Context Save and Reload

Introduction

In power-critical applications, many systems store data (their context) to memory, suspend operation, or turn off components to reduce power. Once operation resumes, power and previously stored data are restored. To perform context save and reload operations, designs must have nonvolatile memory and power management capabilities. Actel Fusion™ FPGAs offer a one-chip solution that contains power management circuitry as well as both embedded SRAM and Flash memory for context saving and reloading. Previously, separate components were needed to save SRAM and other register contents into Flash prior to shutdown. The integrated features of Fusion also support low power standby mode while maintaining critical system data and general data logging (with or without timestamp) using the integrated real-time counter (RTC). If an uncontrolled event affects the system, the system's state can be logged and reviewed when serviced.

Figure 1 shows the block diagram of a design where a system's context is stored in SRAM and permanently saved into embedded Flash. Data is then reloaded into SRAM when required.

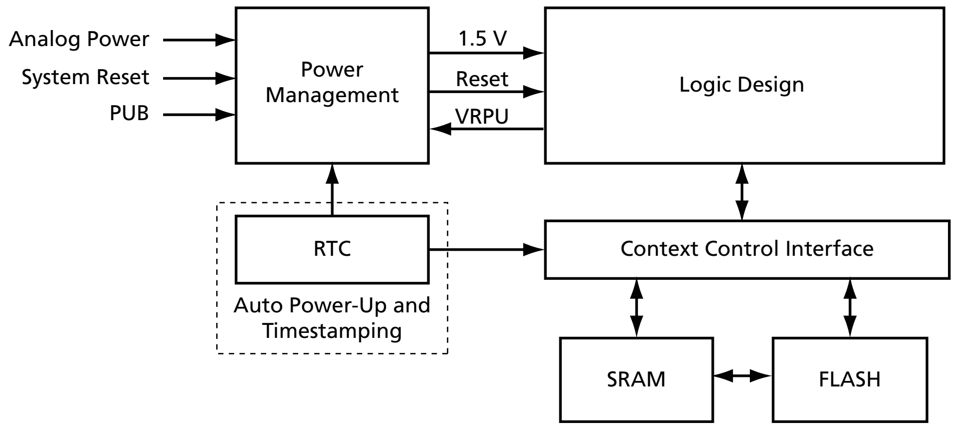


Figure 1 • **Context Save and Reload Application**

Integration Using Fusion Technology

The application and power management with context save and reload logic can be implemented in a single Fusion FPGA device. Fusion devices include embedded SRAM; Flash; RTC; analog inputs to monitor voltage, current, or temperature; gate driver outputs for power supply control; and an internal 3.3 V to 1.5 V regulator to power the FPGA fabric.

This document describes how to create a design example for performing context saving and reloading in a power management application. The first item discussed is the clock network system. Next, the document addresses a power management system using the Fusion voltage regulator and RTC. Finally, the document describes the use of embedded SRAM and Flash memory for context save and reload. All of these items can be configured with the SmartGen system interface found within Actel Libero® Integrated Development Environment (IDE). Refer to the SmartGen online help for details regarding SmartGen.

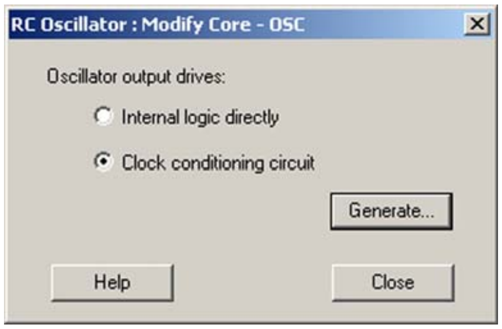
The description of the context save and reload design example is broken down into two sections. The "Peripheral Core Configuration" section on page 2 discusses the configuration of the SmartGen-generated peripheral cores integrated into the design. The "Design Implementation" section on page 7 describes the design details of using the generated cores to implement the context save and reload operation. The complete implementation is available from Actel as a design example at http://www.actel.com/techdocs/appnotes/products.aspx#fusion.

Peripheral Core Configuration

Clock Network System

RC Oscillator

The RC oscillator is an on-chip, free-running clock resource that generates a 100 MHz clock. It can be used as a source clock for both on-chip and off-chip resources. An on-chip resource commonly sourced by the RC oscillator clock is the Fusion phase-locked loop (PLL) core. The PLL can then be used to generate clocks of varying frequency, particularly to address the needs of the Fusion initialization interface used to manage a design's context. Figure 2 illustrates the proper configuration for the RC oscillator when sourcing the PLL.



RC Oscillator : Modify Core - OSC

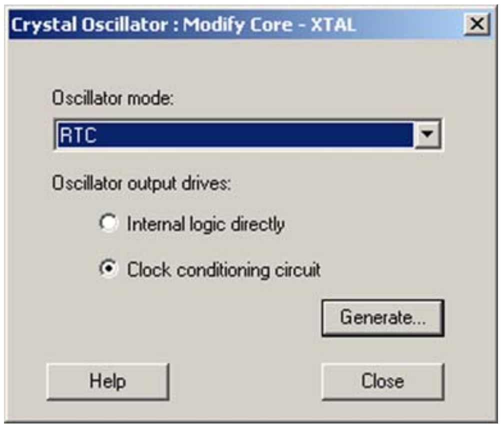
Oscillator output drives:

Internal logic directly [✓ Clock conditioning circuit]

Figure 2 • **RC Oscillator SmartGen Configuration**

Crystal Oscillator

The on-chip crystal oscillator circuit is combined with an external crystal to generate a high-precision clock. It is capable of providing system clocks for the Fusion on-chip resources as well as other on- and off-chip system clock networks. The Fusion on-chip RTC was designed to be sourced by the crystal oscillator output clock. To utilize the RTC, the crystal oscillator must be configured to operate in RTC mode. Figure 3 shows the proper mode configuration for the crystal oscillator when sourcing the RTC.



Crystal Oscillator : Modify Core - XTAL

Oscillator mode: RTC

Oscillator output drives:

Internal logic directly [✓ Clock conditioning circuit]

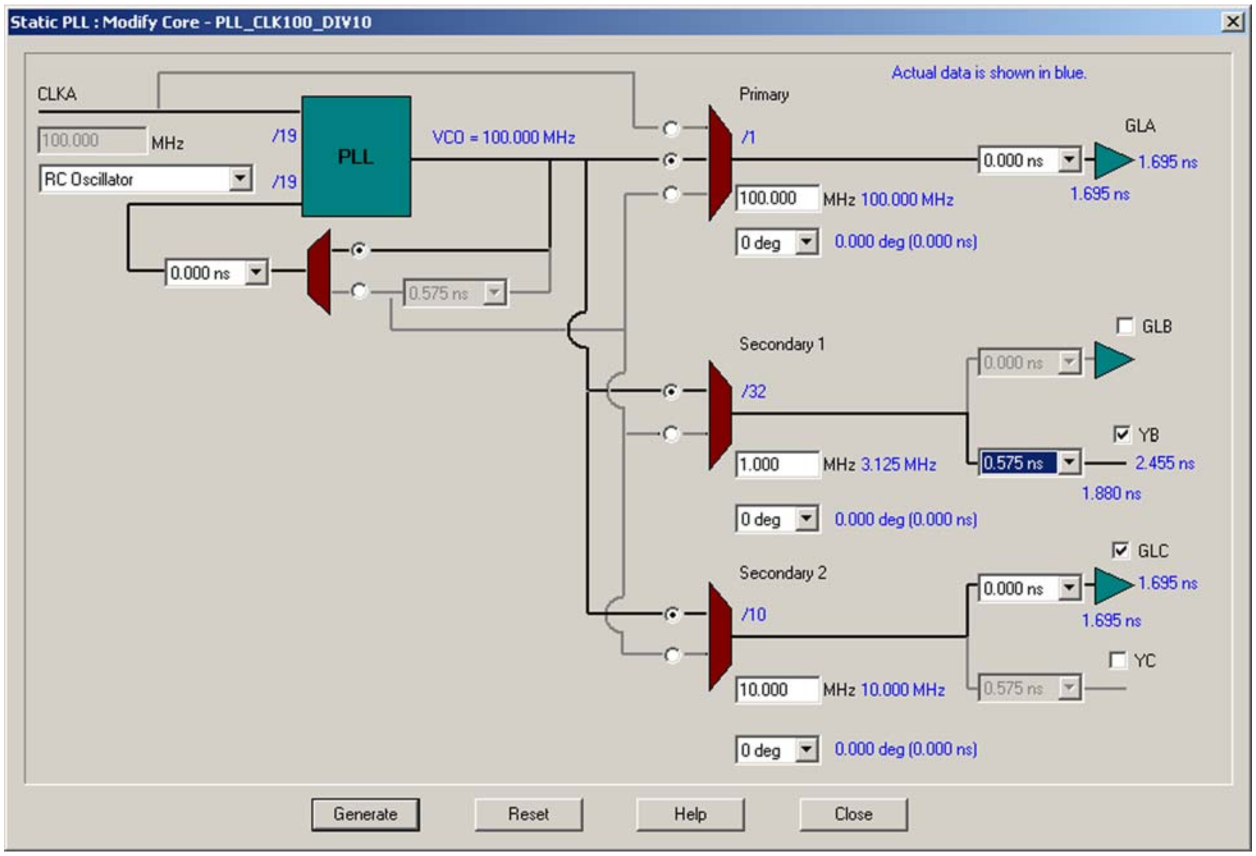
Figure 3 • **Crystal Oscillator SmartGen Configuration**

Static PLL

The on-chip PLL circuit is capable of performing a multiply, divide, synchronize, advance, or delay of its clock source. It has three different outputs, each connected to a global network in the FPGA fabric; each output has its own unique configuration.

In the context save and reload design example, the clock input of the PLL is being sourced by the 100 MHz RC oscillator, and both the GLA and GLC PLL outputs source all other clock networks. The GLA output sources the 100 MHz clock network for the system's high-speed resources. The GLC output sources the 10 MHz clock network required by the initialization interface (must operate between 1 MHz and 10 MHz), which is used to perform the context save and reload operation. The initialization interface connects resources such as the Analog System and SRAM peripherals to the Flash memory.

Figure 4 illustrates the proper configuration of the PLL GLA and GLC outputs and clock source in SmartGen. To synchronize the GLA and GLC outputs, the output of the VCO, configured for 100 MHz, directly drives the GLA output. The VCO output is then divided by 10 and drives the GLC output to generate a 10 MHz clock. If different frequencies or another clock source are required by the application, SmartGen can be used to reconfigure the PLL to meet the system requirements.


Static PLL : Modify Core - PLL_CLK100_DIV10

Actual data is shown in blue.

CLKA: 100.000 MHz, RC Oscillator, /19, /19, VCO = 100.000 MHz, 0.000 ns, 0.575 ns

Primary: /1, 100.000 MHz 100.000 MHz, 0 deg 0.000 deg (0.000 ns); GLA: 0.000 ns, 1.695 ns, 1.695 ns

Secondary 1: /32, 1.000 MHz 3.125 MHz, 0 deg 0.000 deg (0.000 ns); GLB: 0.000 ns; [✓ YB]: 0.575 ns, 2.455 ns, 1.880 ns

Secondary 2: /10, 10.000 MHz 10.000 MHz, 0 deg 0.000 deg (0.000 ns); [✓ GLC]: 0.000 ns, 1.695 ns, 1.695 ns; YC: 0.575 ns

Figure 4 • **PLL SmartGen Configuration**

No-Glitch MUX (NGMUX)

The on-chip NGMUX can be used to switch between two different clock domains. In the context save and reload design example, the NGMUX is used to switch the SRAM between the 100 MHz clock domain driving the main logic in the system and the 10 MHz initialization interface clock domain.

By design, the NGMUX CLK0 input must be connected to the GLA output, and the CLK1 input to the GLC output of the PLL. In addition, both PLL outputs driving the NGMUX inputs must have a fanout of one. SmartGen can be used to add the NGMUX core to the design with no special configuration requirements.

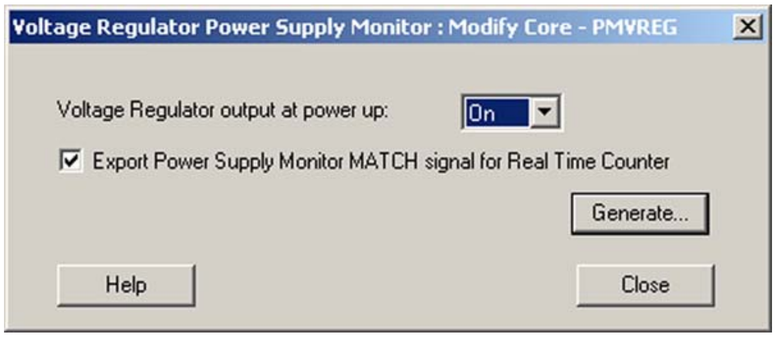
Power Management System

Voltage Regulator Power Supply Monitor (VRPSM)

The on-chip voltage regulator generates a 1.5 V output from a 3.3 V external power supply. The 1.5 V output is intended to supply power to all 1.5 V resources in the Fusion device and must be routed externally to the appropriate V_{CC} pads. When using the internal 1.5 V regulator, the TRST pad must be pulled down using a 470 Ω resister to ensure proper functionality.

The VRPSM provides a power-saving shutdown feature (standby mode) where the FPGA fabric can be powered down either by an external pin or an internal net through the VRPU core input. The voltage regulator can wake from standby and supply power to the FPGA fabric either via an external pin (PUB) or automatically with the on-chip RTC. A simple RC circuit, externally connected to the 1.5 V output, can then be used to generate a power-on reset signal input to automatically reset the FPGA fabric.

In the context save and reload design example, the VRPSM core is configured in SmartGen to supply the 1.5 V output at power-up and to wake from standby via the RTC match output signal (RTCPSMMATCH). Figure 5 illustrates the proper configuration of the VRPSM core in SmartGen.


Voltage Regulator Power Supply Monitor : Modify Core - PMVREG

Voltage Regulator output at power up: On

[✓] Export Power Supply Monitor MATCH signal for Real Time Counter

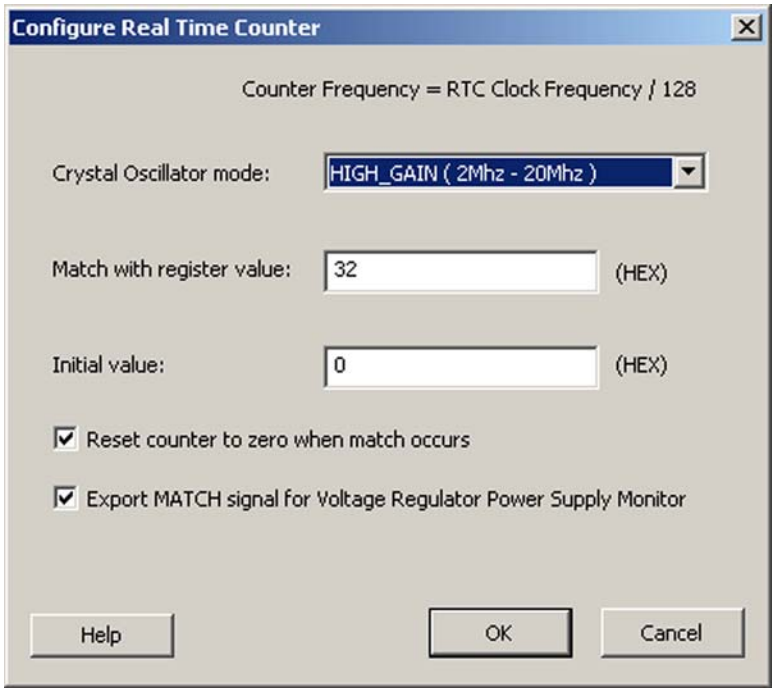
Figure 5 • **Voltage Regulator SmartGen Configuration**

Real-Time Counter

The on-chip RTC is a 40-bit synchronous counter whose clock is sourced by the crystal oscillator for high-precision counts. It has a 40-bit match register and two match output flags (RTCMATCH and RTCPSMMATCH). The RTC is powered by the Fusion 3.3 V power supply and can be used to enable the 1.5 V supply to the FPGA fabric, as described in the "Voltage Regulator Power Supply Monitor (VRPSM)" section.

The RTC is considered part of the Fusion Analog System and is accessible through its Analog Configuration MUX (ACM) interface. The ACM interface provides the FPGA fabric access to the contents of the RTC's 40-bit counter and match register for the implementation of a digital timestamp. It also allows for the application's main system logic in the FPGA fabric to modify the contents of the match register dynamically, which changes the period of the match output flags. Modification of the match register contents changes the time the FPGA fabric remains in standby. The ACM interface must operate at a frequency between 1 MHz and 10 MHz. Direct access to the ACM bus can be enabled through SmartGen under the Advanced Options of the Analog System Builder configuration window. In the context save and reload design example, the digital timestamping feature is not implemented; therefore, the ACM bus is not directly accessed.

Figure 6 on page 5 illustrates the proper configuration of the RTC core in SmartGen for the context save and reload application. Once the RTC peripheral is added to the Analog System in the Analog System Builder, the RTC configuration window opens. The crystal oscillator mode and the 40-bit counter and match register initial values can all be set in the SmartGen RTC configuration window. For the context save and reload design example, the crystal oscillator is configured in high-gain mode, and a 10 MHz frequency is applied to the XTL input of the crystal oscillator core. The 40-bit counter initial value is set to zero, and the match register initial value is set to 0x32, waking the device from standby every 0.64 ms. The RTC is also configured to reset after a match occurs and to export the RTCPSMMATCH output to the VRPSM core. If a different XTL frequency or match register value is required for a specific application, SmartGen can be used to reconfigure the RTC to meet the system requirements.


Configure Real Time Counter

Counter Frequency = RTC Clock Frequency / 128

Crystal Oscillator mode: HIGH_GAIN (2Mhz - 20Mhz)

Match with register value: 32 (HEX)

Initial value: 0 (HEX)

☑ Reset counter to zero when match occurs

☑ Export MATCH signal for Voltage Regulator Power Supply Monitor

Figure 6 • **RTC SmartGen Configuration**

Embedded Memory System

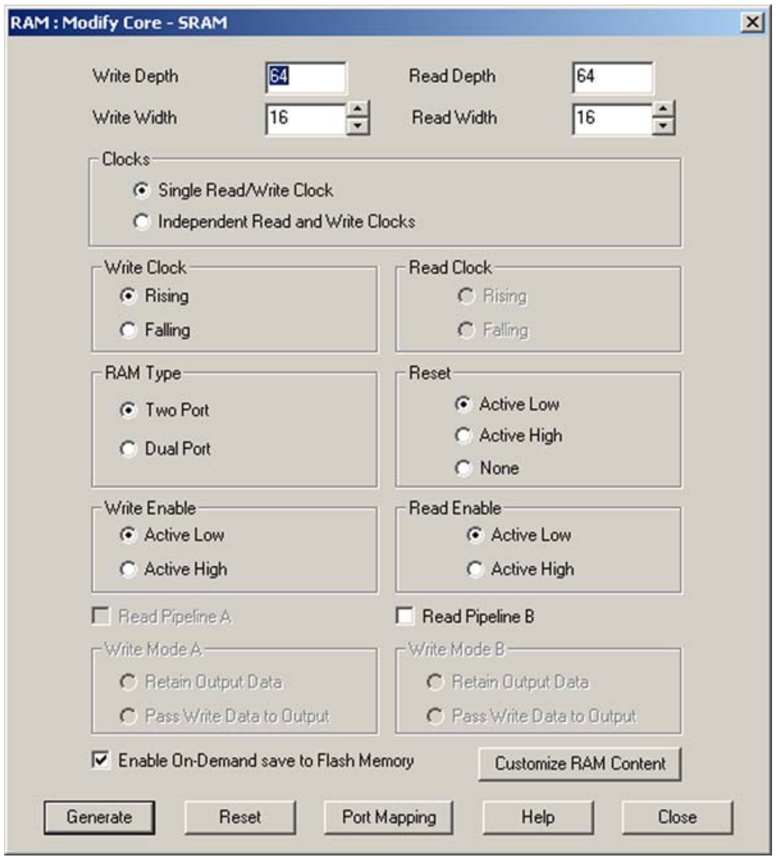
SRAM

The embedded SRAM blocks available on Fusion devices have up to 4,608 bits each and can be configured in variable depths and widths. They can be configured as a two- or dual-port memory with single or independent read and write clocks. The SRAM blocks can be initialized from the Flash memory blocks.

In a context save and reload application, the SRAM blocks are used to store the system context. The context in the SRAM blocks is then permanently saved in the Fusion embedded Flash memory blocks using the system's on-demand save feature. At power-up, the SRAM is then automatically initialized with the system context stored in Flash.

In this design example, the SRAM with initialization memory configuration is selected through SmartGen. By selecting this particular memory core, a separate initialization port is added to the SRAM to be directly connected to the Flash memory for the on-demand save feature. The SRAM core is configured as a two-port, single-read/write-clock memory block with both read and write access arranged in 64 rows, each 16 bits wide. Figure 7 on page 6 shows the proper configuration of the SRAM core. If a different size or configuration of the SRAM is required for an application, SmartGen can be used to reconfigure the SRAM core to meet the system requirements.

The initialization interface added between the SRAM and Flash memories must operate at a frequency between 1 MHz and 10 MHz. The SRAM initialization interface port shares its clocks with the standard clock inputs—RWCLK, RCLK(A/B), and WCLK(A/B). Therefore, if the application's main logic design requires high-speed transactions, the SRAM's clock network must be designed to cross clock domains, for example, with the use of the NGMUX core. For the Fusion dual-port memory, the initialization interface port shares port A's clock for the SRAM update from Flash and port B's clock for the SRAM save to Flash memory.



Write Depth: 64
Read Depth: 64
Write Width: 16
Read Width: 16

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

Enable On-Demand save to Flash Memory: ✓

Figure 7 • **SRAM SmartGen Configuration**

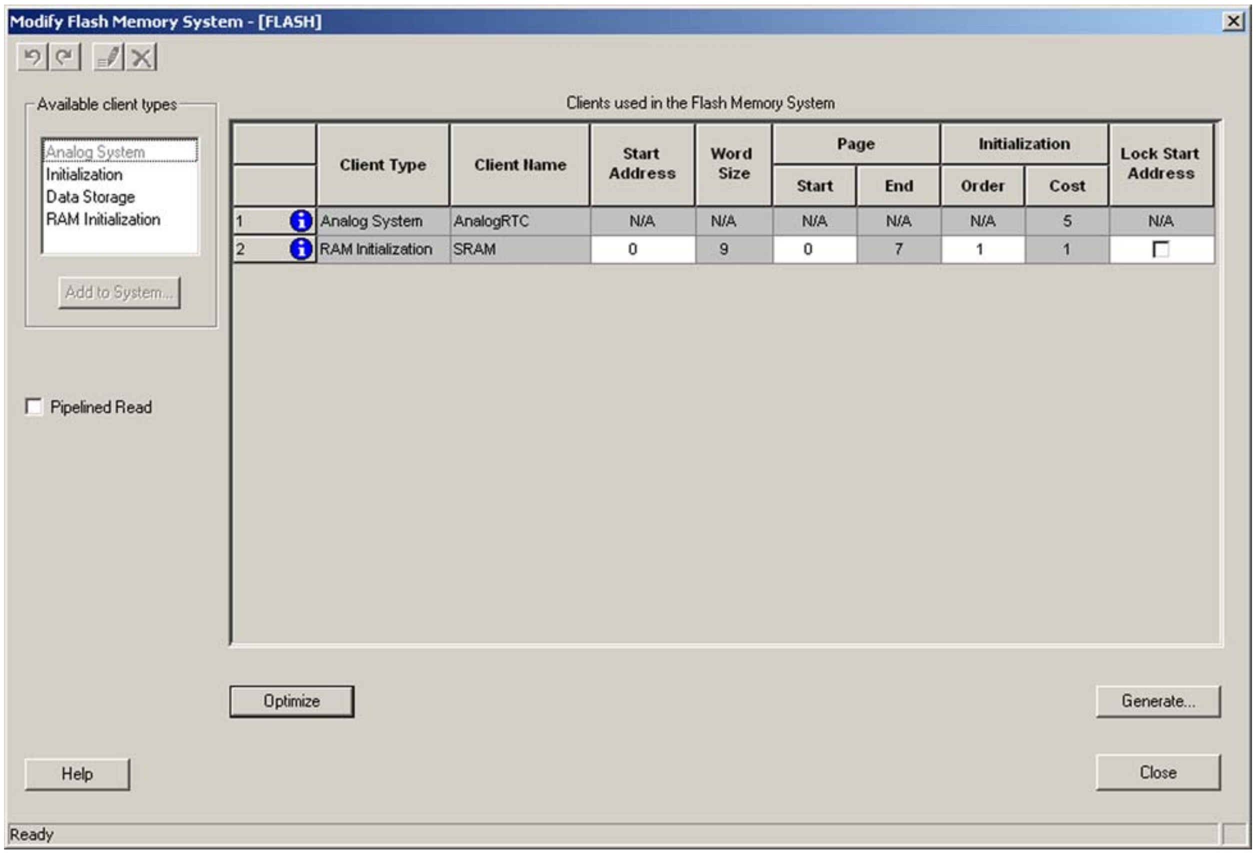
Flash Memory

One unique feature of a Fusion device is its embedded Flash memory. Each Flash memory block contains 256 kbytes of nonvolatile memory and the associated circuitry required to easily read, modify, and optionally rewrite the memory contents from the FPGA fabric.

The embedded Flash is general purpose and may also be used to implement ROM-coded state machines, to store microcode instructions for an embedded processor, or for any other storage functionality required by the system application.

When paired with the SRAM and initialization interface port, the embedded Flash memory can be used to permanently store the system context, with or without a timestamp, before the FPGA fabric powers down or enters standby mode. Once the FPGA fabric is powered or wakes from standby, the context is reloaded (via the initialization interface) into the SRAM for system access.

The Fusion Flash memory is also paired with other resources, including the on-chip Analog System block. Therefore, when configuring the Flash memory through SmartGen for the context save and reload design example, both the Analog System (RTC resource) and SRAM initialization client types must be added to the Flash configuration. Figure 8 on page 7 illustrates the proper configuration of the Flash memory core in SmartGen.


Modify Flash Memory System - [FLASH]

Available client types: Analog System, Initialization, Data Storage, RAM Initialization

Add to System...

☐ Pipelined Read

Clients used in the Flash Memory System

	Client Type	Client Name	Start Address	Word Size	Page		Initialization		Lock Start Address
					Start	End	Order	Cost	
1	Analog System	AnalogRTC	N/A	N/A	N/A	N/A	N/A	5	N/A
2	RAM Initialization	SRAM	0	9	0	7	1	1	☐

Ready

Figure 8 • **Flash Memory SmartGen Configuration**

Design Implementation

A context save and reload design in a power management system requires digital and analog circuits, all available in Fusion FPGA devices. Fusion devices offer a variety of peripherals whose design cores can easily be configured using SmartGen to meet an application's system requirements.

A context save and reload power management application can be implemented using the key Fusion technology design systems and a custom context control logic block. The following are the key technology design systems:

- Clock Network System—includes the RC and crystal oscillators, the PLL, and the NGMUX peripherals used to source the system logic clocks across multiple clock domains.
- Power Management System—includes the RTC and VRPSM to place the FPGA fabric in power-saving standby mode and automatically exit from standby.
- Embedded Memory System—includes both the SRAM and Flash for system context saving and reloading when entering or exiting power-saving standby mode.
- Context Control System—includes a custom state machine that generates the control signals for the entire subsystem.

The following sections describe a context save and reload design example implemented on a Fusion AF600 256-pin FBGA device.

Functional Block Description

Figure 9 illustrates the context save and reload design example in a power management system. The top-level system I/Os of the design are described in Table 1 on page 10.

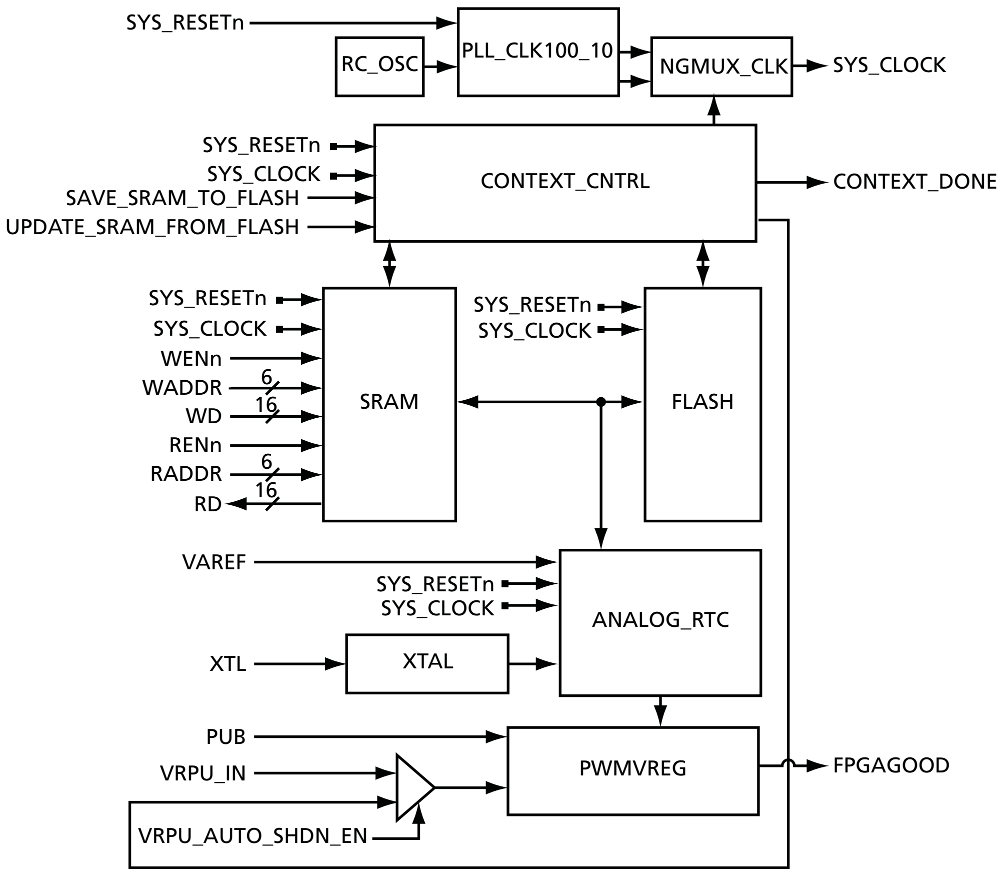


Figure 9 • **Context Save and Reload Block Diagram**

Implementation Details

PM_CONTEXT_SAVING_TOP Module

The top module of the context save and reload design example is the PM_CONTEXT_SAVING_TOP block, whose module hierarchy is described in Figure 10 on page 9. The top module contains all interconnects between the individual cores generated from SmartGen and the custom context control module (CONTEXT_CNTRL). Refer to the *SmartGen Cores Reference Guide* for additional details related to the usage of each core. All other modules required for the system application can be added to the top module, including the application's main top logic block.

The main top logic block would eventually be responsible for issuing the read/write commands to the SRAM block and the context control commands to the CONTEXT_CNTRL module. Currently, the SRAM and context control command signals are brought out through the PM_CONTEXT_SAVING_TOP module to be controlled in the testbench for verification.

The SYS_RESETn, PUB, and VAREF input signals, in an actual application design, would be directly connected to Fusion input pads. Refer to the *Fusion Family of Mixed-Signal Flash FPGAs* datasheet for pad details and pin locations. The XTL input signal's frequency is equivalent to the frequency of the external crystal connected to the XTAL1 and XTAL2 Fusion device pads. The VRPU_IN input signal can be connected to a Fusion device pad or an internal net.

PM_CONTEXT_SAVING_TOP

- RC_OSC: RC oscillator core generated from SmartGen
- PLL_CLK100_10: PLL core generated from SmartGen
- NGMUX_CLK: NGMUX core generated from SmartGen
- XTAL: Crystal oscillator core generated from SmartGen
- ANALOG_RTC: Analog System core, including the RTC, generated from SmartGen
- PMVREG: Voltage regulator power monitor core generated from SmartGen
- CONTEXT_CNTRL: Custom context control state machine module
- SRAM: SRAM core generated from SmartGen
- FLASH: Flash core generated from SmartGen

Figure 10 • **PM_CONTEXT_SAVING_TOP Module Hierarchy**

The FPGAGOOD output signal is used by the testbench to generate the SYS_RESETn input. The FPGAGOOD output is set to logic '1' when the FPGA fabric is powered and '0' when not powered. In an actual application design, a simple external RC circuit connected to the 1.5 V output of the 3.3 V power supply's external pass transistor can be used to generate the active low system reset (SYS_RESETn) pulse. Refer to the "1.5 V Voltage Regulator" section of the *Fusion Family of Mixed-Signal Flash FPGAs* datasheet for details regarding the external power supply connections. Alternatively, the PLL core's LOCK output signal can be used as an internal power-on reset solution.

The SYS_CLOCK output signal is used by the testbench to synchronize the SRAM read and write commands and the context control commands to the CONTEXT_CNTRL module. SYS_CLOCK would eventually source the application's main top logic block, which would be required to generate all command signals.

- Context control command inputs: SAVE_SRAM_TO_FLASH, UPDATE_SRAM_FROM_FLASH
- SRAM read and write signals: WENn, RENn, WADDR[5:0], RADDR[5:0], WD[15:0], RD[15:0]

The CONTEXT_DONE output signal is used by the testbench to determine when the initialization interface for context save and reload is inactive. When the CONTEXT_DONE output is at logic '1', the testbench is free to issue SRAM or context control commands to the system. However, when CONTEXT_DONE is at logic '0', the testbench must wait until the initialization interface completes its activity before issuing further commands. The CONTEXT_DONE output also indicates to which clock domain SYS_CLOCK is connected. When CONTEXT_DONE is at logic '1', SYS_CLOCK is connected to the 100 MHz clock domain. When '0', SYS_CLOCK is connected to the 10 MHz clock domain. In an actual application design, the CONTEXT_DONE signal would be internally monitored by the application's main top logic block.

The VRPU_AUTO_SHDN_EN input signal is controlled by the testbench. It behaves as a selection to a two-input MUX that either enables (if '1') the auto shutdown feature of the voltage regulator or selects command shutdown mode (if '0') via the VRPU_IN input signal. Auto shutdown of the voltage regulator is implemented through the CONTEXT_CNTRL module. After a context save operation completes execution, CONTEXT_CNTRL issues a pulse to the VRPU input of the PMVREG core through the MUX to command a shutdown of the FPGA fabric. This feature offers a one-step context save and shutdown implementation. In an actual application, the VRPU_AUTO_SHDN_EN input can be shorted to either V_{CC} or GND or controlled by the application's main top logic block.

Table 1 • **ContextMain I/O Signals**

Signal Name and Range	Signal Direction	Description
SYS_RESETn	Input	The externally generated active low system reset input from the FPGA pad used to initialize the FPGA internal circuits
SYS_CLOCK	Output	10/100 MHz clock output of the NGMUX block. The signal is brought to the top for testbench synchronization of the controlling inputs.
XTL	Input	The external crystal input to the crystal oscillator's XTAL core. The input represents the frequency generated by XTAL1 and XTAL2 pad inputs when an external crystal is connected to the FPGA.
VAREB	Input	The analog reference voltage FPGA pad input to the analog system's AnalogRTC core
PUB	Input	The power-up bar FPGA pad input to the VRPSM's PMVREG core
VRPU_IN	Input	The voltage regulator power-down input to the VRPSM's PMVREG core. This input can be driven by an FPGA pad or an internal net.
FPGAGOOD	Output	Active high output of the PMVREG core that indicates when the FPGA is logically functional. It is used in the testbench to generate the SYS_RESETn pulse.
VRPU_AUTO_SHDN_EN	Input	Active high input driven by an FPGA pad or an internal net, used to enable the auto shutdown feature of the CONTEXT_CNTRL module.
WENn	Input	Active low write enable input to the SRAM
RENn	Input	Active low read enable input to the SRAM
WADDR[5:0]	Input	Write address input bus to the SRAM
RADDR[5:0]	Input	Read address input bus to the SRAM
WD[15:0]	Input	Write data input bus to the SRAM
RD[15:0]	Output	Read data output bus from the SRAM
SAVE_SRAM_TO_FLASH	Input	Active high input driven by an internal net, currently the testbench, used to command the CONTEXT_CNTRL module to save the context from SRAM to Flash memory.
UPDATE_SRAM_FROM_FLASH	Input	Active high input driven by an internal net, currently the testbench, used to command the CONTEXT_CNTRL module to update the context in SRAM from Flash memory.
CONTEXT_DONE	Output	Active high output driven from the CONTEXT_CNTRL module indicating that the initialization interface has completed its transaction. The output is currently used in the testbench to control the generation of commands.

Clock Network System Interconnect

The clock network generates two independent clocks and is comprised of three clock domains in total. The first clock domain is the 100 MHz high-speed clock network that passes through the NGMUX and presents itself on the SYS_CLOCK net. The second clock domain is the 10 MHz initialization clock network that also passes through the NGMUX and presents itself on the SYS_CLOCK net. The last clock domain is the external crystal 10 MHz clock network that passes through the crystal oscillator core output as RTC_CLOCK. Figure 11 illustrates the connections between the clock network system cores generated by SmartGen.

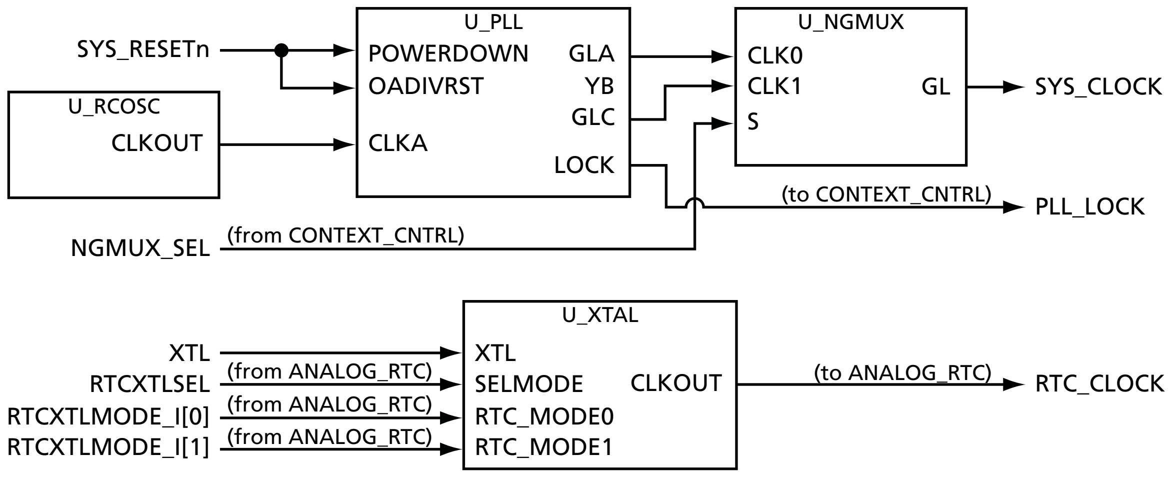


Figure 11 • **Clock Network Connection Diagram**

The simplest clock network is RTC_CLOCK, which is only used to supply the clock source to the Analog System's RTC circuit. In an actual design, the external crystal and RTC count frequencies are the main issues concerning this clock network. The RTCXTSEL and RTCXTLMODE inputs are brought directly from the ANALOG_RTC core outputs. Both signal states are configured through the RTC configuration window in SmartGen.

The SYS_CLOCK clock network is a bit more complicated because two clock domains are involved. The RC_OSC core is used to generate the source clock for the PLL core. The PLL core then generates the 100 MHz and 10 MHz synchronous clock domains. Both clocks are passed through the NGMUX, and the NGMUX_SEL output signal from the CONTEXT_CNTRL module is used to dynamically switch between the two clock domains.

The NGMUX is designed so the GLA output of the PLL core (100 MHz clock) must connect directly to the CLK0 input with a fanout of one. Similarly, the GLC output of the PLL core (10 MHz clock) must be connected to the CLK1 input with a fanout of one. The GLA, GLC, and GL outputs all occupy a global network in the FPGA fabric. The YB output of the PLL core is not used in this design but can be configured and used to source other logic blocks in the system if needed.

The PLL core generates a LOCK signal (PLL_LOCK) that is used in the CONTEXT_CNTRL module to keep the state machine in the IDLE state until the PLL's VCO has reached the desired frequency and locked into the appropriate phase.

Power Management System Interconnect

The power management system is comprised of the PMVREG and ANALOG_RTC cores and a MUX. Figure 12 illustrates the required connections to the PMVREG and ANALOG_RTC cores generated by SmartGen.

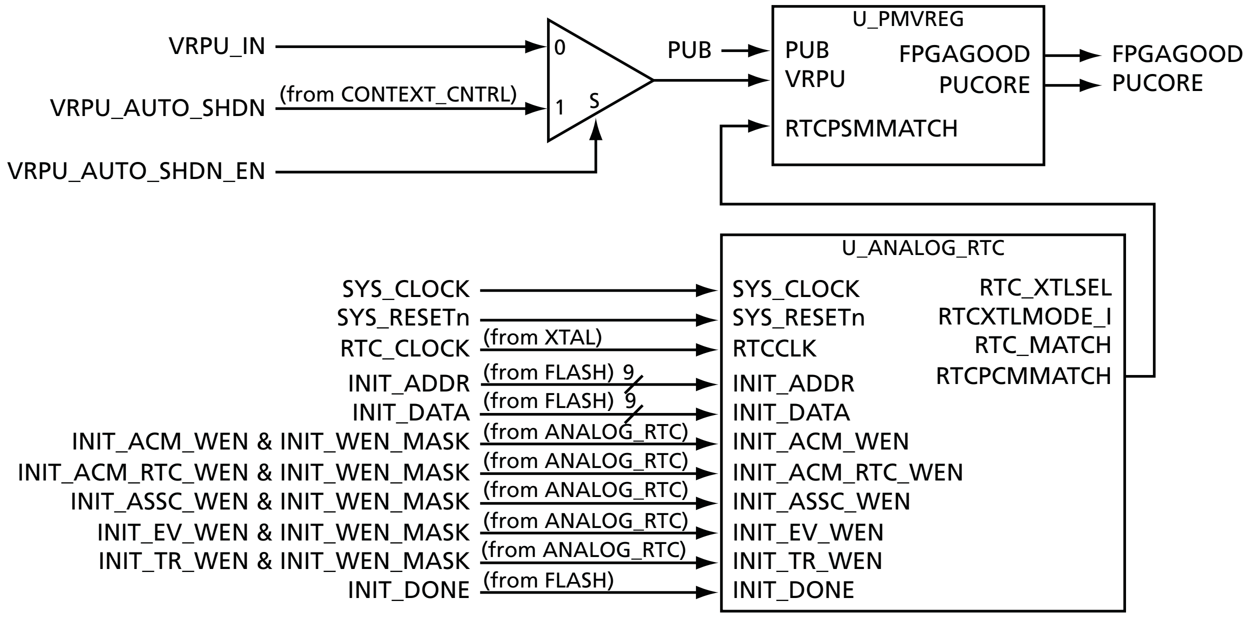


Figure 12 • **Power Management Connection Diagram**

The PMVREG core has three inputs and two outputs. The PUB and VRPU inputs are used to turn on or turn off the 1.5 V regulator, respectively. The PUCORE and FPGAGOOD outputs are used to indicate when the FPGA has been powered and is functionally good. The 1.5 V supply must be externally routed to the FPGA fabric V_{CC}. The PMVREG core does not provide the actual internal PTBASE and PTEM port connections to the pad for the external 3.3 V power supply portion of the circuit. The pads are internally pre-wired in the analog section of the FPGA to the 1.5 V regulator. For additional details regarding the external connection of the 1.5 V regulator, refer to the *Fusion Family of Mixed-Signal Flash FPGAs datasheet*.

An additional external connection, needed but not mentioned in the datasheet, is the SYS_RESETn input to the FPGA. An external system reset is needed for the context save and reload operation to work in a power management environment. An easy way to generate an external power-on reset from the 1.5 V regulator output is to add an external RC circuit to the 3.3 V power supply. The resulting signal should then be externally connected to the SYS_RESETn input pad.

The PUB input pad can be used to power the 1.5 V regulator. However, in the context save and reload design example, the RTCPCMMATCH signal is used instead. The RTCPCMMATCH signal is an active high pulse generated by the Analog System’s RTC circuit. The rising edge of this signal is used by the voltage regulator to detect a power-up condition.

The VRPU input to the PMVREG core controls the power-down action of the voltage regulator. The falling edge of the VRPU input is used to detect a voltage regulator power-down condition. A MUX is externally connected to the PMVREG core to accommodate the auto shutdown (VRPU_AUTO_SHDN) feature from the CONTEXT_CNTRL module. If the VRPU_AUTO_SHDN_EN signal is at logic '1', the VRPU_AUTO_SHDN signal drives the VRPU input to power the voltage regulator down after a context save (from SRAM to Flash memory) operation completes. However, if the VRPU_AUTO_SHDN_EN signal is at logic '0', the VRPU_IN signal drives the VRPU input and must be controlled externally or through the application’s main system logic block.

Embedded Memory System Interconnect

The embedded memory system consists of both the SRAM and Flash memory peripherals. The SRAM is used by the application's main logic block to store the application's system context. The Flash memory is then interfaced with the SRAM and other resources, such as the Analog System's RTC, via the initialization interface for permanent storage of its context. The initialization interface is controlled by the initialization control circuit embedded in the Flash core for the context save and reload interactions among dependent peripherals. Figure 13 illustrates the connections between the embedded memory cores generated by SmartGen.

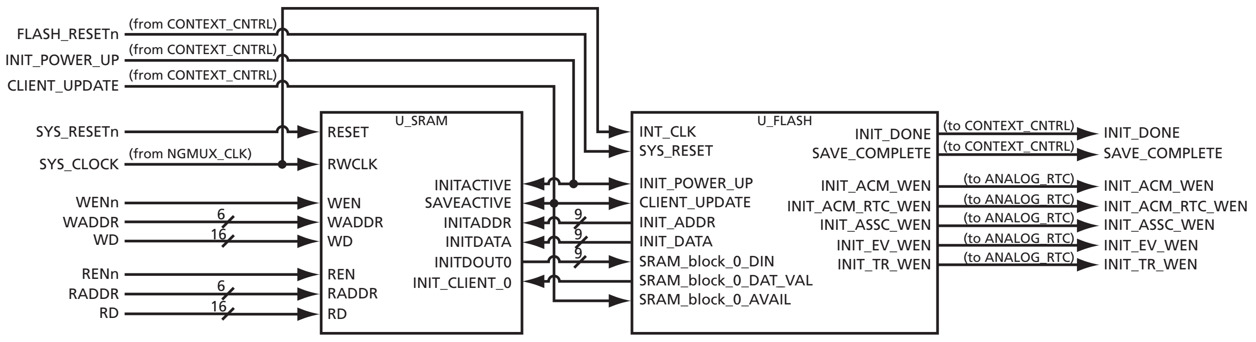


Figure 13 • **SRAM and Flash Connection Diagram**

At power-up, the CONTEXT_CNTRL module issues the command to begin the context reload (update) operation to the initialization control circuit. To perform a context save (from SRAM to Flash memory) operation, the application's main logic block (before a power-down) must issue a context save command to the CONTEXT_CNTRL module. The CONTEXT_CNTRL module then instructs the Flash core to perform the context save operation.

The Flash inputs driven by the CONTEXT_CNTRL module consist of the following signals:

- The FLASH_RESETn signal connects to the SYS_RESET port of the Flash core. It is used as the initialization control circuit's active low reset and must be issued prior to activating the initialization interface's context reload operation. The FLASH_RESETn signal allows for the on-demand context reload function implemented in the CONTEXT_CNTRL state machine.
- The INIT_POWER_UP signal is an active high output from the CONTEXT_CNTRL module and activates the initialization control circuit in the Flash memory. When high, the initialization interface becomes operational and all peripheral control signals become functional.
- The CLIENT_UPDATE signal commands the initialization control circuit to perform a context save operation from the SRAM to the Flash memory. The CLIENT_UPDATE signal is also connected to the SAVE_ACTIVE port of the SRAM core and the SRAM_block_0_AVAIL port of the Flash core. These additional connections are used to change the SRAM read port output range from [15:0] to [9:0] and make the SRAM available to the Flash initialization control circuit.

The INIT_DONE and SAVE_COMPLETE outputs of the Flash core are connected to the CONTEXT_CNTRL module. Both signals are active high and indicate when the context reload or context save operation is complete. Once a context reload completes, the INIT_DONE signal becomes active and remains high until the FLASH_RESETn signal triggers a reset. Once a context save completes, the SAVE_COMPLETE signal pulses high and remains high for several cycles before falling low.

The initialization interface address (INIT_ADDR[8:0]) and data (INIT_DATA[8:0]) bus interconnects are shared among all Flash-dependent peripherals that require initialization. The SRAM has an additional data bus (INITDOUT0[8:0]) port that supplies the write data to the Flash memory during a context save operation.

Since the initialization interface is shared by multiple peripherals, the control circuit within the Flash core must enable the peripheral's memory for updating. The SRAM_block_0_DAT_VAL output port of the Flash core (connected to the SRAM's INIT_CLIENT_0 input port) indicates to the SRAM that valid write data is available on the interface data port. The INIT_*_WEN[1] output ports connected to the ANALOG_RTC core are write enables for the RTC's configuration registers. However, the on-demand context reload feature of the CONTEXT_CNTRL module can update the RTC configuration registers even after its initial update at power-up has occurred. Therefore, an INIT_WEN_MASK output is used to mask the INIT_*_WEN signals prior to the ANALOG_RTC core connections to prevent unwanted updates of the RTC. If other peripherals requiring initialization are added to the system, the INIT_WEN_MASK signal can also be used to prevent unwanted updates of its configuration registers or memory.

The application's main logic block must fill the SRAM with its system context via the SRAM's standard read and write I/O ports. Read and write port access should occur only when the CONTEXT_DONE output port from the CONTEXT_CNTRL module is high. The read and write ports behave like typical synchronous SRAM I/Os. Refer to the *Fusion Family of Mixed-Signal Flash FPGAs datasheet* for details regarding the read and write ports and timing waveforms.

Context Control System

The context control system is designed to control the general operation of the clock, power management, and embedded memory system for a context saving and reload application. Figure 14 illustrates the input and output port connections of the CONTEXT_CNTRL module. The control operation of the context system is implemented using a 2-bit (four-state) state machine. The state diagram is shown in Figure 15.

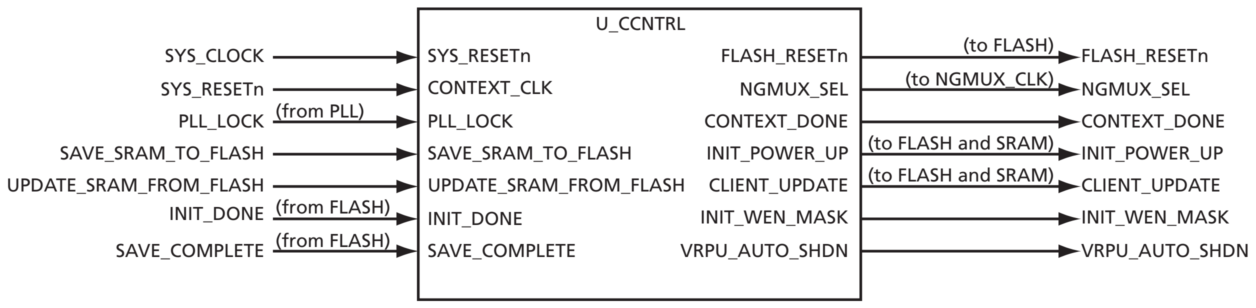


Figure 14 • **Context Control Connection Diagram**

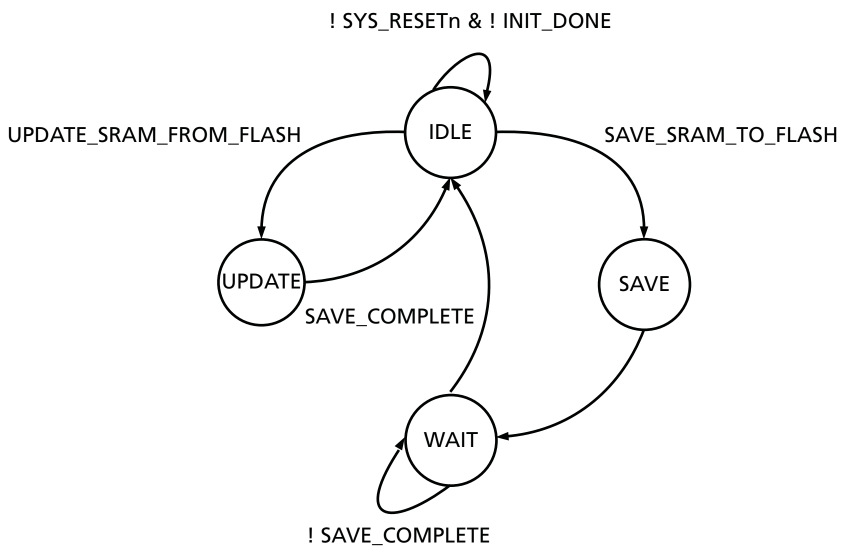


Figure 15 • **CONTEXT_CNTRL State Diagram**

1. *INIT_*_WEN signals consist of the INIT_ACM_RTC_WEN, INIT_ACM_WEN, INIT_ASSC_WEN, INIT_EV_WEN, and INIT_TR_WEN ports.*

The CONTEXT_CNTRL module operates in both the 100 MHz and 10 MHz clock domains via the SYS_CLOCK output signal from NGMUX_CLK core. The CONTEXT_CNTRL module has several I/Os used to manage the system's clock networks. The NGMUX_SEL output of the CONTEXT_CNTRL module switches the context save and reload system between the 100 MHz and 10 MHz clock domains. When the NGMUX_SEL signal is '0', SYS_CLOCK is sourced by a 100 MHZ clock. The application's main logic block is then free to issue read/write commands to the SRAM core or issue the context save or reload commands to the CONTEXT_CNTRL module. When the NGMUX_SEL signal is '1', SYS_CLOCK is sourced by a 10 MHz clock. The initialization interface is active and executing a context save or reload command issued by the CONTEXT_CNTRL state machine. The active-high PLL_LOCK input is an output of the PLL_CLK100_10 core and is used to keep the context-control state machine in an IDLE state until PLL_LOCK becomes active. The PLL_LOCK signal indicates when the PLL's internal VCO circuit is locked to its configured frequency and is in phase with its clock source.

The 100 MHz clock domain is used to sample the UPDATE_SRAM_FROM_FLASH and SAVE_SRAM_TO_FLASH commands issued by the application's main logic operating at 100 MHz. Once either command is detected, the state machine transitions to the appropriate state to begin execution of the context command. If an UPDATE_SRAM_FROM_FLASH command is detected, the state machine transitions to the UPDATE state. If a SAVE_SRAM_TO_FLASH command is detected, the state machine transitions to the SAVE state.

Before issuing the context instruction to the Flash core, SYS_CLOCK must be switched from the 100 MHz to the 10 MHz clock domain by transitioning the NGMUX_SEL output from low to high. However, the NGMUX takes two clock cycles before settling to a new frequency. Therefore, the NGMUX_CYCLE_DLY signal is used to address the two-cycle settling time. It simply makes a one-cycle delayed duplicate of the NGMUX_SEL signal. The NGMUX_CYCLE_DLY signal is then used by the state machine to wait an extra cycle before transitioning states. If transitioning from the IDLE to UPDATE or SAVE states, a high level on the NGMUX_CYCLE_DLY signal enables the transition. If transitioning from one of the command control states to the IDLE state, a low level on the NGMUX_CYCLE_DLY signal enables the transition.

The INIT_POWER_UP signal activates the initialization interface context reload transaction. The INIT_POWER_UP output signal is cleared by the FLASH_RESETn signal, and is held low until the NGMUX_CYCLE_DLY transitions from low to high, switching SYS_CLOCK to the 10 MHz clock source. Once NGMUX_CYCLE_DLY transitions and while INIT_DONE is low, the INIT_POWER_UP signal is held high. Finally, when INIT_DONE transitions high, INIT_POWER_UP then transitions low and the state machine switches the clock domain back to the 100 MHz clock source.

The on-demand context reload feature of the CONTEXT_CNTRL module is enabled by the FLASH_RESETn signal. Essentially, it is the SYS_RESETn signal ORed with the state machine control from the UPDATE state. When the state machine detects the context reload request and transitions to the UPDATE state, the FLASH_RESETn signal is pulsed low for a cycle, resetting INIT_DONE in the Flash core and triggering a new INIT_POWER_UP cycle. However, if an on-demand context update request is issued, the INIT_POWER_UP sequence updates all memory blocks including the peripheral configuration registers. The INIT_WEN_MASK signal is then used to mask the INIT_*_WEN signals to prevent unwanted updates. INIT_WEN_MASK is an active low output of the state machine and should be externally ANDed with the INIT_*_WEN signals to the ANALOG_RTC core.

The IDLE state accommodates the completion for both the initial power-up and on-demand context reload operations. The state machine will remain in IDLE until the INIT_DONE signal is high and a new context command has been issued. The CONTEXT_DONE output signal transitions high and remains until a new command has been detected by the state machine.

The context save operation only executes when a SAVE_SRAM_TO_FLASH command is issued. Once detected, the state machine transitions from the IDLE to the SAVE state. The NGMUX_SEL is then set to '1' and the CONTEXT_DONE signal is cleared. Once the NGMUX_CYCLE_DLY signal becomes high, the CLIENT_UPDATE is set to '1' and the state machine is transitioned from the SAVE to the WAIT_SAVE_COMPLETE state. The context save operation begins execution and its completion is detected when the SAVE_COMPLETE signal pulses high.

The WAIT_SAVE_COMPLETE state places the state machine in a holding state, keeping all signals stable until the SAVE_COMPLETE input goes high and CLIENT_UPDATE falls low. To support the auto shutdown of the voltage regulator after a context save feature, the VRPU_AUTO_SHDN output signal is pulsed high when the SAVE_COMPLETE is high. Once the SAVE_COMPLETE signal falls low, the NGMUX_SEL signal is cleared at the state machine and will not transition to the IDLE state until the NGMUX_CYCLE_DLY signal falls low.

Design Verification

The controls signals that are typically driven by the application's main logic block are brought out through the PM_CONTEXT_SAVING_TOP module to be driven by the testbench (TEST_PM_CONTEXT_SAVING). The testbench performs the following operations:

1. Powers the device and issues a SYS_RESETn to begin the first context reload (update) operation.
2. SRAM is then filled with incremental data values starting at 0xA500 and verified.
3. A SAVE_SRAM_TO_FLASH command is issued.
4. Once complete, an on-demand context reload command, followed by both save and reload commands, is issued.
5. The content of the SRAM is then verified.
6. A shutdown command via the VRPU_IN input port is executed.
7. Once the device is powered up by the RTC, the auto shutdown feature is enabled.
8. The content of the SRAM is verified once again.
9. Another SAVE_SRAM_TO_FLASH command is issued.
10. The device is then auto powered down and up.
11. Once re-powered, yet another content verification is performed to the SRAM.

The testbench generates the read verification results in the text window of Model*Sim*®.

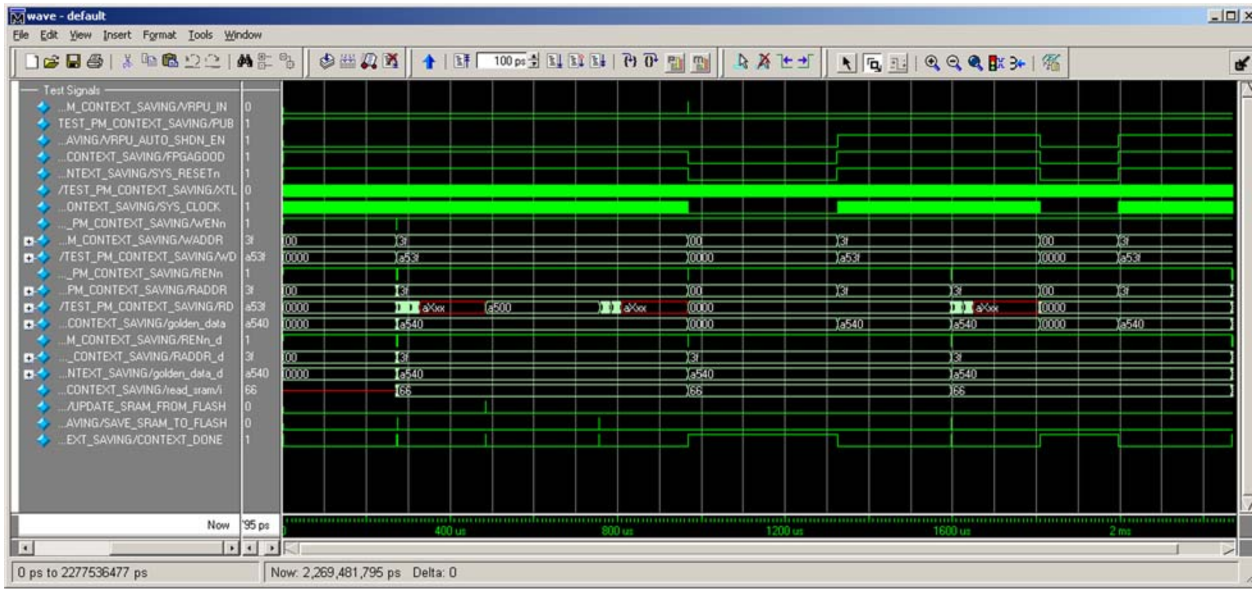

wave - default
File Edit View Insert Format Tools Window
100 ps
Test Signals
...M_CONTEXT_SAVING/VRPU_IN 0
TEST_PM_CONTEXT_SAVING/PUB 1
...AVING/VRPU_AUTO_SHDN_EN 1
...CONTEXT_SAVING/FPGAGOOD 1
...NTEXT_SAVING/SYS_RESETn 1
/TEST_PM_CONTEXT_SAVING/XTL 0
...ONTEXT_SAVING/SYS_CLOCK 1
..._PM_CONTEXT_SAVING/WENn 1
...M_CONTEXT_SAVING/WADDR 3f
/TEST_PM_CONTEXT_SAVING/WD a53f
..._PM_CONTEXT_SAVING/RENn 1
...PM_CONTEXT_SAVING/RADDR 3f
/TEST_PM_CONTEXT_SAVING/RD a53f
...CONTEXT_SAVING/golden_data a540
...M_CONTEXT_SAVING/RENn_d 1
..._CONTEXT_SAVING/RADDR_d 3f
...NTEXT_SAVING/golden_data_d a540
...CONTEXT_SAVING/read_sram/i 66
.../UPDATE_SRAM_FROM_FLASH 0
...AVING/SAVE_SRAM_TO_FLASH 0
...EXT_SAVING/CONTEXT_DONE 1
Now '95 ps
400 us
800 us
1200 us
1600 us
2 ms
0 ps to 2277536477 ps
Now: 2,269,481,795 ps Delta: 0

Figure 16 • **Complete Testbench Simulation Results**

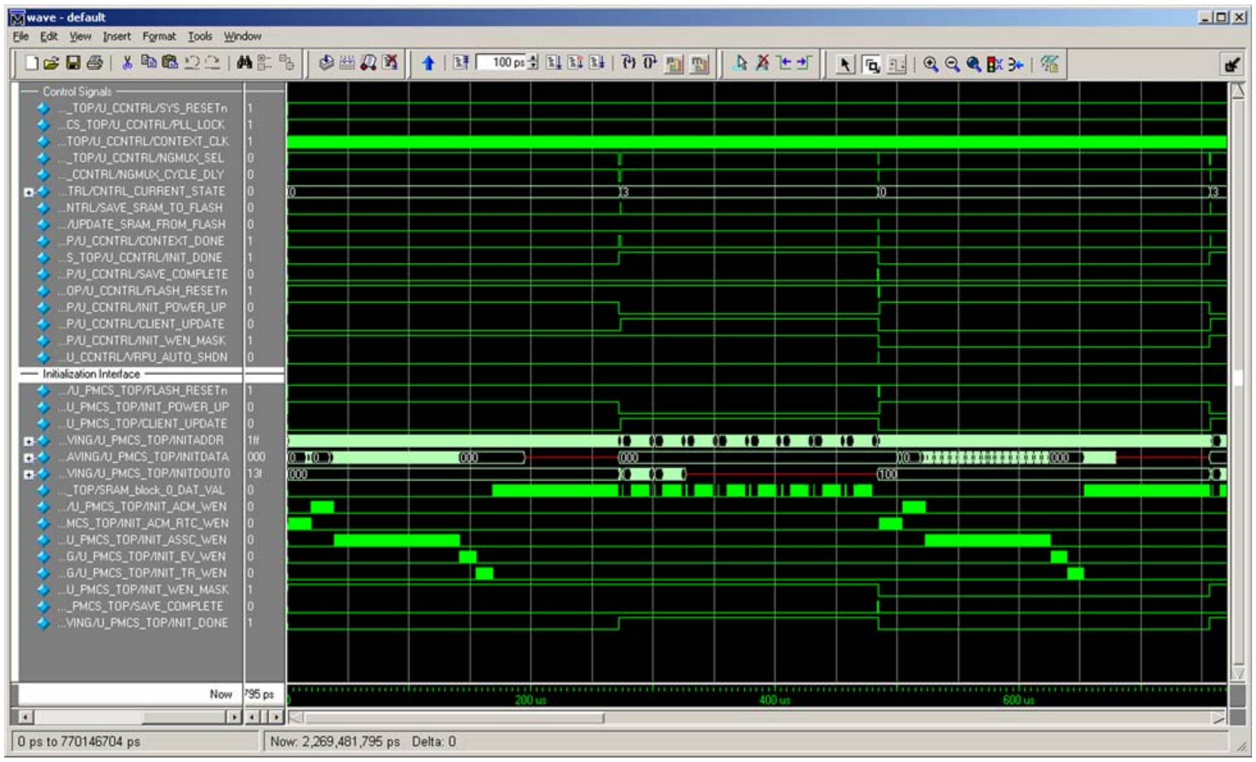

wave - default
File Edit View Insert Format Tools Window
100 ps
Control Signals
..._TOP/U_CCNTRL/SYS_RESETn 1
...CS_TOP/U_CCNTRL/PLL_LOCK 1
...TOP/U_CCNTRL/CONTEXT_CLK 1
..._TOP/U_CCNTRL/NGMUX_SEL 0
..._CCNTRL/NGMUX_CYCLE_DLY 0
...TRL/CNTRL_CURRENT_STATE 0
...NTRL/SAVE_SRAM_TO_FLASH 0
.../UPDATE_SRAM_FROM_FLASH 0
...P/U_CCNTRL/CONTEXT_DONE 1
...S_TOP/U_CCNTRL/INIT_DONE 1
...P/U_CCNTRL/SAVE_COMPLETE 0
...OP/U_CCNTRL/FLASH_RESETn 1
...P/U_CCNTRL/INIT_POWER_UP 0
...P/U_CCNTRL/CLIENT_UPDATE 0
...P/U_CCNTRL/INIT_WEN_MASK 1
...U_CCNTRL/VRPU_AUTO_SHDN 0
Initialization Interface
.../U_PMCS_TOP/FLASH_RESETn 1
...U_PMCS_TOP/INIT_POWER_UP 0
...U_PMCS_TOP/CLIENT_UPDATE 0
...VING/U_PMCS_TOP/INITADDR 1ff
...AVING/U_PMCS_TOP/INITDATA 000
...VING/U_PMCS_TOP/INITDOUT0 13f
..._TOP/SRAM_block_0_DAT_VAL 0
.../U_PMCS_TOP/INIT_ACM_WEN 0
...MCS_TOP/INIT_ACM_RTC_WEN 0
...U_PMCS_TOP/INIT_ASSC_WEN 0
...G/U_PMCS_TOP/INIT_EV_WEN 0
...G/U_PMCS_TOP/INIT_TR_WEN 0
...U_PMCS_TOP/INIT_WEN_MASK 1
..._PMCS_TOP/SAVE_COMPLETE 0
...VING/U_PMCS_TOP/INIT_DONE 1
Now 795 ps
200 us
400 us
600 us
0 ps to 770146704 ps
Now: 2,269,481,795 ps Delta: 0

Figure 17 • **Context Save and Reload Simulation Results**

When running the simulation, the following errors may be reported:

```
# ADCCLK frequency should be between 1MHz and 10MHz
```

```
# ** Error: C:/Libero/Model/win32acoem/../actel/vlog/src/fusion.v(41720):
$width( posedge ACMCLK:2269401795 ps, :2269406795 ps, 50 ns );
```

These errors occur when the ANALOG_RTC and Flash cores are issued a 100 MHz clock. These errors are not valid in the case of the context save and reload design example, since the initialization interface is not enabled unless the 10 MHz clock is active. The error message may be suppressed by Model*Sim* in order to view the text generated by the testbench.

Timing Analysis

The context save and reload design's timing analysis must be performed in two stages because of the use of the NGMUX. The first stage of the timing analysis should operate all circuits at 10 MHz. This will provide timing details related for all circuits involved with initialization interface. The second stage of the timing analysis should operate the 100 MHz clock domain only, with false paths used to break the 100 MHz clock paths to the initialization interface circuits (the ANALOG_RTC and FLASH cores).

Conclusion

The context saving and reload design example integrates a simple context-control state machine with the basic building blocks required in a power management system. These basic building blocks are all readily available to designers in the *Fusion, IGLOO/e, and ProASIC3/E Macro Library Guide* for easy integration in system designs implemented on Actel Fusion FPGA devices. Together with the context-control logic, the FPGA system power can now be shut down for current conservation without losing its critical data.

The overall system peripherals used in the design example were configured and their cores generated by the click of a button with the use of the Actel SmartGen system builder. Other Fusion peripherals, such as the on-chip 10/12-bit analog to digital converter (ADC), can also be added to the design example (or any Fusion design) using SmartGen. The ADC can then be configured for voltage, current, and temperature monitoring. After configuring, the designer creates an IP and adds Actel DirectCore or CompanionCore IP to the design. Finally, the FPGA logic design is linked to the peripherals, power, and input-output signals, completing the system.

Actel Fusion technology offers designers integrated management solutions with the added convenience of an FPGA fabric, making it an ideal candidate for implementing context save and reload logic in data-critical applications.

Related Documents

Application Notes

Context Save and Reload with Real-Timestamp application brief

http://www.actel.com/documents/Fusion_ContextSaving_AB.pdf

Context Save and Reload Design Example

http://www.actel.com/techdocs/appnotes/products.aspx#fusion

Datasheets

Fusion Family of Mixed-Signal Flash FPGAs datasheet

http://www.actel.com/documents/Fusion_DS.pdf

User’s Guides

Peripherals User's Guide

http://www.actel.com/documents/peripheral_ug.pdf

SmartGen Cores Reference Guide

http://www.actel.com/documents/gen_refguide_ug.pdf

Fusion, IGLOO/e, and ProASIC3/E Macro Library Guide

http://www.actel.com/documents/pa3_libguide_ug.pdf

Fusion Starter Kit User’s Guide

http://www.actel.com/documents/Fusion_StartKit_UG.pdf

http://www.actel.com/documents/AS_PwrM_TemM_RTC.zip

Fusion Design Flow Tutorial

http://www.actel.com/documents/fusion_df_ug.pdf

http://www.actel.com/documents/FusionDesignFlow_DesignFiles.zip

Libero IDE User's Guide

http://www.actel.com/documents/libero_ug.pdf

Designer User's Guide

http://www.actel.com/documents/designer_ug.pdf

MultiView Navigator User's Guide—includes NetlistViewer, PinEditor, I/O Attribute Editor, ChipPlanner

http://www.actel.com/documents/mvn_ug.pdf

SmartTime User’s Guide

http://www.actel.com/documents/smarttime_ug.pdf

www.actel.com

Actel Corporation	Actel Europe Ltd.	Actel Japan	Actel Hong Kong
		www.jp.actel.com	www.actel.com.cn
2061 Stierlin Court Mountain View, CA 94043-4655 USA	Dunlop House, Riverside Way Camberley, Surrey GU15 3YL United Kingdom	EXOS Ebisu Bldg. 4F 1-24-14 Ebisu Shibuya-ku Tokyo 150 Japan	Suite 2114, Two Pacific Place 88 Queensway, Admiralty Hong Kong
Phone 650.318.4200 **Fax** 650.318.4600	**Phone** +44 (0) 1276 401 450 **Fax** +44 (0) 1276 401 490	**Phone** +81.03.3445.7671 **Fax** +81.03.3445.7668	**Phone** +852 2185 6460 **Fax** +852 2185 6488

51900144-0/6.06