJTAG
- VCC: Voltage supply to the FPGA core
  - VCC is 1.5 V ± 0.075 V for IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3 devices operating at 1.5 V.
  - VCC is 1.2 V ± 0.06 V for IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices operating at 1.2 V.
  - V5 devices will require a 1.5 V VCC supply, whereas V2 devices can utilize either a 1.2 V or 1.5 V VCC.
- VCCIBx: Supply voltage to the bank's I/O output buffers and I/O logic. Bx is the I/O bank number.
- VMVx: Quiet supply voltage to the input buffers of each I/O bank. x is the bank number. (Note: IGLOO nano, IGLOO PLUS, and ProASIC3 nano devices do not have VMVx supply pins.)

The I/O bank VMV pin must be tied to the VCCI pin within the same bank. Therefore, the supplies that need to be powered up/down during normal operation are VCC and VCCI. These power supplies can be powered up/down in any sequence during normal operation of IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, ProASIC3, and ProASIC3 nano FPGAs. During power-up, I/Os in each bank will remain tristated until the last supply (either VCCIBx or VCC) reaches its functional activation voltage. Similarly, during power-down, I/Os of each bank are tristated once the first supply reaches its brownout deactivation voltage.

Although Microsemi's low power flash devices have no power-up or power-down sequencing requirements, Microsemi identifies the following power conditions that will result in higher than normal transient current. Use this information to help maximize power savings:

Microsemi recommends tying VCCPLX to VCC and using proper filtering circuits to decouple VCC noise from the PLL.

a. If VCCPLX is powered up before VCC, a static current of up to 5 mA (typical) per PLL may be measured on VCCPLX.

   The current vanishes as soon as VCC reaches the VCCPLX voltage level.

   The same current is observed at power-down (VCC before VCCPLX).

b. If VCCPLX is powered up simultaneously or after VCC:

   i. Microsemi's low power flash devices exhibit very low transient current on VCC. For ProASIC3 devices, the maximum transient current on V_{CC} does not exceed the maximum standby current specified in the device datasheet.

The source of transient current, also known as inrush current, varies depending on the FPGA technology. Due to their volatile technology, the internal registers in SRAM FPGAs must be initialized before configuration can start. This initialization is the source of significant inrush current in SRAM FPGAs during power-up. Due to the nonvolatile nature of flash technology, low power flash devices do not require any initialization at power-up, and there is very little or no crossbar current through PMOS and NMOS devices. Therefore, the transient current at power-up is significantly less than for SRAM FPGAs. Figure 18-1 on page 390 illustrates the types of power consumption by SRAM FPGAs compared to Microsemi's antifuse and flash FPGAs.

[2] *For more information on Microsemi FPGA voltage supplies, refer to the appropriate datasheet located at http://www.microsemi.com/soc/techdocs/ds.*

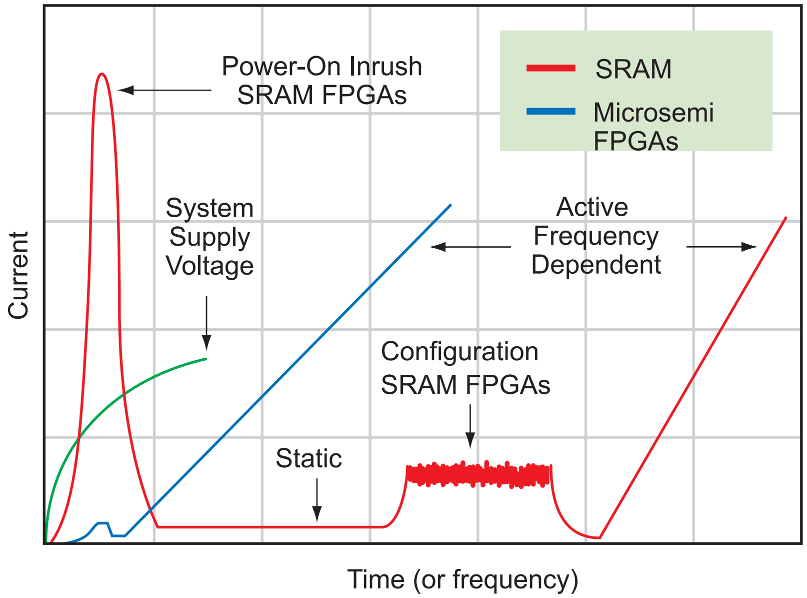


***Figure 18-1* • Types of Power Consumption in SRAM FPGAs and Microsemi Nonvolatile FPGAs**

Transient Current on VCC

The characterization of the transient current on VCC is performed on nearly all devices within the IGLOO, ProASIC3L, and ProASIC3 families. A sample size of five units is used from each device family member. All the device I/Os are internally pulled down while the transient current measurements are performed. For ProASIC3 devices, the measurements at typical conditions show that the maximum transient current on VCC, when the power supply is powered at ramp-rates ranging from 15 V/ms to 0.15 V/ms, does not exceed the maximum standby current specified in the device datasheets. Refer to the DC and Switching Characteristics chapters of the *ProASIC3 Flash Family FPGAS* datasheet and *ProASIC3E Flash Family FPGAs* datasheet for more information.

Similarly, IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices exhibit very low transient current on VCC. The transient current does not exceed the typical operating current of the device while in active mode. For example, the characterization of AGL600-FG256 V2 and V5 devices has shown that the transient current on VCC is typically in the range of 1–5 mA.

Transient Current on VCCI

The characterization of the transient current on VCCI is performed on devices within the IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3, ProASIC3 nano, and ProASIC3L groups of devices, similarly to VCC transient current measurements. For ProASIC3 devices, the measurements at typical conditions show that the maximum transient current on VCCI, when the power supply is powered at ramp-rates ranging from 33 V/ms to 0.33 V/ms, does not exceed the maximum standby current specified in the device datasheet. Refer to the DC and Switching Characteristics chapters of the *ProASIC3 Flash Family FPGAS* datasheet and *ProASIC3E Flash Family FPGAs* datasheet for more information.

Similarly, IGLOO, IGLOO PLUS, and ProASIC3L devices exhibit very low transient current on VCCI. The transient current does not exceed the typical operating current of the device while in active mode. For example, the characterization of AGL600-FG256 V2 and V5 devices has shown that the transient current on VCCI is typically in the range of 1–2 mA.

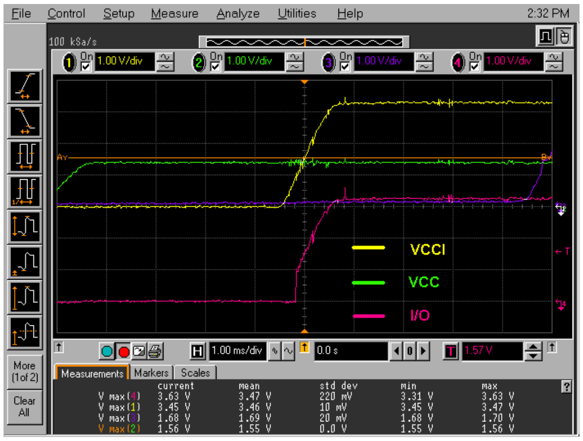
I/O Behavior at Power-Up/-Down

This section discusses the behavior of device I/Os, used and unused, during power-up/-down of V_{CC} and V_{CCI}. As mentioned earlier, VMVx and V_{CCI}Bx are tied together, and therefore, inputs and outputs are powered up/down at the same time.

I/O State during Power-Up/-Down

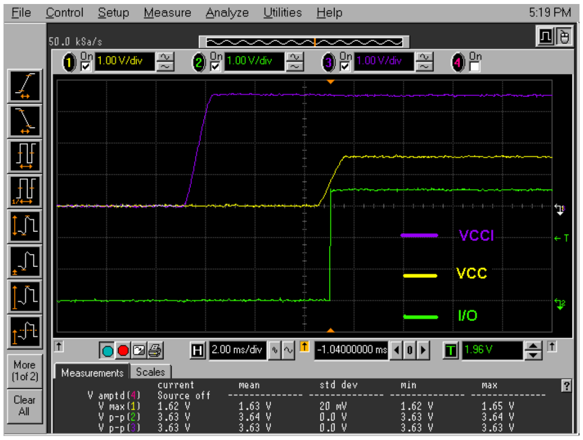
This section discusses the characteristics of I/O behavior during device power-up and power-down. Before the start of power-up, all I/Os are in tristate mode. The I/Os will remain tristated during power-up until the last voltage supply (VCC or VCCI) is powered to its functional level (power supply functional levels are discussed in the "Power-Up to Functional Time" section on page 392). After the last supply reaches the functional level, the outputs will exit the tristate mode and drive the logic at the input of the output buffer. Similarly, the input buffers will pass the external logic into the FPGA fabric once the last supply reaches the functional level. The behavior of user I/Os is independent of the VCC and VCCI sequence or the state of other voltage supplies of the FPGA (VPUMP and VJTAG). Figure 18-2 shows the output buffer driving HIGH and its behavior during power-up with 10 kΩ external pull-down. In Figure 18-2, VCC is powered first, and VCCI is powered 5 ms after VCC. Figure 18-3 on page 392 shows the state of the I/O when VCCI is powered about 5 ms before VCC. In the circuitry shown in Figure 18-3 on page 392, the output is externally pulled down.

During power-down, device I/Os become tristated once the first power supply (VCC or VCCI) drops below its brownout voltage level. The I/O behavior during power-down is also independent of voltage supply sequencing.



```
              current     mean      std dev     min       max
V max(4)      3.63 V      3.47 V    220 mV      3.31 V    3.63 V
V max(1)      3.45 V      3.46 V    10 mV       3.45 V    3.47 V
V max(3)      1.68 V      1.69 V    20 mV       1.68 V    1.70 V
V max(2)      1.56 V      1.55 V    0.0 V       1.55 V    1.56 V
```

***Figure 18-2* • I/O State when VCC Is Powered before VCCI**



```
              current        mean           std dev        min            max
V amptd(4)    Source off     -------------  -------------  -------------  -------------
V max(1)      1.62 V         1.63 V         20 mV          1.62 V         1.65 V
V p-p(2)      3.63 V         3.64 V         0.0 V          3.63 V         3.64 V
V p-p(3)      3.63 V         3.63 V         0.0 V          3.63 V         3.63 V
```

Figure 18-3 • **I/O State when VCCI Is Powered before VCC**

Power-Up to Functional Time

At power-up, device I/Os exit the tristate mode and become functional once the last voltage supply in the power-up sequence (VCCI or VCC) reaches its functional activation level. The power-up–to–functional time is the time it takes for the last supply to power up from zero to its functional level. Note that the functional level of the power supply during power-up may vary slightly within the specification at different ramp-rates. Refer to Table 18-2 for the functional level of the voltage supplies at power-up.

Typical I/O behavior during power-up–to–functional time is illustrated in Figure 18-2 on page 391 and Figure 18-3.

Table 18-2 **• Power-Up Functional Activation Levels for VCC and VCCI**

Device	VCC Functional Activation Level (V)	VCCI Functional Activation Level (V)
ProASIC3, ProASIC3 nano, IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices running at VCC = 1.5 V*	0.85 V ± 0.25 V	0.9 V ± 0.3 V
IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices running at VCC = 1.2 V*	0.85 V ± 0.2 V	0.9 V ± 0.15 V

*Note: *V5 devices will require a 1.5 V VCC supply, whereas V2 devices can utilize either a 1.2 V or 1.5 V VCC.*

Microsemi's low power flash devices meet Level 0 LAPU; that is, they can be functional prior to V_{CC} reaching the regulated voltage required. This important advantage distinguishes low power flash devices from their SRAM-based counterparts. SRAM-based FPGAs, due to their volatile technology, require hundreds of milliseconds after power-up to configure the design bitstream before they become functional. Refer to Figure 18-4 on page 393 and Figure 18-5 on page 394 for more information.

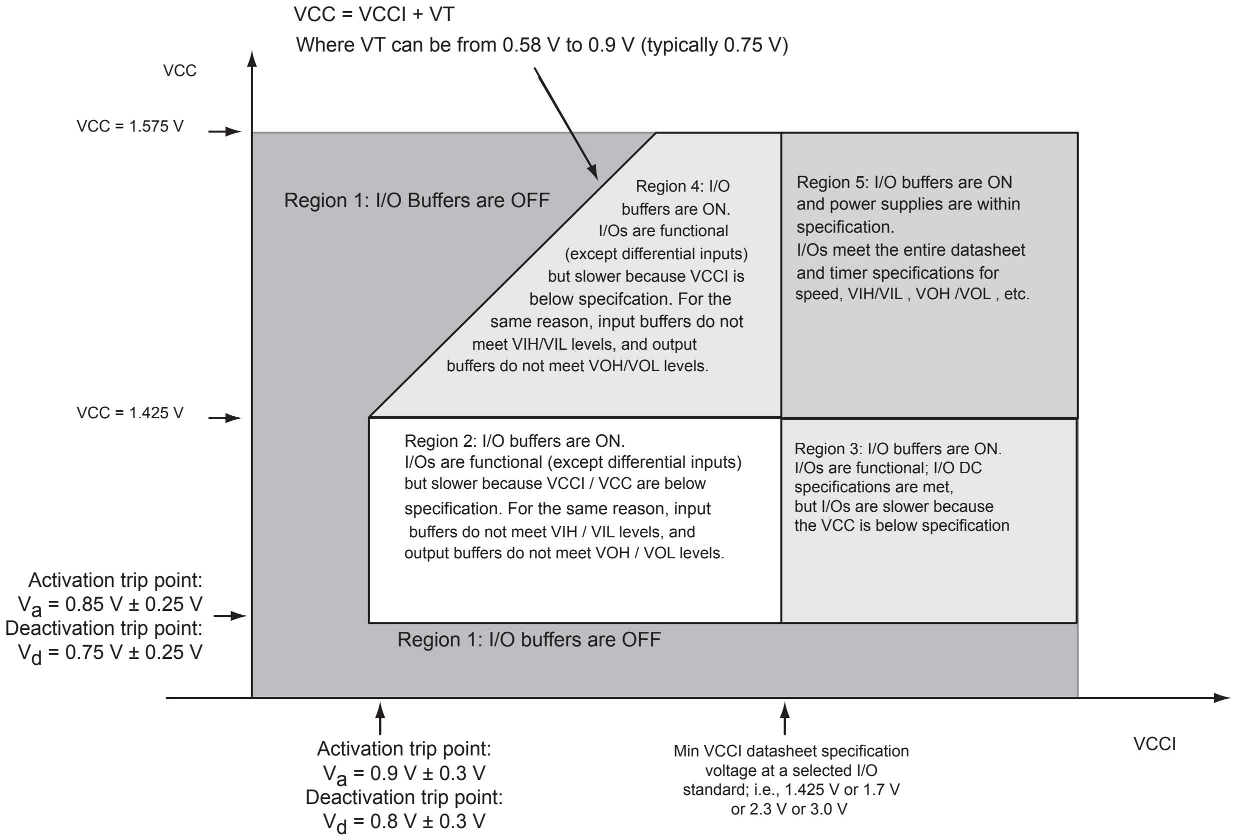

VCC = VCCI + VT
Where VT can be from 0.58 V to 0.9 V (typically 0.75 V)
VCC
VCC = 1.575 V
VCC = 1.425 V
Activation trip point:
V_a = 0.85 V ± 0.25 V
Deactivation trip point:
V_d = 0.75 V ± 0.25 V
Region 1: I/O Buffers are OFF
Region 4: I/O buffers are ON. I/Os are functional (except differential inputs) but slower because VCCI is below specifcation. For the same reason, input buffers do not meet VIH/VIL levels, and output buffers do not meet VOH/VOL levels.
Region 5: I/O buffers are ON and power supplies are within specification. I/Os meet the entire datasheet and timer specifications for speed, VIH/VIL , VOH /VOL , etc.
Region 2: I/O buffers are ON. I/Os are functional (except differential inputs) but slower because VCCI / VCC are below specification. For the same reason, input buffers do not meet VIH / VIL levels, and output buffers do not meet VOH / VOL levels.
Region 3: I/O buffers are ON. I/Os are functional; I/O DC specifications are met, but I/Os are slower because the VCC is below specification
Region 1: I/O buffers are OFF
Activation trip point:
V_a = 0.9 V ± 0.3 V
Deactivation trip point:
V_d = 0.8 V ± 0.3 V
Min VCCI datasheet specification voltage at a selected I/O standard; i.e., 1.425 V or 1.7 V or 2.3 V or 3.0 V
VCCI

***Figure 18-4* • I/O State as a Function of VCCI and VCC Voltage Levels for IGLOO V5, IGLOO nano V5, IGLOO PLUS V5, ProASIC3L, and ProASIC3 Devices Running at VCC = 1.5 V ± 0.075 V**

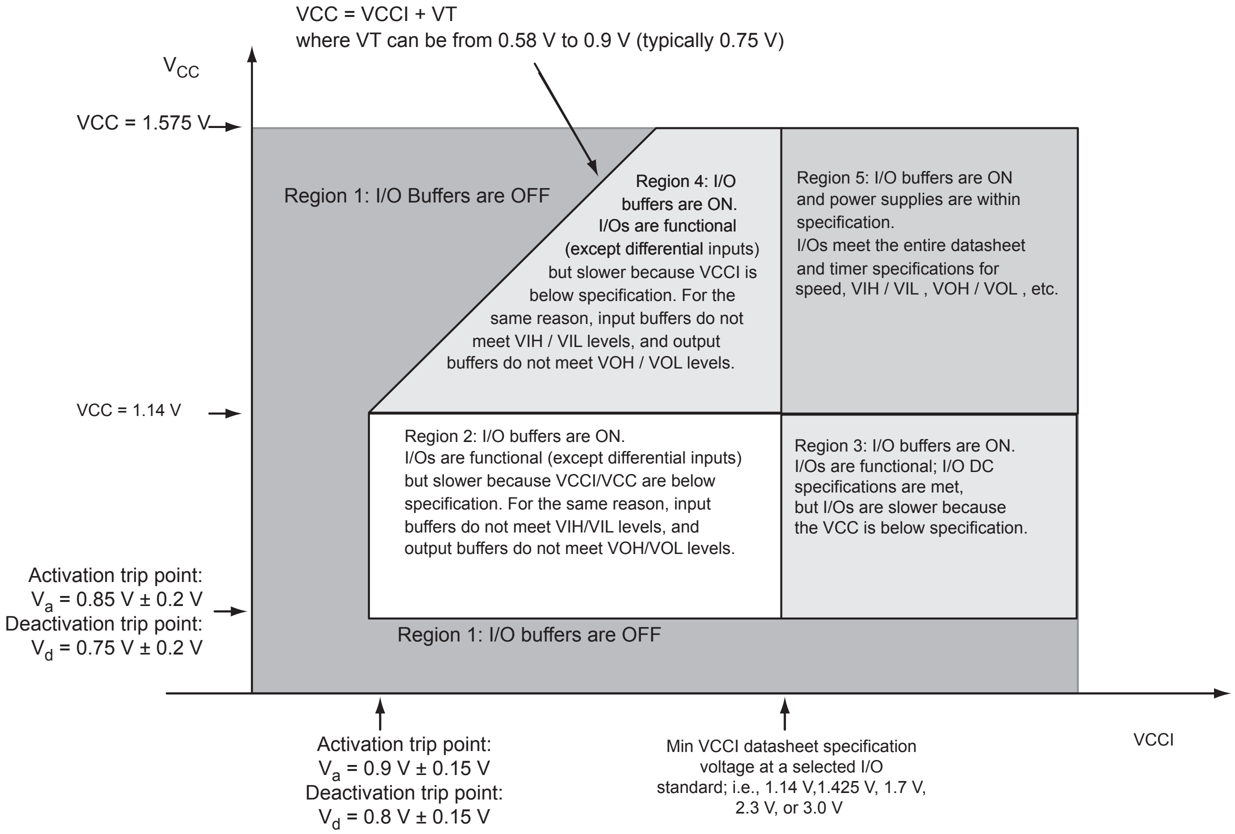

VCC = VCCI + VT
where VT can be from 0.58 V to 0.9 V (typically 0.75 V)
V_{CC}
VCC = 1.575 V
VCC = 1.14 V
Activation trip point:
V_a = 0.85 V ± 0.2 V
Deactivation trip point:
V_d = 0.75 V ± 0.2 V
Region 1: I/O Buffers are OFF
Region 4: I/O buffers are ON. I/Os are functional (except differential inputs) but slower because VCCI is below specification. For the same reason, input buffers do not meet VIH / VIL levels, and output buffers do not meet VOH / VOL levels.
Region 5: I/O buffers are ON and power supplies are within specification. I/Os meet the entire datasheet and timer specifications for speed, VIH / VIL , VOH / VOL , etc.
Region 2: I/O buffers are ON. I/Os are functional (except differential inputs) but slower because VCCI/VCC are below specification. For the same reason, input buffers do not meet VIH/VIL levels, and output buffers do not meet VOH/VOL levels.
Region 3: I/O buffers are ON. I/Os are functional; I/O DC specifications are met, but I/Os are slower because the VCC is below specification.
Region 1: I/O buffers are OFF
Activation trip point:
V_a = 0.9 V ± 0.15 V
Deactivation trip point:
V_d = 0.8 V ± 0.15 V
Min VCCI datasheet specification voltage at a selected I/O standard; i.e., 1.14 V,1.425 V, 1.7 V, 2.3 V, or 3.0 V
VCCI

***Figure 18-5* • I/O State as a Function of VCCI and VCC Voltage Levels for IGLOO V2, IGLOO nano V2, IGLOO PLUS V2, and ProASIC3L Devices Running at VCC = 1.2 V ± 0.06 V**

Brownout Voltage

Brownout is a condition in which the voltage supplies are lower than normal, causing the device to malfunction as a result of insufficient power. In general, Microsemi does not guarantee the functionality of the design inside the flash FPGA if voltage supplies are below their minimum recommended operating condition. Microsemi has performed measurements to characterize the brownout levels of FPGA power supplies. Refer to Table 18-3 for device-specific brownout deactivation levels. For the purpose of characterization, a direct path from the device input to output is monitored while voltage supplies are lowered gradually. The brownout point is defined as the voltage level at which the output stops following the input. Characterization tests performed on several IGLOO, ProASIC3L, and ProASIC3 devices in typical operating conditions showed the brownout voltage levels to be within the specification.

During device power-down, the device I/Os become tristated once the first supply in the power-down sequence drops below its brownout deactivation voltage.

***Table 18-3* • Brownout Deactivation Levels for VCC and VCCI**

Devices	VCC Brownout Deactivation Level (V)	VCCI Brownout Deactivation Level (V)
ProASIC3, ProASIC3 nano, IGLOO, IGLOO nano, IGLOO PLUS and ProASIC3L devices running at VCC = 1.5 V	0.75 V ± 0.25 V	0.8 V ± 0.3 V
IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices running at VCC = 1.2 V	0.75 V ± 0.2 V	0.8 V ± 0.15 V

PLL Behavior at Brownout Condition

When PLL power supply voltage and/or V_{CC} levels drop below the V_{CC} brownout levels mentioned above for 1.5 V and 1.2 V devices, the PLL output lock signal goes LOW and/or the output clock is lost. The following sections explain PLL behavior during and after the brownout condition.

VCCPLL and VCC **Tied Together**

In this condition, both VCC and VCCPLL drop below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level. During the brownout recovery, once VCCPLL and VCC reach the activation point (0.85 ± 0.25 V or ± 0.2 V) again, the PLL output lock signal may still remain LOW with the PLL output clock signal toggling. If this condition occurs, there are two ways to recover the PLL output lock signal:

1. Cycle the power supplies of the PLL (power off and on) by using the PLL POWERDOWN signal.
2. Turn off the input reference clock to the PLL and then turn it back on.

Only VCCPLL **Is at Brownout**

In this case, only VCCPLL drops below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level and the VCC supply remains at nominal recommended operating voltage (1.5 V ± 0.075 V for 1.5 V devices and 1.2 V ± 0.06 V for 1.2 V devices). In this condition, the PLL behavior after brownout recovery is similar to initial power-up condition, and the PLL will regain lock automatically after VCCPLL is ramped up above the activation level (0.85 ± 0.25 V or ± 0.2 V). No intervention is necessary in this case.

Only VCC **Is at Brownout**

In this condition, VCC drops below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level and VCCPLL remains at nominal recommended operating voltage (1.5 V ± 0.075 V for 1.5 V devices and 1.2 V ± 0.06 V for 1.2 V devices). During the brownout recovery, once VCC reaches the activation point again (0.85 ± 0.25 V or ± 0.2 V), the PLL output lock signal may still remain LOW with the PLL output clock signal toggling. If this condition occurs, there are two ways to recover the PLL output lock signal:

1. Cycle the power supplies of the PLL (power off and on) by using the PLL POWERDOWN signal.
2. Turn off the input reference clock to the PLL and then turn it back on.

It is important to note that Microsemi recommends using a monotonic power supply or voltage regulator to ensure proper power-up behavior.

Internal Pull-Up and Pull-Down

Low power flash device I/Os are equipped with internal weak pull-up/-down resistors that can be used by designers. If used, these internal pull-up/-down resistors will be activated during power-up, once both VCC and VCCI are above their functional activation level. Similarly, during power-down, these internal pull-up/-down resistors will turn off once the first supply voltage falls below its brownout deactivation level.

Cold-Sparing

In cold-sparing applications, voltage can be applied to device I/Os before and during power-up. Cold-sparing applications rely on three important characteristics of the device:

1. I/Os must be tristated before and during power-up.
2. Voltage applied to the I/Os must not power up any part of the device.
3. VCCI should not exceed 3.6 V, per datasheet specifications.

As described in the "Power-Up to Functional Time" section on page 392, Microsemi's low power flash I/Os are tristated before and during power-up until the last voltage supply (VCC or VCCI) is powered up past its functional level. Furthermore, applying voltage to the FPGA I/Os does not pull up VCC or VCCI and, therefore, does not partially power up the device. Table 18-4 includes the cold-sparing test results on A3PE600-PQ208 devices. In this test, leakage current on the device I/O and residual voltage on the power supply rails were measured while voltage was applied to the I/O before power-up.

***Table 18-4* • Cold-Sparing Test Results for A3PE600 Devices**

Device I/O	Residual Voltage (V)		Leakage Current
	VCC	VCCI	
Input	0	0.003	<1 µA
Output	0	0.003	<1 µA

VCCI must not exceed 3.6 V, as stated in the datasheet specification. Therefore, ProASIC3E devices meet all three requirements stated earlier in this section and are suitable for cold-sparing applications.

The following devices and families support cold-sparing:

- IGLOO: AGL015 and AGL030
- All IGLOO nano
- All IGLOO PLUS
- All IGLOOe
- ProASIC3L: A3PE3000L
- ProASIC3: A3P015 and A3P030
- All ProASIC3 nano
- All ProASIC3E
- Military ProASIC3EL: A3PE600L and A3PE3000L
- RT ProASIC3: RT3PE600L and RT3PE3000L

The following devices and families do not support cold-sparing:

- IGLOO: AGL060, AGL125, AGL250, AGL600, AGL1000
- ProASIC3: A3P060, A3P125, A3P250, A3P400, A3P600, A3P1000
- ProASIC3L: A3P250L, A3P600L, A3P1000L
- Military ProASIC3: A3P1000

Hot-Swapping

Hot-swapping is the operation of hot insertion or hot removal of a card in a powered-up system. The I/Os need to be configured in hot-insertion mode if hot-swapping compliance is required. For more details on the levels of hot-swap compatibility in low power flash devices, refer to the "Hot-Swap Support" section in the I/O Structures chapter of the user's guide for the device you are using.

The following devices and families support hot-swapping:

- IGLOO: AGL015 and AGL030
- All IGLOO nano
- All IGLOO PLUS
- All IGLOOe
- ProASIC3L: A3PE3000L
- ProASIC3: A3P015 and A3P030
- All ProASIC3 nano
- All ProASIC3E
- Military ProASIC3EL: A3PE600L and A3PE3000L
- RT ProASIC3: RT3PE600L and RT3PE3000L

The following devices and families do not support hot-swapping:

- IGLOO: AGL060, AGL125, AGL250, AGL400, AGL600, AGL1000
- ProASIC3: A3P060, A3P125, A3P250, A3P400, A3P600, A3P1000
- ProASIC3L: A3P250L, A3P600L, A3P1000L
- Military ProASIC3: A3P1000

Conclusion

Microsemi's low power flash FPGAs provide an excellent programmable logic solution for a broad range of applications. In addition to high performance, low cost, security, nonvolatility, and single chip, they are live at power-up (meet Level 0 of the LAPU classification) and offer clear and easy-to-use power-up/-down characteristics. Unlike SRAM FPGAs, low power flash devices do not require any specific power-up/-down sequencing and have extremely low power-up inrush current in any power-up sequence. Microsemi low power flash FPGAs also support both cold-sparing and hot-swapping for applications requiring these capabilities.

Related Documents

Datasheets

ProASIC3 Flash Family FPGAs

http://www.microsemi.com/soc/documents/PA3_DS.pdf

ProASIC3E Flash Family FPGAs

http://www.microsemi.com/soc/documents/PA3E_DS.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
v1.2 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to the document as supported device types.	
v1.1 (October 2008)	The "Introduction" section was updated to add Military ProASIC3EL and RT ProASIC3 devices to the list of devices that can have inputs driven in while the device is not powered.	387
	The "Flash Devices Support Power-Up Behavior" section was revised to include new families and make the information more concise.	388
	The "Cold-Sparing" section was revised to add Military ProASIC3/EL and RT ProASIC3 devices to the lists of devices with and without cold-sparing support.	396
	The "Hot-Swapping" section was revised to add Military ProASIC3/EL and RT ProASIC3 devices to the lists of devices with and without hot-swap support. AGL400 was added to the list of devices that do not support hot-swapping.	397
v1.0 (August 2008)	This document was revised, renamed, and assigned a new part number. It now includes data for the IGLOO and ProASIC3L families.	N/A
v1.3 (March 2008)	The "List of Changes" section was updated to include the three different I/O Structure handbook chapters.	398
v1.2 (February 2008)	The first sentence of the "PLL Behavior at Brownout Condition" section was updated to read, "When PLL power supply voltage and/or V_{CC} levels drop below the VCC brownout levels (0.75 V ± 0.25 V), the PLL output lock signal goes low and/or the output clock is lost."	395
v1.1 (January 2008)	The "PLL Behavior at Brownout Condition" section was added.	395

19 – ProASIC3/E SSO and Pin Placement Guidelines

Introduction

Ground bounce and VCC bounce have always been present in digital integrated circuits (ICs). With the advance of technology and shrinking CMOS features, the speed of designs, I/O slew rates, and the size of I/O busses have increased significantly in the past few years. As a result, simultaneously switching outputs (SSOs) and their effects on signal integrity have become an important factor in any digital IC design. When SSOs are not properly designed into a board layout or digital IC, data corruption and system failure may result.

To prevent SSO-induced issues in modern digital systems, designers must compromise an elegant board layout for reliability. An elegant board layout may include practices such as placing all inputs on one side of a chip, outputs on the opposite side, and all bus pins next to each other to make board layout simple. In today's digital systems, utilizing modern FPGAs such as Microsemi ProASIC®3/E may result in data corruption due to ground bounce, VCC bounce, or crosstalk. To design a reliable system for ProASIC3/E FPGAs, follow three simple rules:

1. Identify the SSOs in a design as early in the design cycle as possible, and spread them out across the entire die periphery. Avoid clusters of more than four adjacent SSO pins.
2. Identify sensitive (and usually asynchronous) system signals, and shield them from the effects of SSO (specific shielding techniques are discussed later in this document).
3. Use the lowest possible I/O slew rate and drive strength the design timing will support.

Furthermore, relatively large lead inductance in PQ, TQ, and VQ packages makes these packages more vulnerable to SSOs and hence undesirable for high-speed designs or designs with a considerable number of SSOs. FG or BG packages are preferred in such designs because they show much better SSO performance. By following the above three rules, you will create reliable systems free from the effects of SSOs. The following sections cover specific SSO recommendations and mitigation techniques for designs that do not comply with these recommendations.

SSO Effects

The total number of SSOs for each bus is determined by identifying the outputs that are synchronous to a single clock domain, have their clock-to-out times within ±200 ps of each other, and are placed next to each other on die pads that are on both sides of a sensitive I/O, as shown in Figure 19-1 on page 400. The sensitive I/O affected by SSO is sometimes referred to as the victim I/O or quiet I/O. SSOs may affect the victim I/O if the total number of SSOs on both sides of the victim I/O exceeds the ProASIC3/E SSO recommendation. It is important to note that the SSOs should be referenced to the die pads and not package pins, since neighboring package pins are not necessarily next to each other on the die (e.g., for BG and FG packages). This can be determined by using MultiView Navigator (MVN) in the Designer software, or die/package bonding diagrams provided by Microsemi. However, when routing traces on the board, it is important to note that SSOs on neighboring traces on the board may affect the quiet I/O surrounded by the SSO traces due to crosstalk or coupling.

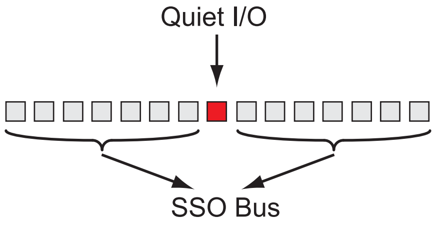


Figure 19-1 • **Basic Block Diagram of Quiet I/O Surrounded by SSO Bus**

SSO Effect on Power and Ground for Quiet Outputs

If SSOs toggle in one direction (either HIGH to LOW or LOW to HIGH), a significant amount of current quickly begins to flow to the ground or V_{CCI} pins. This current is the sum of the simultaneous sink or source currents of the CMOS output buffers. The quick jump in current causes a voltage drop on the parasitic inductance between the board and die VCCI and ground (V = L × di/dt). For more information about the ground and VCCI bounce phenomenon, refer to the *Simultaneous Switching Noise* application note. The local fluctuations of the VCCI and ground levels may cause the signals on quiet outputs (measured with respect to the fluctuating VCCI and ground) to be misinterpreted as unwanted logic glitches.

SSO Effect on Inputs

SSOs may also affect quiet inputs due to the mutual inductance and capacitance on the package in addition to possible crosstalk of signal traces on the board. SSOs can cause logic glitches on any quiet inputs they surround. The unwanted glitches may cause functional failures if they are propagated through the input buffer. In SSO characterization of ProASIC3/E devices, glitches are considered errors if they cause internal latches in the design to trigger.

SSO Effect on Output Delay (push-out)

As the speed and I/O slew rate of digital ICs increase, effects of ground and V_{CCI} bounce start to surface in digital system designs. One of these effects is output delay or push-out. The ground bounce and V_{CCI} dip induced by SSO transitions creates a temporary collapse of internal VCCI and/or GND supply levels in the output buffers. This change in supply level increases the output buffer propagation delay time. It is important to note that push-out occurs on the SSO bus itself as well as on the victim outputs. Multiple factors, such as SSO bus frequency, drive strength, and slew rate, contribute to push-out. These factors can be adjusted to mitigate the push-out phenomenon. If the clock-to-out time of the victim output is important, the push-out delay should be considered in the timing budget of the design.

SSO Effect on Minimum Input Slew Rate (input maximum rise/fall time)

If the SSOs surrounding an input exceed the Microsemi recommendation, the minimum slew rate requirement for that input may be affected. The minimum input slew rate is the slowest signal transition time (from 0 to 1 or vice versa) at the input that does not cause unwanted logic glitches during signal transition. Figure 19-2 on page 401 illustrates the unwanted logic glitches with slow transition times. As shown, the logic glitch due to the slow input transition time may cause logic malfunction at edge-sensitive inputs (i.e., clock signals). If the sensitive inputs are affected by the SSO bus, the input minimum slew rate (maximum rise and fall time) should be reduced from what is listed in the device datasheet. Usually, synchronous, level-sensitive inputs are not prone to malfunction due to this phenomenon because their logic value is important only when sampled by a clock.

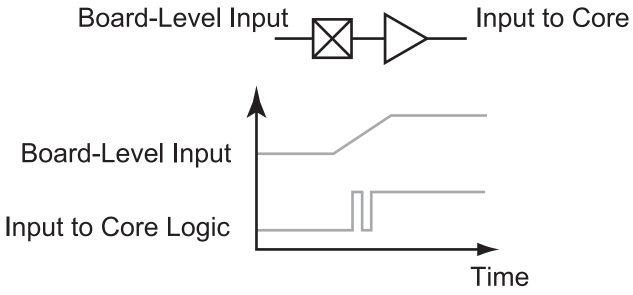


Figure 19-2 • **Slow Rise/Fall Time Causing Glitches at the Output of an Input Buffer**

Shielding from SSOs

When exposure of sensitive signals (e.g., asynchronous reset) to SSOs is inevitable, these signals need to be shielded from the SSOs to mitigate the unwanted effects. Shielding is basically separating the sensitive signals from SSOs using neighboring pins. Figure 19-3 shows a basic block diagram depicting a victim output in the presence of an SSO bus.

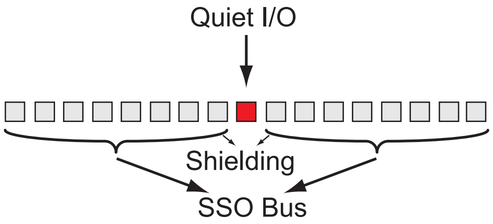


Figure 19-3 • **Shielding Scheme**

There are different shielding techniques that can be used to protect the victim I/O from the SSO bus. Before describing these techniques, the concept of *virtual ground* and *virtual* V_{CCI} should be understood.

Virtual Ground

Virtual ground, also known as soft ground, is used to improve noise performance. As opposed to a real ground, which is connected to planes within the package, a virtual ground is connected to the planes through the impedance of an I/O buffer. A virtual ground is a ground pin implemented using regular I/O ports. To implement a virtual ground, instantiate an output buffer (with highest drive strength and slew rate) in the design. Tie the input of this output buffer to zero within the design so the output buffer is constantly driving to the ground level.

Virtual VCCI

Virtual VCCI is similar to virtual ground. The only difference is that in the case of virtual VCCI, the output buffer is permanently driving to logic HIGH.

In general, there are two shielding methods recommended by Microsemi: a) using GND pins or virtual grounds and b) using any VCCII, GND, VCCI, unused I/O, used (but not sensitive) I/O, or any combination of these pins.

Shielding Using GND or Virtual Ground Pins

When shielding sensitive I/Os from the SSO bus, GND or virtual ground pins can be used if required. In this case, two or three GND or virtual ground pins should be placed on each side of the quiet I/O. The shielding pins should be connected externally to the board-level ground. To prevent any board-level coupling or crosstalk noise on the sensitive I/Os, the shielding pins should be routed on the board alongside the SSO bus for the whole length of the SSO traces and on the same board layer. These shielding traces should be connected to board ground at both ends of their length.

Shielding Using Other Pins

The type of shielding pins is not restricted to GND or virtual grounds. The shielding pins can also be VCCI, VCCII, virtual VCCI, unused I/Os, or used I/Os that are not sensitive to SSO effects (e.g., outputs driving LEDs).

How to Use This Document

The rest of this document is divided based on three different SSO effects: on outputs, on inputs, and on Clock Conditioning Circuits (CCCs). Each section includes tables that identify the recommended maximum number of SSOs and the required shielding if the number of SSOs exceeds the recommendation. The tables are categorized by device/package type (e.g., A3P600-FG484) and I/O configuration of the SSO bus (i.e., drive strength and slew rate). If the desired device/package combination cannot be found in the tables, choose the SSO recommendation for the closest package type and the next smaller die size. The following example describes two scenarios in which the SSO recommendation for another device/package can be used for a member of the ProASIC3/E family:

1. SSO guidelines for A3P250-PQ208 can be used when designing for A3P400-PQ208.
2. SSO guidelines for A3PE600-FG484 can be used when designing for A3PE1500-FG676.

You should study this entire document, consider the desired device/package combination, define the worst-case SSO scenario, and use the SSO guidelines or shielding recommendations described in the tables. At the end of each section, guidelines are given on how to mitigate the effects of SSOs. Note that the data presented in this document is collected at nominal operating conditions (1.5 V core voltage and room temperature). CMOS transistors switch faster when cold, and therefore the edge rates become faster, so SSO effects are usually worse at lower temperatures.

At the end of this document, some general board-level design guidelines are included. Microsemi recommends that you follow these guidelines when designing boards.

SSO Effects on Outputs

This section describes the SSO effects on other outputs. As stated in the "Shielding from SSOs" section on page 401, in ProASIC3 and ProASIC3E devices, the effects of SSOs on quiet outputs are categorized by ground bounce, VCCI bounce, and push-out. The following sections give the characteristics of SSO effects on outputs and provide guidelines on how to mitigate these effects.

Ground and VCCI Bounce

The most widely known effects of SSOs are ground and VCCI bounce. This section characterizes ProASIC3/E ground and VCCI bounce in the presence of SSOs. Since outputs with higher drive strength or faster slew rate source/sink higher current at the time of switching, SSOs are more disruptive when they are configured at higher drive strength and high slew rate. Table 19-1 on page 403 lists the number of SSOs causing specified levels of ground and VCCI bounce for various device, package, and SSO bus configurations. A disruptive ground bounce is one with a 1.25 V peak and 1 ns width—enough to trigger a high-speed input to change its value from zero to one. Similarly, a disruptive VCCI bounce causes oscillations on the quiet output (driving HIGH) with a magnitude of 2 V and width of 1 ns. These values are chosen based on Microsemi bench experiments using typical CMOS input sensitivity.

***Table 19-1* • Number of SSOs Causing Specified Ground and VCCI Bounce**

Device	Drive Strength (mA)	Slew Rate	SSOs Causing GND Bounce	SSOs Causing VCCI Bounce
A3P250-PQ208	24	High	4	2
		Low	4	6
	12	High	8	12
		Low	16	16
A3PE600-PQ208	24	High	6	4
		Low	8	8
	12	High	10	12
		Low	14	16
A3PE600-FG484	24	High	24	10
		Low	56	16
	12	High	>64	32
		Low	>64	>64
A3P1000-FG484	24	High	36	12
		Low	>64	26
	16	High	>64	24
		Low	>64	40
	12	High	>64	36
		Low	>64	>64
	8/6/4/2	High	>64	>64
		Low	>64	>64
A3PE1500-FG484	24	High	28	18
		Low	46	28
	16	High	46	26
		Low	>64	>64
	12	High	>64	62
		Low	>64	>64
	8/6/4/2	High	>64	>64
		Low	>64	>64
A3PE/AGLE3000-FG484	24	High	54	24
		Low	>64	42
	16	High	>64	44
		Low	>64	>64
	12	High	>64	>64
		Low	>64	>64
	8/6/4/2	High	>64	>64
		Low	>64	>64

Table 19-2 **• Ground Bounce and Shielding for A3PE3000-FG896**

Drive Strength (mA)	Slew Rate	Ground Bounce 5 < SSOs < 10	Ground Bounce SSOs >10	Shielding for 4 < SSOs < 8	Shielding for SSOs > 8
24	High	Negligible	1 ns	0	0
	Low	Negligible	Negligible	0	0
12	High	Negligible	800 ps	0	0
	Low	Negligible	Negligible	0	0
< 8	Any	Negligible	Negligible	0	0

Output Push-Out

As described in the "SSO Effect on Output Delay (push-out)" section on page 400, if an output is surrounded by SSOs, the propagation delay of that output may be increased due to the noise on ground or VCCI. Figure 19-4 shows a simple diagram of the push-out effect. As shown, the push-out effect occurs only if the affected output toggles at the same time as the SSO bus. If the outputs surrounded by the SSO bus are not switching simultaneously (within 200 ps of each other) with the SSOs, the outputs are not affected by the push-out phenomenon.

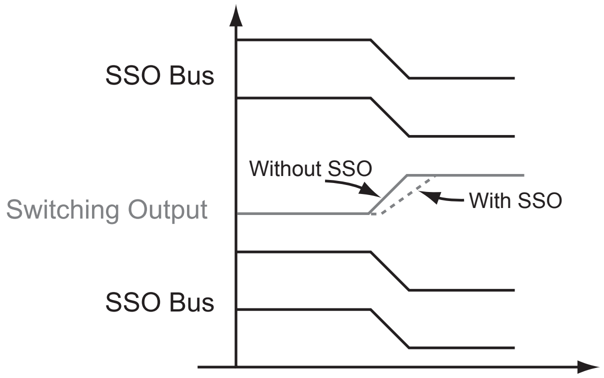


Figure 19-4 **• SSO Push-Out Effect**

Table 19-3 lists the increase in output delay for various SSO widths and configurations.

Table 19-3 **• SSO Push-Out Effect on an Output Surrounded by SSOs**

Package	Drive Strength (mA)	Slew Rate	5 < SSOs < 10	SSOs ≥ 10
PQ208	24	High	<1.1 ns	<1.8 ns
		Low	<600 ps	<1.2 ns
	12	High	<900 ps	<1.5 ns
		Low	Negligible	Negligible
	≤8	Any	Negligible	Negligible
FG484	Any	Any	Negligible	Negligible

Notes:

1. *Table data obtained when output load is 30 pF.*
2. *Larger output load increases the push-out effect. As an example, increasing the output load from 30 pF to 50 pF increases the push-out effect by 40%.*

Mitigating SSO Effects on Outputs

Any effort to mitigate the SSO effect starts with eliminating the SSOs themselves. As described in "Introduction" on page 399, the SSOs should be spread across the die pads to avoid a large SSO bus concentrated in one area of the die. If possible, the clock-to-out timing of output busses should be staggered to reduce the number of SSOs in vicinity of sensitive outputs. If placement of sensitive outputs close to an SSO bus is inevitable, such outputs should be shielded from the bus. The shielding scheme to protect delay-sensitive outputs is similar to the guidelines presented in Table 19-4 on page 406. Whenever shielding is required, it is recommended to use GND or virtual ground pins as shielding. However, it is acceptable to use other shielding pins to protect sensitive outputs from SSOs.

Segmenting SSO busses into smaller sections helps mitigate the SSO effect. The SSO bus can be segmented by inserting spacers among the SSO bus pins when placed on the die pads, as shown in Figure 19-5. The spacers can be GND or virtual ground, V_{CCI} or virtual V_{CCI}, unused I/O, or used I/Os that are not assigned to sensitive signals and do not toggle frequently or synchronously with the SSOs (e.g., signals driving LEDs).

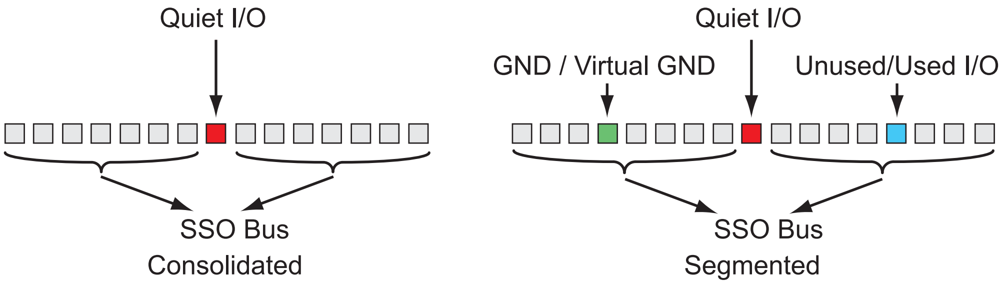


***Figure 19-5* • Example of Consolidated and Segmented SSO Bus**

Also, as described in Table 19-1 on page 403 and Table 19-3, FG and BG packages show much better characteristics with respect to SSO effects than PQ, TQ, or VQ packages. Therefore, for relatively high-speed designs or designs that have a significant number of wide output busses, FG or BG packages are strongly recommended.

In addition to the logic design and device package type, board-level design is a key parameter in mitigating SSO effects. A well-designed PCB, capable of providing clean voltage supplies to the FPGA, is less susceptible to noise and therefore performs better.

Board-Level Timing Analysis with Push-Out

Since the push-out effect changes the clock-to-out timing of the signal surrounded by SSOs, designers should take care when performing board-level timing analysis for such outputs. The following are the Microsemi recommendations for calculating the clock-to-out timing of signals affected by push-out phenomena:

- For board-level setup time calculations:

  Clock-to-out = worst-case clock-to-out reported by SmartTime + push-out delay
- For board-level hold time calculations:

  Clock-to-out = best-case clock-to-out reported by SmartTime

SSO Effects on Inputs

As described in the "Virtual VCCI" section on page 401, if a quiet input is surrounded by SSOs, the logic driven by that input may experience a glitch when the SSO bus is switching. In ProASIC3/E devices in FG or BG packages, the inputs are not affected by an SSO bus. However, in PQ, TQ, and VQ packages, due to larger lead inductance, the SSOs may affect the inputs as described in the "SSO Effect on Inputs" section on page 400 and the "SSO Effect on Minimum Input Slew Rate (input maximum rise/fall time)" section on page 400. Table 19-4 describes the input shielding required for various SSO sizes with different I/O configurations. For example, in a PQ208 package, if a sensitive input (e.g., asynchronous reset) is surrounded by an SSO bus configured with 16 mA drive strength and low slew rate, two shielding pins are required on each side of the sensitive input to prevent any logic glitch on the reset line during transition of the SSO bus.

***Table 19-4* • Shielding Requirement Protecting Inputs from SSO** [1]

Package	Drive Strength (mA)	Slew Rate	Shielding Required [2] for 4 < SSO < 8	Shielding Required [2] for SSO > 8
PQ208	24	High	2	3
		Low	2	2
	16	High	2	3
		Low	2	2
	12	High	0	1
		Low	0	0
	8	Any	0	0
FG484	Any	Any	0	0

Notes:

1. *Measurements were performed with a 3.3 V swing on the SSO bus.*
2. *Shielding pins required on the side of the sensitive input adjacent to the SSO bus.*

In PQ, TQ, and VQ packages, the sensitive inputs may be affected by SSOs as described in the "SSO Effect on Minimum Input Slew Rate (input maximum rise/fall time)" section on page 400. If the edge-sensitive inputs surrounded by an SSO bus rise or fall at the same time as an SSO transition, the maximum rise and fall times of those inputs should be less than 3 ns to avoid any glitches, as described the "SSO Effect on Minimum Input Slew Rate (input maximum rise/fall time)" section on page 400.

Mitigating SSO Effects on Inputs

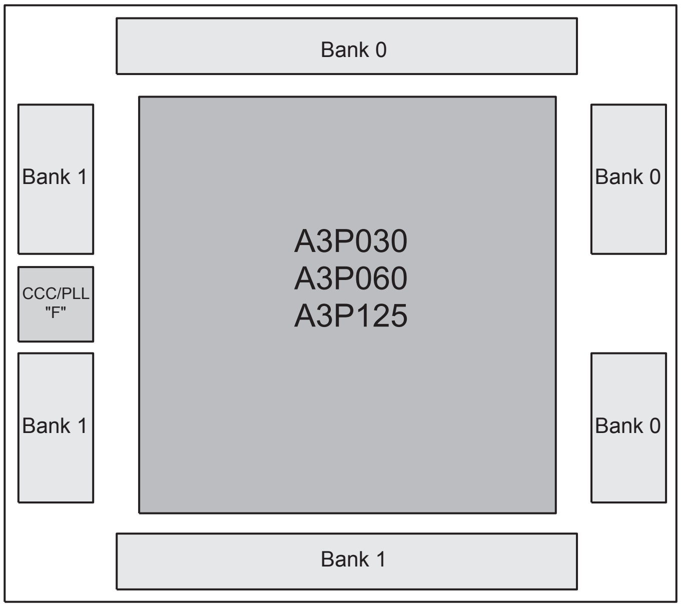
As illustrated in Table 19-1 on page 403, in PQ, TQ, and VQ packages, inputs may be affected by a surrounding SSO bus, depending on the configuration and number of the SSOs. FG and BG packages show much better SSO characteristics due to smaller lead inductance. Therefore, designers are encouraged to use these packages in designs that have SSOs and are sensitive to noise. It is also recommended that designers use the general guidelines described in "Introduction" on page 399 to eliminate SSO conditions that may cause system signal integrity problems.

In addition, experiments at Microsemi show that in ProASIC3/E devices, inputs configured with the Schmitt Trigger option are slightly more tolerant to the noise induced by an SSO bus. Therefore, Microsemi recommends that designers select the Schmitt Trigger option for critical inputs surrounded by SSOs whenever possible.

Whenever shielding is required by Table 19-4 on page 406, it is recommended to use GND or virtual ground pins as shielding (described in the "Shielding Using GND or Virtual Ground Pins" section on page 401); however, it is acceptable to use other shielding pins (as described in the "Shielding Using Other Pins" section on page 402) to protect the sensitive inputs from SSOs.

Mitigating SSO Effects on Clock Conditioning Circuits

In general, analog circuitry is more sensitive to noise than digital signals. As described in the "Shielding from SSOs" section on page 401, any sensitive signal surrounded by an SSO bus is affected by the noise induced by SSO activity. Therefore, if the analog power supply of the ProASIC3/E PLL (i.e., the VCCPLX pin) is surrounded by SSOs, the noise induced by the SSOs in the analog supply will cause an increase in the PLL output jitter. Experiments at Microsemi show that if the analog supply pin of the PLL is surrounded by two or more SSOs, the output jitter of the corresponding PLL will be increased beyond the jitter specification in the ProASIC3/E datasheet. Therefore, if PLLs are used in ProASIC3/E devices, the analog supplies of the PLLs used should be shielded from any SSOs by avoiding the placement of SSOs exceeding the guidelines given in Table 19-1 on page 403. Refer to Figure 19-6, Figure 19-7 on page 408, and Figure 19-8 on page 408 for more information about the I/O bank neighboring the PLL.



Note: The A3P030 device does not support a PLL (V_{COMPLF} and V_{CCPLF} pins).

***Figure 19-6* • Naming Conventions of ProASIC3 Devices with Two I/O Banks**

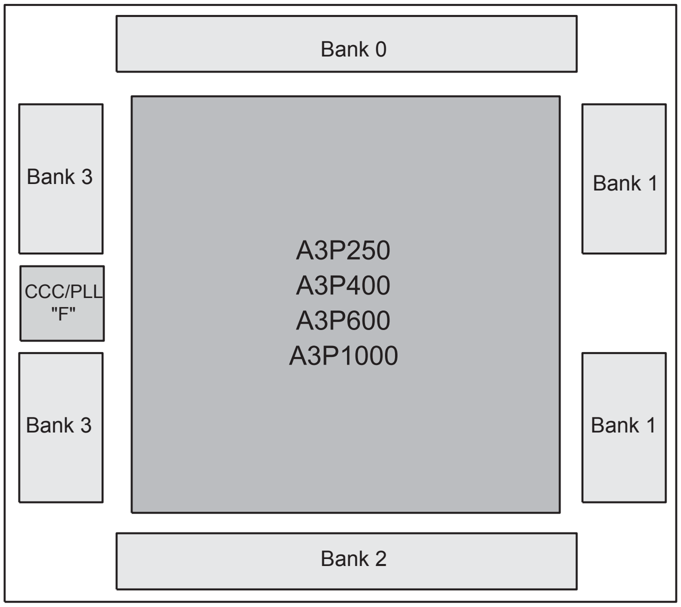


Figure 19-7 • **Naming Conventions of ProASIC3 Devices with Four I/O Banks**

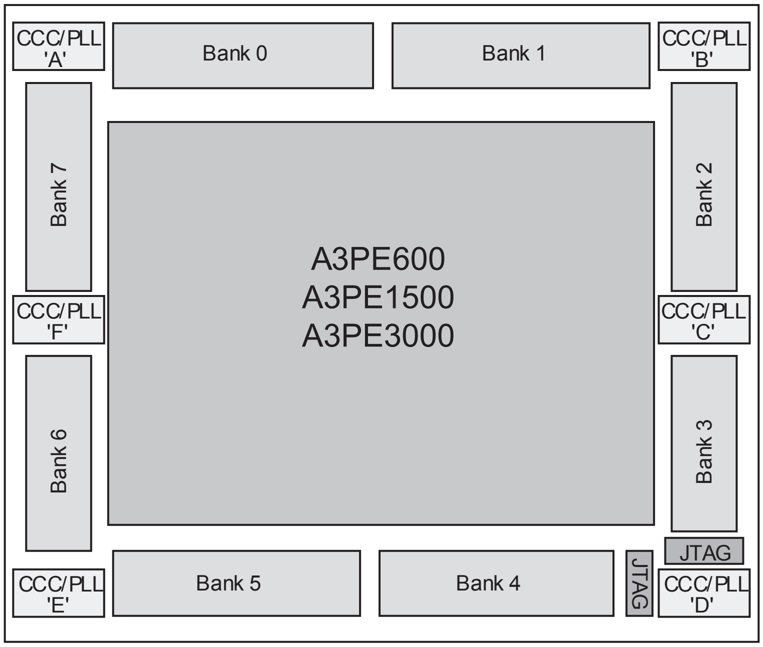


Figure 19-8 • **User I/O Naming Conventions of ProASIC3E Devices**

Conclusion

As digital designs get faster and larger, SSOs and their effects become a more critical part of system signal integrity analysis. This application note provides data characterizing and predicting the effects of SSOs on sensitive inputs and outputs in ProASIC3/E FPGAs. SSO effects should be mitigated to ensure the functionality of the design; this application note provides specific techniques for doing so.

SSO mitigation techniques should be conducted in parallel with chip-level and board-level design, as they play important roles in providing a clean digital system. For board-level design guidelines, refer to the *Board-Level Considerations* application note.

Due to the nature of SSOs, FG and BG packages are more tolerant to SSO effects than PQ or TQ packages. Therefore, for high-speed designs or designs with large numbers of SSOs, FG and BG packages are strongly recommended.

Related Documents

Application Notes

Simultaneous Switching Noise

http://www.microsemi.com/soc/documents/SSN_AN.pdf

Board-Level Considerations

http://www.microsemi.com/soc/documents/ALL_AC276_AN.pdf

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	Changes	Page
August 2012	The "Mitigating SSO Effects on Clock Conditioning Circuits" section was clarifed, referring to the guidelines given in Table 19-1 • Number of SSOs Causing Specified Ground and VCCI Bounce if PLLs are used in ProASIC3/E devices. The section previously stated that you should avoid the placement of SSOs in neighboring I/O banks (SAR 33974).	407
	The hyperlink for the *Board-Level Considerations* application note was corrected (SAR 36663).	409
June 2011	Table 19-1 • Number of SSOs Causing Specified Ground and VCCI Bounce was revised to add information for additional device packages and Table 19-2 • Ground Bounce and Shielding for A3PE3000-FG896 is new (SAR 31139).	403, 404
v1.0 (January 2008)	Figure 19-6 • Naming Conventions of ProASIC3 Devices with Two I/O Banks, Figure 19-7 • Naming Conventions of ProASIC3 Devices with Four I/O Banks, and Figure 19-8 • User I/O Naming Conventions of ProASIC3E Devices were updated.	407 – 408

20 – Metastability Characterization Report for Microsemi Flash FPGAs

Introduction

Whenever asynchronous data is registered by a clocked flip-flop, there is a probability of setup or hold time violation on that flip-flop. In applications such as synchronization or data recovery, due to the asynchronous nature of the data input to the flip-flops, the data transition time is unpredictable with respect to the active edge of the clock. The susceptibility of a circuit to reaching this metastable state can be described using a probabilistic equation. Setup or hold violations cause the output of the flip-flop to enter a symmetrically balanced transient state, called a metastable state. The metastable state is manifested in a bistable device by the outputs glitching, going into an undefined state somewhere between 1 and 0, oscillating, or by the output transition being delayed for an indeterminable time. Once the flip-flop has entered the metastable state, the probability that it will still be metastable later has been shown to be an exponentially decreasing function of time. Because of this property, a designer should simply wait for additional time after the specified propagation delay before sampling the flip-flop output so that the designer can be assured that the likelihood of metastable failure is remote enough to be tolerable. The additional time of waiting becomes shorter, even though still more than zero, as the technology improves and semiconductor devices reach higher ranges of speed.

This document discusses a description of metastability equations followed by metastability characterization of ProASIC,® ProASIC$^{\underline{PLUS}®}$, ProASIC3, and ProASIC3E FPGAs. This application note also provides examples on the usage of metastability equations.

Theory of Metastability

In general, the mean time between failures (MTBF) should be defined statically. Figure 20-1 on page 412 depicts a simple circuit, used to synchronize asynchronous data with the system clock. EQ 20-1 shows the relation between MTBF and the clock-to-out settling time of a flip-flop:

$$MTBF = e^{(Ts / \tau)} / (T_o \times f_d \times f_c)$$

EQ 20-1

$$T_s = T_{co} + T_{met}$$

EQ 20-2

In EQ 20-1 and EQ 20-2:

T_s = Total flip-flop output settling time

T_{co} = Flip-flop clock-to-out delay

T_{met} = Additional settling time added to the normal clock-to-out delay of the flip-flop before sampling the output of the flip-flop

t = Metastable decay constant.

T_0 = Metastability aperture at T_{co} = 0 ns (this parameter represents the likelihood that a flip-flop will enter a metastable state)

f_d = Data transition rate (twice the data frequency for periodic signals, since there are two transitions per period)

f_c = Clock frequency

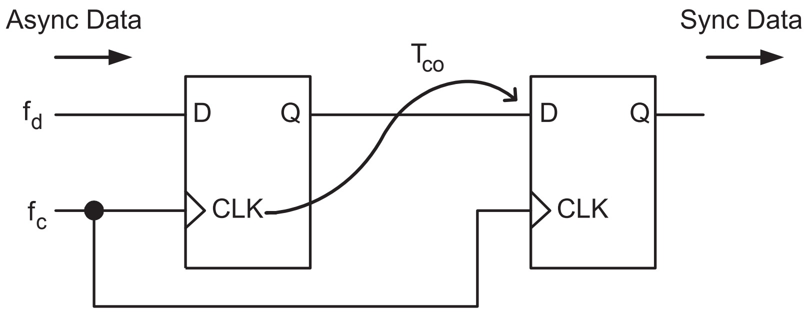


Figure 20-1 **• Example of Synchronization Circuit**

As mentioned earlier, the aperture represents the likelihood of the flip-flop entering a metastable state. The aperture is defined as a time window within the clock period. Data transitioning inside the aperture will cause the flip-flop output settling time to be greater than $T_{co} + T_{met}$. The aperture is calculated by recording the number of instances in which the settling time exceeds the specified $T_{co} + T_{met}$. The metastability aperture decreases exponentially as the allowed settling time ($T_{co} + T_{met}$) increases:

$$\text{Aperture} = T_o \times e^{-(Tco + Tmet)/\tau}$$

EQ 20-3

If the data transition occurs within the aperture, the flip-flop will stay metastable beyond the allocated settling time ($T_{co} + T_{met}$); and therefore, the second flip-flop would register invalid data (Figure 20-1). The probability of an asynchronous data transition is uniformly distributed over the clock period. Therefore, the probability of a single data transition occurring in the metastable aperture is calculated by EQ 20-4:

$$p = \text{aperture} / T_c$$

EQ 20-4

where T_c is the clock period.

In each clock cycle, the failure occurs if the data transition time is within the aperture. Therefore, the number of failures in one clock cycle can be derived by EQ 20-5:

$$n_e = n \times p = n \times (\text{aperture} / T_c)$$

EQ 20-5

where n_e represents the number of errors per clock cycle, and n is the number of data transitions per clock period (f_d / f_c).

The number of clock cycles in the operation time (N) is the total time divided by the clock period, or

$$N = T_{operation} / T_c$$

EQ 20-6

Combining EQ 20-5 and EQ 20-6 results in the total number of failures per operation time (N_e):

$$N_e = N \times n_e = (T_{operation} / T_c) \times (f_d / f_c) \times (\text{aperture} / T_c)$$

EQ 20-7

Since $T_c = 1 / f_c$, EQ 20-7 can be simplified to

$$N_e = T_{operation} \times f_d \times f_c \times \text{aperture}$$

EQ 20-8

MTBF is defined as the operation time divided by the number of failures, or

$$\text{MTBF} = 1/ (f_d \times f_c \times \text{aperture}) = 1/ (T_0 \times e^{-(Tco + Tmet) / \tau} \times f_d \times f_c)$$

EQ 20-9

FPGA Metastability Characterization

Like other FPGA manufacturers, to absorb the fixed value the of e^{Tco} term, Microsemi simplifies EQ 20-9 on page 412 to the following form:

$$MTBF = e^{C2 * Tmet} / (C1 \times f_d \times f_c)$$

EQ 20-10

where C2 is a constant inversely proportional to the metastability decay constant, and C1 is the proportionality constant, which is similar to aperture.

The FPGA metastability characterization is a series of tests conducted to identify the value of C1 and C2. There are several environmental and test condition factors that influence the characterization. These factors include but are not limited to the rise time of data and clock signals, input voltage levels, and operating voltage and temperature. Moreover, increased system noise due to switching of both internal nodes and I/Os can influence the metastability results. Therefore, it is essential to provide a suitable environment for testing.

Test Design Description

Figure 20-2 shows a schematic of the test circuit used to characterize the metastability in Microsemi devices. The propagation delay, operating under specified setup and hold time, is measured from the output of flip-flop DFF#1 to the input of flip-flop DFF#3. This value is denoted by EQ 20-11:

$$T_{min} = T_{cof}(DFF\#1) + T_{delay} + T_{su}(DFF\#3)$$

EQ 20-11

where T_{delay} is the propagation delay from output of DFF#1 to input of DFF#3, T_{cof} is the clock-to-out delay of DFF#3, and T_{su} represents the setup time requirement of DFF#3. T_{min} corresponds to the T_{co} in EQ 20-9 on page 412 and is the reference time to which the additional settling time, T_{met}, is added for characterization of metastability.

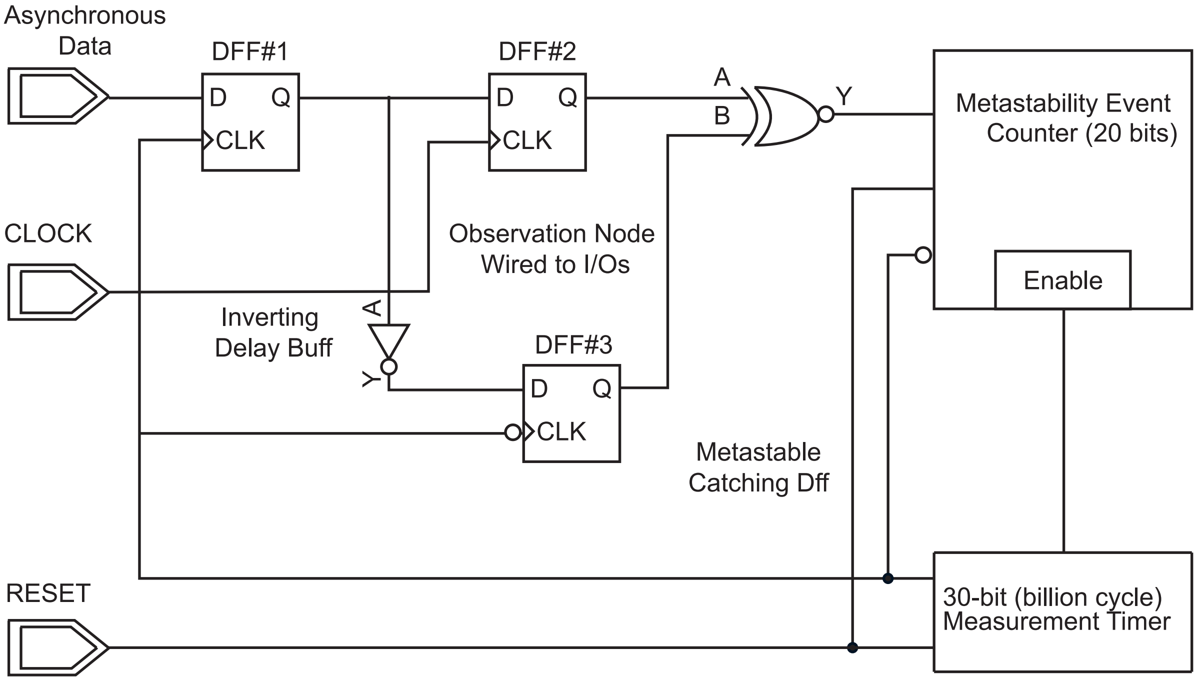


Figure 20-2 • **Test Circuit**

DFF#2 is clocked on the same edge as DFF#1. Conversely, DFF#3 must resolve the signal driven from the metastable DFF#1 before the falling clock edge. As can be seen in the design in Figure 20-2, T_{min} + T_{met} is the difference between the rising and falling edge of the clock. Therefore, it can be easily set or measured by adjusting the duty cycle of the clock signal. A detectable metastable event occurs when DFF#2 and DFF#3 are in the SAME state. In the expected operation, DFF#2 and DFF#3 are in opposite states due to the inverter in the DFF#3 input data path. The XNOR gate allows the event counter to record these metastable events. After a billion clock cycles, the counter is read and the MTBF is calculated.

In this test, T_{min} was resolved to within ±0.01% of the duty cycle at 10 MHz. This translates to an error of ±10 ps.

The other test setup parameters were as follows:

- Clock and data inputs were driven from independent pulse generators (<1 ns rise time).
- Clock input levels were from 0 V to 2.5 V. These levels were required due to the impedance matching of Microsemi's test fixture. Data input levels were 0 V to 3.3 V.
- FPGA power supplies for all tests were at V_{DDP} = 3.3 V and V_{DD} = 2.5 V.

Metastability Measurement Results

EQ 20-10 on page 413 can be reformed into EQ 20-12:

$$\ln(\mathrm{MTBF}) = C2 \times T_{met} - \ln(C1 \times f_d \times f_c)$$

EQ 20-12

The plot of EQ 20-12 is a linear relationship between ln(MTBF) and T_{met}, where C2 is the slope of the line. Figure 20-3 shows the plot of EQ 20-12 for ProASIC and ProASIC$^{\underline{PLUS}}$ FPGA families. C1 and C2 can be calculated from any two data points.

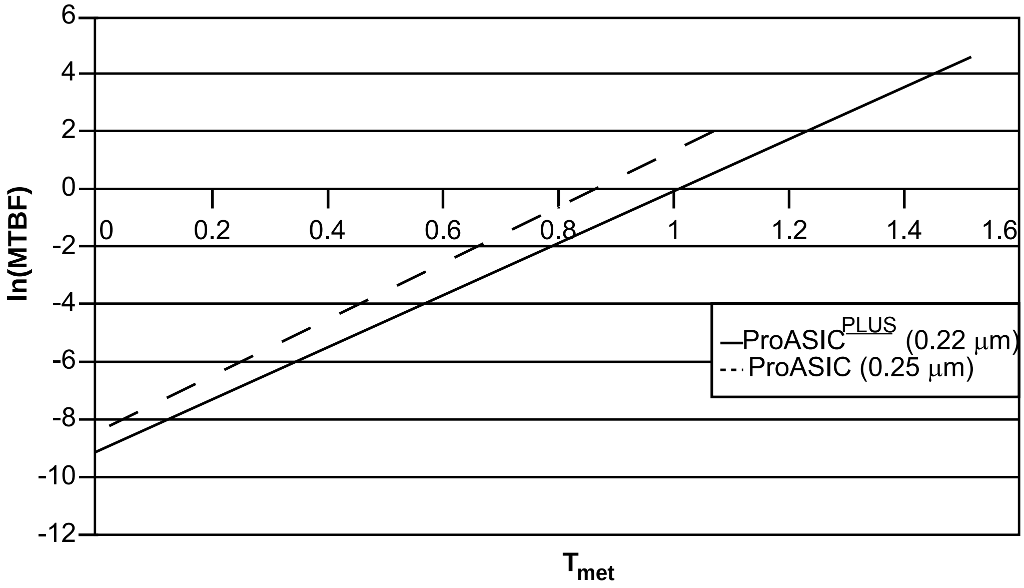


Figure 20-3 • **Metastability Comparison of Microsemi FPGA Families**

The metastability theory indicates that C1 and C2 are independent of the test clock and data frequency. The test results concur within experimental tolerances. The calculations of C1 and C2 are given in Table 20-1.

Table 20-1 **• Metastability Coefficients for Microsemi Flash FPGAs**

f_c = 10 MHz		
Device Family	**C_1 (Hz)**	**C_2 (Hz)**
ProASIC	9.95E–11	1.03E+10
ProASIC$^{\underline{PLUS}}$	1.56E–11	9.148E+09
ProASIC3/E Core Registers	9.11E–12	1.57E+10
ProASIC3/E I/O Registers	2.25E–12	1.91E+10

Examples of Metastability Coefficients Usage

Metastability shows a statistical nature, and designers should allow enough additional time (T_{met}) that the likelihood of metastable failure is remote enough to be tolerable by the design specification.

For example, consider that the simple circuit in Figure 20-1 on page 412 is implemented in a ProASIC$^{\underline{PLUS}}$ device to synchronize an asynchronous data input to the FPGA. The following parameters are given to designer by either design specification or post-layout timing analysis:

T_{co} = 10 ns, corresponding to a clock frequency of 100 MHz

Asynchronous data transition rate = 12.5 MHz

Tolerable MTBF = 1 year

If the designer does not allow additional sampling time (T_{met} = 0 ns) and run the clock at the rate of 100 MHz, EQ 20-12 on page 414 will result in MTBF = 51.2 µs. This means that a metastability error will occur at the output of the second flip-flop every 51.2 µs. This value exceeds the required MTBF of one year indicated in the design specification. To meet this requirement, the designer needs to allow additional T_{met} in the sampling time, which can be calculated as follows:

1 year = 365 × 24 × 3,600 = 31,536,000 seconds

$\ln 31{,}536{,}000 = 9.148E{+}09 \times T_{met} - \ln (1.56E{-}11 \times 100E6 \times 12.5E6) \geq T_{met} = 2.96$ ns

Therefore, an additional 3 ns sampling time will fulfill the required MTBF.

List of Changes

The following table lists critical changes that were made in each revision of the chapter.

Date	**Changes**	**Page**
June 2011	In Table 20-1 • Metastability Coefficients for Microsemi Flash FPGAs, the units for C1 and C2 were changed from seconds to Hz (SAR 29148).	415
v1.0 (January 2008)	Table 20-1 • Metastability Coefficients for Microsemi Flash FPGAs was updated to include ProASIC3/E information.	415
5190062-0	This document was updated to provide a detailed description of the calculations being made.	N/A

Microsemi

A – Summary of Changes

History of Revision to Chapters

The following table lists chapters that were affected in each revision of this document. Each chapter includes its own change history because it may appear in other device family user's guides. Refer to the individual chapter for a list of specific changes.

Revision (month/year)	Chapter Affected	List of Changes (page number)
Revision 4 (September 2012)	"Microprocessor Programming of Microsemi's Low Power Flash Devices" was revised.	370
Revision 3 (August 2012)	"FPGA Array Architecture in Low Power Flash Devices" was revised.	22
	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	115
	"SRAM and FIFO Memories in Microsemi's Low Power Flash Devices" was revised.	159
	"I/O Structures in IGLOO and ProASIC3 Devices" was revised.	196
	The "Pin Descriptions" and "Packaging" chapters were removed. This information is now published in the datasheet for each product line (SAR 34767).	N/A
	"Migrating Designs in ProASIC3 Devices from Higher-Density to Mid-Density Devices" was revised.	274
	"Migrating Designs from A3P250 to Lower-Logic-Density Devices" was revised.	306
	"In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" was revised.	361
	"Boundary Scan in Low Power Flash Devices" was revised.	376
	"ProASIC3/E SSO and Pin Placement Guidelines" was revised.	409
Revision 2 (December 2011)	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	115
	"UJTAG Applications in Microsemi's Low Power Flash Devices" was revised.	386
Revision 1 (June 2011)	"Low Power Modes in ProASIC3/E and ProASIC3 nano FPGAs" was revised.	31
	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	115
	"I/O Structures in IGLOO and ProASIC3 Devices" was revised.	196
	"I/O Software Control in Low Power Flash Devices" was revised.	218
	"Migrating Designs from A3P250 to Lower-Logic-Density Devices" was revised.	306
	"In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" was revised.	361

Revision (month/year)	**Chapter Affected**	**List of Changes (page number)**
Revision 1 (continued)	"ProASIC3/E SSO and Pin Placement Guidelines" was revised.	409
	"Metastability Characterization Report for Microsemi Flash FPGAs" was revised.	415
Revision 0 (July 2010)	The Automotive ProASIC3 Flash Family FPGAs Handbook was divided into two parts to create the Automotive ProASIC3 Flash Family FPGAs Datasheet and the Automotive ProASIC3 FPGA Fabric User's Guide.	N/A
	"Global Resources in Low Power Flash Devices" was revised.	61
	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	115
	"I/O Software Control in Low Power Flash Devices" was revised.	218
	"DDR for Microsemi's Low Power Flash Devices" was revised.	233
	"Programming Flash Devices" was revised.	320
	"In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" was revised.	361
	"Boundary Scan in Low Power Flash Devices" was revised.	373

B – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call 800.262.1060
From the rest of the world, call 650.318.4460
Fax, from anywhere in the world, 650.318.8044

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

Visit the Customer Support website (www.microsemi.com/soc/support/search/default.aspx) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the website.

Website

You can browse a variety of technical and non-technical information on the SoC home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office. Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Index

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.