Rate This Document

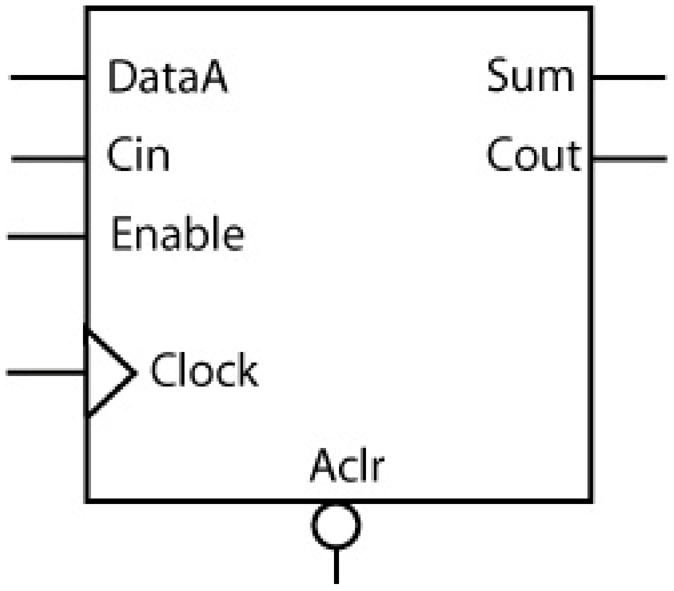
SmartGen Cores Reference Guide

Trademarks

Table of Contents

Basic Blocks

Accumulator



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, SX-S, SX-A, and eX

Related Topics

Accumulator Functionality

Accumulator I/O Description

Accumulator Parameter Description

Accumulator Implementation Rules

Key Features

- Parameterized word length
- Optional carry-in and carry-out signals
- Asynchronous reset
- Accumulator enable
- Multiple gate-level implementations (speed/area tradeoffs)
- Behavioral simulation mode in VHDL and Verilog

Accumulator Functionality

Table 1 · Functional Description

DataA	Sum_{n+1}	Cout[A]
m[width-1 : 0]	(m + Sum_n + Cin)[width-1 : 0]	(m + Sum_n + Cin)[width]

A. Cin and Cout are assumed to be active high

Accumulator I/O Description

Table 2 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
Cin	1	Input	Opt.	Carry-in
Sum	WIDTH	Output	Req.	Sum
Cout	1	Output	Opt.	Carry-out
Enable	1	Input	Opt	Enable
Clock	1	Input	Req.	Clock
Aclr	1	Input	Opt	Asynchronous Reset

Accumulator Parameter Description

Table 3 · Parameter Description

Parameter	Family	Value	Function
WIDTH[A]	ProASIC^{PLUS}	2-128	Word length of DataA, DataB and Sum
	Axcelerator	2-156	
	All others	2-32	
MAXFANOUT	ProASIC^{PLUS}	0	Automatic choice

Parameter	Family	Value	Function
		2-16	Manual setting of Max. Fanout
CI_POLARITY	ALL	0 1 **2**	Carry-in polarity (active low, active high, and not used)
CO_POLARITY	ALL	0 1 2	Carry-out polarity (active low, active high, and not used)
CLR_POLARITY		0 1 **2**	Asynchronous reset (active high, active low, and not used)
EN_POLARITY		0 1 **2**	Asynchronous enable (active high, active low, and not used)
FFTYPE	ALL except Flash	**REGULAR** TMR[B]	FF type used (Regular, Triple Voting)
CLK_EDGE		**RISE** FALL	

A. The Brent-Kung Accumulator extends the ranges from 32 to 128 bit for SX, SX-A.

B. TMR is Triple Module Redundancy. Choosing this option enables TMR Flip-Flops that are used to avoid Single Event Upsets (SEUs) for Rad-hard Designs. Choosing this option causes the Sequential resource usage to be tripled in families where no TMR is implemented in silicon.

Accumulator Implementation Rules

Table 4 · Implementation Rules

Parameter	Family	Value	Description
LPMTYPE	All	LPM_ADD_SUB	Adder category
LPM_HINT	ProASICPLUS, ProASIC	SKACC	Sklansky model
		FBKACC	Fast Brent-Kung model
		BKACC	Compact Brent-Kung model
	ALL	MFACC[A]	Very fast carry select model

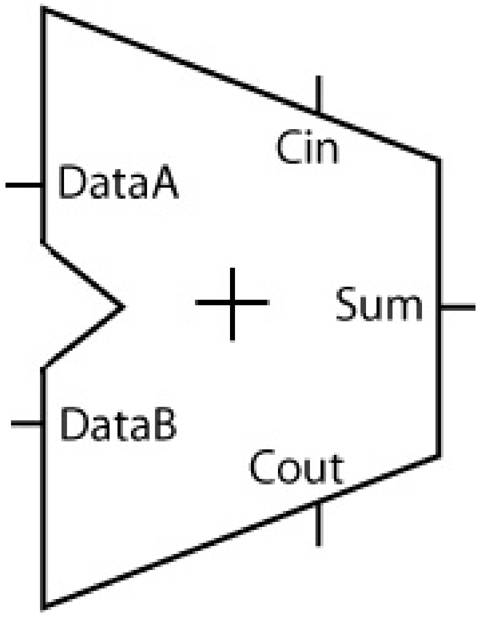
Parameter	Family	Value	Description
	ALL	RIPACC[A]	Ripple carry model
LPMTYPE	Axcelerator	LPM_FC_ADD_SUB	Fast carry chain adder category
LPM_HINT		FC_FACC	Fast carry chain select model
LPM_HINT		FC_RIPACC	Fast carry chain ripple carry model

A. FACC and MFACC are not recommended for ProASIC^{PLUS} devices.

Table 5 · Fan-In Control Parameters

Parameter	Value
CLR_FANIN	**AUTO** MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	**AUTO** MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	**AUTO** MANUAL
CLK_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]

Adder



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, SX-S, SX-A, and eX

Related Topics

Adder Functionality

Adder I/O Description

Adder Parameter Description

Adder Implementation Rules

Key Features

- Parameterized word length
- Optional carry-in and carry-out signals
- Multiple gate-level implementations (speed/area tradeoffs)
- Behavioral simulation RTL in VHDL and Verilog

For the Sklansky Adder, you can clear the Automatic Max. Fanout check box and specify a value for max fanout. This makes the software perform logic replication on high-fanout nets so that the maximum fanout for all the nets in the design is not more than the value specified. If it is set to automatic, the software automatically makes the decision for logic replication based on the size of the design.

Adder Functionality

Table 6 · Functional Description

DataA	DataB	Sum	Cout[A]
m[width-1 : 0]	n[width-1 : 0]	(m + n + Cin)[width-1 : 0]	(m + n + Cin)[width]

A. Cin and Cout are assumed to be active high

The Sklansky Adder enables you to clear the Automatic Max. Fanout check box and specify a value for max fanout. This makes the software perform logic replication on high-fanout nets so that the maximum fanout for all the nets in the design is not more than the value specified. If it is set to automatic, the software automatically makes the decision for logic replication based on the size of the design.

The MAXFANOUT parameter enables you to perform logic replication for all Flash Adders, Subtractors, Adder/Subtractors and Accumulators. Inherently only the Sklansky algorithm generates high-fanout nets (max. fanout = WIDTH/2), so you will see effects only for this algorithm. The area increases exponentially for MAXFANOUT approaching 2 and it flattens out for higher values, as shown in the figure below.

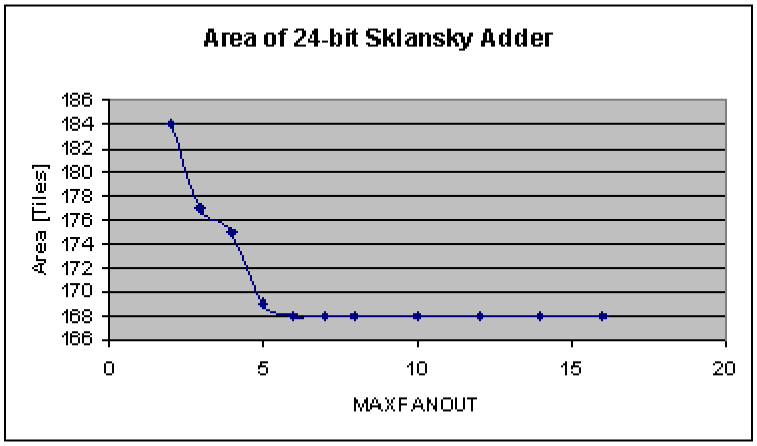


Figure 1 · Adder Area as a Function of MAXFANOUT

Performance is not always as predictable (as shown in the figure below). When you select automatic logic replication, the software automatically chooses a value for MAXFANOUT based on WIDTH. This value returns a good, but not necessarily the best, result for that particular value of WIDTH.

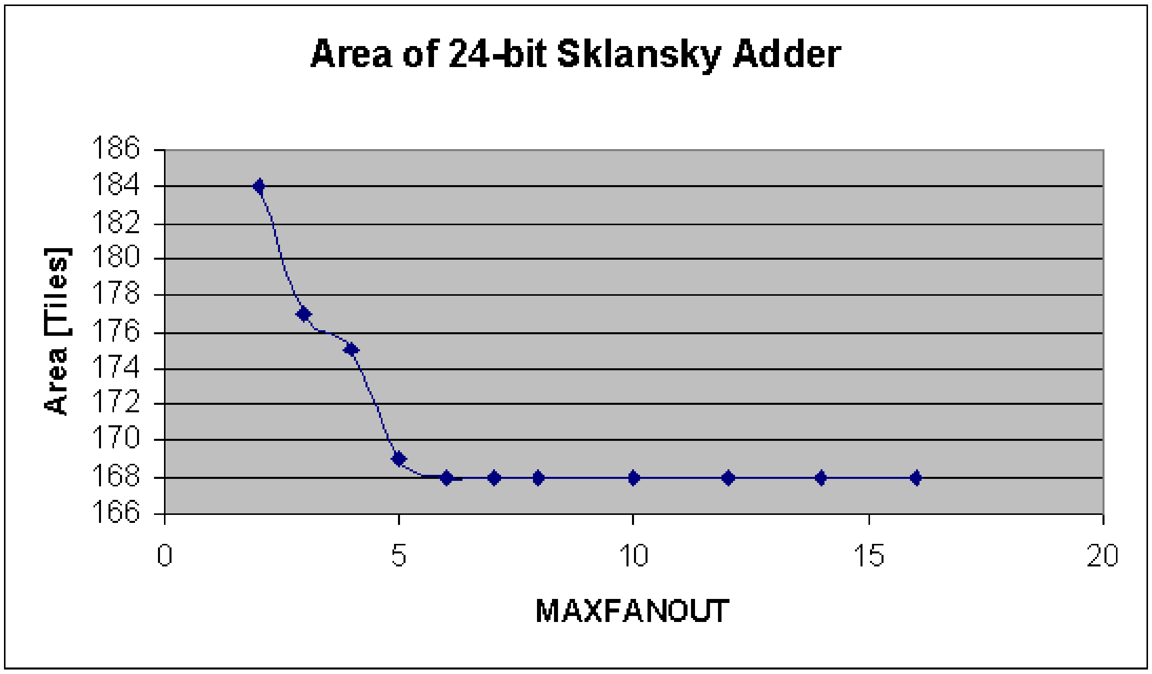


Figure 2 · Adder Performance as a Function of MAXFANOUT

Adder I/O Description

Table 7 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
DataB	WIDTH	Input	Req.	Input Data
Cin	1	Input	Opt.	Carry-in
Sum	WIDTH	Output	Req.	Sum
Cout	1	Output	Opt.	Carry-out

Adder Parameter Description

Table 8 · Parameter Description

Parameter	Family	Value	Description

Parameter	Family	Value	Description
WIDTH[A]	ProASIC^{PLUS}	2-128	Word length of DataA, DataB and Sum
	Axcelerator	2-156	
	All others	2-32	
MAXFANOUT	500K, PA	0	Automatic choice
		2-16	Manual setting of Max. Fanout
CI_POLARITY	ALL	0 1 **2**	Carry-in polarity (active low, active high, and not used)
CO_POLARITY	ALL	0 1 **2**	Carry-out polarity (active low, active high, and not used)

A. The Brent-Kung Adder extends the ranges from 32 to 128 bit for SX, SX-A and from 20 to 128 bit for 500K

Adder Implementation Rules

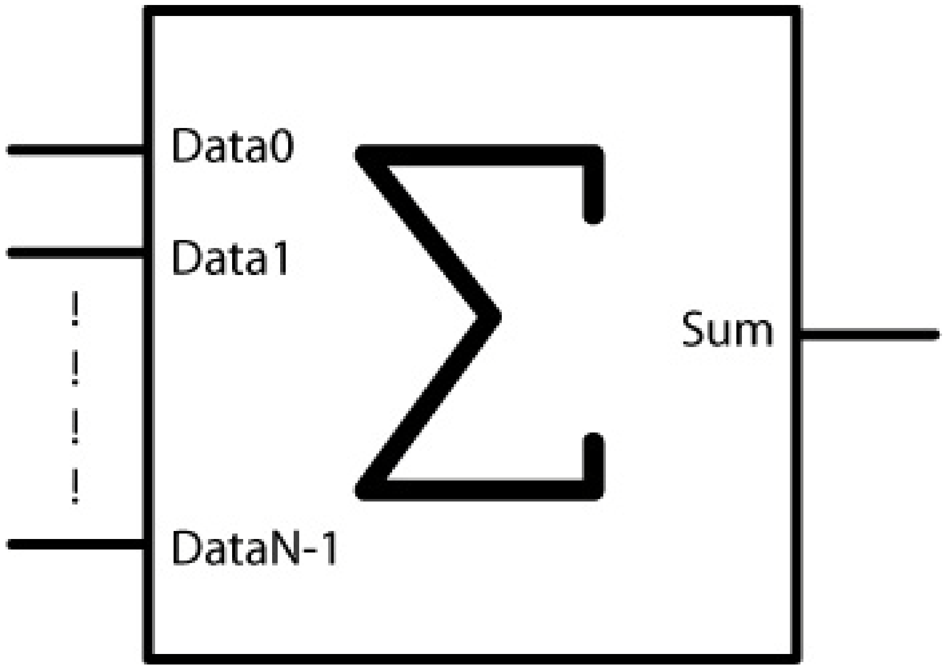
Table 9 · Implementation Rules

Parameter	Family	Value	Description
LPMTYPE	All	LPM_ADD_SUB	Adder category
LPM_HINT	ProASICPLUS, ProASIC	SKADD	Sklansky model
		FBKADD	Fast Brent-Kung model
		BKADD	Compact Brent-Kung model
	ALL	FADD[A]	Very fast carry select model
	ALL	RIPADD[A]	Ripple carry model
	ALL	MFADD[A]	Fast carry select model
LPMTYPE	Axcelerator	LPM_FC_ADD_SUB	Fast carry chain adder category

Parameter	Family	Value	Description
LPM_HINT		FC_FADD	Fast carry chain select model
LPM_HINT		FC_RIPADD	Fast carry chain ripple carry model

A. FADD and MFADD are not recommended for ProASIC3 or ProASIC^{PLUS} devices.

Array Adder



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, SX-S, SX-A, and eX

Related Topics

Array Adder I/O Description

Array Adder Parameter Description

Array Adder Implementation Rules

Key Features

- Parameterized word length and number of input buses

- DADDA tree architecture with optional Final Adder
- Optional pipeline for implementation with Final Adder
- Behavioral simulation RTL in VHDL and Verilog

The Array-Adder implements a Sum-Function over an array of buses:

Sum = [Summation(Data(i))] where i = 0 to Size-1

In applications where designers have to add more than two operands at a time "Carry-Save- Techniques" might be used to build the final Sum. The software makes these techniques available through the Array-Adder core, which is using a DADDA tree algorithm. Usually this algorithm is more compact and faster than using Adder trees consisting of multiple 2-operand adders, especially if the number of operands gets large and/or for large word width.

An example could be the FIR-filter architecture using a "distributed arithmetic" as described in the Application Note from September 1997 Designing FIR Filters with Actel FPGAs. This architecture generates a large number of partial products, which need to be summed. Summing them in an Adder-Tree would both be slow and area-expensive. At the time of writing this document, synthesis tools did not infer Multiple-Operand-Adders. Therefore making use of the Array-Adder in those types of applications might result in a significant gain in both speed and area.

The Array Adder comes with or without Final Adder. The version with Final Adder allows the software to instantiate a pipeline stage between the Dadda-tree and the Final Adder. The output bitwidth for Sum can be calculated using this formula:

OUTWIDTH = log2((m*exp2(n)-1)+1) <= n + log2(m)

The version without Final Adder has two output ports: SumA and SumB, which added together will provide the Final Result. It is

SumA_Width <= SumB_Width <= OUTWIDTH

The differences are at most one bit. This variation of the Array-Adder is particularly useful for an application that would cascade the Array-Adder. In that case only the last stage would need a Final Adder to build the result.

Array Adder I/O Description

Table 10 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Data0	WIDTH	Input	Req.	Input Data (Operand 0)
Data1	WIDTH	Input	Req.	Input Data (Operand 1)
Data2	WIDTH	Input	Req.	Input Data (Operand 2)
Datax	WIDTH	Input	Opt.	Input Data (Operand X) X>2
Sum	OUT-WIDTH	Output	Req.	Sum(Data(i) to i) = 0 to SIZE-1

Port Name	Size	Type	Req/Opt	Function
Clock	1	Input	Opt.	Active High/Low Clock (if pipelined)

Array Adder Parameter Description

Table 11 · Parameter Description

Parameter	Value		Description
WIDTH	width	AX/Flash: 2-64 All others: 2-32	Word length Data(i)
SIZE	size	AX/Flash: 3-64 All others: 3-32	Number of input buses
CKL_EDGE	RISE FALL		Clock (if pipelined)

Table 12 · Parameter Rules

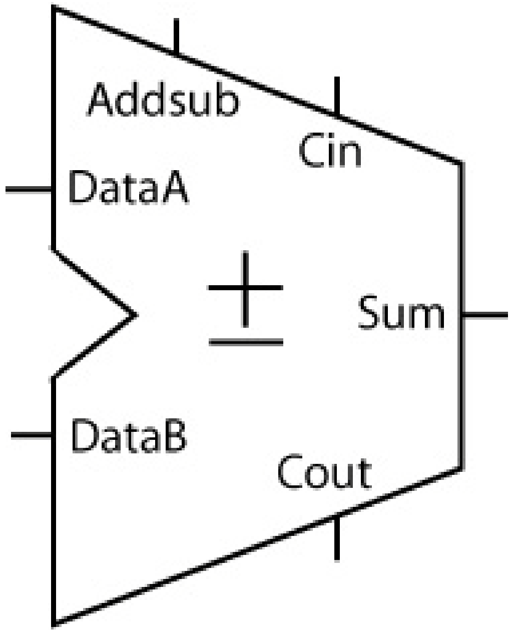
Family	Variation	Parameter Rules
eX	ARRADD / ARRADDP	WIDTH * SIZE <=870
	ARRADD2	WIDTH * SIZE <= 930
SX	ARRADD / ARADDP	WIDTH * SIZE <=110
	ARRADD2	WIDTH * SIZE <=144
Axcelerator	ARRADD/ARRADDP	WIDTH * SIZE <=1920
	ARRADD2	WIDTH * SIZE <=1856

Array Adder Implementation Rules

Table 13 · Implementation Rules

Parameter	Value	Description
LPMTYPE	DADDA	Generic Array-Adder category
LPM_HINT	ARRADD	Array-Adder with Final Adder
	ARRADDP	Pipelined Array-Adder with Final Adder
	ARRADD2	Array-Adder without Final Adder

Adder/Subtractor



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, SX-S, SX-A, and eX

Related Topics

Adder/Subtractor Functionality

Adder/Subtractor I/O Description

Adder/Subtractor Parameter Description

Adder/Subtractor Implementation Rules

Key Features

- Parameterized word length
- Optional carry-in and carry-out signals
- Multiple gate level-implementations (speed/area tradeoffs)
- Behavioral simulation RTL in VHDL and Verilog

Adder/Subtractor Functionality

Table 14 · Functional Description

DataA	DataB	Addsub	Sum	Cout[A]
m[width-1 : 0]	n[width-1 : 0]	(m + n + Cin)[width-1 : 0]	(m + n + Cin)[width]	m[width-1 : 0]
m[width-1 : 0]	n[width-1 : 0]	(m - n - Cin) [width-1 : 0]	(m - n - Cin)[width]	m[width-1 : 0]

A. Cin and Cout are assumed to be active high

Adder/Subtractor I/O Description

Table 15 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
DataB	WIDTH	Input	Req.	Input Data
Cin	1	Input	Opt.	Carry-in
Sum	WIDTH	Output	Req.	Sum
Cout	1	Output	Opt.	Carry-out
Addsub	1	Input	Req.	Addition (AddSub=1) or subtraction (AddSub=0)

Adder/Subtractor Parameter Description

Table 16 · Parameter Description

Parameter	Family	Value	Function
WIDTH	ProASIC$^{\underline{PLUS}}$	2-128	Word length of DataA, DataB and Sum
	Axcelerator	2-156	
	All others	2-32	
MAXFANOUT	ProASIC$^{\underline{PLUS}}$	0	Automatic choice
		2-16	Manual setting of Max. Fanout
CI_POLARITY	ALL	0 1 **2**	Carry-in polarity (active low, active high, and not used)
CO_POLARITY	ALL	0 1 **2**	Carry-out polarity (active low, active high, and not used)

A. The Brent-Kung Adder/Subtractor extends the ranges from 32 to 128 bit for SX, SX-A and from 20 to 128 bit for ProASIC$^{\underline{PLUS}}$.

Adder/Subtractor Implementation Rules

Table 17 · Implementation Rules

Parameter	Family	Value	Description
LPMTYPE	All	LPM_ADD_SUB	Adder category
LPM_HINT	ProASICPLUS, ProASIC	SKADDSUB	Sklansky model
		FBKADDSUB	Fast Brent-Kung model
		BKADDSUB	Compact Brent-Kung model
	ALL	FADDSUB[A]	Very fast carry select model
	ALL	RIPADDSUB[A]	Ripple carry model
LPMTYPE	Axcelerator	LPM_FC_ADD_SUB	Fast carry chain Adder

Parameter	Family	Value	Description
			category
LPM_HINT		FC_ADDSUB	Fast carry chain select model
LPM_HINT		FC_RIPADDSUB	Fast carry chain ripple carry model

A. FADDSUB and MFADDSUB are not recommended for ProASIC3 or ProASIC^{PLUS} devices.

Constant Decoder

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, MX, eX, and SX/SX-A

Related Topics

Constant Decoder Functionality

Constant Decoder I/O Description

Constant Decoder Parameter Description

Constant Decoder Implementation Rules

Key Features

- Parameterized word length
- DEC/BIN/HEX radices for constant
- Equal/Not Equal comparison

Constant Decoder Functionality

Table 18 · Functional Description

Aeb
DataA = Constant

Constant Decoder I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
Aeb	1	Output	Req.	Result

Constant Decoder Parameter Description

Table 19 · Parameter Description

Parameter	Value	Function
WIDTH	2-32[A]	Word length of DataA and Constant
Radix	Dec/Bin/Hex	Base of Constant
Constant	Same as Width in selected Radix	The value with which input data will be compared
AEB_POLARITY	0, 1	A equals B polarity (Active High, Active Low)

A. For ProASIC, width is 2-128

Parameter Rules:

1. DataA is always binary and of the size of Width.
2. Constant must be of the selected Radix and be of the selected width for HEX/BIN.

e.g.: Radix: BIN, Width: 5, Constant: 00010

Radix Hex, Width:8, Constant: 0A

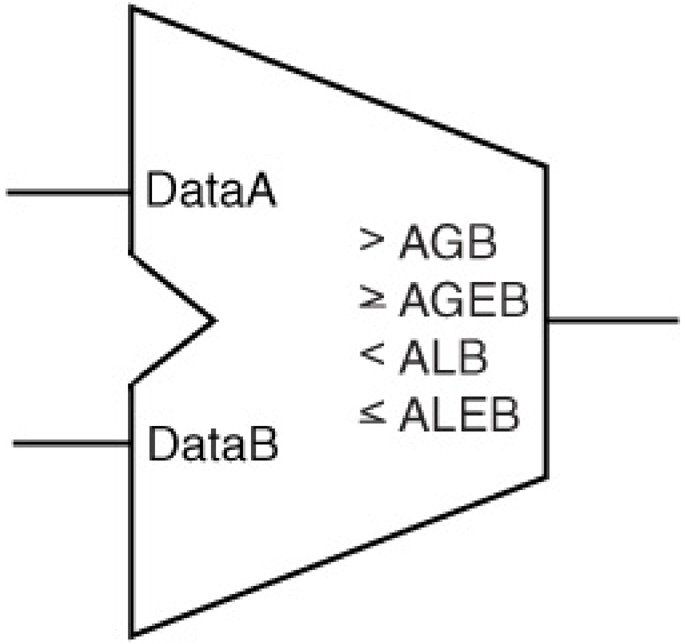
Constant Decoder Implementation Rules

Table 20 · Implementation Rules

Parameter	Value	Description
LPM_TYPE	LPM_COMPARE	Comparator category

Parameter	Value	Description
LPM_HINT	WDEC	Very fast

Magnitude/Equality Comparator



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Magnitude / Equality Comparator I/O Description

Magnitude / Equality Comparator Parameter Description

Magnitude / Equality Comparator Implementation Rules

Key Features

- Parameterized word length
- Unsigned and signed (Two's-Complement) data comparison
- One very fast gate level implementation
- Behavioral simulation RTL in VHDL and Verilog

Magnitude / Equality Comparator I/O Description

Table 21 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input data
DataB	WIDTH	Input	Req.	Input data
AGB	1	Output	Opt.	Output comparison; A greater than B
AGEB	1	Output	Opt.	Output comparison; A greater than or equal to B
ALB	1	Output	Opt.	Output comparison; A less than B
ALEB	1	Output	Opt.	Output comparison; A less than or equal to B
AEB	1	Output	Opt.	Output comparison; A equal to B
ANEB	1	Output	Opt.	Output comparison; A not equal to B

Magnitude / Equality Comparator Parameter Description

Table 22 · Parameter Description

Parameter	Value	Function
WIDTH	2-32	Word length of DataA and DataB
REPRESENTATION	**UNSIGNED** SIGNED	
AGB_POLARITY	0 1 **2**	AGB polarity (active high, active low, and not used)
AGEB_POLARITY	0 1 **2**	AGEB polarity (active high, active low, and not used)
ALB_POLARITY	0 1 **2**	ALB polarity (active high, active low, and not used)

Parameter	Value	Function
ALEB_POLARITY	0 1 **2**	ALEB polarity (active high, active low, and not used)
AEB_POLARITY	0 1 **2**	AEB polarity (active high, active low, and not used)
ANEB_POLARITY	0 1 **2**	ANEB polarity (active high, active low, and not used)

Magnitude / Equality Comparator Implementation Rules

Table 23 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_COMPARE	Comparator category
	LPM_FC_COMPARE	Fast Comparator category
LPM_HINT	COMPARE	Very fast carry select
	FC_MAGCOMP	Very fast magnitude comparator

Parameter Rules:

1. At lease one of the comparisons (AGB, AGEB, ALB, ALEB, AEB or ANEB) must be selected
2. Only one of the magnitude comparisons (AGB, AGEB, ALB or ALEB) can be selected at the same time
3. Only one of the equality comparisons (AEB or ANEB) can be selected at the same time

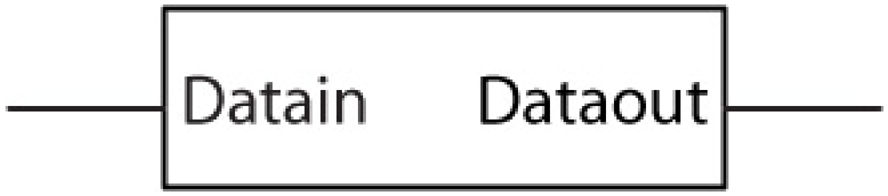
LPM_HINT has additional Implementation Parameters, as shown in the table below.

Implementation (LPM_HINT)	Description
COMPARE	Very fast carry select model
FC_MAGCOMP	Fast carry Magnitude Comparator

Table 24 · Functional Description

DataA	DataB	AGB	AGEB	ALB	ALEB	AEB	ANEB
m	n	m > n	m greater than or equal to n	m < n	m less than or equal to n	m = n	m not equal to n

Binary to Gray / Gray to Binary Converters



You can generate Binary to Gray and Gray to Binary Converters parameterized for a specified Data Width.

Supported Families

SX, Axcelerator

Related Topics

Binary to Gray / Gray to Binary Converter I/O Description

Binary to Gray / Gray to Binary Converter Parameter Description

Binary to Gray / Gray to Binary Converter Implementation Rules

Key Features

- Parameterized for data width

Binary to Gray / Gray to Binary Converter I/O Description

Figure 3 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Datain	WIDTH	Input	Req.	Input Data
Dataout	WIDTH	Output	Req.	Output Data

Binary to Gray / Gray to Binary Converter Parameter Description

Table 25 · Parameter Description

Parameter	Value	Function
GRAYDECODE/WIDTH	2-99	Input/Output Data Width

Binary to Gray / Gray to Binary Converter Implementation Rules

Table 26 · Implementation Rules

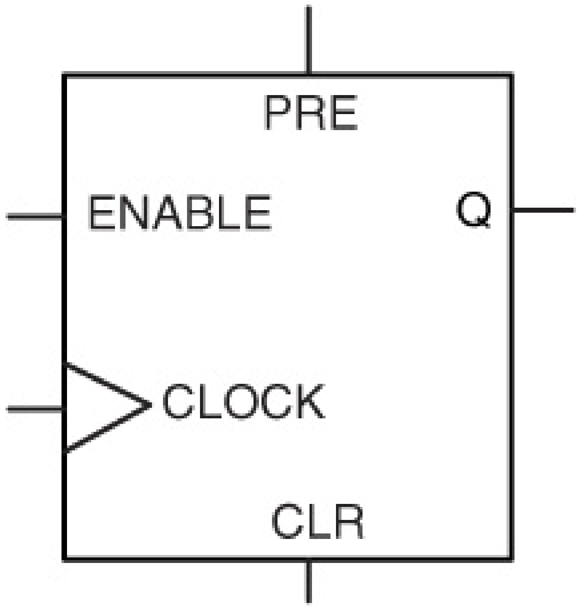
Parameter	Value	Function
LPMTYPE	LPM_GRAYENCODE/ LPMGRAYDECODE	Binary to Gray and Gray to Binary Converter

Table 27 · Functional Description[A]

Data	Aclr	Enable	Sload	Clock	Up down	Qn+1	Tcnt n+1
X	0	X	X	X	X	0's	0
X	1	X	X	*f*	X	Qn	Qn+1== 2^{width}-1
X	1	0	0	*f*	X	Qn	Qn+1== 2^{width}-1
m	1	X	1	*f*	X	m	Qn+1== 2^{width}-1
X	1	1	0	*f*	1	Qn + 1	Qn+1== 2^{width}-1
X	1	1	0	*f*	0	Qn - 1	Qn+1== 2^{width}-1

A. Assume Aclr is active low, Enable is active high, Sload is active high, Clock is rising, and Tcnt is active high.

Gray Counter



Supported Families

SX, Axcelerator

Related Topics

Gray Counter I/O Description

Gray Counter Parameter Description

Gray Counter Implementation Rules

Key Features

- Parameterized for data width
- Asynchronous clear / Asynchronous preset

Gray Counters are available parameterized for a specified Data Width and with a choice of Enable, Asynchronous Clear, and Asynchronous Preset signals.

Gray Counter I/O Description

Table 28 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Clock	WIDTH	Input	Req.	Input Data

Port Name	Size	Type	Req/Opt	Function
Q	WIDTH	Output	Req.	Output Data
Clr	1	Input	Opt.	Clear
Pre	1	Input	Opt.	Preset
Enable	1	Input	Opt.	Enable

Gray Counter Parameter Description

Table 29 · Parameter Description

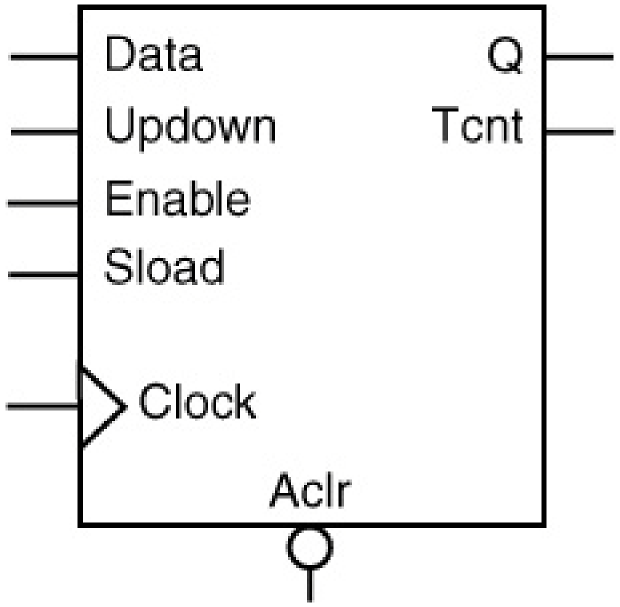
Parameter	Value	Function
GRAYCOUNT	2-99	Output Data Width
CLR_POLARITY	0,1,2	Clear Polarity
PRE_POLARITY	0,1,2	Preset Polarity
EN_POLARITY	0,1	Enable Polarity
CLK_EDGE	RISE,FALL	Clock Edge

Gray Counter Implementation Rules

Table 30 · Implementation Rules

Parameter	Value	Function
LPMTYPE	LPM_GRAY COUNTER	Gray Counter

Linear Binary Counters



Supported Families

ProASIC^{PLUS}, ProASIC, Axcelerator, MX, eX, and SX/SX-A

Note: If you are using IGLOO, ProASIC3, SmartFusion and Fusiondevices, please see the Linear Binary Counter description for those devices.

Note: The counters that appear in the Catalog depend on your family specified for your project.

Related Topics

Linear Binary Counter Functionality

Linear Binary Counter I/O Description

Linear Binary Counter Parameter Description

Linear Binary Counter Implementation Rules

Key Features

- Parameterized word length
- Up, Down, and Up/Down architectures
- Asynchronous clear
- Asynchronous preset (available only for Flash devices)
- Synchronous counter load
- Synchronous count enable

- Terminal count flag (not available for Axcelerator)
- Multiple gate-level implementations (area/speed tradeoffs)
- Behavioral simulation RTL in VHDL and Verilog

Binary counters are general purpose UP, DOWN, or UP/DOWN (direction) counters.

When the count value equals 2^{width}-1, the signal Tcnt (terminal count), if used, is asserted high.

The counters are WIDTH bits wide and have 2^{width} states from “000…0” to “111…1”. The counters are clocked on the rising (RISE) or falling (FALL) edge of the clock signal Clock (CLK_EDGE).

TheClear signal(CLR_POLARITY), active low or high, provides an asynchronous reset of the counter to “000…0”. You may choose to not implement the reset function. If you do not use the Clear signal, Actel recommends that you use Sload to set the initial counter contents to a known value.

In the case of an Up/Down counter, the Updown signal controls whether the counter counts up (Updown = 1) or down (Updown = 0).

The counter could be loaded with Data. The Sload signal (LD_POLARITY), active high or low, provides a synchronous load operation with respect to the clock signal Clock. You can choose to not implement this function. If you do not use the Sload signal, Actel recommends that you use Clear to set the initial counter contents to a known value.

The counters have a count enable signal Enable (EN_POLARITY). Enable can be active high or low. When Enable is not active, the counter is disabled and the internal state is unchanged.

Linear Binary Counter Functionality

This section decribes the implementation of the Pre-Scaled Counter, Register Look Ahead Counter, Fast Balanced Counter and the Balanced Counter.

Pre-Scaled Counter

The pre-scaled counter achieves absolute maximum count and count enable performance by sacrificing synchronous load performance. This counter registers the two least significant bits and uses them as an enable for the upper bits. Count performance is limited only by the delay in the lower two bits and the enable path for the upper bits. Because the upper bits are only updated (enabled) every fourth cycle, they can accommodate more delay (up to one-fourth the clock frequency).

There are two limitations related to the use of the pre-scaled counter. The first is in analyzing the actual performance of the counter. The second is correctly performing data load functions; these two limitations are related. Two parameters must be measured to overcome these two limitations. The first parameter that must be measured is the worst internal delay inside the counter. The second parameter is the worst delay from Q0/Q1 to any upper bit. The minimum count period is then defined by the greater value of these two parameters.

The load function is a slave of the maximum internal path delay in the pre-scaled counter. The load function must be held for as many clock periods as required to exceed the maximum internal delay; this ensures that all internal nodes are settled and that correct count operation can be performed. This requirement can be waived if you can guarantee that '0's will always be loaded in Q0 and Q1 (resulting in only a single load cycle).

The count path in pre-scaled counters without Sload or Enable functions only have a single logic level for ACT 2, ACT 3, 3200DX, MX SX, SX-A, and eX. All other combinations of pre-scaled counters have two logic levels in their count path. In these cases, given the two limitations mentioned previously related to the pre-scaled counter, use the Register Look Ahead or Fast Balanced counters.

Register Look Ahead Counter

This counter achieves the absolute maximum performance for the count, count enable, and synchronous load functions. The counter operates by registering intermediate count values providing “look-ahead” carry circuitry. As a result, this counter variation requires more flip-flops (sequential modules) than other counters.

Fast Balanced Counter

This counter is only available for the SX, SX-A, and eX families. It takes advantage of the architectural features of these families, including flip-flops with built-in enable and more powerful combinatorial cells. Using these two features, it is possible to build a very fast and compact binary counter without using “look-ahead” carry circuitry. This counter should be preferred over all the others available for this family.

Balanced Counter

This counter achieves high performance for both the count and enable functions using standard design approaches. Module count performance is sacrificed to maintain high speed. This counter is the result of the performance balance between the count/enable functions and the balance between the performance/cost in building this architecture. This counter should address most counter needs for the ACT 1, ACT 2, ACT 3, 3200DX, MX and MX families.

Pseudo Random Counter

A Pseudo Random Counter is available using a Linear Feedback Shift Register (LFSR) architecture. The LFSR offers an efficient architecture for building very fast Pseudo Random Counters.

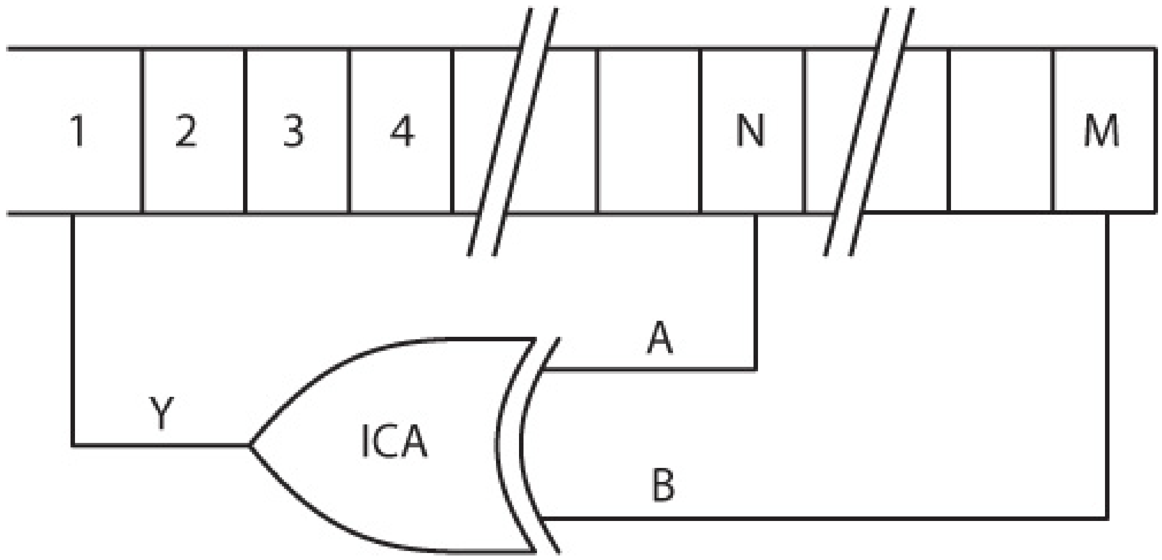


Figure 4 · Generic Random Counter Architecture

The Pseudo Random core architecture core is a simple shift register chain that uses two taps (one logic level) for the following widths: 2-7, 9-11, 15, 17, 18, 20-23, 25, 28, 29, and 31. The PRNG core uses five taps (three logic levels) for the following widths: 8, 12-14, 16, 19, 24, 26, 27, 30, and 32. The five-tap architecture operates slower than the two-tap implementation.

Fast Enable Counter

This compact counter is fully synchronous and has higher performance than the ripple counter. However, this counter should only be used in moderate performance applications, especially for large widths.

Ripple Counter

The ripple counter is an asynchronous counter where the Q of each bit feeds the clock of the next bit; performance is sacrificed to build this variation. However, the ripple counter uses the least amount of logic resources. This counter should only be used in very low-performance applications or for very small counters.

Because of the asynchronous nature of the count function, this counter does not have a synchronous load function.

Modulo Counter

As counter size increases, the amount and complexity of support logic also increases. LFSR base counters achieve high performance using very few logic resources. The Modulo Counter is designed to provide two logic levels independently of the chosen modulo value. The architecture borrows some look-ahead techniques previously used in the register look-ahead counter.

The example below is based on a modulo-6 counter with the following characteristics:

- Active-HIGH clock edge
- Active-LOW asynchronous clear

- Active-HIGH synchronous clear
- No Enable

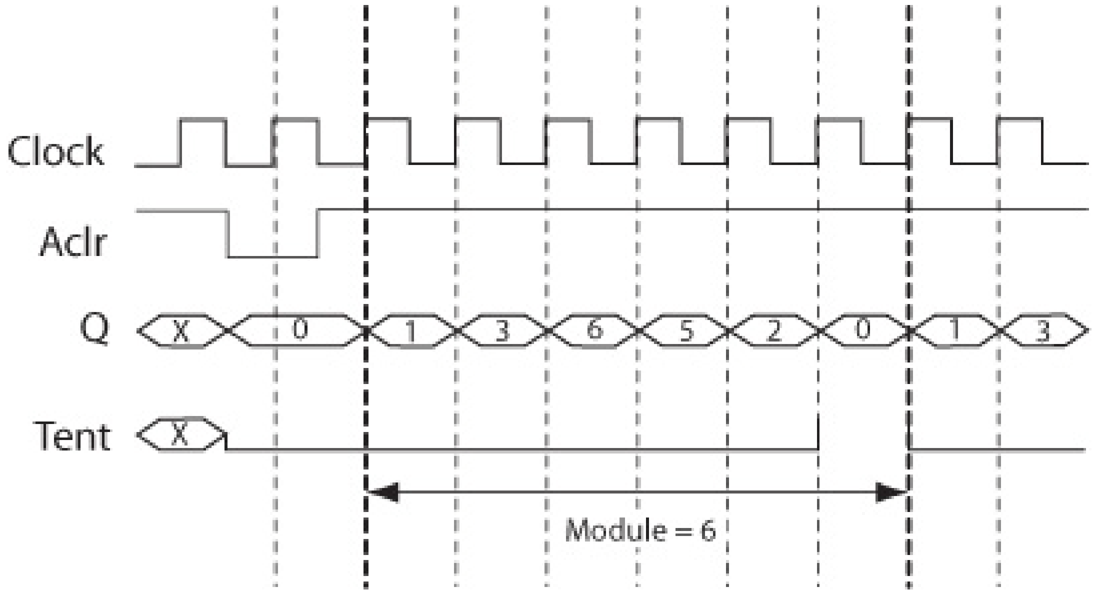


Figure 5 · Modulo Counter Behavior

Linear Binary Counter I/O Description

Table 31 · I/O Description

Port Name	Size	Type	Req./Opt.	Function
Data	WIDTH	input	Opt.	Counter load input
Aclr	1	input	Opt.	Asynchronous counter reset
Enable	1	input	Req.	Counter enable
Sload	1	input	Opt.	Synchronous counter load
Clock	1	input	Req.	Clock
Updown	1	input	Opt.	UP (Updown = 1), DOWN (Updown = 0)
Q	WIDTH	out-put	Req.	Counter output bus
Tcnt	1	out-put	Opt.	Terminal count (active high)

Linear Binary Counter Parameter Description

Table 32 · Parameter Description

Parameter	Value	Function
WIDTH	2-32	Word length of Data and Q
DIRECTION	**UP** DOWN UPDOWN	Counter direction
CLR_POLARITY	**0** 1 2	Aclr can be active low, active high, or not used
EN_POLARITY	0 **1**	Enable can be active low, or active high
LD_POLARITY	0 **1** 2	Sload can be active low, active high, or not used
CLK_EDGE	**RISE** FALL	
TCNT_POLARITY	1 **2**	Tcnt can be active high or not used

Table 33 · Fan-in Control Parameters

Parameter	Value
CLR_FANIN	**AUTO** MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
LD_FANIN	**AUTO** MANUAL
LD_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	AUTO **MANUAL**
CLK_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]

Table 34 · Functional Description[A]

Data	Aclr	Enable	Sload	Clock	Up down	Qn+1	Tcnt n+1
X	0	X	X	X	X	0's	0
X	1	X	X	¦	X	Qn	Qn+1== 2^{width}-1
X	1	0	0	¦	X	Qn	Qn+1== 2^{width}-1
m	1	X	1	¦	X	m	Qn+1== 2^{width}-1
X	1	1	0	¦	1	Qn + 1	Qn+1== 2^{width}-1
X	1	1	0	¦	0	Qn - 1	Qn+1== 2^{width}-1

A. Assume Aclr is active low, Enable is active high, Sload is active high, Clock is rising, and Tcnt is active high

Linear Binary Counter Implementation Rules

Table 35 · Implementation Rules

Parameter	Value	Description	Family
LPMTYPE	LPM_COUNTER	Counter category	
LPM_HINT	LLCNT	Prescaled model	All
	TLACNT	Register look ahead model	All
	FBCNT	Fast Balanced model	SX, SX-A
	BCNT	Balanced model	All
	LECNT	Fast Enable Balanced	All
	COMPCNT	Compact model	All
	RIPPLE	Ripple model	All

Linear Binary Counters IGLOO, ProASIC3, SmartFusion and Fusion Summary

If you are not using a IGLOO, ProASIC3, SmartFusion and Fusiondevice, please see the Linear Binary Counter description for all other devices.

Related Topics

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion: Functionality

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion I/O Description

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion Parameter Description

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion Implementation Parameters

Features

- Parameterized word length
- Up, Down, and Up/Down architectures
- Asynchronous clear
- Asynchronous preset
- Synchronous counter load
- Synchronous count enable
- Terminal count flag
- Multiple gate-level implementations (area/speed tradeoffs)
- Behavioral simulation RTL in VHDL and Verilog

The binary counters are general purpose UP, DOWN, or UP/DOWN (direction) counters.

If used, the Tcnt (terminal count) signal is asserted when the count value equals 2^{width}-1 for Up counters, or 0 for Down counters (Tcnt is not supported for Up/Down counters).

The counters are WIDTH bits wide and have 2^{width} states from “000…0” to “111…1”. The counters are clocked on the rising (RISE) or falling (FALL) edge of the clock signal *Clock* (CLK_EDGE).

The Aclr signal (CLR_POLARITY), active low or high, provides an asynchronous clear of the counter to “000…0”. You may choose to not implement the clear function. If you do not use the Aclr signal, then you must use at least one of the Aset or Sload signals to set the initial counter contents to a known value.

The Aset signal (SET_POLARITY), active low or high, provides an asynchronous preset of the counter to “111…1”. You may choose to not implement the preset function. If you do not use the Aset signal, then you must use at least one of the Aclr or Sload signals to set the initial counter contents to a known value.

If used, the Aclr/Aset signals must be made global, otherwise a 2-tile implementation of the flip-flops is used, doubling the number of SEQ (sequential) modules. An example of the pdc entry is:

```
assign_global_clock -net Aclr
```

In the case of an Up/Down counter, the *Updown* signal controls whether the counter counts up (Updown = 1) or down (Updown = 0).

The counter could be loaded with Data. The Sload signal (LD_POLARITY), active high or low, provides a synchronous load operation with respect to the clock signal Clock. You can choose to not implement this function. If you do not use the Sload signal, then you must use either Aclr or Aset to set the initial counter contents to a known value.

The counter can have an Enable signal (EN_POLARITY), active low or high. You are not required to have an Enable signal. When Enable is not active, the counter is disabled and the internal state is unchanged.

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion: Functionality

Implementations

This section decribes the implementation of the Compact Counter and Register Look Ahead Counter.

Compact Counter

A basic counter with one flip-flop (sequential module) per bit. Performance decreases as the counter WIDTH increases.

Register Look Ahead Counter

This counter achieves the absolute maximum performance for the count, count enable, and synchronous load functions. The counter operates by registering intermediate count values providing “look-ahead” carry circuitry. As a result, this counter variation requires more flip-flops (sequential modules) than other counters.

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion I/O Description

Table 36 · I/O Description

Port Name	Size	Type	Req / Opt	Function
Data	WIDTH	input	Opt	Counter load input
Aclr	1	input	Opt	Asynchronous counter clear
Aset	1	Input	Opt	Asynchronous counter preset
Enable	1	input	Opt	Counter enable
Sload	1	input	Opt	Synchronous counter load

Port Name	Size	Type	Req / Opt	Function
Clock	1	input	Req	Clock
Updown	1	input	Opt	UP (Updown = 1), DOWN (Updown = 0)
Q	WIDTH	output	Req	Counter output bus
Tcnt	1	output	Opt	Terminal count

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion Parameter Description

Table 37 · Parameter Description.

Parameter	Value	Function
WIDTH	2-45	Word length of Data and Q
DIRECTION	**UP** DOWN UPDOWN	Counter direction UPDOWN is not supported for Register Look Ahead counters
CLR_POLARITY	0 **1** 2	Aclr can be active low, active high, or not used
EN_POLARITY	0 1 **2**	Enable can be active low, or active high or not used
LD_POLARITY	0 1 **2**	Sload can be active low, active high or not used
CLK_EDGE	**RISE** FALL	Sets rising or falling clock edge
TCNT_POLARITY	0 1 **2**	Tcnt can be active low, active high or not used Tcnt is not supported for Up/Down counters
SET_POLARITY	0 1 **2**	SET_POLARITY can be active low, active high or not used

Table 38 · Fan-in Control Parameters

Parameter	Value
CLR_FANIN CLR_VAL	AUTO **MANUAL** [default value for MANUAL is 1]
SET_FANIN SET_VAL	AUTO**MANUAL** [default value for MANUAL is 1]
CLK_FANIN CLK_VAL	AUTO**MANUAL** [default value for MANUAL is 1]
LD_FANIN LD_VAL	**AUTO**MANUAL [default value for AUTO is 12]
UPDOWN_FANIN UPDOWN_VAL	**AUTO**MANUAL [default value for AUTO is 12]

Linear Binary Counters - IGLOO, ProASIC3, SmartFusion and Fusion Implementation Parameters

Table 39 · Implementation Parameters

Parameter	Value	Description
LPM_HINT	TLACNT	Register look ahead model
	COMPCNT	Compact model

Table 40 · Functional Description[A]

Data	Aclr	Enable	Sload	Clock	Up down	Qn+1	Tcnt n+1
X	0	X	X	X	X	0's	0
X	1	X	X	f	X	Qn	Qn+1 = 2^{width}-1
X	1	0	0	f	X	Qn	Qn+1 = 2^{width}-1
m	1	X	1	f	X	M	Qn+1 = 2^{width}-1
X	1	1	0	f	1	Qn+1	Qn+1 = 2^{width}-1

Data	Aclr	Enable	Sload	Clock	Up down	Qn+1	Tcnt n+1
X	1	1	0	f	0	Q	Qn+1 = 2^{width}-1

A. Assume Aclr is active low, Enable is active high, Sload is active high, Clock is rising, and Tcnt is active high

Counter Performance Summary

The performance values below were achieved with a ProASIC3E A3PE600 device and default speed (-2) and temp range (COM).

Table 41 · Compact Counter Performance Summary

Counter Type	Width	Performance	COMB Tiles	SEQ Tiles	Total Tiles
Up with Clr or Pre	8	371.471	10	8	18
	16	316.056	24	16	40
	32	215.703	58	32	90
	45	202.922	86	45	131
Up Loadable (no Clr/Pre)	8	293.772	20	28	28
	16	256.739	48	64	64
	32	181.587	109	141	141
	45	165.893	163	208	208
Up/Down with Clr or Pre	8	294.464	28	36	36
	16	224.669	70	86	86
	32	172.533	169	201	201
	45	155.836	251	296	296
Up/Down Loadable (no Clr/Pre)	8	243.962	40	48	48
	16	206.186	98	114	114
	32	144.134	228	260	260

Counter Type	Width	Performance	COMB Tiles	SEQ Tiles	Total Tiles
	45	141.223	336	381	381

Table 42 · Register Look Ahead Counter Performance Summary

Counter Type	Width	Performance	COMB Tiles	SEQ Tiles	Total Tiles
Up with Clr or Pre	8	374.672	11	8	19
	16	372.024	27	17	44
	32	297	63	36	99
	45	249.813	99	53	152
Up Loadable (no Clr/Pre)	8	299.401	18	18	26
	16	298.151	47	47	64
	32	249.252	113	36	149
	45	216.92	176	53	229

CRC Minicore

Supported Families

Axcelerator

Related Topics

CRC Minicore Functionality

CRC Minicore I/O Description

CRC Minicore Parameter Description

CRC Minicore Implementation Rules

Key Features

- General-purpose Cyclic Redundancy code generator

- Fully synchronous, single-clock operation (greater than 100 MHz for many configurations)
- Parameterized arbitrary polynomial (from 1 up to 64-bit)
- Parameterized data input width
- Parameterized register initialization
- Parameterized bit and byte ordering

CRC Minicore Functionality

The CRC Minicore is a universal Cyclic Redundancy Check (CRC) Polynomial generator that validates data frames and ensures data integrity during data transmission.

To meet different application requirements, the CRC Minicore provides many different configuration parameters. These parameters control data width, a register initialization value, and other CRC output data characteristics.

- Data width specifies the number of bits over which the CRC Minicore generates the CRC value in a single clock cycle. For example, a CRC32 with 8-bit data width performs CRC calculations on 8 bits per clock.
- Register initialization provides the seed value for CRC generation.
- Additional parameters provide additional flexibility in controlling CRC data characteristics

CRC Variations - There are industry standards for the polynomial value (we will use the variable 'Y' to denote the polynomial value), such as Kermit, CANBus, etc. This option merely allows you to specify which polynomial value you wish to use.

CRC XOROUT - After the calculation of (stream of bits)/(polynomial value) is performed, the remainder (aka CRC) is inserted into the data stream and sent to the receiver.

This CRC value can be inserted in a variety of ways:

- **Non-Inverted:** CRC result inserted into data stream as-is.
- **Inverted:** CRC result inserted into data stream with every bit inverted
- **010101:** CRC result inserted into data stream with even bits inverted
- **101010:** CRC result inserted into data stream with odd bits inverted

Bit Order - In many cases, the CRC input data will be greater than a single bit. This option merely specifies which order the bits are processed. The reason is because some standards require a “reflection” of the bits during transmission (i.e. Reversing the bits), thereby for a byte-wide data, bit 7 or bit 0 could be seen first depending upon the transmission protocol used.

- **MSB first:** The high order bit is processed first
- **LSB first:** The low order bit is processed first

Byte Order - The reasoning for this follows above, except it extends it into bytes.

- **MSB First:** The high order byte is processed first

- **LSB First:** The low order byte is processed first

Initial Reg. Value -This is associated with the polynomial you choose, the CRC algorithm uses this value when it starts the algorithm. Review the Standard CRC Generator Parameters table below; each standard contains an initial value. Specify this value according to the standard you have chosen.

- **0000:** Initial Register is all 0's
- **FFFF:** Initial Register is all F's
- **Dynamic:** Initial Register is an input into the generated module of (bit-width) = (the polynomial size). The input name is init_reg.

Run/Shift Control - This option allows the CRC to function as a CRC function AND just a plain serial shifter. By enabling this option, the CRC module will contain an extra 1-bit input pin.

When the input is high, the CRC function operates as a CRC generator. When the input is low, the CRC is serially shifted to the right.

- **No:** Run/Shift input pin will not be generated for the module
- **Yes:** Run/Shift input pin will be generated for the module

For example, the CRC output XOR bit pattern parameter (XOROUT) controls inversion of the CRC value before injecting it into the data stream. Although the CRC Minicore generator provides seven commonly-used CRC polynomials, the core configurator also allows entry of an arbitrary polynomial. The polynomial bit size spans 1 to 64 inclusive.

Table 43 · XOROUT Configuration

XOROUT	Description
1	All bits are not inverted (000000000) xor CRC
2	All bits are inverted (..FFFFFFFF) xor CRC
3	Even bits are inverted, odd bits are not inverted (….10101010) xor CRC
4	Odd bits are inverted, even bits are not inverted (….01010101) xor CRC

CRC Minicore I/O Description

Table 44 · I/O Description

Port Name	width	Description
CLK	1	Clock port
rst_n	1	Asynchronous reset

Port Name	width	Description
init_n	1	Synchronous load CRC value
enable	1	CRC enable/disable control
data_in	Data_width	Input data word
CRC_in	Poly_size	CRC value to be load in
CRC_out	Poly_size	Generated CRC value

CRC Minicore Parameter Description

Table 45 · Standard CRC Generator Parameters

Name	Poly_width	Poly_value (HEX)	Initial	xorout
CRC32	32	04C11DB7	FFFF..	FFFFFF….
CRC16/ARC	16>	1005	FFFF...	FFFFF....
CCIT CRC16	16>	1021	FFFF….	FFFFFF….
CANBUS	16	4599	FFFFF…	FFFFFF….
ATM CRC10	10	233	FFFFF…	FFFFFF….
ATM CRC8	8	7	FFFF….>	FFFFF..….
kermit	16	8408	000000…	000000000

CRC Minicore Implementation Rules

Table 46 · CRC Operation Control

rst_n	init_n	enable	shift_run	Description
0	x	x	x	Asynchronous reset, set to initial register value
1	0	1	x	Synchronous initialization

rst_n	init_n	enable	shift_run	Description
1	1	0	x	Disable CRC generation, register holds the current value
1	1	1	0	CRC free-running, zeroes shifted in
1	1	1	1	Normal CRC operation, generate CRC from input data

Dual Data Rate (DDR) Register

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, and Axcelerator

Related Topics

DDR Register I/O Description

DDR Register Parameter Description

Key Features

- Parameterized for Data Width
- Choice of Input Buffers
- Choice of Output Buffers (ProASIC3/E only)
- Choice of Bidirectional buffers (ProASIC3/E only)

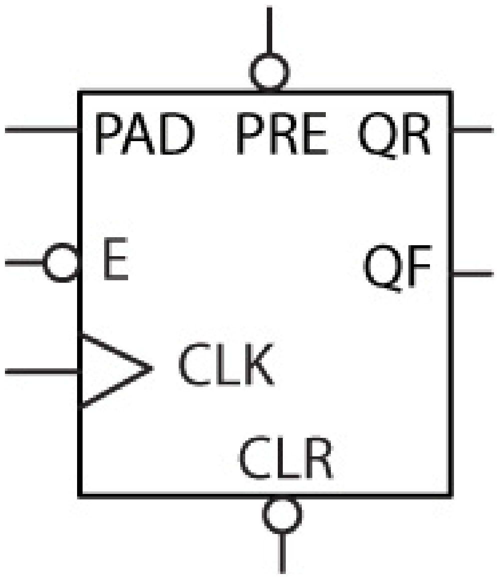
The core configurator software can generate Dual Data Rate Registers parameterized for a specific Data Width and with a choice of the type of Input Buffers.

In IGLOO, ProASIC3, SmartFusion and Fusion devices the DDR Registers may also be combined with output buffers or bi-directional buffers.

The Bidirectional enable signal polarity (TriEN) is selectable.

DDR Register I/O Description

Axcelerator DDR Register

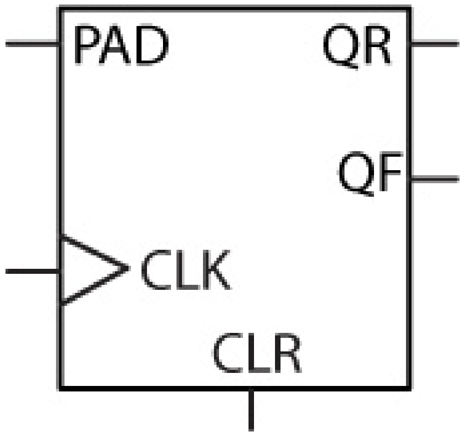


Port Description - Input Buffer plus DDR Register for Axcelerator

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input	Req.	Input Data
QR	WIDTH	Output	Req.	Output Data
QF	WIDTH	Output	Req.	Ouput Data
E	1	Input	Req.	Enable
CLK	1	Input	Req.	Clock
CLR	1	Input	Req.	Clear
PRE	1	Input	Req.	Preset

IGLOO, ProASIC3, SmartFusion and Fusion DDR Registers

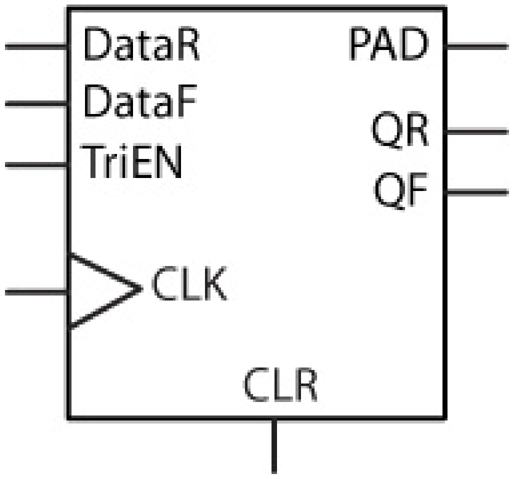
Input Buffer plus DDR Register for IGLOO, ProASIC3, SmartFusion and Fusion



Port Description - Input Buffer plus DDR Register for IGLOO, ProASIC3, SmartFusion and Fusion

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input	Req.	Input Data
QR	WIDTH	Output	Req.	Output Data
QF	WIDTH	Output	Req.	Ouput Data
CLK	1	Input	Req.	Clock
CLR	1	Input	Req.	Clear

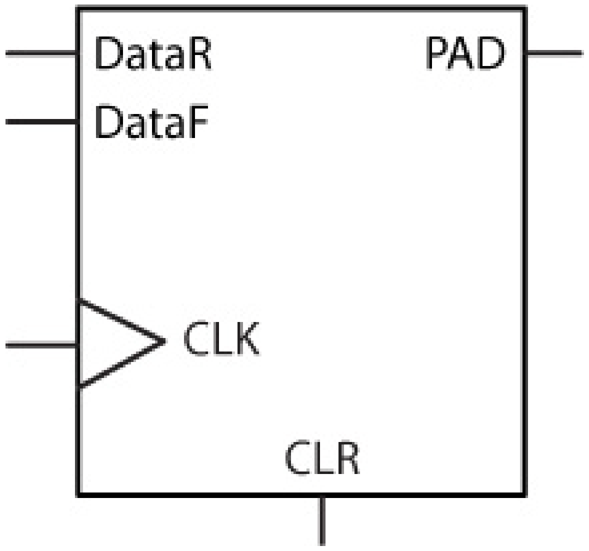
Bidirectional Buffer plus DDR Register for IGLOO, ProASIC3, SmartFusion and Fusion



Port Description - Bidirectional Buffer plus DDR Register for IGLOO, ProASIC3, SmartFusion and Fusion

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input/Output	Req.	Input/Output Data
DataR	WIDTH	Input	Req.	Input Data
DataF	WIDTH	Input	Req.	Input Data
QR	WIDTH	Output	Req.	Output Data
QF	WIDTH	Output	Req.	Output Data
TriEN	1	Input	Req.	Bibuf-Enable
CLK	1	Input	Req.	Clock
CLR	1	Input	Req.	Clear

DDR Register plus Output Buffer for IGLOO, ProASIC3, SmartFusion and Fusion



Port Description - DDR Register plus Output Buffer for IGLOO, ProASIC3, SmartFusion and Fusion

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Output	Req.	Output Data
DataR	WIDTH	Input	Req.	Input Data
DataF	WIDTH	Input	Req.	Input Data
CLK	1	Input	Req.	Clock
CLR	1	Input	Req.	Clear

Tri-State Buffer Plus DDR Register for IGLOO, ProASIC3, SmartFusion and Fusion

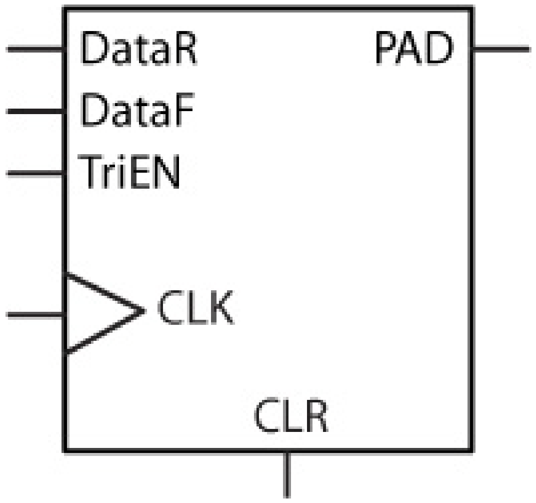


Table 47 · Port Description - Tri-State Buffer plus DDR Register for IGLOO, ProASIC3, SmartFusion and Fusion

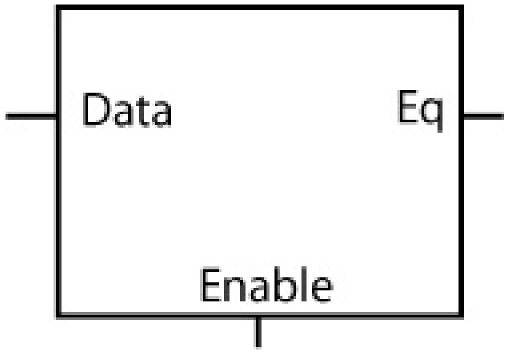
Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input/Output	Req.	Output Data
DataR	WIDTH	Input	Req.	Input Data
DataF	WIDTH	Input	Req.	Input Data
TriEN	1	Input	Req.	Tribuf Enable
CLK	1	Input	Req.	Clock
CLR	1	Input	Req.	Clear

DDR Register Parameter Description

Table 48 · Dual Data Rate Register Parameters

Parameter	Value	Function
WIDTH	1-128	Data Width

Decoder



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Decoder Functionality

Decoder I/O Description

Decoder Parameter Description

Key Features

- Parameterized output size
- Behavioral simulation RTL in VHDL and Verilog

Decoder Functionality

Table 49 · Decoder Functional Description [A]

Data	Enable	Eq
X	0	0's
m	1	dec[B](m)==decodes-1 &&[C] dec(m)==decodes-2 && … && dec(m)==0

A. Assume enable is active low and Eq is active high.

B. dec(m) defines the decimal value of m

C. && indicates bity concatenation

Decoder I/O Description

Table 50 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Data	decln[A]	Input	Req.	Input data
Enable	1	Input	Opt.	Enable
Eq	DECODES	Output	Req.	output

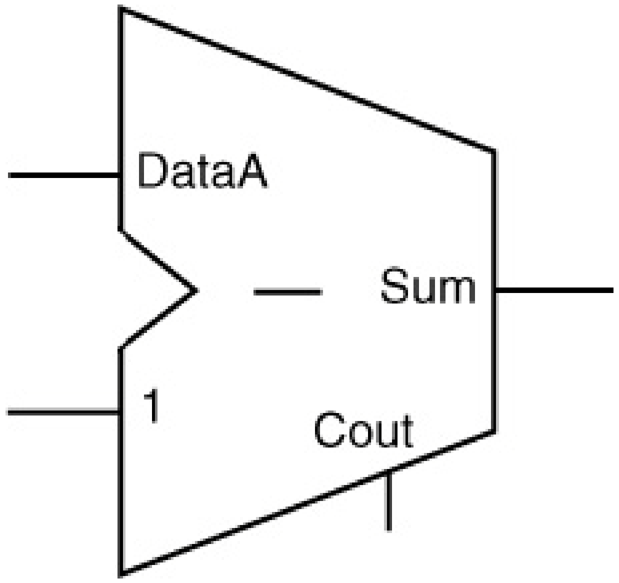
A. decln is an integer and $\log_2$ (DECODES) = decln d<$\log_2$ (DECODES + 1). If decln is equal to 1, then Data is scalar, else Data is a bus.

Decoder Parameter Description

Table 51 · Parameter Description

Parameter	Value	Function
DECODES	2-32	Word length of Eq
EN_POLARITY	0 1 **2**	Enable polarity (active high, active low or not used)
EQ_POLARITY	0 **1**	Eq polarity (active low or active high)

Decrementer



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^PLUS^, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Decrementer Functionality

Decrementer I/O Description

Decrementer Parameter Description

Decrementer Implementation Rules

Key Features

- Parameterized word length
- Optional Carry-out signals
- One very fast gate-level implementation, FC High Speed and FC Ripple available
- Behavioral simulation RTL in VHDL and Verilog

Decrementer Functionality

Table 52 · Functional Description

DataA	DataB	Sum	Cout
m	n	m - 1	(m-1) < 0

Decrementer I/O Description

Table 53 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
Sum	WIDTH	Output	Req.	Sum
Cout	1	Output	Opt.	Carry-out

Decrementer Parameter Description

Table 54 · Parameter Description

Parameter	Value	Function
WIDTH	2-32 2-156 for FC_FDEC and FC_RIPDEC	Word length of DataA and Sum
CO_POLARITY	0 1 **2**	Carry-out polarity (active low, active high, and not used)

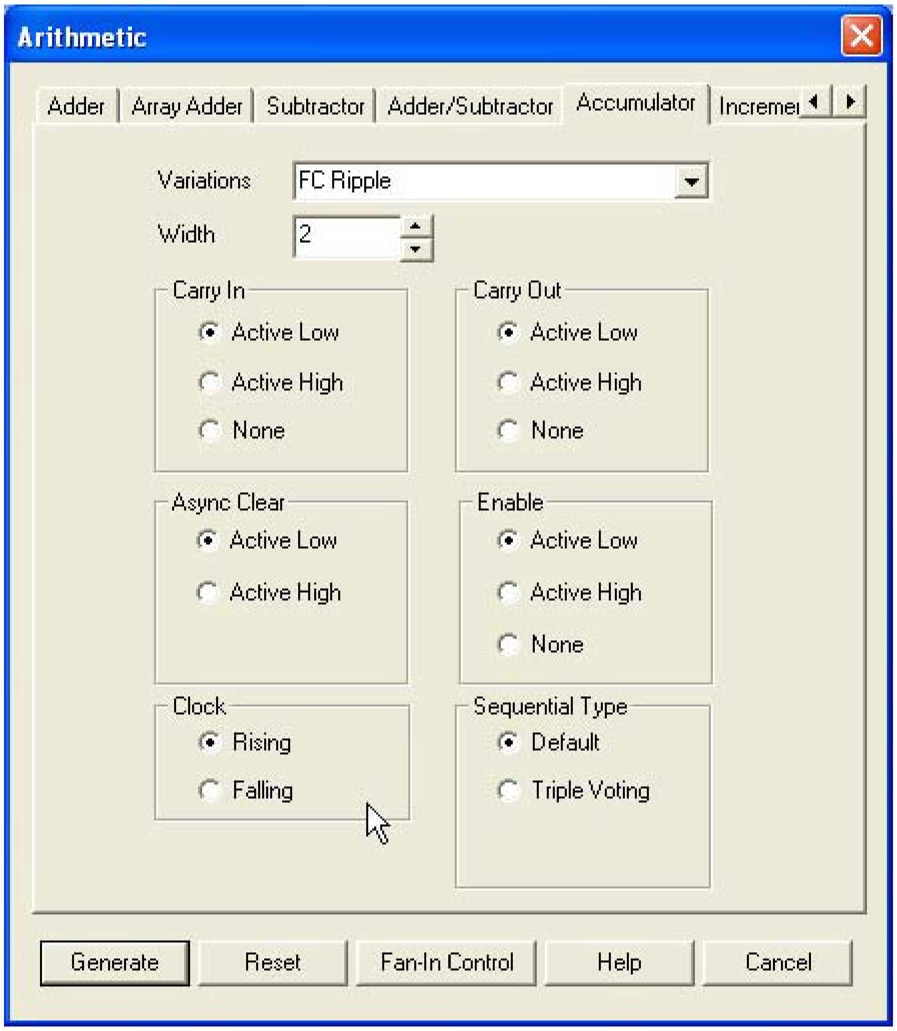
Decrementer Implementation Rules

Figure 6 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_ADD_SUB	Decrementer category
LPM_HINT	FDEC FC_FDEC and FC_RIPDEC, Fast Carry Versions	Very fast carry look ahead

Fast Carry Chains (Axcelerator only)

The Axcelerator Family offers fast-carry-chain cores for a compact design of Arithmetic Macros and Counters. Fast-Carry cores for Axcelerator are available in the Variations drop-down menu.



Variations: FC Ripple
Width: 2
Carry In: [✓ Active Low] Active High None
Carry Out: [✓ Active Low] Active High None
Async Clear: [✓ Active Low] Active High
Enable: [✓ Active Low] Active High None
Clock: [✓ Rising] Falling
Sequential Type: [✓ Default] Triple Voting

Figure 7 · Fast-Carry-Chain Cores in Variations Drop-Down Menu

You can generate FC cores via the module generator or infer them with synthesis tools such as Synplify Pro. FC cores are always the most area-efficient way to implement these modules. They are also superior in performance for designs up to 32 bits, although some modules may be inferior beyond 32-bits (incrementers, for example). Though the software offers both architecture types (FC cores and non-FC cores), Actel recommends you use FC cores to guarantee area efficiency.

In the Core Catalog, you can distinguish the FC cores from non-FC cores by the prefix "FC" or "Fast Carry" (e.g. "FC High Speed" versus "High Speed").

The core parameters used in the GEN file also use "FC" for distinction. For example the "High Speed" Adder using fast carry chains is specified by:

LPMTYPE:LPM_FC_ADD_SUB

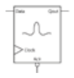
LPM_HINT:FC_FADD

while the corresponding non-FC version is specified by:

LPMTYPE:LPM_ADD_SUB

LPM_HINT:FADD

FIR Filter



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, eX, and SX-A

Related Topics

FIR Filter Functionality

FIR Filter I/O Description

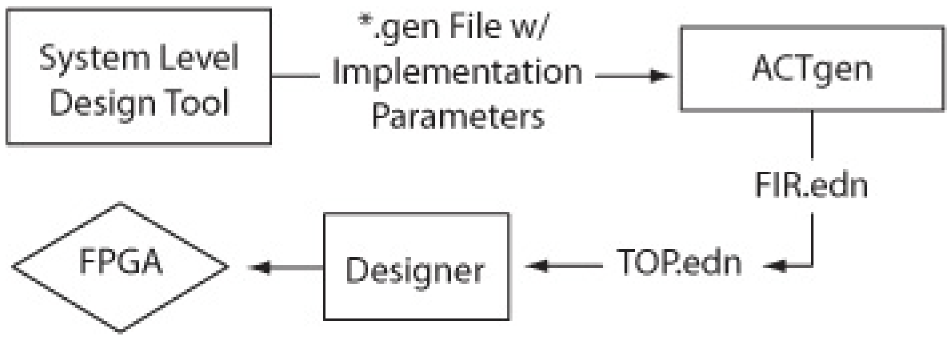
FIR Filter Parameter Description

FIR Filter Implementation Rules / Timing Diagrams

Key Features

- Variable input data width: 2- to 16-bit input data
- Variable output data width: 3- to 64 bit output data
- Support for up to 64 taps
- Support of symmetric coefficients
- Optional I/O insertion
- Optional registers for filter in- and output
- Verilog RTL model for simulation
- VHDL RTL model for synthesis (synthesized filter designs are usually slower, but more compact)

An overview of the design flow required for the FIR filter is shown in the figure below.



FIR Filter Design Flow

Filter Design Flow

Generate the filter coefficients and other implementation parameters using a system level design tool (like Matlab). This information is made available for the software in the form of a <design>.gen file.

From that point on, it follows the regular design flow as described in the Actel Quick Start Guide.

FIR Filter Functionality

The FIR-filter core supports symmetric, high-speed, parallel FIR-filters with up to 64 time taps.

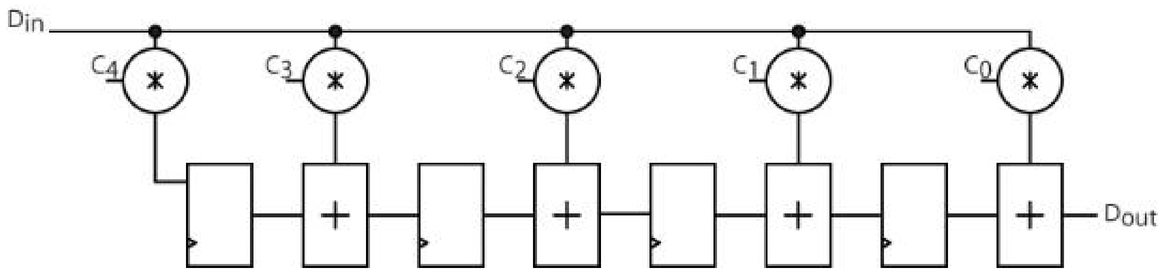


Figure 8 · Tap Transposed from FIR Filter

The architecture is a variation of the "transposed form" of the FIR filter as shown in the figure above, making use of the signed Constant Multiplier. The data is assumed to be signed. Data- and coefficient widths are the same (D_WIDTH).

The FIR filter figure above suggests that coefficients with a value of 0 are desirable for this type of architecture, since they will not generate any multiplication hardware. "Halfband" filters are trying to maximize the number of 0-coefficients and might result in significant area savings over regular filters of the same order.

FIR Filter I/O Description

Port Name	Size	Type	Req/Opt	Function
Data	D_WIDTH	input	Req.	Input Data

Port Name	Size	Type	Req/Opt	Function
Clock	1	input	Req.	Filter clock
Aclr	1	input	Opt.	Asynchronous Clear
Qout	O_WIDTH	input	Req.	Filter output = Sci * di

FIR Filter Parameter Description

Table 55 · Parameter Description

Parameter	Value	Function
D_WIDTH	3 .. 16	Input Data Width
O_WIDTH	3 .. 64	Output Data Width
TAPS	3 .. 64	Number of time taps
CLK_EDGE	**RISE**FALL	Clock sensitivity
CLR_POLA	2 0 1	None, active high, active low
PREC		Internal precision
INSERT_PAD	**NO** YES	Pad insertion
INSERT_IOREG	**NO** YES	Register inputs and outputs
C1 … C32	0 .. 2C_WIDTH	Two's-Complement coefficients (integers)

The output width O_WIDTH has no impact on the filter size. Internally, the core configurator software always uses the maximum precision filter, unless specified otherwise using the internal precision parameter PREC. If you set O_WIDTH to 0, the software uses the maximum output resolution (MAX_RES). For values of O_WIDTH greater than MAX_RES, the result is sign-extended. For values of O_WIDTH smaller than MAX_RES, the software cuts some of the lower bits. An upper estimate for MAX_RES is

MAX_RES ≤ 2 × D_WIDTH + [log2(TAPS)]

For example, a 12-tap filter with 8-bit data and coefficients might yield up to (8 + 8 + 4) bits = 20-bit output resolution.

The coefficients C1 to C16 are positive integers, which will be interpreted as Two's-Complement numbers. That means 0 to $2^{C\_WIDTH-1-1}$ are considered positive, and $2^{C\_WIDTH-1-1}$ to $2^{C\_WIDTH-1}$ will be interpreted as negative numbers.

Only unique coefficients need to be specified properly, all other coefficients need to be set to any value, e.g. '0'. An N-tap filter requires (N / 2) + (N % 2) unique coefficients.

Only unique coefficients need to be specified properly, all other coefficients need to be set to any value, e.g. '0'. An N-tap filter requires (N / 2) + (N % 2) unique coefficients.

FIR Filter Implementation Rules / Timing Diagrams

Table 56 · FIR Filter Parameter Rules

Family	Variation	Parameter Rules
All	FIR2	PREC >= O_WIDTH
SX, SX-A	All	O_WIDTH <= 32
SX, SX-A	All	TAPS <= 32

Table 57 · FIR Filter Implementation Parameters

Parameter	Value	Description
LPMTYPE	LPM_FIR	FIR-filter category
LPM_HINT	FIR1	Basic options
	FIR2	Advanced options

Table 58 · Internal Precision (PREC)

Variation	Value	Description
Basic Options	**97**, 0	Maximum output resolution, same as O_WIDTH
Advanced Options	PREC	See parameter rules

Internal Precision (PREC) specifies the minimum number of bits:

- For the time tab registers

- From multiplier outputs kept for further processing
- From adder outputs kept for further processing

Currently, the RTL-model does not reflect the PREC parameter, so there may be differences between the simulated output of the structural netlist and the RTL-model for the low-order bits.

Integer Values Coefficient File

The Integer Values Coefficient File consists of the conversion of the quantized coefficients into regular integers. This file can be directly imported into your project.

Sample Integer Coefficient File

```
2048
2037
0
48
2048
1892
0
630
1026
630
0
1892
2048
48
0
2037
2048
```

Bi-Directional Buffers

Generates bi-directional buffers with a specified data width.

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC<u>PLUS</u>, ProASIC, Axcelerator, MX, eX, and SX/SX-A

Related Topics

Bi-Directional Buffers I/O Description

Bi-Directional Buffers Parameter Description

Bi-Directional Buffers Implementation Rules

Key Features

- Parameterized for data width
- Choice of data buffers (Regular, Special, Pull-Up, Pull-Down)

Bi-Directional Buffers I/O Description

Table 59 · I/O Description

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Inout	Req.	Inout Data
Data / A (Flash)	WIDTH	Input	Req.	Input Data
Trien / ENABLE (Flash)	1	Input	Req.	Enable
Y	WIDTH	Output	Req.	Output Data

Bi-Directional Buffers Parameter Description

Table 60 · Parameter Description

Parameter	Value	Function
WIDTH	1-99 (Limit may vary depend-ing on the family)	Data Width
VOLT (Flash Only)	0,1,2,3,4,5	Choice of different voltage levels. 3.3v(PCI), 3.3v & Low Strength, 2.5v & High Strength*, 2.5v & Low Strength*, 2.5v (Low Power) & High

Parameter	Value	Function
		Strength, or 2.5v (Low Power) & Low Strength
SLEW (Flash Only)	0,1,2	Choice of the slew rates: Low, Normal, or High
PULLUP	NO / YES	Choice of Pull up version
TRIEN_POLARITY / EN_POLARITY (Flash)	0,1	Enable Polarity
TYPE (IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, and RTAX-S Only)	REG, S_8, S_12, S_16, S_24, F_8, F_12, F_16, F_24, LVCMOS25, LVCMOS18, LVCMOS15, PCI, PCIX, GTLP25, GTLP33, S_8U, S_12U, S_16U, S_24U, F_8U, F_12U, F_16U, F_24U, S_8D, S_12D, S_16D, S_24D, F_8D, F_12D, F_16D, F_24D, LVCMOS25U, LVCMOS25D, LVCMOS18U, LVCMOS18D, LVCMOS15U, LVCMOS15D, LVCMOS12, LVCMOS12D, LVCMOS12U, HSTL_I, SSTL2_I, SSTL2_II, SSTL3_I, SSTL3_II	Type of Buffer. Note : "S" in S_* denotes Low Slew Rage and "F" in F_* denotes High Slew Rate. Also 8,12,16,24 denote Output drive strengths of 1x, 2x, 3x, 4x respec-tively

*Not supported in ProASICPLUS

Bi-Directional Buffers Implementation Rules

Table 61 · Implementation Rules

Parameter	Value	Function
LPMTYPE	LPM_IO / LPM_IOB_IO	Bi-directional Buffers

Parameter	Value	Function
LPM_HINT	BIBUF / IOB, GLMIOB (Flash)	Regular Bi-directional Buffers / IO pad with Global Connection, Two Multiplexed Pads & Global Con-nection (ProASIC)
	BIBUF_SP (Axcelerator Only)	Special Bi-directional Buffers
	BIBUF_PU (Axcelerator Only)	Pull-up Bi-directional Buffers
	BIBUF_PD (Axcelerator Only)	Pull-down Bi-directional Buffers

Global Buffers

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, and ProASIC

Related Topics

Global Buffers I/O Description

Global Buffers Parameter Description

Global Buffers Implementation Rules

Key Features

- Parameterized for data width
- Choice of buffers (Regular, Multiplexed, Internal Driver)

Global buffers with a specified data width.

Global Buffers I/O Description

Table 62 · I/O Description

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input	Req.	Inout Data

Port Name	Size	Type	Req/Opt	Function
A	WIDTH	Input	Req.	Input Data
ENABLE	1	Input	Req.	Enable
GL	1	Output	Req.	Output Data
Y	WIDTH	Output	Req.	Output Data

Global Buffers Parameter Description

Table 63 · Parameter Description

Parameter	Value	Function
WIDTH	1-499 (Limit may vary depending on the type)	Data Width
VOLT	0,1,2	Choice of different voltage levels: 3.3V, 2.5V*, 2.5V (Low Power)
PULLUP	NO / YES	Choice of Pull-up version

Global Buffers Implementation Rules

Table 64 · Implementation Rules

Parameter	Value	Function
LPMTYPE	LPM_GL_IO	All buffers
LPM_HINT	GL	Standard Global buffer
	GLIB	Standard Global buffer w/ an Input buffer
	GLMIB	Standard Global buffer with Multiplexed Input buffer
	GLINT	Global internal driver

Input Buffers

Input buffers with a specified data width.

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Input Buffers I/O Description

Input Buffers Parameter Description

Input Buffers Implementation Rules

Key Features

- Parameterized for data width
- Choice of data buffers (Regular, Special, Pull-Up, Pull-Down)

Input Buffers I/O Description

Table 65 · I/O Description

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input	Req.	Input Data
PADP (LVDS and LVPECL, Axcelerator Only)	WIDTH	Input	Req.	Input Data for LVDS and LVPECL
PADN (LVDS and LVPECL, Axcelerator Only)	WIDTH	Input	Req.	Input Data for LVDS and LVPECL
Y	WIDTH	Output	Req.	Output Data

Input Buffers Parameter Description

Table 66 · Parameter Description

Parameter	Value	Function
WIDTH	1-99 (Limit may vary depending on the family)	Data Width
PULLUP (flash Only)	NO / YES	Choice of Pull-up version
VOLT (flash Only)	0,1,2	Choice of different volt-age levels. 3.3v, 2.5v* or 2.5v(Low Power)
TYPE (IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, and RTAX-S Only)	REG, LVCMOS25, LVCMOS18, LVCMOS15, PCI, PCIX, GTLP25, GTLP33, HSTL_I, HSTL_II, SSTL3_I, SSTL3_II, SSTL2_I, SSTL2_II, LVDS, LVPECL, LVCMOS25U, LVCMOS25D, LVCMOS18U, LVCMOS18D, LVCMOS15U, LVCMOS15D, LVCMOS12, LVCMOS12D, LVCMOS12U	Type of Buffer

*Not supported in ProASICPLUS

Input Buffers Implementation Rules

Table 67 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_IO/ LPM_IB_IO (Flash)	Input Buffers
LPM_HINT	INBUF / IB (Flash)	Regular Input Buffers
	INBUF_SP (Axcelerator Only)	Special Input Buffers
	INBUF_PU (Axcelerator Only)	Pull-up Input Buffers
	INBUF_PD (Axcelerator Only)	Pull-down Input Buffers

Output Buffers

Output buffers with a specified data width.

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Output Buffers I/O Description

Output Buffers Parameter Description

Output Buffers Implementation Rules / Timing Diagrams

Key Features

- Parameterized for data width
- Choice of data buffers (Regular, Special, Pull-Up, Pull-Down)

Output Buffers I/O Description

Table 68 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Data/A(ProASICPLUS)	WIDTH	Input	Req.	Input Data
PAD	WIDTH	Output	Req.	Output Data

Output Buffers Parameter Description

Table 69 · Parameter Description

Parameter	Value	Function
WIDTH	1-99 (Limit may vary depending on the family)	Data Width
VOLT (ProASICPLUS Only)	0,1,2,3,4,5	Choice of different voltage levels. 3.3v(PCI), 3.3v & Low

Parameter	Value	Function
		Strength, 2.5v & High Strength*, 2.5v & Low Strength*, 2.5v (Low Power) & High Strength, or 2.5v (Low Power) & Low Strength
SLEW	0,1,2	Choice of different slew rates. Low, Normal or High
TYPE (IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, and RTAX-S Only)	REG, S_8, S_12, S_16, S_24, F_8, F_12, F_16, F_24, LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, PCI, PCIX, GTLP25, GTLP33, HSTL_I, HSTL_II, SSTL3_I, SSTL3_II, SSTL2_I, SSTL2_II, LVDS, LVPECL.	Type of Buffer Note : "S" in S_* denotes Low Slew Rate and "F" in F_* denotes High Slew Rate. Also 8,12,16,24 denote Output drive strengths of 1x, 2x, 3x, 4x respectively.

*Not supported in ProASICPLUS

Output Buffers Implementation Rules / Timing Diagrams

Table 70 · Output Buffers Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_IO / LPM_OB_IO (Proasic)	Output Buffers
LPM_HINT	OUTBUF / OB (Proasic)	Regular Output Buffers
	OUTBUF_SP (Axcelerator Only)	Special Output Buffers

PECL Global Buffers

Supported Families

ProASIC

Related Topics

PECL Global Buffers I/O Description

PECL Global Buffers Parameter Description

PECL Global Buffers Implementation Rules

Key Features

- Parameterized for data width
- Choice of buffers (Direct to Global, Multiplexed with Internal Signal)

PECL global buffers with a specified data width.

PECL Global Buffers I/O Description

Table 71 · I/O Description

Port Name	Size	Type	Req/Opt	Function
PECLIN	WIDTH	Input	Req.	Input Data
PECLREF	WIDTH	Input	Req.	Reference Data
A	WIDTH	Input	Req.	Input Data
ENABLE	1	Input	Req.	Enable
GL	WIDTH	Output	Req.	Output Data
Y	WIDTH	Output	Req.	Output Data

PECL Global Buffers Parameter Description

Table 72 · Parameter Description

Parameter	Value	Function
WIDTH	1-2	Data Width

PECL Global Buffers Implementation Rules

Table 73 · Implementation Rules

Parameter	Value	Function
LPMTYPE	LPM_GLPE_IO	PECL Global buffers
LPM_HINT	GLPE	Direct to Global
	GLPEMIB	Multiplexed with Internal Signal

Tri-State Buffers

Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, MX, eX, and SX/SX-A

Related Topics

Tri-State Buffers I/O Description

Tri-State Buffers Parameter Description

Tri-State Buffers Implementation Rules

Key Features

- Parameterized for data width
- Choice of data buffers (Regular, Special, Pull-Up, Pull-Down)

Tri-state buffers with a specified data width.

Tri-State Buffers I/O Description

Table 74 · I/O Description

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Inout	Req.	Inout Data
Data / A (Flash)	WIDTH	Input	Req.	Input Data
Trien / ENABLE (Flash)	1	Input	Req.	Enable

Tri-State Buffers Parameter Description

Table 75 · Parameter Description

Parameter	Value	Function
WIDTH	1-99 (Limit may vary depend-ing on the family)	Data Width
VOLT (Flash only)	0,1,2,3,4,5	Choice of different voltage levels. 3.3v (PCI), 3.3v & Low Strength, 2.5v & High Strength*, 2.5v & Low Strength*, 2.5v (Low Power) & High Strength, or 2.5v (Low Power) & Low Strength
SLEW (Flash only)	0,1,2	Choice of the slew rates: Low, Normal, or High
TRIEN_POLARITY / EN_POLARITY (Flash only)	0,1	Enable Polarity
TYPE (IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, and RTAX-S Only)	REG, S_8, S_12, S_16, S_24, F_8, F_12, F_16, F_24, LVCMOS25, LVCMOS18, LVCMOS15, PCI, PCIX, GTLP25, GTLP33, S_8U, S_12U, S_16U, S_24U, F_8U, F_12U, F_16U, F_24U, S_8D, S_12D, S_16D, S_24D, F_8D, F_12D, F_16D, F_24D, LVCMOS25U, LVCMOS25D, LVCMOS18U, LVCMOS18D, LVCMOS15U, LVCMOS15D, LVCMOS12, LVCMOS12D, LVCMOS12U, HSTL_I, SSTL2_I, SSTL2_II, SSTL3_I, SSTL3_II	Type of Buffer. Note : "S" in S_* denotes Low Slew Rage and "F" in F_* denotes High Slew Rate. Also 8,12,16,24 denote Output drive strengths of 1x, 2x, 3x, 4x respec-tively

*Not supported in ProASICPLUS

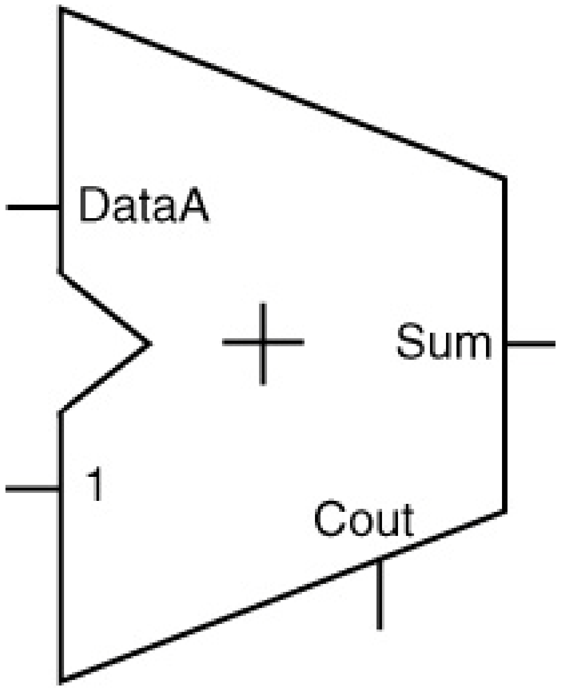
Tri-State Buffers Implementation Rules

Table 76 · Implementation Rules

Parameter	Value	Function
LPMTYPE	LPM_IO / LPM_OB_IO	Tri-State buffers
LPM_HINT	TRIBUFF / OTB (Flash)	Regular Tri-State Buffers
	TRIBUFF_SP (Axcelerator Only)	Special Tri-State Buffers
	TRIBUFF_PU (Axcelerator Only)	Pull-up Tri-State Buffers
	TRIBUFF_PD (Axcelerator Only)	Pull-down Tri-State Buffers

* not supported in ProASICPLUS

Incrementer



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^PLUS^, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Incrementer Functionality

Incrementer I/O Description

Incrementer Parameter Description

Incrementer Implementation Rules

Key Features

- Parameterized word length
- Optional Carry-out signals
- One very fast gatelevel implementation, FC High Speed and FC Ripple available
- Behavioral simulation RTL in VHDL and Verilog

Incrementer Functionality

Table 77 · Functional Description

DataA	Sum	Cout
m	m + 1	(m + 1) Š 2^{width}

Incrementer I/O Description

Table 78 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
Sum	WIDTH	Output	Req.	Sum
Cout	1	Output	Opt.	Carry-out

Incrementer Parameter Description

Table 79 · Parameter Description

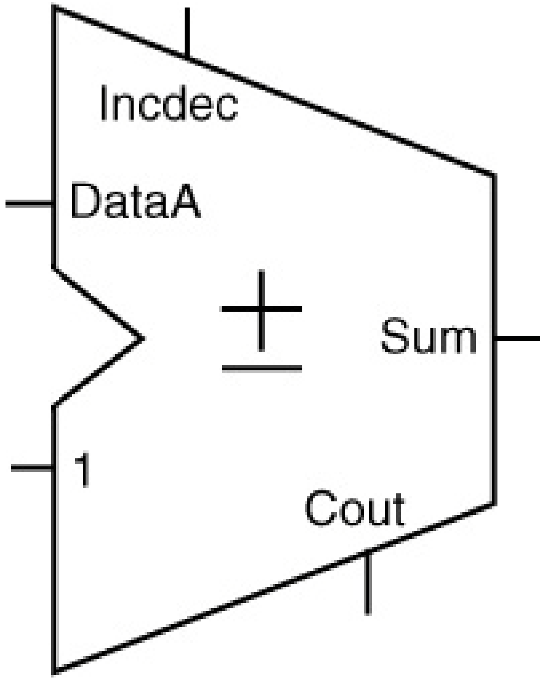
Parameter	Value	Function
WIDTH	2-32 2-156 for FC Macros	Word length of DataA and Sum
CO_POLARITY	0 1 **2**	Carry-out polarity (active low, active high, and not used)

Incrementer Implementation Rules

Table 80 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_ADD_SUB	Incrementer category
LPM_HINT	FINC; FC_FINC, FC_RIPINC	Very fast carry look ahead

Incrementer / Decrementer



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Incrementer / Decrementer Functionality

Incrementer / Decrementer I/O Description

Incrementer / Decrementer Parameter Description

Incrementer / Decrementer Implementation Rules

Key Features

- Parameterized word length
- Optional Carry-out signals
- One very fast gate-level implementation
- Behavioral simulation RTL in VHDL and Verilog

Incrementer / Decrementer Functionality

Table 81 · Functional Description

DataA	Incdec	Sum	Cout
m	1	m + 1	(m + 1) less than or equal to 2^{width}
m	0	m - 1	(m - 1) < 0

Incrementer / Decrementer I/O Description

Table 82 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
Sum	WIDTH	Output	Req.	Sum
Cout	1	Output	Opt.	Carry-out
Incdec	1	Input	Req.	Increment (Incdec = 1) or decrement (Incdec = 0)

Incrementer / Decrementer Parameter Description

Table 83 · Parameter Description

Parameter	Value	Function
WIDTH	2-32 2-156 for FC_FINCDEC and FC_RIPINCDEC	Word length of DataA and Sum
CO_POLARITY	0 1 **2**	Carry-out polarity (active low, active high, and not used)

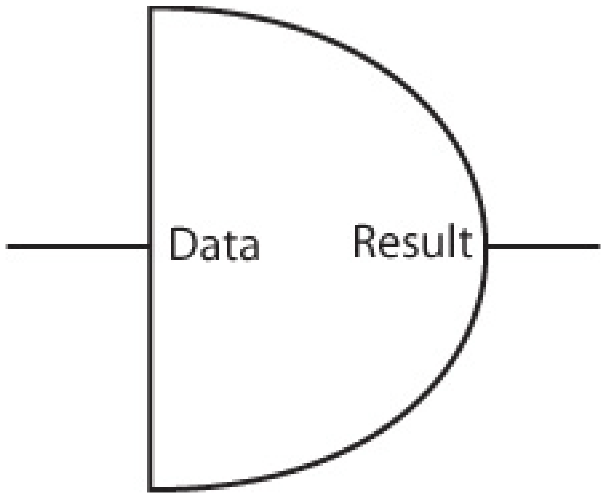
Incrementer / Decrementer Implementation Rules

Table 84 · Implementation Rules

Parameter	Value	Description

Parameter	Value	Description
LPMTYPE	LPM_ADD_SUB	Incrementer/Decrementer category
LPM_HINT	FINCDEC FC_FINCDEC FC_RIPINCDEC	Very fast carry look ahead

Logic - AND



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Logic - AND Functionality

Logic - AND I/O Description

Logic - AND Parameter Description

Key Features

- Parameterized OR size
- Behavioral simulation RTL in VHDL and Verilog

Logic - AND Functionality

Table 85 · Functional Description (result is Active High)

Data	Result
m	m[0] and m[1] and … and m[SIZE-1]

Logic - AND I/O Description

Table 86 · I/O Description

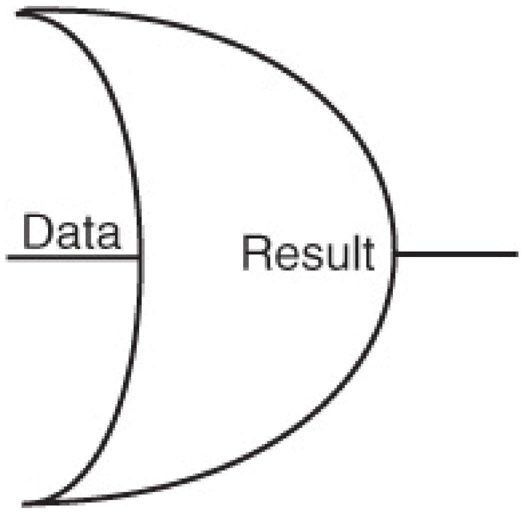
Port Name	Size	Type	Req/Opt	Function
Data	SIZE	Input	Req.	Input data
Result	1	Output	Req.	Output

Logic - AND Parameter Description

Table 87 · Parameters

Parameter	Value	Function
SIZE	2-64	Word length of data
RESULT_POLARITY	0 **1**	Output polarity (active low or active high)

Logic - OR



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Logic - OR Functionality

Logic - OR I/O Description

LLogic - OR Parameter Description

Key Features

- Parameterized OR size
- Behavioral simulation RTL in VHDL and Verilog

Logic - OR Functionality

Table 88 · Logic - OR Functional Description (result is Active High)

Data	Result
m	m[0] or m[1] or … or m[SIZE-1]

Logic - OR I/O Description

Table 89 · Logic - OR I/O Description

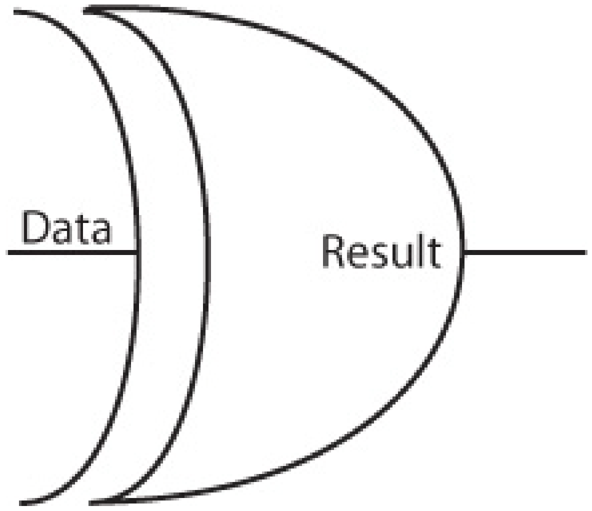
Port Name	Size	Type	Req/Opt	Function
Data	SIZE	Input	Req.	Input data
Result	1	Output	Req.	Output

LLogic - OR Parameter Description

Table 90 · Logic - OR Parameter Description

Parameter	Value	Function
SIZE	2-64	Word length of data
RESULT_POLARITY	0 **1**	Output polarity (active low or active high)

Logic - XOR



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Logic - XOR Functionality

Key Features

- Parameterized XOR size
- Behavioral simulation RTL in VHDL and Verilog

Logic - XOR Functionality

Table 91 · Functional Description (result is Active High)

Data	Result
m	m[0] xor m[1] xor … xor m[SIZE-1]

Logic - XOR I/O Description

Table 92 · I/O Description

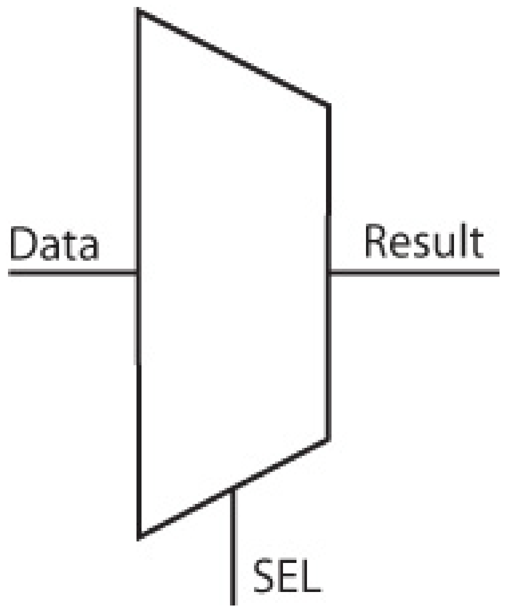
Port Name	Size	Type	Req/Opt	Function
Data	SIZE	Input	Req.	Input data
Result	1	Output	Req.	Output

Logic - XOR Parameter Description

Table 93 · Parameter Description

Parameter	Value	Function
SIZE	2-64	Word length of data
RESULT_POLARITY	0 **1**	Output polarity (active low or active high)

Multiplexor



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Multiplexor I/O Description

Multiplexor Parameter Description

Key Features

- Parameterized word length
- Parameterized multiplexer input number
- Behavioral simulation RTL in VHDL and Verilog

Multiplexor I/O Description

Table 94 · I/O Description

Port Name	Size	Type	Req/Opt	Function
$Data_{0\_port}$	WIDTH	Input	Req.	Input data
$Data_{1\_port}$	WIDTH	Input	Req.	Input data

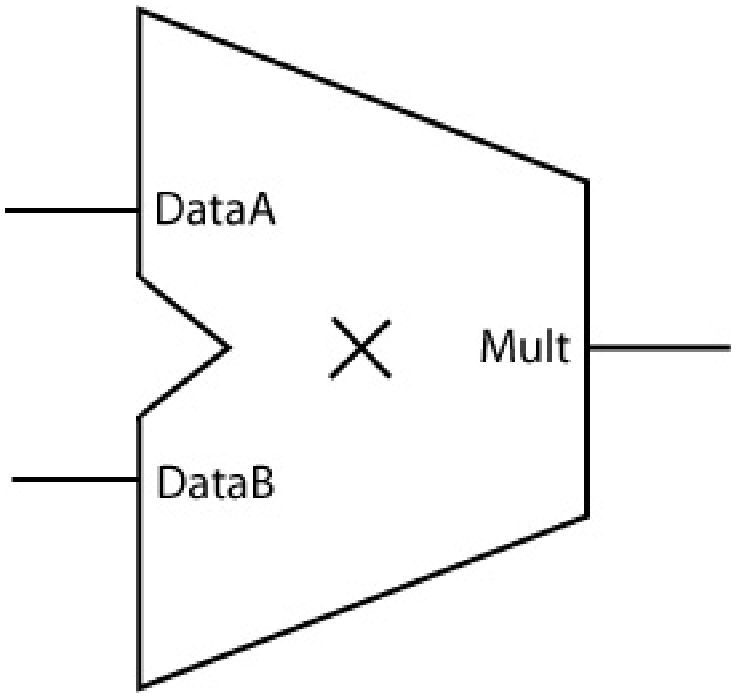
Port Name	Size	Type	Req/Opt	Function
…	…	…	…	…
$Data_{SIZE-1\_port}$	WIDTH	Input	Req.	Input data
Sel_0	1	Input	Req.	Select line
Sel_1	1	Input	Req.	Select line
…	…	…	…	…
$Sel_{SIZELN-1}$	1	Input	Req.	Select line
Result	WIDTH	Output	Req.	Output

Multiplexor Parameter Description

Table 95 · Parameter Description

Parameter	Family	Value	Function
WIDTH	ProASIC^{PLUS}	1-48	Word length of Data
	All Others	1-32	
SIZE	All	2-32	Number of data inputs

Multiplier



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Multiplier Functionality

Multiplier I/O Description

Multiplier Parameter Description

Multiplier Implementation Rules

Key Features

- Parameterized word lengths and constant values
- Unsigned and signed (Two's-Complement) data representation
- Booth or array implementation
- Optional pipelining
- Behavioral simulation RTL in VHDL and Verilog

Multiplier Functionality

Table 96 · Functional Description

DataA	DataB	Mult1[A]
m	n	m * n

A. If pipelined, the sum is correct (available) after <latency> cycles. Latency is a function of WIDTHA and WIDTHB, or the number of pipelined stages mentioned specifically (e.g. 1 or 2 pipelines).

Table 97 · Functional Description

DataA	DataB	Mult0/1[A]
m	n	Mult1 + Mult2 = m * n

A. Mult1<0> is always 0

In the Architecture Comparison tables below, WIDTHA = WIDTHB.

Table 98 · Axcelerator Multiplier Architecture Comparison: Speed

Architecture/Speed	1 (fastest)	2	3 (slowest)
Parallel-2 Array Multiplier	width <= 8 bit	8 bit < width <= 10 bit	width > 10 bit
FC Booth-1	8 bit < width <= 20 bit	width <= 8 bit or width > 20 bit	
FC Booth-2	width > 20 bit	10 bit < width <= 20 bit	width <= 10 bit

Table 99 · Axcelerator Multiplier Architecture Comparison: Area

Architecture/Speed	1 (smallest)	2	3 (largest)
Parallel-2 Array Multiplier	always		
FC Booth-1			always
FC Booth-2		always	

Advanced Options

Click the Advanced button (available for ProASIC<u>PLUS</u> and Axcelerator devices) to specify pipeline stages. If you are using a ProASIC<u>PLUS</u> device, you can insert (default setting) or omit the final Adder stage.

Omitting the Final Adder

You can choose not to instantiate the final adder in the multiplier and add up the two buses Mult0 and Mult1 to the final result later in the design flow. This is often the most efficient implementation when a lot of partial results get summed up in a large summation network. The figure below shows an example for Y = (A x B) + C + D using the multiplier with two outputs in combination with the Array-Adder.

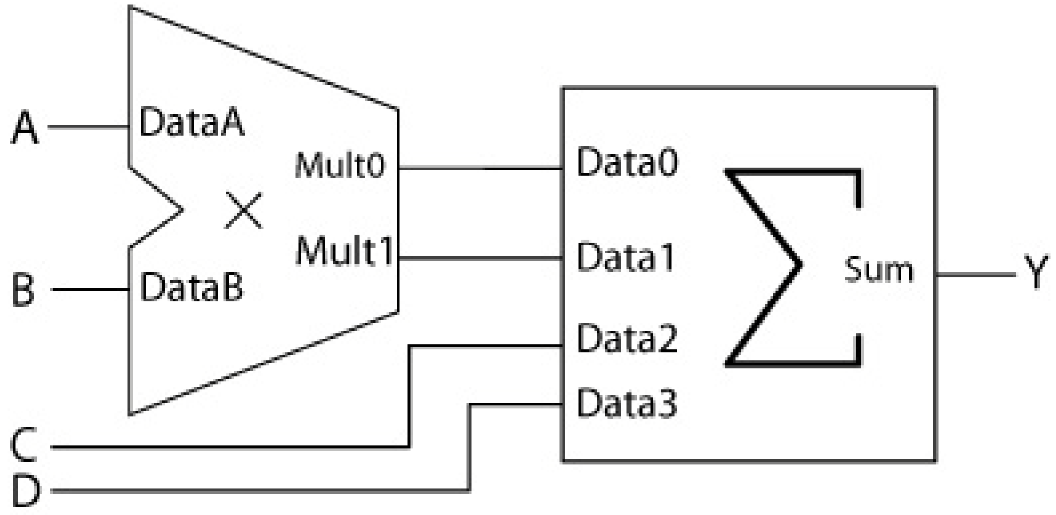


Figure 9 · Efficient Implementation Using the Two-Output Multiplier in Combination with the Array-Adder

Multiplier Pipelining

For ProASIC<u>PLUS</u> and Axcelerator devices you can specify the number of pipeline stages (1, 2, or 3). However, three pipeline stages increases performance only for high bitwidth. Click the Advanced button in the GUI to access pipelining.

Pipeline Stages	**WidthB**	
	w/ Final Adder	w/o Final Adder
1	>= 2	>= 5
2	>= 5	>= 7
3	>= 7	Not applicable

Multiplier Pipelining Functional Description

For 3200DX, MX, SX, SX-A, and eX the multiplier architecture does not allow you to select the latency of the pipelined multiplier or the number of logic levels between the pipeline stages. Registers are automatically inserted between the major components of the architecture, primarily the multiplexor and adder cores, as shown in the figure below.

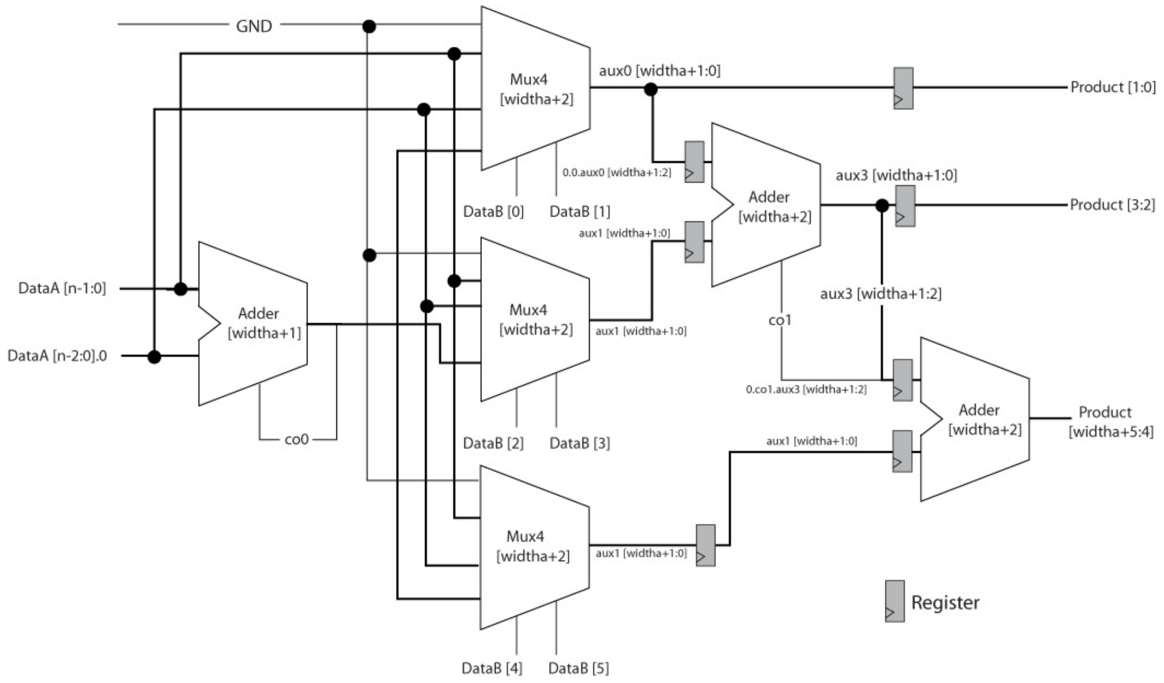


Figure 10 · Booth Multiplier Architecture

The number of pipeline stages is a function of the width of the DataB input. The number of logic levels per pipeline stage is a function of the width of the DataA input. Therefore, the number of logic levels per pipeline stage is equal to the number of logic levels of the first adder (WIDTHA + 1) plus 1 for the 4 to 1 multiplexor, as shown in the figure above.

Table 100 · Pipeline Stages as a Function of WidthB

WidthB Range	Pipeline Stages
2	0
3-4	1
5-8	2
9-16	3

Table 101 · Logic Levels per Pipeline Stage as a Function of WidthA

WidthA Range	Logic Levels
2-5	3
6-17	4
18-29	5

For more information on the Fast Carry-Chain cores available with the Axcelerator family, please refer to Fast Carry Chains (Axcelerator only).

Multiplier I/O Description

Table 102 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTHA	Input	Req.	Input data
DataB	WIDTHB	Input	Req.	Input data
Clock	1	Input	Opt.	Clock
Mult	WIDTHA+WIDTHB	Output	Opt.	DataA*DataB
Mult0	WIDTHA+WIDTHB	Output	Opt.	Mult0 + Mult1 = DataA*DataB
Mult1	WIDTHA+WIDTHB	Output	Opt.	

Multiplier Parameter Description

Table 103 · Parameter Description

Parameter	Family	Value	Function
WIDTHA [A]	ProASIC^{PLUS}, Axcelerator	2-64	Word length of DataA
	eX	2-14	
	Other	2-30	

Parameter	Family	Value	Function
WIDTHB	Same as WIDTHA		Word length of DataB
REPRESENTATION		**UNSIGNED** SIGNED	Data representation
FFTYPE [B]	ALL except flash	**REGULAR** TMR CC	FF Type Used (Default, Triple Voting, Combinatorial)
CLK_EDGE		**RISE** FALL	Clock (if pipelined)

A. For some of the multiplier variations there are small deviations from the limits mentioned to ensure that the multiplier fits in the largest device of the selected family.

B. TMR: Triple Module Redundancy. Choosing this option makes the core configurator software use TMR Flip-Flops, which are used to avoid Single Event Upsets (SEUs) for Rad-hard Designs. Choosing this option causes the Sequential resource usage to be tripled in families where no TMR is implemented in silicon.
CC: When combinatorial option is chosen for the Sequential Type, the FF is implemented using two Combinatorial Cells instead of one Sequential Cell. This is useful when no Sequential resources are available in the designs.

This option is applicable only to the pipelined multipliers.

The most important parameter rules are shown in the table below; additional rules may apply.

Table 104 · Parameter Rules

Family	Variation	Parameter rules
All	All	WIDTHA Š WIDTHB
eX	BOOTHMULT/P	WIDTHA + WIDTHB <= 15 (signed) / 16 (unsigned)
	BOOTHMULTP	For TMR restrictions for WIDTHA, WIDTHB
	BOOTHMULT2	WIDTHA + WIDTHB <= 17 (signed) / 18 (unsigned)
SX/SX-A	BOOTHMULT/P	WIDTHA + WIDTHB <= 32
	BOOTHMULT2	WIDTHA + WIDTHB <= 55
Axcelerator	ARRAYMULT	WIDTHA + WIDTHB <= 128
	PARRAYMULT	WIDTHA + WIDTHB <= 128

Family	Variation	Parameter rules
	FC_BOOTHMULT1	WIDTHA + WIDTHB <= 106
	FC_BOOTHMULT1	WIDTHA + WIDTHB <= 106
500K, PA	All	WIDTHA + WIDTHB <= 106
Other	All	WIDTHA + WIDTHB <= 32

Multiplier Implementation Rules

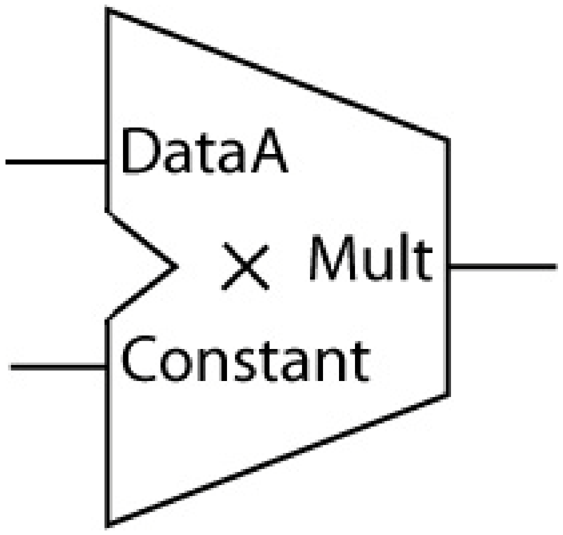
Table 105 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_MULT	Multiplier category
LPM_HINT	BOOTHMULT	Booth multiplier
	BOOTHMULT2	Booth multiplier without final Adder
	BOOTHMULTP[A]	Pipelined booth multiplier
LPMTYPE	LPM_FC_MULT	Fast Carry multiplier category (Axcelerator)[B]
	PARRAYMULT	Fast Carry array multipliers in parallel. Each array multipliers consists of 1-bit Multipliers (MULT1); the rows of the array use fast carry chains, but there is regular routing between columns.
	FCBOOTHMULT1	Booth-encoded Wallace-tree with Fast Carry final adder
	FCBOOTHMULT2	Booth-encoded multiplier with n-bit Fast Carry adder tree

A. Available for SX, SX-A, eX, ProASICPLUS

B. For information on multiplier area and performance please refer to the latest Actel application note available at http://www.actel.com

Constant Multiplier



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^PLUS, ProASIC, Axcelerator, SX-S, SX-A, and eX

Related Topics

Constant Multiplier Functionality

Constant Multiplier I/O Description

Constant Multiplier Parameter Description

Constant Multiplier Functionality

Key Features

- Parameterized word lengths and constant values
- Unsigned and signed (Two's-Complement) data representation
- Booth/Wallace architecture
- Behavioral simulation RTL (for non-pipelined multiplier only) in VHDL and Verilog

The Constant Multiplier performs the multiplication of a data-input with a constant value. Area and performance of the Constant Multiplier depend on the value of the constant. Specifically, area and performance depend on the number of groups of 1s in the bit pattern of the constant. As a result, the worst-case constant has a bit pattern of alternating 1s and 0s (…010101…). However, even for that worst-case the area and performance of the Constant Multiplier is superior to a regular Multiplier.

The Constant Multiplier core output wordlength is always double the input word length. Depending on the value of the constant, some of the most significant bits might be sign-extension bits. You may be able to reduce hardware by calculating the actual number of bits needed and cutting all sign-extension bits. For example:

width =4, Constant = 1100, representation=signed

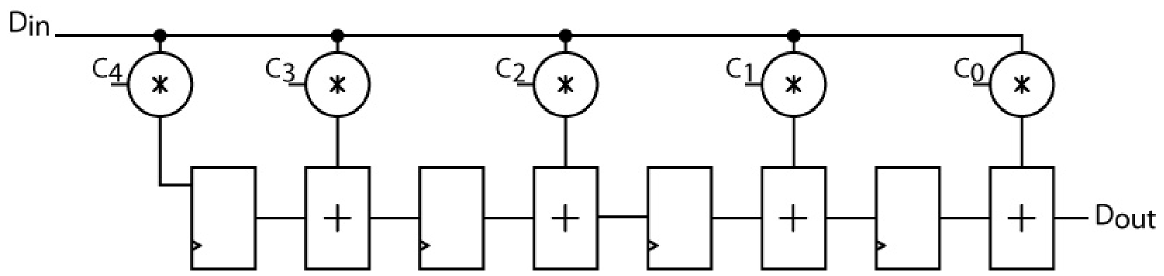
The worst case data for this example would be 1000 (-8) and therefore the worst case output data would be 010 0000 (-8 * -4 = 32). So with that we know that Mult<8> is just a sign-extension bit (Mult<8> = Mult<7>).

Keep in mind that some constant multiplications might be generated even more effectively, e.g. constants to the power of 2 are just shift-operations, or constants like 3,5,7,9,10, etc. can be generated using shift operations and a simple addition/subtraction (2+1, 4+1, 8-1, 8+1, 8+2, etc.) For these constants, the implementation of the Constant Multiplier might not be as efficient as using shift operations and/or Adders/Subtractors.

Usually synthesis infers regular Multipliers even for constant values. Therefore the use of the Constant Multiplier core in a design, which performs one or more multiplications with constant values, is expected to be very beneficial.

Constant Multiplier Functionality

An application example is FIR-filters with constant coefficients, where the computation is organized in the “transposed form” as indicated in the figure below.



FIR-Filter Organized in the Transposed Form Using Constant Multipliers

Constant Multiplier I/O Description

Table 106 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Data	WIDTH	Input	Req.	Input data
Mult	2*WIDTH	Output	Req.	Constant * Data

Constant Multiplier Parameter Description

Table 107 · Parameter Description

Parameter	Value	Function
WIDTH[A]	2-64	Word length Data
CONST	Constant	Constant value
RADIX	**HEX** BIN DEC	Radix for constant value
SIGN[B]	**0** 1	Positive, negative constant sign

A. For eX WIDTH is supported from 2-11

B. For signed constant multiplier

Parameter Rules:

1. DataA is always binary and of the size of Width.
2. Constant must be of the selected Radix and be of the selected width for HEX/BIN. The core configurator software automatically pads zeroes if they are missing.

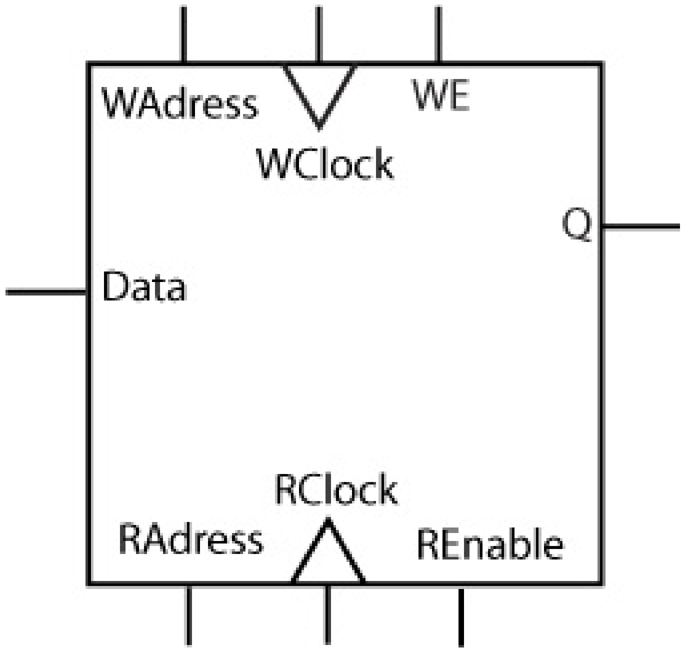
e.g.: Radix: BIN, Width: 5, Constant: 00010
Radix Hex, Width:8, Constant: 0A

Constant Multiplier Implementation Rules

Table 108 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_MULT	Constant multiplier category
LPM_HINT	UCMULT	Unsigned constant multiplier
	SCMULT	Signed constant multiplier

Register File for eX and SX-A Summary



Supported Families

eX, SX-A

Related Topics

[Register File I/O Description]

[Register File Parameter Description]

[Register File Implentation Rules / Timing Diagrams]

Key Features

- Parameterized word length and depth
- Dual-port synchronous RAM architecture
- Dual-port synchronous write, asynchronous read RAM architecture
- Write and read enable
- Behavioral simulation RTL in VHDL and Verilog

Description

The register file is a core unique to the SX, SX-A, and eX families. This core synthesizes the equivalent of small RAM blocks using ordinary logic, thereby making memory cells available to you even though the silicon does not explicitly have hardware support for RAM.

In synchronous mode, the read and write operations are totally independent and can be performed simultaneously. The operation of the register is fully synchronous with respect to the clock signals **WClock** and **RClock**. Data of value **Data** are written to the WAddress of the register memory space on the rising (RISE) or falling (FALL) edge of the clock **WClock** (WCLK_EDGE). Data are read from the register memory space at RAddress into Q on the rising (RISE) or falling (FALL) edge of the clock **RClock** (RCLK_EDGE).

The behavior of the Register is unknown, if designers write and read at the same address and **WClock** and **RClock** are not the same. The output Q of the register depends on the time relationship between the write and the read clock.

In asynchronous mode, the operation of the register is only synchronous with respect to the clock signal **WClock**. Data of value **Data** are written to the WAddress of the register memory space on the rising (RISE) or falling (FALL) edge of the clock **WClock** (WCLK_EDGE). Data are read from the register memory space at RAddress into Q after some delay when RAddress has changed.

The WIDTH (word length) and DEPTH (number of words) have continuous values but the choice of WIDTH limits the choice of DEPTH and vice versa.

The write enable (WE) and read enable (RE) signals are active high request signals for writing and reading, respectively. You may not utilize them.

Register File I/O Description

Table 109 · I/O Description

Port Name	Size	Type
Data	WIDTH	input
WE	1	Input
RE	1	Input
WClock	1	Input
RClock	1	Input
Q	WIDTH	Output

Register File Parameter Description

Table 110 · Parameter Description

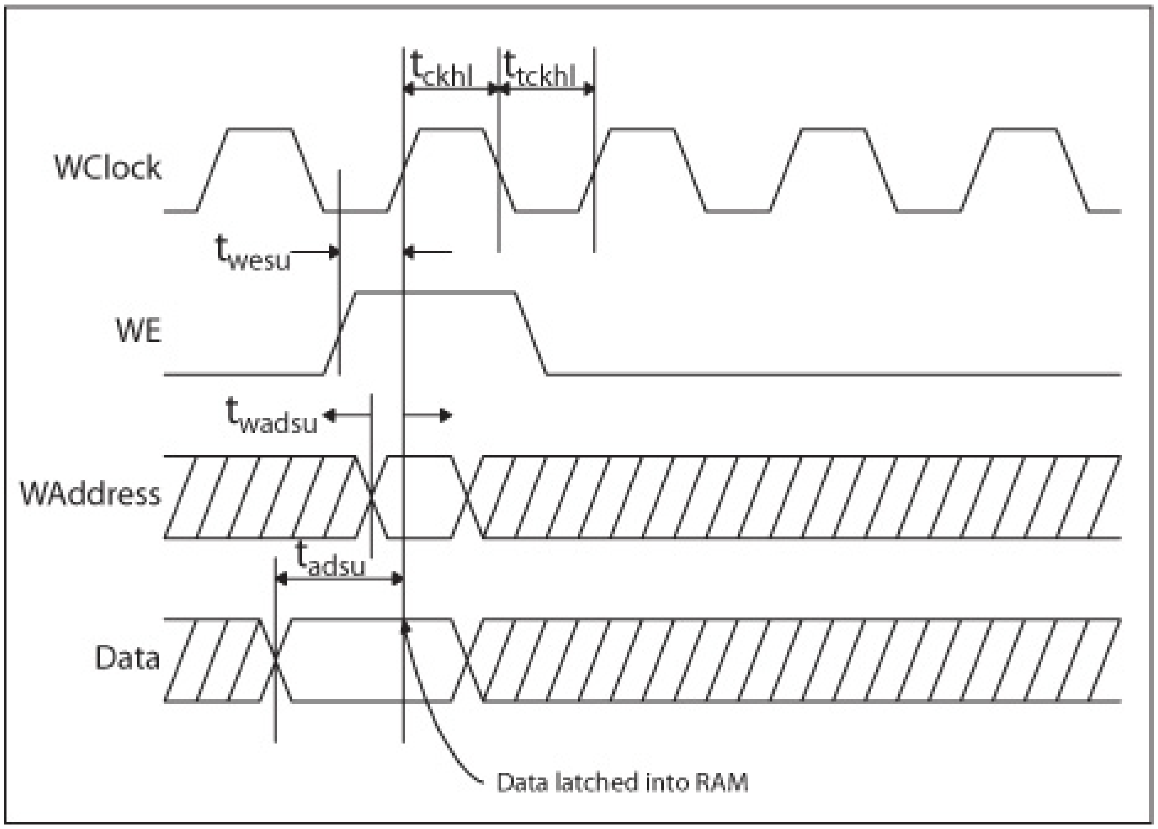
>Parameter	Value	Function
WIDTH	Width	Word length of Data and Q

>Parameter	Value	Function
DEPTH	Depth	Number of RAM words
WE_POLARITY	**1** 2	WE can be active high or not used
RE_POLARITY	**1** 2	RE can be active high or not used
WCLK_EDGE	**RISE** FALL	WClock can be rising or falling
RCLK_EDGE	**RISE** FALL NONE	RClock can be rising, falling, or not used

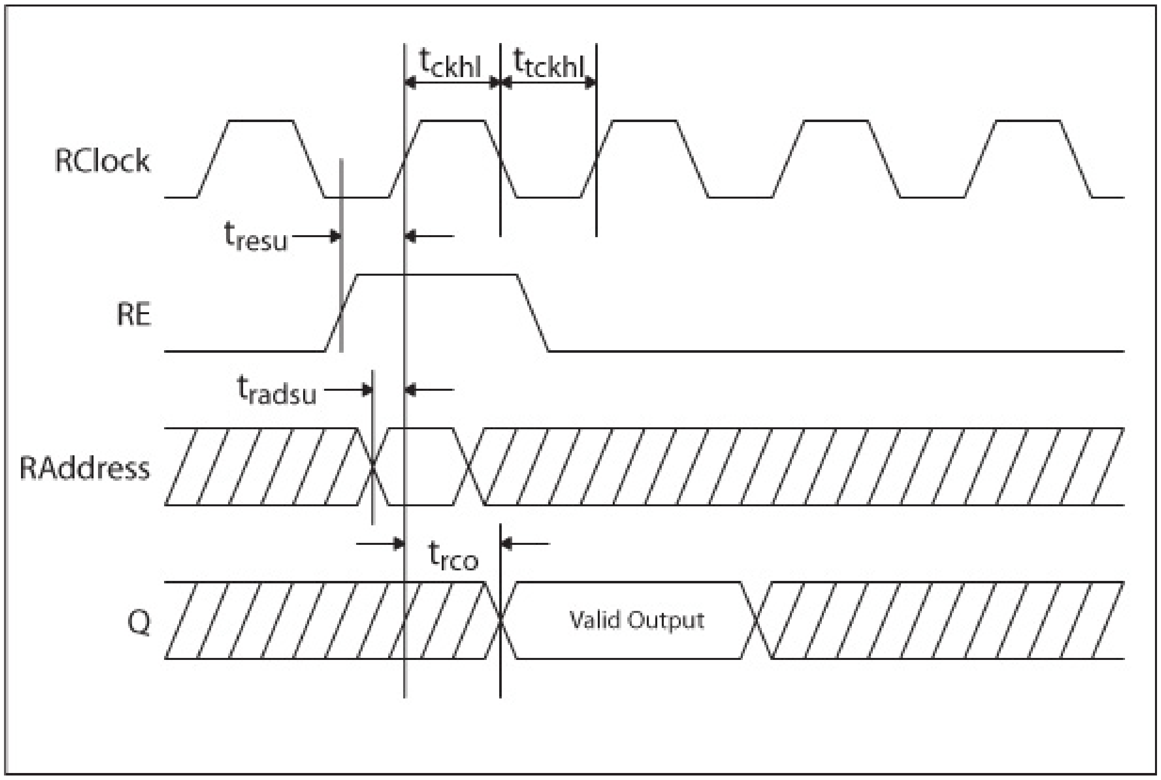
Register File Implentation Rules / Timing Diagrams

Timing Waveform Terminology

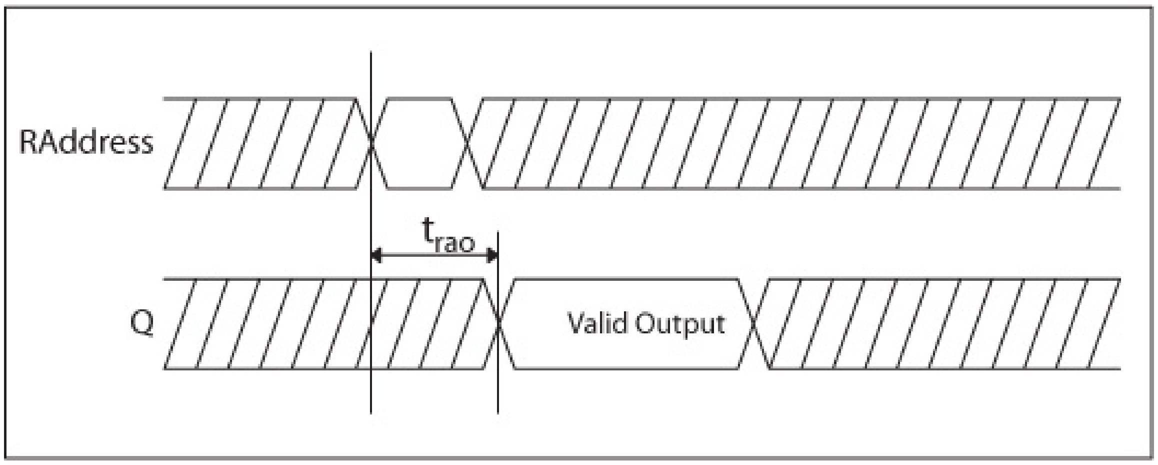
Term	Description	Term	Description
t_{ckhl}	Clock high/low period	t_{dsu}	Data setup time
t_{rp}	Reset pulse width	t_{rco}	Data valid after clock high/low
t_{wesu}	Write enable setup time	t_{rao}	Data valid after read address has changed
t_{resu}	Read enable setup time	t_{co}	Flip-flop clock to output



Ram Write Cycle

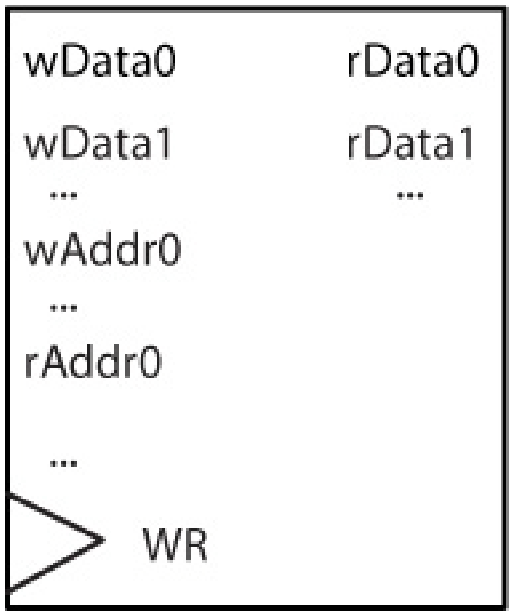


RAM Synchronous Read Cycle



RAM Asynchronous Read Cycle

Register File for ProASICPLUS Summary



Related Topics

Register File for ProASICPLUS I/O Description

Register File for ProASICPLUS Parameter Description

Register File for ProASICPLUS Implementation Rules

Features

- Parameterized word length and depth
- Two-port asynchronous register file
- Rising-edge triggered or level-sensitive
- Supported netlist formats: VHDL and Verilog

Description

Distributed memory can be generated as a two-port asynchronous register file or as an asynchronous FIFO. Distributed memories are made up of the logic tiles of the device. These memory files are netlists consisting of logic tiles and do not use embedded memory cells.

Register File for ProASICPLUS I/O Description

Table 111 · I/O Description

Port Name	Size	Type	Req/Opt	Function
wData<i>	1	Input	Req.	Input (Write) Data (i = 0 .. WIDTH-1)
wAddr<i>	1	Input	Req.	Write Address (i = 0 .. log2(WIDTH)-1)
rAddr<i>	1	Input	Req.	Read Address (i = 0 .. log2(WIDTH)-1)
WR	1	Input	Req.	Write Clock/Pulse (rising-edge triggered or level-sensitive)
rData<i>	1	Output	Req.	Output (Read) Data (i = 0 .. WIDTH-1)

Register File for ProASICPLUS Parameter Description

Table 112 · Parameter Description

Parameter	Value	Function
WIDTH	See Parameter Rules	Word length input/output data
DEPTH	2..64	Number of words for APA150
	2..64	Number of words for all other devices
TRIGGER	**edge**, level	Select between rising edge triggered and level sensitive write clock

Table 113 · Implemenation Parameters

Parameter	Value	Description
LPMTYPE	LPM_DIST_RAM	Generic Register File category
LPM_HINT	RAM_DISTH<#>	Horizontal Orientation; # represents the part number, and can be 050, 130, 180, 270 for ProASICPLUS 150, 300, 450, 600, 750, 1000 for ProASIC

Parameter	Value	Description
	RAM_DISTV<#>	Vertical Orientation

Table 114 · Parameter Rules

Device	Orientation	Parameter rules
A500K050	Horizontal	WIDTH = 2..30
	Vertical	WIDTH = 2..46
A500K130	Horizontal	WIDTH = 2..38
	Vertical	WIDTH = 2..78
A500K180	Horizontal	WIDTH = 2..46
	Vertical	WIDTH = 2..94
A500K270	Horizontal	WIDTH = 2..58
	Vertical	WIDTH = 2..110
APA075	Horizontal	WIDTH = 2..64
	Vertical	WIDTH = 2..22
APA150	Horizontal	WIDTH = 2..22
	Vertical	WIDTH = 2..62
APA300	Horizontal	WIDTH = 2..30
	Vertical	WIDTH = 2..62
APA450	Horizontal	WIDTH = 2..30
	Vertical	WIDTH = 2..94
APA600	Horizontal	WIDTH = 2..46
	Vertical	WIDTH = 2..110

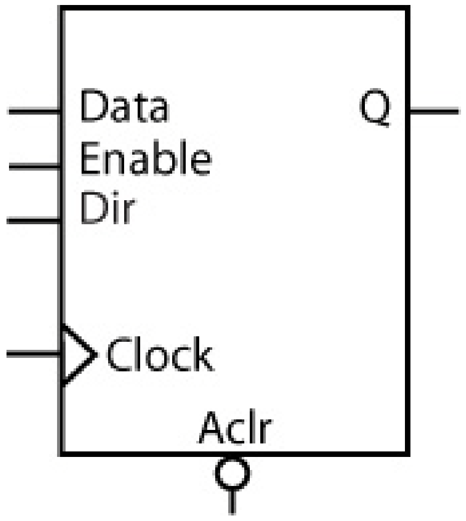
Device	Orientation	Parameter rules
APA750	Horizontal	WIDTH = 2..62
	Vertical	WIDTH = 2..126
APA1000	Horizontal	WIDTH = 2..78
	Vertical	WIDTH = 2..174

.

Register File for ProASICPLUS Implementation Rules

Please refer to the timing diagrams in the ProaSICPLUS datasheet. The datasheets are available on the Actel website at http://www.actel.com.

Barrel Shifter



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Barrel Shifter Functionality

Barrel Shifter I/O Description

Barrel Shifter Parameter Description

Barrel Shifter Implementation Rules

Key Features

- Parameterized word length
- Standard or pipelined
- Shift right, left, or both
- Wrap around or feed bit
- Fixed or programmable shift

Barrel Shifter Functionality

The Barrel Shifter can be generated for a fixed shift or range of shift, with feedbit shift or rotation in left, right, or both directions. The non-pipelined Barrel Shifter is designed to shift any number of positions at one time. For the pipelined version, it takes log2(MAXSHIFT) clock cycles for the shifted data to appear at the output.

The architecture is based on 2:1 multiplexors.

Table 115 · Functional Description[A] (Standard)

Data	Enable	Clock	Q
M	1	*f*	Qn
M	0	*f*	$M_{shifted}$

A. Assume Aclr is active low, Enable is active high, Clock is rising.

Table 116 · Functional Description[A] (Pipelined)

Data	Aclr	Enable	Clock	Q
X	0	X	X	0's
X	1	0	X	$Qn = M_{shifted} - \log_2(MAXSHIFT)$
m	1	1	*f*	$Q_{n+1} = M_{shifted} - \log_2(MAXSHIFT) + 1$

A. Assume Aclr is active low, Enable is active high, Clock is rising.

Barrel Shifter I/O Description

Table 117 · I/O Description

Name	Type	Required/Optional	Description
Data	IN	Req	Register load input
Aclr	IN	Opt	Asynchronous register reset
Dir	IN	Opt	For selecting Left or Right shift
RFill	IN	Opt	For Right Feed Bit
LFill	IN	Opt	For Left Feed Bit
S0, S1...	IN	Opt	For programmable, depends on Maximum shift
Enable	IN	Opt	Synchronous Parallel load enable
Clock	IN	Req	Clock
Q	OUT	Req	Register output bus

Barrel Shifter Parameter Description

Table 118 · Parameter Description

Parameter	Value	Function
WIDTH	2-63 (Standard) 2-99 (PA Fixed Programmable) 2-63 (PA Range Programmable)	Word length of Data and Q
MAXSHIFT	1 Width-1	Maximum Shift length
CLR_POLARITY	**0** 1 2	Aclr can be active low, active high, or not used
PROG	**Fixed** or Range	For a Fixed or Programmable shift
FILL	**No**, Yes	Wrap around or Feed a bit
DIRECTION	**Right** Left Both	Direction can be Right, Left, or Both

Parameter	Value	Function
EN_POLARITY	0 **1** 2	Enable can be active low or active high
CLK_EDGE	**RISE** FALL	Clock can be rising or falling

Table 119 · Fan-in Control Parameters

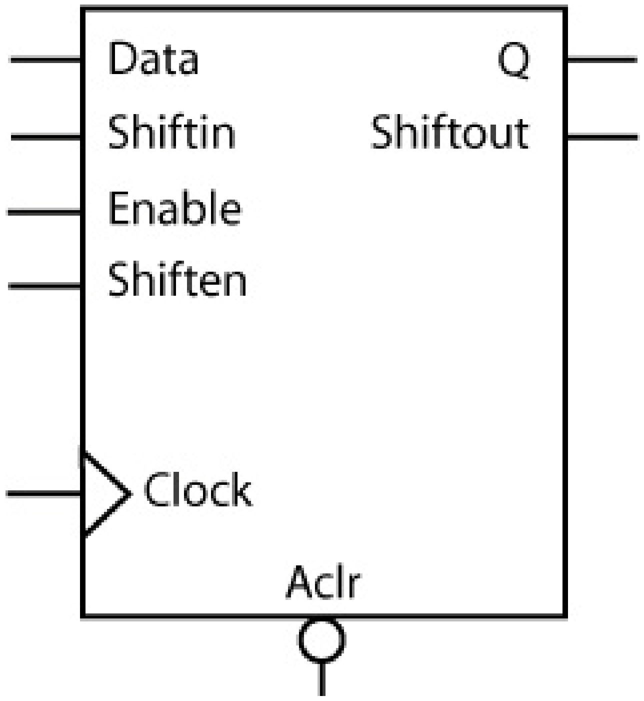
Parameter	Value
CLR_FANIN	AUTO MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	AUTO MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	AUTO MANUAL
CLK_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
SEL0_FANIN	AUTO MANUAL
SEL0_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]

Barrel Shifter Implementation Rules

Table 120 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_DFF	Register category
LPM_HINT	SHIFT, PIPE	Standard or Pipelined

Shift Register



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, RTAX-S, eX, and SX-A

Related Topics

Shift Register Functionality

Shift Register I/O Description

Shift Register Parameter Description

Shift Register Implementation Rules

Key Features

- Parameterized word length
- Asynchronous clear
- Synchronous parallel load
- Behavioral simulation RTL in VHDL and Verilog

Shift Register Functionality

Shift registers have parallel-in/parallel-out (PIPO), parallel-in/serial-out (PISO), serial-in/parallel-out (SIPO) and serial-in/serial-out (SISO) architecture. The registers are WIDTH bits. They are clocked on the rising (RISE) or falling (FALL) edge of the clock signal (CLK_EDGE).

The Clear signal (CLR_POLARITY), active high or low, provides an asynchronous reset of the registers to "000…0". You may choose to not implement the reset function.

Shift registers can be loaded with Data. The Enable signal (EN_POLARITY), active high or low, provides a synchronous load enable operation with respect to the clock signal *Clock*. You may choose to not implement this function. Shift registers are then implemented in a serial-in mode (SIPO or SISO).

Shift registers have a shift enable signal *Shiften* (SHEN_POLARITY) that can be active high or low. When *Shiften* is active, the register is shifted internally. The LSB is loaded with *Shiftin*.

In the current implementation, Enable has priority over Shiften.

Table 121 · Functional Description[A]

Data	**Aclr**	**Enable**	**Shiften**	**Clock**	**Q[B]**	**Shiftout[B]**
X	0	X	X	X	0	0
X	1	X	X	¬	Qn	Qn = [WIDTH-1]
X	1	0	0	*f*	Qn	Qn = [WIDTH-1]
X	1	0	1	*f*	Qn[WIDTH-2:0] && Shif-tin	Qn = [WIDTH-1]
m	1	1	X	*f*	Qn+1 = m	Qn+1 =m [WIDTH-1]

A. Aclr is active low, Enable is active high, Shiften is active high, Clock is rising.

B. For the PISO and SISO implementations, Q is an internal register.

C. For the PIPO and SIPO implementations, Shiftout is not present.

Shift Register I/O Description

Table 122 · I/O Description

Name	**Type**	**Required/Optional**	**Description**
Data	IN	Opt	Register load input data

Name	Type	Required/Optional	Description
Shiftin	IN	Opt	Shift in signal
Aclr	IN	Opt	Asynchronous register reset
Enable	IN	Opt	Synchronous parallel load enable
Shiften	IN	Req	Synchronous register shift enable
Clock	IN	Req	Clock
Q	OUT	Opt	Register output bus
Shiftout	OUT	Opt	Serial output

Shift Register Parameter Description

Table 123 · Parameter Description

Parameter	Family	Value	Function
WIDTH	500K, PA	2-512	Word length of Data and Q
	All others	2-99	
CLR_POLARITY	All	**0** 1 2	Aclr can be active low, active high, or not used
EN_POLARITY	All	0 **1** 2	Enable can be active low or active high
SHEN_POLARITY	All	0 **1**	Shiften can be active low, active high, or not used
CLK_EDGE	All	**RISE** FALL	Clock can be rising or falling

Table 124 · Fan-in Control Parameters

Parameter	Value

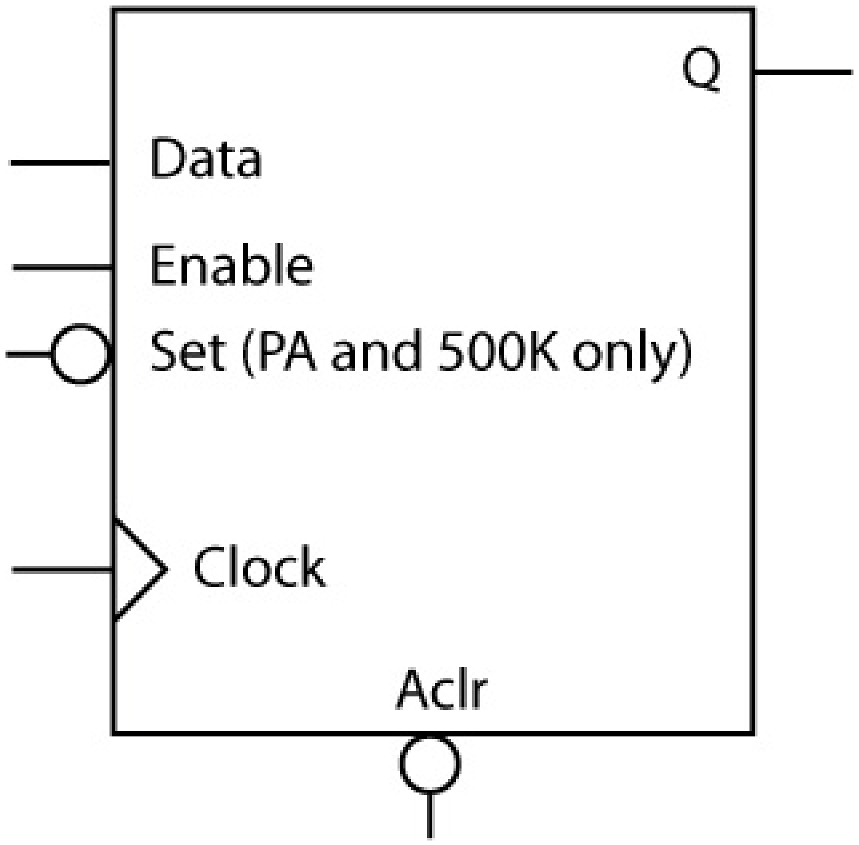
Parameter	Value
CLR_FANIN	**AUTO** MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	**AUTO** MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
SHEN_FANIN	**AUTO** MANUAL
SHEN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	AUTO **MANUAL**
CLK_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]

Shift Register Implementation Rules

Table 125 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_DFF	Register category
LPM_HINT	PIPOS	Parallel-in/Parallel-out shift register
	PISO	Parallel-in/Parallel-out shift register
	SIPO	Serial-in/Parallel-out shift register
	SISO	Serial-in/Serial-out shift register

Storage Register



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC<u>PLUS</u>, ProASIC, Axcelerator, MX, eX, and SX/SX-A

Related Topics

Storage Register Functionality

Storage Register I/O Description

Storage Register Parameter Description

Storage Register Implementation Rules

Key Features

- Parameterized word length
- Asynchronous clear
- Synchronous register parallel load
- Behavioral simulation RTL in VHDL and Verilog

Storage Register Functionality

Storage registers have a parallel-in/parallel-out (PIPO) architecture. The registers are WIDTH bits. They are clocked on the rising (RISE) or falling (FALL) edge of the clock Clock (CLK_EDGE).

The Clear signal (CLR_POLARITY), active high or low, provides an asynchronous reset of the registers to “000…0”. You may choose to not implement the reset function.

The Enable signal (EN_POLARITY), active high or low, provides a synchronous load enable operation with respect to the Clock signal. You can choose to not implement this function. Storage registers are then loaded with a new value every clock cycle.

The Set signal, active high or low, provides an asynchronous set of the registers to "1111...1". You may choose not to implement the Set function.

Table 126 · Functional Description [A]

Data	Aclr	Enable	Clock	Q
X	0	X	X	0's
X	1	X	¬	Qn
X	1	0	*f*	Qn
m	1	1	*f*	Qn+1 = m

A. Assume Aclr is active low, Enable is active high, Clock is rising (edge-triggered).

Storage Register I/O Description

Table 127 · I/O Description

Name	Type	Required/Optional	Description
DATA	IN	Req	Register load input
Aclr	IN	Opt	Asynchronous register reset
Enable	IN	Opt	Synchronous Parallel load enable
Clock	IN	Req	Clock
Q	OUT	Req	Register output bus

Storage Register Parameter Description

Table 128 · Parameter Description

Parameter	Family	Value	Function
WIDTH	500K, PA	1-512	Word length of Data and Q
	All others	1-99	
CLR_POLARITY	All	**0** 1 2	Aclr can be active low, active high, or not used
EN_POLARITY	All	0 **1** 2	Enable can be active low or active high
CLK_EDGE	All	**RISE** FALL	Clock can be rising or falling

Table 129 · Fan-in Control Parameters

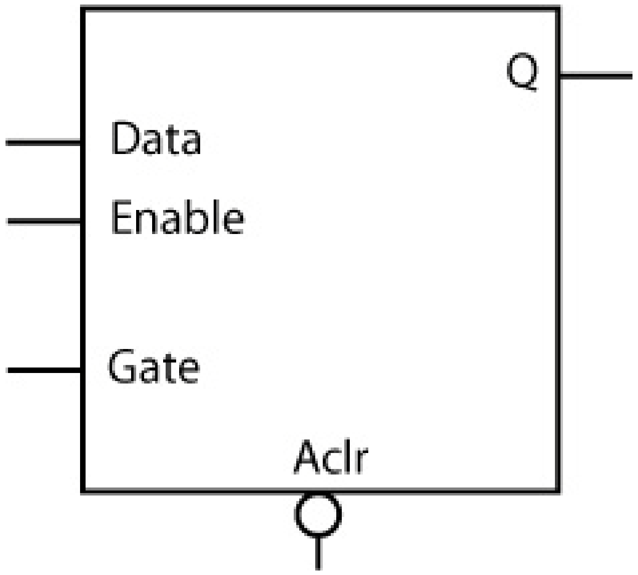
Parameter	Value
CLR_FANIN	**AUTO** MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	**AUTO** MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	AUTO **MANUAL**
CLK_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]

Storage Register Implementation Rules

Table 130 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_DFF	Register category
LPM_HINT	PIPO	Parallel-in/Parallel-out

Storage Latch



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^{PLUS}, ProASIC, Axcelerator, MX, eX, and SX/SX-A

Related Topics

Storage Latch Functionality

Storage Latch I/O Description

Storage Latch Parameter Description

Storage Latch Implementation Rules

Key Features

- Parameterized word length
- Asynchronous clear
- Synchronous latch enable
- Behavioral simulation RTL in VHDL and Verilog

Storage Latch Functionality

Latches have a parallel-in/parallel-out architecture (PIPO). The latches are WIDTH bits. The latches are gated on the active high (HIGH) or low (LOW) state of the gate *Gate (*GATE_POLARITY).

The *Clear* signal (CLR_POLARITY), when active high or low, provides an asynchronous reset of the latch to "000…0". You may choose to not implement this function.

The *Enable* signal (EN_POLARITY), when active high or low, provides a synchronous latch enable operation with respect to the gate *Gate*. You may choose to not implement this function. Latches are then loaded with a new value when both Enable and *Gate* are active.

Table 131 · Functional Description[A]

Data	Aclr	Enable	Gate	Q
X	0	X	X	0's
X	1	X	0	Qn
X	1	0	1	Qn
m	1	1	1	Qn+1 = m

A. Aclr is active low, Enable is active high, Shiften is active high, Clock is rising.

Storage Latch I/O Description

Table 132 · I/O Description

Name	Type	Required/Optional	Description
Data	IN	Req	Latch load input
Aclr	IN	Opt	Asynchronous latch reset
Enable	IN	Opt	Synchronous parallel latch enable
Gate	IN	Req	Gate input
Q	OUT	Req	Latch output bus

Storage Latch Parameter Description

Table 133 · Parameter Description

Parameter	Family	Value	Function

Parameter	Family	Value	Function
WIDTH	500K, PA	1-512	Word length of Data and Q
	All others	1-99	
CLR_POLARITY	All	**0** 1 2	Aclr can be active low, active high, or not used
EN_POLARITY	All	0 **1** 2	Enable can be active low or active high
CLK_EDGE	All	**RISE** FALL	Clock can be rising or falling

Table 134 · Fan-in Control Parameters

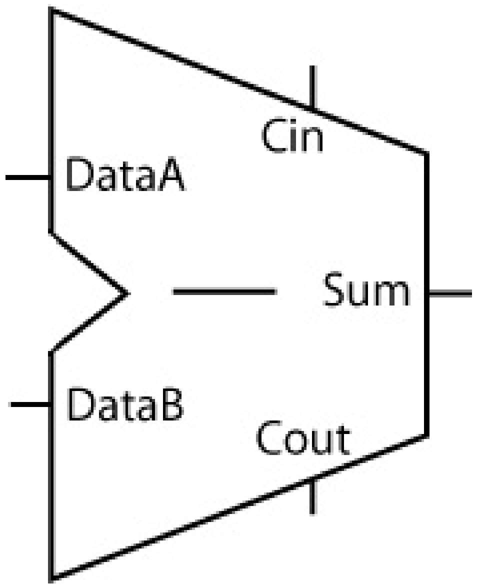
Parameter	Value
CLR_FANIN	**AUTO** MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	**AUTO** MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
GATE_FANIN	AUTO **MANUAL**
GATE_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]

Storage Latch Implementation Rules

Table 135 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_LATCH	Latch category
LPM_HINT	N/A	Not needed

Subtractor



Supported Families

IGLOO, ProASIC3, SmartFusion, Fusion, ProASIC^PLUS, ProASIC, Axcelerator, MX, eX, and SX/SX-A

Related Topics

Subtractor Functionality

Subtractor I/O Description

Subtractor Parameter Description

Subtractor Implementation Rules

Key Features

- Parameterized word length
- Optional carry-in and carry-out signals
- Multiple gate-level implementations (speed/area tradeoffs)
- Behavioral simulation RTL in VHDL and Verilog

Subtractor Functionality

Table 136 · Functional Description

DataA	DataB	Sum	Cout[A]
m[width-1 : 0]	n[width-1 : 0]	(m - n - Cin) [width-1 : 0]	(m - n - Cin)[width]

A. Cin and Cout are assumed to be active high

Subtractor I/O Description

Table 137 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DataA	WIDTH	Input	Req.	Input Data
DataB	WIDTH	Input	Req.	Input Data
Cin	1	Input	Opt.	Carry-in
Sum	WIDTH	Output	Req.	Sum
Cout	1	Output	Opt.	Carry-out

Subtractor Parameter Description

Table 138 · Parameter Description

Parameter	Family	Value	Function
WIDTH[A]	ProASICPLUS	2-128	Word length of DataA, DataB and Sum
	Axcelerator	2-156	
	All others	2-32	
MAXFANOUT	500K, PA	0	Automatic choice
		2-16	Manual setting of Max. Fanout
CI_POLARITY	ALL	0 1 **2**	Carry-in polarity (active low, active high, and not used)
CO_POLARITY	ALL	0 1 **2**	Carry-out polarity (active low, active high,and not used)

A. The Brent-Kung Subtractor extends the ranges from 32 to 128 bit for SX, SX-A.

Subtractor Implementation Rules

Table 139 · Implementation Rules

Parameter	Family	Value	Description
LPMTYPE	All	LPM_ADD_SUB	Subtractor category
LPM_HINT	ProASICPLUS, ProASIC	SKSUB	Sklansky model
		FBKSUB	Fast Brent-Kung model
		BKSUB	Compact Brent-Kung model
	ALL	FSUB[A]	Very fast carry select model
	ALL	RIPSUB[A]	Ripple carry model
LPMTYPE	Axcelerator	LPM_FC_ADD_SUB	Fast carry chain Subtractor category
LPM_HINT		FC_FSUB	Fast carry chain select model
LPM_HINT		FC_RIPSUB	Fast carry chain ripple carry model

A. FSUB and MFSUB are not recommended for ProASIC3/E or ProASIC$^{\underline{PLUS}}$ devices.

Clock and Management

Fusion Dynamic CCC

Related Topics

Fusion Dynamic CCC Functionality

Fusion Dynamic CCC I/O Description

Key Features

The only difference between the Fusion Dynamic CCC and Fusion Static PLL is the availability of the dynamic shift register signals that enable a dynamic reconfiguration of the PLL. The Dynamic CCC (clock conditioning core) enables you to change the CCC configuration by shifting it in via a serial interface. You have a fixed default configuration and the two configurations can be interchanged dynamically.

The Dynamic CCC for Fusion contains a PLL core, delay lines, clock multipliers/dividers, PLL reset generator (you have no control over the reset), global pads, and all circuitry for the selection and interconnection of the "global" pads to the global network. The PLL Core consists of a Phase Detector, L.P. Filter, and a 4-Phase VCO, and the following:

- RC Oscillator Clock Source - If you choose RC Oscillator as the clock source the input frequency is fixed at 100MHz. The divide-by-half feature is available if you bypass the PLL for the primary output.
- Divide by half behavior - Available if clock source is RC Oscillator and PLL is bypassed for the given output (A, B, C). When activated, the output divider (U, V, or W) gets divided by 2. Thus if the divider is 3, divide-by-half ON makes the divider 1.5.
- Crystal oscillator clock source - no special configuration options are available if you use the crystal oscillator as your clock source. Select this option if you are using a crystal oscillator as your clock source.
- Availability of output dividers in bypass mode - If you bypass the PLL in the primary output, you can specify an output frequency that is some divisible of the input frequency. The dividing factor must be an integer between 1 and 32.

The Static PLL performs the following basic functions:

- Clock phase adjustment
- Clock delay minimization
- Clock frequency synthesis

In addition it also:

- Enables access from the global pads to the global network and the PLL block

- Permits the three global lines on each side of the chip to be driven either by the global pads, core, and/or the outputs from the PLL block
- Enables access from PLL to the core

The block contains several programmable dividers, each of them providing division factors 1, 2, 3, 4……k (where k depends on the number of bits used for the division selection). Overall, you can define a wide range of multiplication and division factors, constrained only by the PLL frequency limits, according to:

m/(n*u)

m/(n*v)

m/(n*w)

The clock conditioning circuit block performs a positive / negative clock delay operation in increments of 200 ps, of up to 6.735 ns (at 1.5V, 25C, typical process) before or after the positive clock edge of the incoming reference clock. Furthermore, the system allows for the selection of one of four clock phases of fout, at 0, 90, 180, and 270 degrees.

A “Lock” signal is provided to indicate that the PLL has locked on to the incoming signal. A “Power-down” signal switches off the PLL block when it is not used.

Fusion Dynamic CCC Functionality

The input clock, f_{in}, is first passed through the adjustable divider (FINDIV) prior to application to the PLL core phase detector's PLLFIN input.

The feedback signal, to which fin is compared, can be selected from several sources, giving the Static PLL its flexibility. All sources of the feedback signal can be divided by 1, 2, 3, …128 in divider FBDIV. This has the effect of multiplying the input signal. The source signals are:

- The VCO output signal, with 0o phase shift and zero additional time delay
- A delayed version of the VCO output, in selectable increments of 200 ps, up to 6.735 ns
- An external feedback signal from I/O

Each of the above feedback source signals can be further delayed by a fixed amount designed to emulate the delay through the chip’s clock tree. This allows for clock-line de-skewing operations.

When the loop has acquired lock, the Lock Detect signal will be asserted. This signal will be available to the logic core, via the output port LOCK.

Once locked, the various output combinations will be available to the Global lines.

In addition, the Dynamic CCC prints out all the values of the configuration pins in a report. You can use these to specify the bitstream that can be shifted in via the shift register.

Configuring the Fusion Static PLL

The Fusion Static PLL includes the following features (shown in the figure below):

- An option to choose the source of the input clock as one of the following:
  - Hardwired I/O driven
  - External I/O driven
  - Core Logic driven
  - Crystal oscillator
  - RC oscillator
- The option to bypass the PLL for the primary output.
- Configuration selections for frequency, delay, and phase.
- Secondary 1 and Secondary 2 inputs available on the Secondary 1 and Secondary 2 outputs. The Secondary 1 and 2 inputs are available only in bypass mode.

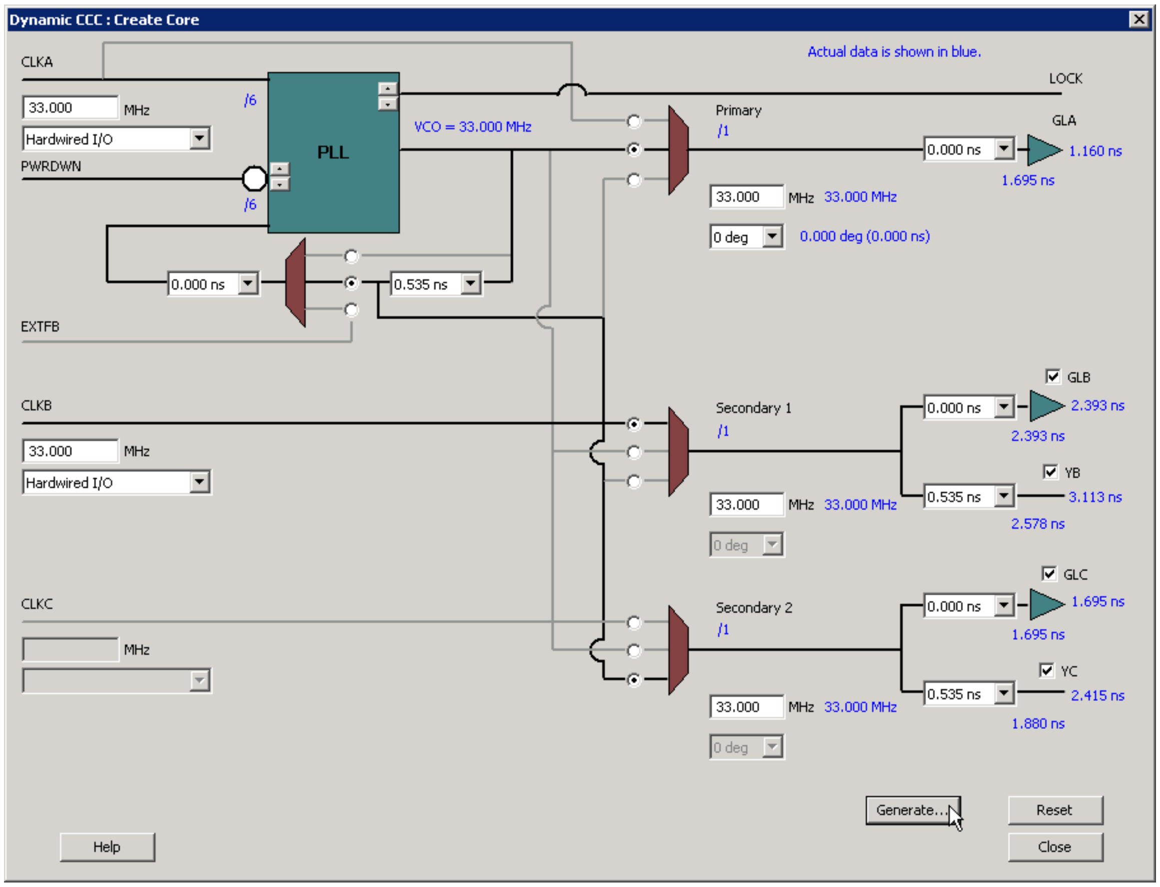


Figure 11 · Fusion Dynamic CCC Screen

After you open select the Static PLL from the Catalog, you must configure it. To do so:

1. Select your output. A total of five outputs can be obtained from the Static PLL. Select the check box next to each required output to select it.
    - GLA is always selected
    - GLB and YB have the same output frequency. They can be delayed by different amounts by setting the individual delays. GLB drives a Global while YB drives a core net. Using only YB also burns the global driver for GLB. However, the global rib is available.
    - GLC and YC have the same output frequency. They can be delayed by different amounts by setting the individual delays. GLB drives a Global while YB drives a core net. Using only YC also burns the global driver for GLB. However, the global rib is available.
    - The input signal CLKA is the reference clock for all five outputs
2. Specify your Internal Feedback. The source of the feedback signals will be the VCO output signal, with 0 degree phase shift and zero additional time delay. (top Selection on the Feedback MUX) or A delayed version of the VCO output, in selectable increments of 200 ps, up to 6.735 ns. This delay advances the feedback clock, thereby advancing all outputs by the delay value specified for the feedback delay element (middle selection of the Feedback MUX).
3. Set your Fixed System Delay. By choosing the non-zero value for this delay, the feedback source signal can be further delayed by a fixed amount of mask delay designed to emulate the delay through the chip's clock tree. This allows for clock-line de-skewing operations.
4. Specify your input clock:
    - Input Clock Frequency between 1.5 – 350 MHz
    - Input Clock Source as one of the following: Driven by the hardwired I/O; Driven by an external I/O from a different I/O location; Driven by Core Logic
5. Specify the primary output beginning with the source of the input clock (shown below in *italics*)
    - *Output bypassing the PLL (top selection of the GLA MUX)*. In this case, VCO phase shift and output frequency selection are not available. Output frequency is the same as input frequency in this case.
    - *Output directly from the VCO (middle selection of the GLA MUX)*. The phase shift of 0, 90, 180, or 270 is available in this case.
    - *Delayed version of the zero phase shift output from the VCO*. Phase-shift selection is unavailable for this (bottom selection of the GLA MUX). This output can be used for two purposes: a) to use the feedback delay as an additional delay on the output if feedback advance has not been specified (top and bottom selections of the feedback MUX); b) to compensate for the feedback advance for this particular output if feedback advance has been specified (middle selection of the feedback MUX).

- Output frequency (1.5 – 350 MHz)
- VCO Phase-Shift (one of 0, 90, 180, or 270 degrees); the phase shift is out of the VCO. The phase shift will be impacted by the value of the divider after the VCO.
- An optional Extra Output Delay, in selectable increments of 200 ps, up to 6.735 ns.

6. Specify Secondary1 and Secondary2 Outputs. Select the source of the output clock (shown below in *italics*):
    - *Output directly from the VCO (top selection of the GLB/GLC MUX)* - The phase shift of 0, 90, 180, or 270 is available in this case.
    - *Delayed version of the zero phase shift output from the VCO* - Phase-shift selection is unavailable for this (bottom selection of the GLB/GLC MUX). This output can be used for two purposes: a) to use the feedback delay as an additional delay on the output if feedback advance has not been specified (top and bottom selections of the feedback MUX); b) to compensate for the feedback advance for this particular output if feedback advance has been specified (middle selection of the feedback MUX).
    - Set your Output frequency (1.5 – 350 MHz)
    - VCO Phase-Shift (one of 0, 90, 180, or 270 degrees); the phase shift is out of the VCO. The phase shift will be impacted by the value of the divider after the individual optional Extra Output Delay for each of the Global and Core outputs, in selectable increments of 200 ps, up to 6.735 ns.

Fusion Static PLL Core Restrictions

After you make all your selections, the software generates a core with your configurations. However, there are a number of restrictions in the possible values for the input and output frequencies. They are:

- Input to the PLL must be between 1.5 and 350 MHz
- Output from the PLL must be between 1.5 and 350 MHz
- The reference input to the PLL core (fin/n) must be between 1.5 and 5.5 MHz. The PLL Core output must be between 24 and 350 (fin *m/n) .

If the software cannot generate the frequency you requested, it tries to generate a frequency that is as close as possible after it satisfies all the above conditions. The software prints a message in the log file indicating the actual PLL output frequency. If more than one output is specified, the software tries to find the multiplication and division factors with the smallest total error among all the outputs.

Total Delays and Input Delays

The software prints out the total delays of the selected outputs in the Log window after feedback delay, feedback advance, system delay, and extra output delay are evaluated.

Total Delay on an Output = -feedback advance – de-skew system delay + feedback delay + extra output delay + intrinsic delay.

Fusion Dynamic CCC I/O Description

Table 140 · I/O Description

Name	Type	Required/Optional	Description
GLA	OUT	Req	Primary clock output
CLKA	IN	Req	Reference clock
POWERDOWN	IN	Req	Power down signal. A low on this turns off the PLL
LOCK	OUT	Req	Pll lock
EXTFB	IN	Opt	External feedback
GLB	OUT	Opt	Global output for Secondary1 clock
YB	OUT	Opt	Core output for Secondary1 clock
GLC	OUT	Opt	Global output for Secondary2 clock
YC	OUT	Opt	Core output for Secondary2 clock

IGLOO and ProASIC3 Dynamic CCC Summary

The only difference between the IGLOOe and ProASIC3E Dynamic CCC and IGLOO and ProASIC3 Static PLL is the availability of the dynamic shift register signals that enable a dynamic reconfiguration of the PLL.

Related Topics

IGLOO and ProASIC3 Dynamic CCC Functionality

IGLOO and ProASIC3 Dynamic CCC I/O Description

IGLOO and ProASIC3 Dynamic CCC Implementation Rules / Timing Diagrams

The Dynamic CCC (clock conditioning core) enables you to change the CCC configuration by shifting it in via a serial interface. You have a fixed default configuration and the two configurations can be interchanged dynamically.

The IGLOOe and ProASIC3E Clock Conditioning Circuit (CCC) contains a PLL core, delay lines, clock multipliers/dividers, PLL reset generator (you have no control over the reset), global pads, and all circuitry for the selection and interconnection of the “global” pads to the global network. The PLL Core consists of a Phase Detector, L.P. Filter, and a 4-Phase VCO.

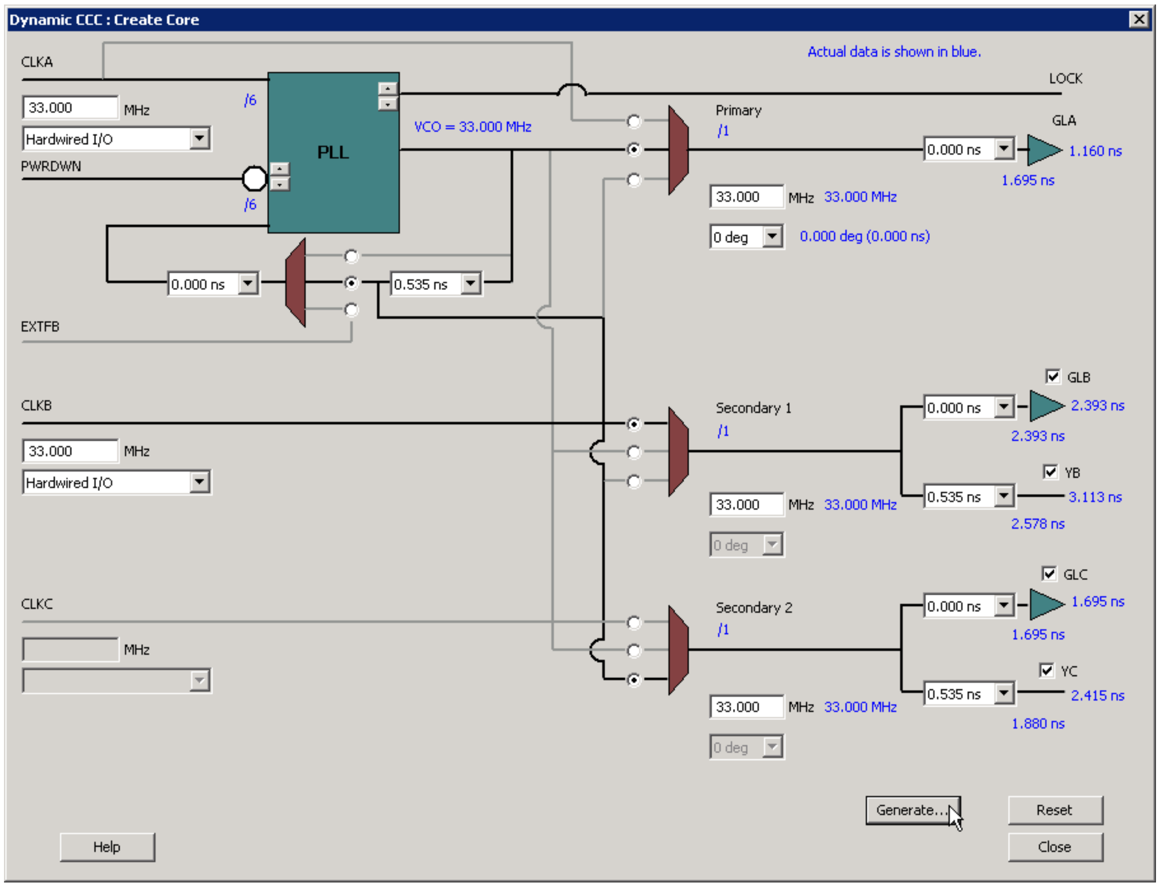
The clock conditioning circuit block is fully configurable, either via flash configuration bits (set in the programming bits stream) or through a simple asynchronous interface dynamically accessible from customer signals inside the device to permit parameter changes (such as PLL divide ratios) during device operation.

The clock conditioning circuit performs the following basic functions:

- Clock phase adjustment
- Clock delay minimization
- Clock frequency synthesis

In addition to all the functionality available in the IGLOO and ProASIC3 Static PLL, the Dynamic CCC prints out all the values of the configuration pins in a report. You can use these to specify the bitstream that can be shifted in via the shift register.

The Dynamic CCC is configured exactly the same way as the IGLOO and ProASIC3 Static PLL. The only differences are the Secondary 1 and Secondary 2 inputs available on the Secondary 1 and Secondary 2 outputs. The Secondary 1 and 2 inputs are available only in bypass mode. The Dynamic CCC is shown in the figure below.



IGLOO and ProASIC3 Dynamic CCC Functionality

Configuring Control Bits in the Dynamic CCC

The software prints out all the values of the configuration pins in a report. You can use these to specify the bitstream that can be shifted in via the shift register.

You can use the “control bits” to select the ratios used in the various dividers, the signals selected by the multiplexors and “power-down” control for the CCC block. The signals applied to the control inputs can come from one of two sources:

- Flash configuration bits set by the software or by you. These bits are set in the bitstream file and provide the default state and mode of the PLL core.
- Synchronous serial interface with access to and from the logic core. This method is very powerful, since it allows core driven dynamic PLL reconfiguration. The reconfiguration does not unlock the PLL as long as it does not change the state of the input divider or feedback elements. (This interface also includes an asynchronous “update” latch for the configuration inputs to the multiplexer.) The input and output signals in this mode are listed in table Table 10-6 on page 120. SUPDATE must be low during any clock cycle where SSHIFT is active.

A total of 81 configuration bits must be specified to change the configuration. When you use the software to define the configuration that will be shifted-in via the serial interface, it prints out the values of the 81 configuration bits.

The combiner infers STATASEL, STATBSEL, STATCSEL.DYNASEL, DYNBSEL, DYNCSEL and RESET_ENABLE.

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 <77:79> are don't cares. The core configurator software defines 74 bits. Six more bits are not available until after layout and are defined in the post-layout report. The last bit is RESETENABLE; it is always 1.

The table below defines all the configuration bits required to enter a new configuration.

NAME	FUNCTION
FINDIV<6:0>	7-BIT INPUT DIVIDER (/N)
FBDIV<6:0>	7-BIT FEEDBACK DIVIDER (/M)
OADIV<4:0>	5-BIT OUTPUT DIVIDER (/U)
OBDIV<4:0>	5-BIT OUTPUT DIVIDER (/V)
OCDIV<4:0>	5-BIT OUTPUT DIVIDER (/W)
OAMUX<2:0>	3-BIT POST-PLL MUXA (BEFORE DIVIDER /U)
OBMUX<2:0>	3-BIT POST-PLL MUXB (BEFORE DIVIDER /V)
OCMUX<2:0>	3-BIT POST-PLL MUXC (BEFORE DIVIDER /W)
FBSEL<1:0>	2-BIT PLL FEEDBACK MUX
FBDLY<4:0>	FEEDBACK DELAY
XDLYSEL	1-BIT PLL FEEDBACK MUX
DLYHCA<4:0>	DELAY ON GLOBAL A
DLYHCB<4:0>	DELAY ON GLOBAL B
DLYHCC<4:0>	DELAY ON GLOBAL C
DLYB<4:0>	DELAY ON YB

NAME	FUNCTION
DLYC<4:0>	DELAY ON YC
STATASEL	MUX SELECT ON INPUT A
STATBSEL	MUX SELECT ON INPUT B
STATCSEL	MUX SELECT ON INPUT C
VCOSEL<2:0>	3-BIT VCO GEAR CONTROL (4 FREQUENCY RANGES)
DYNASEL	DYNAMIC SELECT ON INPUT B
DYNBSEL	DYNAMIC SELECT ON INPUT A
DYNCSEL	DYNAMIC SELECT ON INPUT C
RESET_ENABLE	

IGLOO and ProASIC3 Dynamic CCC I/O Description

Table 141 · I/O Description

Name	Size	Type	Required/ Optional	Function
GLA	1	Output	Req	Primary clock output
CLKA	1	Input	Req	Reference clock
POWERDOWN	1	Input	Req	Power Down Signal. A low on this signal turns off the Dynamic CCC
LOCK	1	Output	Req	Dynamic CCC lock
SDOUT	1	Output	Req	Serial interface shift register output
SCLK	1	Input	Req	Shift clock
SSHIFT	1	Input	Opt	Serial Shift enable
SDIN	1	Input	Opt	Serial Data in for Dynamic CCC

Name	Size	Type	Required/ Optional	Function
				configuration bits
SUPDATE	1	Input	Opt	Serial update
MODE	1	Input	Opt	Dynamic or Static mode indicator. A Low on this signal indicates static mode and a High indicates dynamic.
EXTFB	1	Input	Opt	External feedback
CLKB	1	Input	Opt	Input clock for Secondary 1 clock; valid only in bypass mode.
GLB	1	Output	Opt	Global Output for Secondary1 Clock
YB	1	Output	Opt	Core Output for Secondary1 Clock
CLKC	1	Input	Opt	Input clock for Secondary 2 clock.; valid only in bypass mode.
GLC	1	Output	Opt	Global Output for Secondary2 Clock
YC	1	Output	Opt	Core Output for Secondary2 Clock

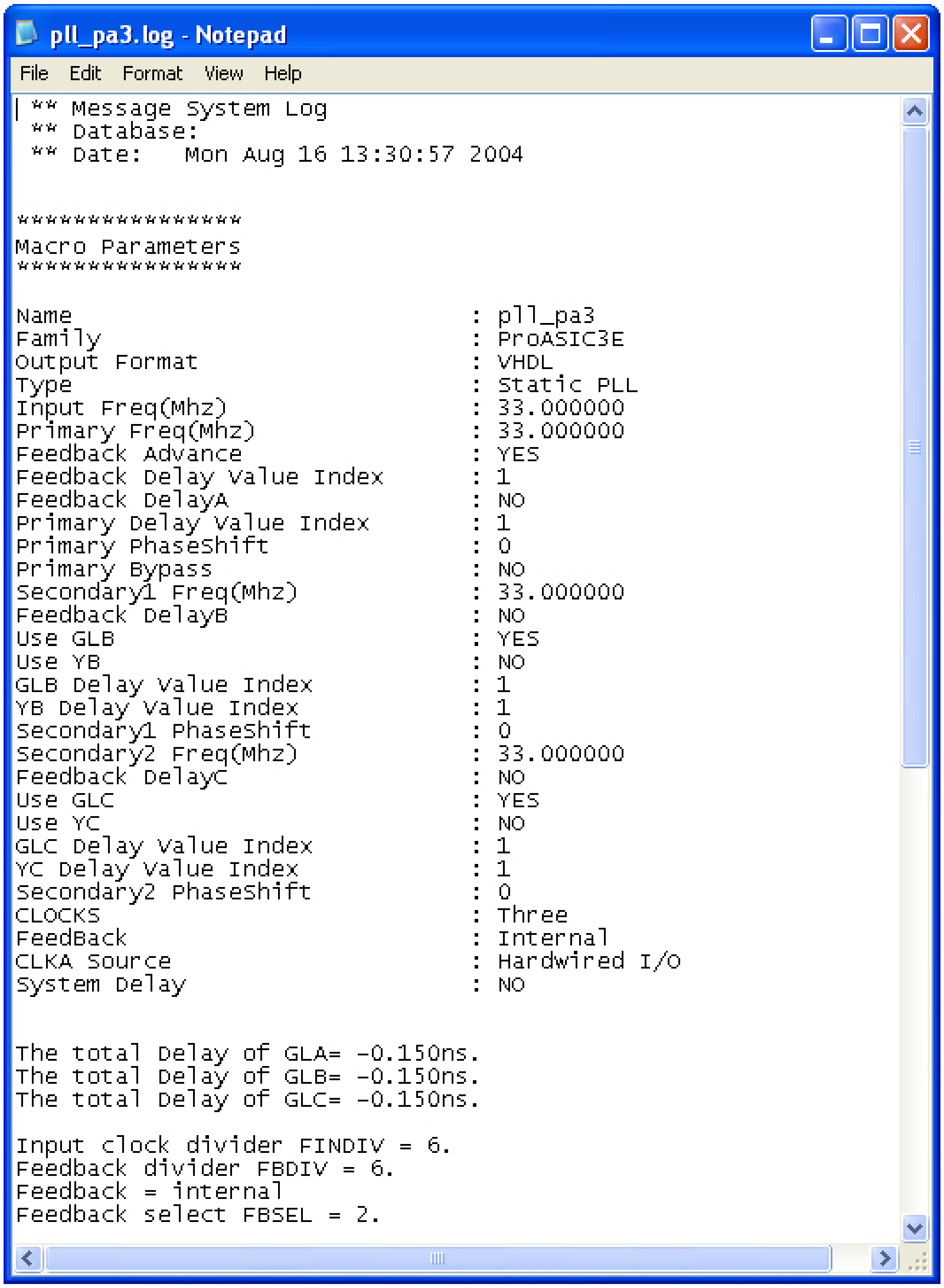
IGLOO and ProASIC3 Dynamic CCC Implementation Rules / Timing Diagrams

After you make all your selections, the software generates a core with your configurations. However, there are a number of restrictions in the possible values for the input and output frequencies. They are:

- Input to the clock conditioning core (CCC) must be between 1.5 and 350 MHz
- Output from the CCC must be between 1.5 and 350 MHz
- The reference input to the PLL core (f_{in}/n) must be between 1.5 and 5.5 MHz. The PLL Core output must be between 24 and 350 (f_{in} *m/n)

Your requested PLL values are not possible in all cases, because of the VCO input, output frequency limitations, available divider ranges and inter-dependencies between the multiple outputs. In such cases, the core configurator tries to generate a value that is as close as possible to the value you requested. The actual values that the configurator can achieve are shown on the screen (in blue). If you hit generate, the core is generated with the actual values rather than the specified values. The actual values are also included in the log file for future reference.

Here is a sample of the Log file with all the information.



```
 ** Message System Log
 ** Database:
 ** Date:   Mon Aug 16 13:30:57 2004


****************
Macro Parameters
****************

Name                             : pll_pa3
Family                           : ProASIC3E
Output Format                    : VHDL
Type                             : Static PLL
Input Freq(Mhz)                  : 33.000000
Primary Freq(Mhz)                : 33.000000
Feedback Advance                 : YES
Feedback Delay Value Index       : 1
Feedback DelayA                  : NO
Primary Delay Value Index        : 1
Primary PhaseShift               : 0
Primary Bypass                   : NO
Secondary1 Freq(Mhz)             : 33.000000
Feedback DelayB                  : NO
Use GLB                          : YES
Use YB                           : NO
GLB Delay Value Index            : 1
YB Delay Value Index             : 1
Secondary1 PhaseShift            : 0
Secondary2 Freq(Mhz)             : 33.000000
Feedback DelayC                  : NO
Use GLC                          : YES
Use YC                           : NO
GLC Delay Value Index            : 1
YC Delay Value Index             : 1
Secondary2 PhaseShift            : 0
CLOCKS                           : Three
FeedBack                         : Internal
CLKA Source                      : Hardwired I/O
System Delay                     : NO


The total Delay of GLA= -0.150ns.
The total Delay of GLB= -0.150ns.
The total Delay of GLC= -0.150ns.

Input clock divider FINDIV = 6.
Feedback divider FBDIV = 6.
Feedback = internal
Feedback select FBSEL = 2.
```

If more than one output is specified, the software tries to find the multiplication and division factors with the smallest total error among all the outputs.

Total Delays

The software prints out the total delays of the selected outputs after feedback delay, feedback advance, system delay, and extra output delay are taken into consideration.

Total Delay on an Output = -feedback advance – de-skew system delay + feedback delay + extra output delay + intrinsic delay

Input Delays

The delay between the input of the PLL and a given output can be calculated by the following equation.

Total Delay = Intrinsic delay +/- feedback delay – mask delay + phase delay + output delay

Intrinsic delay is the total delay of all the muxes and divider elements in the path. This is a fixed value for a given connectivity in a configuration. This delay varies based on the mux selection, frequency values and phase-shifts. Changing the delay element values has no impact on the intrinsic delay.

Feedback delay can be both a positive and a negative delay based on how it is configured.

Mask delay is a fixed system delay to emulate the skew of the CCC, such that the output can be deskewed by selecting this delay.

Output delay is the programmable delay independently selectable for each output.

Phase delay is the shift caused in the output with respect to the input when the VCO output is shifted by one of the 4 possible values of 0, 90, 180 or 270 degrees. This is a function of both input and output frequencies.

The delay calculation is executed using the same values for the software, the Simulation model and Timer such that, for typical, -2 parts under normal operating conditions, these numbers are identical. This enables you to fine-tune your delays by only adjusting the programmable output / feedback delays.

Delayed Clock Summary

When resources are available, the Delay element of the Secondary1 and Secondary2 Global outputs of the CCC can be configured independent of the PLL. The delayed clock is a simple CLKMUX with some additional delay.

Select the Global output delay, in steps of 160 ps, for the Output. Select the Input clock source appropriate for your design. If you are using a Fusion device, the Input clock source list includes options for a Crystal Oscillator and an RC Oscillator.

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Divided and Delayed Clock

Use this core to divide down a clock and, if necessary, delay it by a given amount.

This core has two outputs, one global output and an additional output that drives the internal logic. They are equivalent to the GL and Y outputs of a PLL. The divider ranges from 1-32.

Supported Families

Fusion

No-Glitch MUX (NGMUX)

There is no configuration required. You can use this core for switching between clocks without glitches.

Supported Families

Fusion

Axcelerator PLL

Related Topics

[Axcelerator PLL I/O Description]

[Axcelerator PLL Parameter Description]

Key Features

- Clock Delay Minimization
- Clock Frequency Synthesis
- Programmable delay lines for clock delay adjustment
- 6-bit divider in the feedback path for clock multiplication
- 6-bit divider in one of the output paths for clock division
- Cascadable up to two PLLs

The Axcelerator PLL has three main features. They are:

Clock Delay Minimization

In this mode the PLL can perform either a positive or negative clock delay operation of up to 3.75 ns in increments of 250 ps before or after the clock edge of the incoming reference clock. The value of the delay is programmable via the five bits of the DelayLine bus.

Clock Frequency Synthesis

The multiplier and divider can be used together to synthesize a wide range of output frequencies from the reference clock. Input frequencies are allowed to be in the range of 14 MHz to 200 MHz. Multiplication and division factors are integers in the range of 1 to 64. The maximum allowable output frequency is 1 GHz. The output duty cycle is fixed at 50/50.

Cascading Blocks

The device supports cascading of up to 2 PLLs.

The Axcelerator family provides eight PLLs, four on the north side and four on the south side of the device. The outputs of the north-side PLLs can be connected to either hard-wired clock networks or regular nets. The outputs of the south-side PLLs can be connected to either routed clock networks or regular nets. The Axcelerator family PLLs have many outstanding features, including the following:

- PLLs can multiply and/or divide the reference clock frequency by factors ranging from 1 to 64. As a result, there are many available output frequencies for each PLL, based on the input frequency. The software automatically calculates the values of the multiplier and divider based on the Input and Output frequencies specified. If the exact value cannot be achieved, the core configurator generates the output frequency that is the closest possible to the required value.
- PLLs are capable of inserting programmable delays on the REF Clock from –3.75 ns to +3.75 ns with the steps of 250ps. The delay is programmed either statically or dynamically. Dynamic programming means that you can change the delay value during the operation when the device is functional. If you select the dynamic delay, then the 5-bit Delay Line port is added to the generated code and accessible to you.

Refclk is the reference input to the PLL. The frequency of Refclk can vary from 14 MHz to 200 MHz. The reference can be supplied from a dedicated pad or an internal net.

You can select to have an internal or external feedback. Selecting an external feedback adds a port (named FB) to the PLL block, through which the external feedback is passed into the PLL and the internal feedback is blocked.

Clk(freq) are the output signals from the PLL. The CLK(primary) is defined as refclk * i/j where i is the multiplier and j is the divider. CLK(secondary) is defined as refclk * i.

Cascading

Cascading is an option that helps you generate a wider range of output frequencies. If cascading is set to **No** and the output frequency is chosen as a value that cannot be achieved by fREF * i/j, then the PLL will try to set i and j in order to reach to the closest vicinity of the desired frequency. If cascading is set to **Yes**, then for the conditions in which the desired frequency is unattainable by a single PLL, another PLL will be cascaded to the first PLL and then the final output frequency is:

$$f_{out} = f_{REF} \times \left(\frac{i1}{j1}\right) \times \left(\frac{i2}{j2}\right)$$

In cascading PLLs, the input frequency of each PLL should remain in the range of 14 MHz to 200 MHz.

You must specify the desired output frequencies and the networks that the outputs should drive for the PLL outputs CLK1 and CLK2. Note that if cascading is disabled, the CLK2 frequency can only be a multiple of the reference frequency. As mentioned earlier, if the selected values for output frequencies cannot be achieved, they will be set to the closest possible frequency.

For each output, there are three routing resources. Hard-wired is the HCLK network which reaches to the clock input of R-cells. Selecting a hard-wired output for the PLL implies that the PLL should be located at the north side of the device. If one of the outputs is connected to hard-wired global network, the routed clock network cannot be chosen as the second output because the routed clock network is only accessible by the PLLs on the south side. The software helps you select the output type by keeping the possible outputs active and disabling the illegal combinations.

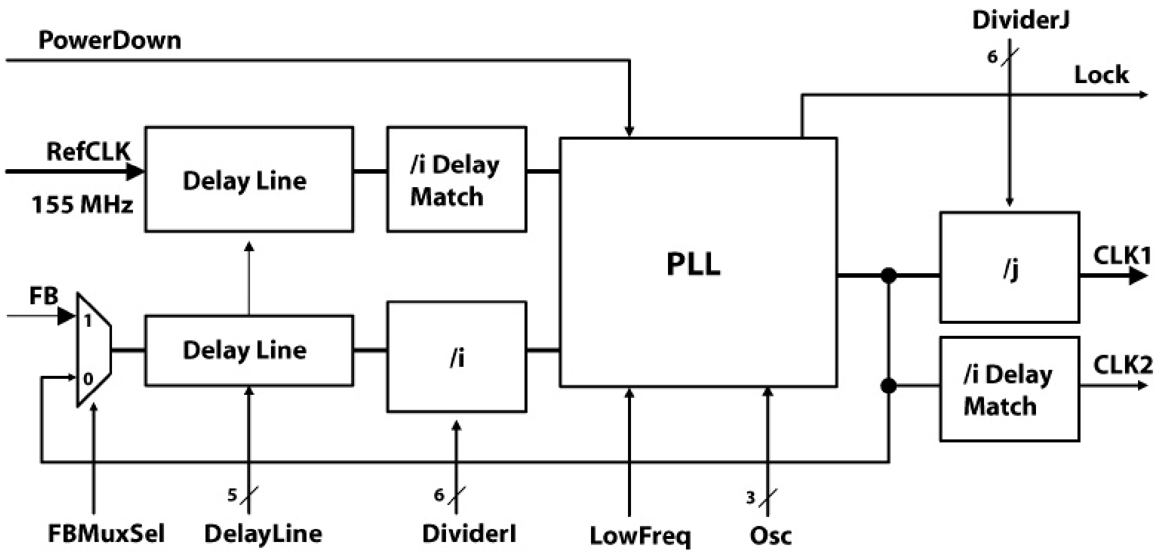
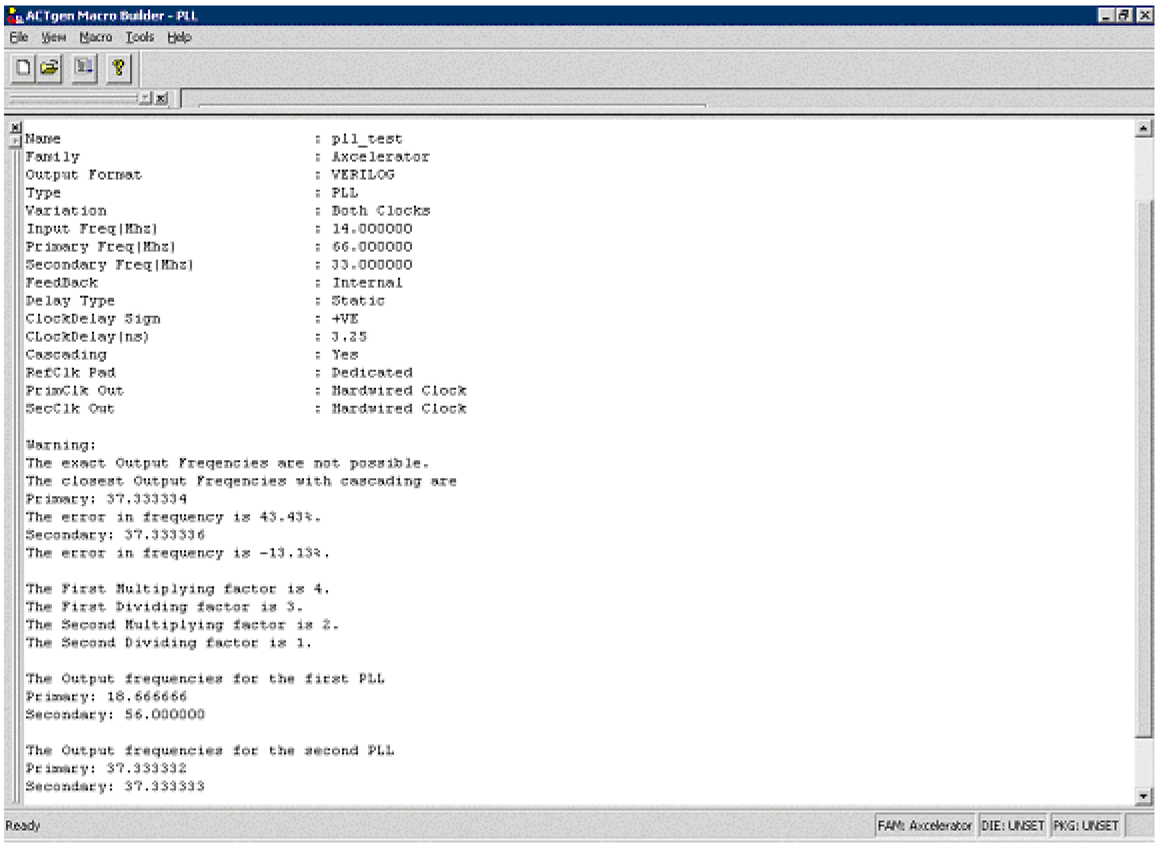


Figure 12 · Basic Axcelerator PLL Architecture

Message Log

All the parameters specified by the user and the values calculated by the core configurator software are saved in the <design>.log file (below).



```
Name                          : pll_test
Family                        : Axcelerator
Output Format                 : VERILOG
Type                          : PLL
Variation                     : Both Clocks
Input Freq(Mhz)               : 14.000000
Primary Freq(Mhz)             : 66.000000
Secondary Freq(Mhz)           : 33.000000
FeedBack                      : Internal
Delay Type                    : Static
ClockDelay Sign               : +VE
CLockDelay(ns)                : 3.25
Cascading                     : Yes
RefClk Pad                    : Dedicated
PrimClk Out                   : Hardwired Clock
SecClk Out                    : Hardwired Clock

Warning:
The exact Output Freqencies are not possible.
The closest Output Freqencies with cascading are
Primary: 37.333334
The error in frequency is 43.43%.
Secondary: 37.333336
The error in frequency is -13.13%.

The First Multiplying factor is 4.
The First Dividing factor is 3.
The Second Multiplying factor is 2.
The Second Dividing factor is 1.

The Output frequencies for the first PLL
Primary: 18.666666
Secondary: 56.000000

The Output frequencies for the second PLL
Primary: 37.333332
Secondary: 37.333333
```

For more detailed information on the various features of the Axcelerator PLL, please refer to the Axcelerator Family PLL and Clock Management application note at http://www.actel.com.

Axcelerator PLL I/O Description

Table 142 · I/O Description

Name	Size	Type	Req/Opt	Function
RefClk	1	Input	Req.	Reference Clock
PWRDN	1	Input	Req.	Power Down
Lock	1	Output	Req.	PLL Lock
FB	1	Input	Opt.	Feedback (only external feedback)
CLK(freq)	1	Output	Opt.	Clk1 with the required freq
CLK(freq)	1	Output	Opt.	CLK2 with the required freq

Axcelerator PLL Parameter Description

Table 143 · Parameter Description

Parameter	Value	Function
LPMTYPE	LPMPLL	PLL category
LPM_HINT	PRIM	Only primary output
	SEC	Only secondary output
	BOTH	Both outputs
FB	**Internal** External	Feedback
IFREQ_	14.0 - 200.0 MHz	Input Frequency
PFREQ_	14.0 - 1000.0	Primary Clock frequency
SFREQ_	14.0 - 1000.0	Secondary Clock frequency
DT	**STATIC** DYNAMIC	Delay type
DELAYSIGN	+ve -ve	Positive or negative delay
DELAYVALUE	0 - 3.75 ns	In steps of 250 ps**A**
CASCADE	**YES** NO	Cascade two PLLs to achieve the required output fre-quency
REFCLKPAD	**DEDICATED** EXTERNAL	Source of REFCLK, the Dedicated Pad, or any external net
CLK1OUT	**HW** RC RN	Clock network to which PLL is connected, Hard-wired Clock, Routed Clock, or Routed Net

A. In the GUI, the delay is entered directly as a value between -3.75 and +3.75 without breaking it into sign and value.

Fusion Static PLL

Related Topics

Fusion Static PLL Functionality

Fusion Static PLL I/O Description

Key Features

The Static PLL for Fusion contains a PLL core, delay lines, clock multipliers/dividers, PLL reset generator (you have no control over the reset), global pads, and all circuitry for the selection and interconnection of the "global" pads to the global network. The PLL Core consists of a Phase Detector, L.P. Filter, and a 4-Phase VCO, and the following:

- RC Oscillator Clock Source - If you choose RC Oscillator as the clock source the input frequency is fixed at 100MHz. The divide-by-half feature is available if you bypass the PLL for the primary output.
- Divide by half behavior - Available if clock source is RC Oscillator and PLL is bypassed for the given output (A, B, C). When activated, the output divider (U, V, or W) gets divided by 2. Thus if the divider is 3, divide-by-half ON makes the divider 1.5.
- Crystal oscillator clock source - no special configuration options are available if you use the crystal oscillator as your clock source. Select this option if you are using a crystal oscillator as your clock source.
- Availability of output dividers in bypass mode - If you bypass the PLL in the primary output, you can specify an output frequency that is some divisible of the input frequency. The dividing factor must be an integer between 1 and 32.

The Static PLL performs the following basic functions:

- Clock phase adjustment
- Clock delay minimization
- Clock frequency synthesis

In addition it also:

- Enables access from the global pads to the global network and the PLL block
- Permits the three global lines on each side of the chip to be driven either by the global pads, core, and/or the outputs from the PLL block
- Enables access from PLL to the core

The block contains several programmable dividers, each of them providing division factors 1, 2, 3, 4……k (where k depends on the number of bits used for the division selection). Overall, you can define a wide range of multiplication and division factors, constrained only by the PLL frequency limits, according to:

m/(n*u)

m/(n*v)

m/(n*w)

The clock conditioning circuit block performs a positive / negative clock delay operation in increments of 200 ps, of up to 6.735 ns (at 1.5V, 25C, typical process) before or after the positive clock edge of the incoming reference clock. Furthermore, the system allows for the selection of one of four clock phases of fout, at 0, 90, 180, and 270 degrees.

A "Lock" signal is provided to indicate that the PLL has locked on to the incoming signal. A "Power-down" signal switches off the PLL block when it is not used.

Fusion Static PLL Functionality

The input clock, f_{in}, is first passed through the adjustable divider (FINDIV) prior to application to the PLL core phase detector's PLLFIN input.

The feedback signal, to which fin is compared, can be selected from several sources, giving the Static PLL its flexibility. All sources of the feedback signal can be divided by 1, 2, 3, …128 in divider FBDIV. This has the effect of multiplying the input signal. The source signals are:

- The VCO output signal, with 0o phase shift and zero additional time delay
- A delayed version of the VCO output, in selectable increments of 200 ps, up to 6.735 ns
- An external feedback signal from I/O

Each of the above feedback source signals can be further delayed by a fixed amount designed to emulate the delay through the chip's clock tree. This allows for clock-line de-skewing operations.

When the loop has acquired lock, the Lock Detect signal will be asserted. This signal will be available to the logic core, via the output port LOCK.

Once locked, the various output combinations will be available to the Global lines.

PLL Power Down

The PLL can be placed in power-down mode by setting the power down signal PWRDWN to low. When in power-down mode, the PLL draws less than 100mA of current and sends 0V signals on all outputs.

Configuring the Fusion Static PLL

The Fusion Static PLL includes (shown in the figure below) an option to choose the source of the input clock as one of the following:

- Hardwired I/O driven
- External I/O driven
- Core Logic driven
- Crystal oscillator
- RC oscillator

The option to bypass the PLL for the primary output.

Configuration selections available for frequency, delay and phase.

After you select the Static PLL from the Catalog, you must configure it. To do so:

1. Select your output. A total of five outputs can be obtained from the Static PLL. Select the check box next to each required output to select it.
    - GLA is always selected
    - GLB and YB have the same output frequency. They can be delayed by different amounts by setting the individual delays. GLB drives a Global while YB drives a core net. Using only YB also burns the global driver for GLB. However, the global rib is available.
    - GLC and YC have the same output frequency. They can be delayed by different amounts by setting the individual delays. GLB drives a Global while YB drives a core net. Using only YC also burns the global driver for GLB. However, the global rib is available.
    - The input signal CLKA is the reference clock for all five outputs
2. Specify your Internal Feedback. The source of the feedback signals will be the VCO output signal, with 0 degree phase shift and zero additional time delay. (top Selection on the Feedback MUX) or A delayed version of the VCO output, in selectable increments of 200 ps, up to 6.735 ns. This delay advances the feedback clock, thereby advancing all outputs by the delay value specified for the feedback delay element (middle selection of the Feedback MUX).
3. Set your Fixed System Delay. By choosing the non-zero value for this delay, the feedback source signal can be further delayed by a fixed amount of mask delay designed to emulate the delay through the chip's clock tree. This allows for clock-line de-skewing operations.
4. Specify your input clock:
    - Input Clock Frequency between 1.5 – 350 MHz
    - Input Clock Source as one of the following: Driven by the hardwired I/O; Driven by an external I/O from a different I/O location; Driven by Core Logic
5. Specify the primary output beginning with the source of the input clock (shown below in *italics*)
    - *Output bypassing the PLL (top selection of the GLA MUX)*. In this case, VCO phase shift and output frequency selection are not available. Output frequency is the same as input frequency in this case.
    - *Output directly from the VCO (middle selection of the GLA MUX)*. The phase shift of 0, 90, 180, or 270 is available in this case.
    - *Delayed version of the zero phase shift output from the VCO*. Phase-shift selection is unavailable for this (bottom selection of the GLA MUX). This output can be used for

two purposes: a) to use the feedback delay as an additional delay on the output if feedback advance has not been specified (top and bottom selections of the feedback MUX); b) to compensate for the feedback advance for this particular output if feedback advance has been specified (middle selection of the feedback MUX).

- Output frequency (1.5 – 350 MHz)
- VCO Phase-Shift (one of 0, 90, 180, or 270 degrees); the phase shift is out of the VCO. The phase shift will be impacted by the value of the divider after the VCO.
- An optional Extra Output Delay, in selectable increments of 200 ps, up to 6.735 ns.

6. Specify Secondary1 and Secondary2 Outputs. Select the source of the output clock (shown below in *italics*):
   - *Output directly from the VCO (top selection of the GLB/GLC MUX)* - The phase shift of 0, 90, 180, or 270 is available in this case.
   - *Delayed version of the zero phase shift output from the VCO* - Phase-shift selection is unavailable for this (bottom selection of the GLB/GLC MUX). This output can be used for two purposes: a) to use the feedback delay as an additional delay on the output if feedback advance has not been specified (top and bottom selections of the feedback MUX); b) to compensate for the feedback advance for this particular output if feedback advance has been specified (middle selection of the feedback MUX).
   - Set your Output frequency (1.5 – 350 MHz)
   - VCO Phase-Shift (one of 0, 90, 180, or 270 degrees); the phase shift is out of the VCO. The phase shift will be impacted by the value of the divider after the individual optional Extra Output Delay for each of the Global and Core outputs, in selectable increments of 200 ps, up to 6.735 ns.

Fusion Static PLL Core Restrictions

After you make all your selections, the software generates a core with your configurations. However, there are a number of restrictions in the possible values for the input and output frequencies. They are:

- Input to the PLL must be between 1.5 and 350 MHz
- Output from the PLL must be between 1.5 and 350 MHz
- The reference input to the PLL core (fin/n) must be between 1.5 and 5.5 MHz. The PLL Core output must be between 24 and 350 (fin *m/n) .

If the software cannot generate the frequency you requested, it tries to generate a frequency that is as close as possible after it satisfies all the above conditions. It prints a message in the Log file indicating the actual PLL output frequency. If more than one output is specified, the software tries to find the multiplication and division factors with the smallest total error among all the outputs.

Total Delays and Input Delays

The software prints out the total delays of the selected outputs in the Log file after feedback delay, feedback advance, system delay, and extra output delay are evaluated.

Total Delay on an Output = -feedback advance – de-skew system delay + feedback delay + extra output delay + intrinsic delay.

Fusion Static PLL I/O Description

Table 144 · Static PLL Signal Description

Name	Type	Required/Optional	Description
GLA	OUT	Req	Primary clock output
CLKA	IN	Req	Reference clock
POWERDOWN	IN	Req	Power down signal. A low on this turns off the PLL
LOCK	OUT	Req	Pll lock
EXTFB	IN	Opt	External feedback
GLB	OUT	Opt	Global output for Secondary1 clock
YB	OUT	Opt	Core output for Secondary1 clock
GLC	OUT	Opt	Global output for Secondary2 clock
YC	OUT	Opt	Core output for Secondary2 clock

IGLOO and ProASIC3 Static PLL Summary

The ProASIC3E Clock Conditioning Circuit (CCC) contains a PLL core, delay lines, clock multipliers/dividers, PLL reset generator (you have no control over the reset), global pads, and all the circuitry for the selection and interconnection of the “global” pads to the global network. The PLL Core consists of a Phase Detector, L.P. Filter, and a 4-Phase VCO.

Related Topics

IGLOO and ProASIC3 Static PLL Functionality

IGLOO and ProASIC3 Static PLL I/O Description

IGLOO and ProASIC3 Static PLL Implementation Rules / Timing Diagrams

The clock conditioning circuit performs the following basic functions:

- Clock phase adjustment
- Clock delay minimization
- Clock frequency synthesis

In addition it also

- Allows access from the global pads to the global network and the PLL block
- Permits the three global lines on each side of the chip to be driven either by the global pads, core, and/or the outputs from the PLL block
- Allows access from PLL to the core

The block contains several programmable dividers, each of them providing division factors 1, 2, 3, 4……k (where k depends on the number of bits used for the division selection). Overall, you can define a wide range of multiplication and division factors, constrained only by the PLL frequency limits, according to:

m/(n*u)

m/(n*v)

m/(n*w)

The clock conditioning circuit block performs a positive / negative clock delay operation in increments of 200 ps, of up to 6.735 ns (at 1.5V, 25C, typical process) before or after the positive clock edge of the incoming reference clock. Furthermore, the system allows for the selection of one of four clock phases of f_{out}, at 0, 90, 180 and 270 degrees.

A “Lock” signal is provided to indicate that the PLL has locked on to the incoming signal. A “Power-down” signal switches off the PLL block when it is not used.

IGLOO and ProASIC3 Static PLL Functionality

The input clock, f_{in}, is first passed through the adjustable divider (FINDIV) prior to application to the PLL core, phase detector's PLLFIN input.

The feedback signal, to which f_{in} is compared, can be selected from several sources, giving the CCC its flexibility. All sources of the feedback signal can be divided by 1, 2, 3, …128 in divider FBDIV. This has the effect of multiplying the input signal. The source signals are:

- The VCO output signal, with 0 degree phase shift and zero additional time delay
- A delayed version of the VCO output, in selectable increments of 200 ps, up to 6.735 ns
- An external feedback signal from I/O

Each of the above feedback source signals can be further delayed by a fixed amount designed to emulate the delay through the chip’s clock tree. This allows for clock-line de-skewing operations.

When the loop has acquired lock, the Lock Detect signal will be asserted. This signal will be available to the logic core, via the output port LOCK.

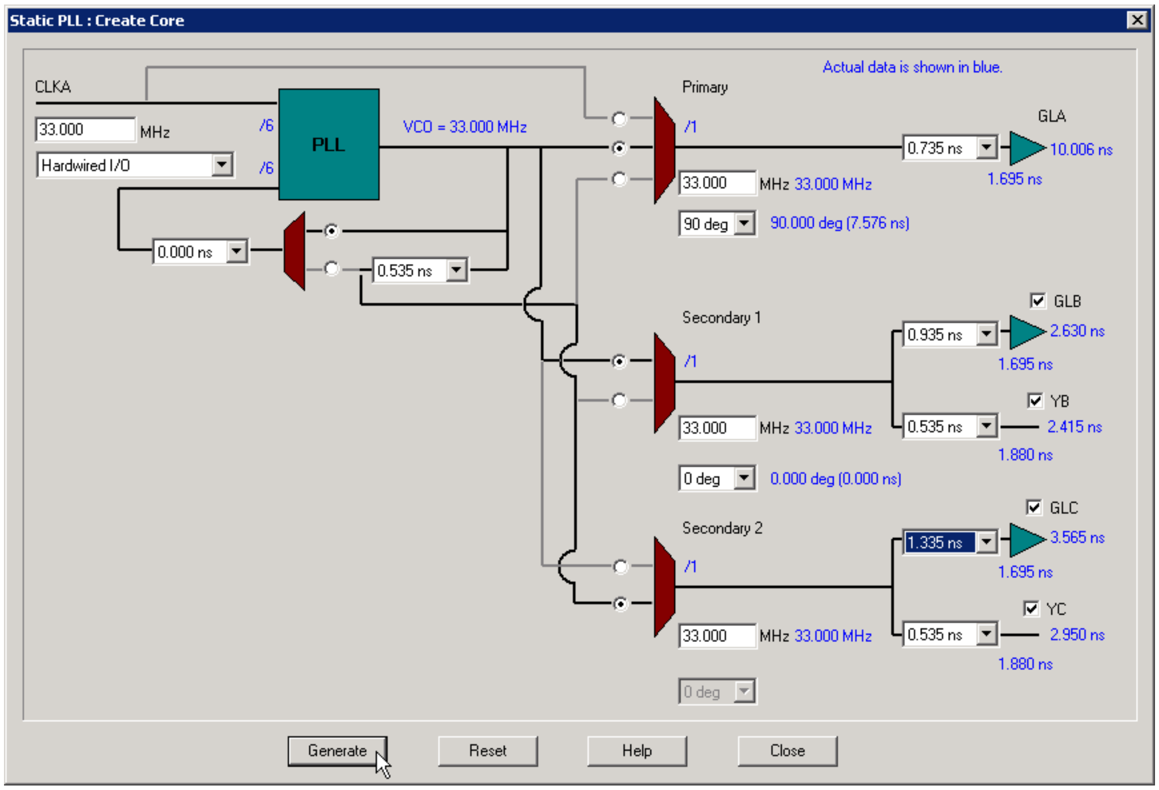
Once locked, the various output combinations will be available to the Global lines.

PLL Power Down

The PLL can be placed in power-down mode by setting the power down signal PWRDWN to low. When in power-down mode, the PLL draws less than 100mA of current and sends 0V signals on all outputs.

The Fusion Static PLL and ProASIC3 Static PLL include the following features.

- An option to choose the source of the input clock as one of the following.
  Hardwired I/O driven
  External I/O driven
  Core Logic driven
  Crystal oscillator (Fusion only)
  RC oscillator (Fusion only)
- The option to bypass the PLL for the primary output.
- Configuration selections available for frequency, delay and phase.


Static PLL : Create Core

Actual data is shown in blue.

CLKA
33.000 MHz
Hardwired I/O
/6
/6
VCO = 33.000 MHz
0.000 ns
0.535 ns

Primary
/1
33.000 MHz 33.000 MHz
90 deg 90.000 deg (7.576 ns)
GLA
0.735 ns
10.006 ns
1.695 ns

Secondary 1
/1
33.000 MHz 33.000 MHz
0 deg 0.000 deg (0.000 ns)
GLB
0.935 ns
2.630 ns
1.695 ns
YB
0.535 ns
2.415 ns
1.880 ns

Secondary 2
/1
33.000 MHz 33.000 MHz
0 deg
GLC
1.335 ns
3.565 ns
1.695 ns
YC
0.535 ns
2.950 ns
1.880 ns

Figure 13 · Configuring the Fusion Static PLL

After you select one of the PLL cores from the Catalog, you must configure it. To do so:

1. Select your output. After you choose to configure the CCC, you must select the number of outputs required. A total of five outputs may be obtained from the CCC. Select the check box next to each required output to select it.

- GLA is always selected.
- GLB and YB have the same output frequency. They can be delayed by different amounts by setting the individual delays. GLB drives a Global while YB drives a core net. Using only YB also burns the global driver for GLB. However, the global rib is available.
- GLC and YC have the same output frequency. They can be delayed by different amounts by setting the individual delays. GLB drives a Global while YB drives a core net. Using only YC also burns the global driver for GLB. However, the global rib is available.
- The input signal CLKA is the reference clock for all five outputs.

2. Specify your Internal Feedback: The source of the feedback signals will be the VCO output signal, with 0 degree phase shift and zero additional time delay (top Selection on the Feedback MUX) or a delayed version of the VCO output, in selectable increments of 200 ps, up to 6.735 ns. This delay advances the feedback clock, thereby advancing all outputs by the delay value specified for the feedback delay element (middle selection of the Feedback MUX).

3. Set your Fixed System Delay. By choosing the non-zero value for this delay, the feedback source signal can be further delayed by a fixed amount of mask delay designed to emulate the delay through the chip's clock tree. This allows for clock-line de-skewing operations.

4. Specify your input clock.

- Input Clock Frequency between 1.5 – 350 MHz
- Input Clock Source as one of the following:
  Driven by the hardwired I/O
  Driven by an external I/O from a different I/O location
  Driven by Core Logic

5. Specify the primary output. Select source of the output clock.

- **Output bypassing the PLL** (top selection of the GLA MUX). In this case, VCO phase shift and output frequency selection are not available. Output frequency is the same as input frequency in this case.
- **Output directly from the VCO** (middle selection of the GLA MUX). The phase shift of 0, 90, 180, or 270 is available in this case.
- **Delayed version of the zero phase shift output from the VCO.** Phase-shift selection is unavailable for this (bottom selection of the GLA MUX). This output can be used for two purposes: a) to use the feedback delay as an additional delay on the output if feedback advance has not been specified (top and bottom selections of the feedback MUX); b) to compensate for the feedback advance for this particular output if feedback advance has been specified (middle selection of the feedback MUX).

- Output frequency (1.5 – 350 MHz)
- VCO Phase-Shift (one of 0, 90, 180, or 270 degrees); the phase shift is out of the VCO. The phase shift will be impacted by the value of the divider after the VCO.
- An optional Extra Output Delay, in selectable increments of 200 ps, up to 6.735 ns .

6. Specify Secondary1 and Secondary 2 Outputs. Select the source of the output clock from the following two choices

   - **Output directly from the VCO** (top selection of the GLB/GLC MUX). The phase shift of 0, 90, 180 or 270 is available in this case.
   - **Delayed version of the zero phase shift output from the VCO.** Phase-shift selection is unavailable for this (bottom selection of the GLB/GLC MUX). This output can be used for two purposes: a) to use the feedback delay as an additional delay on the output if feedback advance has not been specified (top and bottom selections of the feedback MUX); b) to compensate for the feedback advance for this particular output if feedback advance has been specified (middle selection of the feedback MUX).

- Set your Output frequency (1.5 – 350 MHz)
- VCO Phase-Shift (one of 0, 90, 180 or 270 degrees); the phase shift is out of the VCO. The phase shift will be impacted by the value of the divider after the VCO.
- An individual optional Extra Output Delay for each of the Global and Core outputs, in selectable increments of 200 ps, up to 6.735 ns.

IGLOO and ProASIC3 Static PLL I/O Description

Table 145 · I/O Description

Name	Size	Type	Required/ Optional	Function
GLA	1	Output	Req	Primary clock output
CLKA	1	Input	Req	Reference clock
POWERDOWN	1	Input	Req	Power Down Signal. A low on this signal turns off the PLL
LOCK	1	Output	Req	PLL lock
EXTFB	1	Input	Opt	External feedback
GLB	1	Output	Opt	Global Output for Secondary1 Clock
YB	1	Output	Opt	Core Output for Secondary1 Clock

Name	Size	Type	Required/ Optional	Function
GLC	1	Output	Opt	Global Output for Secondary2 Clock
YC	1	Output	Opt	Core Output for Secondary2 Clock

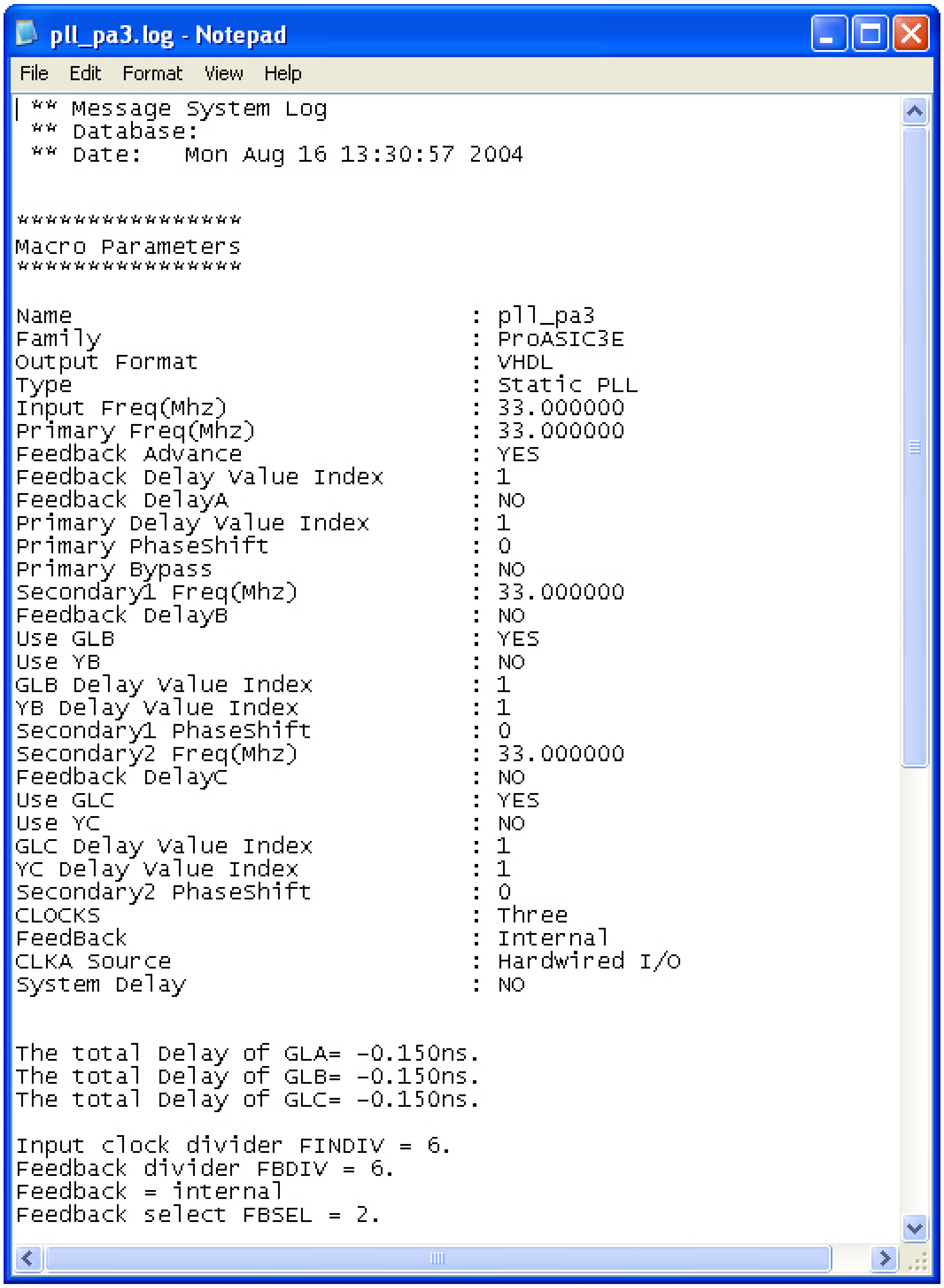
IGLOO and ProASIC3 Static PLL Implementation Rules / Timing Diagrams

After you make all your selections, the configurator generates a core with your configurations. However, there are a number of restrictions in the possible values for the input and output frequencies. They are:

- Input to the clock conditioning core (CCC) must be between 1.5 and 350 MHz
- Output from the CCC must be between 1.5 and 350 MHz
- The reference input to the PLL core (f_{in}/n) must be between 1.5 and 5.5 MHz. The PLL Core output must be between 24 and 350 (f_{in} *m/n)

Your requested PLL values are not possible in all cases, because of the VCO input, output frequency limitations, available divider ranges and inter-dependencies between the multiple outputs. In such cases, the configurator tries to generate a value that is as close as possible to the value you requested. The actual values that the configurator can achieve are shown on the screen (in blue). If you hit generate, the core is generated with the actual values rather than the specified values. The actual values are also included in the log file for future reference.

Here is a sample Log file with all the information.



```
** Message System Log
** Database:
** Date:   Mon Aug 16 13:30:57 2004


****************
Macro Parameters
****************

Name                               : pll_pa3
Family                             : ProASIC3E
Output Format                      : VHDL
Type                               : Static PLL
Input Freq(Mhz)                    : 33.000000
Primary Freq(Mhz)                  : 33.000000
Feedback Advance                   : YES
Feedback Delay Value Index         : 1
Feedback DelayA                    : NO
Primary Delay Value Index          : 1
Primary PhaseShift                 : 0
Primary Bypass                     : NO
Secondary1 Freq(Mhz)               : 33.000000
Feedback DelayB                    : NO
Use GLB                            : YES
Use YB                             : NO
GLB Delay Value Index              : 1
YB Delay Value Index               : 1
Secondary1 PhaseShift              : 0
Secondary2 Freq(Mhz)               : 33.000000
Feedback DelayC                    : NO
Use GLC                            : YES
Use YC                             : NO
GLC Delay Value Index              : 1
YC Delay Value Index               : 1
Secondary2 PhaseShift              : 0
CLOCKS                             : Three
FeedBack                           : Internal
CLKA Source                        : Hardwired I/O
System Delay                       : NO


The total Delay of GLA= -0.150ns.
The total Delay of GLB= -0.150ns.
The total Delay of GLC= -0.150ns.

Input clock divider FINDIV = 6.
Feedback divider FBDIV = 6.
Feedback = internal
Feedback select FBSEL = 2.
```

If more than one output is specified, the configurator tries to find the multiplication and division factors with the smallest total error among all the outputs.

Total Delays

The configurator prints out the total delays of the selected outputs after feedback delay, feedback advance, system delay, and extra output delay are taken into consideration.

Total Delay on an Output = -feedback advance – de-skew system delay + feedback delay + extra output delay + intrinsic delay

Input Delays

The delay between the input of the PLL and a given output can be calculated by the following equation.

Total Delay = Intrinsic delay +/- feedback delay – mask delay + phase delay + output delay

Intrinsic delay is the total delay of all the muxes and divider elements in the path. This is a fixed value for a given connectivity in a configuration. This delay varies based on the mux selection, frequency values and phase-shifts. Changing the delay element values has no impact on the intrinsic delay.

Feedback delay can be both a positive and a negative delay based on how it is configured.

Mask delay is a fixed system delay to emulate the skew of the CCC, such that the output can be deskewed by selecting this delay.

Output delay is the programmable delay independently selectable for each output.

Phase delay is the shift caused in the output with respect to the input when the VCO output is shifted by one of the 4 possible values of 0, 90, 180 or 270 degrees. This is a function of both input and output frequencies.

The delay calculation is executed using the same values for the configurator, the Simulation model and Timer such that, for typical, -2 parts under normal operating conditions, these numbers are identical. This enables you to fine-tune your delays by only adjusting the programmable output / feedback delays.

ProASICPLUS Summary

Related Topics

ProASICPLUS I/O Description

ProASICPLUS Parameter Description

Key Features

- Clock Delay Adjustment
- Clock Frequency Synthesis
- Clock phase shifting
- MIL operating conditions

Summary of the menu items available when you generate a PLL for ProASICPLUS

Input Clock Frequency - Floating point value between 1.5 and 240 MHz

Feedback - A radio button to select between Internal, External, and Deskewed feedback.

The clock-conditioning circuitry enables you to implement the feedback clock signal using either the output of the PLL, an internally generated clock, or an external clock. When external feedback is selected, an additional port, EXTFB, is made available to the user to drive the feedback. The internal feedback signal can be further delayed by a fixed amount designed to emulate the delay through the chip's clock tree. This allows for clock-line de-skewing operations. This delay is included in the feedback path when de-skewed feedback is chosen. This value is dependent on the device you are using.

Configuration - Dynamic or Static

In dynamic mode, designers are able to set all the configuration parameters using either the external JTAG port or an internally-defined serial interface. The dynamic-mode PLL can be switched to static mode during operation by just changing a mode selection bit. This way you can have one stable static configuration, yet for selected sequences of events, you can switch to dynamic mode and run the clock at a different frequency if required. For the Dynamic mode, the configurator is used to specify a stable default configuration.

Primary Clock

Bypass PLL in Primary Clock - Selecting this checkbox bypasses the PLL for the primary clock. This feature enables you to bypass the PLLCORE functionality and use the surrounding divider and delay elements. When the PLL is bypassed, the primary clock frequency must be equal to or be 1/2, 1/3 or ¼ of input frequency, as only a divider is available in the output path.

Frequency - Floating point value between 1.5 and 240 MHz. If the specified frequency cannot be achieved, the closest approximate frequency will be provided. There are some restrictions on the possible values of this frequency even in the specified range, based on the PLLCORE limitations. The core configurator takes all these limitations into consideration when generating a PLL. If the specified frequency cannot be achieved, the closest approximate frequency will be provided.

Delay - This is a floating point between -4.0 and 8.0 with increments of 0.25. When PLL is bypassed for primary clock, only 0, 0.25, 0.5 and 4 ns are valid delays.

Phase Shift - Supports four values 0, 90, 180, and 270 degrees. Not valid when PLL is bypassed for primary clock. The secondary clock cannot be phase-shifted.

Selecting a phase shift of 90 degrees and an output divider other than 1 causes the configurator to return a message about the actual phase shift being 90 divided by the divider.

Secondary Clock

Bypass PLL in Secondary clock - Selecting this check box bypasses the PLL for secondary clock. When the PLL is bypassed, the secondary clock frequency must be equal to or be 1/2, 1/3 or ¼ of secondary input frequency. This feature allows the user to bypass the PLLCORE functionality and use the surrounding divider and delay elements.

Input Frequency - Floating point value between 1.5 and 240 MHz. This is valid only when secondary clock is selected and PLL is bypassed.

Frequency - Floating point value between 1.5 and 240 MHz. This is valid only when secondary clock is selected. If the specified value cannot be achieved, the closest approximate frequency will be provided.

Delay - This is a floating point between -4.0 and 8.0 with increments of 0.25. When PLL is bypassed for secondary clock, only 0, 0.25,0.5 and 4 ns are the valid delays.

Use MIL Operating Conditions

MIL operating conditions changes the valid range on the Input (F_{in}) and Output (F_{out}) frequency requirements. If you use MIL operating conditions then the ranges change as follows:

$2\text{ MHz} <= F_{in} <= 180\text{ MHz}$

If $F_{in} <= 40$ MHz, then $F_{out} >= 18$ MHz

If $F_{in} > 40$ MHz, then $F_{out} >= 16$ MHz

$60\text{ MHz} <= F_{vco} <= 180\text{ MHz}$

For more detailed information and more schematics of the APA PLL, please refer to Using ProASIC**PLUS** Clock Conditioning Circuits,the ProASIC**PLUS** PLL Dynamic Reconfiguration Using JTAG, and the ProASICPLUS Datasheet (http://www.actel.com/documents/ProASICPlus_DS.pdf) at http://www.actel.com.

ProASICPLUS I/O Description

Table 146 · I/O Description

Name	Size	Type	Req/Opt	Function
GLA	1	Output	Opt	Secondary clock output
GLB	1	Output	Req	Primary clock output
LOCK	1	Output	Req	PLL Lock
SDOUT	1	Output	Req	Output of serial interface shift register
CLK	1	Input	Req	Input clock for primary clock
CLKA	1	Input	Opt	Input clock for secondary clock. Valid only in Bypass Mode
EXTFB	1	Input	Opt	External Feedback
SCLK	1	Input	Opt	Shift Clock (Only Dynamic Mode)
SSHIFT	1	Input	Opt	Serial Shift enable (Only Dynamic Mode)

Name	Size	Type	Req/Opt	Function
SDIN	1	Input	Opt	Serial Data in for PLL configuration bits (Only Dynamic Mode)
SUPDATE	1	Input	Opt	Serial Update (Only Dynamic Mode)
MODE	1	Output	Opt	Dynamic or Static mode indicator

ProASICPLUS Parameter Description

Table 147 · Parameter Description

Parameter	Value	Function
CLKS	**1** 2	Primary or Both outputs
FIN	1.5 - 180 MHz	Input Frequency
PRIMFREQ	1.5 - 240 MHz	Primary Output Frequency
PDELAYVAL	0 - 8 ns	Primary Delay value, in steps of .25 ns
PDELAYSIGN	0 1	Positive or Negative primary delay
PPHASESHIFT	**0** 90 180 270	Primary Phase-shift
PBYPASS	**0** 1	**No** Yes. Primary Bypass
FIN2	1.5 - 240 MHz	Secondary Input Frequency, Only if PLL is bypassed for Secondary Output
SECFREQ	1.5 - 240 MHz	Primary Output Frequency
SDELAYVAL	0 - 8 ns	Primary Delay value, in steps of 0.25 ns A
SDELAYSIGN	0 1	Positive or Negative primary delay
SPHASESHIFT	**0** 90 180 270	Primary Phase-shift
SBYPASS	**0** 1	**No** Yes. Primary Bypass
FB	**Internal** Deskewed	Feedback

Parameter	Value	Function
	External	
CONF	**STATIC** DYNAMIC	Configuration

A. In the GUI, the delay is entered directly as a value between -3.75 and +3.75 without breaking it into sign and value.

Analog System Builder and Flash Memory System Builder

The Analog System Builder and Flash Memory System Builder are explained in separate sections of the help.

Fusion Peripherals

Crystal Oscillator Summary

You can use the crystal oscillator to drive any of the clock macros directly. To drive any macros in the core, it must be routed through a CLKSRC. The core configurator software automatically instantiates the CLKSRC if you choose the Drive Internal Logic Directly option.

Supported Families

Fusion

You must set the mode of the crystal oscillator:

RTC - Real time counter. If you are using a RTC in your design, then you must use the RTC mode for your crystal oscillator.

The SELMODE and RTCMODE pins exported in this configuration must be connected to the same signals exported from the Analog System Builder. The CLKOUT output must drive the RTC clock.

There is an optional output port that drives the internal logic.

The frequency of the XTL signal must match the frequency specified during the RTC peripheral configuration (see the Analog System Builder).

External Crystal or Ceramic Resonator - Oscillator is configured to work with an external crystal or ceramic resonator.

Based on the frequency specified and the crystal oscillator mode selection, The core configurator software automatically configures the mode, as shown in the table below:

Mode	Recommended Capacitor	Frequency Range
LOW_GAIN	100 pf	0.032 to 0.20
MEDIUM_GAIN	100 pf	0.21 to 2.0
HIGH_GAIN	15 pf	2.1 to 20.0

RC Network - Oscillator is configured to work with an external resistor-capacitor network.

RC Oscillator (RCOSC) Summary

This is a 100MHz internal RC Oscillator. It can drive any of the clock macros directly. To drive any macros in the core, it must be routed through a CLKSRC. The configurator automatically instantiates the CLKSRC if you choose the **Drive Internal Logic Directly** option.

Voltage Regulator Power Supply Monitor Summary

You can choose to activate the Real Time counter when the Voltage Regulator Power Supply Monitor is on. The VRPSM enables you to set your voltage regulator output at power up (ON or OFF). If on, and your RTC is set to turn on when the monitor is on, then your RTC starts at power up.

Click the checkbox to export the RTCPSMMATCH signal that must be connected to the RTCPSMMATCH of the Real Time Clock (part of the Analog System).

Memory and Controllers

Creating a RAM for IGLOO, ProASIC3, SmartFusion and Fusion

The core configurator automatically cascades RAM blocks to create wider and deeper memories by choosing the most efficient aspect ratio. It also handles the grounding of unused bits. The core configurator software supports the generation of memories that have different Read and Write aspect ratios.

You can create a Dual Port RAM or Two Port RAM core. A Dual Port RAM has read and write access on both ports while a Two Port RAM allows write access on one port and read access on the other port.

Each RAM topic has subtopics with additional information on I/O descriptions and parameters.

Related Topics

RAM with Initialization

RAM with Initialization Timing Diagrams and Design Tips

Dual Port RAM for IGLOO, ProASIC3, SmartFusion and Fusion Summary

Two Port RAM for IGLOO, ProASIC3, SmartFusion and Fusion Summary

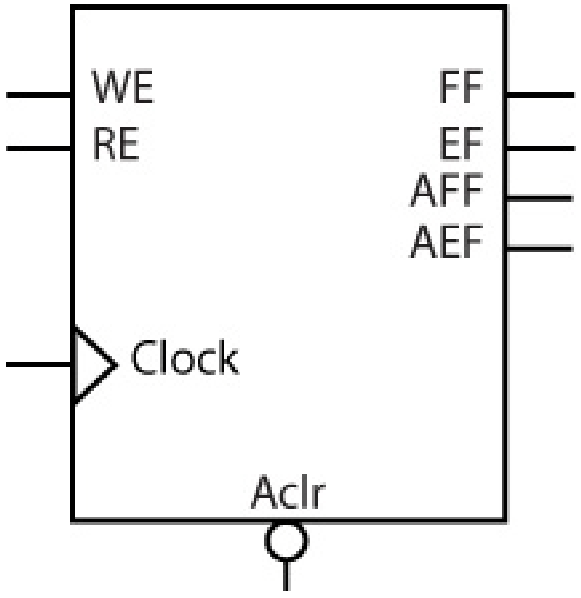
Caveats for RAM generation in the Libero IDE

- If a word width of 9 is used for Read, then Write configurations of 1, 2, or 4 will cause the MSB of the output to be undefined. These configurations are not supported. However, configurations that do not use the 9th bit (e.g., a Read width of 512x8 and a Write width of 1024x4) are supported.
- The core configurator only supports depth and width RAM cascading up to 64 blocks.
- The core configurator does not generate RAM based on a specific device. It is your responsibility to make sure the RAM fits physically on the device.
- Dynamic configuration of the aspect ratios is supported only in the Fusion RAM with Initialization core.
- The core configurator will give a configuration error for unsupported configurations.

Tips

- Writing different data to the same address using both ports in Dual Port RAM is undefined and should be avoided.
- All unused inputs must be grounded.
- WMODE is ignored during read operation.
- RESET does not reset the memory contents. It resets only the output.
- Writing to and reading from the same address is undefined and should be avoided. When using the RAM4K9 in Two Port mode, care should be taken that Read and Write operations are not going on simultaneously, by properly driving the WEN and BLK signals. This becomes extremely important in cases where multiple RAM blocks are cascaded for deeper memories. In such case, BLK must be used for address decoding.

FIFO Flag Controller (no RAM) Summary



Supported Families

3200DX, MX, SX, SX-A, eX

Related Topics

FIFO Flag Controller I/O Description

FIFO Flag Controller (no RAM) Parameter Description

Key Features

- Off-chip RAM
- Parameterized word length and depth
- FIFO full and empty flags
- Statically programmable almostfull flag to indicate when the FIFO core reaches a specific level, usually when writing into the FIFO
- Statically programmable almostempty flag to indicate when the FIFO core reaches a specific level, usually when reading from the FIFO
- Global reset of the FIFO address pointers and flag logic

Description

The FIFO Flag Controler is designed for off-chip RAM. It is a state machine generating the Flags typically used by a FIFO.

The asynchronous clear (Aclr) can be active low or active high (low is the default option and should be preferably used as for all synchronous elements in the two supported families). We will further use the word active to specify the state of a given signal. When the asynchronous clear is active, all internal registers are reset to '0'. The FIFO Controler is now in an empty state. At power up time, the FIFO must be initialized with a asynchronous clear cycle.

The full flag signal FF is optional. The FF signal is active high only (if selected) and indicates when the FIFO is full. The signal is asserted high on the rising (RISE) or falling (FALL) edge of the clock signal Clock with no delay. The FULL flag is always a function of the total block size not the user depth setting. This is tied to the silicon counter.

The empty flag signal EF is optional. The EFsignal is active low only (if selected) and indicates when the FIFO is empty. The signal is asserted low on the rising (RISE) or falling (FALL) edge of the clock signal Clock with no delay.

The write enable (WE) and read enable (RE) signals are active high requests signals for for controlling the FIFO flags. They should be logically equivalent to the write and read enable controlling the off-chip RAM.

The FIFO Controller offers a parameterizable almost-full flag (AFF). AFF flag is asserted high when the FIFO contains aff_val words or more as defined by the parameter AFF_VAL. Otherwise, AFF is asserted low. The value aff_val value is a parameter to the core, and thus logic is built at generation time to realize the almost-full flag function.

The FIFO Controller offers a parameterizable almost-empty flag (AEF). The AEF flag is asserted low when the FIFO contains aef_val words or less as defined by the parameter AEF_VAL. Otherwise, AEF is asserted high. The value aef_val value is a parameter to the core, and thus logic is built at generation time to realize the almost-empty flag function.

FIFO Flag Controller I/O Description

Table 148 · I/O Description

Port Name	Size	Type	Req/Opt?	Function
Clock	1	Input	Req.	Write and read clock
WE	1	Input	Req.	Write enable associated to the flag logic only
RE	1	Input	Req.	Read enable associated to the flag logic only
Aclr	1	Input	Req.	Asynchronous Clear
EF	1	Output	Opt.	Empty Flag
FF	1	Output	Opt.	Full Flag
AEF	1	Output	Opt.	Almost Empty Flag
AFF	1	Output	Opt.	Almost Full Flag

FIFO Flag Controller (no RAM) Parameter Description

Table 149 · Parameter Description

Parameter	Value	Function
WIDTH	Width	Word length of Data and Q
DEPTH	Depth	Number of FIFO words
FF_POLARITY	**1** 2	FF can be active high or not used
EF_POLARITY	**0** 2	EF can be active low or not used
AFF_VAL	aff_val (see parameter rules)	AFF value (not used if aff_val is 0)
AEF_VAL	aef_val see parameter rules)	AEF value (not used if aef_val is 0)
CLK_EDGE	**RISE** FALL	Clock can be rising or falling

Table 150 · Implementation Parameters - MX/DX

Parameter	Value	Description
LPMTYPE	LPM_FIFO_DQ	Generic FIFO category
LPM_HINT	FFIFOCTRL	High speed FIFO Controller
	MFFIFOCTRL	Medium speed FIFO Controller

Table 151 · Implemenation Parameters - SX/SX-A/eX

Parameter	Value	Description
LPM_HINT	FCTR	FIFO Controller

Table 152 · Fan-In Parameters

Parameter	Value	Description
CLR_FANIN	**AUTO** MANUAL	See Fan-in Control section
CLK_FANIN	AUTO **MANUAL**	See Fan-in Control section
WE_FANIN	**AUTO** MANUAL	See Fan-in Control section
RE_FANIN	**AUTO** MANUAL	See Fan-in Control section

Axcelerator FIFO

Related Topics

Axcelerator FIFO Functionality

Axcelerator FIFO I/O Description

Axcelerator FIFO Parameter Description

Axcelerator FIFO Implementation Rules

Key Features

- Parameterized word length and FIFO depth
- Dual-port synchronous FIFO
- Active High/Low enable
- Static/ Programmable/No Almost empty/full flags
- Full and Empty flags

Axcelerator FIFO Functionality

Axcelerator provides dedicated blocks of FIFO. They are actually hardwired using the RAM blocks plus some control logic. Each FIFO block has a read port and a write port. Both ports are configurable (to the same size) to any size from 4Kx1 to 128x36; thereby, allowing built-in bus width conversion (see SRAM Port Aspect Ratio table below). Each port is fully synchronous. The FIFO block offers programmable Almost Empty and Almost Full flags as well as Empty and Full flags. The FIFO block may be reset to the empty state.

Table 153 · SRAM Port Aspect Ratio

Width	Depth	ADDR Bus	Data Bus

Width	Depth	ADDR Bus	Data Bus
1	4096	ADDR [11:0]	DATA [0]
2	2048	ADDR [10:0]	DATA [1:0]
4	1024	ADDR[9:0]	DATA[3:0]
9	512	ADDR[8:0]	DATA[8:0]
18	256	ADDR[7:0]	DATA[17:0]
36	128	ADDR[6:0]	DATA[35:0]

Cascading Blocks

Blocks can be cascaded to create larger sizes, up to the capacity of one whole column of RAM blocks. The software performs all the necessary cascading for achieving the desired configuration.

The maximum WIDTH (word length) value is 65,536. The maximum DEPTH (number of words) value is 576.

The write enable (WE) and read enable (RE) signals are active high or low request signals for writing and reading, respectively; you may choose not to use them.

The RCLK and WCLK pins have independent polarity selection.

Axcelerator FIFO I/O Description

Table 154 · I/O Description

Name	Type	Req/Opt	Description
Data	IN	Req	Data Port
WE	IN	Opt	Write Enable
WClock	IN	Req	Write Clock
Q	OUT	Req	Q Port
RE	IN	Opt	Read Enable
RClock	IN	Req.	Read Clock
Full	OUT	Req.	Full Flag

Name	Type	Req/Opt	Description
Empty	OUT	Req.	Empty Flag
Afval	IN	Opt	Almost Full, Dynamically programmable
Aeval	IN	Opt	Almost Empty, Dynamically programmable
AFull	OUT	Opt	Almost Full Flag
AEmpty	OUT	Opt	Almost Empty Flag

Axcelerator FIFO Parameter Description

Table 155 · Parameter Description

Parameter	Value	Function
WIDTH	Width	Word length of Data, Q
DEPTH	Depth	FIFO Depth
WE_POLARITY	**1** 0 2	Write Enable Polarity
RE_POLARITY	**1** 0 2	Read Enable Polarity
WCLK_EDGE	**RISE** FALL	Write Clock Edge
RCLK_EDGE	**RISE** FALL	Read Clock Edge
AEVAL	Almost Empty Value	Almost Empty Flag
AFVAL	Almost Full Value	Almost Full Flag
DEVICE	75 150 300 600 1000 (May change)	Target Device, to determine blocks available for cascading

Table 156 · Parameter Rules

Device	Parameter Rules	
Axcelerator	WWIDTH	AEVAL/AFVAL UNITS

Device	Parameter Rules	
Axcelerator	000	2^{8-W}
Axcelerator	001	
Axcelerator	010	
Axcelerator	011	
Axcelerator	100	
Axcelerator	101	
Axcelerator	11x	

Axcelerator FIFO Implementation Rules

FIFO Flag Usage

In the Axcelerator FIFO, the AFVAL and AEVAL signals are each eight bits. The step size of the signal varies based on the aspect ratio to which the FIFO blocks are configured.

For example, if the FIFO is configured in the 128X36 aspect ratio, the step size is eight. That means, if a 00000011 is programmed on the AEVAL, the almost empty flag asserts after 3*8 = 24 words are written. The step sizes can be calculated from the above table for other configurations.

The core configration software automatically adjusts the AF and AE thresholds specified by changing them to the nearest step size. A message is also printed in the log file.

Since eight is the least step size for AFVAL and AEVAL, static flag configuration is not supported for widths below eight.

When the core configurator software is used to configure the FIFO to a depth that is less than the total available depth, FULL flag will not assert at the depth specified in the software. For example, if FIFO is configured to a 250X18, then the core configurator provides a total depth of 256, which is the closest size. FULL flag will assert at 256. A message appears in the Project Manager log file indicating what configuration it is providing, taking all these details into consideration.

When FIFOs are cascaded deep, the data gets written to multiple FIFO blocks. The FIFO Error flags (AFULL_ERR, AEMPTY_ERR, FULL_ERR, EMPTY_ERR) indicate if one or more of the FIFOs are in a different state than expected. You can ignore them if you wish. These ports get generated only if you are cascading FIFOs wide.

Soft FIFO Controller

Supported Families

Soft FIFO Controller with memory: IGLOO, ProASIC3, SmartFusion, Fusion, Axcelerator, SX-A, SX

Soft FIFO Controller without Memory: IGLOO, Fusion, ProASIC3, Axcelerator, ProASICPLUS, SX-A, and SX

Related Topics

Soft FIFO Controller with Memory Functionality

Soft FIFO Controller without Memory Functionality

Soft FIFO Controller I/O Description

Soft FIFO Controller Implementation Rules / Timing Diagrams

Key Features

The Soft FIFO is a user-gate alternative to the Embedded Synchronous FIFO. It provides features that are not supported by the Embedded FIFO.

The Soft FIFO has single-RAM-location granularity with the empty / full flags, whereas the Embedded FIFO only asserts the empty / full flags on the RAM block depth boundary of the FIFO configuration used.

For example, if you configure an Embedded FIFO with depth x width of 64x4, the FIFO asserts the full flag at 512. The reason that the silicon configuration satisfying your requirements uses block RAMs of 512x9; thus, the full flag only asserts at the available silicon depths. The available silicon configurations for Embedded FIFOs are 4096x1, 2048x2, 1024x4, and 512x9

The Soft FIFO can support depth and width cascading of RAM Blocks, while the Embedded FIFO only supports width cascading.

The Soft FIFO supports many more optional status ports for increased visibility and usability. These optional ports are described in more detail in the sections below.

The basic rule for configuring Soft FIFOs is: (write width * write depth) must equal (read width * read depth).

Optimize for High Speed (Width Cascading) or Low Power (Depth Cascading)

You can choose to optimize your RAM for High Speed or Low Power.

If you optimize for low power, the core configurator evaluates your RAM configuration and attempts to generate a macro with depth cascading.

Soft FIFO Controller w/ Memory vs. Soft FIFO Controller without Memory

The Soft FIFO with memory generates the FIFO controller logic and instantiates the proper synchronous RAM blocks to support the specified depth / width configuration.

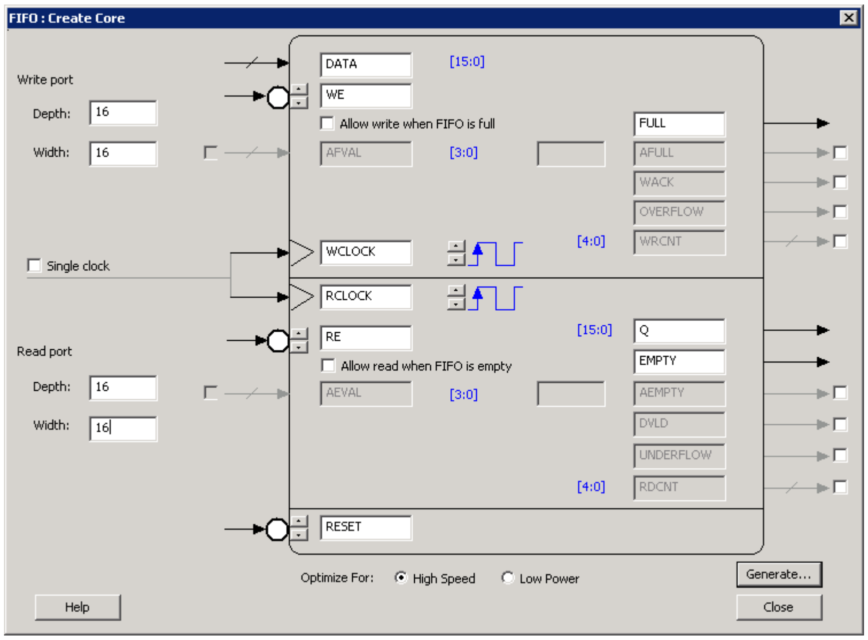
The Soft FIFO without memory generates only the FIFO controller logic. This core is intended for users who wish to use the FIFO controller with an external memory.

Also, the Soft FIFO without memory can be used in conjunction with the EDAC RAM or the numerous RAM varieties in the ProASICPLUS (APA) family.

Soft FIFO Controller with Memory Functionality

The Soft FIFO controller with memory offers a dual- or single-clock design. The dual clock design allows independent read and write clock domains. Operations in the read domain are synchronous to the read clock, and operations in the write domain are synchronous to the write clock.

Selecting the single clock option results in a much simpler, smaller, and faster design.


FIFO : Create Core

Write port
Depth: 16
Width: 16

☐ Single clock

Read port
Depth: 16
Width: 16

☐ Allow write when FIFO is full
☐ Allow read when FIFO is empty

Optimize For: (● High Speed) ○ Low Power

Figure 14 · Soft FIFO with Memory GUI

Generating Flags in the Soft FIFO Controller

Flags in the Soft FIFO Controller are generated as follows:

- The Full, Empty, Almost Full, and Almost Empty flags are registered outputs of this module, unlike the silicon version.
- The Almost Full and Almost Empty flags are optional ports; you can set the threshold values statically or dynamically.

To set a static value for the threshold: deselect the checkbox next to the AFVAL or AEVAL port; this disables the port(s) and enables the text control box next to the AFULL / AEMPTY port(s). Enter your desired static threshold into this field.

To set a dynamic value for the threshold, select the checkbox(es) next to the AFVAL or AEVAL port, this enables core generation with one or both buses. You can then dynamically input your desired threshold values.

- The Full flag is asserted on the same clock that the data that fills the FIFO is written.
- The Empty flag is asserted on the same clock that the last data is read out of the FIFO.
- The Almost Full flag is asserted on the same clock on which the threshold has been reached.
- The Almost Empty flag is asserted on the same clock on which the threshold has been reached.

For example, if you specify an almost empty threshold of 10, the flag asserts on the same read clock that causes the FIFO to contain 10 elements.

Allow Write when FIFO is full - Select this checkbox to enable the FIFO to continue write when it is full. Your existing FIFO value will be OVERWRITTEN if you are using the Soft FIFO Controller with Memory.

Allow read when FIFO is empty - Select this checkbox to enable the FIFO to continue to read when it is empty.

Area and Speed in the Soft FIFO Controller

The size and operating frequency of the Soft FIFO design is dependent upon the configuration and optional features that are enabled; note that:

- A single clock design will be smaller and faster; this is because the synchronizers and gray encoder/decoders are not required.
- Port depths that are not a power of 2 will generate a larger and slower design. The reason is that logic optimization occurs for power-of-2 depths. Thus, if you need a 66 x 8 FIFO, it may be more advantageous to select a FIFO depth of 64 or 128 if area and/or speed are concerns.

Optimization for High Speed or Low Power

High Speed results in a macro optimized for speed and area (width cascading).

Low Power results in a macro optimized for low power, but uses additional logic at the input and output (depth cascading). Performance for a low power optimized macro may be inferior to that of a macro optimized for speed. Some RAM configurations are not possible with depth cascading (such as 512 x 36), but low power optimization is a priority when the option is selected.

Soft FIFO Controller without Memory Functionality

The Soft FIFO controller with memory offers a dual- or single-clock design. The dual clock design allows independent read and write clock domains. Operations in the read domain are synchronous to the read clock, and operations in the write domain are synchronous to the write clock.

Selecting the single clock option results in a much simpler, smaller, and faster design.

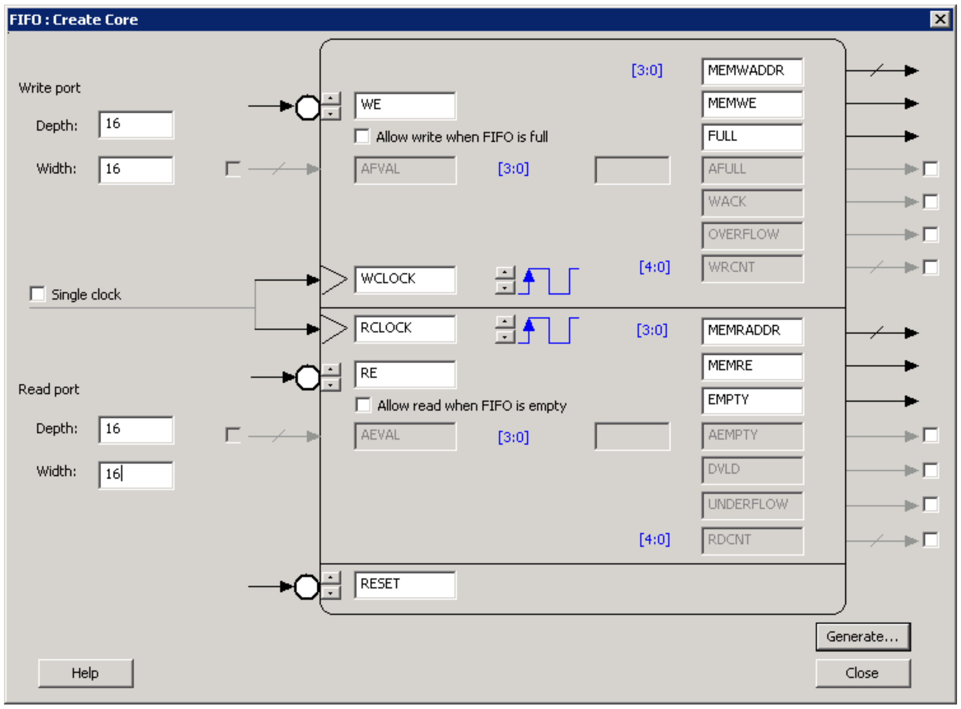


Figure 15 · Soft FIFO without Memory GUI

Generating Flags in the Soft FIFO Controller

Flags in the Soft FIFO Controller are generated as follows:

- The Full, Empty, Almost Full, and Almost Empty flags are registered outputs of this module, unlike the silicon version.
- The Almost Full and Almost Empty flags are optional ports; you can set the threshold values statically or dynamically.

To set a static value for the threshold: deselect the checkbox next to the AFVAL or AEVAL port; this disables the port(s) and enables the text control box next to the AFULL / AEMPTY port(s). Enter your desired static threshold into this field.

To set a dynamic value for the threshold, select the checkbox(es) next to the AFVAL or AEVAL port, this enables core generation with one or both buses. You can then dynamically input your desired threshold values.

- The Full flag is asserted on the same clock that the data that fills the FIFO is written.
- The Empty flag is asserted on the same clock that the last data is read out of the FIFO.
- The Almost Full flag is asserted on the same clock on which the threshold has been reached.
- The Almost Empty flag is asserted on the same clock on which the threshold has been reached.

For example, if you specify an almost empty threshold of 10, the flag asserts on the same read clock that causes the FIFO to contain 10 elements.

Allow Write when FIFO is full - Select this checkbox to enable the FIFO to continue write when it is full. Your existing FIFO value will be OVERWRITTEN if you are using the Soft FIFO Controller with Memory.

Allow read when FIFO is empty - Select this checkbox to enable the FIFO to continue to read when it is empty.

Area and Speed in the Soft FIFO Controller

The size and operating frequency of the Soft FIFO design is dependent upon the configuration and optional features that are enabled; note that:

- A single clock design will be smaller and faster; this is because the synchronizers and gray encoder/decoders are not required.
- Port depths that are not a power of 2 will generate a larger and slower design. The reason is that logic optimization occurs for power-of-2 depths. Thus, if you need a 66 x 8 FIFO, it may be more advantageous to select a FIFO depth of 64 or 128 if area and/or speed are concerns.

Optimization for High Speed or Low Power

High Speed results in a macro optimized for speed and area (width cascading).

Low Power results in a macro optimized for low power, but uses additional logic at the input and output (depth cascading). Performance for a low power optimized macro may be inferior to that of a macro optimized for speed. Some RAM configurations are not possible with depth cascading (such as 512 x 36), but low power optimization is a priority when the option is selected.

Soft FIFO Controller I/O Description

Table 157 · Soft FIFO with Memory I/O Description

Name	Type	GENFILE Parameter	Description
DATA	Input	DATA_IN_PN	The input data bus when writing the FIFO
Q	Output	DATA_OUT_PN	The output data bus when reading the FIFO
WE	Input	WE_PN	Write data into FIFO when signal is asserted
RE	Input	RE_PN	Read data from FIFO when signal is asserted
WCLOCK	Input	WCLOCK_PN	All signals in the write domain are synchronous to this clock

Name	Type	GENFILE Parameter	Description
RCLOCK	Input	RCLOCK_PN	All signals in the read domain are synchronous to this clock
FULL	Output	FF_PN	Indicates that the FIFO is full
EMPTY	Output	EE_PN	Indicates that the FIFO is empty
RESET	Input	ACLR_PN	Asynchronous reset
AEMPTY	Output	AE_PN	Indicates that the FIFO has reached the Almost Empty threshold value
AFULL	Output	AF_PN	Indicates that the FIFO has reached the Almost Full threshold value
AEVAL	Output	AE_PORT_PN	Almost empty threshold value
AFVAL	Output	AF_PORT_PN	Almost full threshold value
WACK	Output	WACK_PN	Indicates that a write on the FIFO has succeeded
DVLD	Output	DVLD_PN	Indicates that a read on the FIFO has succeeded
OVERFLOW	Output	OVRFLOW_PN	Indicates that a write in the previous clock cycle failed
UNDERFLOW	Output	UDRFLOW_PN	Indicates that a read in the previous clock cycle has failed
RDCNT	Output	RDCNT_PN	The remaining number of elements in the FIFO from the read domain
WRCNT	Output	WRCNT_PN	The remaining number of elements in the FIFO from the write domain
CLOCK	Input	CLOCK_PN	Clock (in the case of single clock)

Table 158 · Soft FIFO without Memory I/O Description

Name	Type	GENFILE Parameter	Description
WE	Input	WE_PN	Write data into FIFO when signal is asserted
RE	Input	RE_PN	Read data from FIFO when signal is asserted
WCLOCK	Input	WCLOCK_PN	All signals in the write domain are synchronous to this clock
RCLOCK	Input	RCLOCK_PN	All signals in the read domain are synchronous to this clock
FULL	Output	FF_PN	Indicates that the FIFO is full
EMPTY	Output	EE_PN	Indicates that the FIFO is empty
RESET	Input	ACLR_PN	Asynchronous reset
AEMPTY	Output	AE_PN	Indicates that the FIFO has reached the Almost Empty threshold value
AFULL	Output	AF_PN	Indicates that the FIFO has reached the Almost Full threshold value
AEVAL	Output	AE_PORT_PN	Almost empty threshold value
AFVAL	Output	AF_PORT_PN	Almost full threshold value
WACK	Output	WACK_PN	Indicates that a write on the FIFO succeeded
DVLD	Output	DVLD_PN	Indicates that a read on the FIFO succeeded
OVERFLOW	Output	OVRFLOW_PN	Indicates that a write in the previous clock cycle failed

Name	Type	GENFILE Parameter	Description
UNDERFLOW	Output	UDRFLOW_PN	Indicates that a read in the previous clock cycle has failed
RDCNT	Output	RDCNT_PN	The remaining number of READ domain elements in the FIFO
WRCNT	Output	WRCNT_PN	The remaining number of WRITE domain elements in the FIFO
MEMWADDR	Output	MEMWADDR_PN	Memory write address for external memory
MEMRADDR	Output	MEMRADDR_PN	Memory read address for external memory
MEMWE	Output	MEMWE_PN	Memory write enable for external memory
MEMRE	Output	MEMRE_PN	Memory read enable for external memory
CLOCK	Input	CLOCK_PN	Clock

Soft FIFO Controller Implementation Rules / Timing Diagrams

Write Operation

During a write operation when the WE signal is asserted the FIFO stores the value on the DATA bus into the memory. The WACK signal will be asserted each time a successful write operation occurs on the FIFO. If the FIFO fills up then the FULL flag is asserted indicating that no more data can be written. The AFULL flag is asserted when the number of elements in the FIFO equals the threshold amount.

If a write operation is attempted while the FIFO is full, the OVERFLOW signal is asserted on the next clock cycle, indicating that an error has occurred. The OVERFLOW signal is asserted for each write operation that fails. A sample timing diagram of a FIFO with depth configuration of 4, almost full value set to 3, and rising clock edge is shown in the figure below.

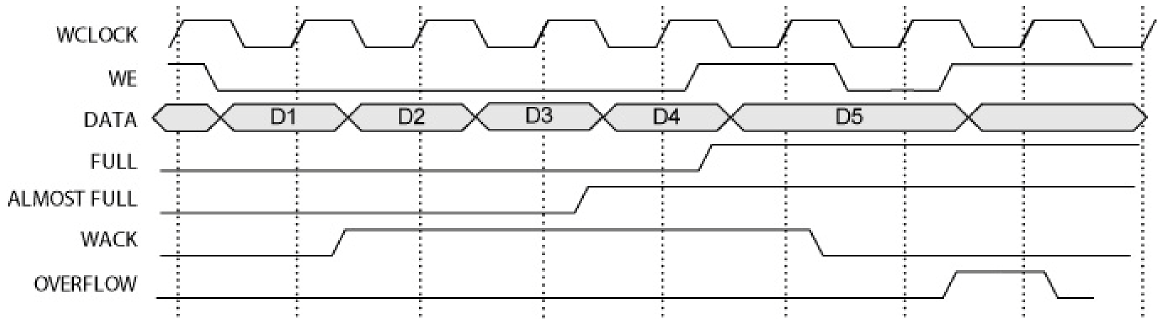


Figure 16 · Write Operation and Flags

Read Operation

During a read operation when the RE signal is asserted the FIFO reads a data value onto the Q bus from the memory. The data is available to the client 2 clock cycles after the assertion of the RE, this data is held on the bus until the next RE is asserted. The DVLD signal is asserted on the same clock cycle that the data is available. Therefore, the client logic can monitor the DVLD signal for indication of valid data. However, DVLD only asserts for the first clock cycle that the new data is available, whereas the actual data may still be on the data bus.

If the FIFO is emptied then the EMPTY flag is asserted to indicate that no more data elements can be read. The AEMPTY flag is asserted when the number of elements in the FIFO equals the set threshold amount.

If a read operation is attempted while the FIFO is empty, the UNDERFLOW signal is asserted on the next clock cycle indicating that an error has occurred. The UNDERFLOW signal is asserted for each read operation that fails.

A sample timing diagram of a FIFO with depth configuration of 4, almost empty value set to 1, and rising clock edge is shown in the figure below.

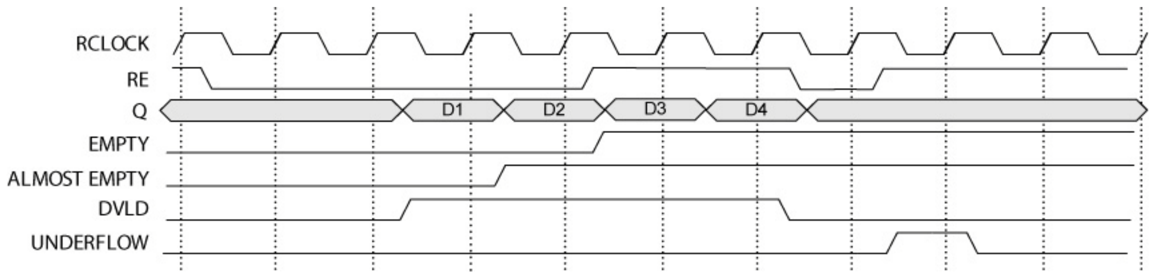


Figure 17 · Read Operation and Flags

Operations with a Variable Aspect Ratio

A FIFO with variable aspect width has different depth and width configurations for the write and read side. There are some special considerations when using this type of FIFO, including:

- Data order – Write side has smaller width than Read side: The FIFO starts writing to the least significant portion of the memory up. (refer to the timing diagram below)

- Data order – Write side has larger width than Read side: The FIFO starts reading from the least significant portion of the memory. Meaning if the first word into the write side is 0xABCD, the words read out of the FIFO will be 0xCD followed by 0xAB.
- Full flag generation – The FULL is asserted when a full word from the write perspective cannot be written in. The FULL deasserted only if there is enough space in the FIFO to write a full word from the write aspect ratio. (refer to the timing diagram below)
- Empty flag generation – The EMPTY is deasserted only when a full word from the read aspect ratio can be read out. The EMPTY is asserted if the FIFO does not contain a full word from the read aspect ratio (refer to the timing diagram below).
- The implication of the status flag generation is that it is possible to have a partial word in the FIFO that may not be immediately visible on the read side. For example, take a situation where the write side has a smaller width than the read side. The write side writes 1 word and finishes. In this type of scenario, the application using the FIFO must consider what a partial data word represents.

If the partial data word can not be processed downstream than it is meaningless to take it out of the FIFO until it has reached a full-word. However, if the partial word is considered valid and can be processed downstream in its 'incomplete' state, then some other type of mechanism needs to be designed to handle this condition.

The diagram below illustrates a condition where the write side is configured has x4 width and the read side as x8 width.

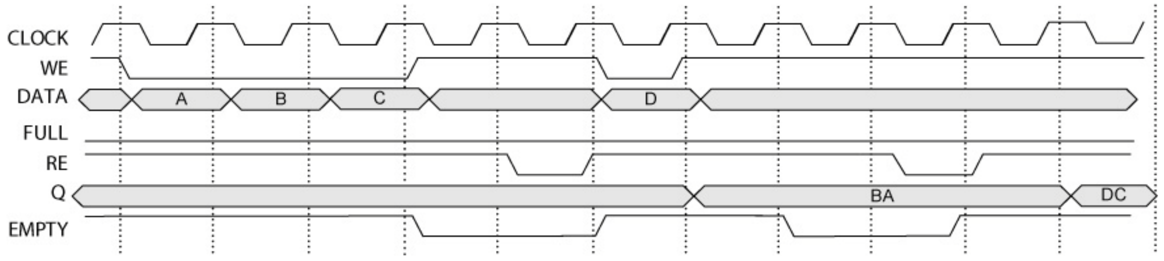


Figure 18 · Write and Read Operations with Variable Aspect

Synchronous/Asynchronous Dual Port FIFO for ProASIC and ProASICPLUS

This core has been obsoleted and must be used with caution.

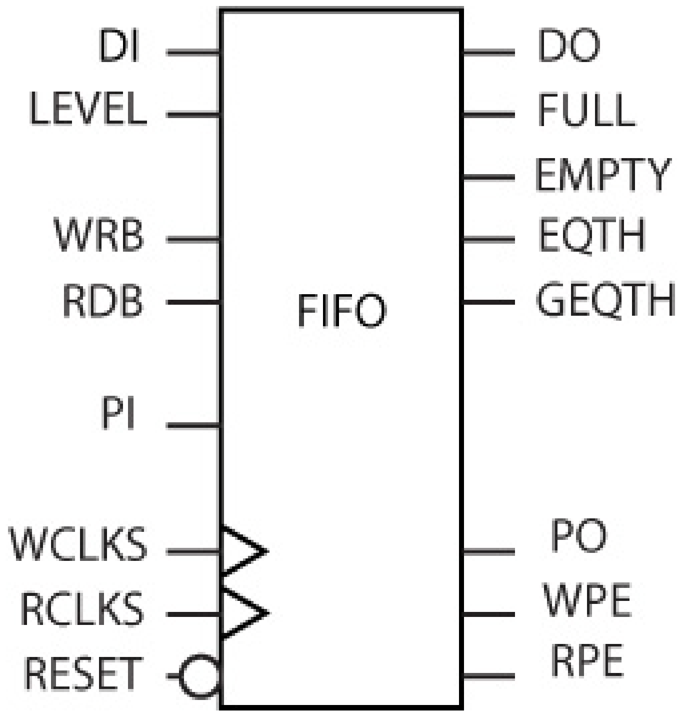
See Implementation Rules.

Related Topics

Synchronous/Asynchronous Dual Port FIFO for ProASIC and ProASICPLUS I/O Description

Synchronous/Asynchronous Dual Port FIFO for ProASIC and ProASICPLUS Parameter Description

Synchronous/Asynchronous Dual Port FIFO for ProASIC and ProASICPLUS Implementation Rules



Key Features

- Parameterized word length and depth
- Dual-port RAM architecture
- Asynchronous, synchronous-transparent or synchronous-pipelined read
- Asynchronous, or synchronous write
- Parity check or generate, both even and odd
- Supported netlist formats: EDIF, VHDL and Verilog

There is no limitation for depth and width. However, it is your responsibility to ensure that the FIFOs used in a design can fit on the device chosen for the design.

Synchronous/Asynchronous Dual Port FIFO for ProASIC and ProASICPLUS I/O Description

This core has been obsoleted and must be used with caution.

See the Implementation Rules.

Table 159 · I/O Description

Port Name	Size	Type	Req./Opt.	Function

Port Name	Size	Type	Req./Opt.	Function
DI<0:8>	9	Input	Req.	Input data bits <0:8>; <8> can be used for parity IN
LEVEL	8[A]	Input	Opt.	Defines when EQTH and GEQTH should react (hardcoded for static trigger level)
WRB	1	Input	Req.	Write pulse (active low)
RDB	1	Input	Req.	Read pluse (active low)
WCLK	1	Input	Req.	Write clock (active high)
RCLK	1	Input	Req.	Read clock (active low)
RESET	1	Input	Req.	Reset for FIFO pointers (active low)
DO<0:8>	9	Output	Req.	Output data bits <0:8>; <8> can be used for parity OUT
EMPTY	1	Output	Req.	Empty flag
FULL	1	Output	Req.	Full flag
EQTH	1	Output	Req.	Flag is true when FIFO hold (LEVEL) words
GEQTH	1	Output	Req.	Flag is true when FIFO hold (LEVEL) words or more
PI	WIDTH	Input	Opt.	Input parity bits
PO	log2(width)	Output	Opt.	Parity bits
WPE	1	Output	Opt.	Write parity error flag (active HIGH), available only for parity checking models
RPE	1	Output	Opt.	Read parity error flag (active HIGH), available only for parity checking models
PARODD	1	Input	Opt.	Selects ODD parity generation/detect when HIGH; selects EVEN parity when LOW

A. LEVEL is always eight bits. That means for values of DEPTH greater than 256 not all values will be possible, e.g. for DEPTH =512, LEVEL can have the values 2, 4, … , 512.

This holds true only for dynamically triggered FIFOs. For a static trigger, all values of the depth are possible. In the case of dynamic trigger, only values that are divisible by the number of 256X9 FIFO blocks cascaded to achieve the required depth are possible.

In simulation, EQTH/GEQTH reacts to LEVEL * [# of 256x9 modules (rounded up)].

For example, with 1000x32 sync dynamic, level=1, EQTH/GEQTH toggles after 4 reads. For a 700x32 sync dynamic, level=1, EQTH/GEQTH toggles after 3 reads.

Synchronous/Asynchronous Dual Port FIFO for ProASIC and ProASICPLUS Parameter Description

This core has been obsoleted and must be used with caution.

See Implementation Rules.

Table 160 · Parameter Description

Parameter	Value	Function
WIDTH	Width	Word length of DI and DO
DEPTH	Depth	Number of RAM words
RDA	Async transparent Pipelined	Read data access
WRA	async sync	Write data access
OPT	speed area	Optimization
PARITY	checkeven checkodd geneven genodd none	Parity check or parity generation

Table 161 · Implementation Parameters

Parameter	Value	Description
LPMTYPE	LPM_FIFO_DQ	Generic FIFO category
LPM_HINT	FIFO_DYN	FIFO with dynamic trigger level
	FIFO_STATIC	FIFO with static trigger level

Parameter Rules for FIFO with Static Trigger Level

LEVEL <= DEPTH

If DEPTH > 256 not all values for Level will be available (automatic value correction).

This holds true only for dynamically triggered FIFOs. For a static trigger, all depth values are possible. For dynamic triggers, only values that are divisible by the number of 256X9 FIFO blocks cascaded to achieve the required depth are possible.

For example, for a depth of 512, which uses two 256 blocks in cascade, only multiples of 2 are possible. For depth of 768, which uses three blocks, multiples of 3 are the only values possible for the LEVEL threshold.

Synchronous/Asynchronous Dual Port FIFO for ProASIC and ProASICPLUS Implementation Rules

The core configurator constructs depth cascaded FIFOs out of individual hard FIFO macros. The particular architecture chosen to perform the depth cascading requires special care when using this core.

A Soft FIFO Controller is available; it is constructed out of soft gates that do not exhibit any of the behaviors described below.

Data is Valid for Only a Single Clock After a Read Operation

The behavior of the depth cascaded FIFO is not the typical behavior of FIFOs. After a read is asserted and the data is made available at the data bus of the FIFO, the data is valid for only a single clock cycle. The cause is the depth cascade uses a ping-pong architecture and a read operation advances the enable to the next FIFO block. Thus, the next FIFO block's data becomes available after a single clock cycle.

This behavior does not appear in pre-synthesis simulations. Only after post-synthesis or post-layout simulations, where timing delays are included, does the issue become visible.

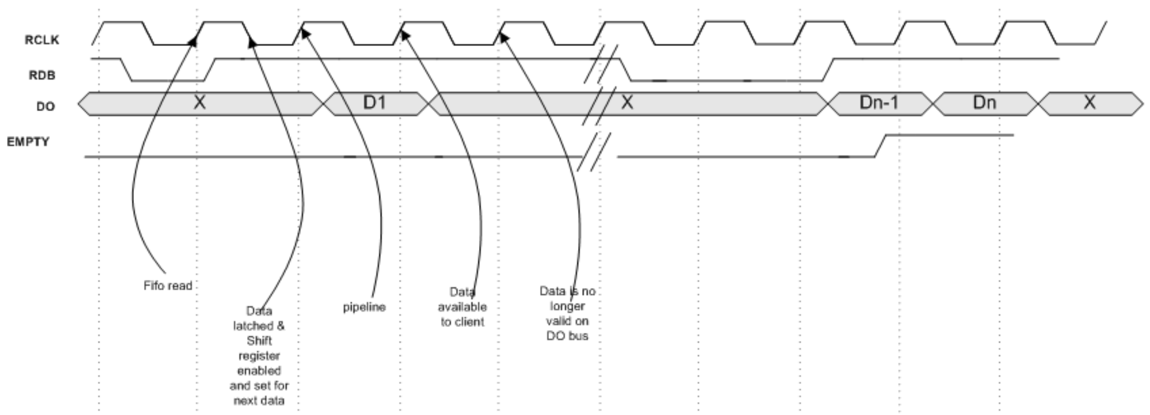

RCLK
RDB
DO
EMPTY
X
D1
X
Dn-1
Dn
X
Fifo read
Data latched & Shift register enabled and set for next data
pipeline
Data available to client
Data is no longer valid on DO bus

Figure 19 · Basic Single and Multiple Read of Depth Cascaded FIFO

If the client logic requires the data to remain available on the data out port, the following workaround is available:

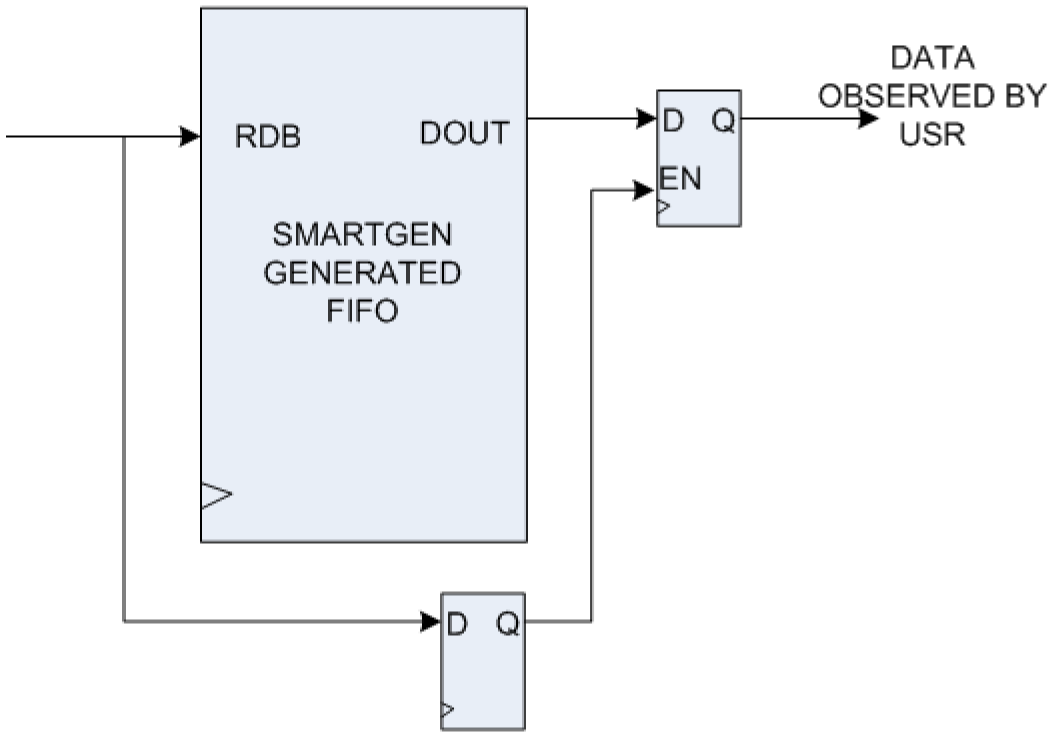


Figure 20 ·

Figure 21 · FIFO Read Data Workaround

An extra data stage pipeline is added right after the FIFO with a delayed read signal as an enable. The above diagram is for the FIFO that does not use the pipeline feature, if the pipeline on the FIFO is enabled then an extra stage of delay for the read signal is required.

Flags Depend on Depth Cascade Size

The FULL/EMPTY/EQTH/GEQTH flags are generated by the AND of all the FIFOs in the depth cascade. Thus, all FIFO's must have the same status before the flag to the user is asserted.

As a result, these status flags are only generated dependent upon the depth and size of the FIFO. The FIFO macro has a port to allow it to be configured for various depth sizes, however these sizes are only powers of 2 such as: 2, 4, 8, 16, 32, 64, 128, and 256.

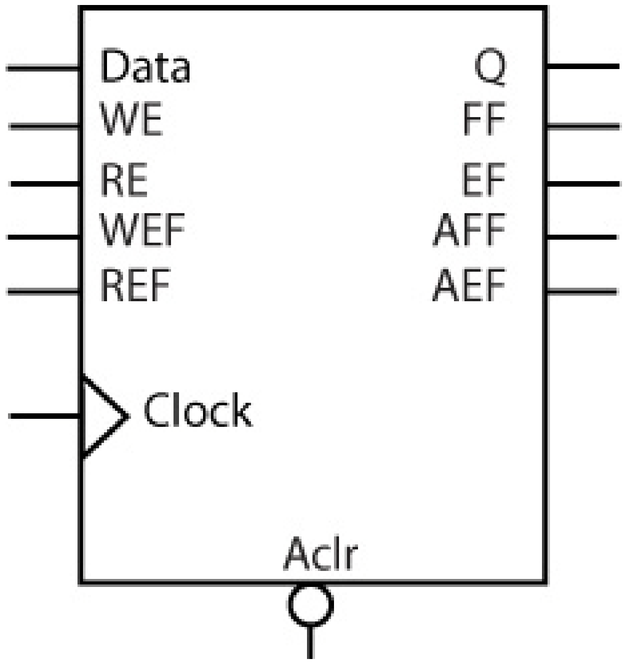
Thus, a depth configuration that does not utilize an evenly distributed factor of those depths produces unexpected status flags.

For example, for a FIFO configuration of 596x8, a depth cascade of 3 FIFO blocks will be used. Each block is configured for 256 deep to meet the depth requirement. Thus, the full flag for each individual FIFO block only asserts when it reaches 256 elements. This implies that the generated FIFO will NOT assert the full flag at 596 but rather at 256*3 or 768.

Timing Waveforms

Please refer to the timing waveforms presented in the flash family datasheets. The datasheets are available on the Actel website at http://www.actel.com.

Synchronous Dual Port Fifo with Flags Summary



Supported Families

eX, SX-A, SX

Related Topics

Synchronous Dual Port Fifo with Flags I/O Description

Synchronous Dual Port Fifo with Flags Parameter Description

Synchronous Dual Port Fifo with Flags Implementation Rules / Timing Diagrams

Key Features

- On-chip RAM
- Parameterized word length and depth
- FIFO full and empty flags
- Statically programmable almost full flag to indicate when the FIFO core reaches a specific level, usually when writing into the FIFO
- Statically programmable almostempty flag to indicate when the FIFO core reaches a specific level, usually when reading from the FIFO
- Global reset of the FIFO address pointers and flag logic
- Dual-port synchronous FIFO

Description

The Actel FIFO cores use the 3200DX and MX 32x8 or 64x4 dual-port RAM cells. Addresses are generated internally using counters and token chains to address the RAM (this is transparent to the user). Dedicated read and write address data paths are used in the FIFO architecture. The read and write operations are totally independent and can be performed simultaneously.

The WIDTH (word length) and DEPTH (number of words) have continuous values but the choice of WIDTH limits the choice of DEPTH and vice versa.

The asynchronous clear signal, *Aclr*, can be active low or active high (low is the default option and should be used for all synchronous elements in the two supported families). When the asynchronous clear is active, all internal registers used to determine the current FIFO read and write addresses (counters and token chains) are reset to 0'.

The FIFO is now in an empty state; the RAM content is not affected. When power is first applied to the FIFO, the FIFO must be initialized with an asynchronous clear cycle to reset the internal address pointers.

The full flag signal, *FF,* is optional and is available only for the High-Speed Flag (FFIFO) and the Medium-Speed Flag (MFFIFO) variations. The *FF* signal is active high only (if selected) and indicates when the FIFO is full. The signal is asserted high on the rising (RISE) or falling (FALL) edge of the clock signal *Clock* with no delay.

The empty flag signal, *EF,* is optional and is available only for the High-Speed Flag (FFIFO) and the Medium-Speed Flag (MFFIFO) variations. The *EF* signal is active low only (if selected) and indicates when the FIFO is empty. The signal is asserted low on the rising (RISE) or falling (FALL) edge of the clock signal *Clock* with no delay.

The write enable signals, *WE* and *WEF*, and read enable signals, *RE* and *REF*, are active high requests for writing into and reading out of the FIFO respectively. The *WE* and *RE* signals only control the logic associated with the FIFO write and read address pointers. The *WEF* and *REF* signals control the logic implementing the different flags. The *WE* and *WEF* signals should be logically driven by the same logic outside the FIFO core. The same behavior applies to the *RE* and *REF* signals as well. For SX and SX-A there are only the RE and WE ports.

When *WE* is asserted high and *FF* is asserted low (not full), the write cycle is initiated and Data are written into the FIFO. When *WE* is asserted high and *FF* is asserted high (full), the FIFO behavior is undefined. When *RE* is asserted high and *EF* is asserted high (empty), the read cycle is initiated and Q is read from the FIFO. When *RE* is asserted high and *EF* is asserted low (empty), the FIFO behavior is undefined. When *RE* and *WE* are asserted high at the same time, Data are written into the FIFO and Q is read from the FIFO simultaneously. The read and write operations are fully synchronous with respect to the clock signal *Clock*.

The FIFO function offers a parameterizable almost-full flag, *AFF*. The *AFF* flag is asserted high when the FIFO contains aff_val words or more as defined by the parameter AFF_VAL. Otherwise, *AFF* is asserted low. The aff_val value is a parameter to the core, and thus logic is built at generation time to realize the almost-full flag function.

The FIFO function offers a parameterizable almost-empty flag, *AEF*. The *AEF* flag is asserted low when the FIFO contains aef_val words or less as defined by the parameter AEF_VAL. Otherwise, *AEF* is asserted high. The aef_val value is a parameter to the core, and thus logic is built at generation time to realize the almost-empty flag function.

Synchronous Dual Port Fifo with Flags I/O Description

Table 162 · I/O Description

Port Name	Size	Type	Req./Opt.	Function
Data	WIDTH	Input	Req.	Input Data
WE	1	Input	Req.	Write Enable with the FIFO only (noflag)
RE	1	Input	Req.	Read Enable with the FIFO only (no flag)
WEF	1	Input	Req.	Write enable associated with the flag logic only (for DX/MX)
REF	1	Input	Req.	Read enable associated with the flag logic only (for DX/MX)
Clock	1	Input	Req.	Write and read clock

Port Name	Size	Type	Req./Opt.	Function
Q	WIDTH	Output	Req.	Output Data
FF	1	Output	Req.	Full Flag
EF	1	Output	Req.	Empty Flag
AFF	1	Output	Optional	Almost Full Flag
AEF	1	Output	Optional	Almost Empty Flag

Synchronous Dual Port Fifo with Flags Parameter Description

Table 163 · Parameter Description

Parameter	Value	Function
WIDTH	Width	Word length of Data and Q
DEPTH	Depth	Number of FIFO words
FF_POLOARITY	**1** 2	FF can be active high or not
EF_POLARITY	**0** 2	EF can be active low or not used
AFF_VAL	aff_val (see parameter rules)	AFF value (not used if aff_val is 0
AEF_VAL	aef_val (see parameter rules	AEF value (not used if aef_val is 0
CLK_EDGE	**RISE** FALL	Clock can be rising or falling

Table 164 · Implementation Parameters - MX/DX

Parameter	Value	Description
LPMTYPE	LPM_FIFO_DQ	Generic FIFO category
LPM_HINT	FFIFO	High speed FIFO with flags

Parameter	Value	Description
	MFFIFO	Medium speed FIFO with flags

Table 165 · Implementation Paramters - SX/SX-A

Parameter	Value	Description
LPM_HINT	FFIFOSX	Synchronous FIFO with no flags

Table 166 · Fan-in Parameters

Parameter	Value	Description
RAMFANIN	**AUTO** MANUAL	See Fan-in Control section in online help

Parameter Rules

If RCLK_EDGE is NONE (Asynchronous mode), then RE_POLARITY must be 2 (not used)

Synchronous Dual Port Fifo with Flags Implementation Rules / Timing Diagrams

Table 167 · Timing Waveform Terminology

Term	Description
t_{ckhl}	Clock high/low period
t_{rp}	Reset pulse width
t_{wesu}	Write enable setup time
t_{resu}	Read enable setup time
t_{adsu}	Data setup time
t_{rco}	Data valid after clock high/low
t_{co}	Flip-flop to clock output

Term	Description
t_{rao}	Data valid after read address has changed

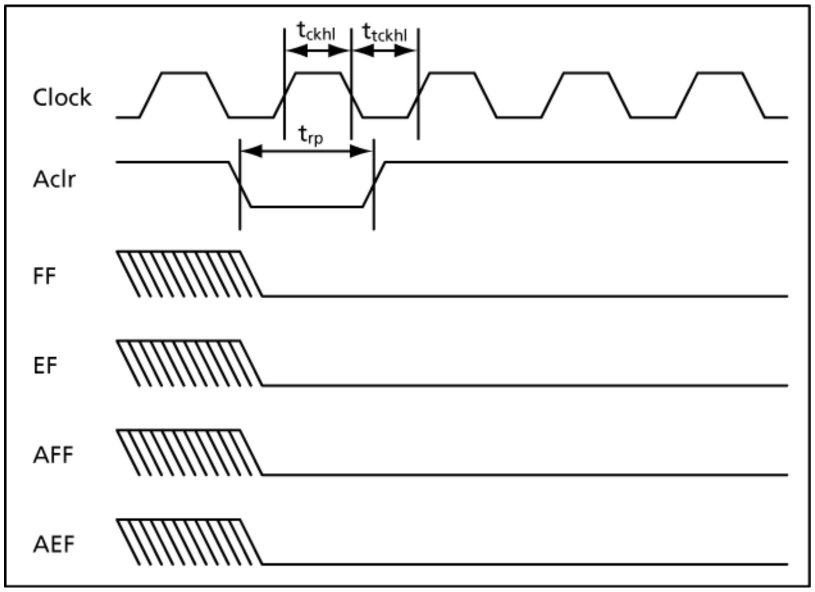


Figure 22 · Reset Cycle

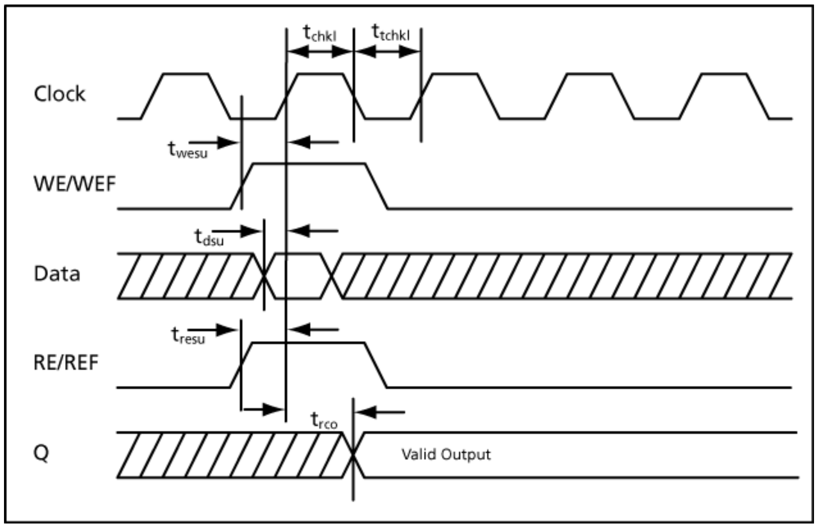


Figure 23 · Write and Read Cycle

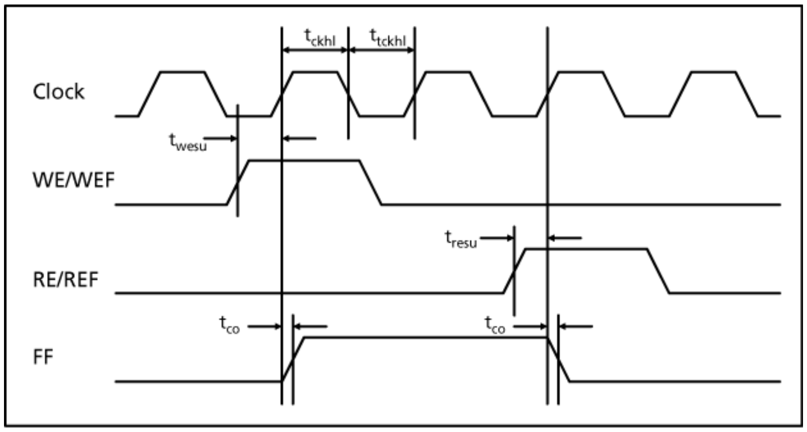


Figure 24 · Full FIFO Timing Diagram

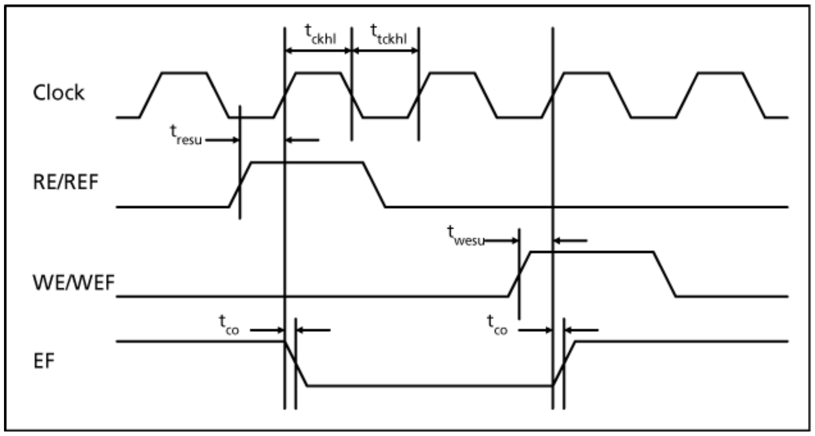


Figure 25 · Empty FIFO Timing Diagram

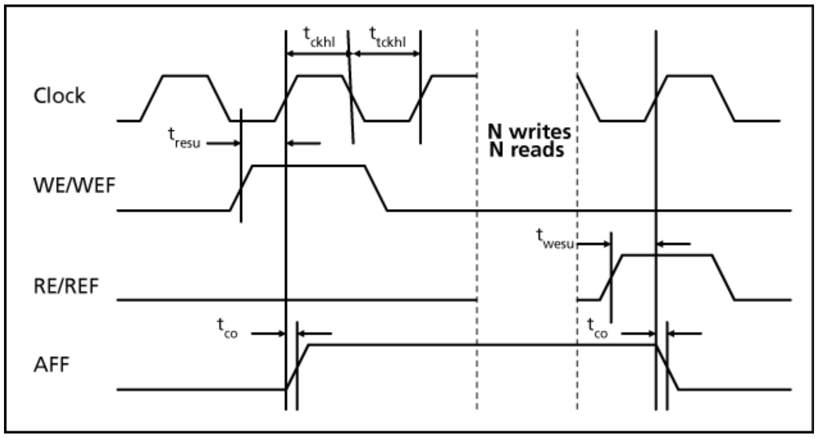
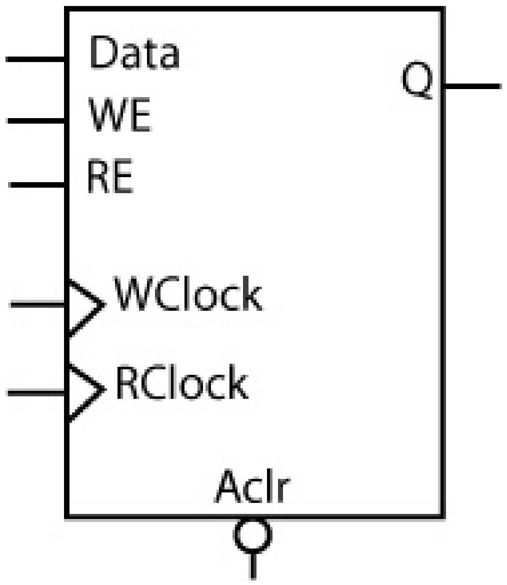


Figure 26 · Almost Full FIFO Timing Diagram

Synchronous Dual Port FIFO without Flags Summary



Supported Families

eX, SX-A, SX

Related Topics

Synchronous Dual Port FIFO without Flags I/O Description

Synchronous Dual Port FIFO without Flags Parameter Description

Synchronous Dual Port FIFO without Flags Implementation Rules / Timing Diagrams

Key Features

- On-chip RAM
- Parameterized word length and depth
- Dual-port synchronous RAM architecture
- Dual-port synchronous FIFO (write and read clocks are separated) with no static flag logic
- Global reset of FIFO address pointers
- Behavioral simulation RTL in VHDL and Verilog

Description

The Actel FIFO cores use the 3200DX and MX 32x8 or 64x4 on-chip RAM cells. The core configurator generates addresses internally using counters and token chains to address the RAM blocks (transparent to the user). Dedicated read and write address data paths are used in the FIFO architecture. The read and write operations are independent and can be performed simultaneously.

The WIDTH (word length) and DEPTH (number of words) have continuous values but the choice of WIDTH limits the choice of DEPTH and vice versa.

The asynchronous clear signal, *Aclr,* can be active low or active high (low is the default option and is the preferred use for all synchronous elements in the two supported families). When the asynchronous clear is active, all internal registers used to determine the current FIFO read and write addresses (counters and token chains) are reset to '0'. The FIFO is now in an empty state; the RAM content is not affected. When power is first applied to the FIFO, the FIFO must be initialized with an asynchronous clear cycle to reset the internal address pointers.

The write enable *WE* and read enable *RE* signals are active high request signals for writing into and reading out of the FIFO respectively. The *WE* and *RE* signals only control the logic associated with the FIFO write and read address pointers.

When *WE* is asserted high, the write cycle is initiated, and Data are written into the FIFO. The design using the FIFO is responsible for handling the full and empty states of the FIFO core.

When *RE* is asserted high, the read cycle is initiated, and Q is read from the FIFO. The design using the FIFO is responsible for handling the full and empty states of the FIFO core.

Synchronous Dual Port FIFO without Flags I/O Description

Table 168 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Data	WIDTH	Input	Req.	Input Data
WE	1	Input	Req.	Write Enable
RE	1	Input	Req.	Read Enable
WClock	1	Input	Req.	Write clock
RClock	1	Input	Req.	Read clock
Q	WIDTH	Output	Req.	Output Data

Synchronous Dual Port FIFO without Flags Parameter Description

Table 169 · Parameter Description

Parameter	Value	Function

Parameter	Value	Function
WIDTH	Width	Word length of Data and Q
DEPTH	Depth	Number of FIFO words
WCLK_EDGE	**RISE** FALL	WClock can be rising or falling
RCLK_EDGE	**RISE** FALL	RClock can be rising falling

Table 170 · Implementation Parameters - MX/DX

Parameter	Value	Description
LPMTYPE	LPM_FIFO_DQ	Generic FIFO category
LPM_HINT	SFIFO	Synchronous FIFO with no flags

Table 171 · Implementation Parameters - SX/SX-A

Parameter	Value	Description
LPM_HINT	SFIFOSX	Synchronous FIFO with no flags

Table 172 · Fan-in Parameters

Parameter	Value	Description
RAMFANIN	**AUTO** MANUAL	See Fan-In Control

Synchronous Dual Port FIFO without Flags Implementation Rules / Timing Diagrams

Table 173 · Timing Waveform Terminology

Term	Description

Term	Description
t_{ckhl}	Clock high/low period
t_{rp}	Reset pulse width
t_{wesu}	Write enable setup time
t_{resu}	Read enable setup time
t_{dsu}	Data setup time
t_{rco}	Data valid after clock high/low
t_{co}	Flip-flop to clock output

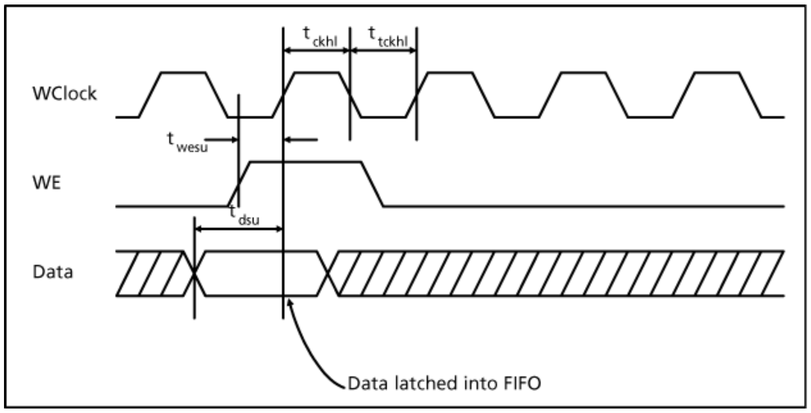


Figure 27 · FIFO Write Cycle

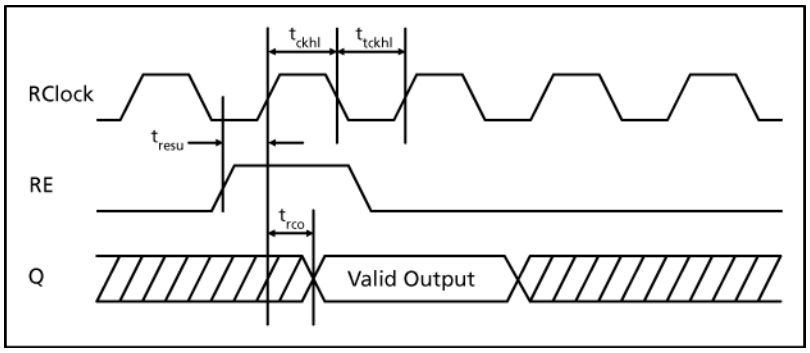


Figure 28 · FIFO Read Cycle

Synchronous FIFO for IGLOO, ProASIC3, SmartFusion and Fusion Summary

The software automatically cascades FIFO blocks to create wider memories by choosing the most efficient aspect ratio. It also handles the grounding of unused bits.

Related Topics

Synchronous FIFO for IGLOO, ProASIC3, SmartFusion and Fusion Functionality

Synchronous FIFO for IGLOO, ProASIC3, SmartFusion and Fusion Description

Synchronous FIFO for IGLOO, ProASIC3, SmartFusion and Fusion Implementation Rules / Timing Diagrams

Specify the following parameters to create a FIFO:

Almost Full/Empty Flags:

Choose from Static, Dynamic and No flags. If you choose No flags, the software grounds AFVAL, AEVAL and AFULL, and AEMPTY signals do not appear as ports on the top level. If you choose Static Flags the software configures the AFVAL and AEVAL accordingly. For Dynamic Flags you can drive the AFVAL and AEVAL through a signal and can change the thresholds dynamically. However, care must be taken that the functionality of the AFVAL and AEVAL is fully understood. For more information on these signals please refer to the section on Using FIFO Flags.

Pipeline

You can choose to have a pipelined or non-pipelined read. The software configures the PIPE signal accordingly. This is a static selection and cannot be changed dynamically by driving it with a signal.

Write/Read Depth:

The core configurator supports the generation of FIFO having a write or read depth between 1 and 4096.

Write/Read Width:

The core configurator supports the generation of RAM having a write or read width between 1 and 576.

Read and Write Clock Polarities:

The core configurator instantiates inverters as necessary to achieve the requested polarity.

Read and Write Enable Polarities:

The core configurator instantiates inverters as necessary to achieve the requested polarity.

Continue Counting Read Counter After FIFO is Empty (ESTOP)

Selecting this option means the software will configure the FIFO in such a way that ESTOP is tied low and counter will keep counting even after FIFO is empty.

Continue Counting Write Counter After FIFO is Full (FSTOP)

Selecting this option means the software will configure the FIFO in such a way that FSTOP is tied low and counter will keep counting even after FIFO is full.

For more information on the above two options refer to the ESTOP, FSTOP Usage section.

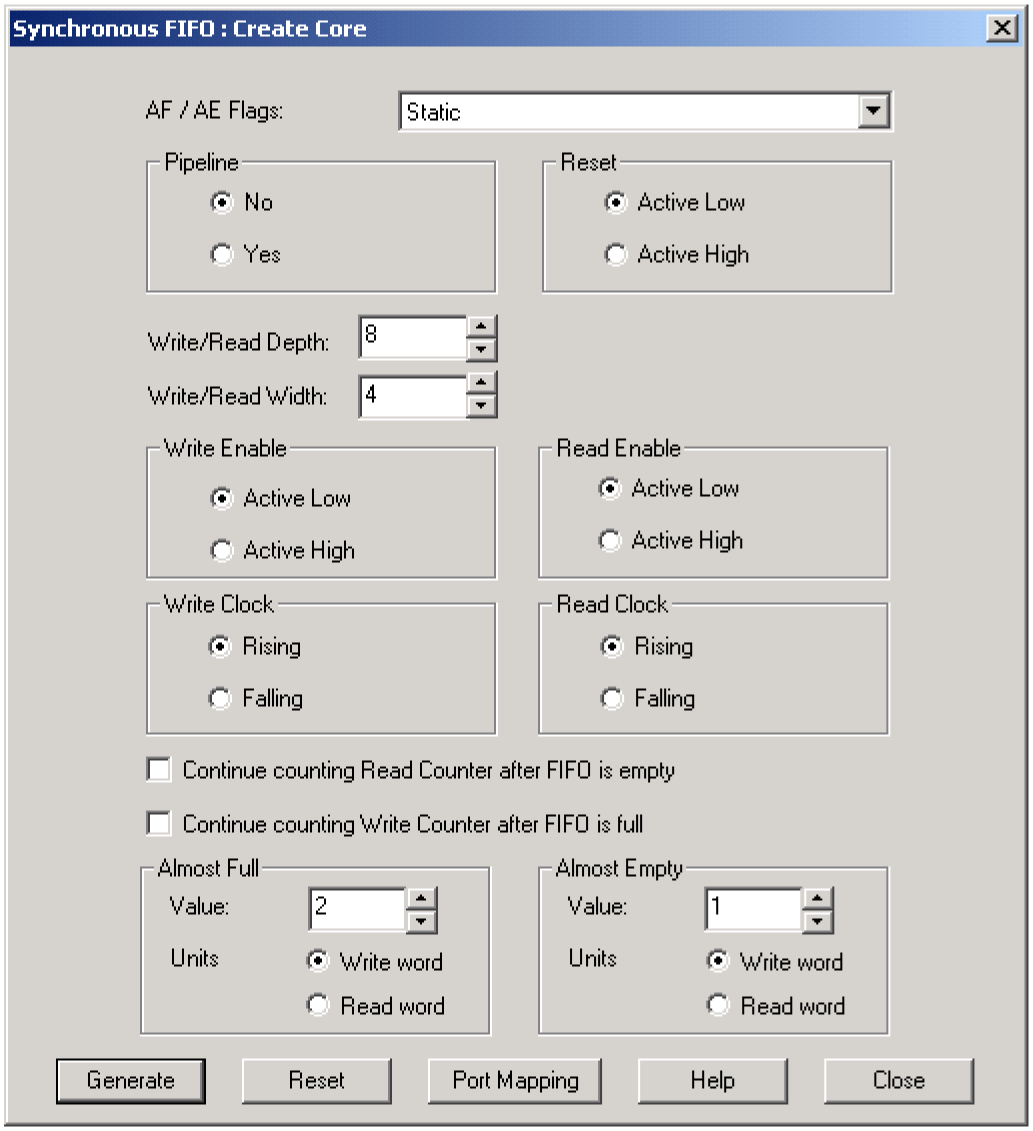
Almost Full Value/Units

This choice is applicable only in the Static Almost Full/Empty selection.

Almost Empty Value/Units

This choice is applicable only in the Static Almost Full/Empty selection.

For more information on these choices please refer to the FIFO Flags Usage section.



Synchronous FIFO : Create Core

AF / AE Flags: Static

Pipeline: [✓ No] Yes

Reset: [✓ Active Low] Active High

Write/Read Depth: 8

Write/Read Width: 4

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 29 · Synchronous FIFO Configuration Screen

Synchronous FIFO for IGLOO, ProASIC3, SmartFusion and Fusion Functionality

Using ESTOP and FSTOP

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). These are configuration pins that should not be dynamically reconfigured. The software configures these signals based on your selection.

The FIFO counters in ProASIC3E start the count from 0, reach the maximum depth for the configuration (e.g. 511 for a 512X9 configuration), and then re-start from 0. A potential application for the ESTOP, where the read counter keeps counting would be, writing to the FIFO once and reading the same content over and over, without doing a write again.

A typical user would not need to use these features and should leave these options un-checked in the GUI.

Using FIFO Flags

The AEVAL and AFVAL pins are used to specify the almost empty and almost full threshold values, respectively. They are 12-bit signals. In order to handle different read and write aspect ratios, the values specified by the AEVAL and AFVAL pins are to be interpreted as the address of the last word stored in the FIFO. The FIFO actually contains separate write address (WADDR) and read address (RADDR) counters. These counters calculate the 12-bit memory address that is a function of WW and RW, respectively. 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 raised. Likewise, whenever the difference between WADDR and RADDR is less than or equal to AEVAL, the AEMPTY output is raised.

Synchronous FIFO for IGLOO, ProASIC3, SmartFusion and Fusion Description

Signals in Generated Netlists

Data: Input Data for the FIFO

Q: Output Data for FIFO

FULL, EMPTY: Full and Empty FIFO flags

AFULL, AEMPTY: Programmable Almost Full and Almost Empty flags (available only in static/dynamic flags configuration)

AFVAL, AEVAL: Signals to specify the thresholds for AFULL and AEMPTY (available only in dynamic flag configuration)

WClock, RClock: Write and Read Clocks

WE, RE: Write and Read Enables

RESET: FIFO Reset

Synchronous FIFO for IGLOO, ProASIC3, SmartFusion and Fusion Implementation Rules / Timing Diagrams

Caveats to FIFO generation

- Depth cascading is currently not supported. Therefore the maximum depth supported is only 4096.

- It supports wide cascading up to 64 blocks.
- The core configurator does not generate a FIFO based on a specific device. It is your responsibility to make sure the FIFO fits physically on the device.
- Dynamic configuration of any signal with exception of AFVAL/AEVAL is not supported.
- The core configurator will give a configuration error for unsupported configurations.
- WBLK and RBLK are always grounded by the configurator, which means the FIFO block always remains enabled. You must control the FIFO with WEN and REN.

FIFO Using Distributed Memory for ProASICPLUS Summary

Related Topics

FIFO Using Distributed Memory for ProASICPLUS I/O Description

FIFO Using Distributed Memory for ProASICPLUS Parameter Description

Key Features

- Parameterized word length and depth
- Asynchronous FIFO
- Asynchronous, or synchronous write
- Rising-edge triggered or level-sensitive
- Supported netlist formats: VHDL and Verilog

Description

Distributed memory can be generated as a two-port asynchronous register file or as an asynchronous FIFO. Distributed memories are made up of the logic tiles of the device. These memory files are netlists consisting of logic tiles and do not use to embedded memory cells.

FIFO Using Distributed Memory for ProASICPLUS I/O Description

Table 174 · I/O Description

Port Name	Size	Type	Req/Opt?	Function
wData<i>	1	Input	Req.	Input (Write) Data (i = 0 .. WIDTH-1)
INIT	1	Input	Req.	FIFO initialization

Port Name	Size	Type	Req/Opt?	Function
WR	1	Input	Req.	Write Clock/Pulse (rising edge triggered or level sensitive)
RD	1	Input	Req.	Read Clock/Pulse (rising edge triggered or level sensitive)
rData<i>	1	Output	Req.	Output (Read) Data (i = 0 .. WIDTH-1)
full	1	Output	Req.	Full Flag
empty	1	Output	Req.	Empty Flag

FIFO Using Distributed Memory for ProASICPLUS Parameter Description

Table 175 · Parameter Description

Parameter	Value	Function
WIDTH	See Parameter Rules	Word length input/output data
DEPTH	2..64	Number of words
TRIGGER	**edge**, level	Select between rising- edge triggered and level-sensitive write clock

Table 176 · Implementation Parameters

Parameter	Value	Description
LPMTYPE	LPM_DIST_FIFO	Generic distributed FIFO category
LPM_HINT	FIFO_DISTH<#>	Horizontal Orientation # represents the part number and can be 050, 130, 180, 270 for 500K

Parameter	Value	Description
		150, 300, 450, 600, 750, 1000 for PA
	FIFO_DISTV<#>	Vertical Orientation

Table 177 · Parameter Rules

Device	Orientation	Parameter Rules
A500K050	Horizontal	WIDTH = 2..62, DEPTH = 2..36
	Vertical	WIDTH = 2..94, DEPTH = 2..23
A500K130	Horizontal	WIDTH = 2..78, DEPTH = 2..62
	Vertical	WIDTH = 2..158, DEPTH = 2..29
A500K180	Horizontal	WIDTH = 2..94, DEPTH = 2..74
	Vertical	WIDTH = 2..190, DEPTH = 2..36
A500K270	Horizontal	WIDTH = 2..118, DEPTH = 2..80
	Vertical	WIDTH = 2..222, DEPTH = 2..45
APA075	Horizontal	WIDTH = 2..22, DEPTH = 2..64
	Vertical	WIDTH = 2..62, DEPTH = 2..48
APA150	Horizontal	WIDTH = 2..46, DEPTH = 2..49
	Vertical	WIDTH = 2..126, DEPTH = 2..16
APA300	Horizontal	WIDTH = 2..62, DEPTH = 2..49
	Vertical	WIDTH = 2..126, DEPTH = 2..23
APA450	Horizontal	WIDTH = 2..62, DEPTH = 2..74
	Vertical	WIDTH = 2..190, DEPTH = 2..23
APA600	Horizontal	WIDTH = 2..94, DEPTH = 2..80

Device	Orientation	Parameter Rules
	Vertical	WIDTH = 2..222, DEPTH = 2..36
APA750	Horizontal	WIDTH = 2..126, DEPTH = 2..80
	Vertical	WIDTH = 2..254, DEPTH = 2..49
APA1000	Horizontal	WIDTH = 2..158, DEPTH = 2..80
	Vertical	WIDTH = 2..350, DEPTH = 2..62

Axcelerator RAM

Related Topics

Axcelerator RAM Functionality

Axcelerator RAM I/O Description

Axcelerator RAM Parameter Description

Axcelerator RAM Implementation Rules

Key Features

- Parameterized word length and depth
- Dual-port synchronous RAM architecture
- Independent Read/Write sizes
- Active High/Low enable
- Active High/Low Read and Write clocks
- Non-pipelined (synchronous - one clock edge)/ Pipelined (synchronous - two clock edges) Read
- Port mapping
- Memory Editor

Axcelerator RAM Functionality

Axcelerator provides dedicated blocks of RAM. Each block has a read port and a write port. Both ports are configurable to any size from 4Kx1 to 128x36; thereby, allowing built-in bus width conversion (see SRAM Port Aspect Ratio table below). Each port is completely independent and fully synchronous.

Table 178 · SRAM Port Aspect Ratio

Width	Depth	ADDR Bus	Data Bus
1	4096	ADDR [11:0]	DATA [0]
2	2048	ADDR [10:0]	DATA [1:0]
4	1024	ADDR[9:0]	DATA[3:0]
9	512	ADDR[8:0]	DATA[8:0]
18	256	ADDR[7:0]	DATA[17:0]
36	128	ADDR[6:0]	DATA[35:0]

Modes

The three major modes available for read and write operations are:

- Read Non-pipelined (synchronous - one clock edge) The read address is registered on the read port clock edge and data appears at read-data after the RAM access time (when all RENs are high, approximately 4.5ns). The setup time of the read address and read enable are minimal with respect to the read clock. Setting the Pipeline to OFF enables this mode.
- Read Pipelined (synchronous - two clock edges) The read-address is registered on the read port clock edge and the data is registered and appears at read-data after the second read clock edge. Setting the Pipeline to ON enables this mode.
- Write (synchronous - one clock edge) On the write clock edge, the write data are written into the USRAM at the write address (when all WENs are high). The setup time of the write address, write enables and write data are minimal with respect to the read clock.

Cascading Blocks

Blocks can be cascaded to create larger sizes. The software performs all the necessary cascading for achieving the desired configuration. To achieve good performance, all cascaded RAM blocks must fit within one RAM column of the selected device. Cascading RAM blocks deep is possible only up to the capacity of one RAM column.

However, if the specified configuration exceeds one RAM column, the core configurator software tries to cascade the RAM wide, up to the available RAM Blocks in the device. This will result in poorer performance as the RAM blocks are not physically close to one another.

The maximum WIDTH (word length) value is 65,536. The maximum DEPTH (number of words) value is 576.

The Read/Write Width/Depth can be different but the aspect ratio should be same for both. For example:

Read Width * Read Depth == Write Width * Write Depth

The write enable (WE) and read enable (RE) signals are active high or low request signals for writing and reading, respectively; you may choose not to use them. When none is selected for an enable, that operation remains enabled all the time.

For example, if WEN is chosen as none, then write operation of the RAM is enabled all the time.

The RCLK and WCLK pins have independent polarity selection.

Conflict Resolution

There is no special hardware for handling read and write operations at the same addresses.

Axcelerator RAM I/O Description

Table 179 · I/O Description

Name	Type	Required/Optional	Description
Data	IN	Req	Write Data Port
WAddress	IN	Req	Write Address Bus
WE	IN	Opt	Write Enable
WClock	IN	Req	Write Clock
Q	OUT	Req	Read Data Port
RAddress	IN	Req	Read Address Bus
RE	IN	Opt	Read Enable
RClock	IN	Req	Read Clock

Axcelerator RAM Parameter Description

Table 180 · Parameter Description

Parameter	Value	Function
WWIDTH	Write Width	Word length of Data

Parameter	Value	Function
WDEPTH	Write Depth	Number of Write Words
RWIDTH	Read Width	Word length of Q
RDEPTH	Read Depth	Number of Read Words
WE_POLARITY	**1** 0 2	Write Enable Polarity
RE_POLARITY	**1** 0 2	Read Enable Polarity
WCLK_EDGE	**RISE** FALL	Write Clock Edge
RCLK_EDGE	**RISE** FALL	Read Clock Edge
PIPE	**NO** YES	Read Pipeline
DEVICE	75 150 300 600 1000 (May change)	Target Device, to determine blocks available for cascading

Table 181 · Signal Description

Name	Type	Required/Optional	Description
Data	IN	Req	Write Data Port
WAddress	IN	Req	Write Address Bus
WE	IN	Opt	Write Enable
WClock	IN	Req	Write Clock
Q	OUT	Req	Read Data Port
RAddress	IN	Req	Read Address Bus
RE	IN	Opt	Read Enable
RClock	IN	Req	Read Clock

Axcelerator RAM Implementation Rules

Table 182 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_RAM	Generic Dual-Port RAM Category

Table 183 · Parameter Rules

Device	Parameter Rules
Axcelerator	RWIDTH*RDEPTH == WWIDTH*WDEPTH

Axcelerator EDAC RAM Module

Please refer to the Using EDAC RAM for RadTolerant RTAX-S FPGAs and Axcelerator FPGAs application note, available on the Actel website (http://www.actel.com), for a complete explanation of the EDAC RAM module.

Key Features

- 8-, 16-, 32-bit word width
- Background refresh and variable refresh rate
- EDAC RAM module supports READ and WRITE clocks from the same clock source OR separate READ and WRITE clocks
- EDAC RAM Encoder/Decoder supports correcting one error and detecting two errors, with a coding efficiency of 44-66%
- Variable RAM depth support from 256 to 4k words

The Error Detection and Correction (EDAC) RAM module is designed to provide a transparent RAM interface that supports EDAC. When you use the Design Block to generate an EDAC RAM module, it creates a top level for the EDAC RAM, an Axcelerator RAM block, and the "edaci" module, which handles all the EDAC functionality.

RAM Content Manager Summary

Related Topics

RAM Content Manager Functionality

RAM Content Manager Implementation Rules

The RAM Content Manager enables you to specify the contents of your memory so that you can avoid the simulation cycles required for initializing the memory, which reduces simulation runtime. For Fusion families, the RAM Content Manager also enables you to specify the RAM content that will be loaded into the Flash Memory System Builder (see Fusion RAM with Initialization for more details).

The RAM core generator takes away much of the complexity required in the generation of large RAMs that utilize one or more RAM blocks on the device. The configurator uses one or more memory blocks to generate a RAM matching your configuration. In addition, it also creates the surrounding cascading logic.

The configurator cascades RAM blocks in three different ways.

- Cascaded deep (e.g. 2 blocks of 4096x1 to create a 8192x1)
- Cascaded wide (e.g. 2 blocks of 4096x1 to create a 4096x2)
- Cascaded wide and deep (e.g. 4 blocks of 4096x1 to create a 8192x2, in a 2 blocks width-wise by 2 blocks depth-wise configuration)

You specify memory content in terms of your total memory size. The configurator must partition your memory file appropriately such that the right content goes to the right block RAM when multiple blocks are cascaded.

Supported Formats

The Actel implementation of these formats interprets data sets in bytes. This means that if the memory width is 7 bits, every 8th bit in the data set is ignored. Or, if the data width is 9, two bytes are assigned to each memory address and the upper 7 bits of each 2-byte pair are ignored.

The following examples illustrate how the data is interpreted for various word sizes:

For the given data: FF 11 EE 22 DD 33 CC 44 BB 55 (where 55 is the MSB and FF is the LSB)

For 32-bit word size:

```
 0x22EE11FF (address 0)
 0x44CC33DD (address 1)
0x000055BB (address 2)
```

For 16-bit word size:

```
0x11FF (address 0)
0x22EE (address 1)
0x33DD (address 2)
0x44CC (address 3)
0x55BB (address 4)
```

For 8-bit word size:

```
 0xFF (address 0)
0x11 (address 1)
 0xEE (address 2)
 0x22 (address 3)
0xDD (address 4)
0x33 (address 5)
0xCC (address 6)
```

```
0x44 (address 7)
0xBB (address 8)
0x55 (address 9)
```

For 9-bit word size:

```
0x11FF -> 0x01FF (address 0)
0x22EE -> 0x00EE (address 1)
0x33DD -> 0x01DD (address 2)
 0x44CC -> 0x00CC (address 3)
 0x55BB -> 01BB (address 4)
```

Notice that for 9-bit, that the upper 7-bits of the 2-bytes are ignored.

Intel-Hex Record Format

A standard format created by Intel. Memory contents are stored in ASCII files using hexadecimal characters. Each file contains a series of records (lines of text) delimited by new line, '\n', characters and each record starts with a ':' character. For more information regarding this format, refer to the Intel-Hex Record Format Specification document available on the web (search Intel Hexadecimal Object File for several examples).

The Intel Hex Record is composed of five fields and arranged as follows:

```
:llaaaatt[dd...]cc
```

Where:

- : is the start code of every Intel Hex record
- ll is the byte count of the data field
- aaaa is the 16-bit address of the beginning of the memory position for the data. Address is big endian.
- tt is record type, defines the data field:

- 00 data record
- 01 end of file record
- 02 extended segment address record
- 03 start segment address record (ignored by Actel tools)
- 04 extended linear address record
- 05 start linear address record (ignored by Actel tools)

- [dd...] is a sequence of n bytes of the data; n is equivalent to what was specified in the ll field
- cc is a checksum of count, address, and data

Example Intel Hex Record:

```
:0300300002337A1E
```

Motorola S-Record Format

This format uses ASCII files, hex characters, and records to specify memory content in much the same way that Intel-Hex does. Refer to the Motorola S-record description document for more information on this format (search Motorola S-record description for several examples). The RAM Content Manager uses only the S1 through S3 record types; the others are ignored.

The major difference between Intel-Hex and Motorola S-record is the record formats, and some extra error checking features that are incorporated into Motorola S.

In both formats, memory content is specified by providing a starting address and a data set. The upper bits of the data set are loaded into the starting address and leftovers overflow into the adjacent addresses until the entire data set has been used.

The Motorola S-record is composed of 6 fields and arranged as follows:

```
Stllaaaa[dd...]cc
```

Where:

- S is the start code of every Motorola S-record
- t is record type, defines the data field
- ll is the byte count of the data field
- aaaa is a 16-bit address of the beginning of the memory position for the data. Address is big endian.
- [dd...] is a sequence of n bytes of the data; n is equivalent to what was specified in the ll field
- cc is the checksum of count, address, and data

Example Motorola S-Record:

```
S10a0000112233445566778899FFFA
```

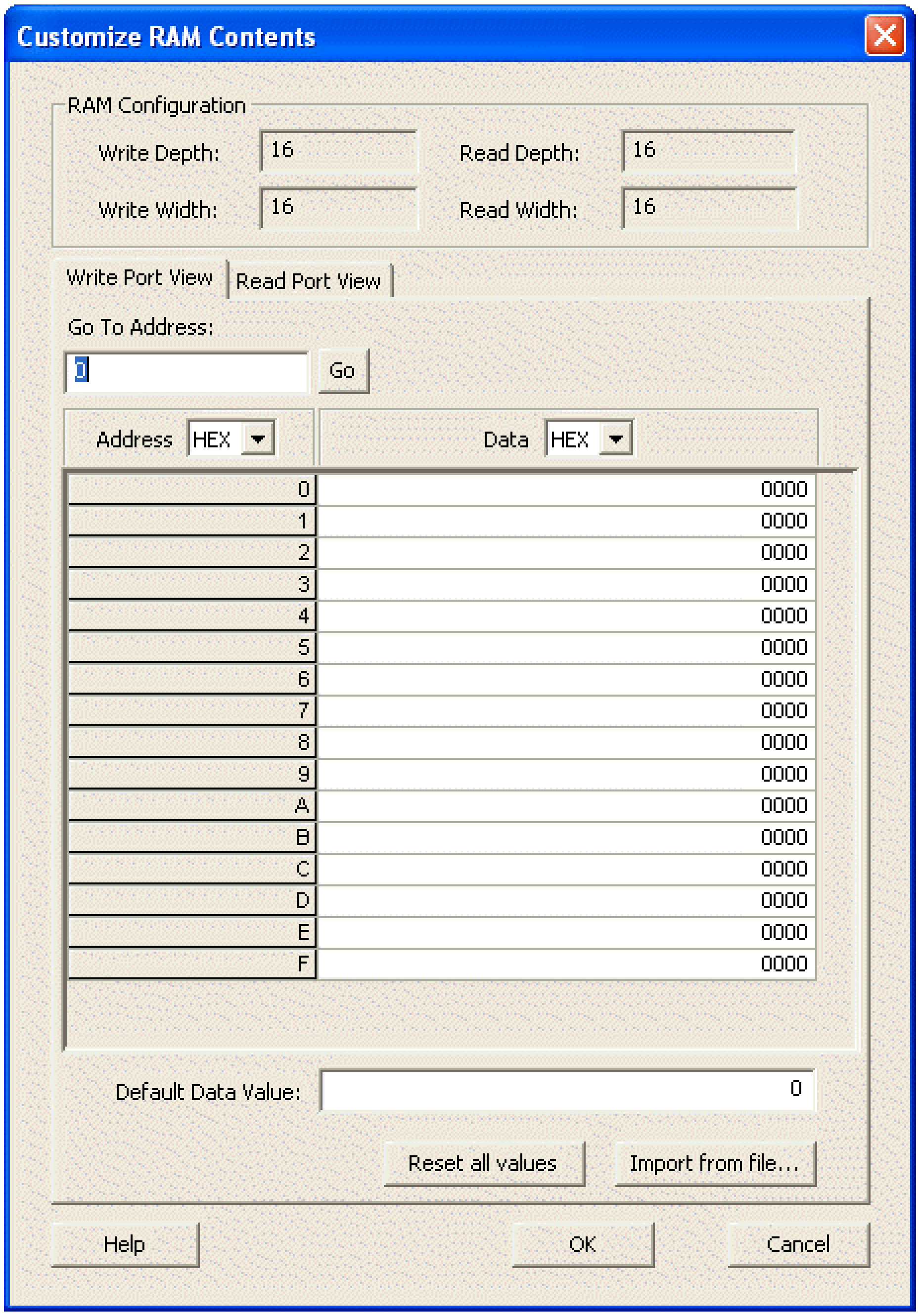
Where 11 is the LSB and FF is the MSB.

RAM Content Manager Functionality

Using the RAM Content Manager is only possible if the device family supports the RAM Content Manager features (IGLOO, ProASIC3, SmartFusion, Fusion, and Axcelerator).

To open the RAM Content Manager:

1. From the **Options** menu, choose **Workspace Settings** and set your device family to **Fusion**, **ProASIC3**, **ProASIC3E**, or **Axcelerator**. Click **OK**.
2. Click **RAM** in the Core Catalog to display the list of RAM types available for your device.
3. Double-click **Synchronous RAM**, Dual Port RAM, or any of the RAM cores for the families listed above to create a new RAM block. Specify your RAM settings (set your Read and Write Depth and Width), select the **Initialize RAM** checkbox, and then click **Customize RAM Content**. The RAM Content Manager appears, as shown in the figure below.



RAM Configuration

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

Go To Address: 0

Address HEX	Data HEX
0	0000
1	0000
2	0000
3	0000
4	0000
5	0000
6	0000
7	0000
8	0000
9	0000
A	0000
B	0000
C	0000
D	0000
E	0000
F	0000

Default Data Value: 0

Figure 30 · RAM Content Manager

RAM Configuration

Write Depth and Write Width - As specified in the RAM core generator dialog box (not editable).

Read Depth and Read Width - As specified in the RAM core generator dialog box (not editable).

Write Port View / Read Port View

Go To Address - Enables you to go to a specific address in the manager. Each memory block has many addresses; it is often difficult to scroll through and find a specific one. This task is simplified by enabling you to type in a specific address. The number display format (Hex, Bin, Dec) is controlled by the value you set in the drop-down menu above the Address column.

Address - The Address column lists the address of a memory location (you cannot specify the address of a memory location). The drop-down menu specifies the number root for your address list (hexadecimal, binary, or decimal).

Data - Enables you to control the data format and data value in the manager. Click the value to change it.

The RAM Content Manager enables you to Import or Export your files through either port.

Files are imported into whichever view you have selected. Importing files through the Write and Read ports with different aspect ratios results in completely different outcomes for your data.

Note that the dialogs show all data with the MSB down to LSB. For example, if the row showed 0xAABB for a 16-bit word size, the AA would the MSB and BB would be LSB.

Import from file (Write Port View)- Opens the Import Memory Content - Write Port View dialog box; enables you to select a memory content file (Intel-Hex, Motorola S-record) to load through the Write Port. During import, file extensions are set to *.hex for Intel-Hex files and *.s for Motorola S-record files.

Import from file (Read Port View) - Opens the Import Memory Content - Read Port View dialog box; enables you to select a memory content file (Intel-Hex, Motorola S-record) to load through the Read Port. During import, file extensions are set to *.hex for Intel-Hex files and *.s for Motorola S-record files.

Default Data Value - The value given to memory addresses that have not been explicitly initialized (by importing content or editing manually). When changed, all default values in the manager are updated to match the new value. The number display format (Hex, Bin, Dec) is controlled by the value you set in the drop-down menu above the Data column.

Reset All Values - Resets the Data values.

Help - Opens the RAM Content Manager online help.

OK- Closes the manager and saves all the changes made to the memory and its contents.

Cancel - Closes the manager, cancels all your changes in this instance of the manager, and returns the memory back to the state it held before the manager was opened.

RAM Content Manager Implementation Rules

MEMFILE (RAM Content Manager output file)

Transfer of RAM data (from the RAM Content Manager) to test equipment is accomplished via MEM files. The contents of your RAM is first organized into the logical layer and then reorganized to fit the hardware layer. Then it is stored in MEM files that are read by other systems and used for testing.

The MEM files are named according to the logical structure of RAM elements created by the configurator. In this scheme the highest order RAM blocks are named CORE_R0C0.mem, where "R" stands for row and "C" stands for column. For multiple RAM blocks, the naming continues with CORE_R0C1, CORE_R0C2, CORE_R1C0, etc.

The data intended for the RAM is stored as ASCII 1s and 0s within the file. Each memory address occupies one line. Words from logical layer blocks are concatenated or split in order to make them fit efficiently within the hardware blocks. If the logical layer width is less than the hardware layer, two or more logical layer words are concatenated to form one hardware layer word. In this case, the lowest bits of the hardware word are made up of the lower address data bits from the logical layer. If the logical layer width is more than the hardware layer, the words are split, placing the lower bits in lower addresses.

If the logical layer words do not fit cleanly into the hardware layer words, the most significant bit of the hardware layer words is not used and defaulted to zero. This is also done when the logical layer width is 1 in order to avoid having left over memory at the end of the hardware block.

Dual Port RAM for IGLOO, ProASIC3 and Fusion Summary

Supported Families

IGLOO, ProASIC3 and Fusion

Related Topics

Dual Port RAM for IGLOO, ProASIC3 and Fusion Functionality

Dual Port RAM for IGLOO, ProASIC3 and Fusion I/O Description

Dual Port RAM for IGLOO, ProASIC3 and Fusion Parameter Description

Dual Port RAM for IGLOO, ProASIC3 and Fusion Implementation Rules

Key Features

Optimize for High Speed (Width Cascading) or Low Power (Depth Cascading)

You can choose to optimize your RAM for High Speed or Low Power.

If you optimize for low power, the core configurator software evaluates your RAM configuration and attempts to generate a macro with depth cascading.

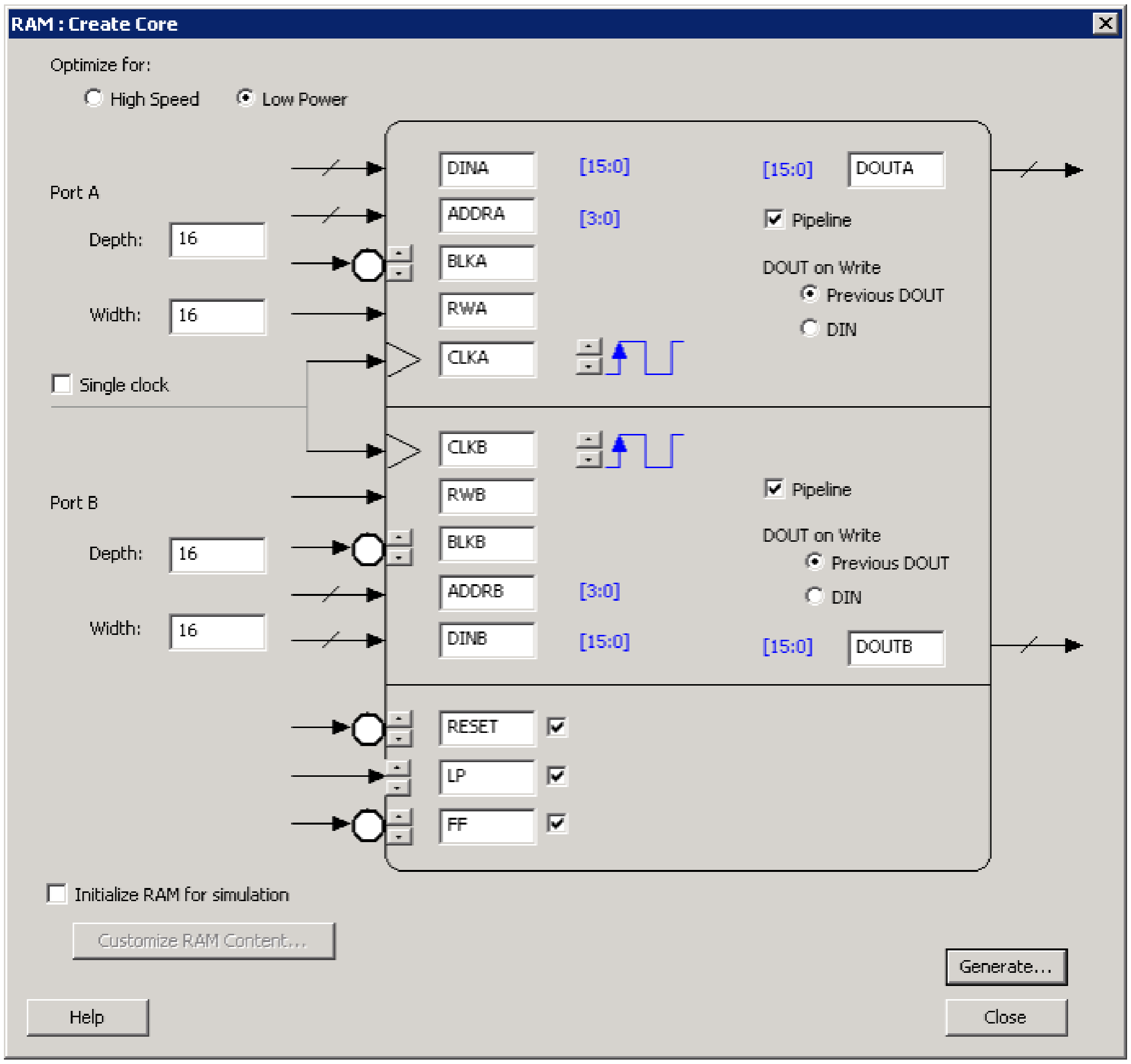
RAM with Initialization

The Fusion RAM with Initialization is nearly identical to the standard RAM, except that it generates extra logic so that it can interface with the Fusion Flash Memory System. The extra logic allows the RAMs to be switched to a x9 configuration during initialization or saved to the Flash Memory. The RAM will dynamically change its user-selected configuration to the x9 configuration to interface more smoothly with the Flash Memory System.

When you configure the RAM With Initialization the user configuration (ie. 64x32) is irrelevant. When you generate the RAM you get 4 ports exported named INITDOUT_0[08:00], INITDOUT_1[08:00], etc. These 4 busses from the RAM need to drive the 4 inputs of the Flash Memory Block.

Additional ports that are exposed for a Fusion RAM with Initialization are shown in the Parameter Description. These ports are meant to be connected to the Flash Memory module.

See the RAM Content Manager for more information.



RAM : Create Core

Optimize for: ( ) High Speed [✓ Low Power]

Port A
Depth: 16
Width: 16

[ ] Single clock

Port B
Depth: 16
Width: 16

DINA [15:0] — [15:0] DOUTA
ADDRA [3:0]
BLKA
RWA
CLKA

[✓] Pipeline
DOUT on Write: [✓ Previous DOUT] ( ) DIN

CLKB
RWB
BLKB
ADDRB [3:0]
DINB [15:0] — [15:0] DOUTB

[✓] Pipeline
DOUT on Write: [✓ Previous DOUT] ( ) DIN

RESET [✓]
LP [✓]
FF [✓]

[ ] Initialize RAM for simulation

Figure 31 · Dual Port RAM Dialog Box

Dual Port RAM for IGLOO, ProASIC3 and Fusion Functionality

The core configurator software automatically cascades RAM blocks to create wider and deeper memories by choosing the most efficient aspect ratio. It also handles the grounding of unused bits. The core configurator software supports the generation of memories that have different Read and Write aspect ratios.

You can create a Dual Port RAM or Two Port RAM in software. A Dual Port RAM has read and write access on both ports while a Two Port RAM allows write access on one port and read access on the other port.

Optimization for High Speed or Low Power

High Speed results in a macro optimized for speed and area (width cascading).

Low Power results in a macro optimized for low power, but uses additional logic at the input and output (depth cascading). Performance for a low power optimized macro may be inferior to that of a macro optimized for speed. Some RAM configurations are not possible with depth cascading (such as 512 x 36), but low power optimization is a priority when the option is selected.

Port A Depth/Width and Port B Depth/Width

The depth range for any port is 1-65536. The width range for any port is 1-576.

In addition to the caveats listed below, (Write Depth * Write Width) must equal (Read Depth * Read Width).

Single Clock (CLKB) or Independent Port A and B Clocks (CLKA and CLKB)

The default for Dual Port RAM is independent clocks (one each for Port A and Port B); click the **Single clock** checkbox to drive CLKA and CLKB with the same clock.

Clock Polarity - Click the up or down arrows to change the active edge of your clock. If you use independent clocks you can select the polarity of both the Port A (CLKA) and Port B (CLKB) clocks. The core configurator software instantiates inverters to achieve the specified polarity.

Block Enables (BLKA and BLKB)

Asserting BLKA when RWA is high reads the RAM at the address (ADDRA) onto the data port (DOUTA).

Asserting BLKA when RWA is low writes the data (DINA) into the RAM at the address (ADDRA).

Asserting BLKB when RWB is high reads the RAM at the address (ADDRB) onto the data port (DOUTB).

Asserting BLKB when RWB is low writes the data (DINB) into the RAM at the address (ADDRB).

Read/Write Mode Control (RWA and RWB)

Use this signal to switch between read or write mode for a given port. LOW = WRITE, HIGH = READ.

Pipeline for Port A and Port B

Click the Pipeline checkbox if you want the software to configure the PIPEA and PIPEB signals to make the output pipelined. This is a static selection and cannot be changed dynamically by driving it with a signal.

If you choose to Initialize RAM, you can customize your RAM content with the RAM Content Manager.

Note: Dual Port RAM configured in INIT mode does not support pass write data to output. This is because in INIT mode, write always occurs on Port A and read occurs on Port B.

LP and FF Inputs

LP (Low Power) Port (Active High): You drive this port during active and idle modes to lower static Icc. When driving this pin, you must deselect the SRAM/FIFO for some determined time (counter value) or for a particular condition in your logic. This port has no effect if the port BLK is de-asserted (BLK port is active low), i.e. if BLK is high, then the LP port value is irrelevant.

FF (Flash*Freeze) Port (Active Low): Connect this port directly to your Flash*Freeze pin (INBUF_FF output) in Flash*Freeze Type 1. It must be inverted when driven by housekeeping logic involving the ULSICC macro in Flash*Freeze Type 2. Connect this port directly to the Flash_Freeze_Enabled port of the Flash*Freeze Management IP if you are using it to implement Flash*Freeze Type 2. Asserting this port ensures that the SRAM/FIFO does not stay in WRITE and/or READ mode when the device enters Flash*Freeze mode.

Dual Port RAM for IGLOO, ProASIC3 and Fusion I/O Description

Table 184 · IGLOO, ProASIC3 and Fusion Dual Port RAM I/O Description

Name	Direction	Description
INITADDR	IN	Address from Flash Memory module
INITDATA[08:00]	IN	Data from Flash Memory module
INIT_CLIENT_i	IN	Write enable signal from the Flash Memory module.This indicates that the data on the INITDATA bus is to be written into the RAM at the INITADDR location. There will be a signal per RAM block used. For example, if the memory configuration required 4 RAM blocks, then there will be 4 of these signals exported. These signals need to be connected to the <client_name>_block_i_DAT_VAL from the Initialization / Flash Memory module.

Name	Direction	Description
INITACTIVE	IN	Needs to be asserted when Initialization is being performed. Switches the aspect ratio to x9 width and changes the RAM to respond to the Initialization interface.
INITDOUT[08:00]	OUT	Data to be saved into the Flash Memory module.
SAVEACTIVE	IN	Needs to be asserted when Save is being performed. Switches the aspect ratio to x9 width and changes the RAM to respond to the Save interface. The address location to be read is also driven from the INITADDR port for SAVE operations.

Dual Port RAM for IGLOO, ProASIC3 and Fusion Parameter Description

The table below lists the Dual Port RAM signals in generated netlists.

Table 185 · Dual Port RAM Parameter Description

Name	Type	Genfile Parameter	Bit/Bus	Description
ADDRA	IN	ADDRESSA_PN	BUS	Address for port A
DINA	IN	DATAA_IN_PIN	BUS	Data in for Port A
BLKA	IN	BLKA_PN	Bit	Block enable for Port A
RWA	IN	RWA_PN	Bit	Signal to switch between Read and Write modes; Low = Write, High = Read
CLKA	IN	CLKA_PN	Bit	Clock for Port A
ADDRB	IN	ADDRESSB_PN	Bus	Address for Port B
DINB	IN	DATAB_IN_PN	Bus	Data in for Port B
BLKB	IN	BLKB_PN	Bit	Block enable for Port B
RWB	IN	RWB_PN	Bus	Signal to switch between Read and Write

Name	Type	Genfile Parameter	Bit/Bus	Description
				modes; Low = Write, High = Read
CLKB	IN	CLKB_PN	Bit	Clock for Port B
CLKAB	IN	CLOCK_PN	Bit	Clock for single clock
DOUTA	OUT	DATAA_OUT_PN	Bus	Data output for Port A
DOUTB	OUT	DATAB_OUT_PN	Bus	Data output for Port B
LP	IN	LP_PN	Bit	Low power input pin
FF	IN	FF_PN	Bit	Flash*Freeze input pin
RESET		RESET		Asynchronous reset

Dual Port RAM for IGLOO, ProASIC3 and Fusion Implementation Rules

Caveats for Dual Port RAM generation

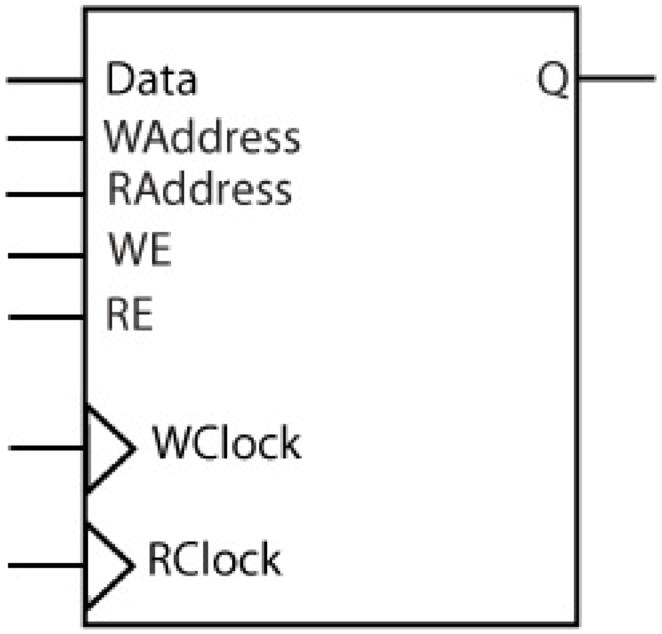
- If a word width of 9 is used for Read, then Write configurations of 1, 2, or 4 will cause the MSB of the output to be undefined. These configurations are not supported. However, configurations that do not use the 9th bit (e.g., a Read width of 512x8 and a Write width of 1024x4) are supported.
- The core configurator only supports depth and width RAM cascading up to 64 blocks.
- The core configurator software does not generate RAM based on a specific device. It is your responsibility to make sure the RAM fits physically on the device. Dynamic configuration of the aspect ratios is supported only in the Fusion RAM with Initialization core.
- The software returns a configuration error for unsupported configurations.

Tips

- Writing different data to the same address using both ports in Dual Port RAM is undefined and should be avoided.
- All unused inputs must be grounded.
- WMODE is ignored during read operation.
- RESET does not reset the memory contents. It resets only the output.

- Writing to and reading from the same address is undefined and should be avoided. When using the RAM4K9 in Two Port mode, care should be taken that Read and Write operations are not going on simultaneously, by properly driving the WEN and BLK signals. This becomes extremely important in cases where multiple RAM blocks are cascaded for deeper memories. In such case, BLK must be used for address decoding.

Synchronous/Asynchronous Dual Port Ram for DX/MX Summary



Related Topics

[Synchronous/Asynchronous Dual Port Ram for DX/MX I/O Description]

[Synchronous/Asynchronous Dual Port Ram for DX/MX Parameter Description]

[Synchronous/Asynchronous Dual Port Ram for DX/MX Implementation Rules]

Key Features

- Parameterized word length and depth
- Dual port synchronous RAM architecture
- Dual port synchronous write, asynchronous read RAM architecture

The RAM cores use 3200DX and MX, 32x8 or 64x4, dual port RAM cells.

In the synchronous mode, the read and write operations are totally independent and can be performed simultaneously. The operation of the RAM is fully synchronous with respect to the clock signals, WClock and RClock. Data of value Data are written WAddress of the RAM memory space on the rising (RISE) or falling (FALL) edge of the clock

WClock (WCLK_EDGE). Data are read from the RAM memory space at RAddress into Q on the rising (RISE) or falling (FALL) edge of the clock signal RClock (RCLK_EDGE).

The behavior of the RAM is unknown if you write and read at the same address and signals WClock and RClock are not the same. The output Q of the RAM depends on the time relationship between the write and the read clock.

In the asynchronous mode, the operation of the RAM is only synchronous with respect to the clock signal WClock. Data of value Data are written to the WAddress of the RAM memory space on the rising (RISE) or falling (FALL) edge of the clock signal WClock (WCLK_EDGE). Data are read from the RAM memory space at RAddress into Q after some delay when RAddress has changed.

The behavior of the RAM is unknown if you write and read at the same address. The output Q depends on the time relationship between the write clock and the read address signal.

The WIDTH (word length) and DEPTH (number of words) have continuous values but the choice of WIDTH limits the choice of DEPTH and vice versa.

The write enable (WE) and read enable (RE) signals are active high request signals for writing and reading, respectively; you may choose not to use them.

Synchronous/Asynchronous Dual Port Ram for DX/MX I/O Description

Table 186 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Data	WIDTH	input	Req.	Input Data
WE	1	input	Opt.	Write Enable
RE	1	input	Opt.	Read Enable
WClock	1	input	Req.	Write clock
RClock	1	input	Opt.	Read clock
Q	WIDTH	output	Req.	Output Data

Synchronous/Asynchronous Dual Port Ram for DX/MX Parameter Description

Table 187 · Parameter Description

Parameter	Value	Function

Parameter	Value	Function
WIDTH	width	Word length of Data and Q
Depth	depth	Number of RAM words
WE_POLARITY	**1** 2	WE can be active high or not used
RE_POLARITY	**1** 2	RE can be active high or not used
WCLK_EDGE	**RISE** FALL	WClock can be rising or falling
RCLK_EDGE	**RISE** FALL NONE	RClock can be rising, falling, or not used

Table 188 · Implementation Parameters

Parameter	Value	Description
LPMTYPE	LPM_RAM_DQ	Generic Dual Port RAM category

Table 189 · Fan-in Parameters

Parameter	Value	Description
RAMFANIN	AUTO MANUAL	See Fan-In Control section below

Parameter Rules

If RCLK_EDGE is NONE (Asynchronous mode), then RE_POLARITY must be 2 (note used)

The number of RAM blocks used (function of width and depth) must be less than or equal to the number of RAM blocks in one column of the largest device.

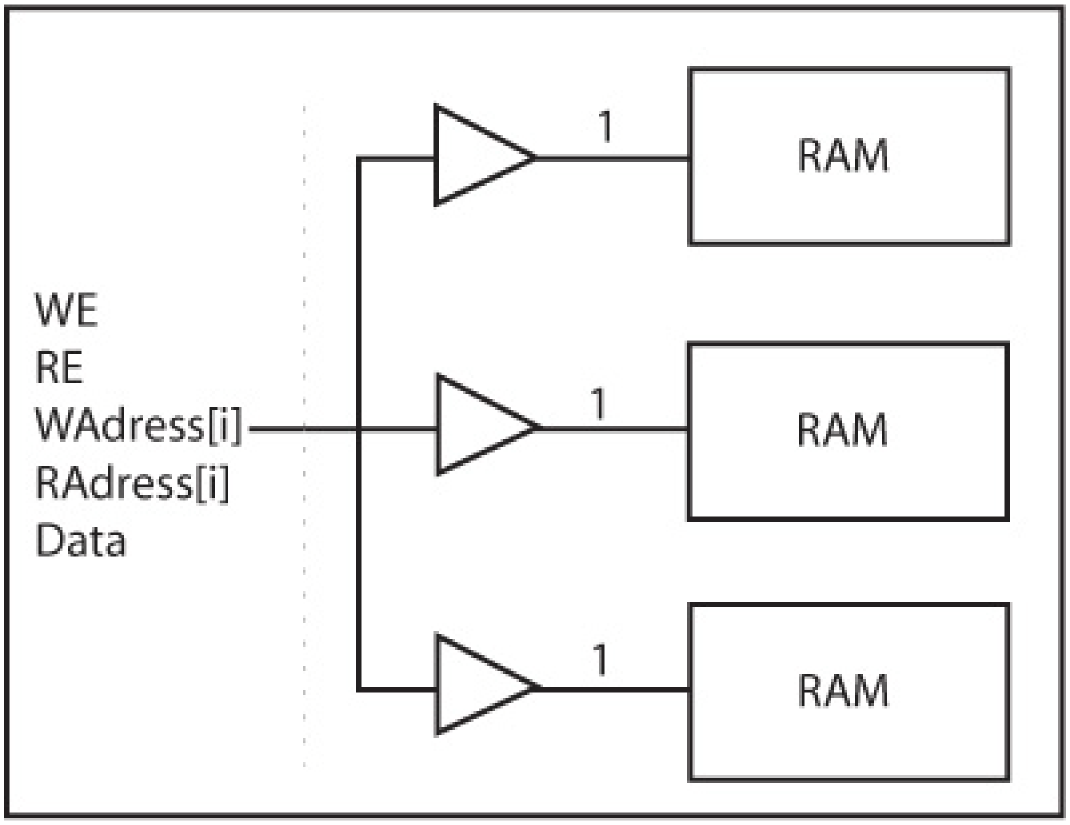
Synchronous/Asynchronous Dual Port Ram for DX/MX Implementation Rules

Fan-In Control

One of the key issues when building RAM cores is control of the routing congestion near the RAM cells. The problem becomes more critical when deep RAM cores are built. You need to broadcast signals throughout the height of the chip. The place-and-route algorithm could have difficulties satisfying all routing constraints. As a result, much slower routing resources could be allocated to satisfy all constraints. To make this problem less likely, a special buffering scheme has been implemented to relieve the congestion near the RAM cells. However, you may choose to control the buffering yourself to improve performances when needed. The RAM core can be built using either the automatic buffering architecture or the manual buffering architecture.

Automatic Buffering

In this mode (default), a buffering scheme is automatically built into the RAM core architecture (see the figure below). This mode should always be considered first. However, if the performance is not met, it may be better to use the manual buffering option.

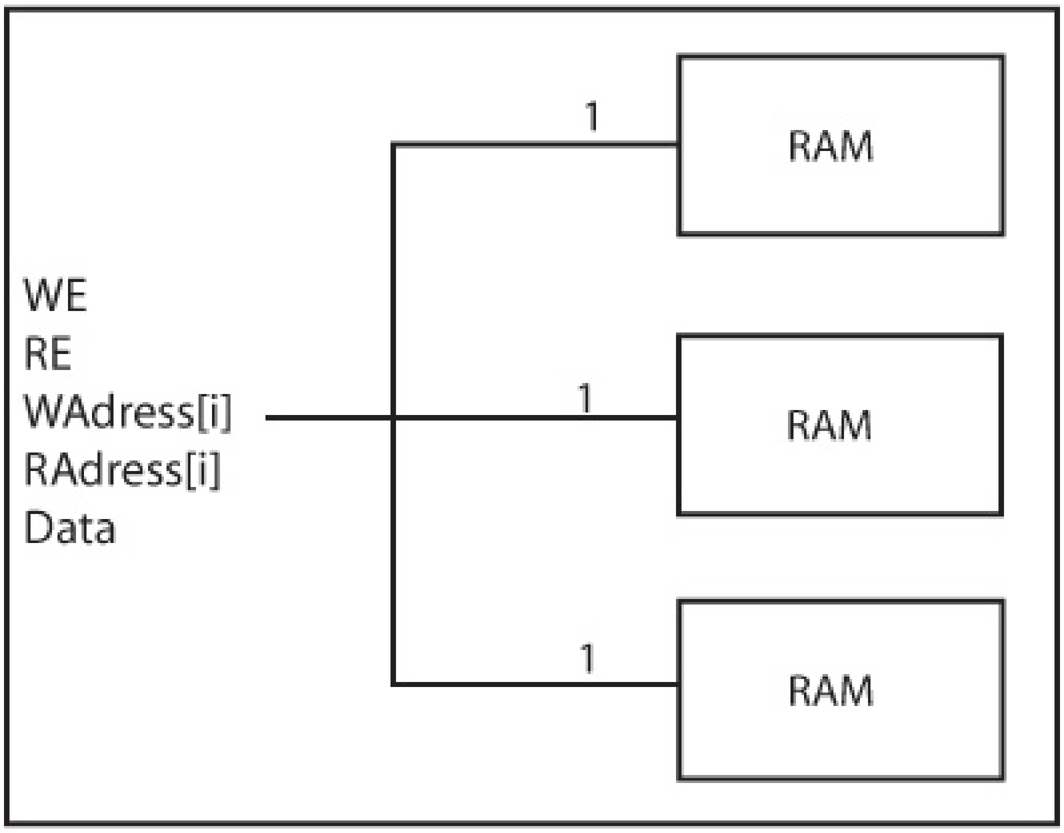


Automatic Buffering for RAM cores

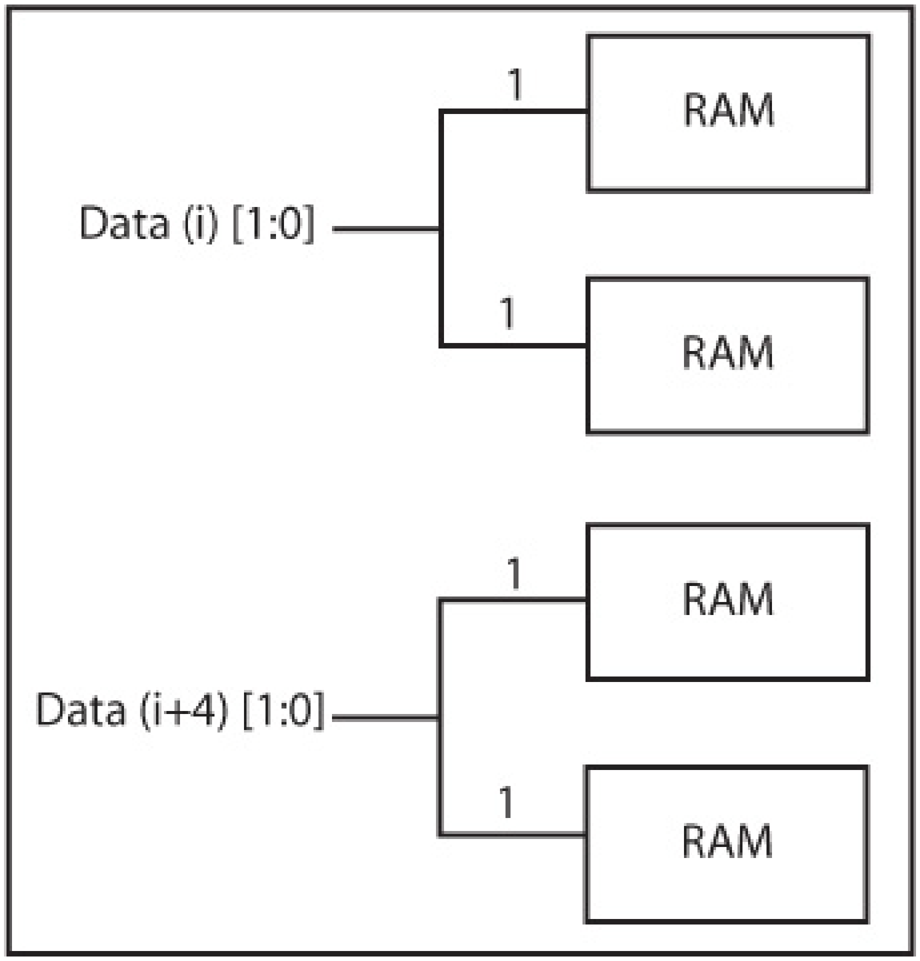
Manual Buffering

The figure below (Manual Buffering) shows how manual buffering is executed. A fan-in of one (1) is enforced on all signals fanning out to more than one RAM cell. If these signals were broadcast to all RAM cells, very slow routing resources (long freeways) would be required to route the signals impacting the RAM performance.

Manual buffering should only be used if the expected performance is not realized using the automatic buffering scheme, or if you know ahead of time that you need to use this scheme to meet your timing goals. In this architecture, the idea is not to buffer the signals internally but rather give some kind of access to the RAM core internal signals. Then, you must buffer the signals outside the core and either use traditional buffers or duplicate the logic that drives these signals externally. If you choose manual buffering, the WE, RE, Waddress(i), RAddress(i), and Data[i] signals become busses external to the core. For all these signals, the bus width is equal to the number of RAM cells (used to build a given configuration) driven by each signal. The Manual Buffering figure below illustrates the manual buffering architecture for a 96x8 RAM configuration, built of three 32x8 configured RAM cells. In this configuration, the WE, RE, WAddress and RAddress signals drive all RAM cells simultaneously. The Manual Buffering for the Data Bus shows a 128x8 RAM configuration, built using four 64x4 configured RAM cells. In that configuration, the 8-bit data bus is split into two completely independent 4-bit data busses.

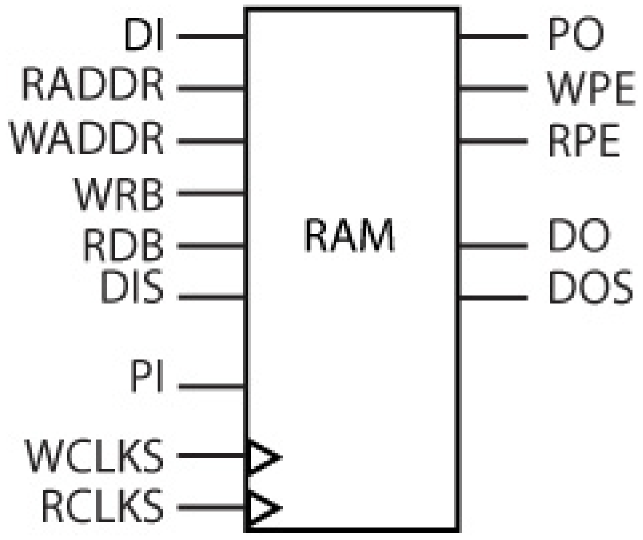


Manual Buffering (96x8 RAM Configuration)



Manual Buffering for the Data Bus (128x8 RAM Configuration)

Synchronous / Asynchronous Dual Port RAM for ProASICPLUS



Related Topics

Synchronous/Asynchronous Dual Port RAM for ProASICPLUS I/O Description

Synchronous/Asynchronous Dual Port RAM for ProASICPLUS Parameter Description

Key Features

- Parameterized word length and depth
- Dual-port RAM architecture
- Asynchronous, synchronous-transparent or synchronous-pipelined read
- Asynchronous, or synchronous write
- Parity check or generate, both even and odd
- Supported netlist formats: EDIF, VHDL and Verilog

There is no limitation for depth and width. However, it is your responsibility to insure that the RAM's used in a design can physically fit on the device chosen for the design.

Synchronous/Asynchronous Dual Port RAM for ProASICPLUS I/O Description

Table 190 · I/O Description

Port Name	Size	Type	Req/Opt	Function
DI	WIDTH	Input	Req.	Input Data

Port Name	Size	Type	Req/Opt	Function
RADDR	log2 (DEPTH)	Input	Req.	Read Address
WADDR	log2 (DEPTH)	Input	Req.	Write Address
WRB	1	Input	Req.	Write pulse (active low)
DIS	1	Input	Opt.	DMUX select; please refer to the Deep Memories section of the ProASICPLUS RAM/FIFO blocks application note
RDB	1	Input	Req.	Read pulse (active low)
WCLK	1	Input	Req.	Write Clock (active high)
RCLK	1	Input	Req.	Read Clock (active high)
DO	WIDTH	Output	Req.	Output data
DOS	1	Output	Opt.	DMUX select; please refer to the Deep Memories section of the ProASICPLUS RAM/FIFO blocks application note
PI	WIDTH	Input	Opt.	Input parity bits
PO	log2(WIDTH)	Output	Opt.	Parity bits
WPE	1	Output	Opt.	Write parity error flag
RPE	1	Output	Opt.	Read parity error flag

Synchronous/Asynchronous Dual Port RAM for ProASICPLUS Parameter Description

Table 191 · Parameter Description

Parameter	Value	Function

Parameter	Value	Function
WIDTH	Width	Word length of DI and DO
DEPTH	Depth	Number of RAM words
RDA	async transparent pipelined	Read Data Access
WRA	async sync	Write Data Access
OPT	speed area	Optimization
PARITY	checkeven check-odd geneven genodd none	Parity check or parity generation

Table 192 · Implementation Parameters

Parameter	Value	Description
LPMTYPE	LPM_RAM_DQ	Generic Dual Port RAM category

Two Port RAM

Supported Families

IGLOO, ProASIC3, SmartFusion and Fusion

Related Topics

Two Port RAM Functionality

Two Port RAM I/O Description

Two Port RAM Parameter Description

Two Port RAM Implementation Rules

Key Features

Optimize for High Speed (Width Cascading) or Low Power (Depth Cascading)

You can choose to optimize your RAM for High Speed or Low Power.

If you optimize for low power, the software evaluates your RAM configuration and attempts to generate a macro with depth cascading.

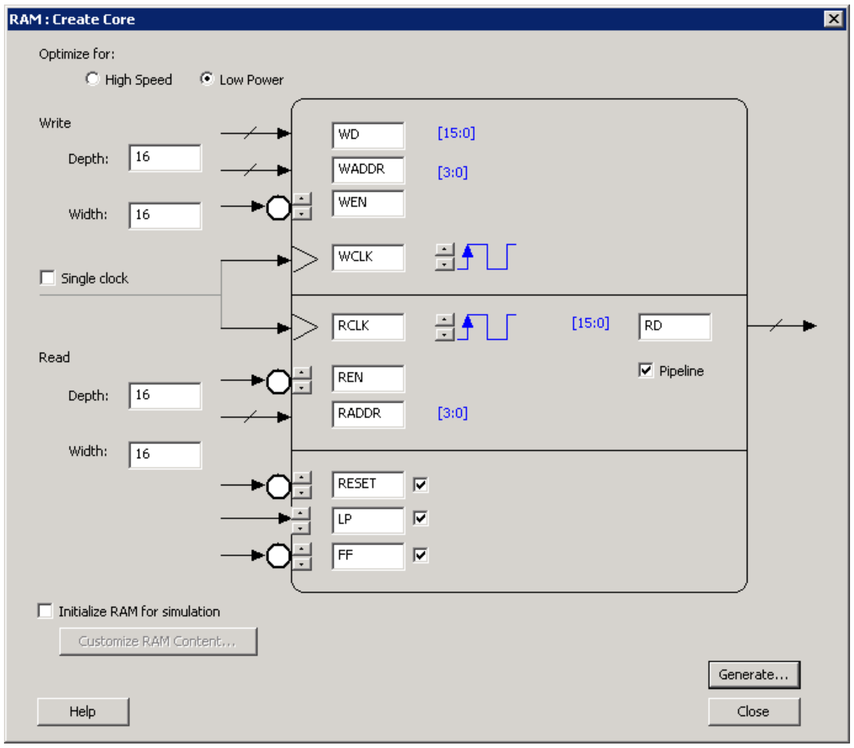
RAM with Initialization

The Fusion RAM with Initialization is nearly identical to the standard RAM, except that it generates extra logic so that it can interface with the Fusion Flash Memory System. The extra logic allows the RAMs to be switched to a x9 configuration during initialization or saved to the Flash Memory. The RAM will dynamically change its user-selected configuration to the x9 configuration to interface more smoothly with the Flash Memory System.

When you configure the RAM With Initialization the user configuration (ie. 64x32) is irrelevant. When you generate the RAM you get 4 ports exported named INITDOUT_0[08:00], INITDOUT_1[08:00], etc.. These 4 busses from the RAM need to drive the 4 inputs of the Flash Memory Block.

Additional ports that are exposed for a Fusion RAM with Initialization are shown in the table below. These ports are meant to be connected to the Flash Memory module.

See the RAM Content Manager for more information.


RAM : Create Core

Optimize for: ( ) High Speed [✓ Low Power]

Write
Depth: 16
Width: 16

[ ] Single clock

Read
Depth: 16
Width: 16

WD [15:0]
WADDR [3:0]
WEN
WCLK
RCLK
RD [15:0]
[✓] Pipeline
REN
RADDR [3:0]
RESET [✓]
LP [✓]
FF [✓]

[ ] Initialize RAM for simulation

Figure 32 · Two Port RAM Dialog Box

Two Port RAM Functionality

The core configurator automatically cascades RAM blocks to create wider and deeper memories by choosing the most efficient aspect ratio. It also handles the grounding of unused bits. The core configurator supports the generation of memories that have different Read and Write aspect ratios.

You can create a Dual Port RAM or Two Port RAM. A Dual Port RAM has read and write access on both ports while a Two Port RAM allows write access on one port and read access on the other port.

Optimization for High Speed or Low Power

High Speed results in a macro optimized for speed and area (width cascading).

Low Power results in a macro optimized for low power, but uses additional logic at the input and output (depth cascading). Performance for a low power optimized macro may be inferior to that of a macro optimized for speed. Some RAM configurations are not possible with depth cascading (such as 512 x 36), but low power optimization is a priority when the option is selected.

Write Depth/Width and Read Depth/Width

The depth range for any port is 1-65536. The width range for any port is 1-576.

In addition to the caveats listed below (Write Depth * Write Width) must equal (Read Depth * Read Width).

Single Clock (CLKB) or Independent Write and Read Clocks (WCLK and RCLK)

The default for Two Port RAM is independent clocks (one each for Write and Read); click the **Single clock** checkbox to drive CLKA and CLKB with the same clock.

Clock Polarity - Click the up or down arrows to change the active edge of your Write and Read clocks. If you use a single clock you can select on only RWCLK; if you use independent clocks you can select the polarity of both the WCLK and RCLK. The core configurator instantiates inverters to achieve the specified polarity.

Write Enable (WEN) and Read Enable (REN)

Write enable controls when the write data (WD) is written to the Write Address (WADDR) of the RAM at the clock edge.

Asserting the read enable causes the RAM data at the read address (RADDR) location to be loaded to the data port (RD).

Pipeline for Read Data Output

Click the **Pipeline** checkbox to enable pipelining for Read data output (RD) This is a static selection and cannot be changed dynamically by driving it with a signal.

If you choose to Initialize RAM, you can customize your RAM content with the RAM Content Manager.

Note: Dual Port RAM configured in INIT mode does not support pass write data to output. This is because in INIT mode, write always occurs on Port A and read occurs on Port B.

LP and FF Inputs

LP (Low Power) Port (Active High): You drive this port during active and idle modes to lower static Icc. When driving this pin, you must deselect the SRAM/FIFO for some determined time (counter value) or for a particular condition in your logic. This port has no effect if the port BLK is de-asserted (BLK port is active low), i.e. if BLK is high, then the LP port value is irrelevant.

FF (Flash*Freeze) Port (Active Low): Connect this port directly to your Flash*Freeze pin (INBUF_FF output) in Flash*Freeze Type 1. It must be inverted when driven by housekeeping logic involving the ULSICC macro in Flash*Freeze Type 2. Connect this port directly to the Flash_Freeze_Enabled port of the Flash*Freeze Management IP if you are using it to implement Flash*Freeze Type 2. Asserting this port ensures that the SRAM/FIFO does not stay in WRITE and/or READ mode when the device enters Flash*Freeze mode.

Two Port RAM I/O Description

Table 193 · I/O Description

Name	Direction	Required/Optional	Description
INITADDR	IN	Optional	Address from Flash Memory module
INITDATA[08:00]	IN	Optional	Data from Flash Memory module
INIT_CLIENT_i	IN	Optional	Write enable signal from the Flash Memory module.This indicates that the data on the INITDATA bus is to be written into the RAM at the INITADDR location. There will be a signal per RAM block used. For example, if the memory configuration required 4 RAM blocks, then there will be 4 of these signals exported. These signals need to be connected to the <client_name>_block_i_DAT_VAL from the Initialization / Flash Memory

Name	Direction	Required/Optional	Description
			module.
INITACTIVE	IN	Optional	Needs to be asserted when Initialization is being performed. Switches the aspect ratio to x9 width and changes the RAM to respond to the Initialization interface.
INITDOUT[08:00]	OUT	Optional	Data to be saved into the Flash Memory module.
SAVEACTIVE	IN	Optional	Needs to be asserted when Save is being performed. Switches the aspect ratio to x9 width and changes the RAM to respond to the Save interface. The address location to be read is also driven from the INITADDR port for SAVE operations.

Two Port RAM Parameter Description

The table below lists the Two Port RAM signals in generated netlists.

Table 194 · Parameter Description

Name	Type	Genfile Parameter	Bit/Bus	Description
WADDR	OUT	WADDRESS_PN	BUS	Write address
WD	IN	DATA_IN_PIN	BUS	Write data input
WEN	IN	WE_PN	Bit	Write enable
WCLK	IN	WCLOCK_PN	Bit	Write clock
RADDR	OUT	RADDRESS_PN	Bus	Read address
RD	OUT	DATA_OUT_PN	Bus	Read data input

Name	Type	Genfile Parameter	Bit/Bus	Description
RESET	IN	RESET_PN	Bit	Asynchronous reset
REN	IN	RE_PN	Bit	Read enable
RCLK	IN	RCLOCK_PN	Bit	Read clock
LP	IN	LP_PN	Bit	Low power input
FF	IN	FF_PN	Bit	Flash*Freeze input
RWCLK	IN	CLOCK_PN	Bit	Single clock for Two Port RAM

Two Port RAM Implementation Rules

Caveats for Two Port RAM generation

- If a word width of 9 is used for Read, then Write configurations of 1, 2, or 4 will cause the MSB of the output to be undefined. These configurations are not supported. However, configurations that do not use the 9th bit (e.g., a Read width of 512x8 and a Write width of 1024x4) are supported.
- The core configurator only supports depth and width RAM cascading up to 64 blocks.
- The core configurator does not generate RAM based on a specific device. It is your responsibility to make sure the RAM fits physically on the device. Dynamic configuration of the aspect ratios is supported only in the Fusion RAM with Initialization core.
- The software returns a configuration error for unsupported configurations.

Tips

- Writing different data to the same address using both ports in Dual Port RAM is undefined and should be avoided.
- All unused inputs must be grounded.
- WMODE is ignored during read operation.
- RESET does not reset the memory contents. It resets only the output.
- Writing to and reading from the same address is undefined and should be avoided. When using the RAM4K9 in Two Port mode, care should be taken that Read and Write operations are not going on simultaneously, by properly driving the WEN and BLK signals. This becomes extremely important in cases where multiple RAM blocks are cascaded for deeper memories. In such case, BLK must be used for address decoding.

RAM with Initialization

The Fusion RAM with Initialization is nearly identical to the standard RAM, except that it generates extra logic so that it can interface with the Fusion Flash Memory System. The extra logic allows the RAMs to be switched to a x9 configuration during initialization or save-back to the Flash Memory. The RAM will dynamically change its user-selected configuration to the x9 configuration to interface more smoothly with the Flash Memory System.

This core has 2 sets of interfaces. The normal RAM interface and an additional interface known as the Flash Memory Block Interface. The Flash Memory Block Interface becomes active when either INITACTIVE or SAVEACTIVE are asserted. This switches the basic RAM blocks inside the core into a x9 data width mode to easily interface with the Flash Memory Block Interface.

See the Analog System Clocks topic for more information on how to connect up the RAM clock for initialization.

Due to the cascading capabilities of the RAM macro, a RAM may be composed of multiple basic RAM blocks. Each of these RAM blocks will have its own memory initialization file, thus Flash Memory Block treats each as a separate client. This also means that there is a separate enable and data out port for each RAM block.

For Dual Port RAMs, Port A is used for initialization and Port B is used for save-back to Flash Memory. For Two Port RAMs, the Write Port is used for initialization and the Read Port is used for save-back to Flash Memory.

The macro will automatically include the necessary logic for initialization. However, selecting the “Enable on demand save to Flash Memory” checkbox will create the necessary save-back interface.

Note: When using the save functionality, the Pipeline option must not be used on the Read port (2 Port) or on Port B (Dual Port). This is because the Flash Memory Block save controller assumes that data will be made available on the clock cycle following the address being given. Refer to the timing diagram below.

The additional ports that are exposed as part of the Flash Memory Block Interface for the RAM with Initialization are shown in the table below. These ports are meant to be connected to the Flash Memory module.

Table 195 · RAM with Initialization I/O Description

Name	Direction	Description
INITADDR	Input	Active High; Address from Flash Memory module
INITDATA[08:00]	Input	Active High; Data from Flash Memory module
INIT_CLIENT_i	Input	Active High; Write enable signal from the Flash Memory module. This indicates that the data on the INITDATA bus is to be written into the RAM at the INITADDR location. There will be a signal per RAM block used. For example, if the memory configuration required 4 RAM blocks, then there will be 4 of these signals exported.

Name	Direction	Description
		These signals need to be connected to the <client_name>_block_i_DAT_VAL from the Flash Memory Block.
INITACTIVE	Input	Active High; Needs to be asserted when Initialization is being performed. Switches the aspect ratio to x9 width and changes the RAM to respond to the Initialization interface.
INITDOUT_i[08:00]	Output	Active High; Data to be saved into the Flash Memory module. There will be a data out port per RAM block used. For example, if the memory configuration required 4 RAM blocks, then there would be 4 of these DOUT ports. These ports need to be connected to the <client_name>_block_i_DIN from the Flash Memory Block.
SAVEACTIVE	Input	Active High; Needs to be asserted when Save is being performed. Switches the aspect ratio to x9 width and changes the RAM to respond to the Save interface. The address location to be read is also driven from the INITADDR port for SAVE operations.

RAM with Initialization Timing Diagrams and Design Tips

Initialization from Flash Memory

After the Flash Memory Block is reset, it begins the initialization for all of its configured clients (see the Flash Memory System Builder for more information). During this phase, the INITACTIVE signal on the RAM with Initialization should be asserted. The following timing diagram illustrates the timing of the initialization operation.

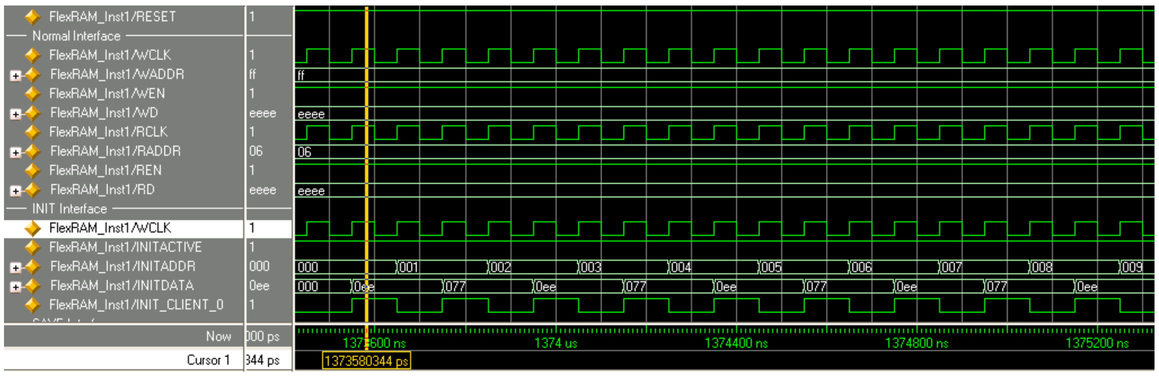


Figure 33 · FlexRAM Initialization Diagram

Save-Back to Flash Memory

The save operation will be initiated by asserting the CLIENT_UPDATE signal to the Flash Memory Block. The save operation is completed when the SAVE_COMPLETE signal is asserted. see the Flash Memory System Builder for more information. The following timing diagrams illustrate the timing of the save operation.

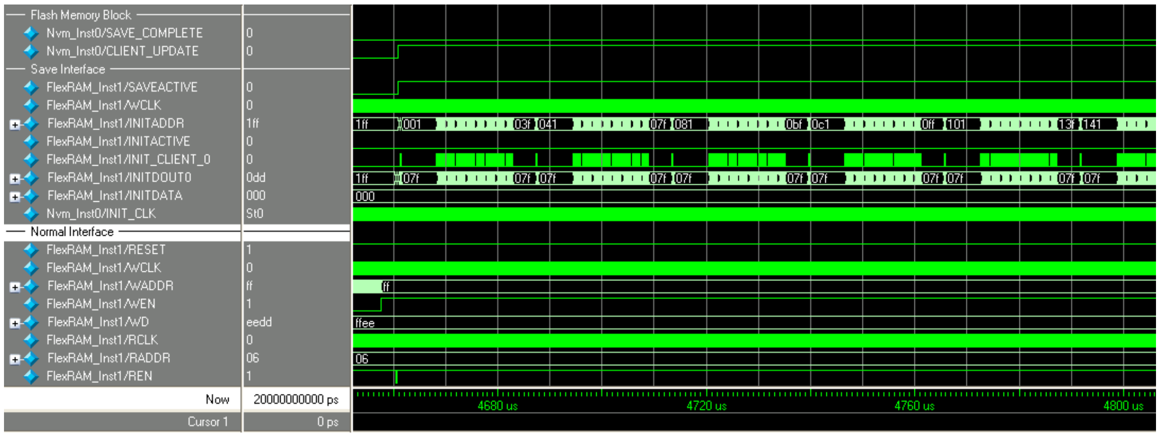


Figure 34 · FlexRAM Save Diagram (Full)

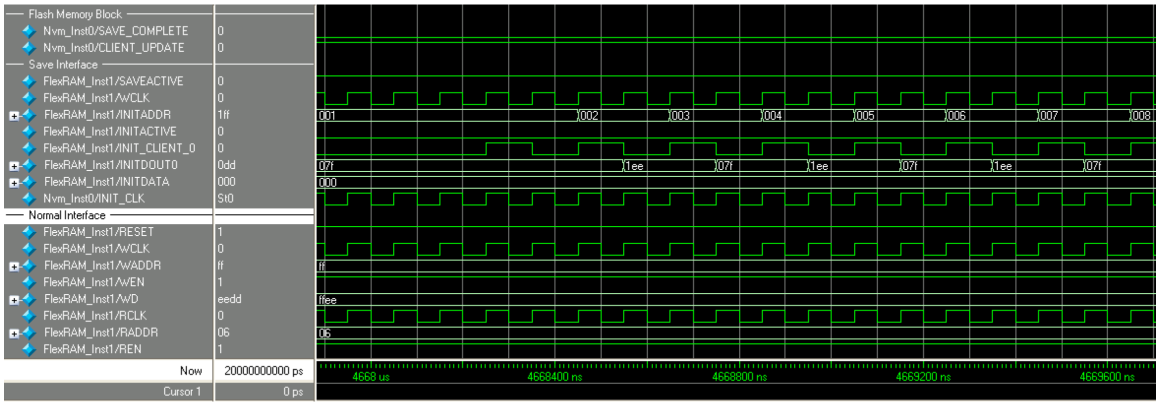


Figure 35 · FlexRAM Save Diagram (Detail)

Design Tips

INITACTIVE needs to be asserted during the initialization operation, thus it may be directly connected to the inverted version of INITDONE from Flash Memory Block.

SAVEACTIVE needs to be asserted during the save operation, there is no control signal from the Flash Memory Block that can be used for this indication. Thus, your design must generate this signal. The same control logic that asserts the CLIENTUPDATE to the Flash Memory Block can also assert this signal simultaneously, and then deassert it on SAVE_COMPLETE.

Power Management

Flash*Freeze Management Core Summary

Supported Families

IGLOO, ProASIC3L

When we list a family name, we refer to the device family and all its derivatives, unless otherwise specified. 'IGLOO' indicates all the IGLOO families (IGLOO, IGLOOe, etc).

Related Topics

Flash*Freeze Management Core Functionality

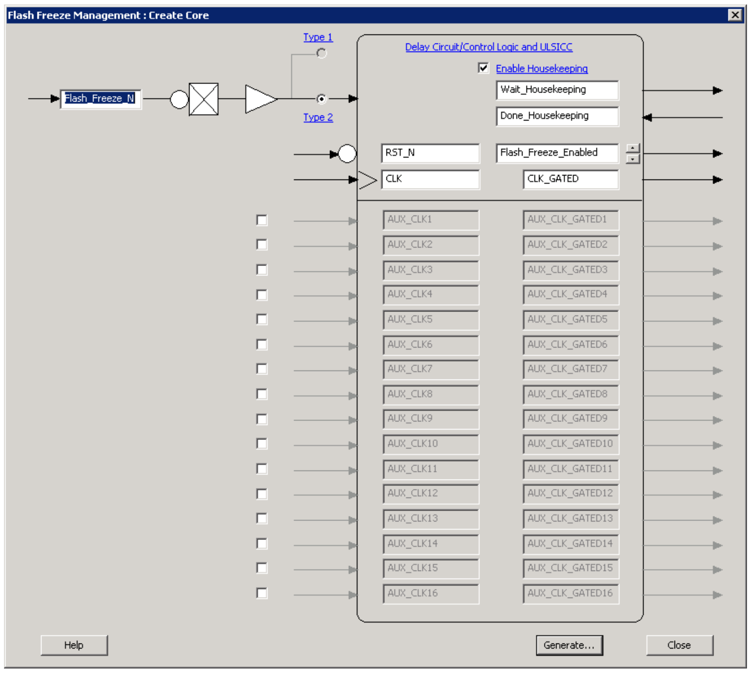
Flash*Freeze Management Core Parameter Description

Flash*Freeze Management Core I/O Description

Flash*Freeze Management Core Implementation Rules / Timing Diagrams

Key Features

- Manages entry and exit from Flash*Freeze mode
- Protects user logic from narrow pulses on the clock and allows critical processes to complete


Flash Freeze Management : Create Core
Type 1
Type 2
Delay Circuit/Control Logic and ULSICC
Help
Generate...
Close

Flash_Freeze_N

[✓] Enable Housekeeping

Wait_Housekeeping

Done_Housekeeping

RST_N

Flash_Freeze_Enabled

CLK

CLK_GATED

AUX_CLK1	AUX_CLK_GATED1
AUX_CLK2	AUX_CLK_GATED2
AUX_CLK3	AUX_CLK_GATED3
AUX_CLK4	AUX_CLK_GATED4
AUX_CLK5	AUX_CLK_GATED5
AUX_CLK6	AUX_CLK_GATED6
AUX_CLK7	AUX_CLK_GATED7
AUX_CLK8	AUX_CLK_GATED8
AUX_CLK9	AUX_CLK_GATED9
AUX_CLK10	AUX_CLK_GATED10
AUX_CLK11	AUX_CLK_GATED11
AUX_CLK12	AUX_CLK_GATED12
AUX_CLK13	AUX_CLK_GATED13
AUX_CLK14	AUX_CLK_GATED14
AUX_CLK15	AUX_CLK_GATED15
AUX_CLK16	AUX_CLK_GATED16

Description

This core manages entry and exit from the Flash*Freeze state by notifying user logic that the device is about to enter Flash*Freeze. This enables the user logic to reach a proper state, and ensures that any critical operations that are in process are completed (such as completing control register updates for peripherals, or waiting for incrementing/decrementing address pointers that are in process, etc).

The Flash*Freeze management core includes an INBUF_FF macro with a user-defined port name (default is Flash_Freeze_N). This port must be connected to the top level of your design. It is used to connect to the Flash*Freeze pin on your device.

Flash*Freeze Management Core Functionality

The Flash*Freeze Management core consists of two blocks, the FlashFreeze_FSM (finite state machine) block and the Filter block, as shown in the figure below.

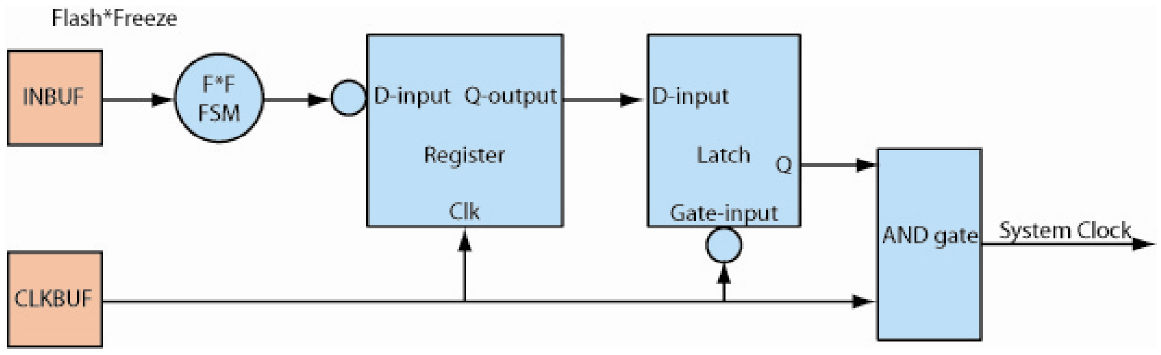


Figure 36 · Clock Gating (Filter) Block

Type 1 vs. Type 2

In Flash*Freeze Type 1, entering and exiting the mode is controlled by the assertion and deassertion of the Flash*Freeze pin. In order to use Flash*Freeze Type 1 you must instantiate INBUF_FF at the top level directly.

In Flash*Freeze Type 2, entering and exiting the mode is controlled by both the Flash*Freeze pin and the user-defined LSICC signal available in the ULSICC macro.

See the table below for Flash*Freeze (FF) pin assertion and deassertion values.

Signal	Assertion Value	Deassertion Value
Flash*Freeze (FF) Pin	Logic 0	Logic 1

See the Flash*Freeze section of the device handbook for more information on Flash*Freeze implementation

Actel strongly recommends that clock domains that require state-saving during Flash*Freeze have the clock routed through the Flash*Freeze Management core.

FlashFreeze_FSM

The FlashFreeze_FSM runs in the following order:

1. The FSM makes sure that Flash*Freeze pin is asserted and persistent.
2. The FSM signals to user logic to complete critical operations that are still in process with the signal WAIT_HOUSEKEEPING.
3. User logic indicates that critical operations are complete via the signal DONE_HOUSEKEEPING.
4. The FSM stops the clock connected to user logic using the Filter.
5. The FSM takes the device into Flash*Freeze mode by asserting the LSICC input of the ULSICC macro instantiated inside the Flash*Freeze Management core.
6. The device enters Flash*Freeze mode.
7. When the Flash*Freeze pin is de-asserted, the FSM wakes up. If it wakes up in the incorrect state it self-recovers.

8. On wake-up, the FSM de-asserts the LSICC input and releases the clock to user logic. This enables the FSM to protect the user logic from narrow pulses on the clock and allows critical processes to finish.

Filter - Filters the clocks to user logic based on the control signal received from the FSM; uses the CLKINT global buffer.

A minimum of one Filter exists in the core. There are 17 filters available in the core. The FSM will be clocked by the primary clock specified by the user.

HouseKeeping

Your design can enter Flash*Freeze mode at any time, even in the middle of critical tasks. Rather than stop the design clock immediately (as a prelude to entering Flash*Freeze), housekeeping enables you to delay it until the tasks are complete.

User logic determines when the user logic clock stops after the Flash*Freeze pin was asserted.

The housekeeping feature is optional. If you do not use it you can configure Flash*Freeze to bypass housekeeping. To do so, disable Enable Houskeeping in the GUI. This loops back the Wait_Housekeeping output with the Done_Houskeeping input.

Use Model

Figure 37 ·

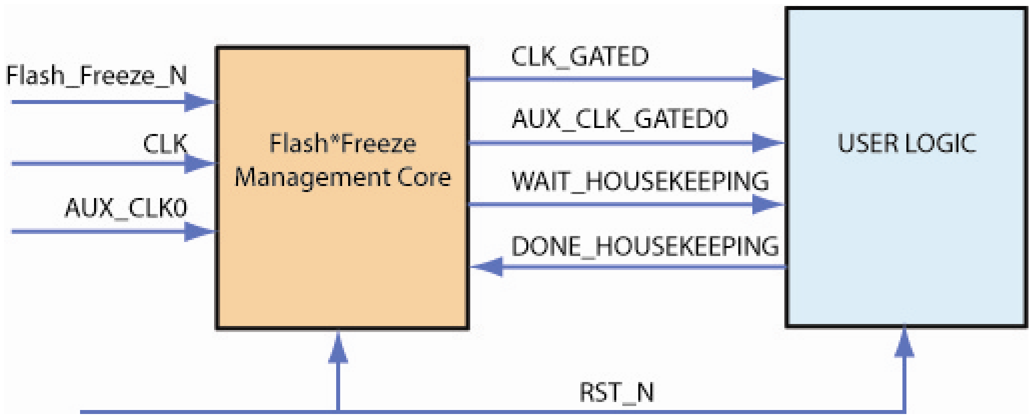


Figure 38 · Flash*Freeze Use Model

Flash*Freeze Management Core I/O Description

Table 196 · Signals

Signal	Direction	Polarity	Description
Flash_Freeze_n	Input	Active Low	Asynchronous input for Flash*Freeze mode
RST_N	Input	Active Low	Asynchronous Reset

Signal	Direction	Polarity	Description
CLK	Input		Free running clock to FSM; this is the clock you intend to use for state saving
AUX_CLK1 , ..., AUX_CLK16	Input		Other clocks connected to design that need to be filtered
Flash_Freeze_Enabled	Output	**High** or Low	When asserted this port indicates that the device is entering Flash*Freeze. Use this port to drive any logic in your design that needs to be driven by the Flash*Freeze state. This port should be used to drive the Flash*Freeze (FF) port of the RAM module generated from the Catalog when present in a design.
WAIT_HOUSEKEEPING	Output	Active High	Signal from the FSM to user logic to start housekeeping as clocks are going to be stalled.
DONE_HOUSEKEEPING	Input	Active High	Signal from User Logic that housekeeping is done and it is safe to stop the clock
CLK_GATED	Output		Gated version of CLK (ensures state saving)
AUX_CLK_GATED1, ..., AUX_CLK_GATED16	Output		Gated versions of the AUX_CLK1, ..., AUX_CLK16

Flash*Freeze Management Core Parameter Description

Table 197 · Parameter Description

Parameter / Generic	Description	Value Range

Parameter / Generic	Description	Value Range
NUMBER_OF_ADDITIONAL_CLOCKS	Total number of additional clocks that need filtering (besides CLK)	Minimum = 0; Maximum = 16

Flash*Freeze Management Core Implementation Rules / Timing Diagrams

Timing Diagrams

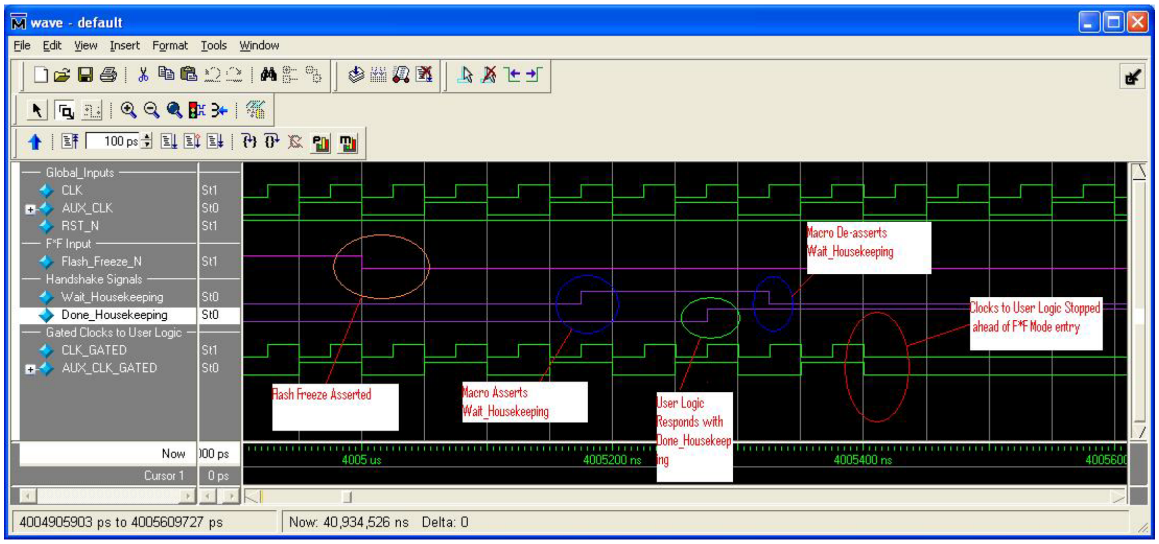
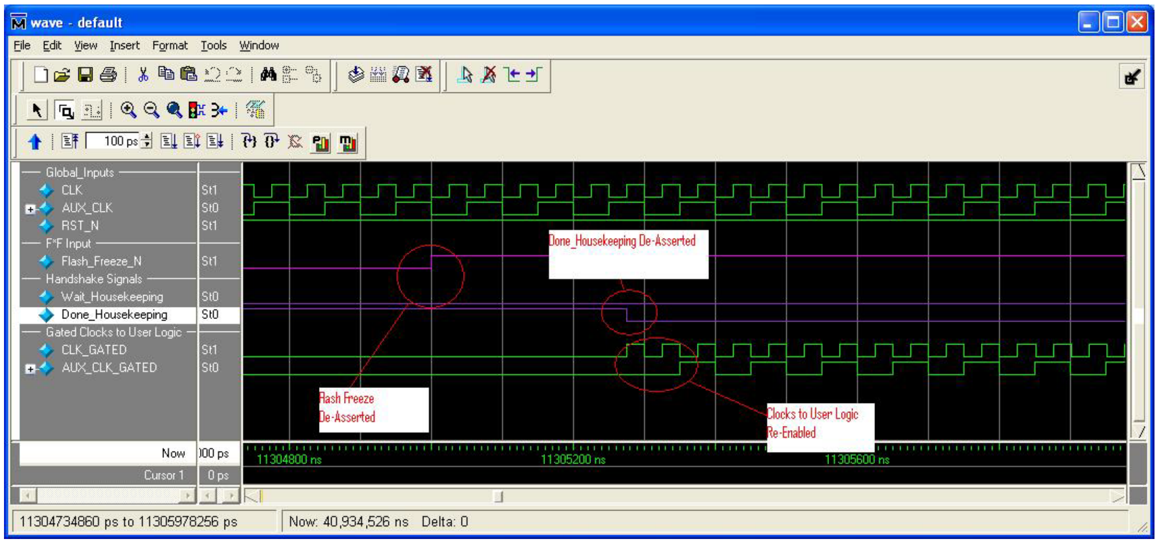
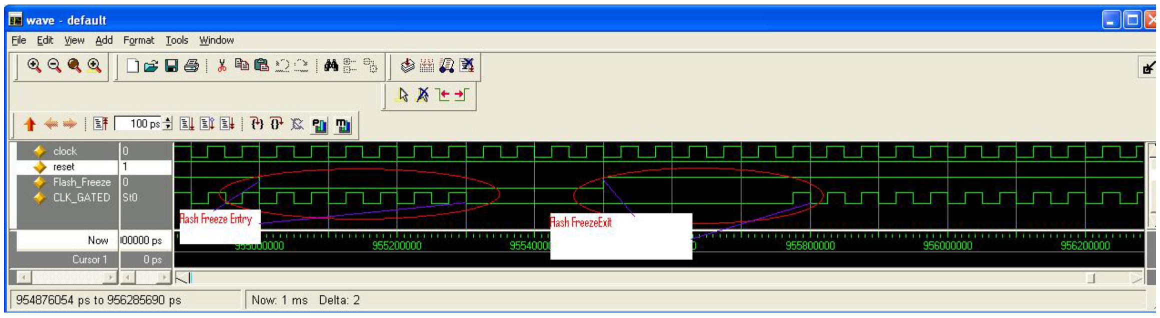

wave - default
File Edit View Insert Format Tools Window
100 ps
Global_Inputs
CLK
AUX_CLK
RST_N
F*F Input
Flash_Freeze_N
Handshake Signals
Wait_Housekeeping
Done_Housekeeping
Gated Clocks to User Logic
CLK_GATED
AUX_CLK_GATED
St1
St0
St1
St1
St0
St0
St1
St0
Now
Cursor 1
0 ps
4005 us
4005200 ns
4005400 ns
Macro De-asserts Wait_Housekeeping
Clocks to User Logic Stopped ahead of F*F Mode entry
Flash Freeze Asserted
Macro Asserts Wait_Housekeeping
User Logic Responds with Done_Housekeeping
4004905903 ps to 4005609727 ps
Now: 40,934,526 ns Delta: 0

Figure 39 · Flash*Freeze Asserted



11304734860 ps to 11305978256 ps   Now: 40,934,526 ns Delta: 0

Figure 40 · Flash*Freeze De-Asserted

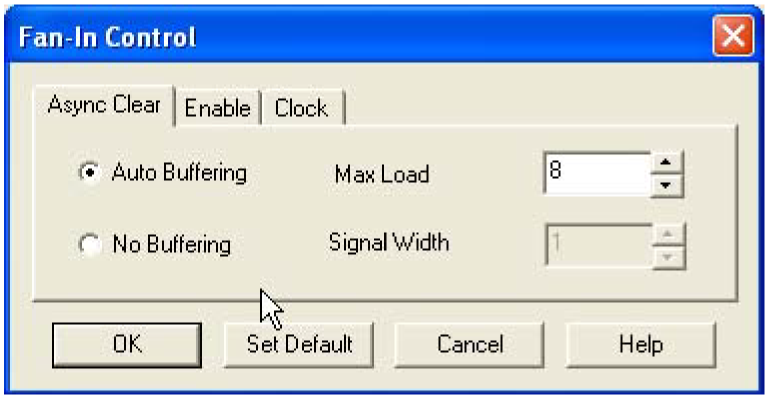


954876054 ps to 956285690 ps   Now: 1 ms Delta: 2

Figure 41 · No Housekeeping with Flash*Freeze

Fan-In Control Summary

The Fan-In Control tool gives advanced users the ability to control the buffering of clocks, asynchronous presets and clears, and other control signals. This tool is optional because default buffering values are provided for all signals. The tool supports two types of buffering control, automatic and no buffering, which provide maximum buffering flexibility.



Auto Buffering [✓] Max Load 8
No Buffering Signal Width 1

Figure 42 · Fan-In Control Dialog Box

Using Fan-In Control

1. Set your core options.
2. Open the Fan-In Control dialog box and input your values. If you modify your core options after you set your fan-in values, you must check them to ensure that they are unaffected.

Auto Buffering

Automatic buffering inserts buffers as required, and provides ease of use for fanning out heavily-loaded signals. Automatic buffering is the default buffering option for most signals. The value defined for automatic buffering indicates the maximum loading on the network for the given control signal. The core configurator software provides a single input for the signal and automatically inserts buffers/inverters with this option. The software also balances the loading as required.

No Buffering

The 'no buffering' option restricts the software from inserting buffers. This allows designers to manually use global clock resources for control signals. This also provides the ability to enhance performance of control signals by performing a logic function and correcting for fan-in by duplicating logic external to the core. If the signal is to be driven by a clock resource, you must set the signal width on the clock to 1; a signal width value of one (1) causes all loads to be driven by a single input.

Fan-In Control Implementation Rules

The Fan-In Control tool has the following limitations:

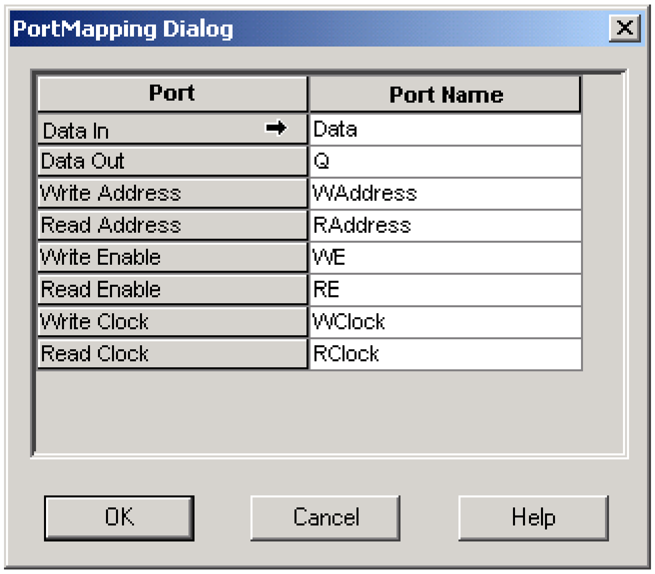
- The tool has been designed to be a slave to the primary core definition screen. Therefore, you should define exceptions to default values only after you have made all primary screen selections. Changing the main screen may affect the defined fan-in values. Information on modified fan-in will be provided in the Report window and should always be verified for correctness.

- The ability to perform no buffering on some control signals is limited to a single polarity because of hardware limitations. For example, ACT 2, ACT 3, 3200DX, MX, SX, SX-A, and eX limit asynchronous clears to Active Low only. Choosing Active High for this signal causes the No Buffering option to be unavailable. When this situation occurs, go back to the primary screen and change the active level for the given signal if no buffering is a must.
- Some control signals, such as the Count Enable signal are not included in the Fan-In Control tool because fan-out is corrected internally using AND and OR logic functions.

Use the Fan-In Control dialog box to specify Auto Buffering or No Buffering, Max Load, and Signal Width.

Port Mapping dialog box

You can use the Port Mapping function to specify the port naming for cores. Click the Port Mapping button to open the Port Mapping dialog box.



Port	Port Name
Data In	Data
Data Out	Q
Write Address	WAddress
Read Address	RAddress
Write Enable	WE
Read Enable	RE
Write Clock	WClock
Read Clock	RClock

Figure 43 · Port Mapping Dialog Box

The Port Mapping dialog box appears and displays the default port name values. Enter changes and click OK to submit, or click Cancel to return to the default values.

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel 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 Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650. 318.8044**

Actel Customer Technical Support Center

Actel staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions. The Customer Technical Support Center spends a great deal of time creating application notes and answers to FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Actel Technical Support

Visit the Actel Customer Support website (http://www.actel.com/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 Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at http://www.actel.com/.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

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 tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application 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 Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/company/contact/default.aspx.

Actel is the leader in low-power and mixed-signal FPGAs and offers the most comprehensive portfolio of system and power management solutions. Power Matters. Learn more at http://www.actel.com .

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 • USA*
Phone 650.318.4200 • Fax 650.318.4600 • Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • River Court, Meadows Business Park • Station Approach, Blackwater • Camberley Surrey GU17 9AB • United Kingdom*
Phone +44 (0) 1276 609 300 • Fax +44 (0) 1276 607 540

Actel Japan *• EXOS Ebisu Building 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • http://jp.actel.com

Actel Hong Kong *• Room 2107, China Resources Building • 26 Harbour Road • Wanchai • Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

5-02-9108-28/11.10