

---

# SmartGen Cores Reference Guide

For Libero SoC v11.7

NOTE: PDF files are intended to be viewed on the printed page; links and cross-references in this PDF file may point to external files and generate an error when clicked. **View the online help included with software to enable all linked content.**





---

# Table of Contents

Accumulator .....	9
Accumulator Functionality .....	9
Accumulator I/O Description .....	10
Accumulator Parameter Description .....	10
Accumulator Implementation Rules .....	10
Adder .....	11
Adder Functionality .....	12
Adder I/O Description .....	13
Adder Parameter Description .....	13
Adder Implementation Rules .....	14
Array Adder .....	14
Array Adder I/O Description .....	15
Array Adder Parameter Description .....	16
Array Adder Implementation Rules .....	16
Adder/Subtractor .....	16
Adder/Subtractor Functionality .....	17
Adder/Subtractor I/O Description .....	17
Adder/Subtractor Parameter Description .....	18
Adder/Subtractor Implementation Rules .....	18
Constant Decoder .....	18
Constant Decoder Functionality .....	19
Constant Decoder I/O Description .....	19
Constant Decoder Parameter Description .....	19
Constant Decoder Implementation Rules .....	20
Magnitude/Equality Comparator .....	20
Magnitude / Equality Comparator I/O Description .....	21
Magnitude / Equality Comparator Parameter Description .....	21
Magnitude / Equality Comparator Implementation Rules .....	22
Linear Binary Counters Summary .....	22
Linear Binary Counters Functionality .....	23
Linear Binary Counters - I/O Description .....	24
Linear Binary Counters - Parameter Description .....	24
Linear Binary Counters - Implementation Parameters .....	25
Dual Data Rate (DDR) Register .....	27
DDR Register Parameter Description .....	30
Decoder .....	30
Decoder Functionality .....	31
Decoder I/O Description .....	31
Decoder Parameter Description .....	31
Decrementer .....	32
Decrementer Functionality .....	32
Decrementer I/O Description .....	33

Decrementer Parameter Description .....	33
Decrementer Implementation Rules .....	33
FIR Filter .....	33
FIR Filter Functionality .....	34
FIR Filter I/O Description .....	35
FIR Filter Parameter Description .....	35
FIR Filter Implementation Rules / Timing Diagrams .....	36
Hard Multiplier Accumulator, AddSub and Signed.....	37
Bi-Directional Buffers .....	37
Bi-Directional Buffers I/O Description .....	37
Bi-Directional Buffers Parameter Description .....	38
Bi-Directional Buffers Implementation Rules .....	38
Global Buffers .....	39
Global Buffers I/O Description .....	39
Global Buffers Parameter Description .....	39
Global Buffers Implementation Rules .....	40
Input Buffers.....	40
Input Buffers I/O Description.....	40
Input Buffers Parameter Description.....	41
Input Buffers Implementation Rules.....	41
Output Buffers .....	41
Output Buffers I/O Description .....	42
Output Buffers Parameter Description .....	42
Output Buffers Implementation Rules / Timing Diagrams.....	43
PECL Global Buffers.....	43
PECL Global Buffers I/O Description.....	43
PECL Global Buffers Parameter Description.....	44
PECL Global Buffers Implementation Rules.....	44
Tri-State Buffers .....	44
Tri-State Buffers I/O Description .....	44
Tri-State Buffers Parameter Description .....	45
Tri-State Buffers Implementation Rules .....	45
Incrementer .....	46
Incrementer Functionality.....	46
Incrementer I/O Description .....	47
Incrementer Parameter Description .....	47
Incrementer Implementation Rules .....	47
Incrementer / Decrementer .....	48
Incrementer / Decrementer Functionality.....	48
Incrementer / Decrementer I/O Description .....	49
Incrementer / Decrementer Parameter Description .....	49
Incrementer / Decrementer Implementation Rules .....	49
Logic - AND.....	50
Logic - AND Functionality .....	50
Logic - AND I/O Description.....	50
Logic - AND Parameter Description.....	51

Logic - OR .....	51
Logic - OR Functionality.....	51
Logic - OR I/O Description .....	52
Logic - OR Parameter Description .....	52
Logic - XOR.....	52
Logic - XOR Functionality .....	53
Logic - XOR I/O Description.....	53
Logic - XOR Parameter Description.....	53
Multiplexor .....	53
Multiplexor I/O Description.....	54
Multiplexor Parameter Description .....	54
Multiplier .....	55
Multiplier Functionality.....	55
Multiplier I/O Description .....	57
Multiplier Parameter Description.....	57
Multiplier Implementation Rules .....	58
Constant Multiplier .....	58
Constant Multiplier Functionality .....	59
Constant Multiplier I/O Description .....	59
Constant Multiplier Parameter Description .....	60
Constant Multiplier Implementation Rules .....	60
Barrel Shifter .....	61
Barrel Shifter Functionality.....	61
Barrel Shifter I/O Description .....	62
Barrel Shifter Parameter Description .....	62
Barrel Shifter Implementation Rules .....	63
Shift Register.....	64
Shift Register Functionality .....	64
Shift Register I/O Description.....	65
Shift Register Parameter Description.....	65
Shift Register Implementation Rules.....	66
Storage Register .....	67
Storage Register Functionality.....	67
Storage Register I/O Description .....	68
Storage Register Parameter Description .....	68
Storage Register Implementation Rules .....	69
Storage Latch.....	69
Storage Latch Functionality .....	70
Storage Latch I/O Description.....	70
Storage Latch Parameter Description.....	71
Storage Latch Implementation Rules.....	71
Subtractor.....	72
Subtractor Functionality .....	72
Subtractor I/O Description.....	72
Subtractor Parameter Description.....	73
Subtractor Implementation Rules.....	73

Fusion Dynamic CCC.....	74
Fusion Dynamic CCC Functionality .....	75
Fusion Dynamic CCC I/O Description.....	77
IGLOO and ProASIC3 Dynamic CCC Summary .....	78
IGLOO and ProASIC3 Dynamic CCC Functionality .....	79
IGLOO and ProASIC3 Dynamic CCC I/O Description.....	81
IGLOO and ProASIC3 Dynamic CCC Implementation Rules / Timing Diagrams .....	82
Delayed Clock Summary .....	84
Divided and Delayed Clock.....	84
No-Glitch MUX (NGMUX) .....	84
Fusion Static PLL .....	84
Fusion Static PLL Functionality.....	85
Fusion Static PLL I/O Description .....	87
IGLOO and ProASIC3 Static PLL Summary.....	88
IGLOO and ProASIC3 Static PLL Functionality .....	89
IGLOO and ProASIC3 Static PLL I/O Description .....	91
IGLOO and ProASIC3 Static PLL Implementation Rules / Timing Diagrams.....	91
Crystal Oscillator Summary .....	94
RC Oscillator (RCOSC) Summary.....	94
Voltage Regulator Power Supply Monitor Summary .....	94
Analog System Builder Reference .....	95
ASB Port List.....	96
Analog System Builder Main Window.....	97
Modify Sampling Sequence .....	99
External Trigger Signals in Analog System Builder .....	102
Analog System Builder Output Files .....	104
ASB Advanced Options .....	105
ASB Advanced Options - Calibration.....	107
ASB Connectivity and Calibration .....	108
Analog System Builder Calibration Output Files.....	110
ASB Advanced Options - ASSC RAM .....	111
ASB Advanced Options - SMEV RAM.....	112
ASB Advanced Options - SMEV Status.....	113
ASB Advanced Options - ADC results.....	113
ASB Advanced Options - ADC status.....	114
ASB Advanced Options - ACM Bus.....	114
ASB Advanced Options - ACM Clock .....	116
ASB - Calculating a Threshold.....	116
Prescaler Range .....	117
Acquisition Time.....	118
ASB Channel Mapping.....	118
Designing with the Analog System .....	119
Analog System Clocks .....	119
Analog System Builder and Flash Memory Block Basic Configuration .....	120
Analog System Builder with Real Time Counter (RTC).....	121
Analog System Builder with ACM Access.....	122

Current Monitor .....	124
Differential Voltage Monitor.....	125
Direct Digital Input.....	127
Gate Driver.....	127
Internal Temperature Monitor .....	128
Internal Voltage Monitor .....	128
Real Time Counter .....	129
Temperature Monitor.....	132
Voltage Monitor .....	132
Configuring Current, Differential Voltage, Temperature, and Voltage peripherals .....	133
Sampling Rate in Analog System Builder .....	136
Sequencing in Fusion .....	140
Welcome to FlashROM .....	141
Create/Configure FlashROM .....	143
Modify an existing FlashROM configuration .....	146
Simulate Pre/Post Synthesis.....	146
Place-and-Route and FlashROM.....	146
Welcome to the Flash Memory System Builder .....	146
Analog System Client.....	148
Data Storage Client.....	148
CFI Data .....	149
Initialization Client .....	150
RAM Initialization client.....	151
Flash Memory Block with Data Storage Client .....	152
Flash Memory System Output Files.....	152
Memory File Formats in Flash Memory System Builder .....	153
Soft FIFO Controller.....	156
Soft FIFO Controller with Memory Functionality .....	157
Soft FIFO Controller without Memory Functionality .....	158
Soft FIFO Controller I/O Description .....	159
Soft FIFO Controller Implementation Rules / Timing Diagrams .....	161
Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 and Fusion Summary .....	163
Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 and Fusion Functionality .....	165
Using ESTOP and FSTOP.....	165
Using FIFO Flags .....	166
Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 and Fusion Description .....	166
Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 and Fusion Implementation Rules / Timing Diagrams .....	166
RAM Content Manager Summary.....	166
RAM Content Manager Functionality.....	168
RAM Content Manager Implementation Rules .....	170
Dual Port RAM Summary.....	171
Dual Port RAM Functionality.....	172
Dual Port RAM Port Description .....	173
Dual Port RAM Implementation Rules .....	174
Dual Port RAM for SmartFusion and Fusion I/O Description .....	175

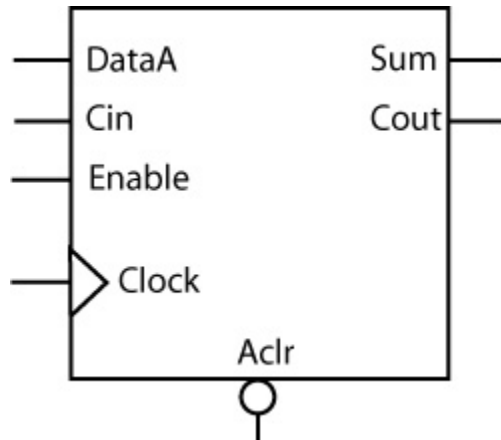
Two Port RAM.....	176
Two Port RAM Functionality .....	176
Two Port RAM Port Description .....	177
Two Port RAM Implementation Rules.....	178
Two Port RAM I/O Description.....	179
RAM with Initialization .....	179
RAM with Initialization Timing Diagrams and Design Tips .....	181
Flash*Freeze Management Core Summary .....	183
Flash*Freeze Management Core Functionality .....	183
Flash*Freeze Management Core I/O Description.....	185
Flash*Freeze Management Core Parameter Description.....	186
Flash*Freeze Management Core Implementation Rules / Timing Diagrams .....	186
Fan-In Control Summary.....	187
Fan-In Control Implementation Rules .....	188
Port Mapping dialog box .....	188
<b>Product Support .....</b>	<b>189</b>
Customer Service.....	189
Customer Technical Support Center.....	189
Technical Support .....	189
Website .....	189
Contacting the Customer Technical Support Center .....	189
ITAR Technical Support.....	190





# Basic Blocks

## Accumulator



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 <sub>n+1</sub>	Cout <sup>A</sup>
m[width-1 : 0]	(m + Sum <sub>n</sub> + Cin)[width-1 : 0]	(m + Sum <sub>n</sub> + 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	Value	Function
WIDTH	2-128	Word length of DataA, DataB and Sum
MAXFANOUT	0	Automatic choice
	2-16	Manual setting of Max. Fanout
CI_POLARITY	0 1 2	Carry-in polarity (active low, active high, and not used)
CO_POLARITY	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)
CLK_EDGE	RISE FALL	

## Accumulator Implementation Rules

Table 4 · Implementation Rules

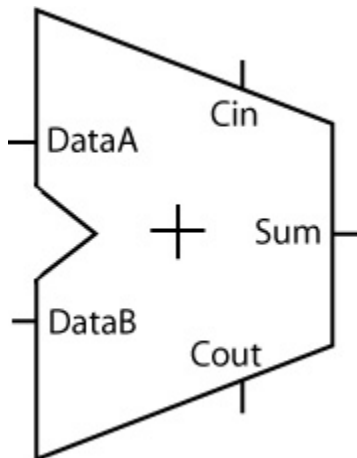
Parameter	Value	Description
-----------	-------	-------------

Parameter	Value	Description
LPMTYPE	LPM_ADD_SUB	Adder category
LPM_HINT	SKACC	Sklansky model
	FBKACC	Fast Brent-Kung model
	BKACC	Compact Brent-Kung model
	MFACC <sup>A</sup>	Very fast carry select model
	RIPACC <sup>A</sup>	Ripple carry model

Table 5 · Fan-In Control Parameters

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

## Adder



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

## 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 <sup>A</sup>
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.

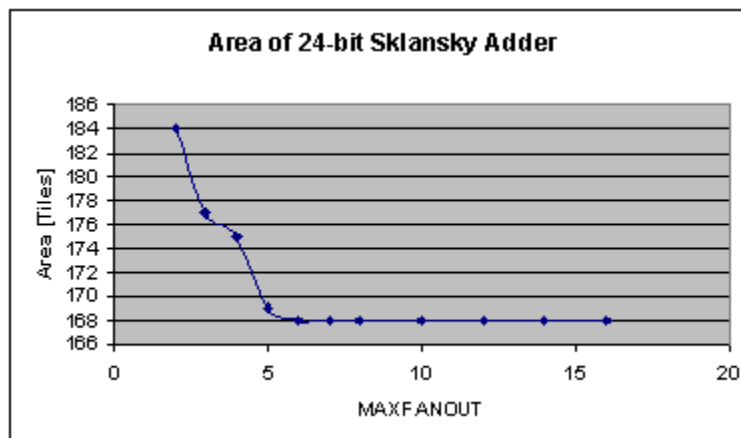


Figure 1 · Adder Area as a Function of MAXFANOUT

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.

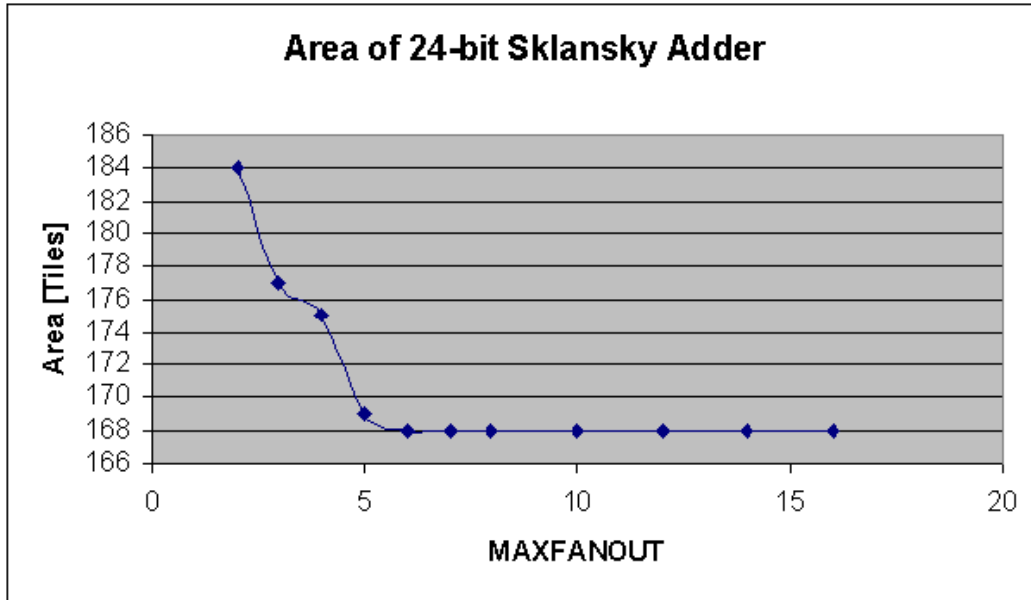


Figure 2 · Adder Performance as a Function of MAXFANOUT

## 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	Value	Description
WIDTH	2-128	Word length of DataA, DataB and Sum
MAXFANOUT	0	Automatic choice
	2-16	Manual setting of Max. Fanout
CI_POLARITY	0 1 2	Carry-in polarity (active low, active high, and not used)
CO_POLARITY	0 1 2	Carry-out polarity (active low, active high, and not used)

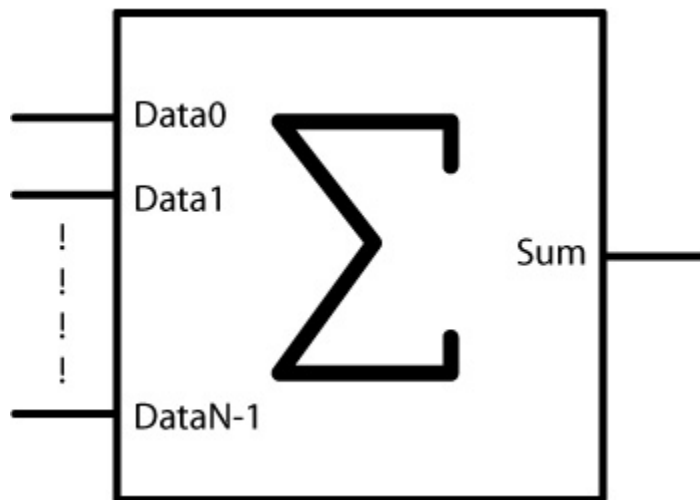
## Adder Implementation Rules

Table 9 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_ADD_SUB	Adder category
LPM_HINT	SKADD	Sklansky model
	FBKADD	Fast Brent-Kung model
	BKADD	Compact Brent-Kung model
	FADD <sup>A</sup>	Very fast carry select model
	RIPADD <sup>A</sup>	Ripple carry model
	MFADD <sup>A</sup>	Fast carry select model

A. FADD and MFADD are not recommended for ProASIC3 devices.

## Array Adder



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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:

$$\text{Sum} = [\text{Summation}(\text{Data}(i))] \text{ where } i = 0 \text{ to Size-1}$$

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 Microsemi 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:

$$\text{OUTWIDTH} = \log_2((m * \exp_2(n) - 1) + 1) \leq n + \log_2(m)$$

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

$$\text{SumA\_Width} \leq \text{SumB\_Width} \leq \text{OUTWIDTH}$$

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
Clock	1	Input	Opt.	Active High/Low Clock (if pipelined)

## Array Adder Parameter Description

Table 11 · Parameter Description

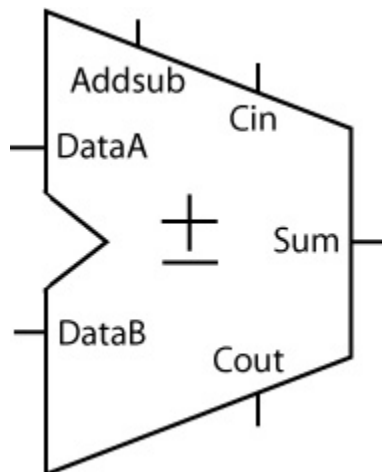
Parameter	Value		Description
WIDTH	width	2-64	Word length Data(i)
SIZE	size	3-64	Number of input buses
CKL_EDGE	RISE FALL		Clock (if pipelined)

## Array Adder Implementation Rules

Table 12 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

## 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 13 · Functional Description

DataA	DataB	Addsub	Sum	Cout <sup>A</sup>
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 14 · 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 15 · Parameter Description

Parameter	Value	Function
WIDTH	2-128	Word length of DataA, DataB and Sum
MAXFANOUT	0	Automatic choice
	2-16	Manual setting of Max. Fanout
CI_POLARITY	0 1 2	Carry-in polarity (active low, active high, and not used)
CO_POLARITY	0 1 2	Carry-out polarity (active low, active high, and not used)

## Adder/Subtractor Implementation Rules

Table 16 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_ADD_SUB	Adder category
LPM_HINT	SKADDSUB	Sklansky model
	FBKADDSUB	Fast Brent-Kung model
	BKADDSUB	Compact Brent-Kung model
	FADDSUB <sup>A</sup>	Very fast carry select model
	RIPADDSUB <sup>A</sup>	Ripple carry model

A. FADDSUB and RIPADDSUB are not recommended for ProASIC3 devices.

## Constant Decoder

### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 17 · Functional Description

<b>Aeb</b>
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 18 · Parameter Description

Parameter	Value	Function
WIDTH	2-128	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)

### Parameter Rules:

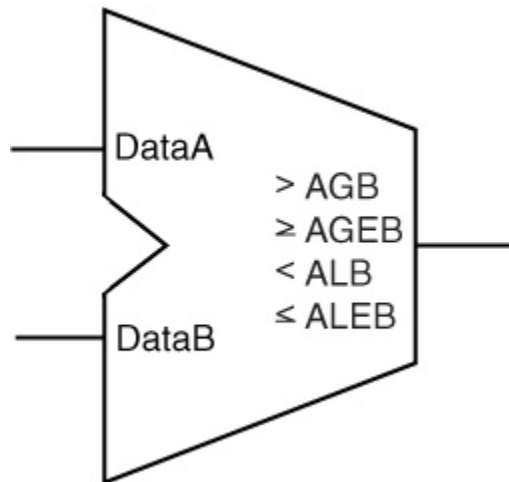
1. DataA is always binary and of the size of Width.
2. Constant must be of the selected Radix and be of the selected width for HEX/BIN.  
e.g.: Radix: BIN, Width: 5, Constant: 00010  
Radix Hex, Width:8, Constant: 0A

## Constant Decoder Implementation Rules

Table 19 · Implementation Rules

Parameter	Value	Description
LPM_TYPE	LPM_COMPARE	Comparator category
LPM_HINT	WDEC	Very fast

## Magnitude/Equality Comparator



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 20 · 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 21 · Parameter Description

Parameter	Value	Function
WIDTH	2-32	Word length of DataA and DataB
REPRESENTATION	<b>UNSIGNED</b> 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)
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)

Parameter	Value	Function
		not used)

## Magnitude / Equality Comparator Implementation Rules

Table 22 · Implementation Rules

Parameter	Value	Description
LPM_TYPE	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 least 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 23 · 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

## Linear Binary Counters Summary

### Related Topics

[Linear Binary Counters: Functionality](#)

[Linear Binary Counters - I/O Description](#)

[Linear Binary Counters - Parameter Description](#)

## [Linear Binary Counters - 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  $2^{\text{width}}-1$  for Up counters, or 0 for Down counters (Tcnt is not supported for Up/Down counters).

The counters are WIDTH bits wide and have  $2^{\text{width}}$  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 Functionality

### Implementations

#### *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 - I/O Description

Table 24 · 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
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 - Parameter Description

Table 25 · Parameter Description.

Parameter	Value	Function
WIDTH	2-45	Word length of Data and Q
DIRECTION	<b>UP</b> 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	<b>RISE</b> 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 26 · Fan-in Control Parameters

Parameter	Value
CLR_FANIN CLR_VAL	AUTO <b>MANUAL</b> [default value for MANUAL is 1]
SET_FANIN SET_VAL	AUTO <b>MANUAL</b> [default value for MANUAL is 1]
CLK_FANIN CLK_VAL	AUTO <b>MANUAL</b> [default value for MANUAL is 1]
LD_FANIN LD_VAL	<b>AUTOMANUAL</b> [default value for AUTO is 12]
UPDOWN_FANIN UPDOWN_VAL	<b>AUTOMANUAL</b> [default value for AUTO is 12]

## Linear Binary Counters - Implementation Parameters

Table 27 · Implementation Parameters

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

Table 28 · Functional Description<sup>A</sup>

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	$Q_{n+1} = 2^{\text{width}} - 1$
X	1	0	0	f	X	Qn	$Q_{n+1} = 2^{\text{width}} - 1$
m	1	X	1	f	X	M	$Q_{n+1} = 2^{\text{width}} - 1$
X	1	1	0	f	1	Qn+1	$Q_{n+1} = 2^{\text{width}} - 1$
X	1	1	0	f	0	Q	$Q_{n+1} = 2^{\text{width}} - 1$

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 29 · 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
	45	141.223	336	381	381

Table 30 · 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

Counter Type	Width	Performance	COMB Tiles	SEQ Tiles	Total Tiles
	45	216.92	176	53	229

## Dual Data Rate (DDR) Register

### Supported Families

SmartFusion2, IGLOO2, SmartFusion, IGLOO, ProASIC3, Fusion

### 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.

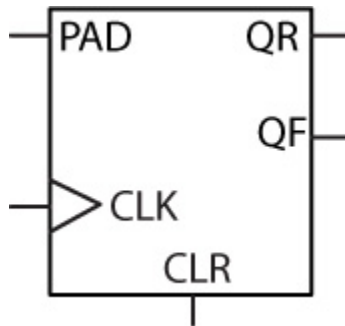
In SmartFusion, IGLOO, ProASIC3 and Fusion devices the DDR Registers may also be combined with output buffers or bi-directional buffers.

The Bidirectional enable signal polarity (TriEN) is selectable.

## DDR Register I/O Description

### SmartFusion, IGLOO, ProASIC3 and Fusion DDR Registers

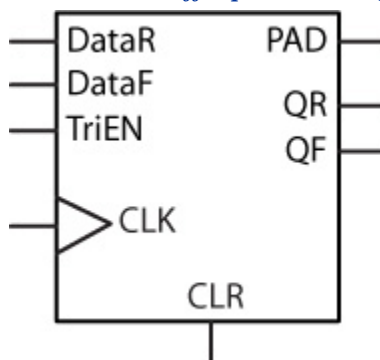
#### *Input Buffer plus DDR Register for SmartFusion, IGLOO, ProASIC3 and Fusion*



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

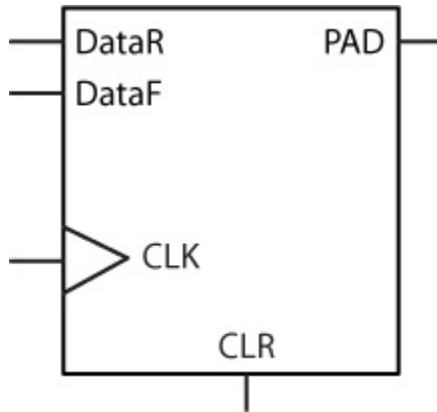


Port Description - Bidirectional Buffer plus DDR Register for SmartFusion, IGLOO, ProASIC3 and Fusion

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

Port Name	Size	Type	Req/Opt	Function
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 SmartFusion, IGLOO, ProASIC3 and Fusion*



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

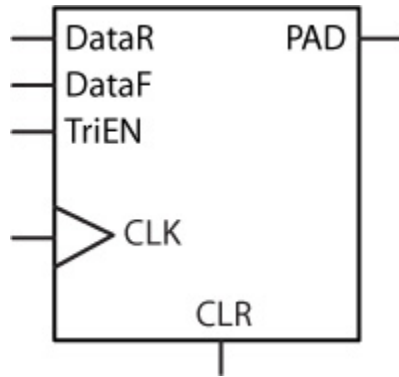


Table 31 · Port Description - Tri-State Buffer plus DDR Register for SmartFusion, IGLOO, ProASIC3 and Fusion

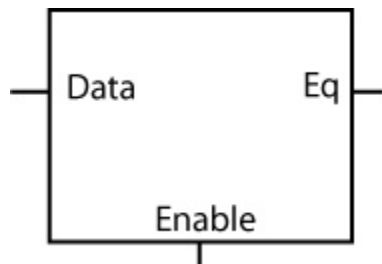
Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input/Output	Req.	Output Data
DataR	WIDTH	Input	Req.	Input Data
DataF	WIDTH	Input	Req.	Input Data
TriEN	1	Input	Req.	Tribuf Enable
CLK	1	Input	Req.	Clock
CLR	1	Input	Req.	Clear

## DDR Register Parameter Description

Table 32 · Dual Data Rate Register Parameters

Parameter	Value	Function
WIDTH	1-128	Data Width

## Decoder



## Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

## 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 33 · Decoder Functional Description <sup>A</sup>

Data	Enable	Eq
X	0	0's
m	1	$\text{dec}^B(m) == \text{decodes-1} \ \&\&^C$ $\text{dec}(m) == \text{decodes-2} \ \&\& \ \dots \ \&\& \ \text{dec}(m) == 0$

- A. Assume enable is active low and Eq is active high.  
 B.  $\text{dec}(m)$  defines the decimal value of m  
 C.  $\&\&$  indicates bity concatenation

## Decoder I/O Description

Table 34 · I/O Description

Port Name	Size	Type	Req/Opt	Function
Data	$\text{decln}^A$	Input	Req.	Input data
Enable	1	Input	Opt.	Enable
Eq	DECODES	Output	Req.	output

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

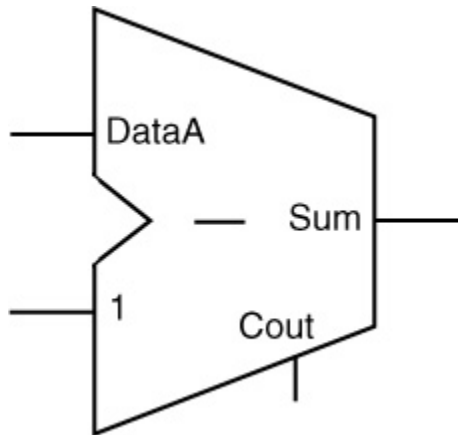
## Decoder Parameter Description

Table 35 · Parameter Description

Parameter	Value	Function
DECODES	2-32	Word length of Eq

Parameter	Value	Function
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

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 36 · Functional Description

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



## Decrementer I/O Description

Table 37 · 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 38 · 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 3 · 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

## FIR Filter



## Supported Families

SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion

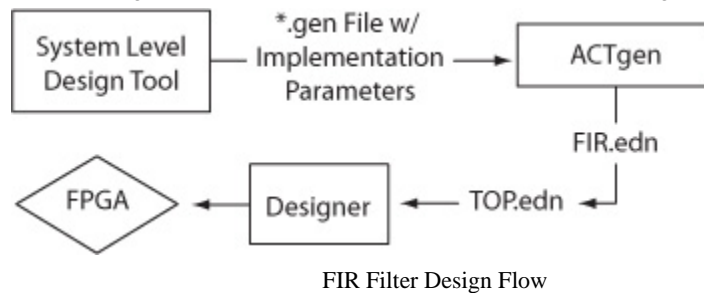
## Related Topics

- [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.



### 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 typical design flow.

## FIR Filter Functionality

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

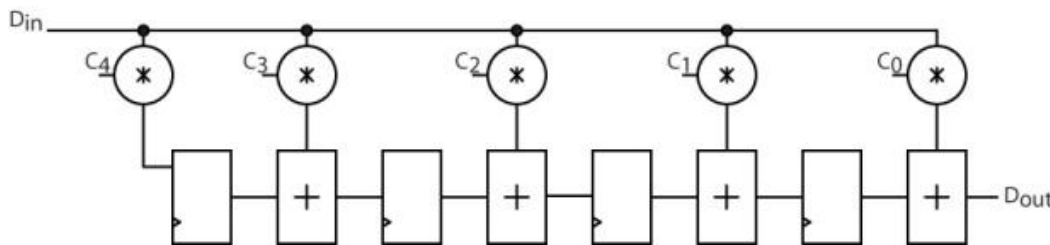


Figure 4 - Tap Transposed from FIR Filter

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
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 39 · 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	<b>RISEFALL</b>	Clock sensitivity
CLR_POLA	2 0 1	None, active high, active low
PREC		Internal precision
INSERT_PAD	<b>NO YES</b>	Pad insertion
INSERT_IOREG	<b>NO YES</b>	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

$$\text{MAX\_RES} \leq 2 \times \text{D\_WIDTH} + \lceil \log_2(\text{TAPS}) \rceil$$

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  $2^{\text{C\_WIDTH}-1}$  are considered positive, and  $2^{\text{C\_WIDTH}-1}$  to  $2^{\text{C\_WIDTH}-1}$  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 40 · FIR Filter Parameter Rules

Family	Variation	Parameter Rules
All	FIR2	PREC >= O_WIDTH

Table 41 · FIR Filter Implementation Parameters

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

Table 42 · 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

## Hard Multiplier Accumulator, AddSub and Signed

Click the Help button in the configurator for the latest Hard Multiplier Accumulator, AddSub or Signed Configurator documentation.

## Bi-Directional Buffers

Generates bi-directional buffers with a specified data width.

### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 43 - 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 44 · 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
PULLUP	NO / YES	Choice of Pull up version
TRIEN_POLARITY / EN_POLARITY (Flash)	0,1	Enable Polarity
TYPE (SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion 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

## Bi-Directional Buffers Implementation Rules

Table 45 · Implementation Rules

Parameter	Value	Function
LPMTYPE	LPM_IO / LPM_IOB_IO	Bi-directional Buffers

Parameter	Value	Function
LPM_HINT	BIBUF / IOB, GLMIOB	Regular Bi-directional Buffers / IO pad with Global Connection, Two Multiplexed Pads & Global Con-nection

## Global Buffers

### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 46 · I/O Description

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input	Req.	Inout Data
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 47 · 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,

Parameter	Value	Function
		2.5V*, 2.5V (Low Power)
PULLUP	NO / YES	Choice of Pull-up version

## Global Buffers Implementation Rules

Table 48 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 49 · I/O Description

Port Name	Size	Type	Req/Opt	Function
PAD	WIDTH	Input	Req.	Input Data
PADP	WIDTH	Input	Req.	Input Data for LVDS and LVPECL



Port Name	Size	Type	Req/Opt	Function
PADN	WIDTH	Input	Req.	Input Data for LVDS and LVPECL
Y	WIDTH	Output	Req.	Output Data

## Input Buffers Parameter Description

Table 50 · Parameter Description

Parameter	Value	Function
WIDTH	1-99 (Limit may vary depending on the family)	Data Width
PULLUP	NO / YES	Choice of Pull-up version
VOLT	0,1,2	Choice of different volt-age levels. 3.3v, 2.5v or 2.5v(Low Power)
TYPE (SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion 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

## Input Buffers Implementation Rules

Table 51 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_IO/ LPM_IB_IO (Flash)	Input Buffers
LPM_HINT	INBUF / IB	Regular Input Buffers

## Output Buffers

Output buffers with a specified data width.

## Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

## 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 52 · I/O Description

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

## Output Buffers Parameter Description

Table 53 · Parameter Description

Parameter	Value	Function
WIDTH	1-99 (Limit may vary depending on the family)	Data Width
VOLT	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	0,1,2	Choice of different slew rates. Low, Normal or High
TYPE (SmartFusion2, SmartFusion, IGLOO, ProASIC3, Fusion 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.

## Output Buffers Implementation Rules / Timing Diagrams

Table 54 · Output Buffers Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_IO / LPM_OB_IO	Output Buffers
LPM_HINT	OUTBUF / OB	Regular 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 55 · 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 56 · Parameter Description

Parameter	Value	Function
WIDTH	1-2	Data Width

## PECL Global Buffers Implementation Rules

Table 57 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 58 · I/O Description

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

Port Name	Size	Type	Req/Opt	Function
Trien / ENABLE (Flash)	1	Input	Req.	Enable

## Tri-State Buffers Parameter Description

Table 59 · Parameter Description

Parameter	Value	Function
WIDTH	1-99 (Limit may vary depend-ing on the family)	Data Width
VOLT	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	0,1,2	Choice of the slew rates: Low, Normal, or High
TRIEN_POLARITY / EN_POLARITY	0,1	Enable Polarity
TYPE	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

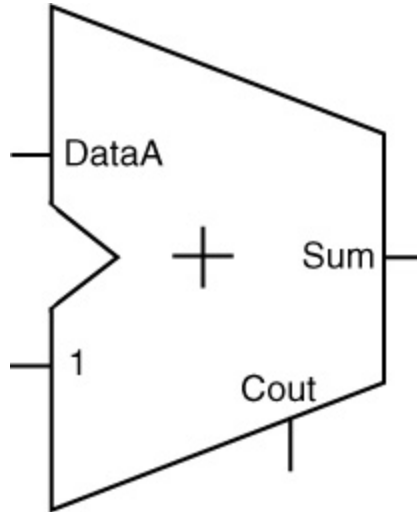
## Tri-State Buffers Implementation Rules

Table 60 · Implementation Rules

Parameter	Value	Function
LPMTYPE	LPM_IO / LPM_OB_IO	Tri-State buffers

Parameter	Value	Function
LPM_HINT	TRIBUFF / OTB	Regular Tri-State Buffers

## Incrementer



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 61 - Functional Description

DataA	Sum	Cout
m	m + 1	$(m + 1) \lesssim 2^{\text{width}}$

## Incrementer I/O Description

Table 62 · 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 63 · Parameter Description

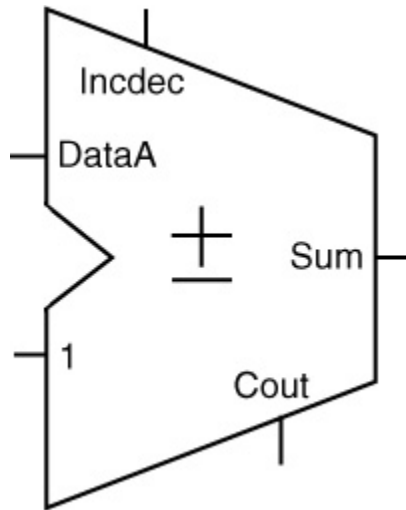
Parameter	Value	Function
WIDTH	2-32 2-156 for FC Macros	Word length of DataA and Sum
CO_POLARITY	0 1 2	Carry-out polarity (active low, active high, and not used)

## Incrementer Implementation Rules

Table 64 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 65 · Functional Description

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



## Incrementer / Decrementer I/O Description

Table 66 · 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 67 · Parameter Description

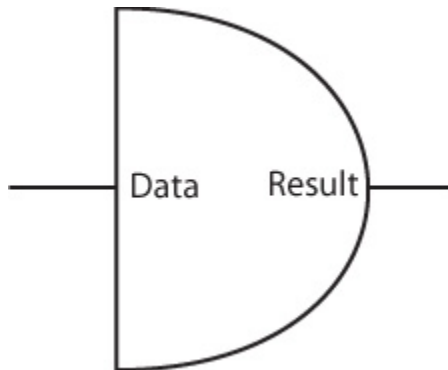
Parameter	Value	Function
WIDTH	2-32 2-156 for FC_FINCDEC and FC_RIPINCDEC	Word length of DataA and Sum
CO_POLARITY	0 1 2	Carry-out polarity (active low, active high, and not used)

## Incrementer / Decrementer Implementation Rules

Table 68 · Implementation Rules

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

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 69 · 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 70 · I/O Description

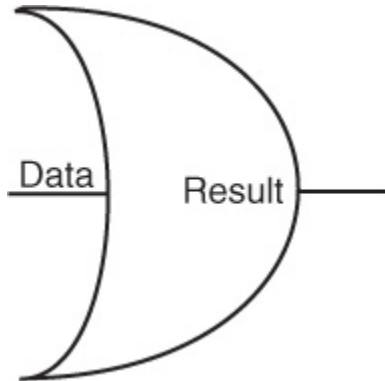
Port Name	Size	Type	Req/Opt	Function
Data	SIZE	Input	Req.	Input data
Result	1	Output	Req.	Output

## Logic - AND Parameter Description

Table 71 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

### Related Topics

[Logic - OR Functionality](#)

[Logic - OR I/O Description](#)

[Logic - OR Parameter Description](#)

### Key Features

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

## Logic - OR Functionality

Table 72 · 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 73 · Logic - OR I/O Description

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

## Logic - OR Parameter Description

Table 74 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

### Related Topics

- [Logic - XOR Functionality](#)
- [Logic - XOR I/O Description](#)
- [Logic - XOR Parameter Description](#)

### Key Features

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

## Logic - XOR Functionality

Table 75 · Functional Description (result is Active High)

Data	Result
m	$m[0] \text{ xor } m[1] \text{ xor } \dots \text{ xor } m[\text{SIZE}-1]$

## Logic - XOR I/O Description

Table 76 · I/O Description

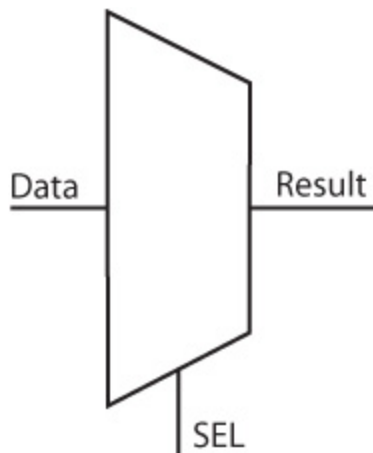
Port Name	Size	Type	Req/Opt	Function
Data	SIZE	Input	Req.	Input data
Result	1	Output	Req.	Output

## Logic - XOR Parameter Description

Table 77 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

## 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 78 · I/O Description

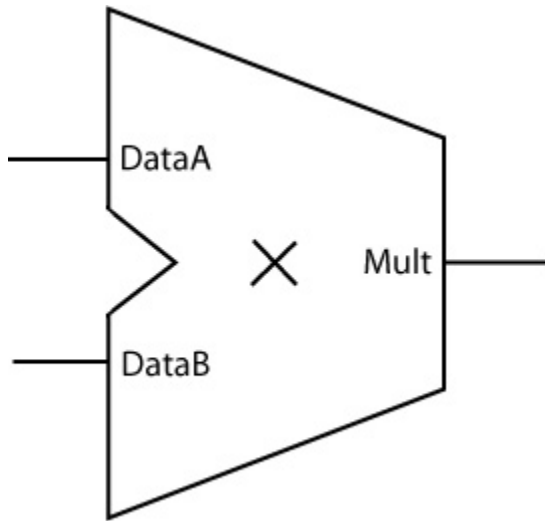
Port Name	Size	Type	Req/Opt	Function
Data <sub>0_port</sub>	WIDTH	Input	Req.	Input data
Data <sub>1_port</sub>	WIDTH	Input	Req.	Input data
...	...	...	...	...
Data <sub>SIZE-1_port</sub>	WIDTH	Input	Req.	Input data
Sel <sub>0</sub>	1	Input	Req.	Select line
Sel <sub>1</sub>	1	Input	Req.	Select line
...	...	...	...	...
Sel <sub>SIZELN-1</sub>	1	Input	Req.	Select line
Result	WIDTH	Output	Req.	Output

## Multiplexor Parameter Description

Table 79 · Parameter Description

Parameter	Family	Value	Function
WIDTH	ProASIC <sup>PLUS</sup>	1-48	Word length of Data
	All Others	1-32	
SIZE	All	2-32	Number of data inputs

## Multiplier



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 80 - Functional Description

DataA	DataB	Mult1 <sup>A</sup>
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 81 · Functional Description

DataA	DataB	Mult0/1 <sup>A</sup>
m	n	Mult1 + Mult2 = m * n

A. Mult1<0> is always 0

In the Architecture Comparison tables below, WIDTHA = WIDTHB.

Table 82 · 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 83 · 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 to specify pipeline stages.

### 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  $Y = (A \times B) + C + D$  using the multiplier with two outputs in combination with the Array-Adder.



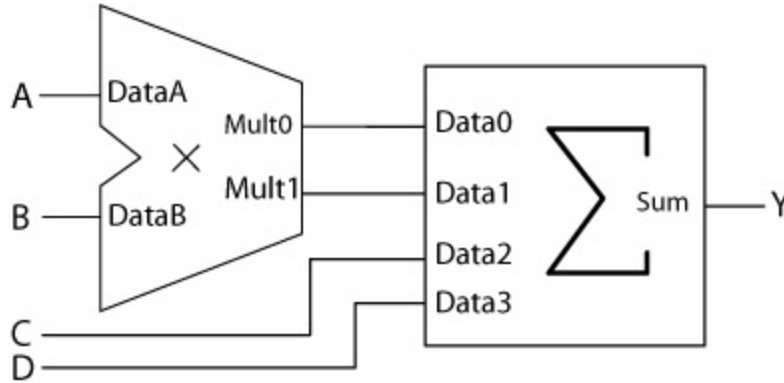


Figure 5 · Efficient Implementation Using the Two-Output Multiplier in Combination with the Array-Adder

## Multiplier I/O Description

Table 84 · 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 85 · Parameter Description

Parameter	Value	Function
WIDTHA <sup>A</sup>	2-64	Word length of DataA
WIDTHB	Same as WIDTHA	Word length of DataB
REPRESENTATION	<b>UNSIGNED</b> SIGNED	Data representation
CLK_EDGE	<b>RISE</b> 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.

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 86 · Parameter Rules

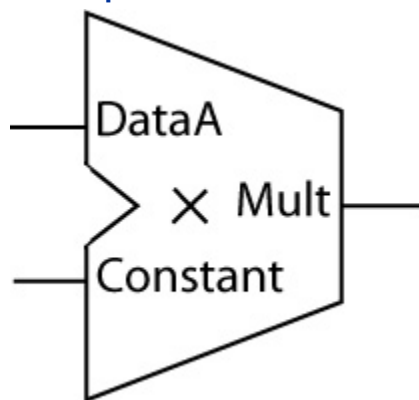
Family	Variation	Parameter rules
All	All	WIDTHA = WIDTHB

## Multiplier Implementation Rules

Table 87 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_MULT	Multiplier category
LPM_HINT	BOOTHMULT	Booth multiplier
	BOOTHMULT2	Booth multiplier without final Adder
LPMTYPE	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

## Constant Multiplier



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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

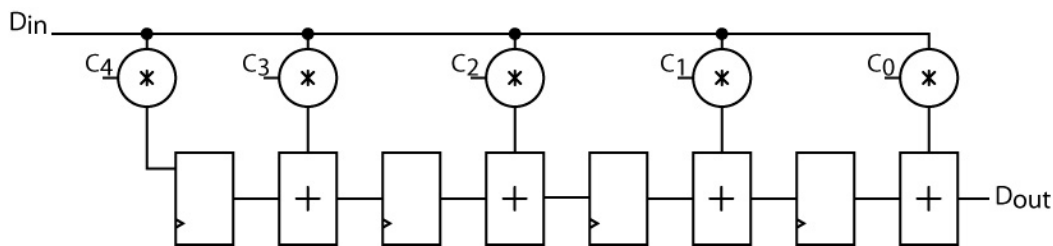
The worst case data for this example would be 1000 (-8) and therefore the worst case output data would be 010 0000 (-8 \* -4 = 32). So with that we know that Mult<8> is just a sign-extension bit (Mult<8> = Mult<7>).

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 88 · I/O Description

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

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

## Constant Multiplier Parameter Description

Table 89 · Parameter Description

Parameter	Value	Function
WIDTH <sup>A</sup>	2-64	Word length Data
CONST	Constant	Constant value
RADIX	HEX BIN DEC	Radix for constant value
SIGN <sup>B</sup>	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

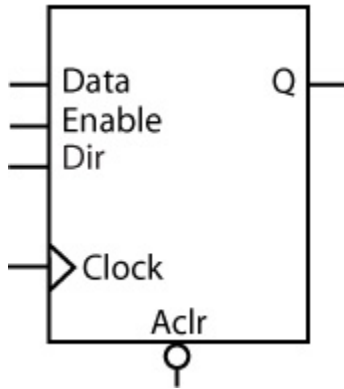
Radix Hex, Width:8, Constant: 0A

## Constant Multiplier Implementation Rules

Table 90 · Implementation Rules

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

## Barrel Shifter



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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  $\log_2(\text{MAXSHIFT})$  clock cycles for the shifted data to appear at the output.

The architecture is based on 2:1 multiplexors.

Table 91 · Functional Description<sup>A</sup> (Standard)

Data	Enable	Clock	Q
M	1	$f$	Qn
M	0	$f$	M <sub>shifted</sub>

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

Table 92 · Functional Description<sup>A</sup> (Pipelined)

Data	Aclr	Enable	Clock	Q

Data	Aclr	Enable	Clock	Q
X	0	X	X	0's
X	1	0	X	$Q_n = M_{\text{shifted}} - \log_2(\text{MAXSHIFT})$
m	1	1	<i>f</i>	$Q_{n+1} = M_{\text{shifted}} - \log_2(\text{MAXSHIFT}) + 1$

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

## Barrel Shifter I/O Description

Table 93 · 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 94 · 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	<b>Fixed</b> or Range	For a Fixed or Programmable shift
FILL	<b>No</b> , Yes	Wrap around or Feed a bit

Parameter	Value	Function
DIRECTION	Right Left Both	Direction can be Right, Left, or Both
EN_POLARITY	0 1 2	Enable can be active low or active high
CLK_EDGE	RISE FALL	Clock can be rising or falling

Table 95 · Fan-in Control Parameters

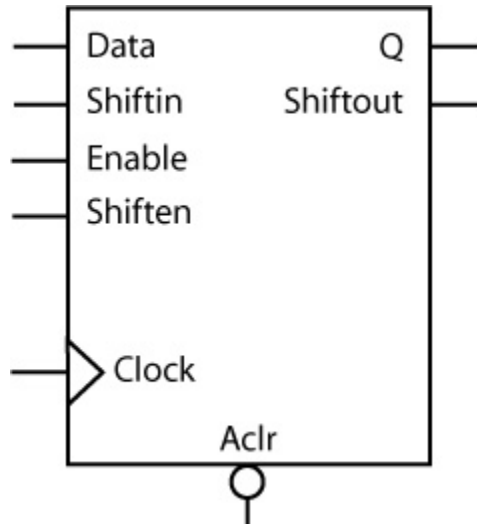
Parameter	Value
CLR_FANIN	AUTO MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	AUTO MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	AUTO MANUAL
CLK_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
SELO_FANIN	AUTO MANUAL
SELO_VAL	<val> [ default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

## Barrel Shifter Implementation Rules

Table 96 · Implementation Rules

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

## Shift Register



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 97 · Functional Description<sup>A</sup>

Data	Aclr	Enable	Shiften	Clock	Q <sup>B</sup>	Shiftout <sup>B</sup>
------	------	--------	---------	-------	----------------	-----------------------



Data	Aclr	Enable	Shiften	Clock	Q <sup>B</sup>	Shiftout <sup>B</sup>
X	0	X	X	X	0	0
X	1	X	X	¬	Q <sub>n</sub>	Q <sub>n</sub> = [WIDTH-1]
X	1	0	0	<i>f</i>	Q <sub>n</sub>	Q <sub>n</sub> = [WIDTH-1]
X	1	0	1	<i>f</i>	Q <sub>n</sub> [ WIDTH-2:0] && Shif-tin	Q <sub>n</sub> = [WIDTH-1]
m	1	1	X	<i>f</i>	Q <sub>n+1</sub> = m	Q <sub>n+1</sub> =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 98 · I/O Description

Name	Type	Required/Optional	Description
Data	IN	Opt	Register load input data
Shiften	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 99 · Parameter Description

Parameter	Family	Value	Function
WIDTH	500K, PA	2-512	Word length of Data and Q
	All others	2-99	

Parameter	Family	Value	Function
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	<b>RISE</b> <b>FALL</b>	Clock can be rising or falling

Table 100 · Fan-in Control Parameters

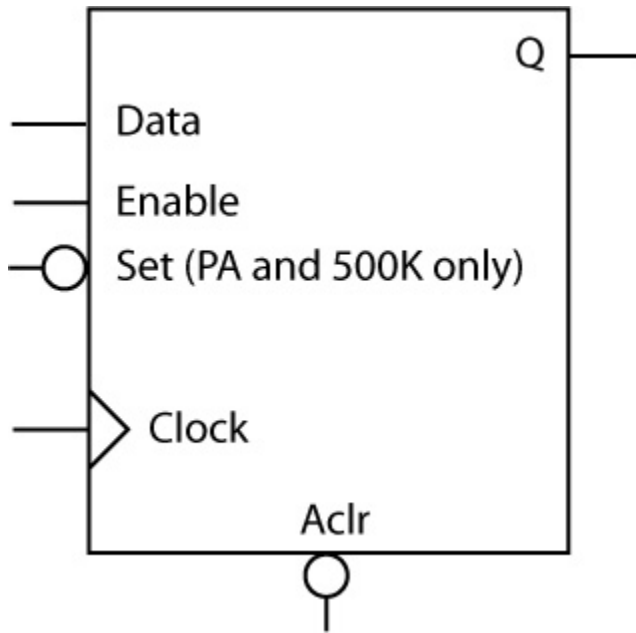
Parameter	Value
CLR_FANIN	<b>AUTO</b> MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	<b>AUTO</b> MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
SHEN_FANIN	<b>AUTO</b> MANUAL
SHEN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	AUTO <b>MANUAL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

## Shift Register Implementation Rules

Table 101 · 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

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 102 · Functional Description <sup>A</sup>

Data	Aclr	Enable	Clock	Q
X	0	X	X	0's
X	1	X	⎯	Qn
X	1	0	<i>f</i>	Qn
m	1	1	<i>f</i>	Qn+1 = m

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

## Storage Register I/O Description

Table 103 · 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 104 · 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	<b>RISE</b> FALL	Clock can be rising or falling

Table 105 · Fan-in Control Parameters

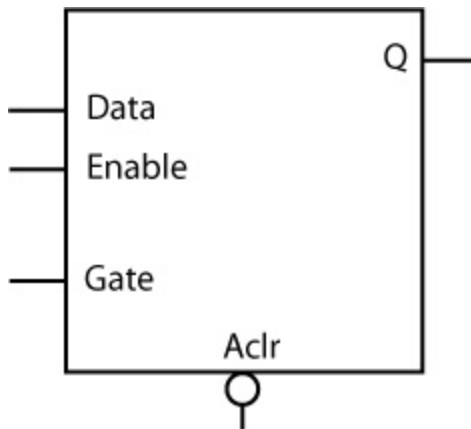
Parameter	Value
CLR_FANIN	<b>AUTO</b> MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	<b>AUTO</b> MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
CLK_FANIN	AUTO <b>MANUAL</b>
CLK_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]

## Storage Register Implementation Rules

Table 106 · Implementation Rules

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

## Storage Latch



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 107 · Functional Description<sup>A</sup>

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 108 · 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 109 · 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	<b>RISE</b> FALL	Clock can be rising or falling

Table 110 · Fan-in Control Parameters

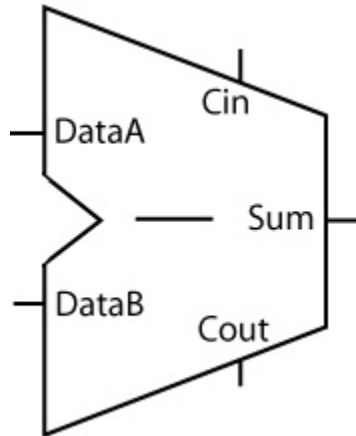
Parameter	Value
CLR_FANIN	<b>AUTO</b> MANUAL
CLR_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]
EN_FANIN	<b>AUTO</b> MANUAL
EN_VAL	<val> [default value for AUTO is 6, 1 for MANUAL]
GATE_FANIN	<b>AUTO</b> <b>MANUAL</b>
GATE_VAL	<val> [default value for AUTO is 8, 1 for MANUAL]

## Storage Latch Implementation Rules

Table 111 · Implementation Rules

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

## Subtractor



### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### 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 112 · Functional Description

DataA	DataB	Sum	Cout <sup>A</sup>
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 113 · I/O Description

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



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

## Subtractor Parameter Description

Table 114 · Parameter Description

Parameter	Family	Value	Function
WIDTH	ALL	2-128	Word length of DataA, DataB and Sum
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)

## Subtractor Implementation Rules

Table 115 · Implementation Rules

Parameter	Value	Description
LPMTYPE	LPM_ADD_SUB	Subtractor category
LPM_HINT	SKSUB	Sklansky model
	FBKSUB	Fast Brent-Kung model
	BKSUB	Compact Brent-Kung model
	FSUB <sup>A</sup>	Very fast carry select model
	RIPSUB <sup>A</sup>	Ripple carry model

A. FSUB and MFSUB are not recommended for ProASIC3 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 four, 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  $f_{in}$  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 0° 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.

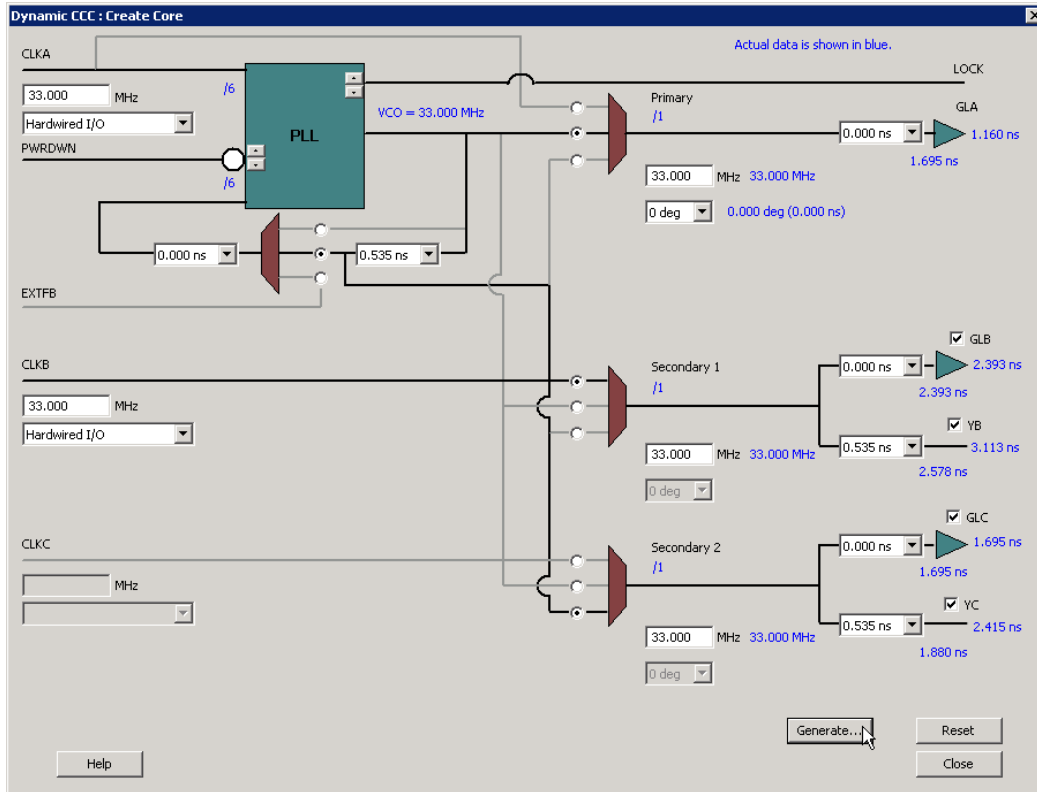


Figure 6 · Fusion Dynamic CCC Screen

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 ( $f_{in}/n$ ) must be between 1.5 and 5.5 MHz. The PLL Core output must be between 24 and 350 ( $f_{in} * m/n$ ).

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 116 · I/O Description

Name	Type	Required/Optional	Description
GLA	OUT	Req	Primary clock output
CLKA	IN	Req	Reference clock

Name	Type	Required/Optional	Description
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.

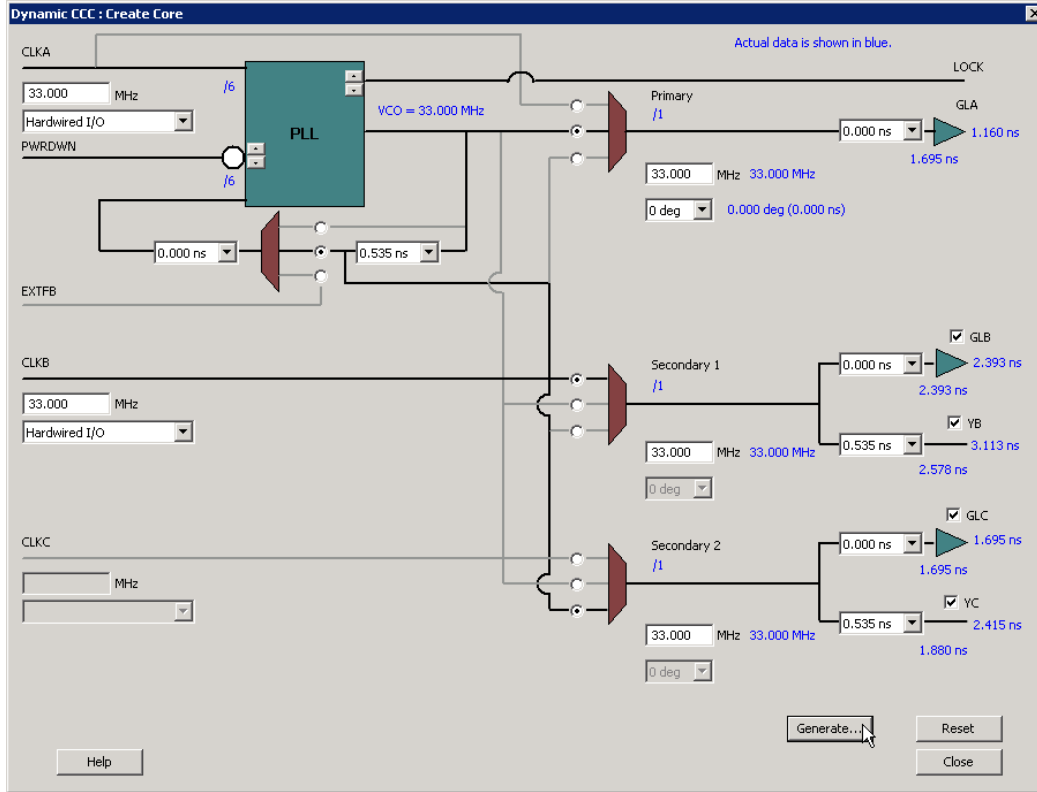
The clock conditioning circuit block is fully configurable, either via flash configuration bits (set in the programming bits stream) or through a simple asynchronous interface dynamically accessible from customer signals inside the device to permit parameter changes (such as PLL divide ratios) during 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
DLYC<4:0>	DELAY ON YC
STATASEL	MUX SELECT ON INPUT A
STATBSEL	MUX SELECT ON INPUT B
STATCSEL	MUX SELECT ON INPUT C
VCOSSEL<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 117 · 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 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

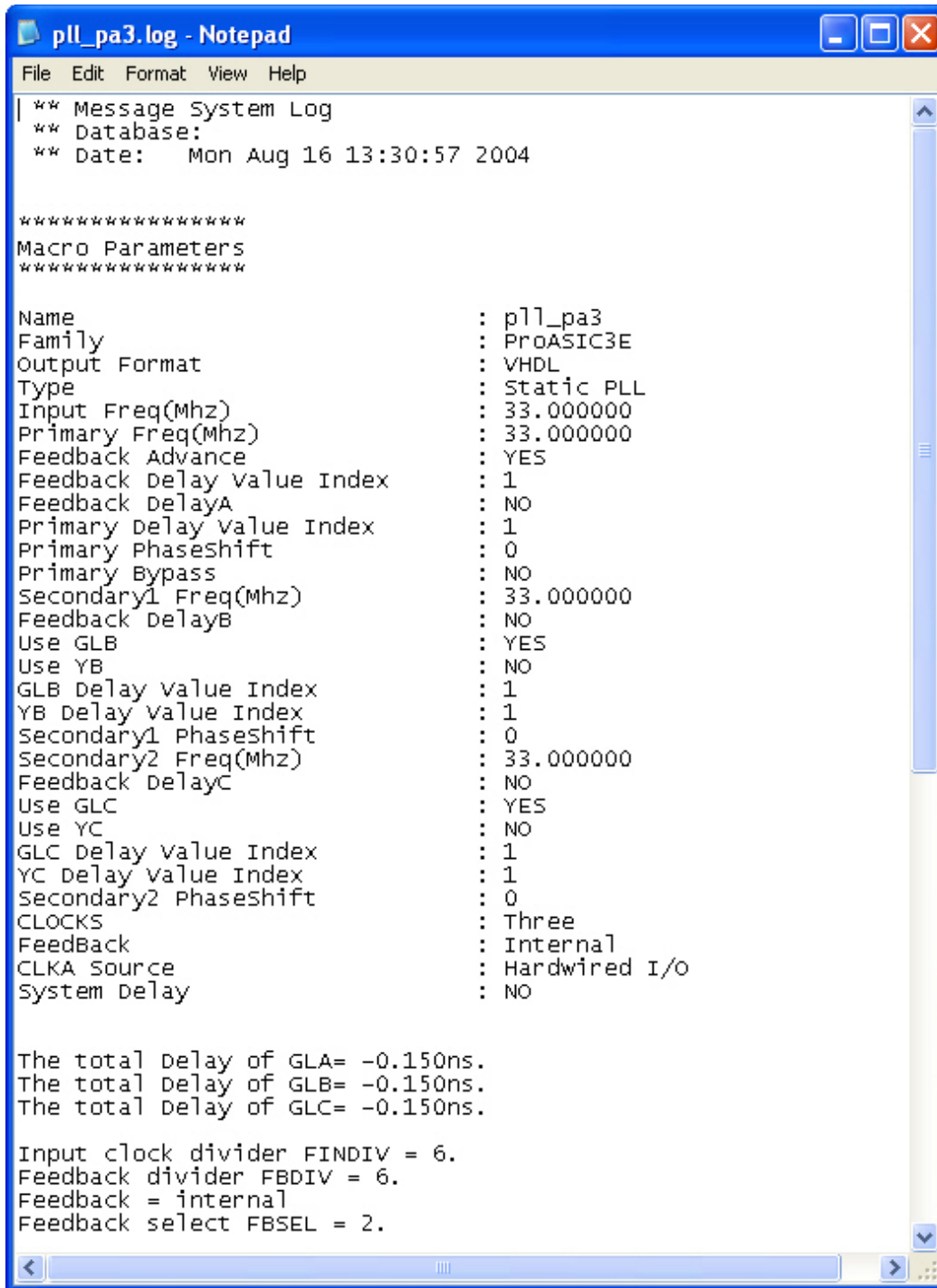
## IGLOO and ProASIC3 Dynamic CCC Implementation Rules / Timing Diagrams

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.



```
pll_pa3.log - Notepad
File Edit Format View Help
| ** 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

Double-click **Clock-Delayed** to open the Clock Conditioning / PLL: Create Core dialog box.

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

SmartFusion, IGLOO, ProASIC3 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

## 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, 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 four, 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  $f_{in}$  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 0° 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 ( $f_{in}/n$ ) must be between 1.5 and 5.5 MHz. The PLL Core output must be between 24 and 350 ( $f_{in} * m/n$ ).

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 118 · 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

Name	Type	Required/Optional	Description
			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.

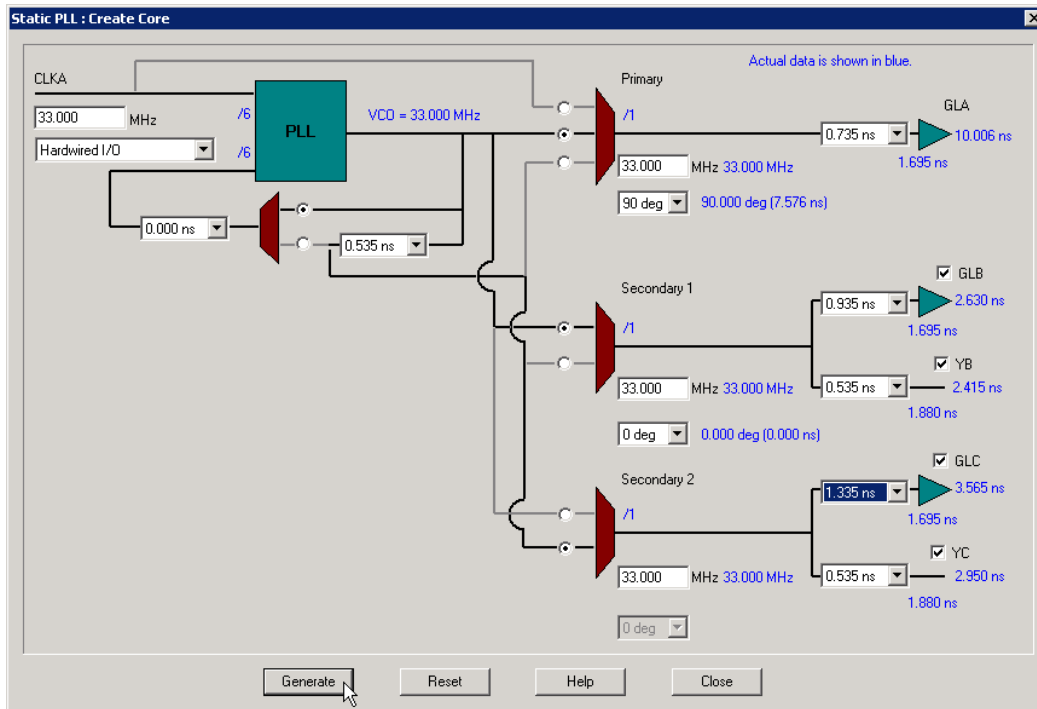


Figure 7 · Configuring the Fusion Static PLL

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 119 · 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
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.

```

pll_pa3.log - Notepad
File Edit Format View Help
| ** 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.

---

# 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](#)).

---

# Welcome to the Analog System Builder (ASB)

---

The Analog System Builder enables you to configure an entire analog system. You can:

- Choose the number of Analog Input Channels to monitor
- Choose the type of each Input Channel
- Choose the number of Analog Output Channels
- Specify the placement of each channel
- Set Channel-specific options
- Sequence the channels in the required sampling order
- Define the operations on converted digital output from the ADC
- Specify the RTC settings.

The ASB is available in the Core Catalog in the Project Manager in Libero SoC.

The Analog System Builder enables you to create, configure, and place the following analog blocks (or "peripherals"):

- [Voltage Monitor](#)
- [Current Monitor](#)
- [Temperature Monitor](#)
- [Differential Voltage Monitor](#)
- [Direct Digital Input](#)
- [Gate Driver](#)
- [Real Time Counter](#)
- [Internal Temperature Monitor](#)
- [Internal Voltage Monitor](#)

## Analog System Builder Reference

The Analog System Builder uses some terminology that may be unfamiliar. Here is a list of terms and acronyms that appear in the software and the help.

Term	Description
ADC	Analog-to-digital converter
ASSC	>Analog sample sequence controller; sets the sample order in the ADC (includes IP + RAM)
Analog System	The complete system, including the analog block (AB) hard IP and one or more of ASSC, SMEV, and SMTR soft IPs.
SMEV	Evaluates the converted analog data (IP + RAM)
SMTR	Processes the evaluated analog data and generates flag signals on certain conditions (includes IP + RAM)
AB	Analog block - The hard macro in the CAE library that includes the analog MUX and the ADC
Analog MUX	The 32 -1 MUX, select signals of which determine the channel being

Term	Description
	sampled by the analog to digital converter.
ACM	Analog Configuration MUX - stores configuration data related to analog channels (channel type, pre-scalar value, polarity, etc.)
FMSB	Flash Memory System Builder
INIT IP	INIT / CFG Soft IP, responsible for all initialization and Save activity to the NVM
ASB	Analog System Block. Analog System top level, includes the Analog Block (AB) and Analog System Soft IP.
FMB	Flash memory block. Flash Memory top level, includes Flash Memory System Builder and INIT IP
FASTCLK	Intended clock during normal system execution
SLOWCLK	Intended clock during system initialization

## ASB Port List

All signals are active high unless explicitly specified.

Name	Type	Description
SYS_CLK	INPUT	Clock input for Analog Block and Soft IP
SYS_RESET	INPUT	Active low asynchronous reset
VAREF	INOUT	Voltage reference; connects to external voltage for external voltage reference. Returns the internal VREF in the case of internal voltage reference. Must be connected to a top-level port without any I/Os.
Analog Input Channels	INPUT	User specified or port names. Analog input channels being used.
Analog Output Channels	OUTPUT	User specified port names; analog output channels being used.
Input Channel Compare Flags	OUTPUT	Threshold flags specified for each peripheral
Flash Memory Block Interface		
INIT_DATA[8:0]	INPUT	Initialization data
INIT_DONE	INPUT	Initialization done signal from the Flash Memory



Name	Type	Description
		Block
INIT_ACM_WEN	INPUT	Initialization ACM write enable
INIT_ASSC_WEN	INPUT	Initialization ASSC RAM write enable
INIT_EV_WEN	INPUT	Initialization SMEV RAM write enable
INIT_TR_WEN	INPUT	Initialization SMTR RAM write enable
INIT_ACM_RTC_WEN	INPUT	Initialization of ACM during second pass; only exported when you use the RTC Peripheral.
Status Signals		
DATAVALID	OUTPUT	Indicates data from the ADC is valid
ASSC_DONE	OUTPUT	Indicates the ASSC is finished processing the current sampling slot
ASSC_WAIT	OUTPUT	Indicates the ASSC is inserting wait states to accommodate SMEV and SMTR processing times.
ASSC_CHSAT	OUTPUT	Sampled channel saturated; indicates that the current channel sampled by the ADC has a value that is saturated (too high for the ADC voltage range). Once this output becomes active, it remains active until the next timeslot is processed. If this output is active, you may need to move up to a higher voltage range (decrease prescaler value for the particular analog input pad).
ASSC_CHLATD	OUTPUT	Channel selector latched; signal indicates that the ADC_CHNR[4:0] (ADC channel selector) value has been latched.

## Analog System Builder Main Window

The Analog System Builder main window enables you to create and configure your analog system (as shown in the figure below).

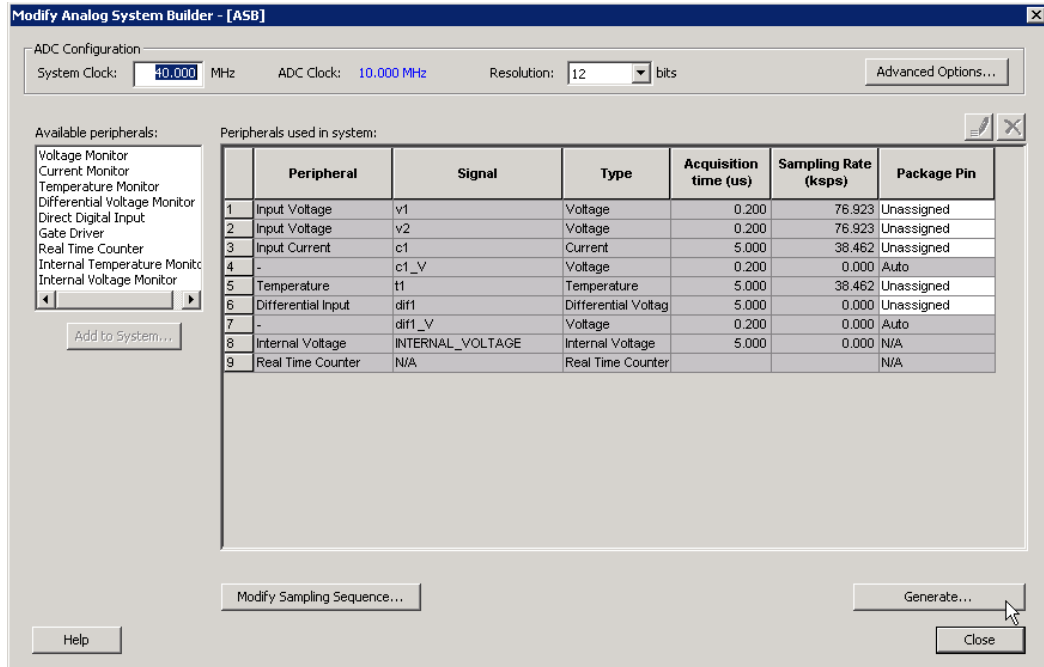


Figure 8 .

Figure 9 · Analog System Builder Dialog Box

The number of peripherals you can add to the system is limited by the size of your device.

The ADC Clock is the frequency at which analog to digital conversions occur. The ASB evaluates the required acquisition times for all peripherals and the system frequency to compute the maximum possible ADC clock frequency. Only certain divider factors exist to create the ADC Clock; because of this and peripheral acquisition times, certain system frequencies result in a faster ADC Clock. Refer to the [Designing with Analog System Builder](#) section for a discussion on Analog System clocks.

You can select the resolution of the ADC (8-, 10-, and 12-bit modes). Selecting the resolution affects the meaningful bits read from the ADCRESULT port, ASSC\_RAM, and SMEV\_RAM.

- In 12-bit mode, the ADC uses 11:0
- In 10-bit mode, the ADC uses 11:2; 1:0 are grounded
- In 8-bit mode, the ADC uses 11:4; 3:0 are grounded

Click [Advanced Options](#) to set your Analog System configuration.

Available Peripherals lists all the analog peripherals you can add to your design. As you add peripherals, some resources are exhausted. If you exceed the resource limit, the ASB returns a warning during file generation. If you want to add additional peripherals, select a larger device, or remove some existing peripherals from your design.

The **Peripherals used in system** grid lists specific information about each peripheral, including

- Peripheral - The type of the peripheral (such as Voltage Monitor, Temp. Monitor, etc.).
- Signal - Name you specified for the signal of your service in the service configuration dialog box.
- Type - Identifies channel type for the service.
- Acquisition Time - The required acquisition time for a given input channel. ASB takes the required acquisition times for all peripherals and computes the maximum possible ADC clock frequency and the number of ADC clocks per sample and per peripheral.
- Sampling Rate (in  $\mu$ s) - This field only displays the sampling rate for the channels specified in the "Main" procedure. See the [Modify Sampling Sequence](#) topic for more information on setting your sampling sequence. See the [Sampling rate in Analog System Builder](#) topic for more information on how the sampling rate is calculated.
- Package Pin - ASB automatically assigns a package pin for each channel in each peripheral added to the system. However, if you require a specific channel for a certain package pin (if you have board layout issues), you can choose a specific pin for that channel.

- [Real Time Counter](#) - You can configure the Real Time Counter so that it functions as a chronometer, allowing it to generate periodic alarms in conjunction with other peripherals (such as the Voltage monitor, etc.).

[Modify Sampling Sequence](#) - Displays the Sample Sequencer. Since there are thirty analog input channels but only one ADC, the channels must be sequenced in the order in which they are to be sampled.

If the Analog System resources you build exceed the total system resources available for your device, ASB issues a warning. You cannot generate a system that exceeds your total system resources. The Analog System Builder also generates a warning if you have a port name conflict between two or more services. You cannot generate a system with port name conflicts.

When you click **Generate the system**, ASB creates [HDL source files, memory \(MEM\) files, configuration files, and log files](#). They all appear in your project folder under the <core\_name> directory. Do not modify any of these generated files or store additional files in this folder. This folder will be recreated every time you overwrite the core.

## Modify Sampling Sequence

Since there are 30 input channels (depending on the device) but only one ADC, the channels must be sequenced in their desired order. There are 64 time slots available for sequencing. You can also run non-sampling operations in the sequencer (such as calibration or powerdown).

Your application requirements dictate the sampling sequence.

The sampling sequence specifies the Analog System's sampling order. For example, the sequence may be specified to sample "voltage channel 1" continuously, or it can be specified to sample "voltage channel 1", "voltage channel 2", "temperature channel 1", and repeat. In either case, the Sample Sequence Controller will drive the ADC signals to sample the channels in the specified sequence.

The sampling sequence has an Automatic Sequence calculation feature. A checkbox to enable or disable this feature is labeled "Allow manual modification of operating sequence". When unchecked, sampling rate requirements may be entered for channels and software attempts to calculate a sequence that meets the rate requirements and satisfies the ordering rules. When unchecked the operating sequence may be specified manually.

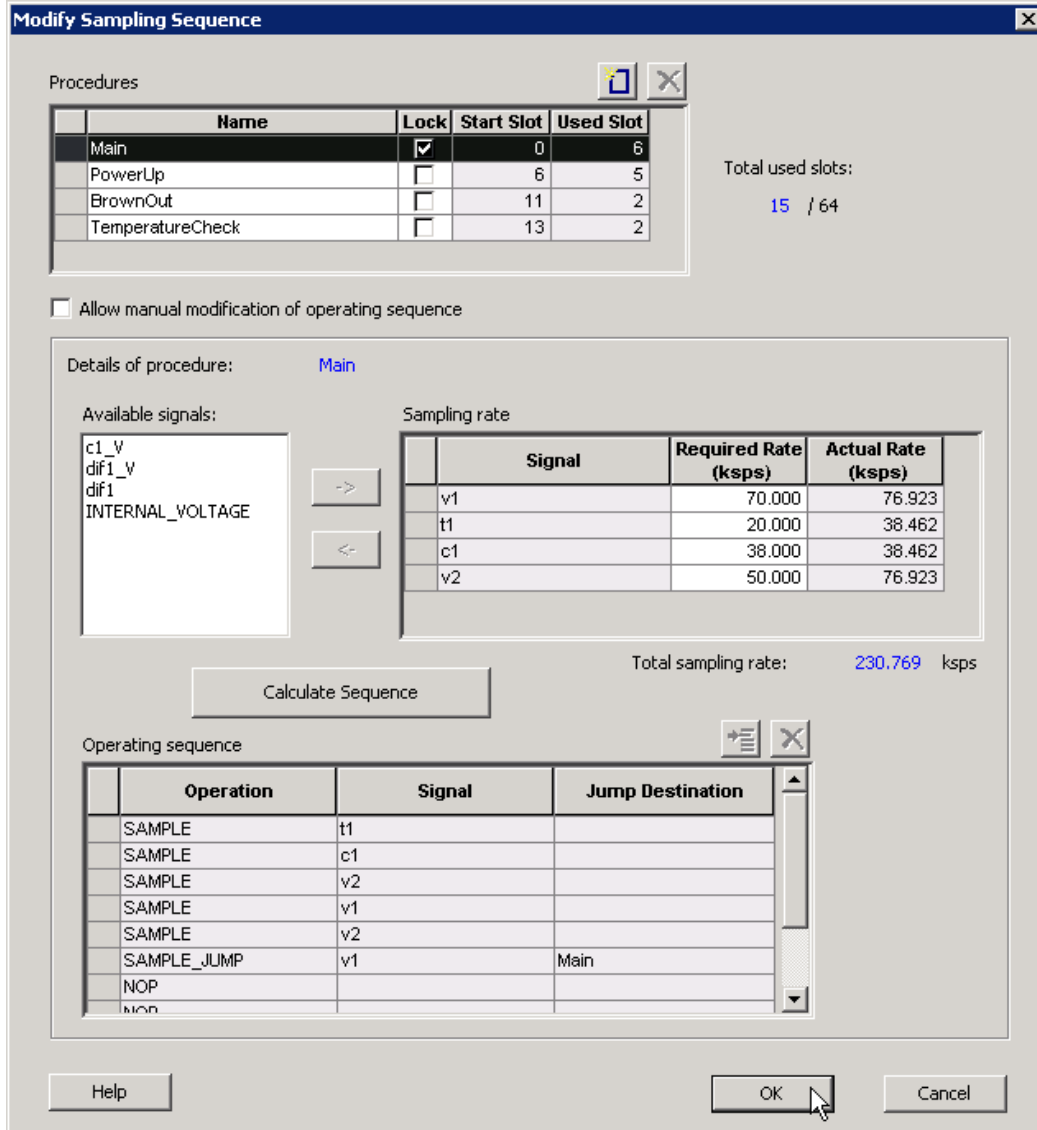


Figure 10 · Modify Sampling Sequence Dialog Box

### Procedures

Procedures are a logical composition of sequences. Each procedure is intended to be completely independent of another procedure.

An example use-model for multiple procedures is a system that requires one set of samples during system power-up, and another set after power-up. For example, upon power-up the system needs voltage channels 1 and 2 monitored. Then after a certain event, such as reaching a stable voltage level, a different set of analog channels need to be monitored.

In this case, you would create 2 procedures: A 'Powerup' procedure that samples voltage 1, voltage 2, and repeat; and a 'SteadyState' procedure that continually monitors the rest of the analog inputs once it has been determined that a steady and stable voltage level has been reached.

The intelligence to determine when to trigger another procedure must be performed by the user through the [external sequencer control interface](#). The External Trigger interface is only exported if there is more than one procedure.

The ability to have multiple procedures that can continually loop upon themselves allows for these types of use-models.

The system defaults to having a single "Main" procedure that can not be deleted or unlocked. It always starts at slot 0 and will always be the procedure that is executed upon reset.

- Name – Enter a name for your procedure
- Lock – Lock the starting slot of the procedure, this is useful if you have an existing design that already has logic to trigger a procedure. If lock is checked, the start slot field is modifiable otherwise it is read-only and software will assign a starting slot for the procedure.
- Start Slot – This is the starting slot number for this procedure. This number is required to trigger this procedure to start executing. Drive this value into the [ASSC\\_SEQIN](#) port.
- Used Slots – Indicates the number of physical slots used up by this procedure. Recall that the sequencer only supports up to a total of 64 slots.
- Total Used Slots – The total number of slots used by all the procedures, if it exceeds 64 it will turn red indicating that a violation has occurred.

Input your procedure values and click the Add Procedure button to create a new procedure. Click the Delete Procedure button to delete a procedure. .

The Operating Sequence and Sampling Rate grids change when you select a procedure.

### Sampling Rate

When **Allow manual modification of operating sequence** is unchecked, you can specify your rate requirements per channel here. The channel must be selected in the list of available signals and added / removed from the grid. After it has been added to the grid, a required sampling rate (in kilosamples per second) may be specified for that channel. Clicking **Calculate Sequence** initiates software to calculate a sequence that most closely meets the your requirement(s).

When **Allow manual modification of operating sequence** is checked, this section is used only to report the sampling rate of the channels. By adding and removing channels to the operating sequence, this grid automatically updates with the actual sample rate for that channel for this procedure. Recall that procedures are completely independent, so the sample rate calculation is performed per procedure and include any of the other procedures.

The actual sampling rate for each channel is displayed in this grid. The total sampling rate indicates the total sampling rate of all channels for the selected procedure.

### Operating Sequence

The operating sequence for a selected procedure. The supported operations are:

- SAMPLE - Sample a channel that is configured in the system and proceed to the next slot
- SAMPLE\_JUMP – Sample a channel that is configured in the system and jump to the start of the specified procedure
- CALIBRATE – Perform a full calibration of the ADC (this requires 3840 ADC Clocks to complete) and proceed to the next slot
- CALIBRATE\_JUMP – Perform a full calibration and jump to the start of the specified procedure
- JUMP – Jump to the start of the specified procedure
- POWERDOWN – Perform a powerdown operation on the ADC; after a powerdown is initiated, a calibration operation is required to resume sampling
- STOP – Stop the sequencer; an external trigger is required to re-start the sequencer
- NOP – No operation is performed and proceed to the next slot. NOP's in the middle of a sequence use up a time slot, but NOP's after the end of the last functional slot do not.

Terminating slots: Each operating sequence must end with a terminating operation. A terminating operation is a SAMPLE\_JUMP, CALIBRATE\_JUMP, JUMP, POWERDOWN, or STOP.

Slots can be inserted or deleted with the Add Slot button or Delete Slot button, respectively.

### Calculate Sequence

The ASB creates a sampling sequence that attempts to meet the sampling rate requirement specified for the procedure.

An additional INTERNAL\_VOLTAGE peripheral with an acquisition and hold time of 5 $\mu$ s may be added to your system to satisfy strobe requirements.

The sequence calculation attempts to fairly balance the sampling rate among the signals by reducing the difference between the actual and required rate.

## External Trigger Signals in Analog System Builder

The external trigger signals are used to control the sequencer from user logic. These signals are exposed if there is more than one procedure in the sequencer.

There are two external jump modes:

- Manual - The system completes operations in the current slot, then waits for the signal to move to the next slot; ASSC\_XMODE = 1
- Auto Forwarding - The system completes the operation in the jump slot and moves to the next slot, until the sequence reaches slot 63, at which point it begins sampling slot 0. If the operation is a STOP or POWERDOWN, the sequencer stops processing until another jump is initiated to a different slot; ASSC\_XMODE = 0

All signals are active high.

Name	Type	Description
ASSC_XMODE	Input	<p>External Trigger Mode: If this input is logic 1, the ADC Sample Sequence Controller will use the ASSC_XTRIG signal to transition to and complete the current sequence timeslot.</p> <p>If this input is logic 0 (default operation for automated sequencing), the internal timeslot counter will be used to automatically advance to the next sequence number.</p>
ASSC_XTRIG	Input	<p>External Trigger: If the ASSC_XMODE input is logic 1 and this input is held at logic 1 for exactly 1 clock cycle, the ASSC block will transition to and complete the current sequence.</p> <p>If the ASSC_XMODE input is logic 0 (default operation for automated sequencing), this input is ignored.</p> <p>If this signal is used to control external triggering, monitor the ASSC_DONE signal to know after which point the ASSC_XTRIG will again have effect.</p>
ASSC_SEQJUMP	Input	<p>Sequence Jump Enable: Setting this signal to logic 1 will jump to the sequence number indicated in the ASSC_SEQIN input pins after the current sequence timeslot has completed.</p>
ASSC_SEQIN[5:0]	Input	<p>Sequence Number In: These inputs are used in conjunction with the ASSC_SEQJUMP signal to jump to a particular sequence number from the current sequence after the current sequence timeslot has completed.</p> <p>These inputs can come from user logic external to the Analog Interface Soft IP blocks, or can be statically tied off to any combination</p>

Name	Type	Description
		of logic 0 and logic 1 values
ASSC_SEQOUT[5:0]	Output	Sequence Number Out: These outputs denote the current sequence timeslot.
ASSC_SEQCHANGE	Output	Sequence Change: This output indicates that the ASSC_SEQOUT outputs are about to change after the very next rising edge of CLK.
ADC_CHNUMBER[4:0]	Output	Channel number being sampled. Refer to the ASB log file for the logical number your peripheral was mapped to.
ASSC_SAMPFLAG	Output	ASSC Sample Function Flag. Indicates that the sample function is active for the currently selected ADC channel.

## Timing Diagrams

The diagrams below show the ASB External Sequencer in full, and a detail of the assert and de-assert states.

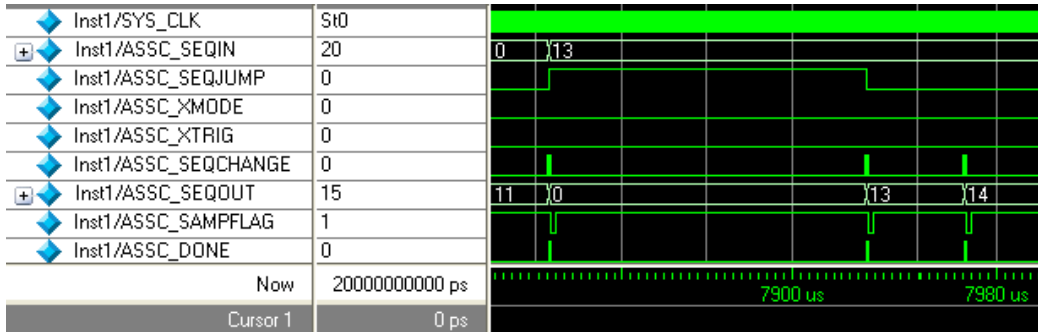


Figure 11 · ASB External Sequencer Control

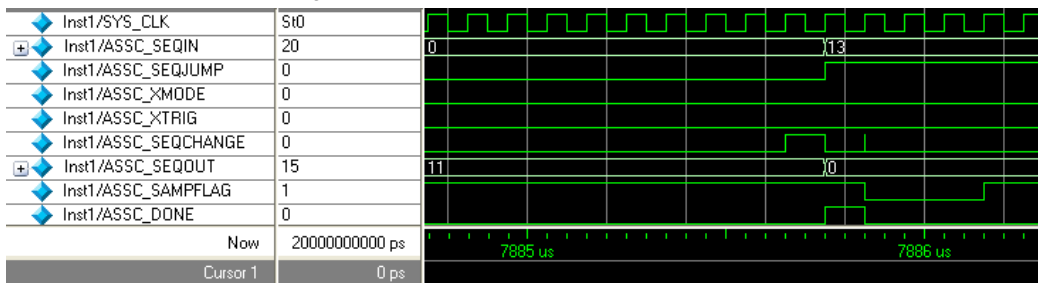


Figure 12 · ASB External Sequencer Control Assert Detail

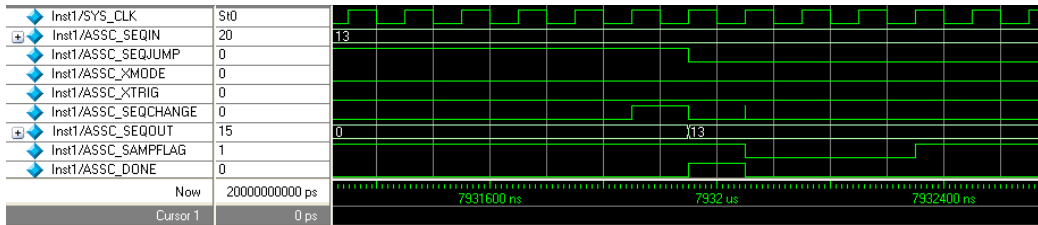


Figure 13 · ASB External Sequencer Control De-Assert Detail

## Analog System Builder Output Files

ASB creates the following output files in your project directory when you generate an analog system.

### HDL Source Files

<user\_name>.vhd/.v – Top-level design that combines all the blocks together

<user\_name>\_assc\_wrapper.vhd/.v – Analog system controller instantiation wrapper for this design

<user\_name>\_smev\_wrapper.vhd/.v – Analog system data processing module instantiation

<user\_name>\_smtr\_wrapper.vhd/.v – Analog system data processing module instantiation

<user\_name>\_assc\_ram.vhd/.v – Analog system controller RAM

<user\_name>\_smev\_ram.vhd/.v – Analog system data processing module RAM

<user\_name>\_smtr\_ram.vhd/.v – Analog system data processing module RAM

<workspace\_directory>/<common>/<Vhdl>/<Verilog>/assc.vhd/.v – Analog sample sequence controller file; common file for all analog system cores.

<workspace\_directory>/<common>/<Vhdl>/<Verilog>/smev.vhd/.v – Analog system data processing module file, common for all analog cores.

<workspace\_directory>/<common>/<Vhdl>/<Verilog>/smtr.vhd/.v – Analog system data processing module file, common for all analog cores.

### Memory Files

These memory files are used by the Flash Memory System (or any external microprocessor) to initialize the contents of the RAM and AB.

<user\_name>\_acm\_ram.hex/ .s - Intel-hex or Motorola S-record memory files for AB Hard IP

<user\_name>\_assc\_ram.hex/ .s - Intel-hex or Motorola S-record memory files for ASSC RAM

<user\_name>\_smev\_ram.hex/ .s - Intel-hex or Motorola S-record memory files for SMEV RAM

<user\_name>\_smtr\_ram.hex/ .s - Intel-hex or Motorola S-record memory files for SMTR RAM

The memory files below are used to initialize the RAM contents for simulation only. These files enable simulation of the Analog system in isolation (there is no need to connect the initialization circuitry).

<user\_name>\_acm\_R0\_C0.mem - Memory File for simulation for AB

<user\_name>\_assc\_ram\_R\*\_C\*.mem - Memory Files for simulation for ASSC RAM

<user\_name>\_smev\_ram\_R\*\_C\*.mem - Memory Files for simulation for SMEV RAM

<user\_name>\_smtr\_ram\_R\*\_C\*.mem - Memory Files for simulation for SMTR RAM

### Configuration Files

<user\_name>.ncf – The embedded Flash configuration file used to communicate information from the Analog System to the Flash Memory system regarding the size of the Analog System Client and the location of the memory content.

<user\_name>.cfg – This captures information about the settings that were specified for the system.

<user\_name>.gen – Enables the software to open the system with your saved specifications.

<user\_name>.cxf – The Core Configuration file that contains information required by the Libero SoC for file management.



### Log Files

The log file contains all the information used to generate your system, as well as any messages related to conflicts or system resource limitations. The file is called <user\_name>.log.

### See Also

[Analog System Builder Calibration output files](#)

## ASB Advanced Options

The Advanced Options in the Analog System Builder (ASB) enable you to set the external reference voltage and generate custom system configurations (as shown in the figure below). Some custom configurations (such as **ADC only**, and **IP cores for ADC sequence control only**) disable some of the functionality in the Analog System.

See the Analog Block Pin Description in the [Fusion Datasheet](#) for more information on these ports.

Setting your External Vref (external voltage) enables you to use a specific Analog to Digital Converter (ADC) reference voltage, allowing more accurate ADC conversions. Vref = VAREF.

This Vref value specifies the voltage reference driven into the Vref interface of the analog block. If you do not enter a value then ASB uses an internal Vref of 2.56V; applying an external Vref affects the [threshold computations](#).

This implies that when modifying the External Vref, the legal range of thresholds for current and voltage may be altered.

**Note:** Note: The legal threshold range for the temperature monitor is not based on Vref.

Therefore, it is possible that you could create a current or voltage peripheral, set up some flags with threshold values, and then decide to use and change the external Vref. This could lead to errors in the existing flag thresholds.

If you invalidate your flag thresholds by setting an external Vref, the ASB main window displays an icon notifying you of errors in the configured peripheral.

**Vref Capacitor value** - Lists values of external capacitors that can be used when generating Vref internally. Select the value that is equal to or greater than the capacitor value used on the board. For information on selecting the Vref capacitor value, please see the [Fusion datasheet](#) and the [Fusion handbook](#).

The chosen capacitor value determines the amount of delay that is inserted before the Analog Block can begin sampling. This delay circuit is automatically inserted by Analog System builder in the form of a counter circuit.

To assist in simulations, a special capacitor value of 0.00  $\mu$ F is provided that enables faster simulations. This selection should be used only for your simulations, as the actual device requires the proper capacitor values to ensure accurate sampling results.

Enabling calibration changes the internal Analog System connectivity and creates additional [output files](#); please see the [ASB - Calibration options](#) for more information on Calibration.

The ASB Advanced Options dialog box enables you to generate the following in your Analog Block:

- IP Cores for ADC data processing and sequence control
- IP Cores for ADC sequence control
- ADC only

Each of these are described below.

### IP Cores for ADC data processing and sequence control

Enables all the Analog Block features: sequencing, flag generation, data averaging, and general ADC management. You can enable or disable access to [ADC results](#), ADC Status [ASSC RAM](#), [SMEV RAM](#), SMEV Status, [ACM Bus](#), and ACM Clock.

ASSC is responsible for setting the sample order in the ADC and SMEV evaluates the converted analog data. This option instantiates the Analog Block and the complete Analog System Controller (includes ASSC RAM, SMEV RAM, and SMTR RAM), as shown in the figure below.

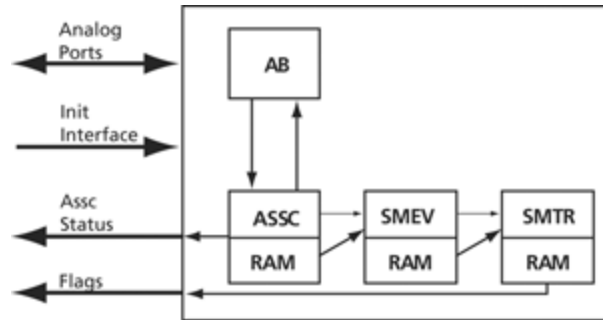


Figure 14 · System Diagram for IP Cores and ADC Data Processing and Sequence Control Options

This option generates the following files:

- ACM MEM files
- ASSC IP, ASSC RAM, ASSC Wrappers, & ASSC MEM files
- SMEV IP, SMEV RAM, SMEV Wrappers & SMEV MEM files
- SMTR IP, SMTR RAM, SMTR Wrappers & SMTR MEM files

Enabling user access to ADC results, ADC status, ASSC RAM, SMEV RAM, and SMEV Status exposes additional interfaces and ports. See the help topics associated with each option for more information.

## IP Cores for ADC Sequence control

This configuration instantiates only the analog block model and the ASSC RAM. The data processing portions of the controller (SMEV and SMTR) are omitted from the design (as shown in the figure below). If you select this option, you must process the ADC data directly from the ADC RESULT bus or the ASSC RAM.

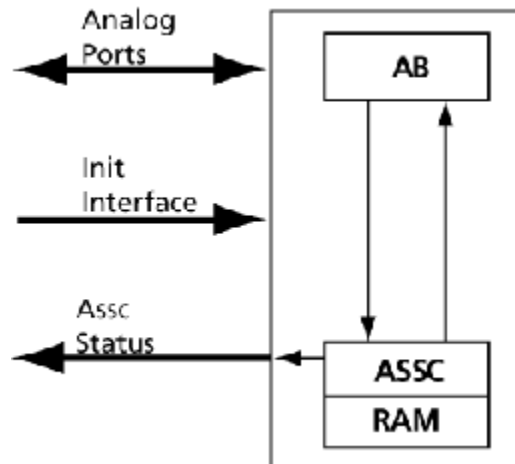


Figure 15 · System Diagram for IP Cores for ADC Sequence Control Only

This configuration disables flag generation for peripherals (the flag grid for peripherals); data averaging (Digital Filtering Factor and Initial Averaging value); SMEV RAM access; and the ability to specify the external resistor in the Current Monitor.

**Note:** Note: You must explicitly choose to expose the ADC result and/or ASSC RAM data interfaces to gain access to the ADC data.

IP cores for ADC sequence control generates the following files:

- ACM MEM files
- ASSC IP, ASSC RAM, ASSC Wrappers, & ASSC MEM files

## ADC only

This configuration instantiates only the Analog Block model (as shown in the figure below). It omits the data processing, sequence controller, and the ADC management features. If you use this configuration you must completely manage the ADC and all related AB functionality.

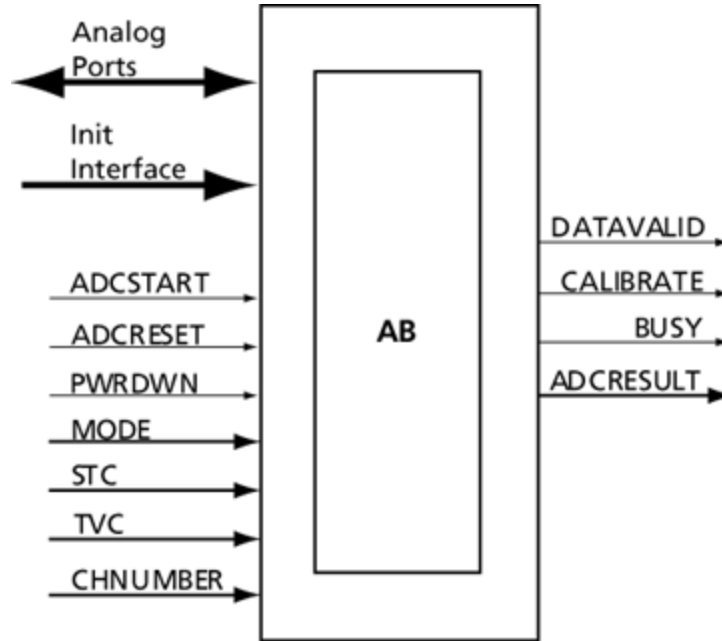


Figure 16 · System Diagram for ADC Only

This option disables sequencing (Sequencer dialog box); flag generation for peripherals (the flag grid for peripherals); data averaging (Digital Filtering Factor and Initial Averaging value); ASSC RAM access, SMEV RAM access; SMEV Status access; and the ability to specify the external resistor in the Current Monitor.

Without the ASSC, you must also manage some general ADC features. They are:

- ADC clock divider (derived from the system clock frequency)
- ADC resolution (resolution combo box on main screen)
- Acquisition time for peripherals (available on each peripheral)

The ADC only option generates ACM MEM files.

### See Also

- [ASB Advanced Options - ASSC RAM](#)
- [ASB Advanced Options - SMEV RAM](#)
- [ASB Advanced Options - SMEV Status](#)
- [ASB Advanced Options - ADC results](#)
- [ASB Advanced Options - ACM Bus](#)

## ASB Advanced Options - Calibration

Analog systems typically require calibration of the analog inputs to achieve more accurate measurements to account for any drift in the manufacturing process. The Analog System Core enables you to include a Calibration IP that performs this function. The Calibration IP performs a “two point” calibration scheme using the formula  $Y = M * X + C$ , where M is the GAIN, X is the Analog converted value, and C is the OFFSET.

Using Calibration changes the default internal Analog System connectivity and creates additional [output files](#). Please see the [ASB connectivity and Calibration](#) topic for an explanation.

## How does Calibration in the ASB work?

During manufacturing each Fusion part is calibrated with the pertinent M (GAIN) and C (OFFSET) data stored inside the Embedded Flash Memory. At run time, these values are pulled automatically from Flash by the Calibration IP and used to perform the Calibration function on each analog conversion. This calibrated value is then passed to the rest of the system.

## Voltage Reference

The M and C data stored into Flash Memory are based on a voltage reference of 2.56V. If you decide to use an external voltage reference that is different than 2.56V then the Calibration option is disabled, and you cannot generate your system with the Calibration IP. Please contact Microsemi technical sales for more information in this case.

## Saturation

There are two options related to saturation of the analog input when using Calibration.

- Calibration is always active and a saturation condition on the analog input can be calibrated and adjusted such that it is no longer in saturation.
- If the analog input is saturated, Calibration is bypassed and the saturated value is passed to the rest of the system.

The ASB - Advanced Options dialog box enables you to select the saturation behavior.

## How does Calibration affect Simulations?

In simulation, the Analog Block core returns an ideal conversion, which means that the data does not need to be calibrated. To account for this, the Flash Memory CAE model is loaded with M (GAIN) and C (OFFSET) data equaling  $M=1$  and  $C=0$ ; in other words, the calibration function simply passes thru the same value.

In the real system, this section of Flash Memory is preprogrammed by Microsemi during manufacturing with the proper M and C data for that particular part, and during real system operation, the actual M and C data will be used.

## Calibration Side Effects

Using calibration requires an extra 14 system clock latency for each conversion (i.e. reduction in sampling rate) and an increase in tile count.

The actual tile count increase depends on your system clock frequency input in the ASB dialog. If it is greater than 60MHz, then the tile count increase is ~470, otherwise it is ~370.

### See Also

[ASB Advanced Options](#)

[ASB connectivity and calibration](#)

[Analog System Builder Calibration output files](#)

## ASB Connectivity and Calibration

Enabling calibration changes the [internal Analog System connectivity](#), as shown in red in the diagrams below, and creates additional [output files](#).

## IP Cores for ADC data processing and sequence control with calibration

Enables all the Analog Block features: sequencing, flag generation, data averaging, and general ADC management. You can enable or disable access to [ADC results](#), ADC Status [ASSC RAM](#), [SMEV RAM](#), SMEV Status, [ACM Bus](#), and ACM Clock.

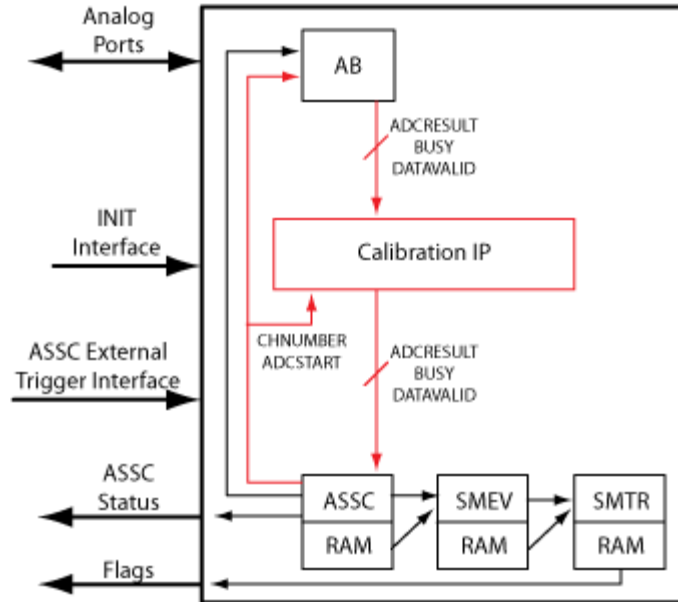


Figure 17 .

Figure 18 · System Diagram for IP Cores and ADC Data Processing and Sequence Control Options

### IP Cores for ADC Sequence control with calibration

This configuration instantiates only the analog block model and the ASSC RAM. The data processing portions of the controller (SMEV and SMTR) are omitted from the design (as shown in the figure below). If you select this option, you must process the ADC data directly from the ADC RESULT bus or the ASSC RAM.

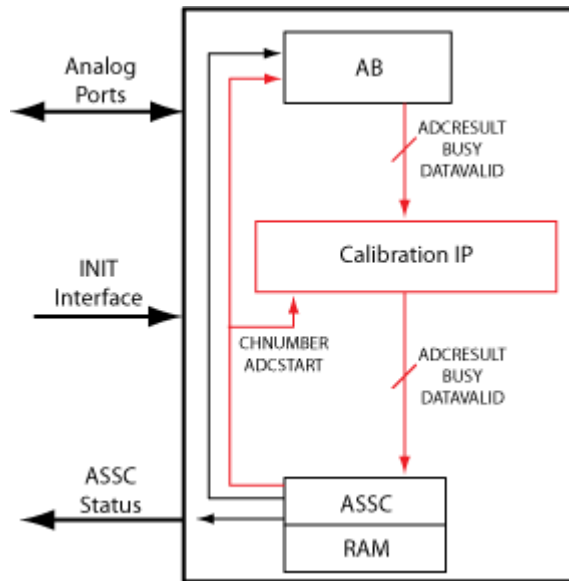


Figure 19 .

Figure 20 · System Diagram for IP Cores for ADC Sequence Control Only

### ADC only with calibration

This configuration instantiates only the Analog Block model (as shown in the figure below). It omits the data processing, sequence controller, and the ADC management features. If you use this configuration you must completely manage the ADC and all related AB functionality. Note that when you export DATAVALID, BUSY, and ADCRESULT signals with calibration, the signals come from the Calibration IP block as shown in the figure below.

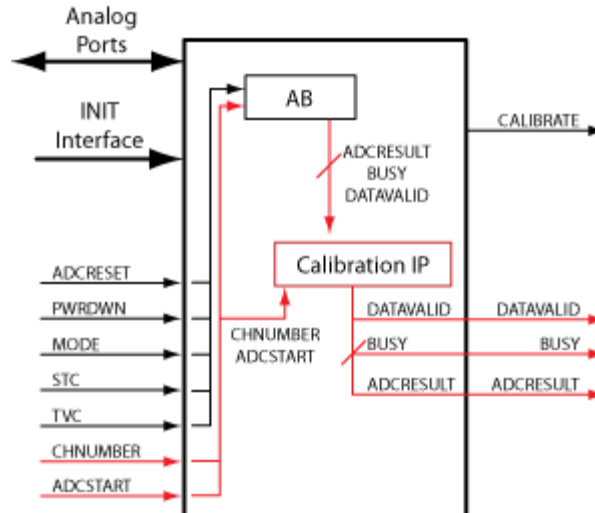


Figure 21 ·

Figure 22 · System Diagram for ADC Only

### See Also

- [ASB Advanced Options - ASSC RAM](#)
- [ASB Advanced Options - SMEV RAM](#)
- [ASB Advanced Options - SMEV Status](#)
- [ASB Advanced Options - ADC results](#)
- [ASB Advanced Options - ACM Bus](#)

## Analog System Builder Calibration Output Files

The ASB Calibration output files are placed in the common folder.

### HDL Source Files (in <workspace\_directory>/<common>/vhdl or /verilog directory)

- calibip.vhd/.v
- calibip\_brentkung\_24.vhd/.v
- calibip\_clram.vhd/.v
- capibip\_compute\_block.vhd/.v
- calibip\_ram512x9\_afs.vhd/.v
- calibip\_ripple\_24.vhd/.v
- /smartgen/<corename>/<corename>\_calibip\_wrapper.v - Top level HDL wrapper for CalibIP

### Memory Files (in /smartgen/<corename> directory)

<corename>\_calibcoefficient.mem - Contains Flash Memory System Builder content for the M (GAIN) and C (OFFSET) coefficient section of Flash Memory System Builder.

This is imported by Flash Memory Builder to generate the Simulation Memory File. It is required to allow simulation to work properly, the M and C data in this file are populated with M=1 and C=0. Since the AB simulation model provides 'ideal' conversion results, no calibration is needed so populating the coefficient data with 1 and 0 emulates that scenario.

<corename>\_calibip\_wrapper.hex - Contains Flash Memory System Builder content for the CalibROM section of the Flash Memory System in IntelHex format.

<corename>\_calibip\_wrapper\_ROC0.mem -Contains Flash Memory System Builder content for the CalibROM section of the Flash Memory System in binary format.

<corename>\_calibrom.mem - Contains Flash Memory System Builder content for the CalibROM section of the Flash Memory System in the correct memory format. This is imported by the Flash Memory System Builder to generate the Simulation and Programming Memory File

## ASB Advanced Options - ASSC RAM

If you choose to enable user access to ASSC RAM in the [ASB Advanced Options dialog box](#), then you can read the contents of the ASSC RAM.

When performing slot processing, the ASSC stores the raw ADC Result value. The address locations for these values are exported in the log file created by the Analog System Builder.

The ASSC stores its result on a per-slot basis. Thus, if your sequencer samples a single channel multiple times, then the samples for that channel will be in multiple locations of the ASSC RAM. However, each sample result could potentially be different since it is sampled at a different time.

The ASSC continually overwrites each slot location every time it processes a sequence slot.

Each data is stored as a 12-bit value, so you must read two RAM locations to retrieve the value.

The exposed ASSC RAM ports are shown in the table below.

Port Name	Input/Output	Description
USER_ASSC_ADDR[8:0]	Input	<b>User RAM Address</b> - You can control these address signals and enable read access from the A-port of the 512x9 ASSC RAM. If unused, these signals must be tied to logic 0 or logic 1.
USER_ASSC_RD	Input	<b>User RAM Read Enable</b> - (Active high) You can control the control signal and enable read access from the A-port of the 512x9 ASSC dual-port RAM (you must connect to the ASSC_RAM_DO_A[8:0] port for read data). If unused, this signal must be tied to logic 0. Make sure that the ASSC_RAM_BUSY signal is inactive at logic 0 when this signal is first activated, otherwise, the data read from the A-port of the ASSC RAM will not be from the USER_ADDR[8:0] address.
USER_ASSC_RAM_BUSY	Output	<b>ASSC RAM Busy</b> - This output signal indicates that either the Init/Config Soft IP block or the System Monitor Evaluation Phase State Machine Soft IP block is busy accessing the A-port of the ASSC RAM. It is Active High. This signal can be used by user logic outside the analog interface soft IP blocks or can be left unconnected if unused.
ASSC_RAM_WR_BUSY_B	Output	<b>ASSC Busy Writing</b> - This active-high signal is for user status monitoring and indicates that the ASSC block is busy writing to the B port of its dual-port RAM. It is Active High.
ASSC_RAM_DOUT	Input	D <sub>out</sub> of Port A of the ASSC RAM

The ASSC memory content report looks like this:

```
*****
ASSC Memory Content Report
*****
```

Slot	Channel	Address	Bits	Value
0	volt1			
		3	[08:00]	Raw ADC Result [08:00]
		4	[02:00]	Raw ADC Result [11:09]

### See Also

- [ASB Advanced Options](#)
- [ASB Advanced Options - SMEV RAM](#)
- [ASB Advanced Options - ADC results](#)
- [ASB Advanced Options - ACM Bus](#)

## ASB Advanced Options - SMEV RAM

If you choose to enable user access to SMEV RAM in the [ASB Advanced Options dialog box](#), then you can read the contents of the SMEV RAM.

When processing channels the SMEV stores the digitally-filtered (i.e., averaged ) ADC Result value into this RAM. The address locations for these values are exported in the log file created by Analog System Builder. Unlike the [ASSC RAM](#), the SMEV stores its result on a per channel basis.

The SMEV will continually overwrite each channel location every time it processes a channel.

As with the ASSC RAM, the data is stored as a 12-bit value, so you must read two RAM locations to retrieve one value.

The exposed SMEV RAM ports are shown in the table below.

Port Name	Input/Output	Description
USER_EV_ADDR[8:0]	Input	<b>User RAM Address</b> - You can control these address signals and enable read access from the A-port of the 512x9 SMEV RAM. If unused, these signals must be tied to logic 0 or logic 1
USER_EV_RD	Input	<b>User RAM Read Enable</b> - You can control the control signal and enable read access from the A-port of the 512x9 SMEV dual-port RAM (you must connect to the EV_RAM_DO_A[8:0] port for read data). If unused, this signal must be tied to logic 0. Make sure that the USER_EV_RAM_BUSY signal is inactive at logic 0 while this signal is first activated, otherwise, the data read from the A-port of the SMEV RAM(s) will not be from the USER_EV_ADDR[EV_ASIZE-1:0] address.
USER_EV_RAM_BUSY	Output	<b>SMEV RAM Busy</b> - This output signal indicates that either the Init/Config Soft IP block or the SMTR Soft IP block is busy accessing the A-port of the SMEV RAM. This signal can be used by user logic external to the analog interface soft IP blocks or can be left unconnected if unused.
EV_RAM_WR_BUSY_B	Output	<b>SMEV Busy Writing</b> - This active-high signal



Port Name	Input/Output	Description
		is for user status monitoring and indicates that the SMEV block is busy writing to the B port of its dual-port RAM.
SMEV_RAM_DOUT	Input	Dout of Port A of the SMEV RAM

The SMEV memory content report looks like this:

```

*****
SMEV Memory Content Report
*****
Channel  Address  Bits  Value
-----
voltage
          75 |  [08:00] |  Averaged ADC Result [08:00]
          76 |  [02:00] |  Averaged ADC Result [11:09]
-----

```

**See Also**

- [ASB Advanced Options](#)
- [ASB Advanced Options - ASSC RAM](#)
- [ASB Advanced Options - ADC results](#)
- [ASB Advanced Options - ACM Bus](#)

## ASB Advanced Options - SMEV Status

This option exposes status signals from the SMEV; these are useful for monitoring the current processing state of the SMEV.

The exposed SMEV Status ports are shown in the table below.

Port Name	Input/Output	Description
EV_DONE	Output	SMEV Done – Indicates that the SMEV is done for the current channel.
EV_EVFLAG	Output	SMEV Active – Indicates that the SMEV is currently in effect for the channel that has just been sampled by the ASSC.
EV_CHHOLD[4:0]	Output	Channel Value – The channel that the SMEV is currently processing

## ASB Advanced Options - ADC results

This option exposes the RESULT port from the AB. It represents the ADC result value that was currently sampled. The exposed port is described in the table below.

The ADC Result bus is a decimal representation of the voltage value for Voltage, Current, and Differential Voltage peripherals. For a temperature peripheral the bus is in degrees Kelvin. See [ASB - Calculating a threshold](#) for more information.

Port Name	Input/Output	Description
ADC_RESULT[11:0]	Output	<p><b>ADC Result</b> - These signals comprise the conversion result from the ADC.</p> <p>In 12-bit mode, the ADC uses 11:0</p> <p>In 10-bit mode, the ADC uses 11:2; 1:0 are grounded</p> <p>In 8-bit mode, the ADC uses 11:4; 3:0 are grounded</p>

### See Also

- [ASB Advanced Options](#)
- [ASB Advanced Options - ASSC RAM](#)
- [ASB Advanced Options - SMEV RAM](#)
- [ASB Advanced Options - ACM Bus](#)
- [ASB - Calculating a threshold](#)

## ASB Advanced Options - ADC status

This option exposes the status ports from the AB.

The exposed ports are:

Port Name	Input/Output	Description
CALIBRATE	Output	ADC Calibration – Indicates the ADC calibration is currently in effect.
BUSY	Output	ADC Busy – Indicates that an Analog conversion is in effect. When this transitions from logic 1 to logic 0, valid conversion data is available on ADC_RESULT.
SAMPLE	Output	ADC Sample – If logic 1 then analog input is sample. Refer to the <a href="#">Fusion datasheet</a> for exact timing of this signal from the AB.

## ASB Advanced Options - ACM Bus

This option provides access to the ACM Address and Data signals to update the values of the counter and match register.

**Note:** Exercise Caution: The ACM address bus is shared by the Analog System and the Analog system configuration could be overwritten if it accesses an incorrect address.

This option exposes the ACM Bus interface allowing users to read / write the ACM registers. Refer to [Designing with Analog System Builder](#) for the suggested connectivity scheme when using this bus.

There are 2 options to accessing the ACM Bus:

1. Independent, or
2. Part of Init/CFG interface

## ACM Bus Independent

If you select this option, generates a MUX internally to multiplex between the Init/Cfg interface and your user interface to the ACM.

Microsemi recommends this option when directly accessing the ACM. The ports that are exposed when selecting this option are:

Port Name	Input/Output	Description
ACMCLK	Input	Clock for ACM interface. The max frequency for this clock is 10 MHz; it must be the same frequency that is used during Initialization from the Flash Memory System.
ACMRDATA_I[7:0]	Output	Data read from the ACM.
ACMADDR[7:0]	Input	Address interface of the ACM.
ACMWDATA[7:0]	Input	Write data to the ACM.
ACMWEN	Input	Write enable to the ACM.

## ACM Bus Part of Init/CFG Interface

If you select this option, you must create your own multiplexors between the Init/CFG interface and your user logic. The INIT\_DONE signal from the Flash Memory System can be used to indicate when the Init/CFG operation is complete. This signal should be used to select the MUX.

The ports related to this interface are:

Port Name	Input/Ooutput	Description
ACMCLK	Input	Clock for ACM interface. The max frequency for this clock is 10 MHz; it must be the same frequency that is used during Initialization from the Flash Memory System.
ACMRDATA_I[7:0]	Output	Data read from the ACM.
INIT_ADDR[7:0]	Input	The ACM bus is shared with the Initialization interface of the Flash Memory System. This port is always exposed; the address of the ACM uses this port as part of the Initialization interface. You must multiplex your ACM address with the Initialization address into this port. The selection of the multiplexer can be based on the INIT_DONE from the Flash Memory System.
INIT_DATA[7:0]	Input	The ACM bus is shared with the Initialization interface of the Flash Memory System. This port is always exposed; the data written into the ACM uses this port as part of the Initialization interface. You must multiplex your ACM write data with the Initialization data into this port. The selection of the multiplexer can be based on the INIT_DONE from the Flash Memory System.
INIT_ACM_WEN	Input	The ACM bus is shared with the Initialization

Port Name	Input/Ooutput	Description
		interface of the Flash Memory System. This port is always exposed; the data write enable of the ACM uses this port as part of the Initialization interface. You must multiplex your ACM write enable with the Initialization write enable into this port. The selection of the multiplexer can be based on the INIT_DONE from the Flash Memory System.

## Accessing RTC Registers

When reading the RTC count or match register, which operates in the XTLCLK domain, the appropriate 40-bit value is first copied to a capture register through clock synchronization circuitry, if and only if the least significant byte of that set of register is addressed. Higher-order bytes of the same set of registers captured with the LSB can then be read on immediately later read cycles. Higher-order bytes of that set of registers can be read in any order but must be read before switching to a different set of registers to ensure data consistency.

For example, RTC counter address ranges from 0x40 to 0x44, register 0x40 must be accessed first before accessing addresses 0x41, 0x42, 0x43, and 0x44 to get the full 40-bit value.

See the [Fusion Datasheet](#) for the detailed register description and how to setup the CTRL/STAT register for RTC read and write.

### See Also

[ASB Advanced Options](#)

[ASB Advanced Options - ASSC RAM](#)

[ASB Advanced Options - SMEV RAM](#)

[ASB Advanced Options - ADC results](#)

ACM Register Map in [Fusion Datasheet](#): ACM Address Decode Table for Analog Quad

## ASB Advanced Options - ACM Clock

This option exposes the ACM Clock to the top level.

See [Designing with the Analog System](#) to view Microsemi's recommendations on clock schemes with Fusion designs.

The port that will be exposed when using this option is:

Port Name	Input/Output	Description
ACMCLK	Input	Clock for ACM interface. The max frequency for this clock is 10 MHz; it must be the same frequency that is used during Initialization from the Flash Memory System.

## ASB - Calculating a Threshold

This section describes the ASB threshold conversion logic. These equations are performed by ASB and do not need to be employed by you unless you are directly monitoring the raw ADC RESULT values.

Scaling Factor is determined based upon the selected prescaler range. Refer to the [Prescaler range](#) topic for a list of prescaler range values.

The value of each LSB bit is determined by dividing VREF by  $2^{12}$ . This calculation is always performed with 12-bit mode, the masking determines the resolution (see below).

## Voltage Monitor

Calculated Threshold = (UserThreshold(V) \* ScalingFactor \* Value of each LSB)

## Current Monitor and Differential Monitor

The Differential Voltage Monitor uses the same block as the Current Monitor so it has the same gain.

Calculated Threshold =  
(user threshold(A) \*  
Resistor Value(Ohm) \*  
Gain Applied by Current Monitor) \*  
(Value of each LSB bit)

Gain = 10 for Current Monitor and Differential Voltage Monitor

Then masking is performed; see below.

## Temperature Monitor

The temperature value from the ADC is in degrees Kelvin. For example, using the internal VREF of 2.56V and 10-bit resolution, each LSB of the ADC result = 1K. This is generalized for internal VREF to:

8-bit mode -> 1LSB = 4K

10-bit mode -> 1LSB = 1K

12-bit mode -> 1LSB = .25K

For example:

8-bit mode -> ADC LSB = 01001010b (74) =  $74 * 4 = 296K$

10-bit mode -> ADC LSB = 01001010b (298) =  $298 * 1 = 298K$

12-bit mode -> ADC LSB = 0001001010 (298) =  $298 * .25 = 74.5K$

The general equation is:

$K = 2.30258 * 0.0000087248$  ( derived from diode equation )

Note: These constants are derived from

Calculated Threshold = (user threshold (K)) \* (K) \* (Gain Applied by Temperature Monitor)

Gain = 12.5 for Temperature Monitor

Then masking is performed (see below).

## Masking on Resolution

The calculated threshold is masked depending upon the resolution.

- 12-bit resolution - No bits are masked
- 10-bit resolution - Bits 0 & 1 are set to '0'
- 8-bit resolution - Bits [3:0] are set to '0'

## Prescaler Range

The prescaler range is determined from the maximum input voltage field. See the table below to calculate your prescaler range.

Scaling Factor: Pad to ADC Input	LSB for 8-bit conversion (mV)	LSB for 10-bit conversion (mV)	LSB for 12-bit conversion (mV)	Full-Scale Voltage	Range Name (V)
0.15625	64	16	4	16.368	16
0.3125	32	8	2	8.184	8
0.625	16	4	1	4.092	4
1.25	8	2	0.5	2.046	2
2.5	4	1	0.25	1.023	1
5.0	2	0.5	0.125	0.5115	0.5
10.0	1	0.25	0.0625	0.25575	0.25
20.0	0.5	0.125	0.03125	0.127875	0.125

Figure 23 · Prescaler Range

If the maximum input voltage is greater than the given range, it will select the higher range.

The corresponding ranges for negative polarity is the same.

The prescaler logic of the AB has a settling time of 10 $\mu$ s (max). It is an application-dependent setting and must be accounted for by you via the acquisition and hold time for each channel. A recommended default value is inserted by the ASB when configuring a new Voltage Monitor but it may be reduced with a possible reduction in sampling accuracy. See the [Fusion datasheet](#) for information.

## Acquisition Time

The required settling/sampling time for this channel. It is the amount of time the Sample and Hold circuit in the ADC charges the capacitor with the input analog signal. The characteristics of your system and monitoring requirements will determine this value. Note also that this value has a direct correlation to the achievable sampling rate of the system.

The prescaler logic of the AB has a settling time of 10  $\mu$ s (max). It is an application-dependent setting and must be accounted for by you via the acquisition and hold time for each channel. A recommended default value is inserted when configuring a new Voltage Monitor but it may be reduced with the possible reduction in sampling accuracy. See the Fusion datasheet for information.

### See Also

[Sampling rate in Analog System Builder](#)

## ASB Channel Mapping

In Analog System Builder you must specify names for each peripheral you configure. For example, you can name your Voltage Monitor “MYVOLT” and it the name will appear in your netlists as “MYVOLT”.

However, the analog block (AB) names Analog Ports AV0, AV1, ... , AC0, AC1, ... AT0, AT1, etc. Also, each port is physically tied to a specific pin on the package. For example, AV0 is physical pin 99 on the package; AV1 is physical pin 103 on the package, etc.

The physical pin placement is communicated directly through the netlist. So, when Designer imports a netlist with an AB, it infers the pin placement from the netlist.

Thus, the core has to map your logical channel into the physical channel on the AB.

**Note:** Note: If you modify the die or package of a project the ASB automatically reverts all user-assigned pins to UNASSIGNED.

## Mapping Requirements

The mapping rules are as follows:

- Voltage Monitors can be placed on AV, AC, or AT pads; Voltage Monitors placed on AV and AC pads can only be of the range -10.5 to 12V. Voltage Monitors on AT pads can only have prescalers in the 4V and 16V range.
- Current Monitors can be placed only on AC pads with the additional requirement that the adjacent AV pad is also available. This is due to the fact that a Current Monitor is an AC-AV pair.
- Temperature Monitors can be placed only on AT pads
- Gate Drivers can be placed only on AG pads
- Digital Inputs can be placed on AV, AC, or AT pads
- Internal channels do not have physical pin mappings, they reside purely on-chip.
- See the Fusion datasheet for the number of peripherals available for each device.

## Designing with the Analog System

The Analog System is initialized via the Analog Configuration MUX (ACM). The ACM supports a maximum frequency of 10 MHz. Therefore, Analog System initialization is limited to less than 10 MHz.

The [PLL](#) and [NGMUX](#) can be used as needed to enable initialization to run at less than 10MHz and normal operation at greater than 10MHz.

### See Also

[Analog System clocks](#)

[ASB and FMB basic configuration](#)

[ASB with RTC](#)

[ASB with ACM access](#)

## Analog System Clocks

The following is a short summary of the important clocks when you use and configure the Analog System.

### Initialization Clock (SLOWCLK)

System initialization frequency. Any save operations occur at this frequency.

The INIT IP was designed to operate only at slow frequencies for initialization and save operations. A large 64 to 1 data mux exists in this design so as the number of clients increase, the achievable frequency decreases.

Microsemi recommends that you keep the initialization frequency below 10Mhz. This is an application-dependent decision unless you are accessing the ACM port interface.

The initialization frequency must be less than 10 Mhz when you use the Flash Memory Block to initialize the ASB (because it accesses the ACM port interface).

### ACM Clock

Analog Configuration MUX frequency. This block resides in the AB library macro. Its purpose is to house the registers for Analog I/O configuration and RTC configuration.

The ACM port interface has a max frequency of 10 Mhz. This is a silicon requirement and must be adhered to whenever using the ACM port interface.

If the ACM is already configured (during initialization) and the interface is not used during normal operation, it is safe to connect this clock to a frequency greater than 10 Mhz, however the ACM is inoperable in this

state. The AB library macro reports warning messages during simulation about the frequency being exceeded but they can be safely ignored at this stage.

## System Clock (FASTCLK)

This is the frequency that is entered into ASB dialog box. It is the frequency at which the ASB and FMB run during normal execution (after initialization is complete).

ASB needs this information because it calculates the ADC Clock from this frequency (see ADC Clock below).

## ADC Clock

All ADC conversions operate at this frequency. Silicon requirements limit it to less than 10 Mhz. This has no relationship to the ACM Clock.

This clock is completely internal to the AB block. ASB sets this clock. The maximum ideal frequency for the ADC is based on the System Frequency entered in the ASB during Analog System configuration. This is based on the acquisition times entered for each peripheral.

There are a limited number of dividers available in the AB macro for this clock calculation, specifically 0, 4, 8, 12, 16, etc. This implies that certain System Clock settings result in faster ADC Clock frequencies. For example, a System Clock frequency of 40 Mhz enables a maximum possible 10 Mhz ADC Clock, whereas a 50 Mhz System Frequency results in a slower ADC Clock.

## Real Time Counter (RTC) Clock

The RTC Clock is the clock frequency for the Real Time Counter logic on the chip. This is modeled inside the AB library macro.

The RTC Clock is exposed on the ASB if the RTC peripheral is configured within the tool (i.e. not configured externally). Connect this port to the CLKOUT of the XTLOSC macro if in use, otherwise it must be grounded.

This clock has no relationship with any of the other clocks listed here.

## Flash Memory Block (FMB) User Clock

When using the FMB with a data storage client, an extra clock port (USRCLK) is exposed on the top level. This clock has a maximum frequency of 100 Mhz because of the NVM.

This clock is intended as the frequency at which the user accesses the NVM. It is muxed with the INIT IP functionality. The arbitration gives INIT IP highest priority.

This clock has no relationship with any of the other clocks listed here.

### See Also

[Designing with Analog System](#)

[ASB and FMB basic configuration](#)

[ASB with RTC](#)

[ASB with ACM access](#)

## Analog System Builder and Flash Memory Block Basic Configuration

The diagram below illustrates the recommended system configuration when using the Analog System Builder and Flash Memory Block (FMB). The ASB and FMB shown below are the default exported systems when using basic ASB features and using FMB with an Analog client only.

You can use a PLL to create the two clock frequencies necessary (FASTCLK and SLOWCLK), the GLA and GLC of this PLL must then be fed into a NGMUX for clock switching. During initialization the SLOWCLK is used to drive both FMB and ASB subsystems, and after initialization the clock switches to the faster clock (as shown in the figure below).



This clock switching is required because of the silicon requirement on the ACM interface. However, once the initialization of the ACM is complete, the ACMCLK can be driven with the faster clock. The library model issues warnings indicating that this clock is being driven faster than 10 Mhz but these warnings can be ignored after the initialization phase, as long as the ACM will not be accessed while being clocked greater than 10 MHz.

If these warnings are unwanted the ACMCLK can be tied directly to the SLOWCLK. To export the ACMCLK open the [Analog System Builder Advanced Options dialog box](#) and [export the ACM clock](#).

If you export the ACM clock, the system exports an ACMCLK port for the Analog System. GLC on the PLL can then be directly connected to the ACMCLK. This eliminates the simulation warning messages.

The INIT\_DONE from the FMB is used for the selection of the NGMUX.

In the following diagrams “AS IP” and “AS RAM” consist of the Analog System Soft IP modules and their associated RAM blocks.

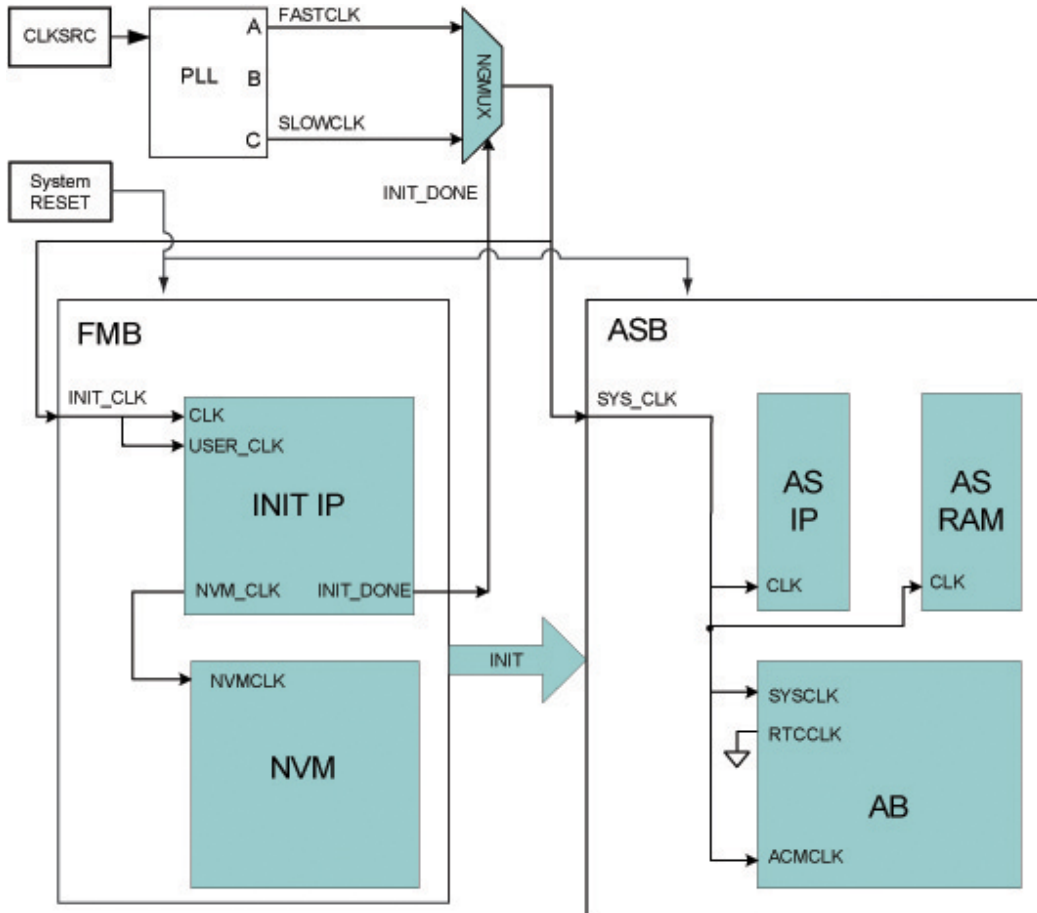


Figure 24 · ASB and FMB Basic Configuration

## Analog System Builder with Real Time Counter (RTC)

When using the RTC in the Analog System Builder (ASB), an extra RTCCLK is exposed on the ASB core that must be properly connected.

This port requires a connection from the CLKOUT of the XTLOSC. The output signals RTCXTLSEL and RTCXTLMODE must also be connected from the ASB module to the XTLOSC module.

The diagram below illustrates this use model.

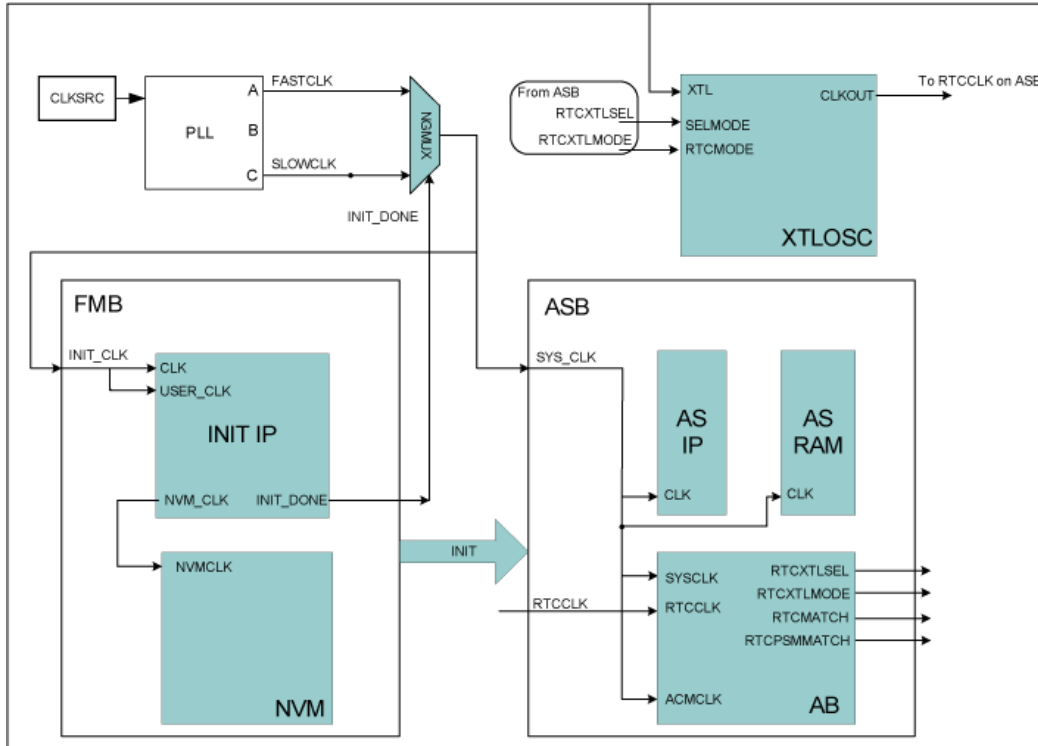


Figure 25 · Analog System Builder with RTC

## Analog System Builder with ACM Access

This use model is effective if you require access to the ACM during normal operation. The primary use is to access the Real Time Counter (RTC) registers for calendar or real-time applications. Alternatively, the ACM can be used to access the Analog I/O configurations.

In this configuration, the ASB contains additional ports to access the ACM. Refer to [ASB Advanced Options – ACM Bus](#) for more information.

The interface to the ACM must be run at less than 10Mhz, and the User block must run at the same clock frequency as SLOWCLK.

Another option is to place some synchronization logic in between your own block and the ACM interface. This enables you to run your own logic at a clock frequency independent of the ACM. shown below describes the setup if you select the ACM Bus as part of the Init/CFG interface.

If you choose the ACM Bus Independent option (in Advanced Options) then the multiplexer shown below is instantiated directly in the Analog System block (ASB below) and you need to hook directly onto the ACM Bus interface.

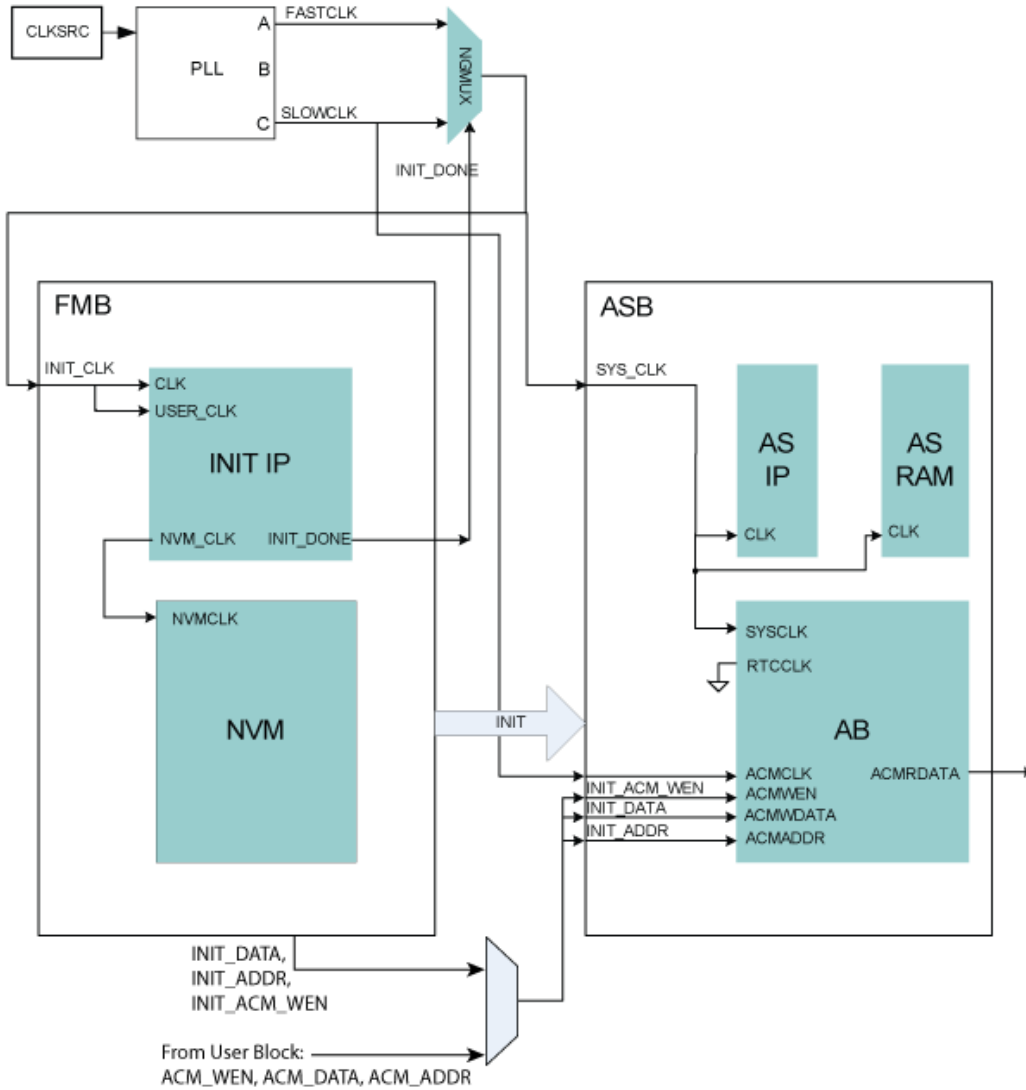


Figure 26 · Analog System Builder with ACM Access

---

# Analog System Builder Peripherals

---

## Current Monitor

The current monitor in Fusion measures current by measuring the differential voltage across a resistor between a pair of Voltage and Current Input channels. This peripheral requires two channels, one of type V and one of type C, and they must be on adjacent package pins.

The differential voltage is multiplied by 10x before it is applied to the ADC; there is no pre-scaling on the differential voltage measurement. The difference in voltages must be less than the value of Vref, external or internal. You must choose an [external resistor](#) that satisfies this condition.

The differential amp gain in the current monitor is 10X. ASB assumes a series resistor because it is being used to measure current. The differential amplifier measures the potential/voltage drop (256 mV max if the reference is 2.56V) across the resistor, which is proportional to the current flowing in the direction AV -> AC ( $I = (\text{change in voltage})/\text{resistance}$ ). The voltage channel in the pair can be used as a voltage monitor to measure the actual voltage that is connected to the Voltage channel.

See the [Configuring Current, Voltage, and Temp peripherals](#) for information on the digital filtering factor, Acquisition time, and Comparison Flag Specifications.

Controls unique to this peripheral are the [External resistor](#) and [Maximum voltage](#).

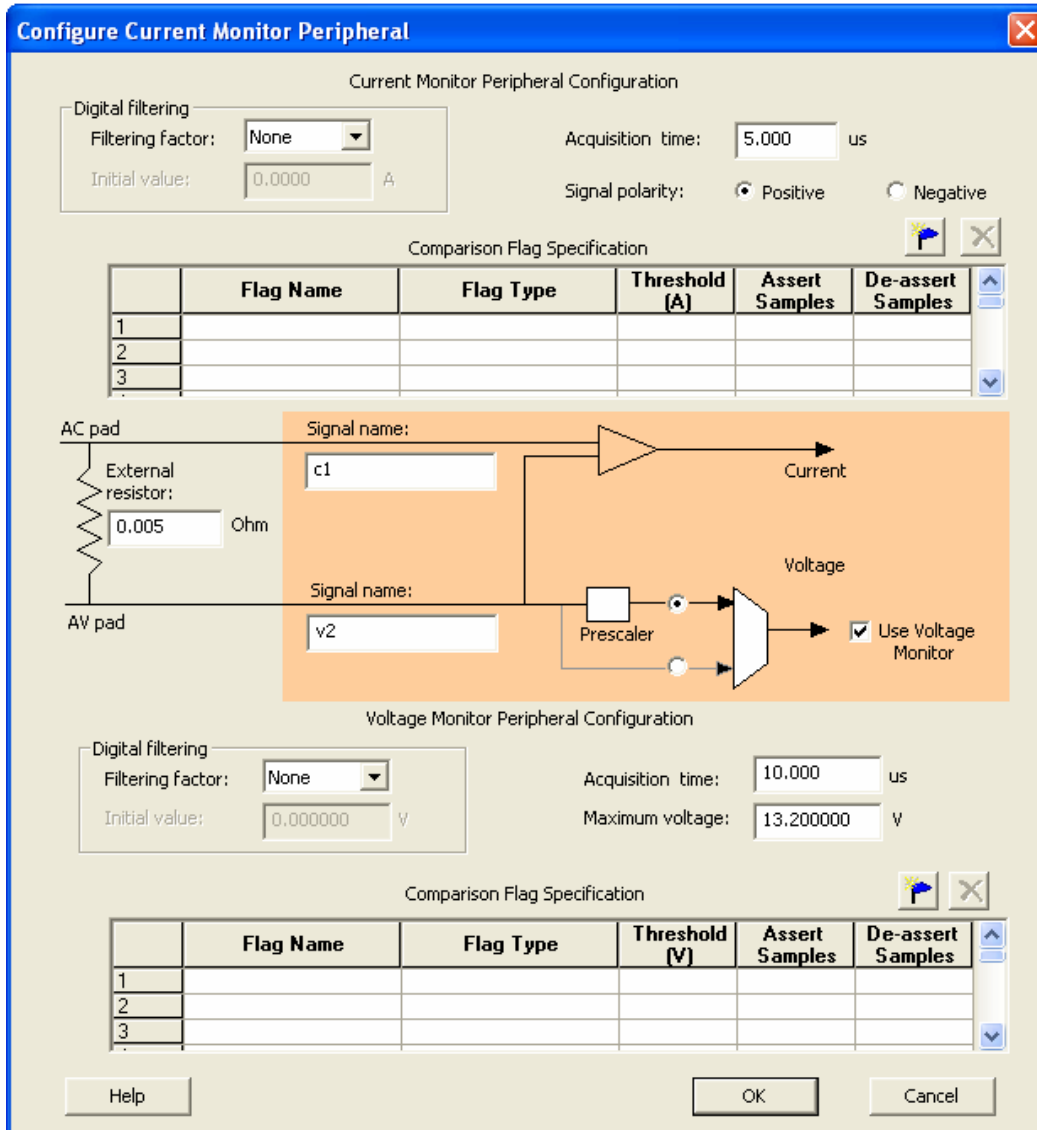


Figure 27 · Configure Current Monitor Peripheral Dialog Box

**External Resistor** - The value of the resistor that is connected across the Current-Voltage pair, external to the device. ASB uses this value to convert the thresholds into voltages.

**Polarity** - Sets the polarity to Positive or Negative. The associated Voltage values must match this polarity. ASB returns a warning if the Voltage values do not match the polarity. When you select a negative polarity, the prescaler option on the associated Voltage Monitor defaults to the "Prescaler" path.

**Maximum voltage** - The maximum anticipated voltage measured by this Voltage Monitor peripheral pad. The range is -10.5V to +16V (the voltage range is NOT bipolar). The ADC is capable of measuring a voltage range of 0 - Vref. For the Internal voltage reference, this value is 2.56V. ASB automatically configures the prescaler in the AB Analog Block for this peripheral to maximize the available voltage range. ASB also post-scales the digital result of ADC conversion so that it returns a result in your specified range.

## Differential Voltage Monitor

The differential voltage monitor uses the same components as the [Current monitor](#).

The differential voltage monitor in Fusion measures the differential voltage between a pair of Voltage and Current Input channels. This peripheral requires two channels, one of type V and one of type C, and they must be on adjacent package pins.

The differential amp gain in the differential voltage monitor is 10x. The differential amplifier measures the potential/voltage drop (256 mV max if the reference is 2.56V) across the two inputs.

The voltage channel in the pair can be used as a voltage monitor to measure the actual voltage that is connected to the Voltage channel.

See [Configuring Current, Voltage, and Temp peripherals](#) for information on the digital filtering factor, Acquisition time, and Comparison Flag Specifications.

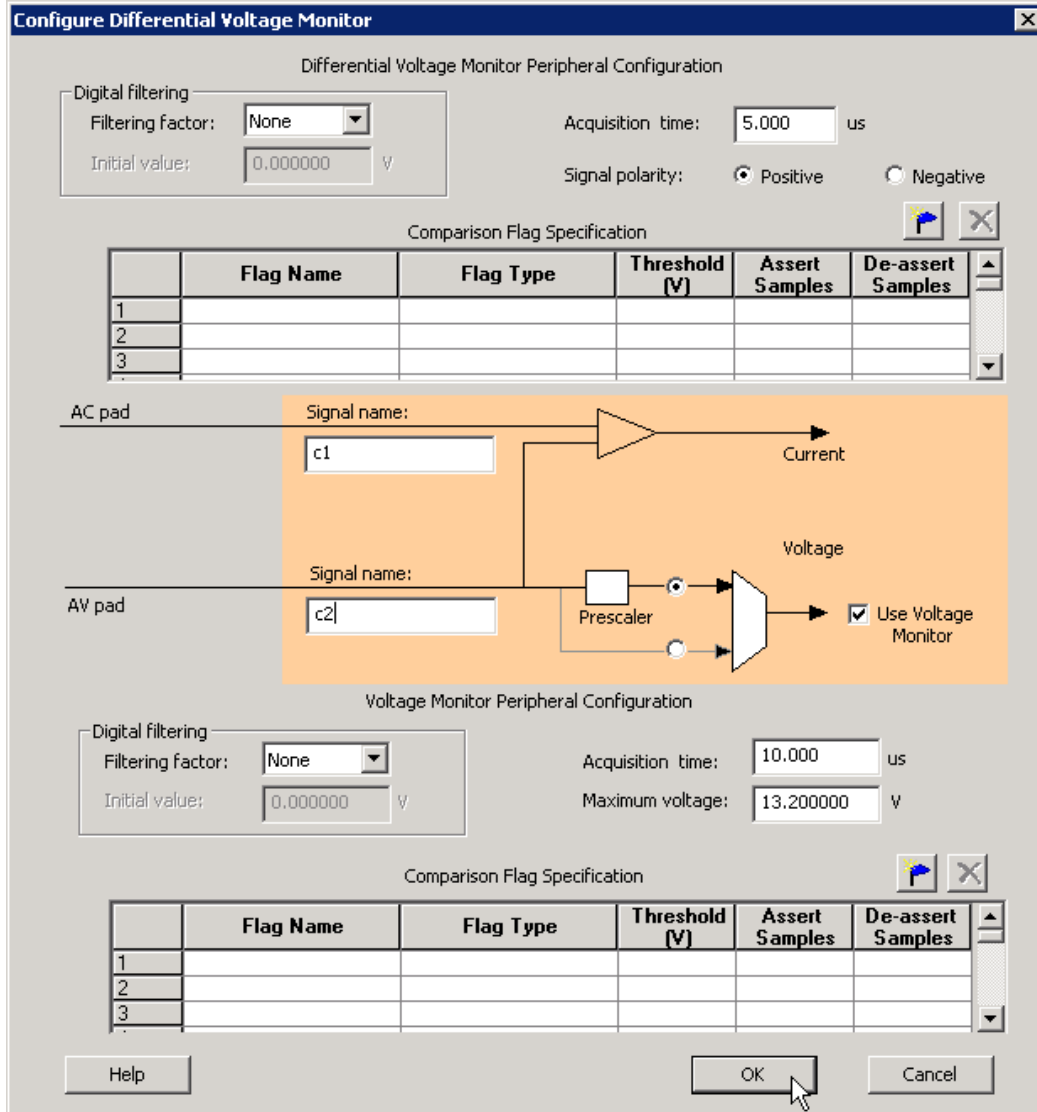


Figure 28 - Differential Voltage Monitor

**Polarity** - The polarity of the peripheral can be set to Positive or Negative. The associated Voltage values must match this polarity. The ASB returns a warning if the voltage values do not match. When you select a negative polarity, the prescaler option on the associated Voltage Monitor defaults to the "Prescaler" path.

**Maximum voltage** - The maximum anticipated voltage measured by this Voltage Monitor peripheral pad. The range is -10.5V to +16V (the voltage range is NOT bipolar). The ADC is capable of measuring a voltage range of 0 -  $V_{ref}$ . For the Internal voltage reference, this value is 2.56V. ASB automatically configures the prescaler in the Analog Block for this peripheral to maximize the available voltage range.

## Direct Digital Input

You can use the Direct Digital Input to configure the unused analog channels or peripherals as digital inputs. Specify the input and output signal name and add the inputs to your sampling sequence. The digital input can be up to 12V but is limited to a frequency of 10 MHz.

Typical delay for the digital input is 10ns.

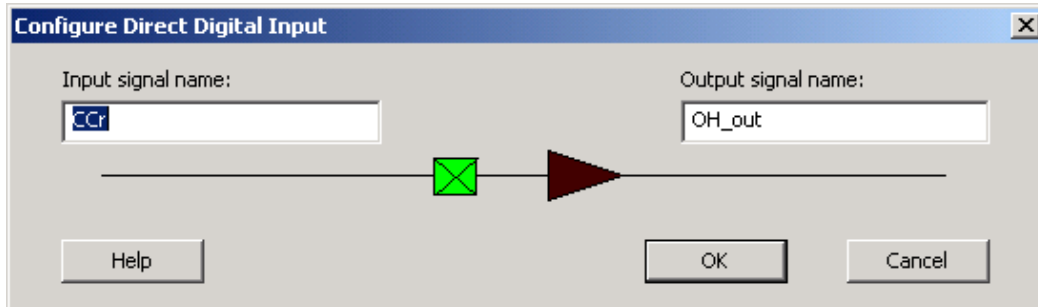


Figure 29 · Configure Direct Digital Input Dialog Box

## Gate Driver

The Gate Driver is the analog output coming from the analog system. You can use it to drive the gate of an external MOSFET on or off. The Gate Driver is designed to work with external MOSFETs as a configurable current sink or source.

The figure below shows the Gate Driver dialog box.

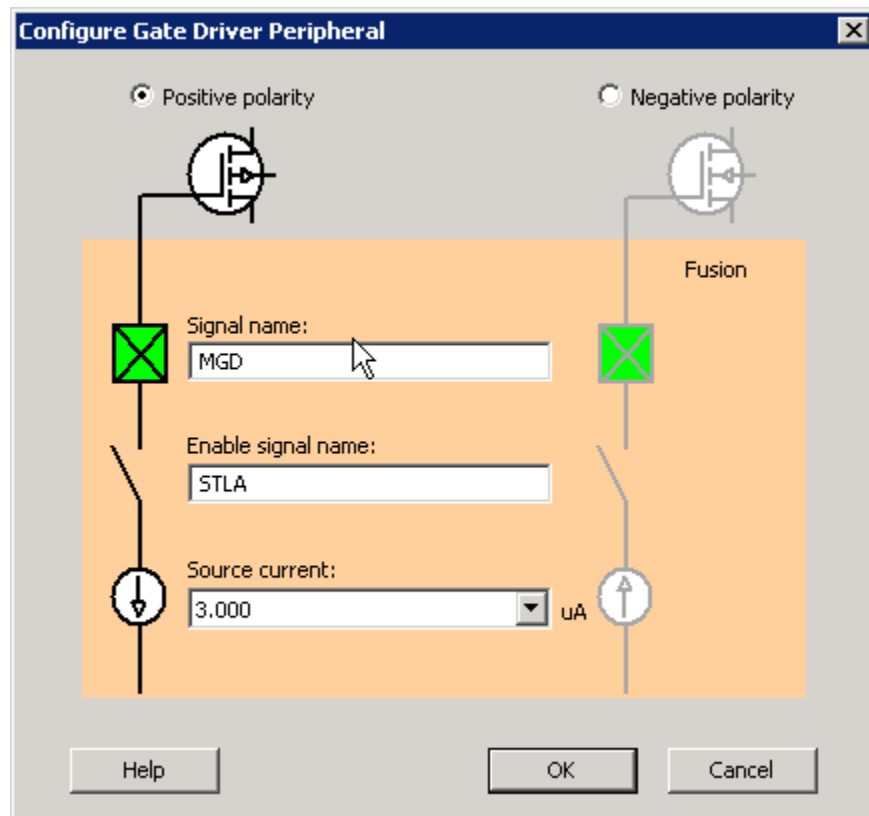


Figure 30 · Output Gate Driver Dialog Box

### Gate Driver Polarity

Indicates the type of MOSFET controlled by the gate driver (NMOS or PMOS).

PMOS controls positive voltage supply and NMOS controls negative voltage supply. The AG pad can work with either P-Channel (pulling down to ground) or N-Channel (pulling up to ground) devices.

**Signal Name**

The name of the signal assigned to this gate driver. It appears as the final port name in the generated system.

**Enable Name**

The name of the signal that enables the gate driver

**Source Current**

The amount of current the device is expected to source. The Gate Driver supports selectable current drive levels: 1  $\mu$ A, 3  $\mu$ A, 10  $\mu$ A, 30  $\mu$ A, and 25mA.

## Internal Temperature Monitor

The internal temperature channel measures the internal die temperature (as shown in the figure below). The peripheral is mapped to logical channel 31 and does not have an associated physical pin.

See the [Configuring Current, Voltage, and Temp peripherals](#) for information on the digital filtering factor, Acquisition time, and Comparison Flag Specifications.

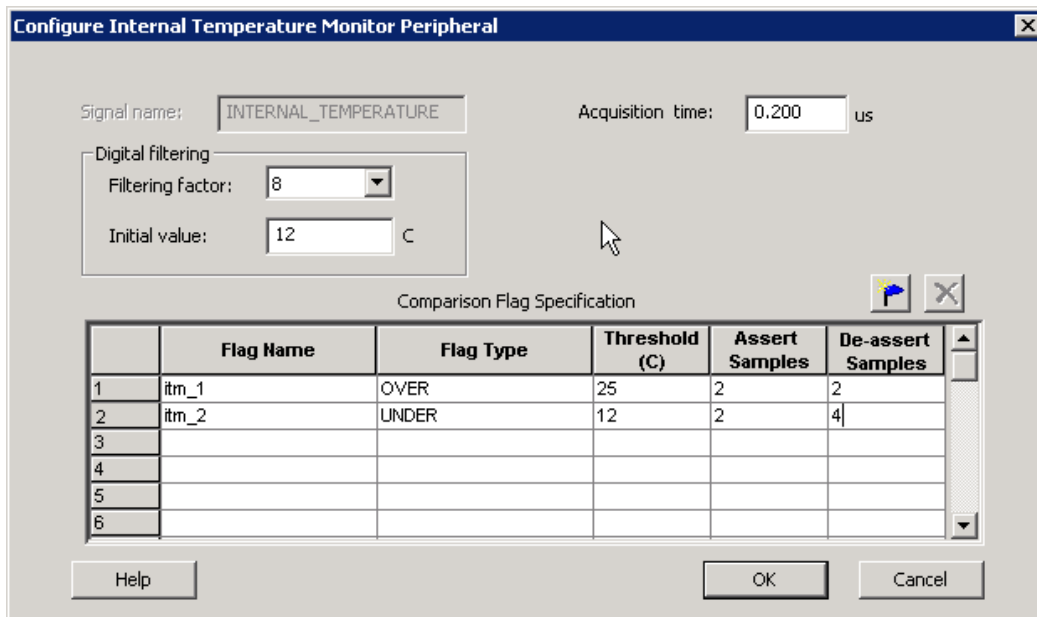


Figure 31 - Internal Temperature Monitor Dialog Box

## Internal Voltage Monitor

The internal voltage channel measures the internal 1.5V power supply. The Maximum Input Voltage for this peripheral is fixed (as shown in the figure below). The peripheral is mapped to logic channel 0 and does not have an associated physical pin.

See the [Configuring Current, Voltage, and Temp peripherals](#) for information on the Digital filtering factor, Acquisition time, and Comparison Flag Specifications.



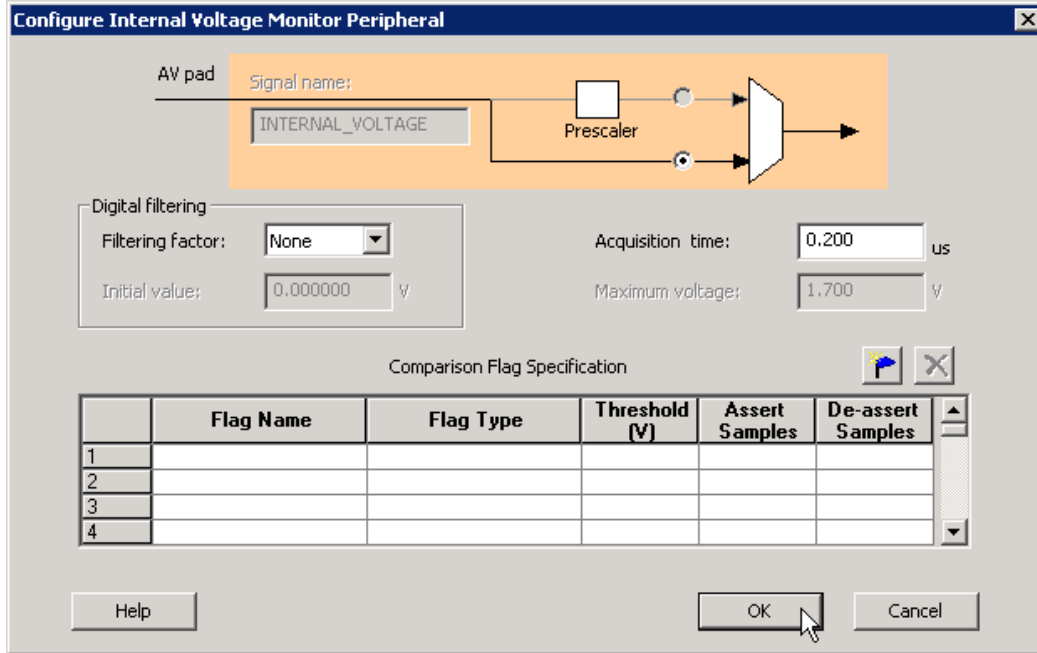


Figure 32 · Internal Voltage Monitor Dialog Box

## Real Time Counter

The on-chip crystal oscillator circuit works with an off-chip crystal to generate a high-precision clock. It has an accuracy of 100 ppm (0.01%) and is capable of providing system clocks for Fusion peripherals and the other system clock networks, both on- and off-chip. When combined with the on-chip CCC/PLL blocks, a wide range of clock frequencies can be created to support various design requirements.

The Real Time Counter inside the Analog Block has the following features:

- The MATCH signal on the output of the system asserts when the value in the counter matches the value specified in the match register. Also, there is an optional output RTCPSMMATCH that is triggered on match. The RTCPSMMATCH signal must be connected to the RTCPSM macro so that the Voltage Regulator activates when the Match signal is asserted.
- If you use the RTC, RTCCLK must be driven by the External Crystal Oscillator driving the Fusion device and the mode of the Crystal oscillator must be controlled by the RTC.
- Displays two different views: Alarm or Custom view. The alarm time specified in the RTC dialog relates to the deassertion of one RTCMATCH signal to the reassertion of the signal.

See [Designing with Analog System Builder](#) for more information on connectivity.

The figure below shows the Real Time Counter dialog box.

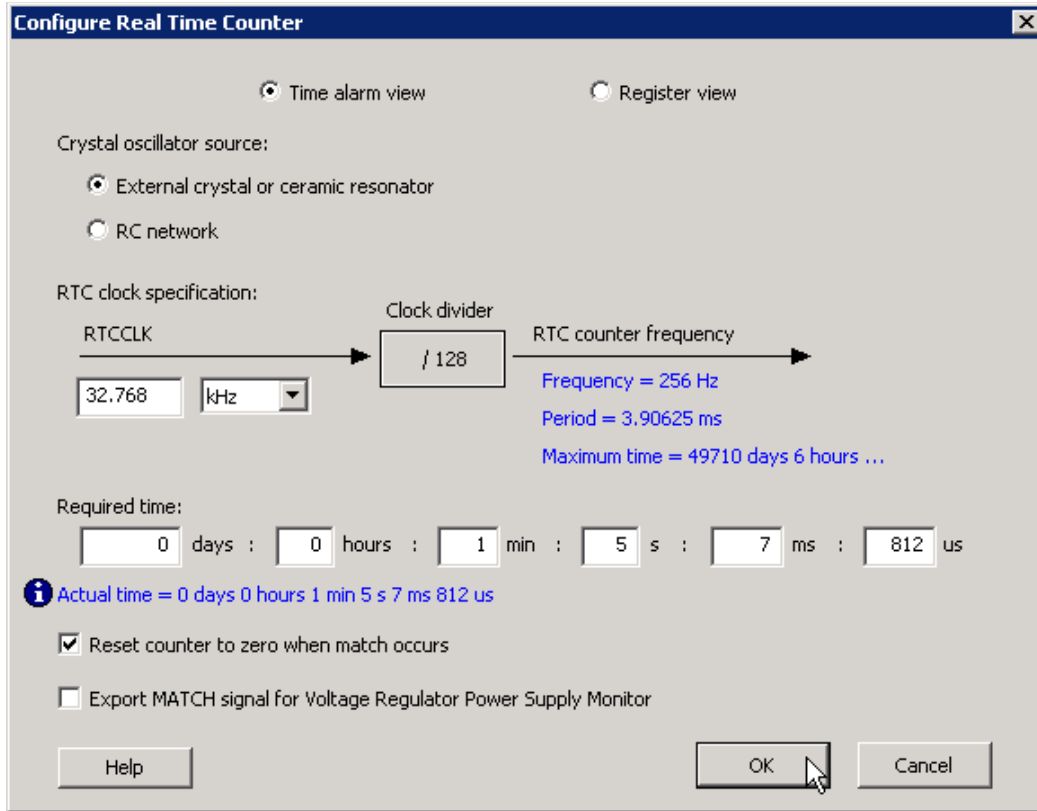


Figure 33 - Real Time Counter Dialog Box

**Time Alarm View** - This view enables you to set time-based configuration of the RTC. The required time can be entered in days, hours, minutes, seconds, milliseconds, and microseconds. Granularity of achievable time values is based on the frequency you set. The actual time used is shown in blue.

The Maximum Time shows the total representable time with the 40-bit register in the RTC and the specified clock frequency.

**Note:** Note: This mode disables the use of the initial value setting of the RTC. It is set automatically to 0.

You can specify a required time, ASB displays an actual time. The actual time reflects what is actually achievable based on the clock frequency of the RTC.

For example, when the clock frequency is 32.768 kHz, each clock tick equals ~3.9ms. If you enter 4ms and 100µs for a required time, ASB approximates using the next achievable time. In this example, 7.8ms (shown in the actual time field).

**Register View** - This view allows you to configure the RTC in its direct hardware representation. Also, this mode enables you to configure the initial value setting.

**Initial Value** - You may specify a different counter start value. The default is zero. This also is a 40 binary or a 10 bit hex value.

**Match with Register Value** - The MATCH signal asserts when the counter is equal to this register value (40-bit binary or 10-bit hex).

#### Crystal Oscillator source

External crystal or ceramic oscillator – Oscillator is configured to work with an external crystal or ceramic resonator.

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

#### RTC clock specification

RTCCLK – The frequency of the CLKOUT from the crystal oscillator

Based on this frequency and the crystal oscillator mode selection, ASB automatically configures the mode, as shown in the table below:

Mode	Recommended Capacitor	Frequency Range (in MHz)
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

The RTCLK is divided internally by 128 to generate the frequency at which the real time counter operates. The frequency and period are shown in blue in the dialog box.

**Reset count to zero when match occurs-** Resets the counter to zero once the match occurs. This can be disabled for an application that measures elapsed time.

**Export Match signal for Voltage Regulator Power Supply Monitor** - Asserts the RTCPSMMATCH to activate the Voltage Regulator Power Supply Monitor (VRPSM).

If you wish to update the match and counter registers of the RTC dynamically, you must use [ASB Advanced Options](#) and select the [ACM Bus](#) checkbox.

## RTC Signals

All signals are active high.

Name	Type	Direction
RTCCLK	INPUT	RTC Clock; must be driven by the XTLOUT of the XTLOSC macro
RTCXTLSEL	OUTPUT	Crystal Oscillator select; must be connected to the XTSEL on the XTLOSC macro
RTCXTLMODE[1:0]	OUTPUT	Crystal Oscillator mode select; must be connected to RTCXTLMODE on the XTLOSC macro
RTCMATCH	OUTPUT	Match signal when match value reached
RTCPSMMATCH	OUTPUT	Match signal when match value reached; must be connected to RTCMATCH of VRPSM macro. Exposed only if "Export Match signal for Voltage Regulator Power Supply Monitor" is enabled.

The following timing diagram shows accessing the ACM for the RTC values. In the diagram, the RTC counter is disabled. Then, the RTC match register is read and finally the match register is overwritten with a new value. The RTC is then re-enabled. Refer to the [Fusion Datasheet](#) - Real Time Counter section for more information.

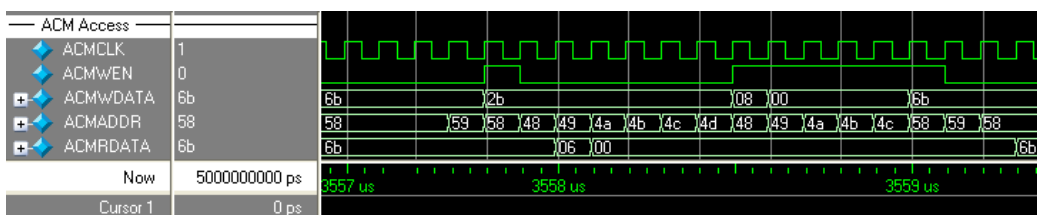


Figure 34 - ASB RTC Access

## Temperature Monitor

When used in conjunction with an external bipolar transistor, the Temperature Monitor is designed to measure temperature of an external location. A temperature monitor circuit can be very sensitive to system noise.

See the [Configuring Current, Voltage, and Temp peripherals](#) for information on the Digital filtering factor, Acquisition time, and Comparison Flag Specifications.

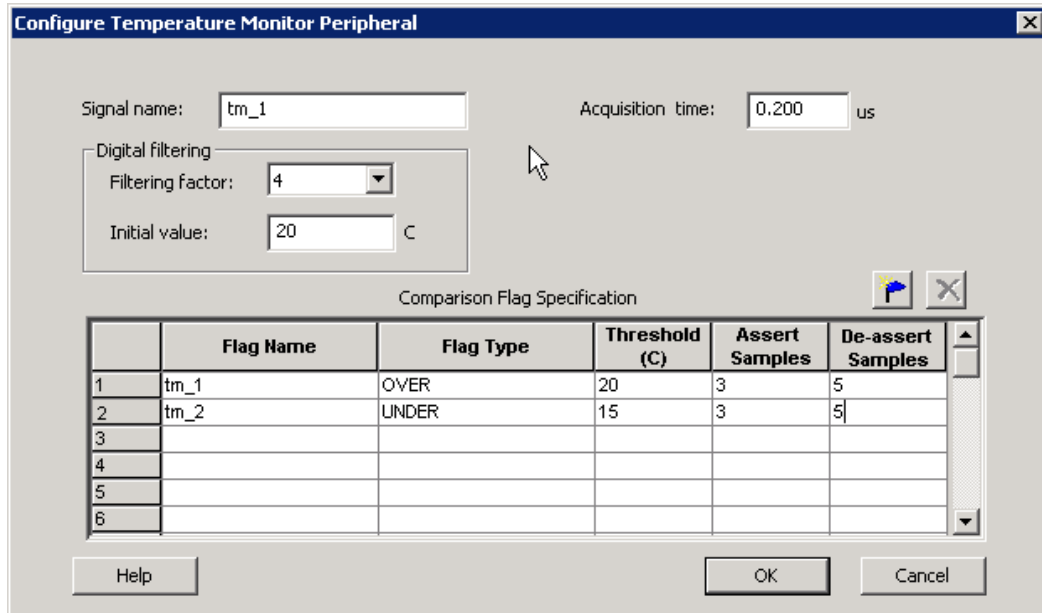


Figure 35 ·

Figure 36 · Configure Temperature Monitor Peripheral Dialog Box

## Voltage Monitor

The Voltage Monitor contains a 2-channel analog multiplexer that allows an incoming analog signal to be routed directly to the analog-to-digital converter, or allows the signal to be routed to a prescaler circuit before being sent to the ADC.

See the [Configuring Current, Voltage, and Temp peripherals](#) for information on the Digital filtering factor, Acquisition time, and Comparison Flag Specifications.

### Use Prescaler or Direct Analog

Each Analog channel has an associated prescaler circuit which can prescale the input signal to a appropriate value supported by the ADC. Using the prescaler circuit imposes certain requirements on the acquisition time of the peripheral. The two radio buttons in front of the multiplexer determine if the Voltage monitor uses prescaling or not.

If the input value is less than the ADC reference voltage and high accuracy is critical for the application, choose the direct analog path. The prescaler circuit could introduce potential gain errors or offset errors. If resolution is more important than accuracy for input voltage ranges lower than the ADC reference voltage, choose the prescaler path.

The prescaler logic of the AB has a settling time of 10 $\mu$ s (max). It is an application-dependent setting and must be accounted for by you via the acquisition and hold time for each channel. A recommended default value is inserted by ASB when configuring a new Voltage Monitor but it may be reduced with the possible reduction in sampling accuracy. See the [Fusion datasheet](#) for information.

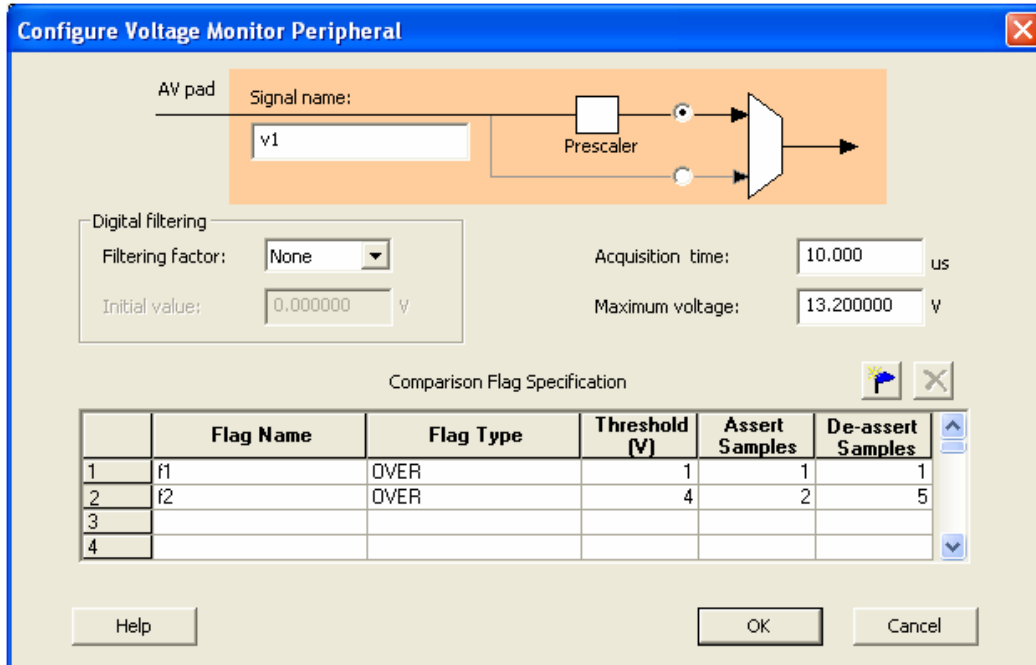


Figure 37 · Configure Voltage Monitor Dialog Box

### Maximum Voltage

The maximum anticipated voltage measured by this Voltage Monitor peripheral pad. The range is -10.5V to +16V (the voltage range is NOT bipolar). The ADC is capable of measuring a voltage range of 0 - Vref. For the Internal voltage reference, this value is 2.56V. ASB automatically configures the prescaler in the AB Analog Block for this peripheral to maximize the available voltage range.

**Note:** Note: Setting the Maximum Voltage to a negative value means that the voltage values driving this signal must be negative. The flag threshold evaluation for negative voltages is on a mathematical scale.

For example, if an OVER -3V flag was configured, the flag would assert whenever the input voltage is -1, -2, up to the threshold value. The flag would deassert when the value is -4, -5, etc.

## Configuring Current, Differential Voltage, Temperature, and Voltage peripherals

The Current, Differential Voltage, Temperature, and Voltage peripherals are all configured the same way. Also, the effects of averaging are the same for all peripherals in the Analog System Builder.

Minor variations, such as [Maximum Voltage](#) in the [Voltage monitor](#), are explained in the help topic for that peripheral.

Signal name is the name of the signal as you want it to appear in the main Analog System Builder dialog box.

### Digital filtering

Once the ADC finishes converting the Analog Signal to a digital value, it filters (averages) the resulting digital output. Digital filtering is a single-pole low-pass filter built in soft gates; you can use it to improve the signal-to-noise ratio. If the ADC input data is very erratic, the filtering will smooth out the input and reduce the noise.

The filtered value is calculated using the following equation:

$$\text{Filtering\_result}_n = \text{filtering\_result}_{n-1} + (\text{ADC\_Result}_n / \text{filtering\_factor}) - (\text{filtering\_result}_{n-1} / \text{filtering\_factor})$$

If the Digital filtering factor is set to 1 it is ignored.

In some cases, where the inputs are very low frequency, and the electrical environment is not very noisy, it may be possible to proceed without any special filtering of input analog signals. However, in most

applications it is desirable to at least implement a simple post-conversion digital filter inside the FPGA by oversampling and averaging several results to reduce the effects of random noise in the conversion signal path and improve overall accuracy. This simple averaging is automatically handled in the software by setting the Digital Filtering factor in the Analog System Builder (ASB) to specify how many samples are averaged (When the factor = F, F samples are averaged together.)

For situations where greater accuracy is required, an external analog filter may be needed to eliminate non-random and out-of-band noise sources. If an analog filter is not used to restrict the input signal content to the band-of-interest, any out-of-band signals or noise will be aliased into the conversion result as random in-band noise.

Some applications (for example, those that require frequency detection) may need both external analog filtering to limit out-of-band effects, and more sophisticated digital processing such as a multi-tap Finite Impulse Response (FIR) filter. A wide variety of digital filtering methods are available through the FPGA gates available in a Fusion Device.

**Initial Filtering Value** - The initial filtering value enables you to specify the starting value for the averaging function (Filtering Result[0]). This enables you to 'seed' your filtering function so that there are no erroneous values produced during the beginning of operation.

If you do not use an initial filtering value, the filtering function always starts with FilteringResult[0] = 0, thereby skewing the results towards 0 during the first range of samples.

**Range** - The Initial value range for the digital filter is identical to the [threshold range](#) for the peripheral.

The system instantiates the logic required to perform averaging as soon as there is at least one channel in the system that requires averaging. There is no extra logic penalty for averaging the other channels of the system. See the figures below for a graphical representation of the effect of digital filtering on a signal.

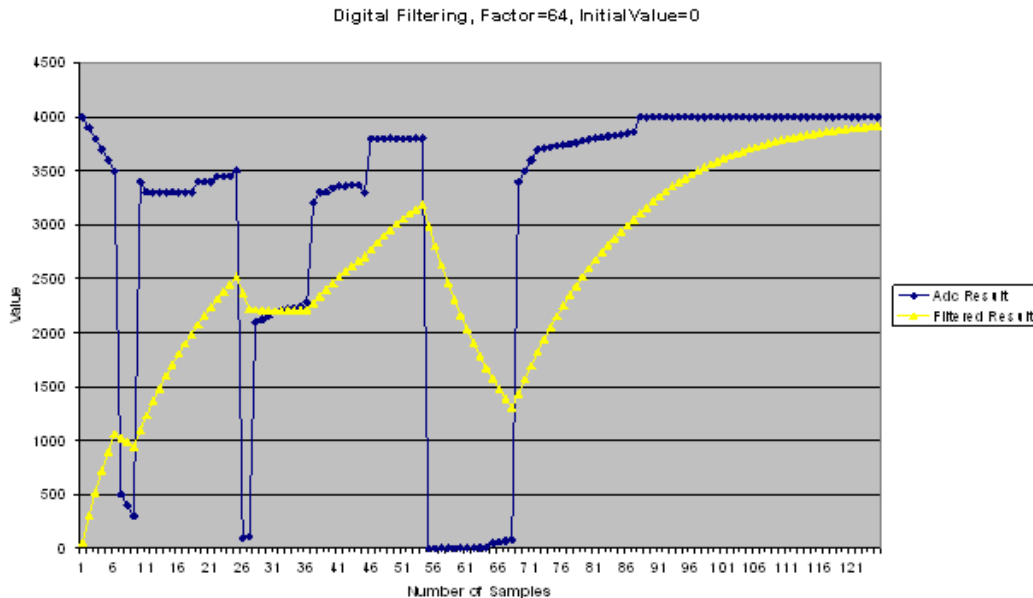


Figure 38 · Effect of Averaging on Voltage, Initial Digital Filtering Value = 0

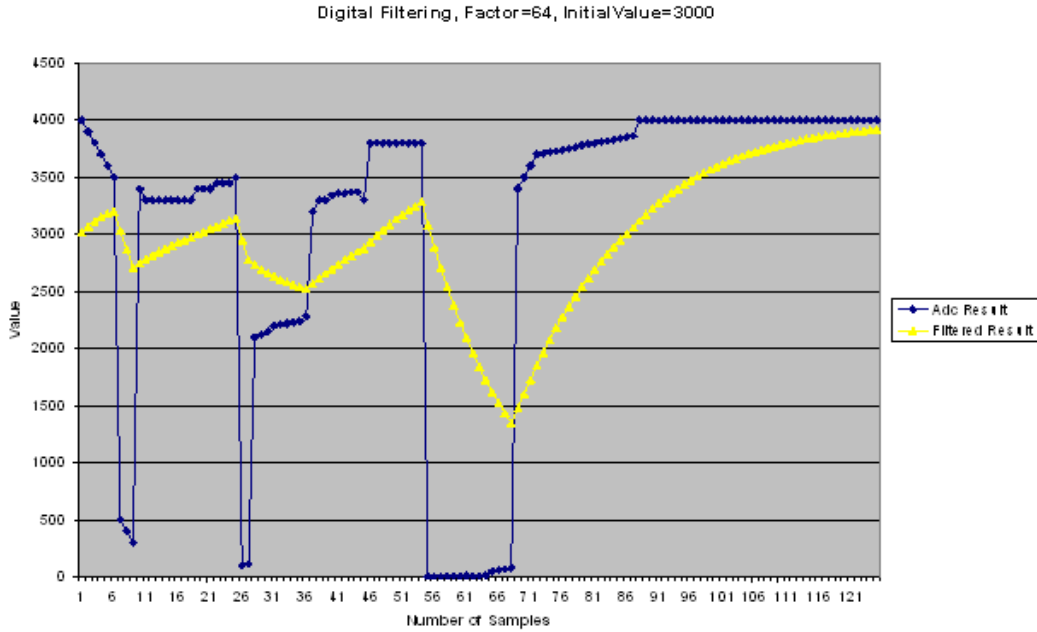


Figure 39 - Effect of Averaging on Voltage, Initial Digital Filtering Value = 3000

#### Acquisition Time

The required settling/sampling time for this channel. It is the amount of time the Sample and Hold circuit in the ADC charges the capacitor with the input analog signal. The characteristics of your system and monitoring requirements will determine this value. Note also that this value has a direct correlation to the achievable sampling rate of the system.

The [prescaler logic](#) of the AB has a settling time of 10 $\mu$ s max. It is an application-dependent setting and must be accounted for by you via the acquisition and hold time for each channel. A recommended default value is inserted by ASB when configuring a new Voltage Monitor but it may be reduced, with a possible reduction in sampling accuracy. See the [Fusion datasheet](#) for information.

ASB evaluates the required acquisition times for all peripherals and the system frequency to compute the maximum possible ADC clock frequency. Only certain divider factors exist to create the ADC Clock; because of this and peripheral acquisition times, certain system frequencies result in a faster ADC Clock.

#### Signal Polarity - Current and Differential Voltage Peripherals ONLY

Use this option to change polarity. The associated voltage monitor must be configured with the same polarity.

During simulation, a negatively configured peripheral must be driven by negative voltage values.

#### Comparison Flag Specification

Once the software calculates the average, you can further process the result by comparing the result to a given threshold and deciding under what conditions to assert the comparison flags.

Select a flag and click the **Delete** button to delete it.

Click the **Add Flag** button to add more flags to the system.

**Flag Name** - This is the name of the flag. This will appear as the output port name in the final output. It will be prefixed with the signal name with which it is associated, to group the input and outputs together.

**Flag Type** - You can choose to assert the flag when the signal is either under a given threshold or over a given threshold.

**Threshold** - Threshold value. The figure below shows the effect of the threshold on a given signal.

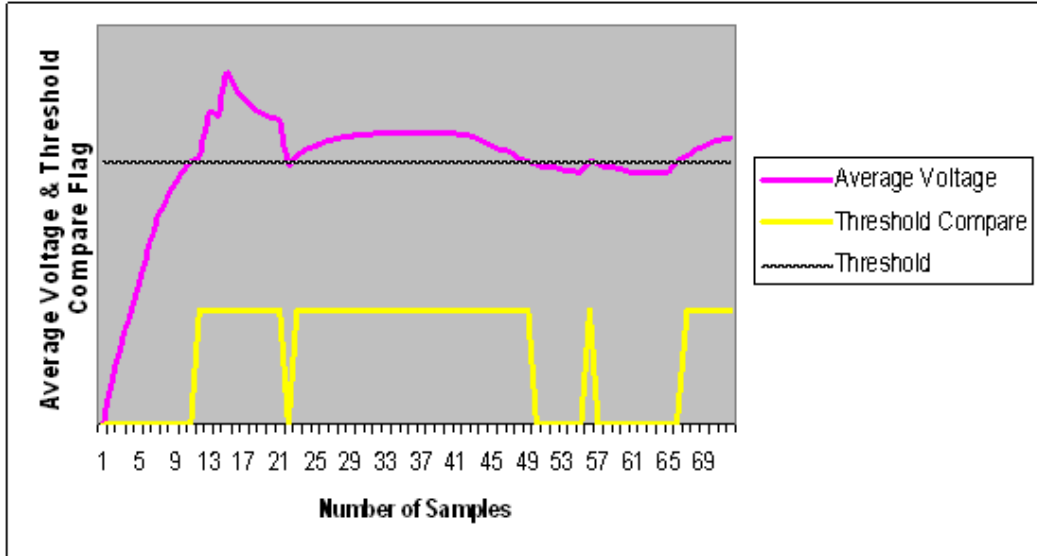


Figure 40 .

Figure 41 . Threshold Comparison

**Assert Samples** - The number of consecutive samples on this channel that reach or exceed the threshold for the flag to assert. This can be a glitch removal feature. If it is set to 1, the final flag is identical to the comparison result.

For example, if your Assert Sample value is 5 and the threshold is set at 3.0V, the channel must reach or exceed 3.0V five times in a row on this channel for the flag to assert. If your voltage values are less than 3.0V, the flag will not assert. The figure below shows the effect of glitch removal on a given signal.

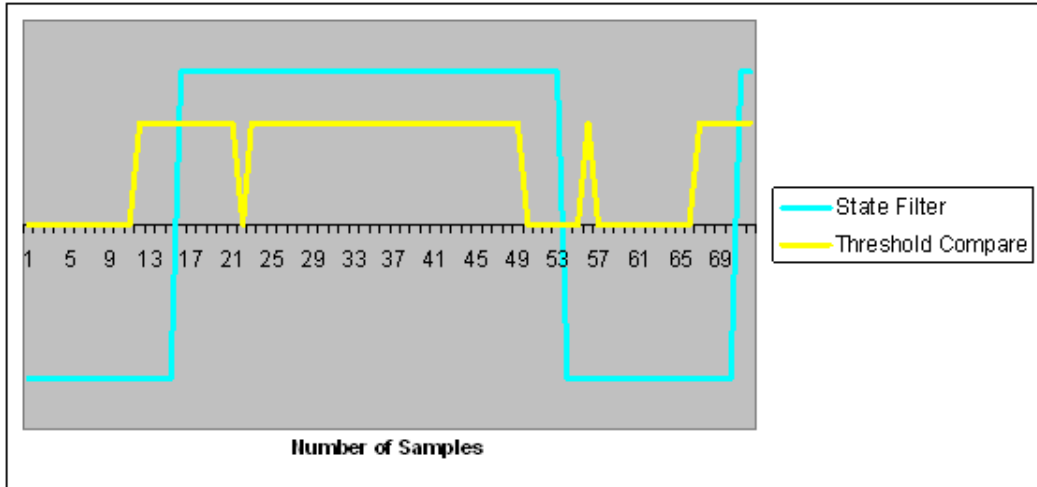


Figure 42 .

Figure 43 . Glitch Removal

**De-assert Samples** - The number of consecutive samples of this channel that are not above the threshold required for the flag to de-assert once it has been asserted. This is a glitch removal feature. If this value is set to 1, the final flag is identical to the comparison result.

For example, if your de-assert Sample value is 10 and the threshold is set at 3.3V, you must have 10 consecutive samples that go below 3.3V in order for the flag to de-assert

## Sampling Rate in Analog System Builder

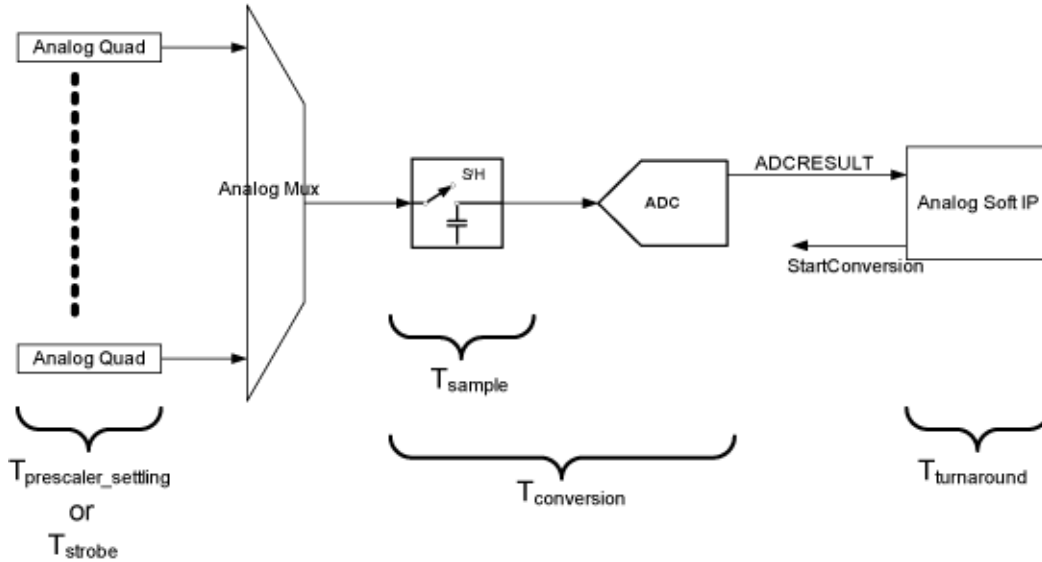
For ADC's that perform one sample per conversion, the throughput rate is also referred to as the sampling rate – this is the case for Fusion. Sampling Rate or frequency is the rate at which the ADC acquires or samples the analog input and converts it to digital data. It is specified as samples per second ( S/s ) or Hertz ( Hz ).



The sampling rate is typically the inverse of the ADC Conversion Time. For example, an ADC that takes 10 microseconds to acquire and convert an analog signal to a digital value will be able to generate about 100,000 samples per second. In the case where only a single channel is being sampled, the channel's sampling rate is equal to the total system sampling rate.

However, in the case where the sampling sequence contains multiple channels and where a channel may be sampled multiple times in relation to another channel, the sampling rate computation becomes more involved.

The diagram below shows a silicon view of the analog signal path.



$T_{\text{prescaler\_settling}}$ : Applicable only in Voltage Monitor peripheral when prescaler circuit is used

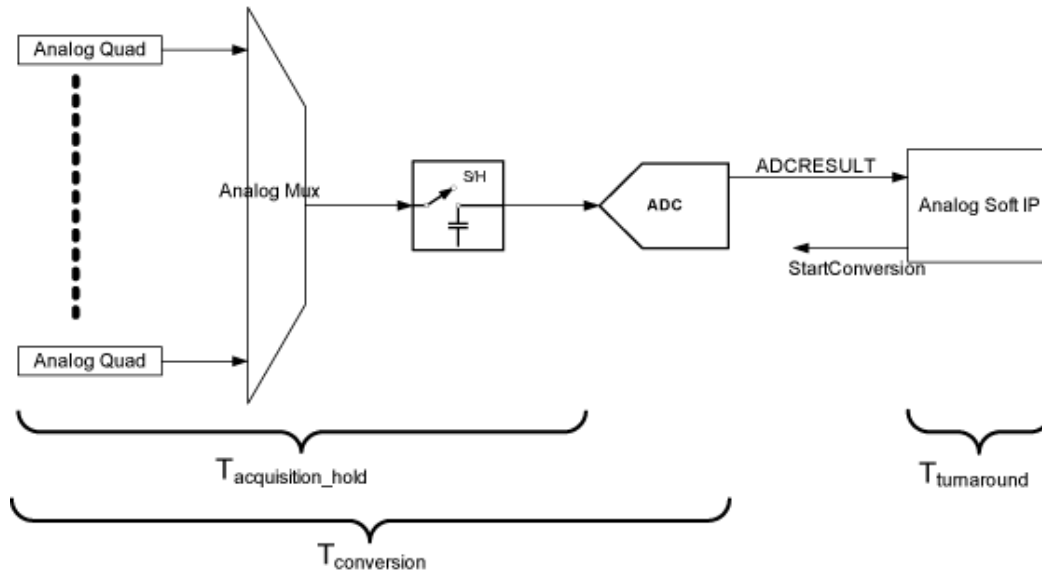
$T_{\text{strobe}}$ : Applicable only in Current, Temperature, or Differential peripherals. See [Sequencing in Fusion](#) for more information.

$T_{\text{sample}}$ : Time for sample and hold circuit to sample the analog input voltage into the input capacitor

$T_{\text{conversion}}$ : Conversion time of ADC

$T_{\text{turnaround}}$ : How fast a client of the ADC can process data and give another start conversion signal. In this case this falls onto the ASSC IP.

To simplify the user interface experience, ASB utilizes a simpler view of the signal path, shown below:



Notice that the prescaler, strobe, and sample time have been condensed into the  $T_{\text{acquisition\_hold}}$  time. This is the value that is specified by you during configuration of the peripheral. Legal ranges are enforced by ASB based upon requirements of the silicon.

The following sections describe the sampling rate computation for ASB. ASB enables you to specify the minimum sampling time and minimum sampling rate. ASB then reports the total system sampling rate and the actual sampling rate of each channel.

## Conversion Time Calculation

The conversion time is a sum of:

$$T_{\text{conv}} = T_{\text{sync\_read}} + t_{\text{sample}} + t_{\text{distrib}} + t_{\text{cal}} + t_{\text{sync\_write}}$$

$$T_{\text{sync\_read}} = T_{\text{sync\_write}} = \text{synchronization time} = \text{sys\_clk}$$

$$T_{\text{acquisition\_hold}} = T_{\text{sample}} + (T_{\text{prescaler\_settling}} \text{ Or } T_{\text{strobe}}) = (2 + \text{stc}) * \text{adc\_clk}$$

The stc is a factor of the user's acquisition and hold time

$$T_{\text{distrib}} = \text{charge distribution time} = \text{adc\_resolution} * \text{adc\_clock}$$

$$T_{\text{cal}} = \text{calibration time} = 2 * \text{adc\_clk}$$

## ADC Clock Calculation

The ADC Clock period is calculated by:

$$(4 * (1 + \text{clock divider setting})) / \text{System Clock Period}$$

The ADC Clock period has a maximum possible frequency of 10Mhz.

ASB automatically computes values for the sample time control (STC), clock divider setting (TVC), and ADC Clock Period based on your sample time requirement. Only certain divider factors exist to create the ADC Clock; because of the divider factors and peripherals' acquisition times, certain system frequencies result in a faster ADC Clock.

The goal of ASB is to meet the specified acquisition and hold time requirements with the highest possible ADC Clock frequency, which implies a low TVC value and high STC value.

See [Designing with Analog System Builder](#) for a discussion of the clocks involved in an Analog System design.

## Sampling Rate Calculation for Fusion

The inverse of the conversion time is the sampling rate for that channel. However, the sampling rate reported by ASB includes the time that the ASSC can process the data and assert another start conversion

signal. This time is referred to as 'turnaround time'. With no wait states, the ASSC takes 10 system clock cycles for turnaround time.

Therefore, in the case of Fusion, the sampling rate of a single channel is:

$$1 / [ ( \text{ASSC turnaround time} ) + ( \text{channel conversion time} ) ]$$

ASSC Turnaround Time - The number of flags for a channel may increase the turnaround time of the ASSC. Increasing the number of flags will potentially reduce your max sampling rate.

## General Formula for Sampling Rate

The general formula for system and per-channel sampling rate is as follows:

Total Sampling Rate = Total # of Samples / Total Conversion Time of all Samples

Channel Sampling Rate =

$$\left( \frac{\text{Total Number of Samples for Channel}}{\text{Total Number of all Samples}} \right) * \text{Total Sampling Rate}$$

The method for arriving at this formula is explained in the examples below.

### Example: Equal Weight and Equal Conversion Time

All Channels have a conversion time of 2µs as shown in the figure below.

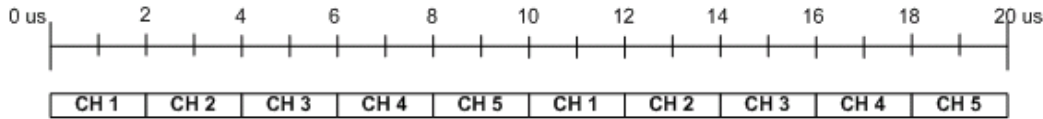


Figure 44 · Equal Weight and Equal Conversion Time

In this case we have 10 samples which take a total of 20µs.

Thus, our total system sampling rate is:  $10 / 20\mu\text{s} = 500 \text{ kS} / \text{s}$

Channel1 Sampling Rate:  $2\mu\text{s} / 10\mu\text{s} = .20 * 500 \text{ kS} / \text{s} = 100 \text{ kS} / \text{s}$

Channel2 Sampling Rate:  $2\mu\text{s} / 10\mu\text{s} = .20 * 500 \text{ kS} / \text{s} = 100 \text{ kS} / \text{s}$

Channel3 Sampling Rate:  $2\mu\text{s} / 10\mu\text{s} = .20 * 500 \text{ kS} / \text{s} = 100 \text{ kS} / \text{s}$

Channel4 Sampling Rate:  $2\mu\text{s} / 10\mu\text{s} = .20 * 500 \text{ kS} / \text{s} = 100 \text{ kS} / \text{s}$

Channel5 Sampling Rate:  $2\mu\text{s} / 10\mu\text{s} = .20 * 500 \text{ kS} / \text{s} = 100 \text{ kS} / \text{s}$

### Example: Unequal Weight and Equal Conversion Time

In this example, the channels are not equally weighted in the sampling sequence, as shown in the figure below.

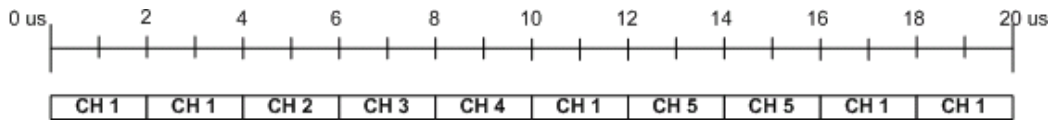


Figure 45 · Unequal Weight and Equal Conversion Time

In this case we have 10 samples that take a total of 20µs, giving us a total system sampling rate of 500 kS/s (as above). However, the individual channel sampling rates are different.

Channel1 Sampling Rate:  $5\mu\text{s} / 10\mu\text{s} = .5 * 500 \text{ kS} / \text{s} = 250 \text{ kS} / \text{s}$

Channel2 Sampling Rate:  $1\mu\text{s} / 10\mu\text{s} = .1 * 500 \text{ kS} / \text{s} = 50 \text{ kS} / \text{s}$

Channel3 Sampling Rate:  $5\mu\text{s} / 10\mu\text{s} = .5 * 500 \text{ kS} / \text{s} = 50 \text{ kS} / \text{s}$

Channel4 Sampling Rate:  $5\mu\text{s} / 10\mu\text{s} = .5 * 500 \text{ kS} / \text{s} = 50 \text{ kS} / \text{s}$

Channel5 Sampling Rate:  $2\mu\text{s} / 10\mu\text{s} = .20 * 500 \text{ kS} / \text{s} = 100 \text{ kS} / \text{s}$

### Example: Unequal Weight and Unequal Conversion Time

In this example, channels have different conversion times and are not equally weighted in the sampling sequence, as shown in the figure below.

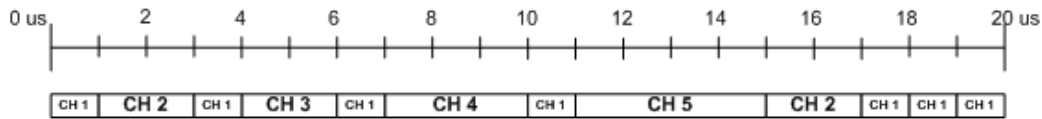


Figure 46 · Unequal Weight and Unequal Conversion Time

In this case we have 12 samples in 20µs, giving us a total system sampling rate of 600 kS/s.

Channel1 Sampling Rate:  $7 / 12 = .583 * 600 \text{ kS /s} = 349 \text{ kS /s}$

Channel2 Sampling Rate:  $2 / 12 = .166 * 600 \text{ kS /s} = 99.6 \text{ kS /s}$

Channel3 Sampling Rate:  $1 / 12 = .083 * 600 \text{ kS /s} = 49.8 \text{ kS /s}$

Channel4 Sampling Rate:  $1 / 12 = .083 * 600 \text{ kS /s} = 49.8 \text{ kS /s}$

Channel5 Sampling Rate:  $1 / 12 = .083 * 600 \text{ kS /s} = 49.8 \text{ kS /s}$

### Wait States and the Analog System Controller

The Analog System Controller automatically inserts wait states to prevent collisions during processing. The wait states are inserted to ensure this condition:

Current conversion time + ASSC processing time  $\geq$  previous sample's (SMEV + SMTR) processing time

Thus a wait state is calculated as:

(Current conversion time + ASSC processing time) – previous sample's (SMEV + SMTR) processing time

This extra time is inserted into the sample rate calculation, thereby decreasing the total sample rate; when wait states are inserted, the calculated sample rate is only an approximation. For example:

Single channel, continuous loop:

System clock = 40 MHz

ADC clock = 10 MHz

Acquisition time = 0.2µs

Resolution = 8-bit

No Flags

$T_{conv} = t_{sync\_read} + t_{sample} + t_{distrib} + t_{cal} + t_{sync\_write}$

$(25 \text{ ns}) + (2 * 100 \text{ ns}) + (8 * 100 \text{ ns}) + (2 * 100 \text{ ns}) + (2 * 25 \text{ ns}) = 1.25 \mu\text{s}$

Add turnaround time:

$1.25 \mu\text{s} + 0.25 \mu\text{s} = 1.5 \mu\text{s}$

Sampling Rate =  $1 / 1.5 \mu\text{s} = 666 \text{ ksps}$

## Sequencing in Fusion

The ADC has a strobe signal per temperature or current channel that must be asserted to initiate the conversion process. This strobe signal has strict timing pulse-width requirements; the strobe signal must stay on and then off for certain periods of time.

The requirement is that a strobe signal must stay low for a minimum of 5µs after a high-to-low transition.

Also, there is a high time requirement:

- 5µs high for current strobe
- 10µs high for temperature strobe

For temperature channels, the sample sequencer automatically inserts a 5µs delay before the ADC is started. During this time the strobe signal for that particular channel is asserted. This is to ensure that the temperature monitor block inside the ADC has settled to the correct value. After this 5µs has elapsed, the ADC sampling occurs. The length of this sampling period is the acquisition time specified in the peripheral configuration dialog.

**Note:** Note: If the Analog System is configured in the ADC only mode, these strobe requirements need to be handled manually.

After the sampling period is completed, then ADC conversion begins.

Because of the automatic insertion of the  $5\mu\text{s}$  delay, the minimum acquisition time for temperature is only required to be  $5\mu\text{s}$ . Using the minimum acquisition time + the  $5\mu\text{s}$  delay satisfies the minimum strobe requirement for temperature monitoring, as shown in the figure below.

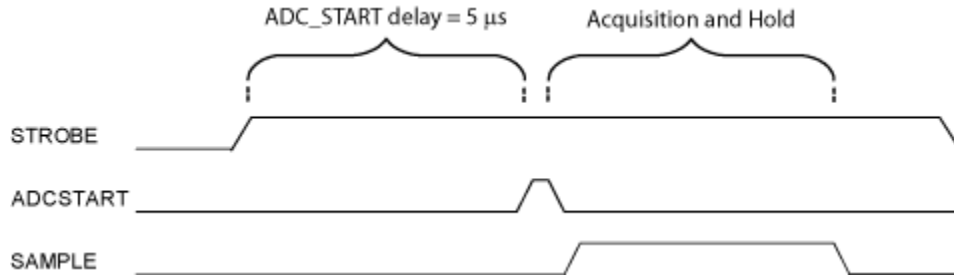


Figure 47 · ADC\_START delay and Acquisition and HOLD Time

The ASSC generates these strobe signals; it has no concept of elapsed time, past samples, or future samples. A strobe pulse is activated for the entire ASSC processing time for a particular channel, and deactivated as soon as it moves to another channel. Thus, this requirement must be accounted for by you and/or ASB during the sequencing.

You must satisfy the following conditions to meet the requirement:

- A channel that requires a strobe cannot be sampled again until  $5\mu\text{s}$  has elapsed
- A channel that requires a strobe must have a conversion time + ASSC processing time  $\geq 5\mu\text{s}$

The second condition is automatically handled because you enter a MINIMUM required sampling time. ASB can then ensure a minimum  $5\mu\text{s}$  conversion time for temperature and current channels by adjusting the STC and ADC clock periods.

## Welcome to FlashROM

FlashROM memory provides the security of stored data in addition to a 128-bit AES decryption core. You can read, modify, and write to the FlashROM using the JTAG interface; however, you can only read it from the FPGA core.

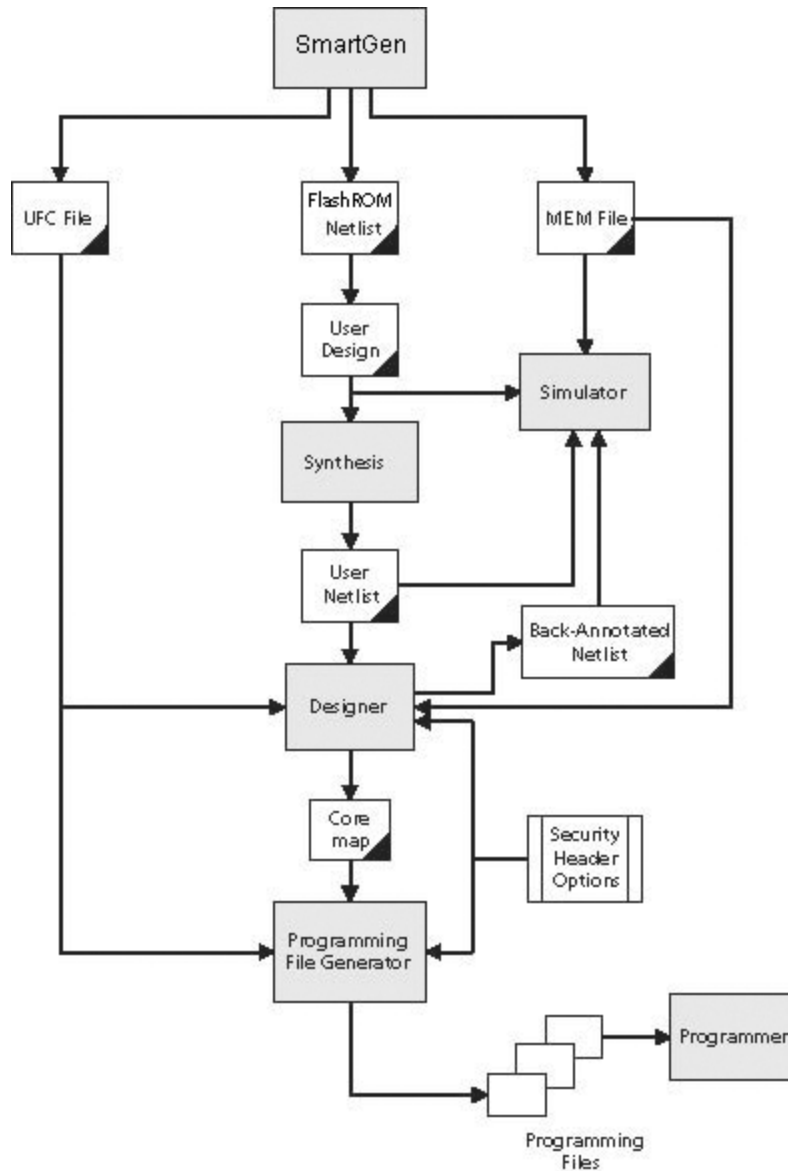


Figure 48 · FlashROM Flow

FlashROM is available from the Catalog the Libero SoC. Add a FlashROM core from the Catalog to your SmartDesign in the Libero SoC Project Manager.

Adding a FlashROM core opens a special [FlashROM core generator](#) that enables you to configure the FlashROM functionality.

**Note:** Note: FlashROM is available only for IGLOO and ProASIC3 devices.

IGLOO and ProASIC3 devices have a flexible programming option. The FlashROM and the FPGA core fabric can be programmed independently of each other, allowing the FlashROM to be updated without changing the FPGA core fabric. The following are just a few examples of possible applications for the FlashROM feature:

- Internet protocol (IP) addressing (wireless or fixed)
- System-calibration settings
- Device serialization and/or inventory control
- Subscription-based business models (e.g. set-top boxes)
- Secure key storage
- Asset management tracking

- Date stamping
- Version management

The FlashROM is programmed using the standard IEEE1532 JTAG programming interface. Pages can be individually programmed (erased and written) and on-chip AES decryption can be used selectively to load data securely into the FlashROM (such as application-based security keys stored in the FlashROM for a design). See the [FlashPoint help](#) or user's guide for information on how to program your FlashROM-enabled devices.

The FlashROM can selectively be read back either through the JTAG programming interface or via direct FPGA core addressing. Its contents can only be updated via the JTAG interface. A seven-bit address from the FPGA core defines which of the eight pages (3 MSBs of the address) is being read and which of the 16 bytes in the page (4 LSBs) are being read.

The FlashROM is physically organized as 8x128 bit blocks and logically organized as eight pages by 16 bytes. Only Flash FPGAs contain on-chip nonvolatile memory (NVM); Microsemi's SmartFusion, IGLOO, ProASIC3 and Fusion devices are the only FPGAs to support this feature.

You can assign specific regions of the FlashROM for specific purposes by floorplanning the FlashROM and assigning properties. The content of these regions can be modified during programming time if you assign a modifiable content property to a given region. If you do not want the FlashROM content to be modified, you can fix the content in SmartDesign.

When you generate a new FlashROM file, the generator saves the following files for you to use throughout the design cycle:

- **CXF file** - Contains project info for Libero SoC.
- **Netlist file** - Use this file to instantiate your core, just as you would instantiate any other core in your design
- **UFC file** - User Flash configuration file; it contains all the configuration information regarding the FlashROM data content and is used for programming. You can export a core map file that contains the core programming information and use it along with the UFC file to generate programming files. Designer software supports importing the UFC file and launching the programming file generator to merge the FPGA core map file and the FlashROM programming file.
- **MEM file** - FlashROM specific memory initialization file. The MEM file has 128 rows of eight bits, representing the contents of the FlashROM. FlashROM defaults to 0s for any unspecified locations of the FlashROM memory. This file is used exclusively for simulation.

Use the FlashROM help to:

- Configure FlashROM
- Simulate Pre/Post Synthesis
- Synthesize
- Place-and-Route
- Run Back-Annotation and Timing Simulation
- Specify security settings
- Specify FlashROM content
- Generate a programming file

## Create/Configure FlashROM

The GUI enables you to create and configure memory regions in FlashROM. The regions you create can be of arbitrary widths (up to sixteen). You can specify whether the content in this region is Fixed (meaning what you enter here in the GUI is fixed), or Modifiable if you expect it to change in the future.

To actually specify the data that goes into the memory region you just created, you must specify the Type of the content you are about to enter: Binary, Hexadecimal, Decimal, or Text (Character String). And finally, you must specify the actual data in the Value field.

**Note:** Note: If you use STAPL files be sure that your memory REGION name does not contain illegal characters. The FlashROM Configurator Dialog enables you to specify a REGION Name that contains characters which are not legal in the STAPL File Language format. STAPL identifier names

are limited to 32 characters and must begin with an alphabetic character - not a number or underscore character (\_). Identifier names consist of alphabetic characters, numeric characters and the underscore character - no other characters are allowed. Identifier names are not case sensitive.

The FlashROM can be partitioned into regions and each region can be used for a specific purpose, like serial number storage, version number saving, etc.

Use the FlashROM core generator (in SmartDesign) to create a region within a page, modify the region, and assign properties to that region.

The FlashROM user interface includes the Configuration Grid, a list of existing regions, and the list of Properties. You can assign values to the following properties:

## Content

**Static** - Data entered manually when the core is configured and is not changeable. This option is useful when you have fixed data stored in this region that is required for the operation of the design in the FPGA. Key storage is one example.

**Auto Inc** - Specify a starting number, a maximum number and the size of each step between. The starting value and maximum value can be modified in FlashPoint.

**Read from File** - Data is read from a content file into the selected region. A different content file may be selected in FlashPoint from either Designer or FlashPro.

When Content is specified as **Read from File**, the following file formats are supported: Binary, Hex, Dec, and Text.

Note that when configuring FlashROM in SmartGen there is a limit on the decimal value when using read from file and format type DEC. The limit is 32-bits, so the largest number is 4294967295. Even if you have an 128-bit region, the largest number will be 32 bits.

In each format, the file is a simple list of data based on your content size. For example:

**Binary format:** 0101110011

**Hex format:** 1112222ffffaaabbbb

**Dec format:** 123456789

**Text format:** myexampletext

## State

**Fixed** - Enables you to fix the data so that it cannot be changed during programming time. This option is useful when you have fixed data stored in this region that is required for the operation of the design in the FPGA. Key storage is one example.

**Modifiable** - Select this option when the data in a particular region is expected to be static data (such as a version number, which remains the same for a long duration, but could conceivably change in the future). This option enables you to identify this region so that you need not come back and change the value every time you enter new data.

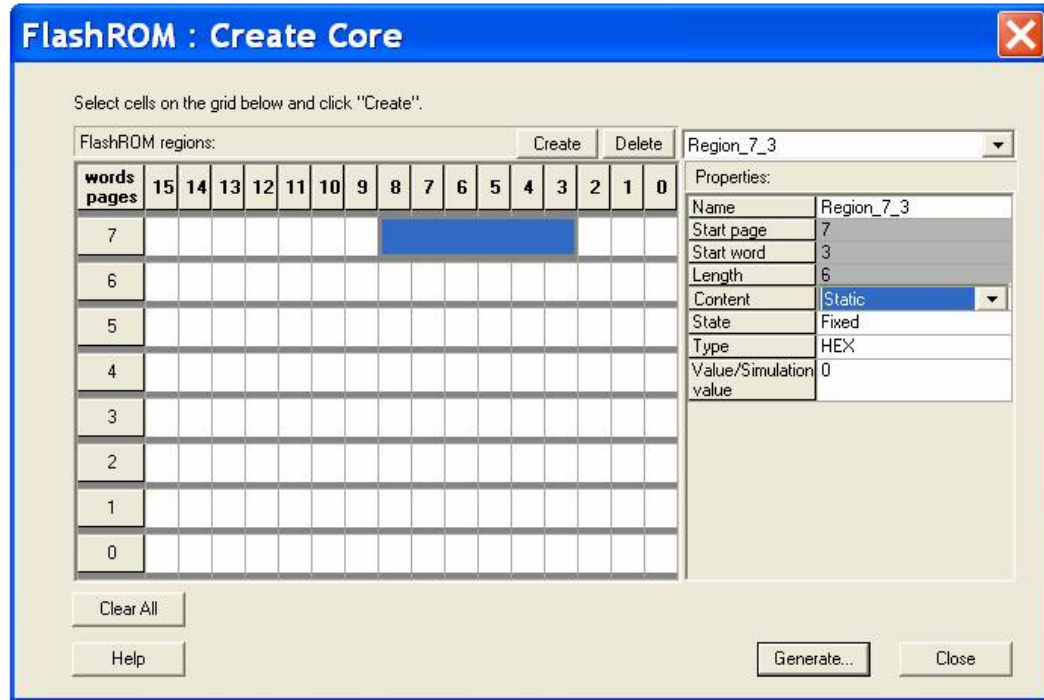
## Type (Format)

Specifies the format that you intend on specifying in the Value/Simulation Value field. For example, if you choose DEC for the type then you can only type a decimal value into the Value/Simulation Value field. Text enables you to enter any character string.

## Value/Simulation value

This is the actual content that you want programmed into that FlashROM region.





FlashROM Core Generator

**To create a new FlashROM:**

1. In the Libero SoC Core Catalog, choose **Memory & Controllers**. Double-click the **FlashROM** core in the Core Catalog to start the core generator.
2. Click and drag the mouse to select words, then click the **Create** button. The core generator displays the new region properties in the **Properties** grid.

You may also right-click a word and choose **Create** from the shortcut menu, or select a word and press the **Insert** key on your keyboard. You can copy and paste regions in FlashROM; to do so, right-click a word and choose **Copy**, then click an empty word, right-click, and choose **Paste**. If the region is not copied, the page does not have enough room. Try another page with more room.

3. Click in the **Properties** grid to modify a region's properties. **Start page**, **Start word**, and **Length** are read-only. The data you enter is verified and stored in the FlashROM as soon as you leave the Properties grid and select another FlashROM region.
4. Click **Generate** to generate a netlist (output format matches the HDL type you specified when you created your project), GEN, CXF, LOG, UFC, and MEM file. The **Generate** button opens the Generate Core dialog box. Specify a name and click **OK**.

**To delete a FlashROM Region:**

1. Click to select a region in the **Regions** window.
2. Click the **Delete** button in the core generator, press the **Delete** key on the keyboard, or right-click and choose **Delete** from the shortcut menu.
3. Click **OK**.

See the [FlashPoint help](#) or user's guide for information on how to program your FlashROM-enabled devices.

## Modify an existing FlashROM configuration

You can modify your existing FlashROM configuration the same way that you modify any configured core. To do so:

1. Open your configured FlashROM core. The FlashROM core opens with all the settings you saved.
2. Modify the values you wish to change and click **Generate** to save your changes. **Generate** opens the **Generate Core** dialog box. Save your new file with the same name if you wish to overwrite the old file.

You cannot edit the configuration of an existing FlashROM GEN file, only the data. If you wish to change the configuration you must generate a new core.

## Simulate Pre/Post Synthesis

FlashROM uses the MEM file for simulation.

The MEM file has 128 rows of eight bits, representing the contents of the FlashROM. FlashROM defaults to 0s for any unspecified locations of the FlashROM memory.

During simulation, employ the MEM file, which contains the memory content, along with the design netlist and testbench. The VITAL and Verilog simulation models accept the generics passed by the netlist, read the MEM file, and perform simulation with the data in the file.

In addition to using the MEM file, you may create a binary file with 128 rows of eight bits and save the file as a MEM file. Microsemi recommends using different names if you plan to generate multiple MEM files. During place-and-route in Designer, the software recognizes the generic property in the netlist and passes the MEM file links through to the output netlist.

## Place-and-Route and FlashROM

There are no special instructions for place-and-route for the FlashROM. Run [Layout](#) in [Designer](#) to place-and-route your design.

## Welcome to the Flash Memory System Builder

The Flash Memory System Builder enables you to configure the entire flash memory block. You can:

- Add analog system initialization data
- Add initialization clients
- Add Fusion RAM clients that require initialization
- Partition non-volatile memory for data access
- Specify the sizes of the partitions
- Specify the memory contents for partitions

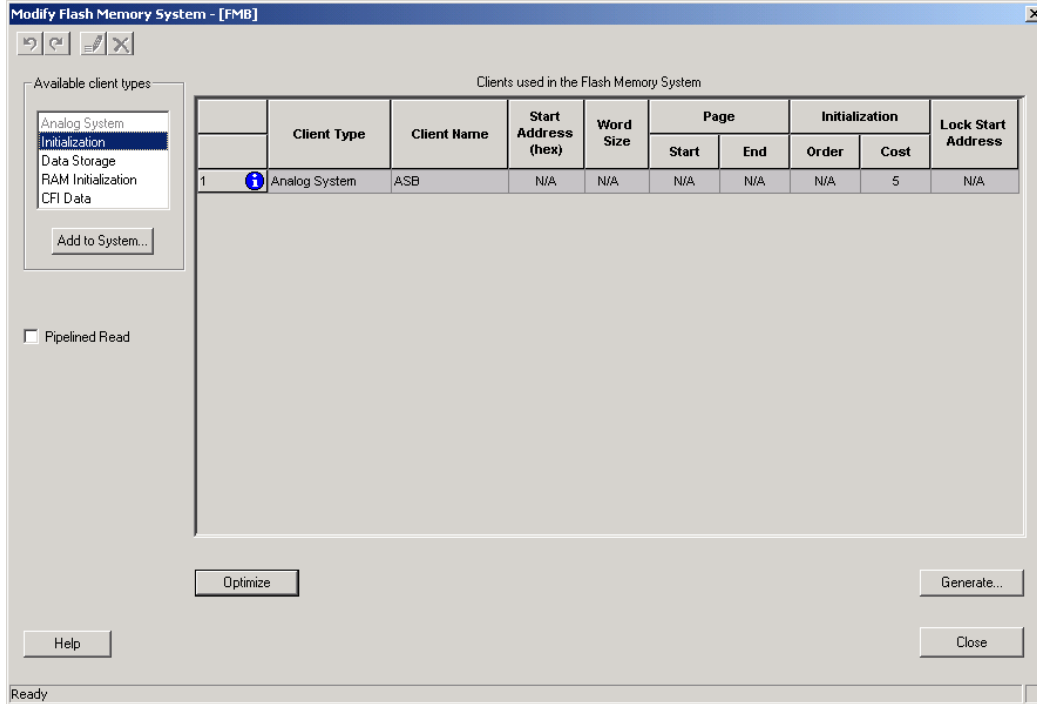


Figure 49 · Flash Memory Block Builder

The Flash Memory Block Builder supports the following clients:

- [Analog System](#)
- [Initialization](#)
- [Data Storage](#)
- [Ram Initialization](#)
- [CFI Data](#)

Each client spans a minimum of one page (128 bytes) and can go up to 2048 pages, based on the number of free pages available. The analog system itself does not take any of the regular pages; it is stored entirely in the reserved pages.

The System Grid in the GUI provides an overview of the system.

**Client Type** - The type of the client that is added to the system.

**Client Name** - The name of the client. It must be unique across the system.

**Start Address** - The decimal or HEX address at which the client starts. It must be on a page boundary. Clients cannot have overlapping start addresses.

**Word Size** - Word size of the client, in bits

**Page Start** - This is the page on which the start address begins or ends. You can modify either the start address or the start page.

**Page End**- This is computed based on the word size and the number of words in the client.

**Initialization Order** - The order in which the clients' required initialization is written with the data from the Flash Memory System. If an analog system is present, it will be initialized first. If any clients must save their data to the Flash Memory System, the save order is the same as the Initialization Order.

**Initialization Cost** - The number of Initialization Enables used by a given client. An Analog System client has an Init Cost of 4. A RAM client has an Init Cost equal to the number of RAM blocks. A regular initialization client has an Init Cost of 1. The data storage client has no Init Cost.

The total init cost must be less than or equal to 64.

**Lock Start Address** - You can specify this option if you do not want the system to change your start address for any reason.

**Optimize** - Click this button to resolve the conflicts on overlapping base addresses for clients. This operation will not modify the base addresses for any clients that have their base addresses locked. It also defragments the memory.

## Analog System Client

You can load the configuration file generated by the [Analog System Builder](#) into the Flash Memory System Builder. Once loaded, all the analog system components can be initialized by the Flash Memory System at start up.

The Add Analog System Client opens with a blank field for the Configuration file. The Modify Analog System Client opens with the field filled in. Click the dropdown menu and select your Analog System core from the list.

**Note:** Note: Initialization of the Analog System must occur at less than 10 Mhz; the Analog Configuration MUX (ACM) inside the Analog Block runs at a maximum of 10 Mhz. The ACM contains the configuration data for the Analog channels. If you initialize at more than 10 MHz, you will overload the MUX.

**Note:** If necessary, a PLL and NGMUX can be used in conjunction to support driving the Analog System at a slow initialization frequency and a fast operating frequency.

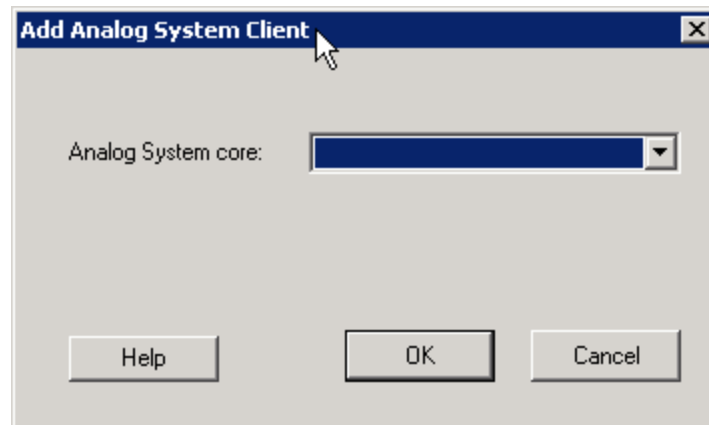


Figure 50 · Add Analog System Client Dialog Box

## Data Storage Client

The Data Storage Client enables you to create a partition in the Flash Memory System and specify the memory content for that partition. You can access the partition directly via the Flash Memory System busses. The Add Data Storage Client dialog box opens with blank fields; the Modify Data Storage Client displays any values you have already set.

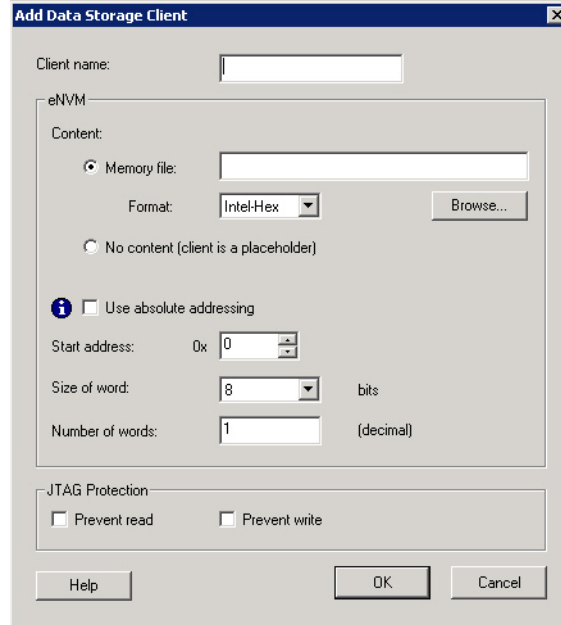


Figure 51 - Add Data Storage Client Dialog Box

**Client name-** Name of the client; the value you enter is attached before the select and enable names to group all the control signals for that client.

## eNVM Content Description

**Content** - Specify the memory content that you want to be loaded into Flash Memory. You may choose one of the two following options:

- **Memory File** - You need to select a file on disk that matches one of the following memory file formats – Intel-Hex, Motorola-S, Actel-S or Actel-Binary.
- **No content** - The client is a place holder. You will be available to load a memory file using FlashPro/FlashPoint.

**Use absolute addressing** - Lets the memory content file dictate where the client is placed in the flash memory block. The addressing in the memory content file for the client becomes absolute to the whole flash memory block. Once you choose the absolute addressing option, the software extracts the smallest address from the memory content file and uses that address as the start address for the client.

**Start Address** - The memory location of the content loaded in the eNVM.

**Size of Word** - Word size, in bits, of the initialized client; can be either 8, 16 or 32.

**Number of words** - Number of words of the client.

## JTAG Protection

Click the checkbox to protect your JTAG from read and write.

## CFI Data

Used to store the query data for CoreCFI. The data is stored in a reserved page location. This client does not take up any of the 2048 pages in the Flash Memory.

Use the dialog box to specify the location of the memory file for CFI query data (as shown in the figure below). This memory file is provided to you when you obtain the CoreCFI IP. The memory file format is Actel Binary. You are not expected to modify this file.

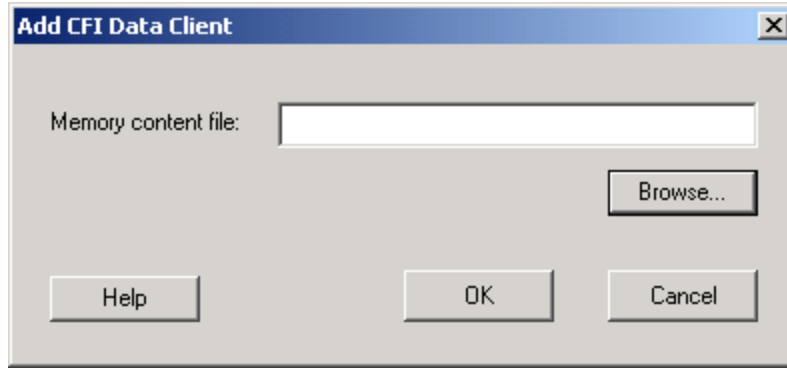


Figure 52 · CFI Data Dialog Box

Output is top\_level.vhd and top\_level.efc if you use only the CFI Data client. The EFC file is used to program the Flash Memory System Builder.

For more documentation on connecting the top-level to CoreCFI, see the CoreCFI datasheet at [http://www.actel.com/ipdocs/CoreCFI\\_HB.pdf](http://www.actel.com/ipdocs/CoreCFI_HB.pdf).

## Initialization Client

The Flash Memory System Builder initializes all the clients with the data stored in the Flash Memory when the system is powered-up. You must assert the INIT\_POWER\_UP signal to High to power up the system. You can use the Initialization Client to set initial values of the RAM/FIFO (such as a table or list of filter coefficients, MAC addresses, etc.) and ROM emulation.

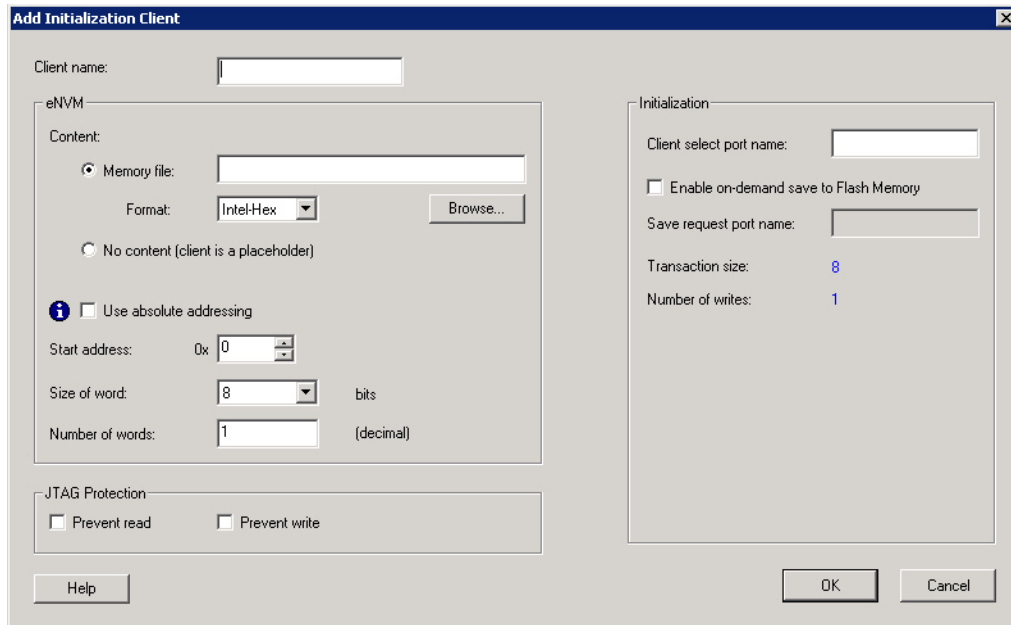


Figure 53 · Add Initialization Client Dialog Box

**Client Name** - Name of the client; the value you enter is attached before the select and enable names to group all the control signals for that client.

## eNVM

**Content** - Specify the memory content that you want to be loaded into Flash Memory. You may choose one of the two following options:

- **Memory File** - You need to select a file on disk that matches one of the following memory file formats – Intel-Hex, Motorola-S, Actel-S or Actel-Binary.

- **No content** - The client is a place holder. You will be available to load a memory file using FlashPro/FlashPoint.

**Use absolute addressing** - Lets the memory content file dictate where the client is placed in the flash memory block. The addressing in the memory content file for the client becomes absolute to the whole flash memory block. Once you choose the absolute addressing option, the software extracts the smallest address from the memory content file and uses that address as the start address for the client.

**Start Address** - The memory location of the content loaded in the eNVM.

**Size of Word** - Word size, in bits, of the initialized client; can be either 8, 16 or 32.

**Number of words** - Number of words of the client.

## Initialization

**Target address** - The address of your storage element in terms of the Cortex-M3 system memory map. Certain regions of the system memory map are not allowed to be specified for this client because they contain reserved system blocks. The tool will inform you of the legal regions for your client.

**Transaction size** - The size of the APB transfers when the data is copied from the eNVM memory region to the target destination by the Microsemi SoC system boot code.

**Number of writes** - The number of APB transfers when the data is copied from the eNVM memory region to the target destination by the Microsemi SoC system boot code. This field is automatically computed by the tool based on the eNVM content information (size and number of words) and the destination transaction size.

## JTAG Protection

Click the checkbox to protect your JTAG from read and write.

## RAM Initialization client

You must create a [RAM with Initialization core](#) before it can be imported here.

When you generate a RAM with Initialization for Fusion, the data for the RAM can be loaded into the Flash Memory System Builder such that at power-up the data is loaded into the RAM from the flash memory.

The difference between this and the regular Initialization Client is that the cascading of multiple RAM blocks is handled automatically. The RAM Initialization client takes as many initialization clients as there are RAM blocks in the cascaded RAM.

**Start Address (hexadecimal only)** - Sets the starting address for the RAM Initialization Client.

**RAM core** - [RAM with Initialization](#) core you wish to import.

## JTAG Protection.

Click the checkbox to protect your JTAG from read and write.

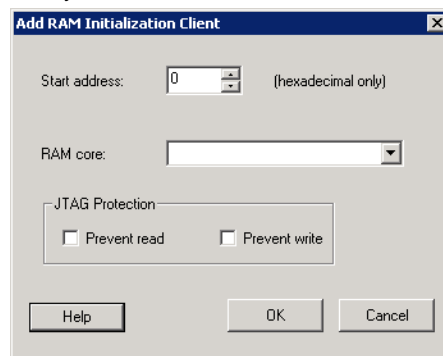


Figure 54 · RAM Initialization Client Dialog Box

## Flash Memory Block with Data Storage Client

In this scenario, the Flash memory block (FMB) is configured with a data storage client, which exposes the USER\_\* interface of the Flash Memory system. This enables complete access to the Flash Memory System interface.

In this case, the FMB serves two purposes, Initialization and Save functionality as well as allowing access to the Flash Memory System. The arbitration between these two functions is handled inside the INIT IP, the arbitration is a simple priority scheme; if any initialization or save activity is in progress then the ownership is given to the INIT IP and removed from the USER.

This implies that the user must be certain that an initialization or save activity is not triggered during their Flash Memory System access; if so, data may be corrupted. Furthermore, a user must also monitor the INITDONE signal to ensure that the INITIP is not busy before attempting to access the Flash Memory System.

In a typical situation, the INITCLK and USRCLK are connected together and exposed on the FMB top level as a single clock port (INIT\_CLK). In this case, the USRCLK is exposed and the user can connect any frequency less than 100Mhz (max frequency of the Flash Memory System) to this port. It has no relationship with any of the other clocks in the system.

To avoid any clock dependency issues between the 2 frequencies any CLIENT\_UPDATE (for save) requests given to the FMB should be held until the INIT\_DONE is deasserted, indicating that it has accepted the transaction.

## Flash Memory System Output Files

When you click **Generate**, the software generates the following files; they are saved in your components directory.

### HDL Source Files

<user\_name>.vhd/.v – Top level design that combines all the blocks together  
<user\_name>\_init\_wrapper.vhd/.v – Initialization and configuration instantiation wrapper for this design  
<common>/<Vhdl>/<Verilog>/initcfg.vhd/v - Soft IP  
<common>/<Vhdl>/<Verilog>/initcfg\_xa.vhd/v – Soft IP  
<common>/<Vhdl>/<Verilog>/initcfg\_xb.vhd/v – Soft IP  
<common>/<Vhdl>/<Verilog>/initcfg\_xc.vhd/v – Soft IP  
<common>/<Vhdl>/<Verilog>/initcfg\_xd.vhd/v -Soft IP  
<common>/<Vhdl>/<Verilog>/initcfg\_xe.vhd/v – Soft IP  
<common>/<Vhdl>/<Verilog>/initcfg\_xf.vhd/v – Soft IP  
<common>/<Vhdl>/<Verilog>/numbits.vhd/v – package file

### Memory Files

<user\_name>.mem - Non-volatile memory file

<user\_name>.efc - Contains all the Embedded Flash Memory partitions defined by each client of the Flash Memory System. FlashPoint uses the EFC file to create the STAPL file that programs the embedded Flash memory.

The Flash Memory System Builder creates only one EFC file per Embedded Flash Memory System. A Fusion device may contain from 1-4 embedded Flash memories. You must use the FlashPoint user interface to associate each embedded Flash memory in the design with an EFC file.

### Configuration Files

<user\_name>.cfg – Captures information about the settings that were specified for the system.

<user\_name>.gen – Core project file; stores information about your Flash Memory System so that you can save your settings.



<user\_name>.cxf – Core configuration file that contains information required by Libero SoC for file management.

## Log Files

<user\_name>.log

# Memory File Formats in Flash Memory System Builder

The following memory file formats are available as input files into the Flash Memory System Builder:

- Actel BINARY
- [INTEL-HEX](#)
- [MOTOROLA S-record](#)
- [ACTEL-HEX](#)

An example of how to [interpret the memory content](#) is listed below.

## Actel BINARY

The simplest memory format. Each memfile contains as many rows as there are words. Each row is one word, where the number of binary digits equals the word size in bits. This format has a very strict syntax. The word size and number of rows must match exactly. The file extension is MEM; for example, file1.mem.

Example: Depth 6, Width is 8

```
01010011
11111111
01010101
11100010
10101010
11110000
```

## INTEL-HEX

Industry standard file. Extensions are HEX and IHX. For example, file2.hex or file3.ihx.

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:

```
:11aaaatt[dd... ]cc
```

Where:

- : is the start code of every Intel Hex record
- 11 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 Microsemi SoC tools )
  - 04 extended linear address record
  - 05 start linear address record ( ignored by Microsemi SoC tools )
- [dd...] is a sequence of n bytes of the data; n is equivalent to what was specified in the 11 field
- cc is a checksum of count, address, and data

Example Intel Hex Record:

```
:10000000112233445566778899FFFA
```

Where 11 is the LSB and FF is the MSB.

## MOTOROLA S-record

Industry standard file. File extension is S, such as file4.s

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:

```
S t l l a a a a [ d d . . . ] c c
```

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.

## Actel-HEX

A simple address/data pair format. All the addresses that have content are specified. Addresses with no content specified will be initialized to zeroes. The file extension is AHX, such as filex.ahx. The format is:

```
AA:D0D1D2
```

Where AA is the address location in hex. D0 is the MSB and D2 is the LSB.

The data size must match the word size. Example: Depth 6, Width is 8

```
00:FF
```

```
01:AB
```

```
02:CD
```

```
03:EF
```

```
04:12
```

```
05:BB
```

All other addresses will be zeroes.

## Interpretation of the Memory Content

### Absolute vs Relative Addressing

In Relative Addressing, the addresses in the memory content file did not determine where the client was placed in memory. You specify the location of the client by entering the start address. This becomes the 0 address from the memory content file perspective and the client is populated accordingly.

For example, if we place a client at 0x80 and the content of the memory file is as follows:

Address: 0x0000 data: 0102030405060708

Address: 0x0008 data: 090A0B0C0D0E0F10

Then the first set of bytes of this data is written to address 0x80 + 0000 in the flash memory block. The second set of bytes is written to address 0x80 + 0008 = 0x88, and so on.

Thus the addresses in the memory content file are relative to the client itself. Where the client is placed in memory is secondary.

For absolute addressing (available in the [Initialization Client](#) and the [Data Storage Client](#)), the memory content file dictates where the client is placed in the flash memory block. So the addressing in the memory content file for the client becomes absolute to the whole flash memory block. Once you enable absolute addressing option, the software extracts the smallest address from the memory content file and uses that address as the start address for the client.

### Data Interpretation Example

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.

---

# Memory and Controllers

---

## Creating a RAM for SmartFusion, IGLOO, ProASIC3 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 SmartFusion, IGLOO, ProASIC3 and Fusion Summary](#)

[Two Port RAM for SmartFusion, IGLOO, ProASIC3 and Fusion Summary](#)

### Caveats for RAM generation in Libero SoC

- 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.

## Soft FIFO Controller

### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

## 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.

## 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.

## 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.

## 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 120 · 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
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

Name	Type	GENFILE Parameter	Description
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 121 · 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



Name	Type	GENFILE Parameter	Description
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
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.

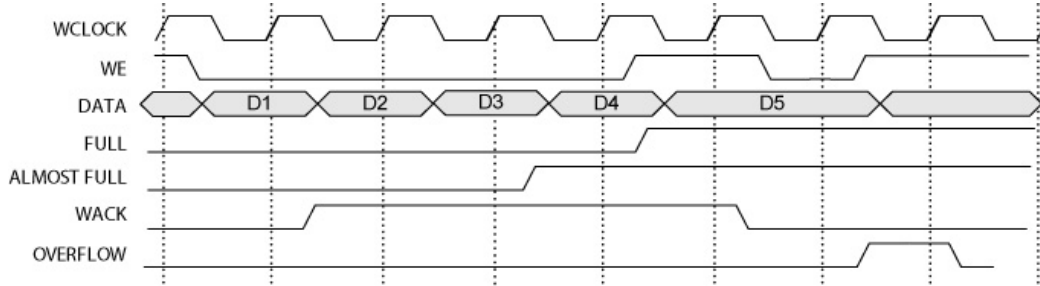


Figure 55 · Write Operation and Flags

## 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.

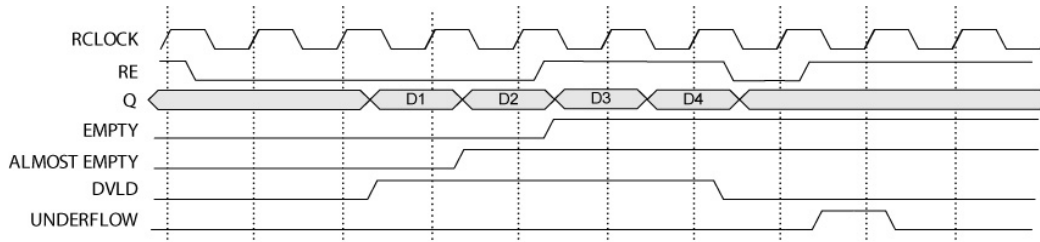


Figure 56 · Read Operation and Flags

## 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.

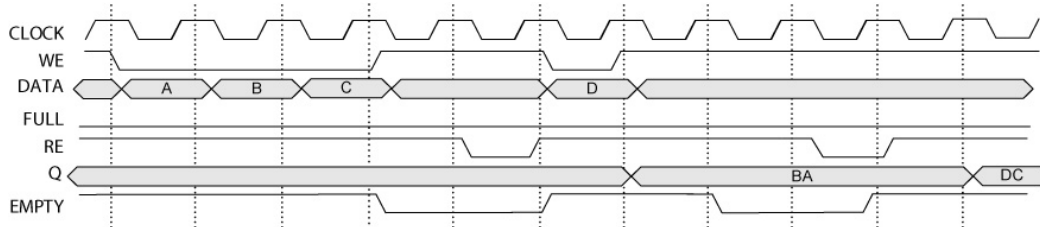


Figure 57 - Write and Read Operations with Variable Aspect

## Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 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 SmartFusion, IGLOO, ProASIC3 and Fusion Functionality](#)

[Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 and Fusion Description](#)

[Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 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.

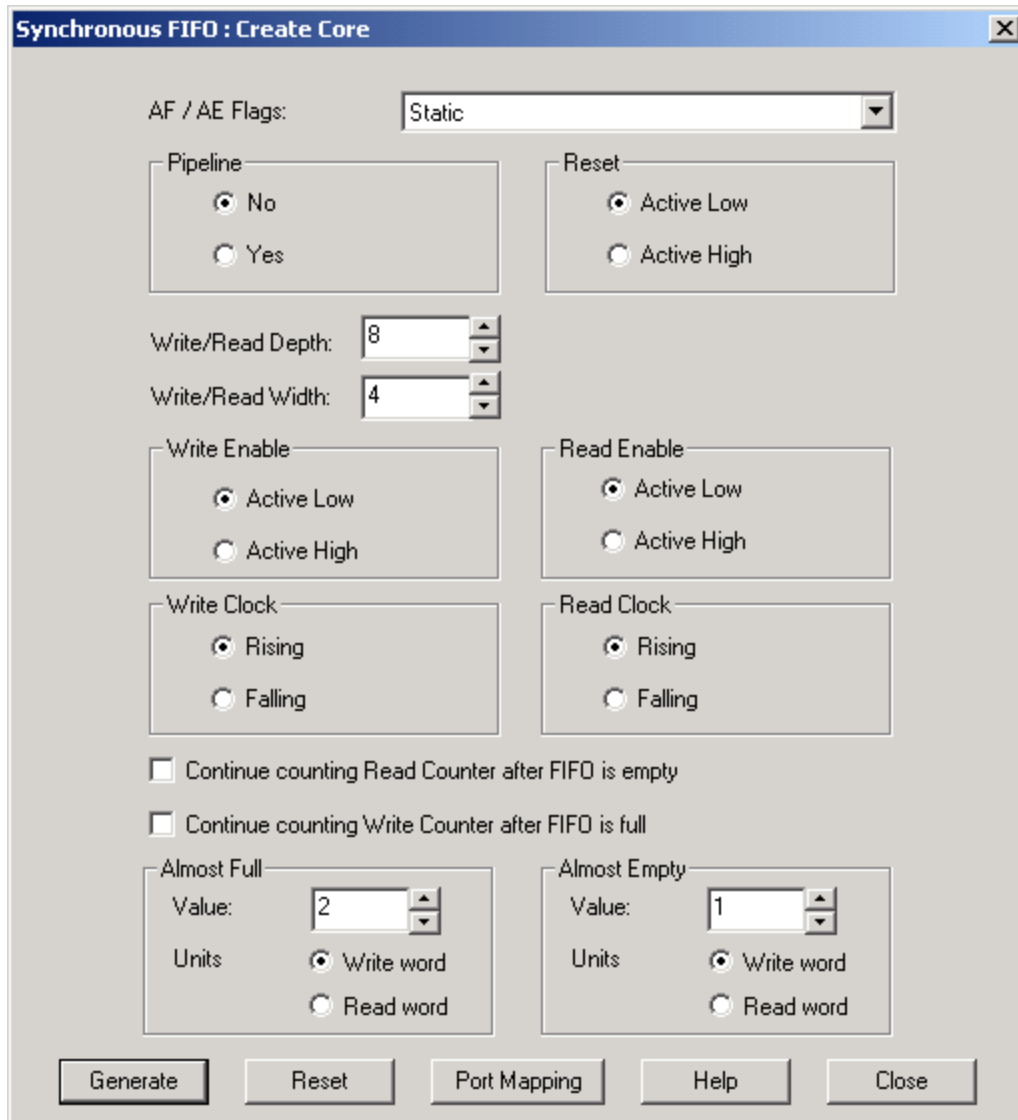


Figure 58 · Synchronous FIFO Configuration Screen

## Synchronous FIFO for SmartFusion, IGLOO, ProASIC3 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 SmartFusion, IGLOO, ProASIC3 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 SmartFusion, IGLOO, ProASIC3 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.

## 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 Microsemi 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:

```
:l1aaaatt[dd... ]cc
```

Where:

- l 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 Microsemi tools )
  - 04 extended linear address record
  - 05 start linear address record ( ignored by Microsemi 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:

```
S1l1aaaa[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 (SmartFusion2, IGLOO2, SmartFusion, IGLOO, ProASIC3, Fusion).



**To open the RAM Content Manager:**

1. From the **Options** menu, choose **Workspace Settings** and set your device family to **Fusion, ProASIC3** or **ProASIC3E**. 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.

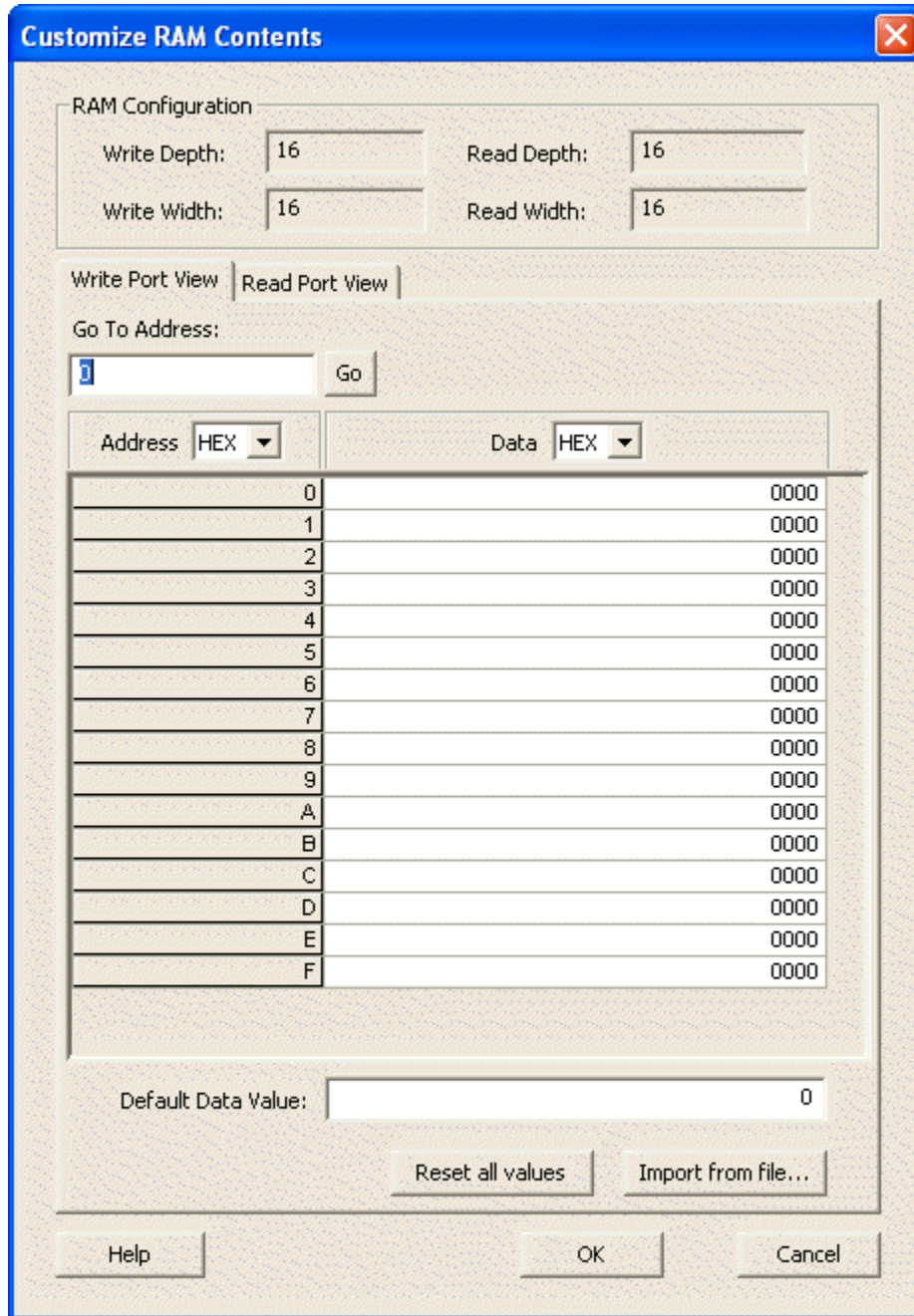


Figure 59 · RAM Content Manager

## 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 be 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 Summary

### Supported Families

SmartFusion, IGLOO, ProASIC3, Fusion

### Related Topics

[Dual Port RAM for SmartFusion, IGLOO, ProASIC3, Fusion Functionality](#)

[Dual Port RAM for SmartFusion, IGLOO, ProASIC3, Fusion Port Description](#)

[Dual Port RAM for SmartFusion, IGLOO, ProASIC3, Fusion I/O Description](#)

[Dual Port RAM for SmartFusion, IGLOO, ProASIC3, 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.

#### ***Initialize RAM for Simulation***

Click the checkbox to enable Customize RAM Content button. Customizing RAM Content opens the [RAM Content Manager](#) and 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.

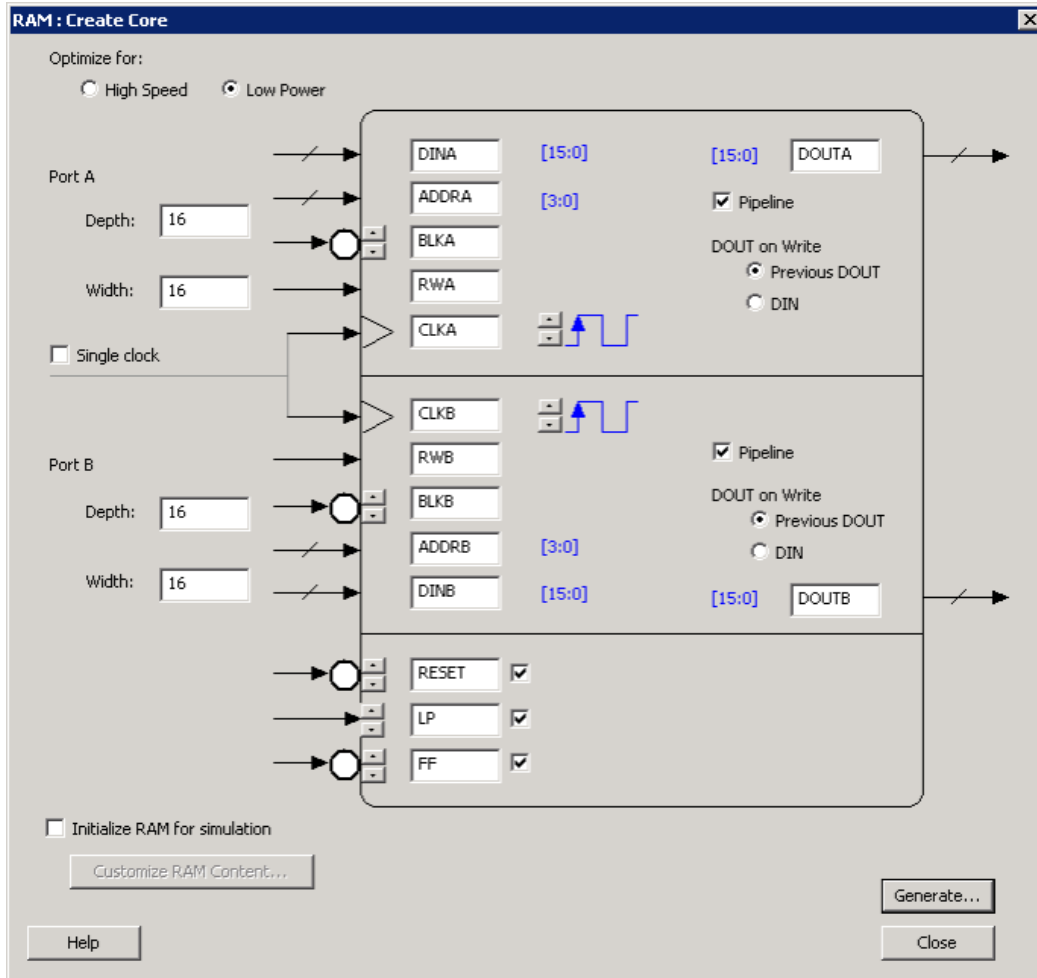


Figure 60 · Dual Port RAM Dialog Box

## Dual Port RAM 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.

**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 Port Description

The table below lists the Dual Port RAM signals in generated netlists.

## RAM with Initialization (SmartFusion and Fusion only)

The [RAM with Initialization](#) is nearly identical to the standard RAM, except that it generates extra logic so that it can interface with the 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 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.

Table 122 · Dual Port RAM Port 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_PIN	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 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 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.
- You must avoid simultaneous read and write from the same address at the same time / clock cycle. 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.

## Dual Port RAM for SmartFusion and Fusion I/O Description

This section summarizes the additional ports available when there is flash memory available in the device.

### RAM with Initialization

The RAM with Initialization is nearly identical to the standard RAM, except that it generates extra logic so that it can interface with the 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 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.

Table 123 · SmartFusion 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.
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

Name	Direction	Description
		INITADDR port for SAVE operations.

## Two Port RAM

### Supported Families

SmartFusion, IGLOO, ProASIC3 and Fusion

### Related Topics

[Two Port RAM Functionality](#)

[Two Port RAM I/O Description](#)

[Two Port RAM Port 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.

#### Initialize RAM for Simulation

Click the checkbox to enable Customize RAM Content button. Customizing RAM Content opens the [RAM Content Manager](#) and 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.



Figure 61 · 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 Port Description

The table below lists the Two Port RAM signals in generated netlists.

## RAM with Initialization (SmartFusion and Fusion only)

The [RAM with Initialization](#) is nearly identical to the standard RAM, except that it generates extra logic so that it can interface with the 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 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.

Table 124 · Port Description

Name	Type	Genfile Parameter	Bit/Bus	Description
WADDR	IN	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	IN	RADDRESS_PN	Bus	Read address
RD	OUT	DATA_OUT_PN	Bus	Read data input
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.
- You must avoid simultaneous read and write from the same address at the same time / clock cycle. 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 signal. This becomes extremely important in cases where multiple RAM blocks are cascaded for deeper memories.

## Two Port RAM I/O Description

The table below shows the Two Port RAM I/O's, as well as additional ports that are exposed for a Fusion RAM with Initialization. These ports are meant to be connected to the Flash Memory module.

Table 125 · 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	<p>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.</p> <p>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.</p> <p>These signals need to be connected to the &lt;client_name&gt;_block_i_DAT_VAL from the Initialization / Flash Memory module.</p>
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	<p>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.</p> <p>The address location to be read is also driven from the INITADDR port for SAVE operations.</p>

## RAM with Initialization

The RAM with Initialization is nearly identical to the standard RAM, except that it generates extra logic so that it can interface with the 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 two 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 126 · 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.  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.

Name	Direction	Description
		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.

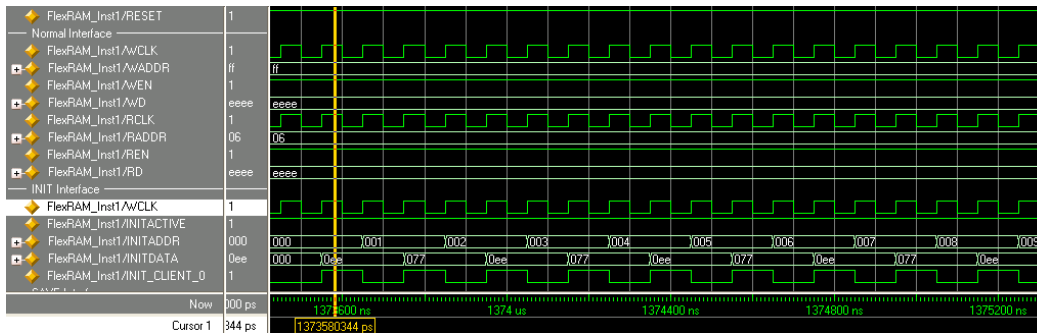


Figure 62 · FlexRAM Initialization Diagram

### 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.

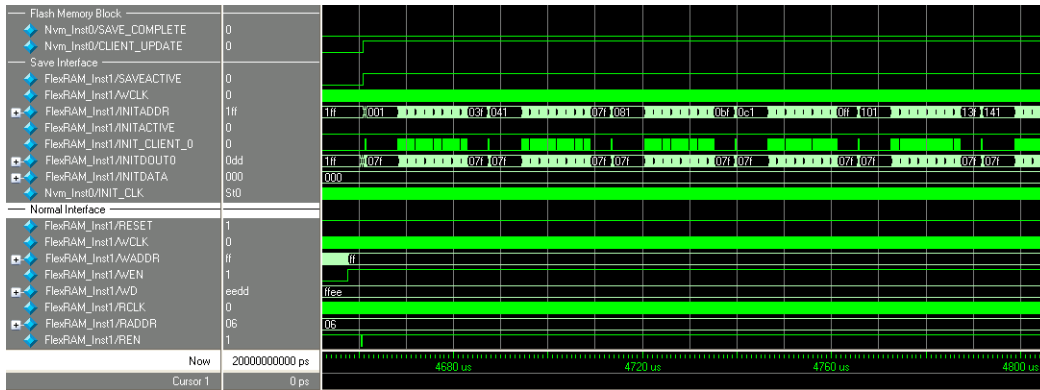


Figure 63 · FlexRAM Save Diagram (Full)

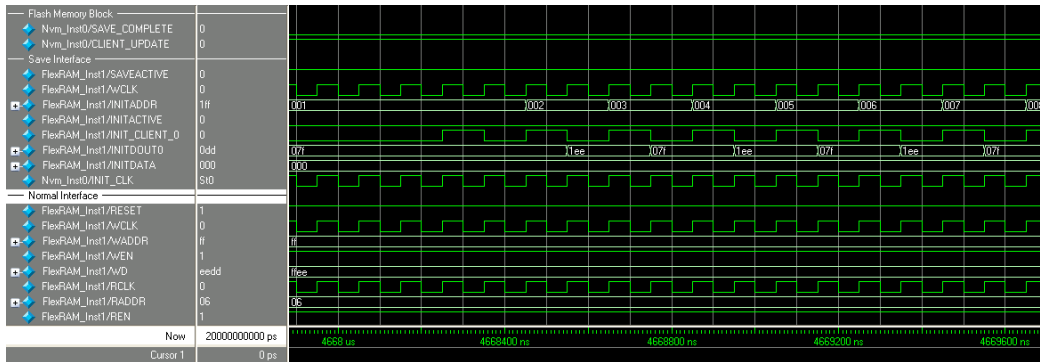


Figure 64 · FlexRAM Save Diagram (Detail)

## 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



### 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.

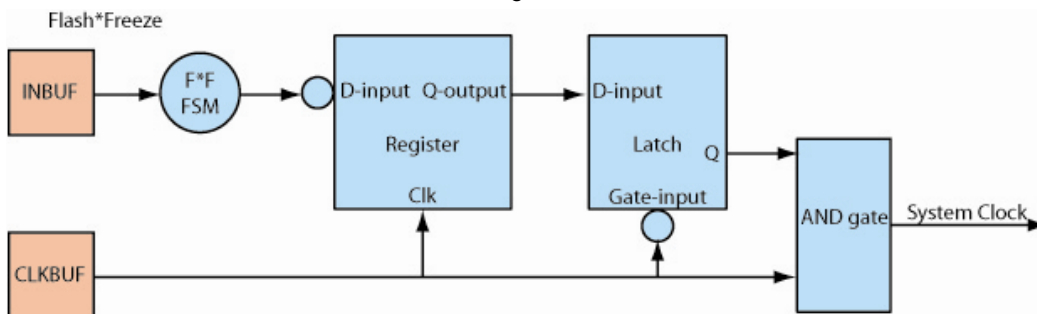


Figure 65 - Clock Gating (Filter) Block

### 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. Microsemi 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

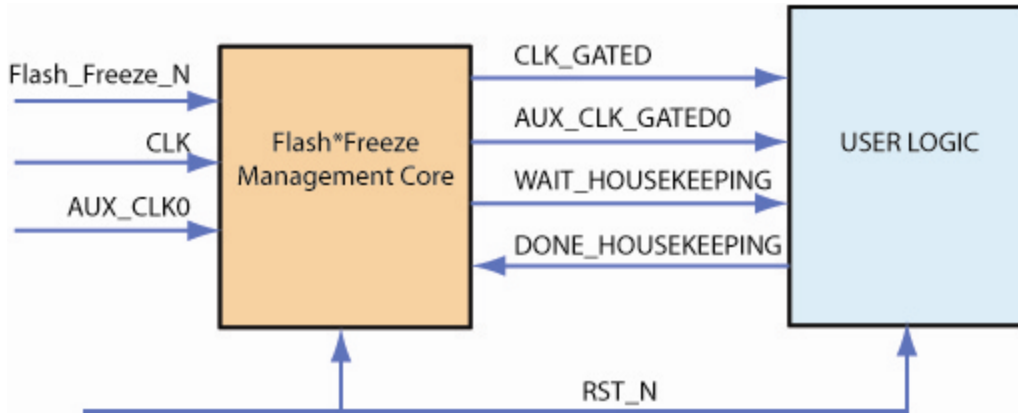


Figure 66 · Flash\*Freeze Use Model

## Flash\*Freeze Management Core I/O Description

Table 127 · Signals

Signal	Direction	Polarity	Description
Flash_Freeze_n	Input	Active Low	Asynchronous input for Flash*Freeze mode
RST_N	Input	Active Low	Asynchronous Reset
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	<b>High</b> or <b>Low</b>	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

Signal	Direction	Polarity	Description
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 128 · Parameter Description

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

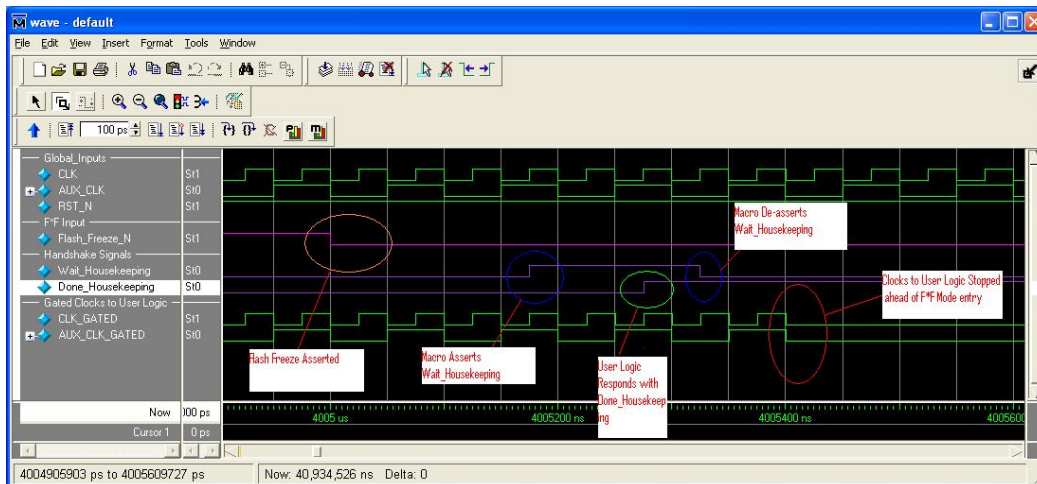


Figure 67 · Flash\*Freeze Asserted

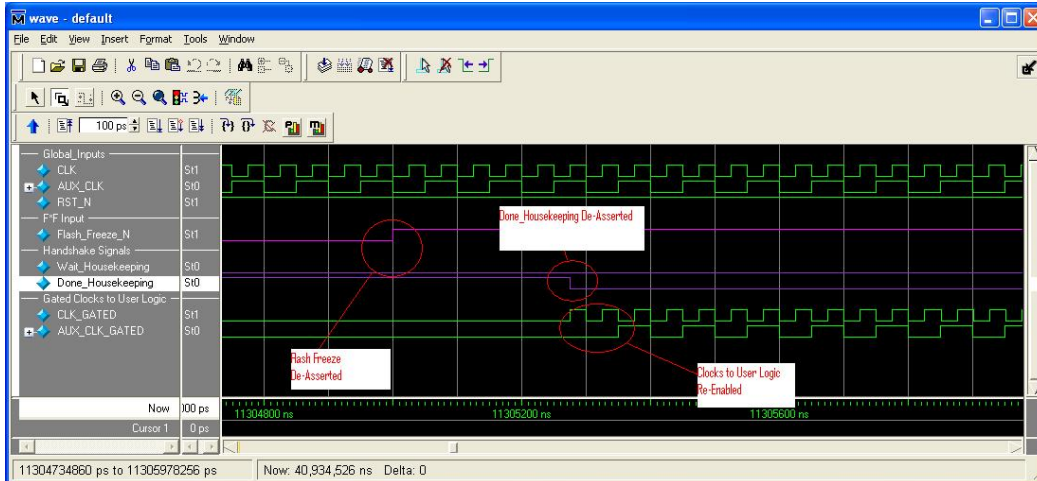


Figure 68 · Flash\*Freeze De-Asserted

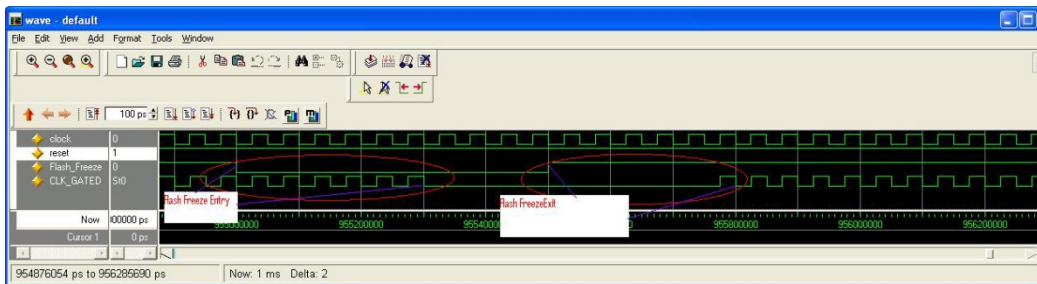


Figure 69 · No Housekeeping with Flash\*Freeze

## 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.

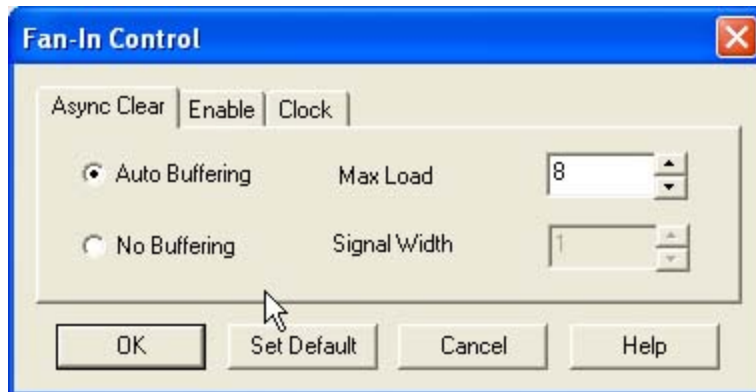


Figure 70 · Fan-In Control Dialog Box

## 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.
- 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.

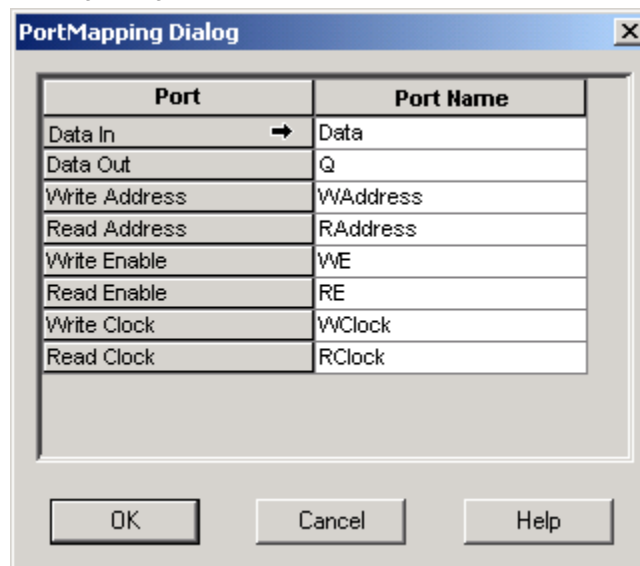


Figure 71 · Port Mapping Dialog Box

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

---

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

## Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**  
From the rest of the world, call **650.318.4460**  
Fax, from anywhere in the world **650. 318.8044**

## Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

## Technical Support

For Microsemi SoC Products Support, visit <http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support>.

## Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at <http://www.microsemi.com/soc/>.

## Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

### Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is [soc\\_tech@microsemi.com](mailto:soc_tech@microsemi.com).

### My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to [My Cases](#).

### Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email ([soc\\_tech@microsemi.com](mailto:soc_tech@microsemi.com)) or contact a local sales office. Visit [About Us](#) for [sales office listings](#) and [corporate contacts](#).

## ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via [soc\\_tech@microsemi.com](mailto:soc_tech@microsemi.com). Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page





Microsemi Corporate Headquarters  
One Enterprise, Aliso Viejo,  
CA 92656 USA

**Within the USA:** +1 (800) 713-4113  
**Outside the USA:** +1 (949) 380-6100  
**Sales:** +1 (949) 380-6136  
**Fax:** +1 (949) 215-4996

**E-mail:** [sales.support@microsemi.com](mailto:sales.support@microsemi.com)

© 2016 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

### About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at [www.microsemi.com](http://www.microsemi.com)

---

Microsemi makes no warranty, representation, or guarantee regarding the information contained herein or the suitability of its products and services for any particular purpose, nor does Microsemi assume any liability whatsoever arising out of the application or use of any product or circuit. The products sold hereunder and any other products sold by Microsemi have been subject to limited testing and should not be used in conjunction with mission-critical equipment or applications. Any performance specifications are believed to be reliable but are not verified, and Buyer must conduct and complete all performance and other testing of the products, alone and together with, or installed in, any end-products. Buyer shall not rely on any data and performance specifications or parameters provided by Microsemi. It is the Buyer's responsibility to independently determine suitability of any products and to test and verify the same. The information provided by Microsemi hereunder is provided "as is, where is" and with all faults, and the entire risk associated with such information is entirely with the Buyer. Microsemi does not grant, explicitly or implicitly, to any party any patent rights, licenses, or any other IP rights, whether with regard to such information itself or anything described by such information. Information provided in this document is proprietary to Microsemi, and Microsemi reserves the right to make any changes to the information in this document or to any products and services at any time without notice