Peripherals User's Guide

v7.0

Actel®

Actel Corporation, Mountain View, CA 94043

Printed in the United States of America

Part Number: 5-02-00063-1

Release: February 2006

Trademarks

Table of Contents

Introduction

The purpose of the Peripherals User's Guide is to provide a central location for information on the Fusion peripherals.

This manual contains the following sections:

- Clock Resources – This includes the following peripherals:
  - RC Oscillator
  - Crystal Oscillator
  - Clock Conditioning Circuits
  - Delayed Clock and divided clock
  - PLL Macro
  - No-Glitch MUX
- Real Time Counter System –This includes the following peripherals:
  - Real-Time Counter
  - Crystal Oscillator (as related to RTC)
  - Voltage Regulator Initialization
  - Voltage Regulator
  - 1.5 V Voltage Regulator
  - Voltage Regulator Power Supply Monitor (VRPSM)
- Embedded Memories –This includes the following peripherals:
  - Flash Memory Block
  - FlashROM
  - SRAM
  - FIFO
- Analog Peripherals – This includes the following peripherals:
  - Analog Quad
  - Voltage Monitor
  - Current Monitor
  - Gate Driver
  - Temperature Monitor
  - Digital I/O
  - ADC
  - Analog Mux
- User JTAG

You can access additional information for Fusion peripherals in the Fusion datasheet (http://www.actel.com/documents/Fusion_DS.pdf). The datasheet for Fusion devices describes specific silicon features of the peripherals. It includes the description of the silicon macro and any associated register settings and configuration options. It also includes any timing characteristics for the peripheral.

This manual describes how to implement each of the peripherals, including software generation and configuration, and export of programming files. It also covers use cases where appropriate and any tips and tricks for optimal usage of the devices.

In some cases, this manual provides information on the board level considerations.

The Peripherals User's Guide also provides reference and navigation to related documents that exist outside of the datasheet and Peripherals User's Guide. In most cases, these are ProASIC3 Application Notes that also apply to Fusion.

Design Flow

The design flow for the Fusion devices is covered in the Fusion Design Flow Tutorial (http://www.actel.com/documents/fusion_df_ug.pdf). This tutorial uses a Power Management design to provide an example that steps through the entire flow. Use this tutorial in conjunction with the datasheet and Peripherals User's Guide to get a complete understanding of the Fusion device.

SmartGen User Interface

The SmartGen software generates a large variety of commonly used functions. You can generate structural netlists in EDIF, VHDL, and Verilog. Furthermore, you can generate VHDL and Verilog behavioral models for most parameterized functions (the behavioral models may be used in a simulation environment). SmartGen includes workspace and core-management features.

SmartGen is divided into four sections: the Core Catalog, the Intellectual Property Catalog (IP Catalog), the Variety View window, the Configured Core View window, and the Log Window (as shown in Figure 1).

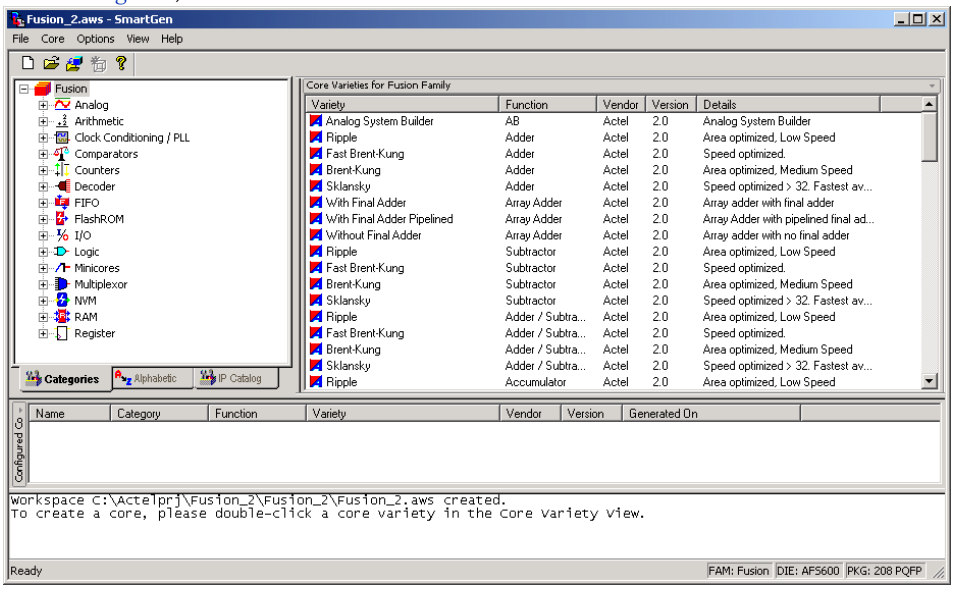


Figure 1. SmartGen Main Window

Use this interface to browse your cores, review your configured cores, and select cores to create or modify.

The **Variety View** window displays a list of core varieties available for the core you selected in the Categories tab. For example, if you select the Arithmetic core type, and then select the Adder core, the Variety View window displays a list that includes Sklansky, Fast Brent-Kung, Ripple, etc.

Click a heading in the Variety View window to sort your list of cores. Cores are listed in Function order by default (Arithmetic cores, Clock Conditioning cores, Comparators, etc.). If you click a column heading they are sorted alphabetically by that column value (except for Version - then they are sorted numerically).

Create a workspace

You must create a workspace to generate a core in SmartGen.

A workspace defines your family and the default directory in which you save your configured cores. If you wish, you may save your workspace in a different directory. The workspace is a logical grouping for your cores; each workspace contains cores for a specific product family.

To create a workspace:

1. Start SmartGen. When you open SmartGen it asks if you wish to create a workspace or open an existing workspace. You may choose to create a workspace or open an existing workspace. Select the check box to hide this dialog the next time you open SmartGen.
2. From the **File** menu, choose **New Workspace**.
3. Specify the **Workspace name**, **Workspace location** (click the **Browse** button to navigate to or create a directory), **Family**, **Die**, **Package** and **Default netlist format.**

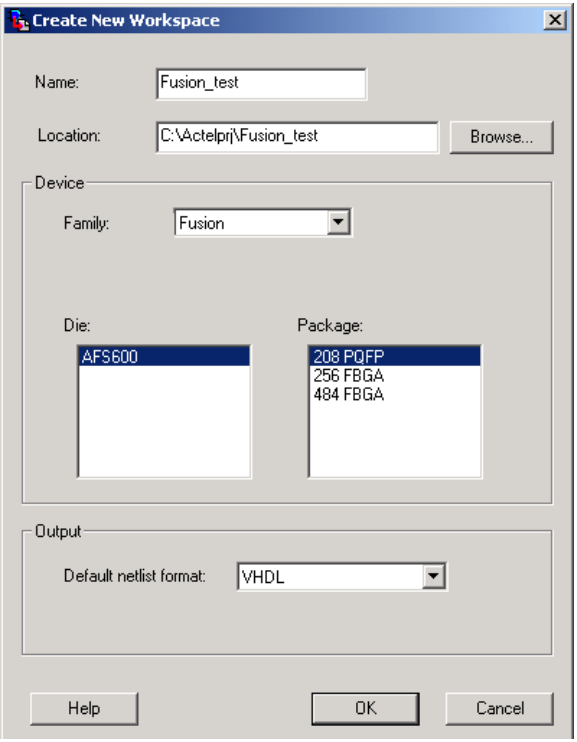


Figure 2. Create New Workspace Dialog Box

4. Click OK to create the new workspace. The new workspace appears in the SmartGen window. By default, the list of cores available in a new workspace is sorted by **Categories**. Click the **Alphabetic** tab to display the complete list of SmartGen cores.

Open a workspace

You must open a workspace in SmartGen to generate cores. If you have never opened a workspace, you can create one. To open an existing workspace:

1. Start SmartGen.
2. From the File menu, choose Open Workspace, or click the Open Workspace button in the SmartGen toolbar. This opens the Open Workspace dialog box (as shown in Figure 3).

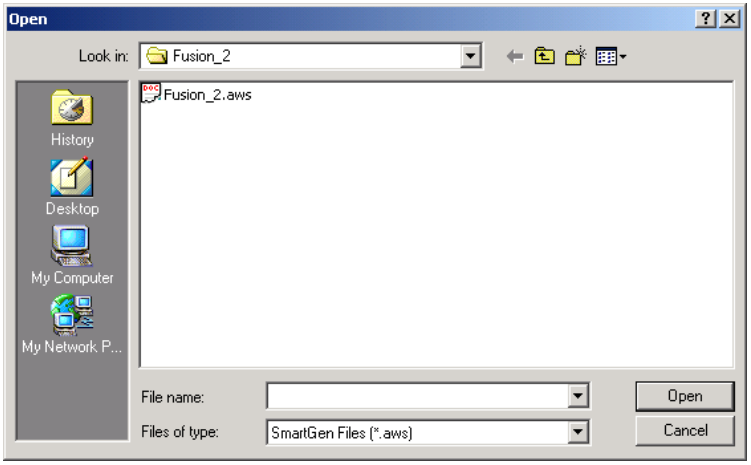


Figure 3. Open Dialog Box

3. Navigate to the workspace you wish to open and click **Open.**

SmartGen audits the workspace for existing cores and their associated files (CXF, LOG, VHD/V, and other auxiliary files) each time you open it. SmartGen reports if any of the files associated with an existing core are missing. If a CXF (core configuration) file for a core is missing, SmartGen loads the core into the workspace with default parameters.

Create a New core in SmartGen

You can create a new core in SmartGen only after you have created a workspace.

To create a new core in SmartGen:

1. Select a core type in the **Categories** tab (Arithmetic, Comparator, Converters, etc.). The **Variety View** window displays the list of configurable cores.
2. Double-click the core variety you wish to generate, or right-click and choose **Create Core**. The core configuration dialog box appears.
   The configuration dialog box varies depending on which core you select.
3. Set the parameters for your core and click **Generate**. The **Save As** dialog box appears.
4. Specify the name of your new core and click **Save** to continue. SmartGen saves your core to the specified directory and adds your core to your workspace (in the Configured Core View Window). If you deliberately save the core in a different directory than the workspace, the core will be saved, but will not be added to the workspace.
   Your core remains in the workspace until you choose to **Remove** it.

Note: You cannot create two cores with the same name. SmartGen is case-insensitive; "core_A" is equivalent to "core_a".

Reconfigure an Existing core in SmartGen

You can reconfigure your cores in SmartGen. To do so, you must first have added a core to your workspace.

To reconfigure a core in SmartGen:

1. Select the core in the Configured Core View window.
2. From the **Core** menu, choose **Modify Core**. The configuration dialog box opens to display the configuration options specified in the core. Your core configuration options vary depending on the device family you selected when you created your workspace.

Import a SmartGen core

You can import SmartGen cores into your current workspace. To do so:

1. From the **File** menu, choose **Import Core**. This displays the Import Core dialog box (as shown in Figure 4).

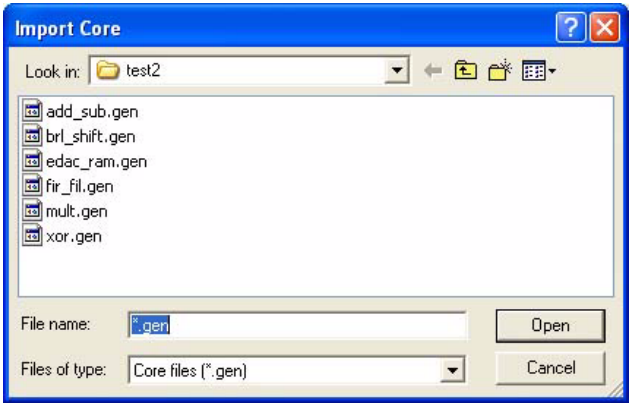


Figure 4. Import Core Dialog Box

2. Navigate to the core you wish to import and click **Open**. The core is added to your current workspace and remains in your workspace until you **Remove** it.

   Note: You cannot import two cores with the same name. SmartGen is case-insensitive; "core_A" is equivalent to "core_a".

Remove or Delete a core

You can delete cores from your Configured Core View (and leave them on the disk), or you may delete them from your disk entirely. Select the core(s) you want to delete and press the **Delete** key, or right-click and choose **Remove from Workspace** or **Remove from Disk and Workspace**.

You do not need to save your workspace.

Removing a core from your workspace does not delete the core from your directory. You can import the core later if you wish. If you choose **Remove from Disk and Workspace** the core is deleted forever.

Document Conventions

The <act_fam> variable represents an Actel device family. To reference an actual family, substitute the name of the Actel device when you see this variable. The <vhd_fam> variable represents Compiled VHDL libraries. To reference an actual compiled library, substitute the name of the library when you see this variable. Compiled VHDL libraries must begin with an alpha character.

Document Assumptions

The information in this manual is based on the following assumptions:

1. You have installed the Libero IDE software.
2. You are familiar with PCs and Windows operating environments.
3. You are familiar with FPGA architecture and FPGA design software.
4. You are familiar with Libero IDE software and have prior experience with the ProASIC or ProASIC^{PLUS} families.

Actel Manual

Designer and Libero include printed and online manuals. The online manuals are in PDF format and available from Libero and Designer's Start Menus and on the CD-ROM. From the Start menu choose:

- Programs > Libero > Libero Documentation
- Programs > Designer > Designer Help > Designer Documentation

Also, the online manuals are in PDF format on the CD-ROM in the "/manuals" directory. These manuals are also installed onto your system when you install the Designer software. To view the online manuals, you must install Adobe® Acrobat Reader® from the CD-ROM.

Online Help

The Libero IDE software comes with online help. Online help specific to each software tool is available in Libero, Designer, SmartGen, Silicon Expert, Silicon Explorer II, and Silicon Sculptor.

Technical Documentation

You can access all additional technical documentation from the Actel web site at http://www.actel.com/techdocs/

1 Clock Resources

The Fusion family has a robust collection of clocking peripherals, as shown in the block diagram in Figure 1-1. These on-chip resources enable the creation, manipulation, and distribution of many clock signals. The Fusion integrated RC oscillator produces a 100 MHz clock source with no external components.

For systems that require more precise clock signals, the Actel Fusion family supports an on-chip crystal oscillator circuit. The integrated PLLs in each Fusion device can use the RC oscillator, crystal oscillator, or another on-chip clock signal as a source. These PLLs offer a variety of capabilities to modify the clock source (multiply, divide, synchronize, advance, or delay).

Fusion incorporates six CCC blocks; the CCCs enable access to Fusion global and local clock distribution nets, as described in the Clock Resources (VersaNets) section of the datasheet.

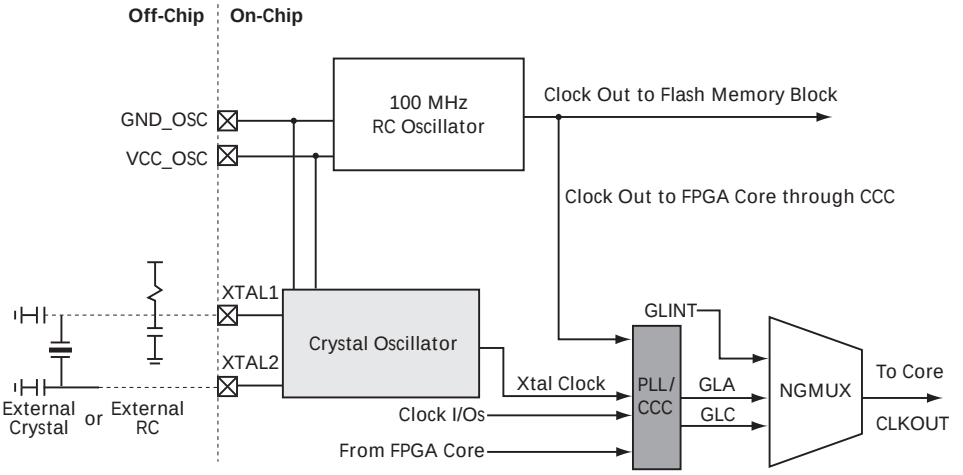


Figure 1-1. Fusion Clocking System1

This chapter includes the following sections:

- RC Oscillator - The Fusion macro for the RC oscillator is RCOSC and this can be configured through SmartGen
- Crystal Oscillator - The Fusion macro for Crystal Oscillator is XTLOSC and can be configured in different modes by SmartGen
- Clock Conditioning - Clock Conditioning for Fusion includes selecting from multiple clock sources, providing Clock Delays and Clock dividers
- PLL Macro - Describes the Static PLL for Fusion
- Dynamic Clock Conditioning - Describes the use of PLL and CCC for dynamic reconfiguration

- The No-Glitch Mux - The Fusion macro for No-Glitch Mux is NGMUX and allows switching between multiple clock sources without creating glitches in the cock network

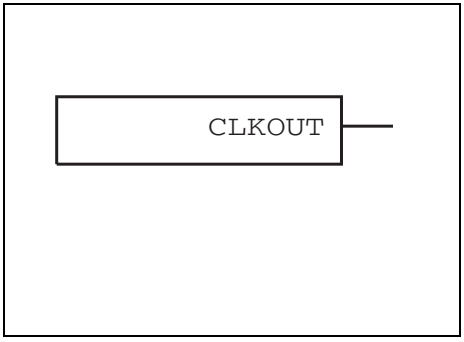
RC Oscillator (RCOSC)

The RC oscillator is an on-chip free running clock source generating a 100 MHz clock. It can be used as a source clock for both on-chip and off-chip resources. When used in conjunction with the Fusion PLL and CCC circuits, the RC oscillator clock source can be used to generate clocks of varying frequency and phase.

The Fusion Datasheet includes Timing Characteristics for the RCOSC Peripheral

The RCOSC can drive any of the clock macros directly. To drive any macros in the core, it must be routed through a CLKSRC. SmartGen automatically instantiates the CLKSRC if you choose the "Drive Internal Logic Directly" option.

RCOSC	Fusion



Function

On-chip free-running clock source that generates a 100 Mhz clock.

Refer to the Clocking Resources of the Fusion datasheet for more information on this macro. The Fusion datasheet is available at http://www.actel.com.

Input	Output
	CLKOUT

RC Oscillator Tips

When using the RC oscillator it is important to note that although there are no additional connections on the Macro for logic connections, you must connect GNDOSC and VCCOSC

externally to provide clean power and ground source for the RCOSC component. VCCOSC is 3.3V; these pins are required for both RCOSC and XTLOSC.

RC Oscillator Connections

The RC Oscillator as shown in Figure 1-1 on page 17 can drive the Flash Memory Block through a direct connection.

In order to drive logic in the core the RCOSC output must be connected to the CLKSRC macro.

VCCOSC Oscillator power supply (3.3 V)

Power supply for both integrated RC oscillator and crystal oscillator circuit.

GNDOSC Oscillator ground

Ground supply for both integrated RC oscillator and crystal oscillator circuit.

Crystal Oscillator

The on-chip crystal oscillator circuit works with an offchip 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 other the system clock networks, both on-chip and off. When combined with the on-chip CCC/PLL blocks, a wide range of clock frequencies can be created to support various design requirements.

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. SmartGen automatically instantiates the CLKSRC if you choose the "Drive Internal Logic Directly" option.

You must specify the mode of the crystal oscillator. There are five available.

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

The Low/Medium/High gain oscillators are configured to support an external crystal or ceramic resonator. The difference is the supported crystal frequency or resonator.

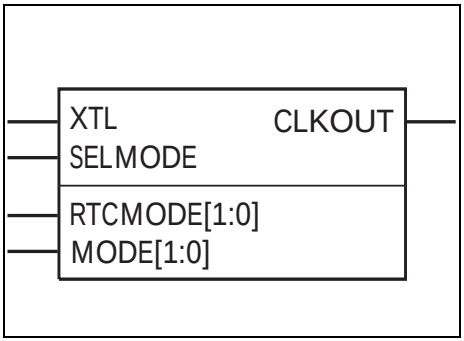
Low Gain - Gain of 2kHz to 200kHz.

Medium Gain - Gain of 200 kHz to 2 MHz.

High Gain - Gain of 2 MHz to 20 MHz.

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

XTLOSC	Fusion



Function

On-chip crystal oscillator circuit that works with an off-chip crystal to generate a high-precision clock.

Refer to the Clocking Resources section of the Fusion datasheet for more information on this macro. The Fusion datasheet is available at http://www.actel.com.

Input	Output
	CLKOUT

RTCMODE is used when SELMODE is 1, and MODE is used when SELMODE is 0.

The XTLOSC requires a physical connection to an external crystal, ceramic resonator, or a resistor/capacitor network. For simulation purposes you can use the XTL pin to provide a clock signal running at the desired input frequency.

Crystal Oscillator Tips

When using the Crystal oscillator it is important to note that not all macro connections are shown on the Logic Macro, you must connect GNDOSC and VCCOSC externally to provide clean power and ground source for the XTLOSC component. VCCOSC is 3.3V. These pins are required for both RCOSC and XTLOSC.

In addition the XTL pin of the macro is connected on the package to XTAL1 Crystal Oscillator circuit Input.

Crystal Oscillator Connections

The Crystal Oscillator can drive the PLL/CC macros or can use the CLKSRC macro to drive to the core directly. This option is available in the SmartGen implementation.

XTAL1 Crystal Oscillator circuit input

Input to crystal oscillator circuit. Pin for connecting external crystal, ceramic resonator, RC network, or external clock input. When using an external crystal or ceramic oscillator Actel recommends that

you use external capacitors (<2 MHz 100 pF, >2 MHz – 15 pF). If using an external RC network or clock input, use XTAL1 and leave XTAL2 unconnected.

XTAL2 Crystal Oscillator circuit input

Input to crystal oscillator circuit. Pin for connecting external crystal, ceramic resonator, RC network, or external clock input. When using an external crystal or ceramic oscillator Actel recommends that you use external capacitors (<2 MHz 100 pF, >2 MHz –15 pF). If using an external RC network or clock input, use XTAL1 and leave XTAL2 unconnected.

VCCOSC Oscillator power supply(3.3 V)

Power supply for both integrated RC oscillator and crystal oscillator circuit.

GNDOSC Oscillator ground

Ground supply for both integrated RC oscillator and crystal oscillator circuit.

Sub-block Interconnection

Common features

- These oscillators are the clock sources. They do not have direct connections to the global line, but must be connected to a CCC macro to get to the global line.
- They can not drive CCC macros placed in quadrant locations.
- They can be connected to the input port of any of the following macros:

  CLKBUF (Only non-vref versions as east and west banks do not support VREF IOs for fusion)

  CLKBIBUF

  CLKSRC

  CLKDLY

  CLKDIVDLY

  CLKDIVDLY1

  PLL

  - Cannot drive 'CLKINT' as connection from these oscillators to the CCC macros is a hardwired connection and CLKINT takes a routed input.

  - **Placement**: Only one valid location for each oscillator. COMPILE places the oscillator macros automatically; users cannot move them.

  - Only one instance of each of the oscillators can be present in any design. However, an output net can drive multiple CCC macros (maximum 6, i.e. the 6 locations for central CCCs.)

XTL special feature

XTLOSC has an input port 'XTL'. This port also has the 'PAD' property which needs to be hooked to a design port directly.

RC special feature

RC oscillator can be used to achieve the divide by half feature available in the Fusion PLL. When the PLL takes an input from the RC oscillator, user can configure the OADIVHALF/ OBDIVHALF/ OCDIVHALF to set the divide by half in addition to any other division applied for these clocks.

Note: Divide by half is available only in the bypass mode of the PLL.

Delayed Clock

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

Select the programmable delay between 0.280 ns to 5.815 ns, in steps of 160 ps, for the Output.

Divided and Delayed Clock

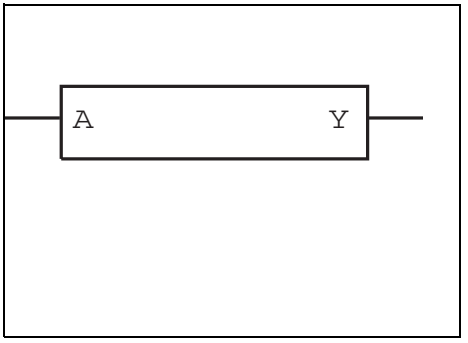
Use this core to divide down a clock and, if necessary, delay it by a given amount (such as to compensate for skew in your circuit).

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.

List of Macros

The following macros are related to peripherals you can enable in the SmartGen user interface.

CLKSRC — Fusion



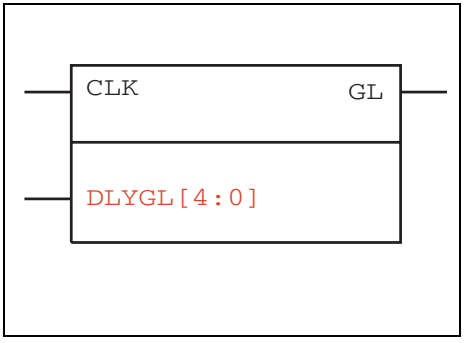
Function

Clock buffer used to connect either the RCOSC or the XTLOSC to the core.

Refer to the Clocking Resources of the Fusion datasheet for more information on this macro. The Fusion datasheet is available at http://www.actel.com.

Input	Output
A	Y

CLKDLY — Fusion, ProASIC3, ProASIC3E



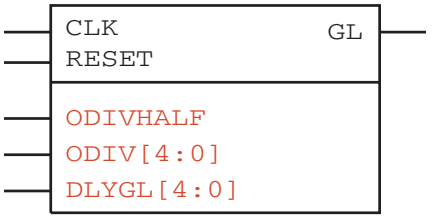
Function

Static clock with delay.

Refer to the Clocking Resources section of the Fusion datasheet for more information on this macro. They are available at http://www.actel.com.

Input	Output
CLK, DLYGL[4:0]	GL

CLKDIVDLY — Fusion



Function

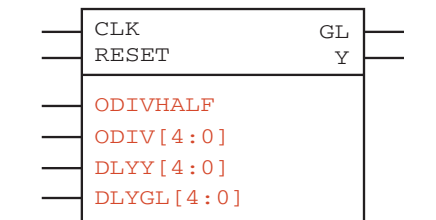
Static clock with divider and/or delay with global output driver only.

Refer to the Clocking Resources of the Fusion and ProASIC3/E datasheets for more information on this macro. They are available at http://www.actel.com.

Input
CLK, RESET, ODIVHALF, ODIV[4:0], DLYGL[4:0]

Output
GL

CLKDIVDLY1 — Fusion



Function

Static clock with divider and/or delay with both global output driver and regular net driver.

Refer to the Clocking Resources of the Fusion datasheet for more information on this macro. It is available at http://www.actel.com.

Input
CLK, RESET, ODIVHALF, ODIV[4:0], DLYY[4:0], DLYGL[4:0]

Output
GL, Y

Independent Clock Dividers: CLKDIVDLY & CLKDIVDLY1

Fusion architecture has independent clock dividers. There are two versions of clock dividers available.

CLKDIVDLY

This macro can be placed in any CCC-GLA/GLB/GLC tile. It has the configuration bits to set up the division and delay values on the input clock supplied.

CLKDIVDLY1

This macro has an extra routed input. Since only the CCC-GLB and GLC tiles have the extra routed inputs, the CLKDIVDLY1 can be placed in CCC-GLB and GLC tiles.

Static PLL for Fusion

The Static PLL 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 Fusion device to permit parameter changes (such as PLL divide ratios) during device operation. The Static PLL for Fusion includes the following features:

- 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 Fusion Static PLL 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 Static PLL 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 160 ps, of up to 5.56 ns (at 1.5V, 25C, typical process) before or after the positive clock edge of the incoming reference clock. Furthermore, the system allows for the selection of one of four clock phases of fout, at 0, 90, 180, and 270 degrees.

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

Fusion Static PLL Functionality

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

The feedback signal, to which 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 0o phase shift and zero additional time delay
- A delayed version of the VCO output, in selectable increments of 160 ps, up to 5.56 ns

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 100 µA of current and sends 0 V signals on all outputs.

Configuring the Fusion Static PLL in SmartGen

The Fusion Static PLL includes the following features (shown in Figure 1-2 on page 28):

- 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.

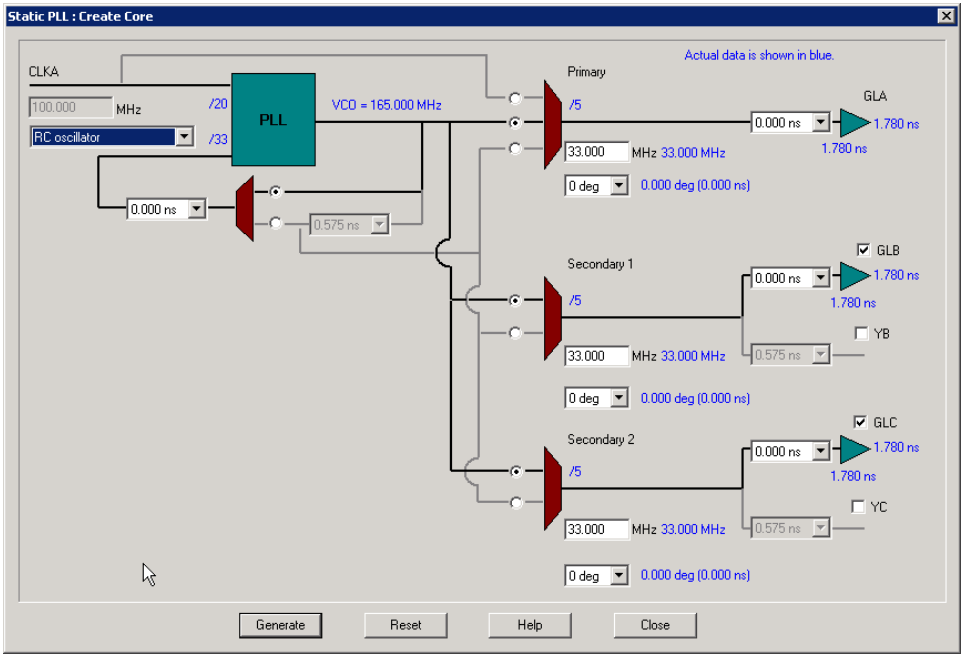


Figure 1-2. Fusion Static PLL Configuration Screen in SmartGen

After you open a new workspace in SmartGen and select the Static PLL, 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 600 ps, up to 5.56 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 the 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 600 ps, up to 5.56 ns.
6. Specify Secondary1 and Secondary2 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 600 ps, up to 5.56 ns.

Fusion Static PLL Core Restrictions in SmartGen

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

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

If SmartGen 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. SmartGen prints a message in the log file indicating the actual PLL output frequency.

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

Total Delays and Input Delays in the Fusion Static PLL

Total delays and input delays for the Fusion Static PLL are identical to the delays explained in "Total Delays" on page 36.

SmartGen Fusion Static PLL Signal Descriptions

PLL signal descriptions in Table 1-1 apply only to Fusion devices.

Table 1-1. SmartGen Fusion Static PLL Signal Description

Name	Size	Type	Required/ Optional	Function
GLA	1	Output	Req	Primary clock output

Table 1-1. SmartGen Fusion Static PLL Signal Description

Name	Size	Type	Required/ Optional	Function
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

The static PLL supports only a single input. The Combiner is able to combine the PLL with the regular CLKBUF macros and any of the CCC macros to utilize available unused globals.

In the symbol shown above, all the required user-accessible inputs and outputs are above the top horizontal line. The ones below the top line are optional inputs and outputs. The static configuration inputs are below the third line. These pins can only be connected to GND or VCC.

OADIVRST may only be used when you bypass the PLL core (i.e. OAMUX = 001).

Table 1-2 summarizes the configuration control bits.

Table 1-2. Configuration Control Bits Summary

NAME	FUNCTION
FINDIV<6:0>	7-BIT INPUT DIVIDER (/N)
FBDIV<6:0>	7-BIT FEEDBACK DIVIDER (/M)
OADIVHALF*	Division by half (see Fusion datasheet for more information)
OADIV<4:0>	5-BIT OUTPUT DIVIDER (/U)
OBDIV<4:0>	5-BIT OUTPUT DIVIDER (/V)
OCDIV<4:0>	5-BIT OUTPUT DIVIDER (/W)

*Note: * OADIVHALF may only be used when you bypass the PLL core (i.e. OAMUX = 001) and the RC Oscillator (RCOSC) drives the CLKA input.*

Table 1-2. Configuration Control Bits Summary (Continued)

NAME	FUNCTION
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
DLYGLA<4:0>	DELAY ON GLOBAL A
DLYGLB<4:0>	DELAY ON GLOBAL B
DLYGLC<4:0>	DELAY ON GLOBAL C
DLYB<4:0>	DELAY ON YB
DLYC<4:0>	DELAY ON YC
VCOSEL<2:0>	3-BIT VCO GEAR CONTROL (4 FREQUENCY RANGES)

*Note: * OADIVHALF may only be used when you bypass the PLL core (i.e. OAMUX = 001) and the RC Oscillator (RCOSC) drives the CLKA input.*

Static Clock with Divider and/or Delay

The Combiner is able to combine the clock conditioning circuit macro with the regular CLKBUF macros and the PLL to utilize available unused globals (as shown in Figure 1-3).

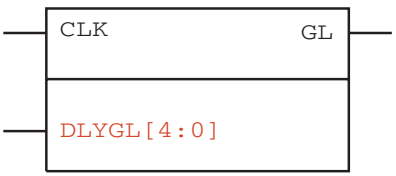
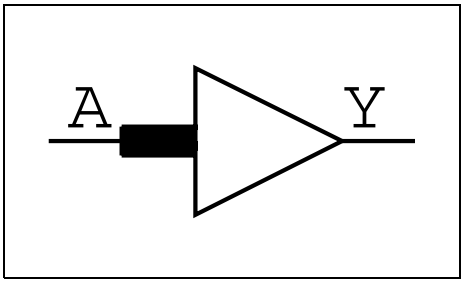


Figure 1-3. Combiner in Fusion

The CLKDLY is essentially a CLKBUF with a delay. The PLLINT macro is included to unambiguously show Designer which routing resources are required to connect the REFCLK input: The PLLINT is used when REFCLK is driven by a pad in a different I/O tile.

PLLINT	Fusion, ProASIC3, ProASIC3E



Function
PLL Int

Truth Table

A	Y
0	0
1	1

Input	Output
A	Y

Connect only to the REFCLK input of PLL when the PLL is driven by a pad other than the one in the same super cluster.

Refer to the latest ProASIC / ProASIC3E datasheets for more information on PLLs. They are available at http://www.actel.com.

Clock Conditioning / PLL Core Restrictions in SmartGen

After you make all your selections, SmartGen 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 (fin/n) must be between 1.5 and 5.5 MHz. The PLL Core output must be between 24 and 350 (fin *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, SmartGen tries to generate a value that is as close as possible to the value you requested. The actual values that SmartGen 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.

Figure 1-4 shows a sample of the SmartGen log file containing all the information.

The Fusion PLL has 2 new ports and PC bits. Because of these changes a ProASIC3/E design with a PLL cannot be ported directly on the Fusion device. They must be updated with the new PLL interface.

The 2 new ports are:

1. OADIVHALF: The PC bit which can be either VCC/GND. To be used along with the input coming from the RC oscillator.

2. OADIVRST: To reset the division CLKA. (Fusion PLL allows setting of independent resets for each clock. It is only used in the bypass model.)

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.
```

Figure 1-4. Sample SmartGen Log File

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

Total Delays

SmartGen 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 ñ 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 ñ 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.

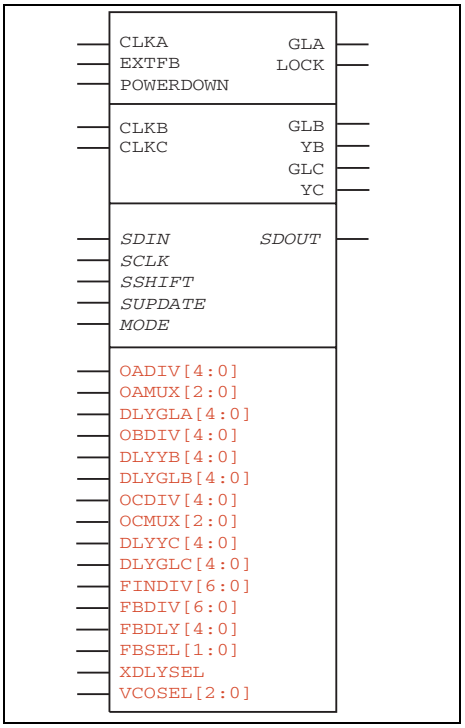
Output delay is the programmable delay independently selectable for each output.

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 SmartGen, 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.

DYNCCC	ProASIC3, ProASIC3E



Function

Dynamic PLL / Clock Conditioning Circuitry

Actel recommends that you use SmartGen to generate your DYNCCCs; SmartGen calculates the settings for all the pins in the DYNCCC for the required input-output frequency combinations.

Refer to the latest Actel datasheets on PLLs for ProASIC3 / ProASIC3E for more information. They are available at http://www.actel.com.

Inputs / Outputs

See the description below for an explanation of the inputs and outputs available on the Dynamic PLL for ProASIC3/E; all inputs in the diagram are shown on the left, and outputs are to the right.

In the symbol above, all the required user-accessible inputs and outputs are above the top horizontal line. Inputs or outputs below the top line are optional. The elements below the second line (SDIN, SCLK, SDOUT, etc.) are associated with the dynamic shift register. The static configuration inputs are below the third line. You can only connect static configuration pins to GND or VCC.

There are seventy-four static configuration ports on the DYNCC model, but there are eighty dynamic configuration bits that may be shifted in. The extra bits control the STATSEL and DYNSEL input multiplexors, a silicon feature that is not reflected in the simulation model. These extra bits are ignored by the simulation model.

Please note the following during dynamic configuration:

- When the “MODE” pin is at logic high, the PLL control bits are the bits from the configuration register. Before “MODE” is switched from logic low to logic high, the desired dynamic configuration bits should already be shifted-in.
- Reconfiguration does not unlock the PLL as long as the new configuration does not change the state of input divider or feedback elements.

No-Glitch MUX (NGMUX)

In Fusion mixed-signal FPGA, there are two No-Glitch MUXes (NGMUX, shown in Figure 1-5) that are positioned downstream from the PLL/CCC blocks. The NGMUX provides a special switching sequence between two asynchronous clock domains, which avoids generating any unwanted narrow clock pulses. It switches between two different clock sources and the output goes to a global network.

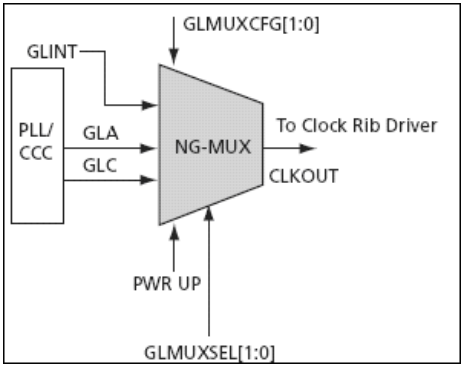


Figure 1-5. NGMUX

The allowable inputs are GLA, GLC, GLINT, or internal nets. There are some restrictions on these signals, which are described in “Using the NGMUX” on page 41. The software handles most of the configuration settings. You do not have access to these pins, for example, PWRUP and GLMUXCFG[1:0].

Mode of Operation

The signals driving NGMUX have same specification and the output clock of PLL/CCC. The following examples show various scenarios of the switching sequence between two asynchronous clock domains CLK0 and CLK1:

Case 1

When both CLK0 and CLK1 inputs to the NGMUX are active, the sequence of switching between two clock sources (from CLK0 to CLK1) is as follows:

1. SEL transitions to initiate a switch.
2. GL drives one last complete CLK0 positive pulse (i.e., one rising edge followed by one falling edge).
3. GL stays low until the second rising edge of CLK1 occurs.
4. At the second CLK1 rising edge, GL continuously delivers CLK1 signals.

Min tsw = 0.05 ns, 25°C (typical condition); an example is shown in Figure 1-6.

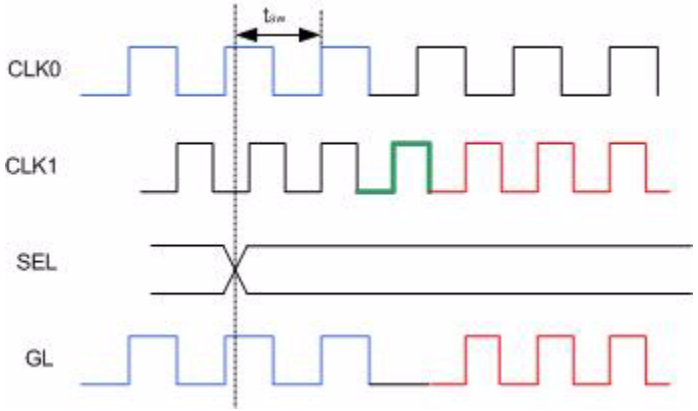


Figure 1-6. Example Showing Active CLK0 and CLK1 Inputs to NGMUX

Case 2

If CLK0 stops or runs at a very low frequency after SEL transition, the Timeout circuitry inside the FPGA is used and the sequence of switching between the two clock sources (from CLK0 to CLK1) is as shown in Figure 1-7.

Case 2A

If no rising CLK0 edge occurs before the 7th CLK1 rising edge:

1. At the 7th CLK1 rising edge, GL will go low until 9th CLK1 rising edge.
2. At the 9th CLK1 rising edge, GL will continuously deliver CLK1 signals.

3. At the 9th CLK2 rising edge, CLKOUT will continuously deliver CLK2 signals.

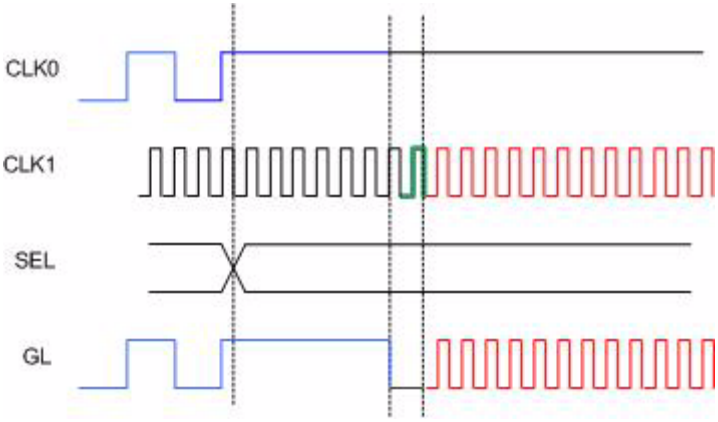


Figure 1-7. Case 2A - Low Frequency CLK0 After SEL Transition, No Rising CLK0 Edge Before 7th Rising CLK1 Edge

Case 2B

If a CLK0 rising edge occurs before the 7th CLK1 rising edge but no CLK0 falling edge occurs before the 15th CLK1 rising edge:

1. At the 15th CLK1 rising edge, GL will go low until the 17th CLK1 rising edge.
2. At the 17th CLK1 rising edge, GL will continuously deliver CLK1 signals (as shown in Figure 1-8).

Please note that you must wait till NGMUX switching sequence is completed. The output of the NGMUX is undefined if SEL switches before the previous switch operation has completed.

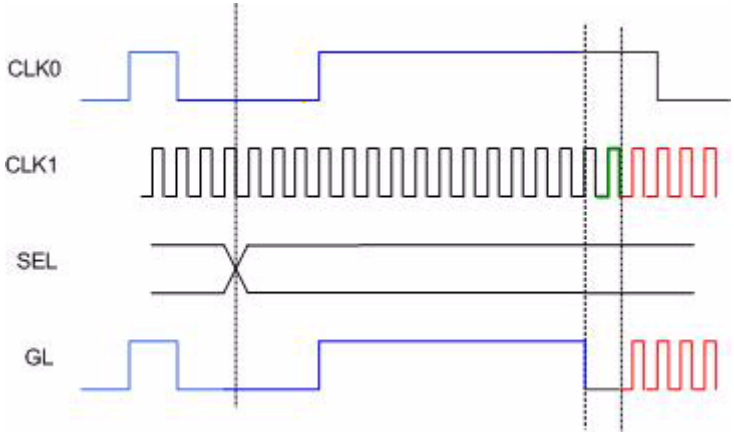


Figure 1-8. Case 2B - Low Frequency CLK0 After SEL Transition, Rising CLK0 Edge Before 7th Rising CLK1 Edge

Using the NGMUX

In software implementation, the NGMUX macro has been simplified to a 2:1 mux. The two inputs ports are CLK0 and CLK1 and the output is GL.

The allowable inputs to NG-MUX are:

1. GLA and GLC output of PLL
2. GLA and GLINT (fanout of GLINT needs to be 1).
3. GLA and internal nets.

You can instantiate NGMUX macro inside the VHDL or Verilog code to use the No-Glitch MUXes. Also, viewdraw symbol is available to add this macro in the schematic based design.

An example showing NGMUX macro in the VHDL code is shown below:

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

entity top is
  ……..
```

```
end top;

architecture DEF_ARCH of top is

  component NGMUX
    port(CLK0, CLK1, S : in std_logic := 'U'; GL : out std_logic);

  end component;
……
    I_ngmux : NGMUX
      port map(CLK0 => GLA, CLK1 => GLC, S => Ngmux_sel,  GL =>
Ngmux_clk);
…….
```

Tips and Tricks

Actel recommends the following when using NGMUX:

- NGMUX has a fixed location and is placed downstream from the PLL.
- Hard-wire the input CLK0 to GLA.
- NGMUX output uses GLA global network.
- If the two inputs to the NGMUX are GLA and GLC, then you may lose global network drive for GLC. Since the global network in fusion is segmented, you can use local clock network although the whole global network is not available.
- Since the NGMUX macro has fixed location (located downstream from the PLL), you need to address the routing delay when the input to NGMUX is coming from regular net.
- When GLC or GLINT drives the input port CLK1 of NG-MUX, the **net should have only one fanout**. Internal net driving NG_MUX does not have this restriction.

Timing Analysis

Timing analysis verifies the functionality of the design with the timing information. To check the functionality in the design for NGMUX, you need to check both static and dynamic timing analysis.

Static Timing Analysis

For static timing analysis, run timing analysis on both clock inputs separately using SmartTime. Run setup and hold check on the source pin of CLK0 and CLK1.

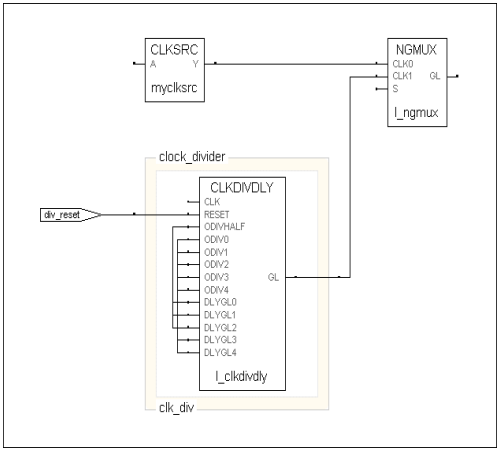


Figure 1-9. Static Timing Analysis

Figure 1-9 shows how the two inputs to NGMUX are connected from I_clkdivdly and myclksrc macros. In this instance, open SmartTime and check the setup and hold for both clocks: I_clkdivdly:GL and myclksrc:Y (as shown in Figure 1-10).

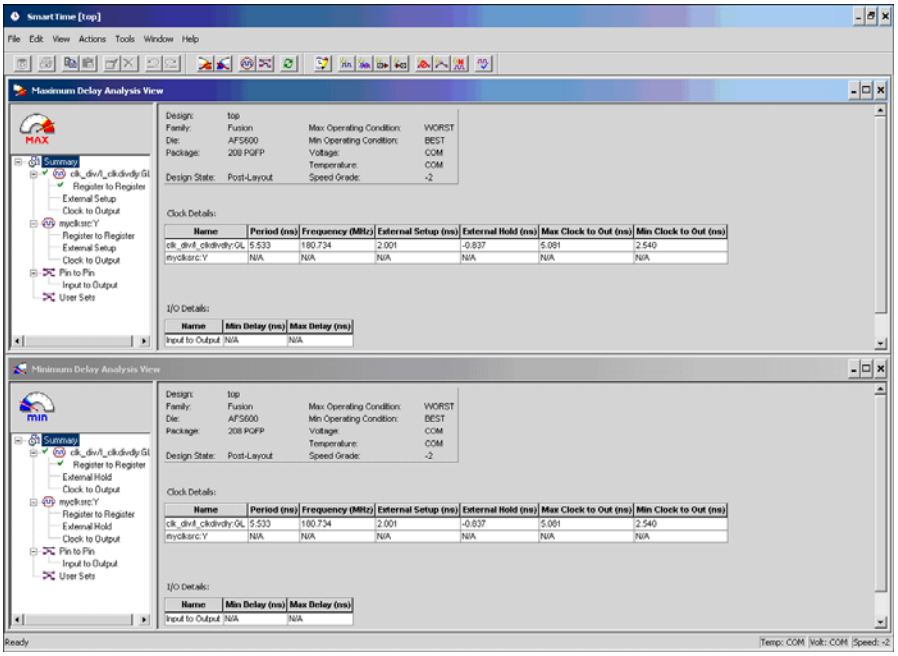


Figure 1-10. Checking Setup and Hold for Clocks

Dynamic Timing Analysis

For dynamic timing analysis, you need to run simulation using modelsim tool and check the NGMUX signals, as shown Figure 1-11.

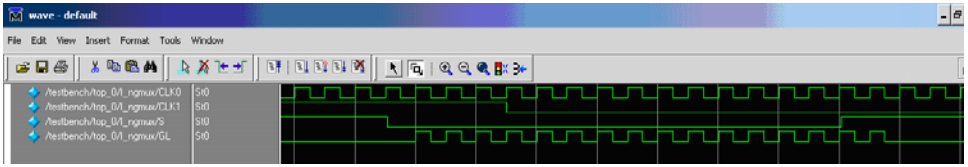
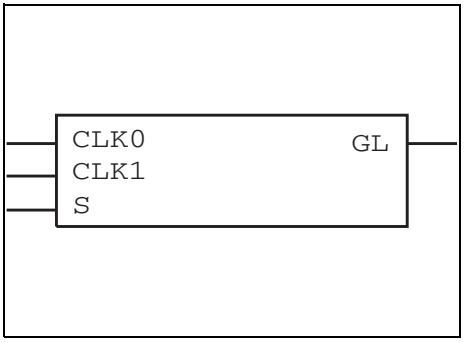


Figure 1-11. NGMUX Signals

NGMUX	Fusion



Function

No-glitch MUX. NGMUX provides a special switching sequence between two asynchronous clock domains to avoid generating any unwanted narrow clock pulses.

Refer to the Clocking Resources of the Fusion datasheet for more information on this macro. It is available at http://www.actel.com.

Input	Output
CLK0, CLK1, S	GL

Transition S from high to low to initiate a switch to CLK0, and from low to high to initiate a switch to CLK1.

The output of NGMUX is undefined if S switches again before the previous switch operation has completed.

Sub-block Interconnection: NGMUX

The No Glitch Mux is a new macro in Fusion.

It resides in the TILE5 of the center CCC. The NGMUX has 2 input ports which are connected to clock signals

Connect CLK0 port to a net which is driven by a clock signal. The fanout of the net connecting the CCC-GLA to NGMUX should always be 1.

The driver of this clock signal will be placed automatically in the GLA tile for the CCC location in which NGMUX is placed.

There are 2 possible connections for CLK1 port of NGMUX

1. The CLK1 port can be driven by a clock signal. The fanout of the net connecting the CCC-GLC should be 1.

   The driver will automatically be placed in the GLC tile for the CCC location in which NGMUX is placed.

2. CLK1 can also be driven by a routed net, in this case there is no restriction on the placement of the logic driving CLK1 and there is no restriction on the fanout of the net driving CLK1.

The Fusion oscillators can not drive the NGMUX directly as they don't produce the clock signals. Users must connect them to NGMUX inputs through a valid clock macro. (See section 3.1).

Note: NGMUX macros are automatically placed during layout.

In you want to place NGMUX manually, you must:

- *Place NGMUX*: Chooses one of the two available locations for NGMUX. The tile 5 of the central CCC locations. (Shown in yellow in Figure 1-12)
- *Place the driver for CLK0*: The driver for CLK0 has to be a CCC macro that can be placed in the CCC-GLA tile (shown in green in Figure 1-12). The CCC macros that can be placed here are:

  - 'CLKBUF' (Only non-vref versions as east and west banks do not support VREF I/O for Fusion)
  - 'CLKBIBUF'
  - 'CLKSRC'
  - 'CLKDLY'
  - 'CLKDIVDLY'
  - 'PLL'
- *Place the driver for CLK1:* If the driver for CLK1 is a CCC macro, it infers a hard-wired connection. This macro must be placed in the CCC-GLC tile (shown in purple in Figure 1-12)

  - 'CLKBUF' (Only non-vref versions as east and west banks do not support VREF I/O for Fusion)
  - 'CLKBIBUF'
  - 'CLKSRC'
  - 'CLKDLY'
  - 'CLKDIVDLY'
  - 'CLKDIVDLY1'
  - 'PLL'

When the CCC macro is driven from the hardwired I/O, placing the I/O controls the placement of the CCC macro.

When the CLK1 port is driven by the GLC port of some PLL instance, the GLA of the same PLL instance must become the driver for CLK0.

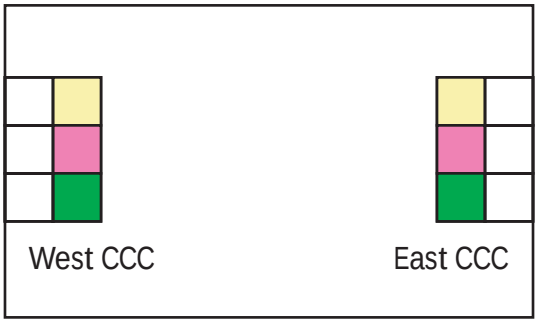


Figure 1-12. NGMUX Subblock Interconnects

2
Real Time Counter

The addition of the Real-time counter (RTC) enables Fusion devices to support both standby and sleep modes of operation, greatly reducing power consumption in many applications.

The Real-time counter system is comprised of six blocks that work together to provide this increased functionality.

- Real-time counter (RTC) - Configured in SmartGen.
- Crystal oscillator - If you use the RTC then the Crystal Oscillator Mode is controlled by the RTC.
- VCC33UP detector
- Voltage regulator initialization
- Voltage regulator logic
- 1.5 V voltage regulator

The Voltage Regulator and Power Supply Monitor (VRPSM) controls the VCC detector and voltage regulators.

This section describes the definition, configuration and control of these blocks. In addition, it provides some use cases for the RTC.

For silicon details for the RTC and related components, refer to the Fusion datasheet (http://www.actel.com/techdocs/ds/default.aspx).

RTC System Builder

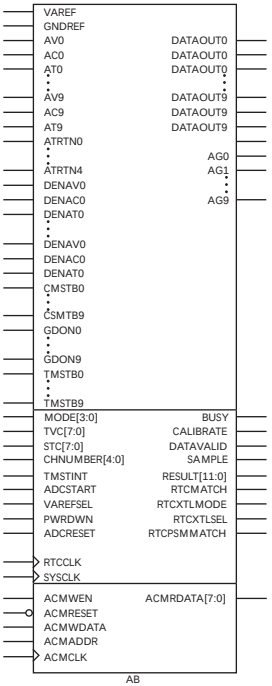
Real Time Counter (RTC) Signals

Table 2-1. RTC Signals in Analog System Builder

Name	Type	Description
RTCCLK	Input	RTC Clock. Must be driven by the XTLOUT of the XTLOSC macro.
RTCXTLSEL	Output	Crystal Oscillator Select. Must be connected to XTLSEL on XTLOSC macro.
RTCXTLMODE[1:0]	Output	Crystal Oscillator Mode Select. Must be connected to RTCXTLMODE on XTLOSC macro.
RTCMATCH	Output	Result of Match
RTCPSMMATCH	Output	Result of Match. Must be connected to RTCMATCH of VRPSM macro. Exposed only if On Match Enable Voltage Regulator is checked.
ACMCLK	Input	Clock for writing to ACM. Used with INIT_ADDR and INIT_DATA to update the RTC Match Register and Counter. Exposed only if Enable Dynamic Update is checked.
ACMRDATA[7:0]	Output	ACM Read Data Bus. Used with ACMCLK and INIT_ADDR to read the contents of ACM. Exposed only if Enable Dynamic Update is checked.

RTC System Macros

Analog Block	Fusion



Function

Analog system builder for use with SmartGen and Fusion. See the Fusion datasheet for a thorough description of the Analog System Builder.

Inputs / Outputs

Inputs are listed on the left, outputs on the right.

See the description below for an explanation of the inputs and outputs available on the Analog System Builder. For a complete description of the features in the ASB, see the Fusion datasheet.

The Analog System Builder in SmartGen enables you to configure an entire analog system including specifying the RTC settings.

The Analog System Builder enables you to create, configure, and place the Real Time Counter

Analog System Builder Main Window

The Analog System Builder main window enables you to create and configure your analog system (as shown in Figure 2-1).

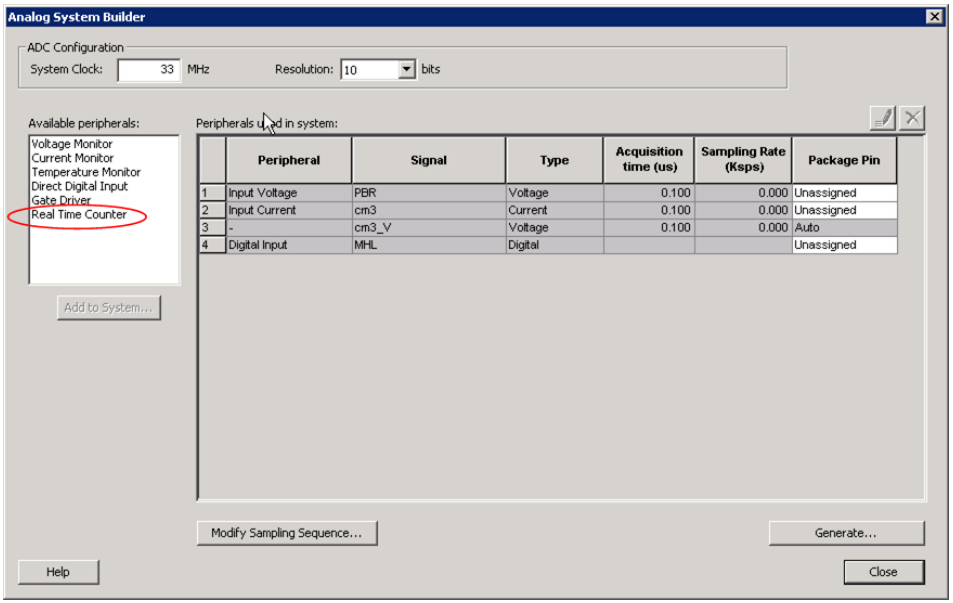


Figure 2-1. Analog System Builder Dialog Box

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

The Analog to Digital Converter (ADC) Configuration sets your System Clock speed and resolution.

Available Peripherals lists all the analog peripherals you can add to your design. As you add peripherals, some resources are exhausted (used up). The Analog System Builder disables peripherals in the list for which there are insufficient resources. 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 per a given input channel. SmartGen takes the required acquisition times for all peripherals and computes the ADC clock frequency and the

number of ADC clocks per sample, per peripheral, such that each peripheral meets or exceeds your required acquisition time.

- Sampling Rate (in µs) - The rate at which a given channel gets sampled by the ADC.
- Real Time Counter - You can configure the Real Time Counter so that it functions as a chronometer, configure it to generate periodic alarms in conjunction with other peripherals (such as the Voltage monitor), etc.
- Package Pin - SmartGen automatically assigns a package pin for each channel in each peripheral added to the system. 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.

  If your Analog System resources you build on your device exceed the total system resources available for your device, SmartGen 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 create generate a system with port name conflicts.

  When you click **Generate the system**, SmartGen creates HDL source files, memory (MEM) files, configuration files, and log files. They all appear in your SmartGen 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.

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 other the system clock networks, both on-chip and off. 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.

Figure 2-2 on page 54 shows the Real Time Counter dialog box.

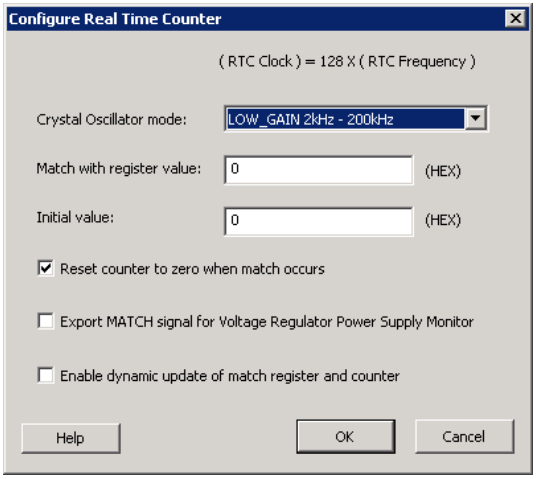


Figure 2-2. Real Time Counter Dialog Box

Crystal Oscillator Mode

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

LOW_GAIN / MEDIUM_GAIN / HIGH_GAIN - Oscillator is configured to support an external crystal or ceramic resonator. The difference is the supported crystal frequency or resonator, as shown in Table 2-2.

Table 2-2. Recommended Capacitor for Mode and Frequency

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

Match with Register Value

The MATCH signal asserts when it is equal to this value; this is a 40 bit binary or a 10 bit hex value.

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.

Reset count to zero when match occurs- Resets the counter 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).

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

Voltage Regulator (VR Logic)

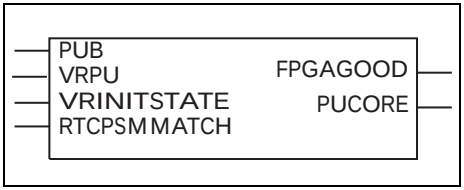
For details on Voltage Regulator, refer to the Fusion datasheet (http://www.actel.com/techdocs/ds/default.aspx).

1.5 V Voltage Regulator

For details on 1.5 Voltage Regulator, refer to the Fusion datasheet (http://www.actel.com/techdocs/ds/default.aspx).

Voltage Regulator and Power Supply Monitor (VRPSM)

Voltage Regulator and Power Supply Monitor (VRPSM)	Fusion



Function

The Voltage Regulator and Power Supply Monitor were combined into one macro because the VR and power supply logic work together to control the power-up state of the FPGA core.

Inputs / Outputs

Inputs are listed on the left, outputs on the right.

See the VRPSM signal description below for an explanation of the inputs and outputs.

The VR generates a 1.5 V power supply (500 mA max) from the 3.3 V power supply. The 1.5 V output is intended to supply all 1.5 V needs of the Fusion product. This regulator requires an external bipolar pass transistor. Enable for this block is generated in the VR logic block, or from an external pad.

The 1.5 V is not supplied internally to the Fusion device. It must be routed externally to the VCC pins on the device. Therefore the user is not required to use the V-Reg and can use an off-chip 1.5 V supply if desired.

The VRPSM can be enabled from several sources: the PUB pin, RTCMATCH signal from the Analog Block's RTC, or triggered by the PUP0 (RTINIT1 and RTINIT1, PC bits). In the simulation library, PUP0 is represented by VRINITSTATE. VRINITSTATE is FPGAGOOD initial power-up value. It enables you to drive FPGAGOOD to '1' or '0', before the 3.3 V is up. The PUCORE output is the Power-Up Bar (PUB) input inverted.

Once triggered the VRPSM remains on because of the latching functions of RS flip-flops. Only the FPGA fabric can reset these flip-flops and turn off the VRPSM. Once the FPGAGOOD signal is established, this VRPSM enable mechanism is no longer active. See the tables below for signal descriptions and recommended power-up sequences.

VRPSM Signal Description and Power-Up Sequences

The signals for the VPRSM macro are listed in Table 2-3. The PUB input comes from the PUB pin on the device and can be pulled high by a signal external to the Fusion device. This can be used to wake up from a standby condition. The inputs VRINITSTATE and RTCPSMMTACH come from the VR Init and RTC blocks respectively and either can initiate a VR power up.

Table 2-3. Signals for the VPRSM Macro

NAME	Number of Bits	Direction	FUNCTION
PUB	1	INPUT	Power-up bar
VRPU	1	INPUT	Voltage regulator power-up
VRINITSTATE	1	INPUT	FPGAGOOD initial value (set by 2 flash bits in the FPGA)
RTCPSMMATCH	1	INPUT	Connected to RTCMATCH signal from RTC
FPGAGOOD	1	OUTPUT	Indicates that the FPGA is logically functional
PUCORE	1	OUTPUT	Power-up to core

Recommended power-up sequences are listed in Table 2-4. ? indicates a don't care value.

Table 2-4. Recommended Power-Up Sequences

	PUB	VRINITSTATE	VRPU	RTCPSMMATCH	FPGAGOOD
Initial power-up	?	1	?	?	1
	1	0	0	0	0
	0	?	0	0	1
	1	?	1	0	1
	1	?	0	1	1

Table 2-4. Recommended Power-Up Sequences

	PUB	VRINITSTATE	VRPU	RTCPSMMATCH	FPGAGOOD
Sequence	1	?	0	0	0
	0	?	0	0	1
	1	?	1	0	1
	1	?	0	0	0
Sequence	1	?	0	0	0
	1	?	0	1	1
	1	?	1	0	1
	1	?	0	0	0

Functional Description

The Fusion datasheet, available at http://www.actel.com/techdocs/ds/default.aspx contains a detailed functional description of the entire VRPSM and its uses.

This macro is used for the Voltage regulator and Power System Monitor functionality.

- VRPSM can be used by itself without an Analog Block.
- If RTC functionality is needed, then VRPSM needs to be connected to AB with the connections as mentioned in section 2.5
- VRPSM has an input port 'PUB' which has a 'PAD' property. This port must be hooked to a top level design port.

RTC System Use Cases

Do Not Power-Up Using RTC

In this configuration, the RTC is not used to control the 1.5 V voltage regulator. This use case only uses the RTC to create time-matching events and/or using the 40-bit counter to keep track of elapsed time via reading/writing over the ACM interface within the FPGA fabric (such as using Core8051, other microcontrollers, or custom user logic). To meet these requirements:

1. The rtm_rst bit of the Control/Status register must be logic 0 and rstb_cnt bit must be 1.
2. The vr_en_mat bit of the Control/Status register must be logic 0.
3. The cntr_en bit of the Control/Status register must be logic 1.

Periodic Power-up/Power-Down Using VR

For this use case, whenever a match event occurs, the RTC controls power-up of the 1.5 V voltage regulator, and the voltage regulator is connected making use of an external pass transistor to provide the 1.5 V supply to the FPGA fabric. After the FPGA fabric has completed its power-up sequence, its custom logic or soft microcontroller (such as Core8051) takes the appropriate actions.

Similarly, the logic controls power-down of the voltage regulator, which turns the 1.5 V FPGA supply off. The 3.3 V supply must remain active, however, to keep the RTC actively counting for the next match event, which causes a repeated power-up/power-down sequence to occur. To meet these requirements:

1. The rtm_rst bit of the Control/Status register must be logic 0 and rstb_cnt bit must be 1.
2. The match register must be set to the period of time that will repeated for the power-up/power-down sequence.
3. The vr_en_mat bit of the Control/Status register must be logic 1.
4. The rst_cnt_omat bit of the Control/Status register must be logic 1.
5. The cntr_en bit of the Control/Status register must be logic 1.

Do Not Clear Counter on Match Events (elapsed time record)

In this configuration, the RTC is never cleared when a match event occurs. The MATCH signal still becomes active when the 40-bit counter equals the value of the 40-bit match register, but the counter continues counting past this value on its next clock cycle. After this occurs, you may wish to manually clear the counter via the rstb_cnt bit of the Control/Status register, or write a different value to the match register to cause further MATCH events. To meet these requirements:

1. The rtm_rst bit of the Control/Status register must be logic 0 and rstb_cnt bit must be 1.
2. The match register must be set to the count expected when start up is desired.

3. The vr_en_mat bit of the Control/Status register must be logic 1.
4. The rst_cnt_omat bit of the Control/Status register must be logic 0.
5. The cntr_en bit of the Control/Status register must be logic 1.

Cumulative Time Record

In this configuration, The RTC keeps track of "use time" of some meaningful variable, such as FPGA on time or FPGA off time. The RTC in this mode may or may not be used to control power-up at specific counter values. Rather, the cntr_en bit of the Control/Status register is set high whenever incrementing of the use time is desired. The value of the counter register can be periodically stored in the NVM memory to avoid losing all accumulated time if VCC33 is lost. The value can be restored into the counter once power is reapplied. To meet these requirements:

1. The rtm_rst bit of the Control/Status register must be logic 0 and rstb_cnt bit must be 1.
2. The cntr_en bit of the Control/Status register must be logic 1 only when counting is desired.

Start Counter from Non-Zero Value (referenced to some epoch)

The RTC may be used to start counting from a non-zero value. In order to accomplish this, you must write the reference count into the five bytes of the counter and enable counting from that point onward. The match register, in this case, may or may not be used to provide MATCH events. To meet these requirements:

1. The rtm_rst bit of the Control/Status register must be logic 0 and rstb_cnt bit must be 1.
2. The five bytes of the counter must be written with the reference count (time) for the user's application.
3. The cntr_en bit of the Control/Status register must be logic 1.

Tips and Tricks

Schematic Entry Pointers

- ViewDraw in Libero supports schematic design creation for all the new Fusion macros.
- The following macros have pins which are like PAD ports and need to be connected to top level design port. Connect a hierarchy connecter to the following ports in the schematic design

  - XTLOSC : XTL
  - VRPSM : PUB
  - AB : RTCPSMMATCH
  - VRINITSTATE tied to VCC - initial power-up state is ON or GND otherwise

Design Considerations

- System clock to Analog block should be less than 100 MHz, but the clock used for initialization of the RTC at power up must not exceed 10 MHz.
- When "Reset Counter to initial value when match occurs" option is *not* selected, the time interval between active RTCMATCH occurrences is equal to the total cumulative time count of 40-bit RTC counter. In other words, the counter should overflow and reach the MATCH value again to create an active RTCMATCH output. The amount of time required for the counter to overflow will not be practical for most applications; Actel recommends you use the "Reset Counter to initial value when match occurs" option, if you need to use RTCMATCH.
- Location of RTC registers are shown in Table 2-5.

Table 2-5. Description of Registers

ACM_ADDR[7:0]	Register Name	Brief Description
0x40	COUNTER0	Counter bits 7:0
0x41	COUNTER1	Counter bits 15:8
0x42	COUNTER2	Counter bits 23:16
0x43	COUNTER3	Counter bits 31:24
0x44	COUNTER4	Counter bits 39:32
0x48	MATCHREG0	Match Register bits 7:0
0x49	MATCHREG1	Match Register bits 15:8
0x4a	MATCHREG2	Match Register bits 23:16
0x4b	MATCHREG3	Match Register bits 31:24
0x4c	MATCHREG4	Match Register bits 39:32
0x50	MATCHBITS0	Individual Match bits 7:0
0x51	MATCHBITS1	Individual Match bits 15:8
0x52	MATCHBITS2	Individual Match bits 23:16
0x53	MATCHBITS3	Individual Match bits 31:24
0x54	MATCHBITS4	Individual Match bits 39:32

Table 2-5. Description of Registers

0x58	CTRL_STAT	Control (write) / Status (read) register bits 7:0
0x59	TEST_REG	Test register(s)

RTC interconnection

If any of these hardwired connections are not to be used then connect the unused input to GND and leave unused outputs dangling (Figure 2-3).

For all the hardwired connections, the fanout of net connecting the two hardwired pins must be 1.

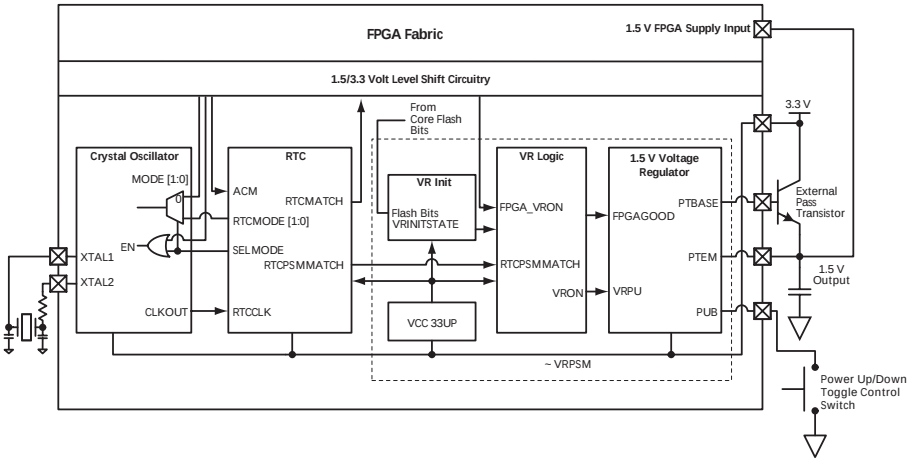


Figure 2-3. RTC Interconnect Diagram

3 Embedded Memory

The Fusion devices contain different types of Embedded Memories.

- Flash Memory Block - Configured with the Flash Memory System Builder (in SmartGen)
- FlashROM - Configured with UFROM
- SRAM - Blocks include RAM4K9 and RAM 512x8 from ProASIC3/E and new macro blocks FLEXRAM4K9 and FLEXRAM512X18
- FIFO - from ProASIC3E

These blocks are implemented directly in silicon and can be configured using a number of different techniques including either SmartGen or by hand instantiation of Macros.

Additional sections include:

- Tips and Tricks
- Sub-block interconnection

Flash Memory Block

Each FB holds 256 Kbytes of data. Functionally, it is similar to a large single-port synchronous RAM. The Fusion datasheet has a detailed description of the silicon implementation of the Flash Memory Block including:

- Flash Memory Block Diagram
- Flash Memory Block Addressing
- Flash Memory Block Protection
- Flash Memory Block Operations including program, erase, read and write.
- Flash Memory Block Timing Characteristics

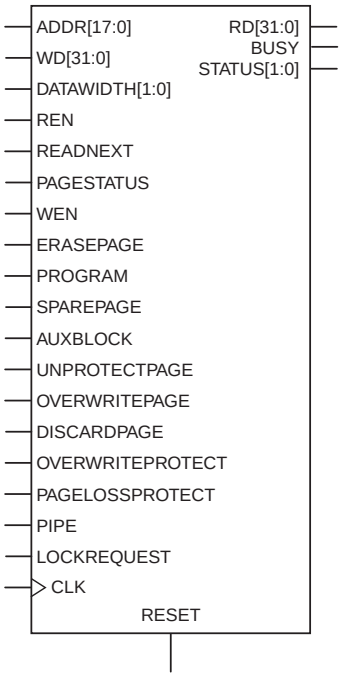
The Flash Memory Block section of the Peripherals User's Guide addresses the relationship between the silicon description and the building, configuration and data entry through software for specific applications.

The Flash Memory Block (when configured by SmartGen) includes the Flash Memory itself and control logic.

The control logic created and the number of tiles used depend on your applications.

You can use the Flash Memory to initialize the Analog Systems and also for RAM initialization and logic initialization. Your data can be saved during operation and used for initialization on next power up.

Flash Memory Block	Fusion



Function

Flash memory block builder for use with SmartGen and Fusion

Inputs / Outputs

See the description below for an explanation of the inputs and outputs available on the Flash memory block macro.

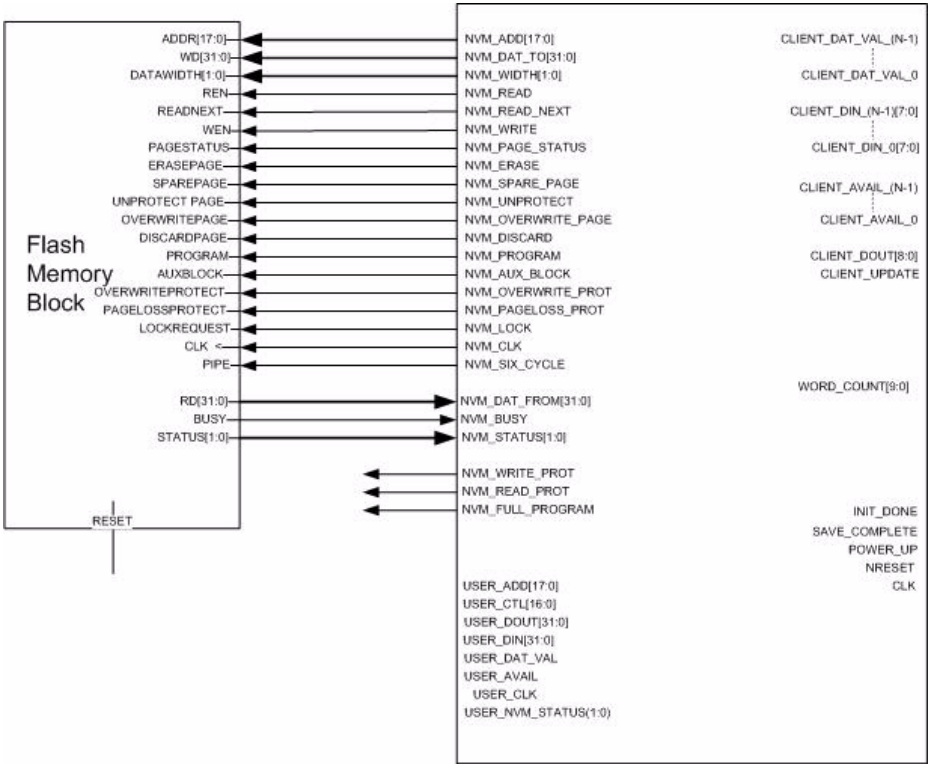


Figure 3-1. Flash Memory Block Interconnect Diagram

Each Flash Memory block holds 256 kb of data. Although it is functionally similar to a large single-port synchronous RAM, it has several significant differences, such as:

1. Address bits are MSB justified, unlike RAM4K9 and RAM512X18 in which the address bits are LSB justified.
2. Write operation updates write data into the block buffer ONLY. To store data permanently into the Flash Memory Block array writes to a page must be followed by a program operation of the same page.
3. The simulation models always execute copy page from Flash memory block array (internal operation) in 65 clock cycles; in silicon the behavior is non-deterministic (63-67 clock cycles). This mismatch is reflected in the number of cycles BUSY is asserted.

Operations on Flash memory block are synchronous to rising-edge of CLK.

Flash Memory Block Pin Description

All Flash memory block signals are active high, except for RESET which is active low. The Flash memory block is a completely synchronous model sensitive to rising edge of CLK input.

Table 3-1. Flash Memory Block Pin Descriptions

Name	Function
ADDR[17:0]	Byte-offset into the Flash memory block array or block buffer of page buffer
WD[31:0]	Write data
DATAWIDTH[31:0]	00 = 1-byte in data_in/out [7:0] 01 = 2-bytes in data_in/out [15:0] 10/11 = 4-bytes in data_in/out [31:0]
REN	When asserted, initiates a read operation
READNEXT	When asserted with REN, initiates a read from next address after read to current address is complete.
PAGESTATUS	When asserted with read, initiates a read page status operation
WEN	When asserted, interface data is stored into the assembly buffer.
ERASEPAGE	When asserted, erase addressed page (program all zeroes).
PROGRAM	When asserted, write the contents of the assembly buffer into the cell array page addressed.
SPAREPAGE	When asserted, the sector addressed is used to access the spare page within that sector.
AUXBLOCK	When asserted, the page addressed is used to access the auxiliary block within that page.
UNPROTECTPAGE	When asserted, the page addressed is copied into the AB and the AB made writeable.
OVERWRITEPAGE	When asserted, the page addressed is overwritten with the contents of the AB if the page is writeable.
DISCARDPAGE	When asserted, the contents of the AB are discarded so that a new page write can be started.
OVERWRITEPROTECT	When asserted, all program operations will set the overwrite protect bit in the auxiliary block of the page being programmed.
PAGELOSSPROTECT	When asserted, a modified assembly buffer must be programmed or discarded before accessing a new page.
PIPE	When asserted with REN, read operation completes in 6 cycles. Required to be asserted for CLK speeds above 50MHz.
LOCKREQUEST	Request to lock user access to Flash memory block array.
CLK	Input clock. All operations and status are synchronous to rising-edge of this clock.
RESET	When asserted resets the state of the Flash memory block.

Table 3-1. Flash Memory Block Pin Descriptions (Continued)

Name	Function
BUSY	When asserted, indicates that the FB is performing an operation.
RD[31:0]	Read data to be sampled when BUSY=0.
STATUS[1:0]	Status of the last operation completed: 00 = successful completion 01 = Read: single error detected and corrected Write: operation addressed a write-protected page Erase-Page/Program: AB is unmodified 10 = Read: two or more errors detected Erase-Page/Program: Compare operation failed 11 = Write: attempt to write to another page before Programming current page.

Simulation Details for Flash Memory Block

The Flash memory block array can be pre-loaded with user-defined data. For simulation purposes, you can specify a memory initialization file by over-riding the parameter "MEMORYFILE" in the Verilog netlist and the generic "MEMORYFILE" in Vital netlist.

The memory array declared in simulation models stores data that is one block wide. It is 64k x 140 bits. The addressing scheme for accessing this array consists of 16 bits, as shown in Figure 3-2.

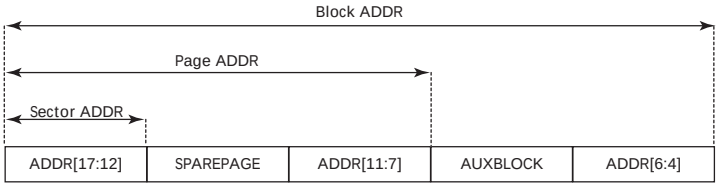


Figure 3-2. Addressing Scheme for Accessing the Flash Memory Block Memory Array

ADDR[17:0] is the Flash memory block interface address, SPAREPAGE and AUXBLOCK are input signals.

The memory file for pre-loading an Flash memory block array consists of "strings of address and data in hexadecimal notation with address delimiters (@)" and MUST conform to the rules shown below:

1. Each line MUST contain a string of fixed length (=35 characters) and start with an "@" if it corresponds to an address.

2. Each line following the address line corresponds to a block of data starting at the block address specified in the address line. This applies until the next line with an address specifier (@) is encountered.
3. Each data block consists of 35 hex chars. Hex[31:0] are the data characters corresponding to 16 bytes of user data with Hex[1:0] corresponding to Byte0 and Hex[31:30] corresponding to Byte15. Hex[34:32] are ECC related bits and must be addressed manually.

Based on these rules, the format looks like:

```
@Block_Address_0
Block_Data_0 ( required )
Block_Data_1 ( optional )
Block_Data_2 ( optional )
...
...
Block_Data_8  ( Aux block data for this page,  optional )
@Block_Address_n
Block_Data_n      ( required )
Block_Data_n+1 ( optional )
Block_Data_n+2 ( optional )
...
...
...
Block_Data_n+8 ( Aux block data for this page, optional )
```

A typical memory file looks like:

```
@000…0000  // beginning with @, start address in hex. format. 0s to be
padded
// between @ and hex address, to get a string of length 35.
ab101fd01…  // 35 hexadecimal characters corresponding to each block of
Flash memory block cell
eab9c4……
@000…4030  // start address for next data stream
c805489e….  // 35 hexadecimal characters corresponding to each block of
Flash memory block cell
96986391…
```

User Controlled Generics

FAST_SIM – The generic/parameter FAST_SIM is included in pre-synthesis and pre-layout simulation models to reduce cycles wasted in executing the PROGRAM operation. The default is '1', which means the PROGRAM operation is executed with a 4 ms simulated delay. You can chose to deactivate the operation by overriding FAST_SIM to 0, in which case PROGRAM is executed with a delay close to real time of 8.4 ms. You can also choose this mode for post-layout simulations.

WR_THR – When the number of writes to a page in the Flash memory block array (program operation) exceeds the write threshold specified in the data sheet, the status returned is non-zero. Since the threshold is a huge value, WR_THR is provided to simulate this failure at a reduced number of writes. You can override the generic with any non-zero count (such as 10 or 12). The write threshold exceeded condition is produced on the 10th write to the same page in the Flash memory block array.

Note: The string assigned to the generic/parameter MEMORYFILE is preserved through synthesis and place-and-route. But for FAST_SIM and WR_THR generics/parameters, you must reassign the desired value each time after synthesis and place-and-route.

Flash Memory System Builder

The Flash Memory System Builder in SmartGen (Figure 3-3) enables you to configure the entire flash memory block.

When an Analog Client is added through the analog system builder a NCF file is created that can be imported to the Flash Memory System Builder, to assist in the correct configuration of the matching memory required for that client.

This also applies to other clients that require associated memory.

Using Flash Memory System Builder 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

In order to use the Flash Memory System Builder you must first create the components that you intend to initialize for example Analog System or RAM block, in order to import these configurations to the Analog System Builder.

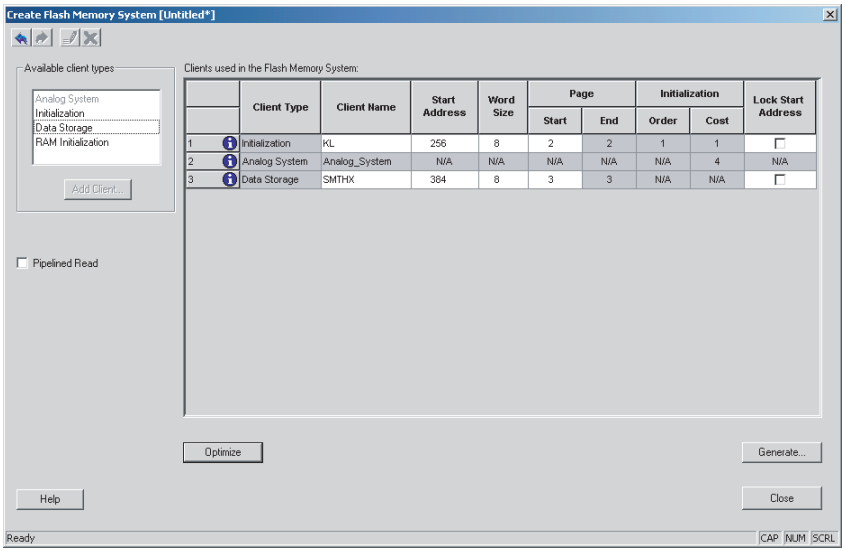


Figure 3-3. Flash Memory System Builder

The Flash Memory Block Builder supports the following clients:

- Analog System
- Initialization
- Data Storage
- RAM Initialization

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 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 address at which the client starts. It must be on a page boundary. Clients cannot have overlapping start addresses.

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, 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 of the system cannot exceed 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 address locked. It also de-fragments 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 (Figure 3-4). Click the **Browse** button to navigate to your configuration file.

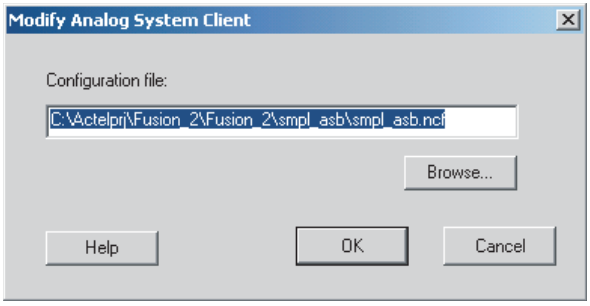


Figure 3-4. Add Analog System Client Dialog Box

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.

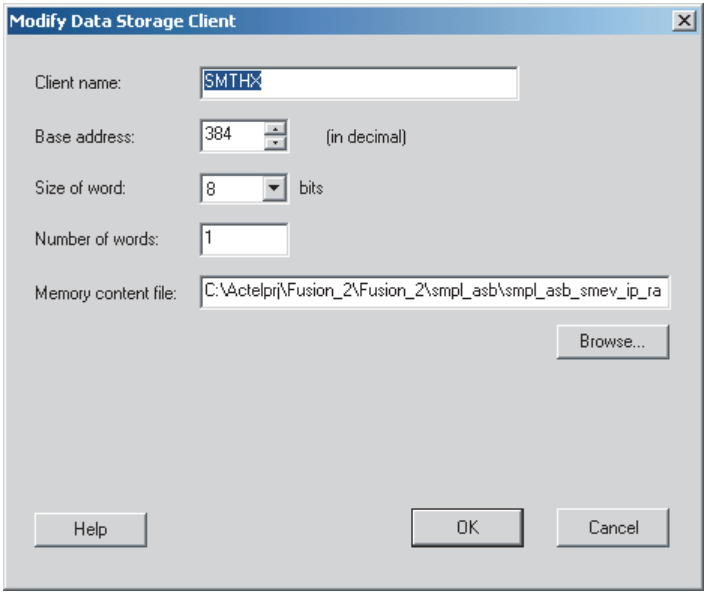


Figure 3-5. Add Data Storage Client Dialog Box

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 together.

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

Number of Words – Can be anywhere from 1-262144 for 8 bit words. For 16 bit words, the maximum number available is half of 8-bit words; for 32-bit words the maximum number available is one fourth of 8-bit words.

Memory Content files – The content for the Flash Memory Block to initialize this client. Must be specified in one of the supported memory formats.

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 raise the power 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 address, etc.) and ROM emulation.

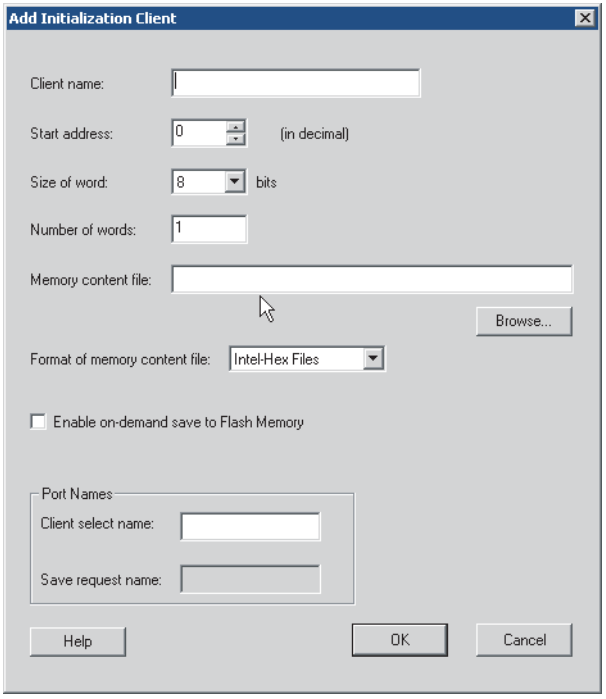


Figure 3-6. Add Initialization Client Dialog Box

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 together.

Start address – Starting address for the initialization client.

Size of word – In bits, the word size of the initialized client. Can be either 8 or 9. On a 9-bit word size, each word takes 2 bytes of Flash Memory System data.

Number of Words – May be anywhere from 1-262144 for 8-bit words. For 9-bit words, the maximum number available is half the 8-bit maximum.

Memory Content file – The content for the Flash Memory System Builder to initialize this client. Can be specified in one of the supported memory formats.

Enable on demand save to Flash Memory – Enables the content of Flash Memory System to be stored back in the Flash Memory. For timing diagrams and signals operations, refer to Save data to FMS in the SmartGen Cores Reference Guide.

Client Select and Save Request port names – Port names for the client chip select for initialization and save request for save-back. The port names are prefixed with the client name to group control signals together.

RAM Initialization Client

The RAM Initialization client is a special type of Initialization client. When you generate a RAM with Initialization from SmartGen 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.

You can also specify a start address and lock the start address if you wish to always use this address for initializing this client. You must create the RAM component first and then create the Initialization Client.

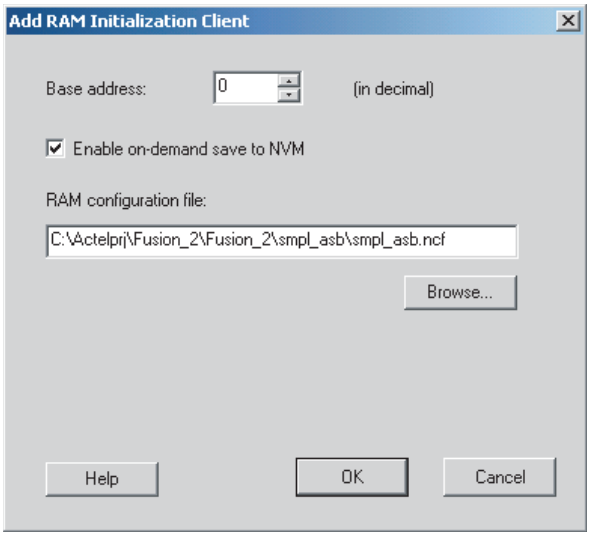


Figure 3-7. RAM Initialization Client Dialog Box

Flash Memory System Output Files (from SmartGen)

When you click **Generate the complete system**, SmartGen generates the following files; they are saved in your SmartGen project 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

<Vhdl>/<Verilog>/address_gen.vhd/v - Soft IP

<Vhdl>/<Verilog>/init_sm.vhd/v – Soft IP

<Vhdl>/<Verilog>/nvm_ctl.vhd/v – Soft IP

<Vhdl>/<Verilog>/save_sm.vhd/v – Soft IP

<Vhdl>/<Verilog>/user_clk_sel.vhd/v -Soft IP

<Vhdl>/<Verilog>/user_ctl.vhd/v – Soft IP

<Vhdl>/<Verilog>/user_valid.vhd/v – Soft IP

<Vhdl>/<Verilog>/numbits.vhd/v – package file

Memory Files

<user_name>.mem - Non-volatile memory file

Configuration Files

<user_name>.cfg – Captures information about the settings that were specified for the system.

<user_name>.gen – SmartGen project file; stores information about your Flash Memory System so that you can save your settings.

<user_name>.cxf – SmartGen core configuration file that contains information required by Libero IDE for file management.

Log Files

<user_name>.log

Memory File Formats in Flash Memory System Builder

The following memory file formats are available in the Flash Memory System Builder.

Actel BINARY

The simplest memory format. Each memfile contains as many rows as there are number of words required. Each row is one word, where the number of binary digits equals the wordsize. 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.

MOTOROLA S

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

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:DDDDD
```

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

```
00:FF
11:BB
```

All other addresses will be zeroes.

Flash Memory System Output

System Level Signals

Table 3-2. System Level Signals in Flash Memory Block Builder

Name	Type	Description	Connecting silicon names to SmartGen
INIT_CLK	Input	Initialization clock. Must be less than 10 MHz	CLK
SYS_RESET	Input	Active Low System Reset	NRESET
INIT_POWER_UP	Input	Indicates the system is ready for Flash Memory Block initialization. Active High.	POWER_UP

Table 3-2. System Level Signals in Flash Memory Block Builder (Continued)

Name	Type	Description	Connecting silicon names to SmartGen
INIT_ADDR[MAX_ADDR-1:0]	Input	Used by client interfaces to generate address for write to RAM.	WORD_COUNT
INIT_DATA[8:0]	Output	Shared data bus used to initialize all the client RAMs as well as the ADC and ACM	CLIENT_DOUT[8:0]
SAVE_COMPLETE	Output	Indicates save operation defined by UPDATE_CLIENT has completed. Driven synchronous to USER_CLK	SAVE_COMPLETE
INIT_DONE	Output	Indicates initialization and clear functions are complete so that you can transition to your operations. Driven synchronous to USER_CLK	INIT_DONE

Analog System Initialization Signals

Table 3-3. Analog System Initialization Signals in Flash Memory Block Builder

Name	Type	Description
INIT_ACM_WEN	Input	Enable the ACM for Initialization from Flash memory block System
INIT_ASSC_WEN	Input	Enable the ASSC for Initialization from Flash Memory Block System
INIT_EV_WEN	Input	Enable the SMEV for Initialization from Flash Memory Block System
INIT_TR_WEN	Input	Enable the SMTR for Initialization from Flash Memory Block System

Other Initialization/Save Client Signals

Table 3-4. Initialization/Save Client Signals in Analog System Builder

Name	Type	Description
<clientname>_DAT_VAL	Output	This is the chip select enable per client

Table 3-4. Initialization/Save Client Signals in Analog System Builder

Name	Type	Description
CLIENT_DIN[8:0]	Input	8-bit data bus from each client interface that will perform upload of data. Numbered from 0 to N-1
CLIENT_AVAIL[N-1:0]	Input	Used during write cycles to the Flash Memory Block. Indicates that the Client interface contains valid data. Numbered from 0 to N-1

Data Storage Client Signals

These signals are exposed when there is at least one data storage client. These are passed through controls via the Init & Config IP to directly manipulate the Flash Memory Block. They are identical to Flash Memory Block hard IP port names, except that they all have the prefix USER_.

Table 3-5. Data Storage Client Signals

Name	Type	Description
CLK	Input	Init clock: Low speed (1-10 MHz)
USER_CLK	Input	Clock for user application
USER_DATA[31:0]	Input	Data bus interface from the User interfaces. Size of bus is determined by the configuration data for the interface.
USER_ADD[17:0]	Input	Address bus interface for the User interfaces.

Table 3-5. Data Storage Client Signals (Continued)

Name	Type	Description
USER_* Control signals for the Flash Memory Block	Input	The control signals are a group of interface signals for each user interface, which mirror the Flash Memory Block Control signals. The details are found in the Flash Memory Block Requirements specification. The signals are: READ: Read Operation READ_NEXT: Read Next Operation WRITE: Write Operation ERASE: Erase operation PROGRAM: Program operation depending on the erase state of the page UNPROTECT_PAGE: Makes page contents writeable DISCARD_PAGE: Discards current write contents of AB WIDTH: 2 bit indicator of size of data bus SPARE_PAGE: Indicates address is for spare page of sector OVERWRITE_PROTECT: Enables overwrite protection that requires that UNPROTECT_PAGE be used to make pages writeable. PAGELOSS_PROTECT: Enables page-loss protection that requires that DISCARD_PAGE be used to work on a different page if the current page has been modified. Page_status: When initiated with a READ performs a page status operation OVERWRITE_PAGE: When initiated with a Program operation will perform a program of the current contents of Flash Memory Block assembly Buffer to the page address specified LOCK: When asserted pc_access should not be changed by PROGCTL SIX_CYCLE: When active the user is using six cycle access mode AUX_BLOCK: User interface for accessing the User Data field of the auxiliary block in the Flash Memory Block
USER_DOUT[31:0]	Output	Data Output bus for user interface.
USER_NVM_STATUS[1:0]	Output	Pass Through of NVM_STATUS to provide all necessary signals for the user interface
USER_DAT_VAL	Output	NVM_BUSY indication given to the user interface

Flash Memory Block Design Tips

The BUSY signal is one of the most important signals in the Flash Memory Block; all operations on the Flash Memory Block should be designed based on the BUSY signal.

The run time of different operations (Write/Program/Erase/Read) are shortened in the simulation model. The goal is to shorten to the simulation run time.

The BUSY signal is asserted whenever there is an operation going on, then de-asserted after the operation is done regardless if it is in simulation or in real silicon. You must design the application based on the BUSY signal assertion and de-assertion status instead of counting the operation cycles.

While the BUSY signal is asserted, if a new address value is presented, this timeframe can be counted as the set up time for the new address. The new address can be de-asserted right at the next valid clock rising edge after the BUSY signal is de-asserted because Flash Memory Block Write/Program/Erase/Read operations have 0 hold time requirement.

Reset Operation

Flash Memory System Reset is asynchronous and active low. It resets the content of the Flash Memory control logic block, such as the content of the Block Buffer and Page Buffer. Once the RESET is asserted, the BUSY signal is asserted. After the RESET is released, the BUSY signal will be de-asserted ~25us later. You should design the application to accommodate this period of BUSY timeframe by monitoring the BUSY signal status. Other operations can be executed once the BUSY signal is de-asserted.

Write/Program/Erase Operation

Write, Program, and Erase are called Data Storage operations. They are all page based operations.

In the Flash Memory Block a Write operation modifies the Page Buffer content, but does not necessarily trigger a Program operation. The content of the Page Buffer will be updated into the Flash Memory Block only when you execute a Program operation.

If you execute a Write operation for next page before executing a Program operation for the current page, the Page Buffer will lose the contents of the current page, and be overwritten by the next page content. To avoid this, you can turn on the PAGELOSSPROTECT option by connecting the PAGELOSSPROTECT pin to '1'. If you use this option, the Flash Memory System Builder returns an error upon the Write operation for the next page and the Write is halted. Actel recommends that the PAGELOSSPROTECT is always on.

Erase operation is similar to Write and Program. It just writes all 0s to the Page Buffer and Programs the Page Buffer contents into the Flash Memory Block. The PAGELOSSPROTECT option works the same way for erase operations as it does for write operations.

Erase/Write to the same page requires less cycles than Erase/Write to a new page.

Read Operation

Read operation is a block based. Whenever you start to read a new block or compare to read in the same block, extra cycles are used. Your application should monitor the BUSY signal to decide when to execute the next operation.

Overwrite Protection

Overwrite protection can prevent a specific page from being written by a mistaken operation. You can use overwriteprotect to protect all pages except the page that needs to be overwritten, to make sure the contents in the Page Buffer will not be written into any other pages.

To update the overwriteprotected pages, you have to de-assert the overwriteprotect bit for those pages first.

To turn on or off the overwrite protection, you can assign overwriteprotect signal to '1' or '0' respectively.

Dynamic Access

You can update the Flash Memory Block from FPGA fabric or from JTAG circuitry through Programming Control logic.

If FPGA fabric is updating the Flash Memory, a BUSY signal is sent to Programming Control logic. Once the updating is done, the BUSY signal is de-asserted. However, even when the BUSY signal is asserted, Programming Control logic has higher priority and it can still take over and start updating the Flash Memory through JTAG.

In order to avoid this type of conflict, you can assert the LOCKREQUEST signal to prevent JTAG access to the Flash Memory Block. Once the access through JTAG is done, or the application requires giving the authority back to the FPGA fabric, you should de-assert the LOCKREQUEST signal.

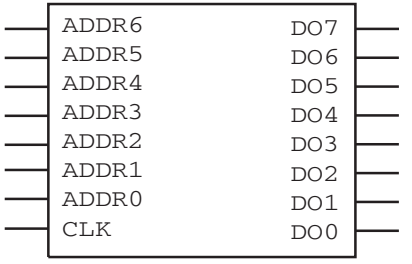
FlashROM

Fusion devices have 1 kbit of on-chip nonvolatile Flash memory that can be read from the FPGA core fabric. The FlashROM is arranged in 8 banks of 128 bits during programming. The 128 bits in each bank are addressable as 16 bytes during the read back of the FlashROM from the FPGA core (Figure 2-45).

You can read, modify, and write to the FlashROM using the JTAG interface; however, you can only read it from the FPGA core.

In addition the ability to read, modify and write to the FlashROM is controlled by the security settings of the device.

UFROM	Fusion, ProASIC3, ProASIC3E



Input	Output
ADDR[0:6], CLK	DO[7:0]

Function

The UFROM is the USER FlashROM macro. It is a simple 128 X 8 synchronous read-only memory. There is only one UFROM per chip. New data appears on the DO pins after the falling edge of the clock pin. The UFROM can only be programmed by the user via the JTAG pins. There is currently no support for programming the UFROM in any of the CAE tools or libraries, however the simulation models will utilize a memory initialization file so users can specify the contents of the memory for simulation purposes. The memory initialization file will be an ASCII format text file containing exactly 128 lines of 8-character binary strings.

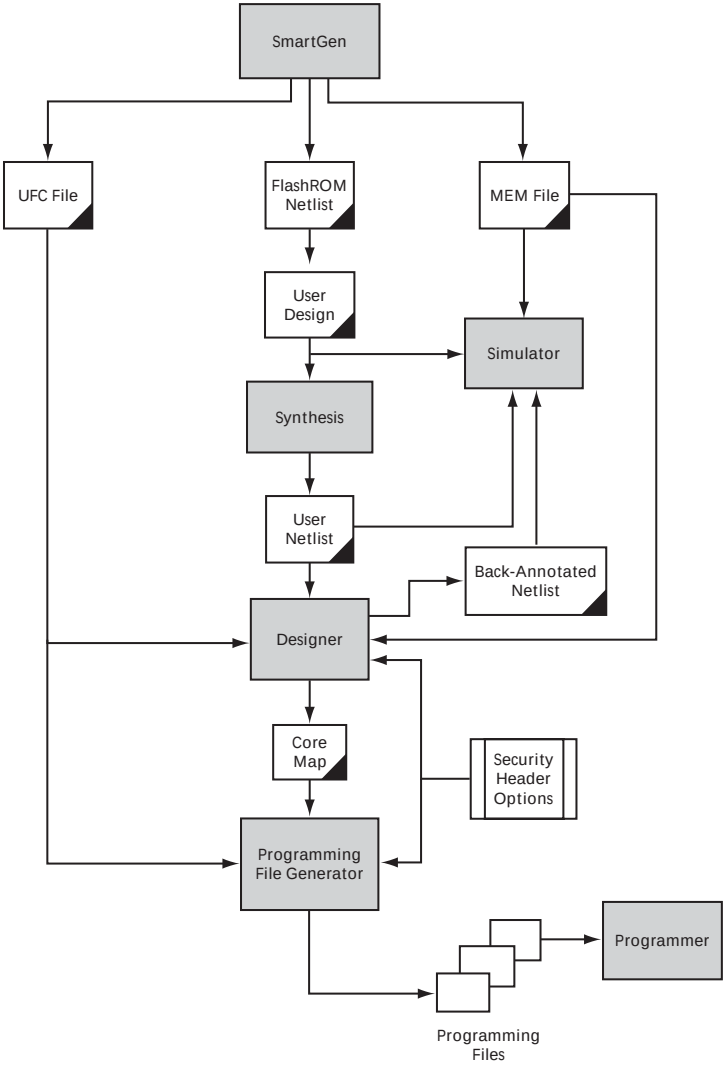


Figure 3-8. FlashROM Flow

Access FlashROM from the SmartGen tool. It opens a special FlashROM core generator that enables you to configure the FlashROM functionality.

Fusion 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 both 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) 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); Actel's Fusion and ProASIC3/E 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 SmartGen.

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

- SmartGen GEN file
- 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. SmartGen will default to 0s for any unspecified locations of the FlashROM memory. This file is used exclusively for simulation.

Use the FlashROM help to:

- Configure FlashROM in SmartGen
- 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 in SmartGen

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 SmartGen) 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 Properties field. The Properties field includes region-specific information. You can assign values to the following properties:

- **Static Fixed Data** – Enables you to fix the data so that it cannot be changed during programming time. This option is useful when the 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 Fixed Data** – Select this option when the data in a particular region is expected to be static data (such as a version number, which remains the same for a long duration, but could conceivably change in future). This option enables you to specify this region for this special purpose so that you need not come back and change the value every time you need to enter new data. The FlashROM configuration file can be handed off to the operations group at this point.
- **Read from File** – This provides the full flexibility of FlashROM usage to the customer. If you have a customized algorithm for generating the FlashROM data, you can specify this setting. You can then generate a file with data for as many devices as you wish to program and load that into FlashPoint programming file generation software to get programming files that include all the data. SmartGen optionally passes the location of the file where the data is stored, if you specify the file in SmartGen.

- **Auto Increment/Decrement** - This scenario is useful when you specify the contents of FlashROM for a large number of devices in a serial manner. You can specify the step value for the serial number and a maximum value for inventory control. During programming file generation, the actual number of devices to be programmed is specified and a start value is input to the software. Software generates the files to complete serial programming of the FlashROM.

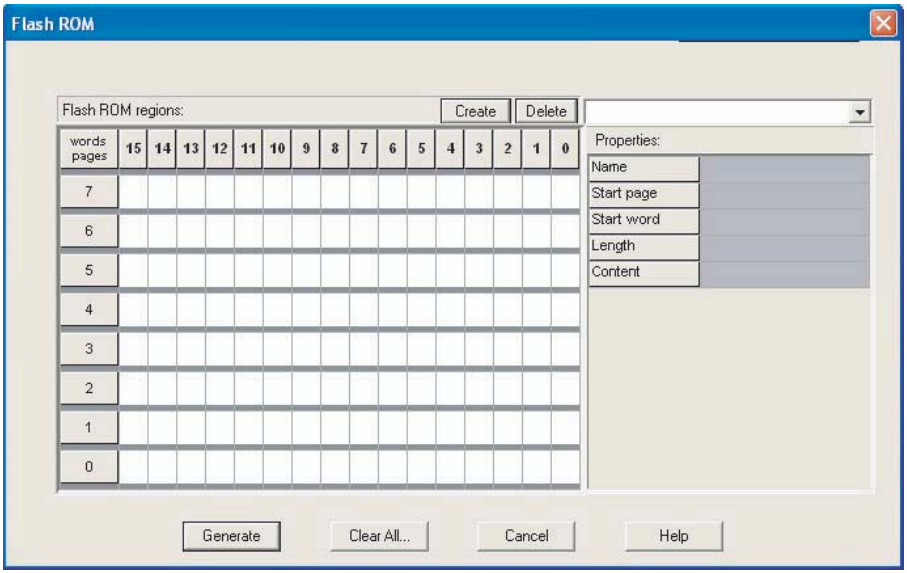


Figure 3-9. FlashROM Core Generator

To create a new FlashROM in SmartGen:

1. In SmartGen, choose the **FlashROM** in the **Function** tab. Double-click the **FlashROM** core in the **Variety View** window 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 does not copy, the page does not have enough room. Try another page with more room.
3. Click in the **Properties** grid to modify a regions 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 (you must specify EDIF, VHDL, or Verilog), SmartGen GEN file, UFC, and MEM file. The **Generate** button opens the **Save As** dialog. Specify a name, location, and file type, and click **Save**.

   After you generate the FlashROM netlist, you can instantiate the core similar to other SmartGen cores in your design.

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 Existing FlashROM Configuration

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

1. Open your SmartGen workspace that contains the configured FlashROM core.
2. Double-click the configured core (in the Configured Core View) to open the FlashROM configuration. The FlashROM core opens with all the settings you saved.
3. Modify the values you wish to change and click Generate to save your changes. Generate opens the Save As dialog. 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 change 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. SmartGen 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.

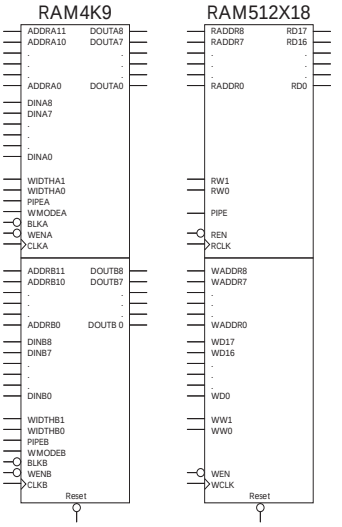
In addition to using the MEM file from SmartGen, you may create a binary file with 128 rows of eight bits and save the file as a MEM file. Actel recommends using different names if you plan to generate multiple MEM files. During place-and-route in Designer, the software recognizes the generic property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

Place-and-Route and FlashROM

There are no special instructions for place-and-route for FlashROM. Run Layout in Designer to place-and-route your design.

SRAM

RAM4K9 and RAM512X18	ProASIC3/E, FUSION



Function

RAM4K9 is a fully synchronous, true dual-port RAM with an optional pipeline stage; RAM512X18 is a fully synchronous, two-port RAM with an optional pipeline stage

Truth tables are listed below.

Input

Inputs are shown on the left of the diagrams. For example, ADDRA11, ADDRA10, ... , ADDRA0.

Output

Outputs are shown at the right on the diagrams. For example, DOUTA8, DOUT7, ... , DOUT0.

There are two main RAM macros in the Fusion libraries: RAM4K9 and RAM512X18. The RAM4K9 is a fully synchronous, true dual-port RAM with an optional pipeline stage. It can be used for word widths up to 9 bits. Both ports are capable of reading and writing, making it possible to write with both ports or read with both ports simultaneously. You can also read from one port while writing to the other. Each port also has an optional pipeline stage that can be controlled separately via the PIPE pins. The RAM512X18 is a fully synchronous, two-port RAM with an optional

pipeline stage. You can use it for word widths of 9 or 18 bits. It has one dedicated read port and one dedicated write port (you can read from one port while writing to the other). The read port also has an optional pipeline stage that you can control separately via the PIPE pin.

During the write operation of the RAM4K9, the WMODE pins control the data that appears on the read pins of the same port. When WMODE is high, the same data appears on the read and write ports at the rising CLK edge. When WMODE is low, the old data stored in the current memory location being addressed appears on the read port. There are no WMODE pins on the RAM512X18.

The aspect ratio of each port can be specified independently via the WIDTHA and WIDTHB pins. For the RAM512X18, the allowable values are 18 x 256 and 9 x 512. For the RAM4K9, the allowable values are 9 x 512, 4 x 1K, 2 x 2K, and 1 x 4K. Although it is possible to dynamically reconfigure the aspect ratios, the RAM4K9 and RAM512x18 were designed with only static configuration in mind, so the timing is unknown and you are discouraged from performing such operations. The FLEXRAM Macros can be used for this the Fusion device. The same is true for the WMODE and PIPE configuration pins.

The RAM4K9 only needs 2 bits to configure the WIDTH. The allowable RAM4K9 WIDTHA and WIDTHB values are shown in Table 3-6.

Table 3-6. RAM4K9 WIDTHA and WIDTHB Values

WIDTHA1, WIDTHA0	WIDTHB1, WIDTHB0	W x D
00	00	1 x 4K
01	01	2 x 2K
10	10	4 x 1K
11	11	9 x 512

The RAM512X18 also needs 2 bits to configure the read and write widths. The allowable RAM512X18 WW and RW values are shown in Table 3-7.

Table 3-7. RAM512x18 WW and RW Values

WW1, WW0	RW1, RW0	W x D
01	01	9 x 512
10	10	18 x 256
00, 11	00, 11	Illegal

When specifying a width that is less than the maximum (e.g. 1), the upper unused data input pins (e.g. DINA8 - DINA1) must be connected to GND. When specifying a depth that is less than the maximum (e.g. 512), the upper unused address pins (e.g. ADDRA11 - ADDRA9) must also be connected to GND.

When widths of 1, 2, and 4 are used, the ninth bit is skipped. This can cause counter-intuitive effects when these widths are used for read operations and larger widths are used for write operations (or vice versa). For example, if a width of 9 is used for writing and a width of 1 for reading, every 9th bit will be dropped. This effect may be desirable for removing parity bits. If a write width of 4 and read width of 9 is used, the 9th bit may either contain garbage or remnants of previous write operations when a write width of 9 or higher was being used. For this reason, SmartGen only supports the following aspect ratio combinations when one of the ports is configured with a 1-, 2-, or 4-bit width using the RAM4K9 (Table 3-8).

Table 3-8. SmartGen Supported Aspect Ratio Combinations for the RAM4K9

READ	WRITE
1 x 4K	1 x 4K
1 x 4K	2 x 2K
1 x 4K	4 x 1K
2 x 2K	1 x 4K
2 x 2K	2 x 2K
2 x 2K	4 x 1K
4 x 1K	1 x 4K
4 x 1K	2 x 2K
4 x 1K	4 x 1K

The RAM4K9 can still be used for 9-bit width applications, but no other bit-width can be used with it other than 9-bits (Table 3-9).

Table 3-9. SmartGen Supported Aspect Ratios for 9-bit Width Applications

READ	WRITE
9 x 512	9 x 512

There are several restrictions that apply when you use an 18 x 256 aspect ratio. For this reason, SmartGen uses the RAM512X18 whenever 18-bit widths are specified. The only allowable combinations of read and write configurations for the RAM512X18 are shown in Table 3-10:

Table 3-10. RAM512X18 Read and Write Combinations

READ	WRITE
18 x 256	18 x 256
18 x 256	9 x 512
9 x 512	18 x 256

The RADDR pins are always used for the read address in the above configurations and the WADDR pins are used for the write address. The RW pin is used to specify the read width and the WW pin for the write width. The WD pins are used for writing data and the RD pins for reading data.

Table 3-11. RAM4K9 Truth Table

Operation	Address	CLK	BLK	WMODE	WEN	RESET	DI	DO
Deselect	X	X	H	X	X	H	X	Data-Last
Reset	X	X	X	X	X	L	X	L
Read	ADDR	Rising Edge	L	L	H	H	X	Data
Write (0)	ADDR	Rising Edge	L	L	L	H	WData	Data-Last
Write (1)	ADDR	Rising Edge	L	H	L	H	WData	WData

When deserted, the BLK pins will cause the DO outputs to hold their last value. When asserted, the WEN pins can be used to switch each port between write and read mode. The RESET pin sets all outputs low but does not reset the memory. The WMODE pins are used to either allow the write data to appear immediately on the output pins or to hold the last value.

Table 3-12. RAM512x18 Truth Table

Operation	Address	WCLK	REN	WEN	RESET	WD	RD
Reset	X	X	X	X	L	X	L

Table 3-12. RAM512x18 Truth Table (Continued)

Operation	Address	WCLK	REN	WEN	RESET	WD	RD
Read	RADDR	Rising Edge	L	X	H	X	Stored Data
Write	WADDR	Rising Edge	X	L	H	WData	Data-Last

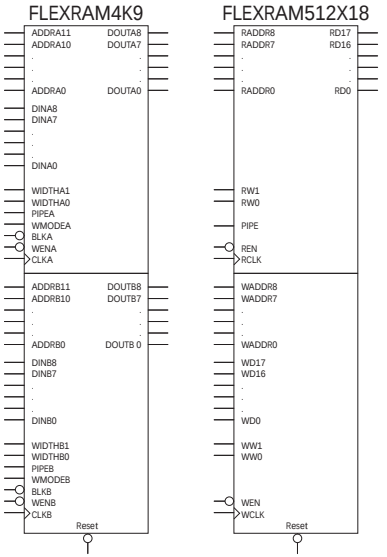
Use SmartGen to configure the RAM for typical use. SmartGen will not support dynamic reconfiguration or cascading width-wise. Customers wishing to avail themselves of such features must instantiate and configure the RAM macro manually.

Warnings

- Simultaneous write to the same address from both ports is possible, but the results are undefined. Avoid writing to the same address simultaneously from both ports.
- Dynamic reconfiguration of any pins possible but not supported by SmartGen.
- Cascading is possible and limited only by the number of available RAM blocks in a row, which is device dependent. SmartGen prompts you for device type information in order to correctly calculate the maximum.
- RESET has priority over BLKA and BLKB.
- In read mode (i.e. when WEN high) WMODE is ignored.

FLEXRAM4K9 and FLEXRAM512X18

Fusion



Function

FLEXRAM4K9 is a fully synchronous, true dual-port RAM with an optional pipeline stage; FLEXRAM512X18 is a fully synchronous, two-port RAM with an optional pipeline stage. You can configure the aspect ratios of these macros dynamically.

Truth tables are listed below.

Input

Inputs are shown on the left of the diagrams. For example, ADDRA11, ADDRA10, ... , ADDRA0.

Output

Outputs are shown at the right on the diagrams. For example, DOUTA8, DOUT7, ... , DOUT0.

There are two additional RAM macros in the Fusion library: FLEXRAM4K9 and FLEXRAM512X18. The FLEXRAM4K9 is a fully synchronous, true dual-port RAM with an optional pipeline stage. It can be used for word widths up to 9 bits. Both ports are capable of reading and writing, making it possible to write with both ports or read with both ports simultaneously. You can also read from one port while writing to the other. Each port also has an optional pipeline stage that can be controlled separately via the PIPE pins. The FLEXRAM512X18 is a fully synchronous, two-port RAM with an optional pipeline stage. You can use it for word widths of 9 or 18 bits. It has

one dedicated read port and one dedicated write port (you can read from one port while writing to the other). The read port also has an optional pipeline stage that you can control separately via the PIPE pin.

During the write operation of the FLEXRAM4K9, the WMODE pins control the data that appears on the read pins of the same port. When WMODE is high, the same data appears on the read and write ports at the rising CLK edge. When WMODE is low, the old data stored in the current memory location being addressed appears on the read port. There are no WMODE pins on the FLEXRAM512X18.

The aspect ratio of each port can be specified independently via the WIDTHA and WIDTHB pins. For the FLEXRAM512X18, the allowable values are 18 x 256 and 9 x 512. For the FLEXRAM4K9, the allowable values are 9 x 512, 4 x 1K, 2 x 2K, and 1 x 4K. You can configure aspect ratios of the FLEXRAM4K9 and FLEXRAM512X18 dynamically. The timing is unknown and you are discouraged from dynamically changing WMODE and PIPE configuration pins.

The FLEXRAM4K9 only needs 2 bits to configure the WIDTH. The allowable FLEXRAM4K9 WIDTHA and WIDTHB values are shown in Table 3-13.

Table 3-13. FLEXRAM4K9 WIDTHA and WIDTHB Values

WIDTHA1, WIDTHA0	WIDTHB1, WIDTHB0	W x D
00	00	1 x 4K
01	01	2 x 2K
10	10	4 x 1K
11	11	9 x 512

The FLEXRAM512X18 also needs 2 bits to configure the read and write widths. The allowable FLEXRAM512X18 WW and RW values are shown in Table 3-14.

Table 3-14. FLEXRAM512x18 WW and RW Values

WW1, WW0	RW1, RW0	W x D
01	01	9 x 512
10	10	18 x 256
00, 11	00, 11	Illegal

When specifying a width that is less than the maximum (e.g. 1), the upper unused data input pins (e.g. DINA8 - DINA1) must be connected to GND. When specifying a depth that is less than the

maximum (e.g. 512), the lower unused address pins (e.g. ADDRA2 - ADDRA0) must also be connected to GND.

When widths of 1, 2, and 4 are used, the ninth bit is skipped. This can cause counter-intuitive effects when these widths are used for read operations and larger widths are used for write operations (or vice versa). For example, if a width of 9 is used for writing and a width of 1 for reading, every 9th bit will be dropped. This effect may be desirable for removing parity bits. If a write width of 4 and read width of 9 is used, the 9th bit may either contain garbage or remnants of previous write operations when a write width of 9 or higher was being used. For this reason, SmartGen only supports the following aspect ratio combinations when one of the ports is configured with a 1-, 2-, or 4-bit width using the FLEXRAM4K9.

Table 3-15. SmartGen Supported Aspect Ratio Combinations for the FLEXRAM4K9

READ	WRITE
1 x 4K	1 x 4K
1 x 4K	2 x 2K
1 x 4K	4 x 1K
2 x 2K	1 x 4K
2 x 2K	2 x 2K
2 x 2K	4 x 1K
4 x 1K	1 x 4K
4 x 1K	2 x 2K
4 x 1K	4 x 1K

The FLEXRAM4K9 can still be used for 9-bit width applications, but no other bit-width can be used with it other than 9-bits.

Table 3-16. SmartGen Supported Aspect Ratios for 9-bit Width Applications

READ	WRITE
9 x 512	9 x 512

There are several restrictions that apply when you use an 18 x 256 aspect ratio. For this reason, SmartGen uses the FLEXRAM512X18 whenever 18-bit widths are specified. The only allowable combinations of read and write configurations for the FLEXRAM512X18 are as follows:

Table 3-17. FLEXRAM512X18 Read and Write Combinations

READ	WRITE
18 x 256	18 x 256
18 x 256	9 x 512
9 x 512	18 x 256

The RADDR pins are always used for the read address in the above configurations and the WADDR pins are used for the write address. The RW pin is used to specify the read width and the WW pin for the write width. The WD pins are used for writing data and the RD pins for reading data.

Table 3-18. FLEXRAM4K9 Truth Table

Operation	Address	CLK	BLK	WMODE	WEN	RESET	DI	DO
Deselect	X	X	H	X	X	H	X	Data-Last
Reset	X	X	X	X	X	L	X	L
Read	ADDR	Rising Edge	L	L	H	H	X	Data
Write (0)	ADDR	Rising Edge	L	L	L	H	WData	Data-Last
Write (1)	ADDR	Rising Edge	L	H	L	H	WData	WData

When de-asserted, the BLK pins will cause the DO outputs to hold their last value. When asserted, the WEN pins can be used to switch each port between write and read mode. The RESET pin sets all outputs low but does not reset the memory. The WMODE pins are used to either allow the write data to appear immediately on the output pins or to hold the last value.

Table 3-19. FLEXRAM512x18 Truth Table

Operation	Address	WCLK	REN	WEN	RESET	WD	RD
Reset	X	X	X	X	L	X	L

Table 3-19. FLEXRAM512x18 Truth Table

Operation	Address	WCLK	REN	WEN	RESET	WD	RD
Read	RADDR	Rising Edge	L	X	H	X	Stored Data
Write	WADDR	Rising Edge	X	L	H	WData	Data-Last

Use SmartGen to configure the RAM for typical use. SmartGen will not support dynamic reconfiguration FLEXRAM or cascading width-wise. Customers wishing to avail themselves of such features must instantiate and configure the RAM macro manually.

Warnings

- Simultaneous write to the same address from both ports is possible, but the results are undefined. Avoid writing to the same address simultaneously from both ports.
- Dynamic reconfiguration of any pins possible but not supported by SmartGen.
- RESET has priority over BLKA and BLKB.
- In read mode (i.e. when WEN high) WMODE is ignored.
- Dual-port operation not possible unless both ports have the same aspect ratio.

RAM Fusion

SmartGen 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. SmartGen supports the generation of memories that have different Read and Write aspect ratios.

Parameters Fusion RAM

Two Port or Dual Port:

You may choose between a Two Port and a Dual Port configuration. Different read and write aspect ratios are not supported in Dual Port mode. Also, the RAM512X18 element cannot be used to implement a dual port RAM. Using a dual port RAM can potentially increase the number of resources required.

For example, a 256X18 RAM can be created in one RAM block for Two Port, but requires 2 RAM Blocks for Dual Port. SmartGen always creates Dual Port RAM when this selection is made even if it consumes more resources.

Note: When switching from two port to dual port, the GUI names are not displayed properly.

In Two Port mode the naming labels the Write and Read ports correctly. However, when you select a Dual Port Memory, wherever the GUI displays Write it corresponds to PORTA, and where ever it displays Read it corresponds to PORTB.

Single Read/Write Clock or Independent Read Write Clocks

You may choose to have the same clock driving both RCLK and WCLK and Two Port Mode or CLKA and CLKB in Dual Port Mode. Or you may choose to have independent Read and Write Clocks.

Write/Read Depth:

SmartGen supports the generation of RAM having a write or read depth between 1 and 65536. However, all depths are not available in all configurations. Write and Read Depth values can be different. When choosing a Dual Port RAM only Write depth is available as different aspect ratios are not supported in dual port mode.

Write/Read Width:

SmartGen supports the generation of RAM having a write or read width between 1 and 576. However, all depths are not available in all configurations. Write and Read Width values can be different. When choosing a Dual Port RAM only Write width is available as different aspect ratios are not supported in dual port mode.

Read and Write Clock Polarities:

SmartGen instantiates inverters as necessary to achieve the requested polarity. In the case of Dual Port RAM only Write Clock polarity is selectable and it applies to both CLKA and CLKB.

Read and Write Enable Polarities:

SmartGen instantiates inverters as necessary to achieve the requested polarity. This feature is available only for the Two Port RAM.

Write Mode A and Write Mode B:

SmartGen configures the WMODE signals based on your selection. This is a static selection and cannot be changed dynamically by driving it with a signal. For Two Port RAM only Write Mode A is available.

The RAM512X18 element has no WMODE selection and the default behavior of output data for this element is to hold the previously read data. In a case where you specify pass-through mode for WMODE then SmartGen uses RAM4K9 even if it results in usage of more resources. This situation arises only in the Two Port configurations, as RAM512X18 is not used for dual port RAM.

Read Pipeline A and Read Pipeline B:

SmartGen configures the PIPEA and PIPEB signals to make the output pipelined or non-pipelined based on your selection. This is a static selection and cannot be changed dynamically by driving it with a signal. For Two Port RAM only Read Pipeline A is available.

Signals in SmartGen Generated Netlists

DataA, DataB: Input Data for Dual Port RAM

QA, QB: Output Data for Dual Port RAM

AddressA, AddressB: Address Busses for Dual Port RAM

CLKA, CLKB: Clocks for Dual Port RAM for independent Clocks

Clock: Clock for Dual Port RAM for Single Clock

RWA, RWB: Signals to switch between Read and Write Modes for Dual Port RAM; Low = Write, High = Read

BLKA, BLKB: Active Low Block Enables for Dual Port RAM

RESET: Output Reset

Data: Input Data For Two Port RAM

Q: Output Data for Two Port RAM

WAddress, RAddress: Write and Read Address Busses for Two Port RAM

WEN, REN: Write and Read Enable For Two Port RAM

Wclock, Rclock: Write and Read Clocks for Two Port RAM for independent Clocks

Clock: Clock for Read and Write for Two Port RAM for single Clock

RESET: Output Rest

Caveats to RAM Generation with SmartGen

It does not support configurations that use a word width of 1,2 or 4 for Write and a word width of 9 for Read. This configuration causes the MSB of the output to be undefined. However, configurations that do not use the 9th bit, like writing 1024X4 and reading 512X8 are possible.

SmartGen supports deep and wide RAM cascading only up to 64 blocks.

SmartGen 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 any signal is not supported in SmartGen.

SmartGen will give a configuration error for unsupported configurations.

Tips

- Writing different data to same address using both ports in Dual Port RAM is undefined and should be avoided.
- Writing to and reading from the same address is undefined and should be avoided.
- Aspect Ratios should not be dynamically reconfigured.
- All unused inputs must be grounded.
- WMODE is ignored during read operation.
- RESET does not reset the memory contents. It resets only the output.

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.

SmartGen RAM Content Manager

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 improves (reduces) simulation runtime.

The SmartGen RAM core generator takes away much of the complexity required in the generation of large RAMs that utilize 1 or more of the RAM blocks on the device. SmartGen generates a RAM to match your configuration by the total number of memory blocks required and creating the required surrounding logic to cascade them.

SmartGen cascades RAM blocks in 3 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 memory size. SmartGen must partition your memory file appropriately such that the right content goes to the right block RAM when multiple blocks are cascaded.

Supported Formats

Intel-Hex Record Format

A standard format created by Intel. Memory contents are stored in ASCII files using hexadecimal characters. Each file is made of 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).

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 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 Actel implementation of these formats interprets data sets in bytes. This means that if the memory width is 7 bits, every 8th bit in the data set is ignored. Or, if the data width is 9, two bytes are assigned to each memory address and the upper 7 bits of each 2-byte pair are ignored.

Using the RAM Content Manager

In order to use the RAM Content Manager you must use devices that support the RAM Content Manager features, such as Fusion.

To open the RAM Content Manager:

1. From the Options menu, choose Workspace Settings and set your device family to Fusion. Click OK.
2. Click RAM in the Core Catalog to display the list of RAM types available for Axcelerator.
3. Double-click Synchronous RAM to create a new RAM block for Axcelerator. 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 Figure 3-10 on page 102.

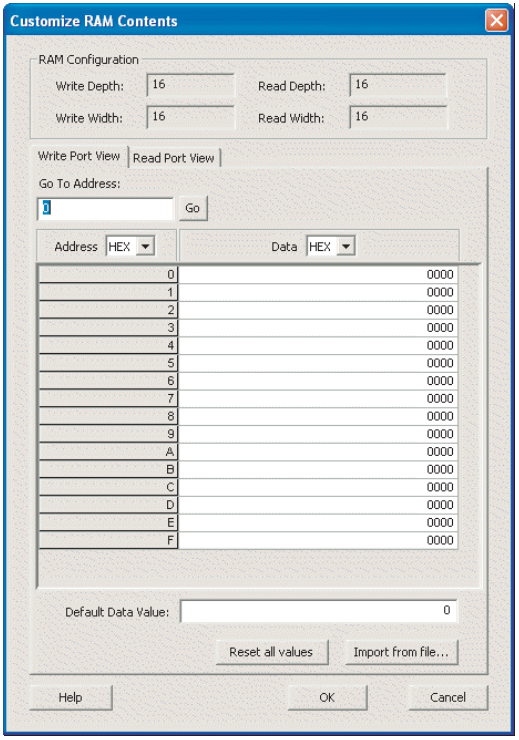


Figure 3-10. RAM Content Manager in SmartGen

RAM Configuration

Write Depth and Write Width – Write Depth and Width, specified in the RAM core generator dialog box.

Read Depth and Read Width – Read Depth and Width, specified in the RAM core generator dialog box.

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.

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) to load through the Write Port. During import, file extensions are set to *.hex for Intel-Hex files and *.s for Motorola S 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) to load through the Read Port. During import, file extensions are set to *.hex for Intel-Hex files and *.s for Motorola S 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.

MEMFILE (RAM Content Manager Output File)

Transfer of RAM data (from the RAM Content Manager) to testing systems 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 SmartGen. 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.

Tips and Tricks

Please refer to the Fusion SRAM/FIFO Blocks application note (http://www.actel.com/techdocs/appnotes/products.aspx)

FIFO

The Fusion SRAM memory block includes dedicated FIFO control logic to generate internal addresses and external flag logic (Full, Empty, AFULL, AEMPTY).

The description of the FIFO controller and flags are included in the Fusion Datasheet including timing Characteristics for flag generation.

FIFO4K18

FIFO4K18 is fully synchronous and has its own embedded controller. Like the RAM, the FIFO can have different write and read aspect ratios that can be configured dynamically. The WW and RW pins are used to specify one of five allowable aspect ratios, as shown in Table 3-20.

Table 3-20. FIFO4K18 Aspect Ratios

WW2, WW1, WW0 and RW2, RW1, RW0	W x H
000	1 x 4K
001	2 x 2K
010	4 x 1K
011	9 x 512
100	18 x 256
101, 110, 111	Illegal

The AEVAL and AFVAL pins are used to specify the almost empty and almost full threshold values, respectively. 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. Therefore AEVAL and AFVAL must be left-justified for widths greater than one (i.e. unused lsb's must be grounded).

Table 3-21. Aspect Ratio and Related Bits to Ground

Aspect Ratio	AEVAL/AFVAL Bits to Ground
1 x 4K	none
2 x 2K	0
4 x 1K	1:0
9 x 512	2:0
18 x 256	3:0

When the number of words stored in the FIFO reaches the amount specified by AEVAL while reading, the AEMPTY output will go high. Likewise when the number of words stored in the FIFO reaches the amount specified by AFVAL while writing, the AFULL output will go high. The FULL and EMPTY outputs will go high when the FIFO is completely full or empty, respectively.

It should be noted that the internal memory size is 512 X 9. When widths of 1, 2, and 4 are specified, the 9th bit is skipped.

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. SmartGen treats them as static configuration pins and always ties them high.

Independent read and write operations are allowed, however only the read port can be pipelined. Data on the appropriate WD pins are written to the FIFO every rising WCLK edge as long as WEN and WBLK are low. Data is read from the FIFO and output on the appropriate RD pins every rising RCLK edge as long as REN is high and RBLK is low.

The active low RESET pin is used to asynchronously clear the outputs of the FIFO and reset the internal read and write address counters. It sets all the RD pins low, the FULL and AFULL pins low, and the EMPTY and AEMPTY pins high, however the contents of the memory remain unchanged. RESET has priority over RBLK and WBLK.

When instantiating the FIFO4K18, all unused input pins must be connected to GND.

Warnings

- The WW, RW, AEVAL, and AFVAL pins can be dynamically configured, but only static configuration will be supported by SmartGen.

- The RPIPE signal can be dynamically configured, but only static configuration will be supported by SmartGen.
- No pipeline on the write port.
- Cascading allowed and supported in the width direction only by SmartGen. Cascading in the depth direction requires the use of a soft controller (i.e. implemented with core logic).
- ESTOP and FSTOP applications not clear. The effect of activating ESTOP is to allow the read pointer to wrap around, allowing the memory contents to be read over and over again with rewriting after EMPTY. The effect of activating FSTOP is not clear, however, since the write pointer could wrap around allowing overwriting of data which is never read. Therefore SmartGen will always tie these pins off high.

FIFO Fusion

The SmartGen tool automatically cascades FIFO blocks to create wider memories by choosing the most efficient aspect ratio. It also handles the grounding of unused bits. SmartGen also supports the generation of FIFOs that have different Read and Write aspect ratios.

Fusion FIFO Parameters

Almost Full/Empty Flags

The user is allowed to choose among Static, Dynamic, and No flags. When No flags is chosen, SmartGen grounds AFVAL, AEVAL, and AFULL. AEMPTY signals do not appear as ports on the top level. When Static Flags are chosen, SmartGen configures the AFVAL and AEVAL accordingly. For Dynamic Flags, users 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 "Using FIFO Flags" on page 109.

Pipeline

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

Write/Read Depth

SmartGen supports the generation of FIFO having a write or read depth between 1 and 4096. Write and Read Depth values can be different.

Write/Read Width

SmartGen supports the generation of RAM having a write or read width between 1 and 576. Write and Read Width values can be different.

Read and Write Clock Polarities

SmartGen instantiates inverters as necessary to achieve the requested polarity.

Read and Write Enable Polarities

SmartGen will instantiate inverters as necessary to achieve the requested polarity.

Continue Counting Read Counter After FIFO is Empty (ESTOP)

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

Continue Counting Write Counter After FIFO is Full (FSTOP)

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

For more information on the above two options, refer to the section, "Using ESTOP and FSTOP" on page 108.

Almost Full Value/Units

This choice is applicable only in the Static Almost Full/Empty selection. The threshold for Almost Full is specified in terms of Write Words or Read Words.

Almost Empty Value/Units

This choice is applicable only in the Static Almost Full/Empty selection. The threshold for Almost Empty is specified in terms of Write Words or Read Words.

For more information on these choices please refer to the section, "Using FIFO Flags" on page 109.

Signals in SmartGen-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

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. SmartGen will configure these signals based on user 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. Other applications for this feature need to be identified.

A typical user would not need to use these features and should leave these options unselected 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 rations, 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.

Caveats to FIFO Generation with SmartGen

- Depth cascading is currently not supported in SmartGen. Therefore the maximum depth supported is only 4096.
- It supports wide cascading up to 64 blocks.
- SmartGen does not generate a FIFO based on a specific device. It is the user's responsibility to make sure the FIFO fits physically on the device.
- Dynamic configuration of any signal with exception of AFVAL/AEVAL is not supported in SmartGen.
- SmartGen will give a configuration error for unsupported configurations.
- WBLK and RBLK are always grounded in SmartGen, which means the FIFO block always remains enabled. Users should control the FIFO with WEN and REN.

Sub-Block Interconnection

In order to connect the embedded memory blocks to other peripheral blocks there are no connectivity rules required. These are connected straight to the core.

4 Analog Block

The Fusion devices contain up to 10 analog channels. You can configure the analog blocks by using the SmartGen Analog System Builder.

The Analog System Builder in SmartGen 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 required order of sampling
- Define the operations on converted digital output from ADC
- Specify the RTC settings. Refer to the RTC Chapter

The analog block consists of:

- Analog I/O structure (Analog Quad—AQ)
- Real-Time Counter (for details, refer to "Real Time Counter" on page 49)
- Analog to Digital Converter (ADC)
- Analog Configuration Multiplexer (ACM)

All of these elements are combined in a single Analog Block core. Figure 4-1 on page 112 shows the analog quad implementation.

The Fusion datasheet includes this diagram of the Analog Block, with all the input and output signals. It is possible to use the analog block in this form and manually create all the configuration signals. However, in order to simplify the use of the Analog structures within the FPGA design flow, Actel has provided tools within SmartGen to create and configure the Analog System. Instructions for configuring these blocks manually are not included in this document.

This chapter includes the usage of the simplified tool set for the Analog design. In addition, if you would like to use the standalone Analog Block in the future, it is useful to start with an automated configuration to create a template for manual alteration.

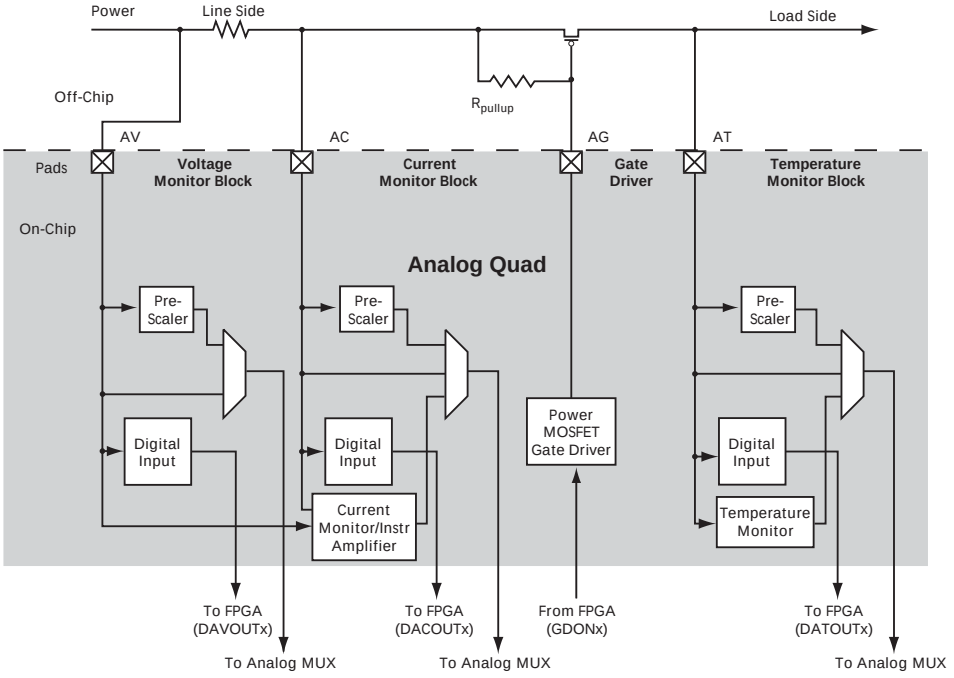


Figure 4-1. Analog Quad

Analog Block Builder System

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

The Analog to Digital Converter (ADC) Configuration sets your System Clock speed and resolution.

Available Peripherals lists all the analog peripherals you can add to your design. As you add peripherals, some resources are exhausted (used up). The Analog System Builder disables peripherals in the list for which there are insufficient resources. 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 per a given input channel. SmartGen takes the required acquisition times for all peripherals and computes the ADC clock frequency and the number of ADC clocks per sample, per peripheral, such that each peripheral meets or exceeds your required acquisition time.

Sampling Rate (in µs) - The rate at which a given channel gets sampled by the ADC. See the Modify Sampling Sequence section below for more information on setting your sampling rate. Sampling rate is affected by the following parameters:

- System Clock Frequency
- ADC Resolution
- Number of channels in the system
- Settling times for all channels
- Number of Flags specified per channel
- Operating Sample Sequence
- Real Time Counter - You can configure the Real Time Counter so that it functions as a chronometer, configure it to generate periodic alarms in conjunction with other peripherals (such as the Voltage monitor), etc. Package Pin - SmartGen automatically assigns a package pin for each channel in each peripheral added to the system. 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.

Modify Sampling Sequence

Since there is only one ADC but 30 input channels, the channels must be sequenced in the required order for the system to function. There are 64 time slots to sequence the channels added to the system. You can also run operations on the ADC in addition to sampling the channels.

The sampling sequence specifies the Analog System's sampling order. For example, the sequence may be specified to sample channel 1 continuously, or it can be specified to sample channels 1 through 5 then repeat. In either case, the Sample Sequence Controller will drive the ADC signals to sample the channels in the specified sequence.

Your application and requirements dictate the sampling sequence.

SmartGen provides you with two operating sequences: "main" and "jump". The main operating sequence operates the majority of the time in a constant loop. The jump sequence is used primarily for interrupt events and must be triggered by user logic.

The jump sequence is not a factor in the sampling rate calculations for the channels or the system. Furthermore, any operations following a RESET operation will not be included in the sample rate calculations

Functions can be inserted into the sequence that do not perform any sampling, but are used to configure the ADC. These functions most likely would affect the sampling rate of the system. The Modify Sampling Sequence dialog box is shown in Figure 4-2.

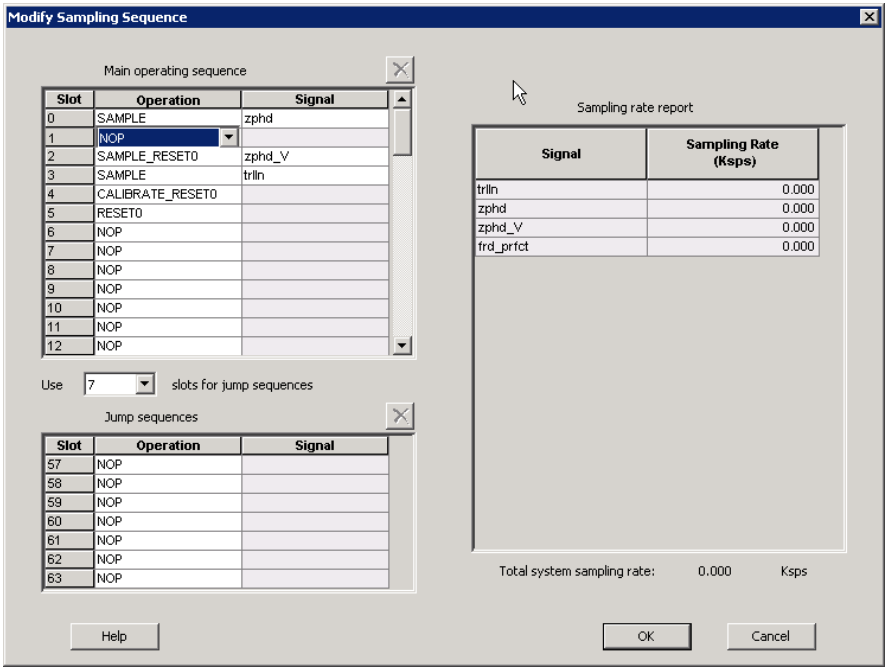


Figure 4-2. Modify Sampling Sequence Dialog Box

Main Operating Sequence

This is the main sequence with SAMPLE, SAMPLE_RESET0, RESET0 or CALIBRATE as the last operation (so that the sequence repeats itself). You can go to a slot in the jump sequence, and return to slot 0 to restart the main operating sequence. The operations in the main sequence are:

- SAMPLE - Sample a channel that is added to the system
- SAMPLE_RESET0 - Sample a channel that has been added to the system and reset to Slot 0.
- RESET0 - Reset to Slot 0
- CALIBRATE - Calibrate the analog to digital converter.

  If you do not specify a terminating option in the main operating sequence (no RESET0) the sequence continues up to the 64th slot and resets to slot 0, with no operations on the intermediate slots (NOP).

  SAMPLE_RESET0 completes the sample and reset action in the same time slot. SAMPLE and RESET0 require two time slots.

Jump Sequences

These are extra sampling options that are accessed via an external jump. 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.
- **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, or resets to slot 0 if it is an operation with reset0.

  If the operation is a STOP or POWERDOWN, the sequencer stops processing until another jump is initiated to a different slot. STOP stops the ADC from sampling. POWERDOWN forces ADC to power down. Returning from this state requires calibration of the ADC.

  You can reserve the number of slots required for jump sequence. The slot with the smallest number used in the jump sequence determines how many slots can be made available to the main operating sequence.
- **Resulting sampling rate** - The number of times that given channel gets sampled in a second, expressed in Ksps (thousands of samples per second).

Analog MUX

For the V-quad, the analog MUX can choose the V-prescaler or the V-direct analog input; for the C-quads, the analog MUX can select the C-prescaler, C-direct analog input or current monitor; for the T-quads, the analog MUX can choose the T-prescaler, the T-direct analog input or the Temperature Monitor.

Set the V, C and T-prescaler Opamp, V, C and T-direct analog input switches and Current Monitor switch need be set according to Table 4-1 for power efficiency and/or ADC conversion accuracy.

Table 4-1. Switch Conditions

Selected MUX input Switch	V-Prescaler	V-Direct analog input	C-Prescaler	C-Direct analog input	Current Monitor	T-Prescaler	T-Direct analog input	Temp. Monitor
V-Prescale Op Amp	ON	OFF	X	X	X	X	X	X
V-Direct analog input switch	OFF	ON	X	X	"OFF" or "ON" and 0<AV<varef	X	X	X
C-Prescale Op-Amp	X	X	ON	OFF	OFF	X	X	X
C-Direct analog input switch	X	X	OFF	ON	OFF	X	X	X
Current Monitor switch	X	X	OFF	OFF	ON	X	X	X
T-Prescaler Op-Amp	X	X	X	X	X	ON	OFF	OFF
T-Direct input switch	X	X	X	X	X	OFF	ON	OFF

If you do not meet the switch conditions shown in Table 4-1, the analog MUX output is 0.0 and an error message appears.

In addition, when you the AV direct analog input, the scaling factor for the V-prescaler (B0[2-0]) must be

- 100 for the 1.0 V range
- 101 for the 0.5 V range
- 111 for the 0.125 V range

The same restriction applies when you select the AC direct input; the scaling factor for the C-prescaler (B1[2-0]) must be 100 or above. This restriction does not apply to the AT direct analog input.

ADC

The power-up calibration time after ADC comes out of reset is 3840 ADC_CLK cycles.

The conversion time can vary greatly depending on the SYSCLK frequency, ADCCLK frequency (determined by TVC), the STC settings, and the conversion bit-resolution (MODE).

t_conv = t_sync_read + t_sample + t_distrib + t_post_cal + t_sync_write

t_conv = SYSCLK period + ((2 + STC) * ADCCLK period) + (8, 10 or 12 * ADCCLK period) + (2 * ADCCLK period) + SYSCLK period

t_sync_read: Time for latching the input data

t_sample: Time for sampling the analog signal

t_distrib: Time for charge distribution

t_post_cal: Time for post-calibration

t_sync_write: Time for latching the output data

A Verilog parameter / VHDL generic enables faster conversion time. See the User parameter xxxx sectionxx for more info.

Analog Blocks

Using the Analog System Builder you can create, configure, and place the following analog blocks or peripherals:

- Voltage Monitor
- Current Monitor
- Temperature Monitor
- Direct Digital Input
- Gate Driver

- Real Time Counter

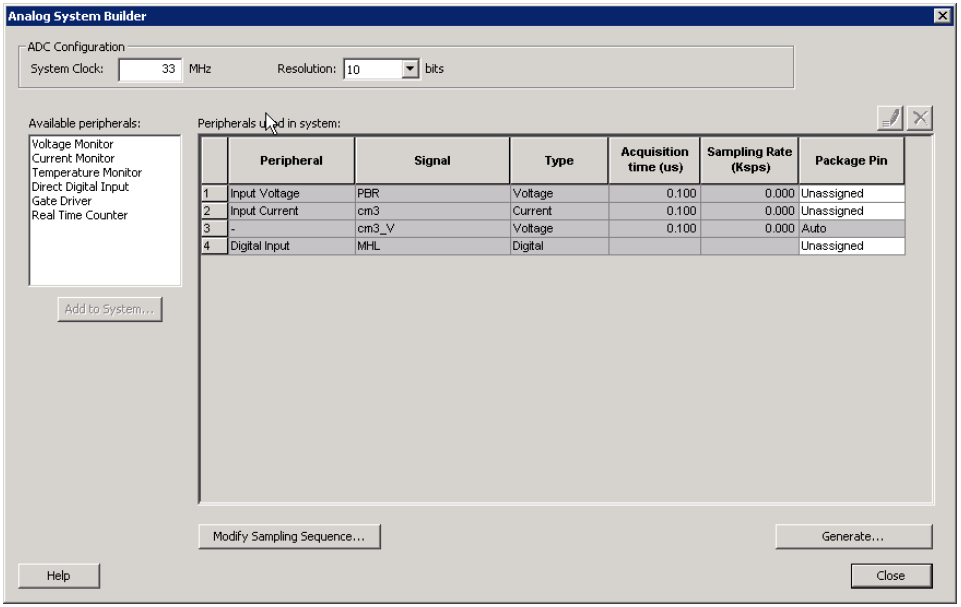


Figure 4-3. Analog System Builder

Voltage Monitor

The Voltage Monitor (Figure 4-4) contains a 2-channel analog multiplexor 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, Temperature, and Voltage Peripherals" on page 125 for information on the Digital filtering factor, Acquisition time, and Comparison Flag Specifications.

The only control unique to this peripheral is Maximum voltage (described below).

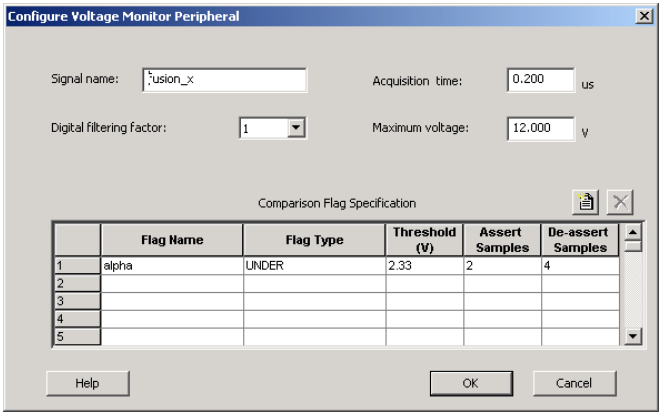


Figure 4-4. Configure Voltage Monitor Dialog Box

Maximum Voltage

The maximum anticipated voltage measured by this Voltage Monitor peripheral pad. The range is -12V to +12V (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. SmartGen automatically configures the Pre-scalar in the AB Analog Block for this peripheral based on the maximum voltage such that the input voltage is scaled up / down (if the voltage given to ADC is in the proper range). SmartGen also post-scales the digital result of ADC conversion so that it returns a result in your specified range.

Polarity

Each quad has a polarity bit, Bx[6], (e.g. B0[6] for AV0-polarity and B4[6] for AV1-polarity). The default polarity is Positive, Bx[6] = '0'. Polarity error occurs when the polarity bit is inconsistent with the quad sign (e.g. AV0 > 0 and AV0-polarity = '1').

AT-quad can only be positive and therefore its polarity can only be set to Positive, Bx[6] = '0'. For AT-quad, polarity error occurs if AT is negative or if AT-polarity is set to Negative, Bx[6] = '1'.

Prescaler

Each quad has a Prescaler Opamp mode bit, Bx[7], (e.g. B0[7] for AV0- prescaler op-amp and B4[7] for AV1 prescaler op-amp). Default is Powerdown, Bx[7] = ‘0’.

If the factor of the prescaler input and scaling factor is greater than the internal reference voltage, the prescaler output will saturate and the prescaler output will be equal to the internal reference voltage (default 2.56 V).

If a Polarity error occurs (e.g. AV0 > 0 and AV0-polarity = '1'), the prescaler output will be '0.0'.

Current Monitor

The current monitor in Fusion (Figure 4-5) measures the differential voltage across a resistor between a pair of Voltage and Current Input channels to measure the current. 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. SmartGen 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 = (change in voltage)/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, Temperature, and Voltage Peripherals" on page 125 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.

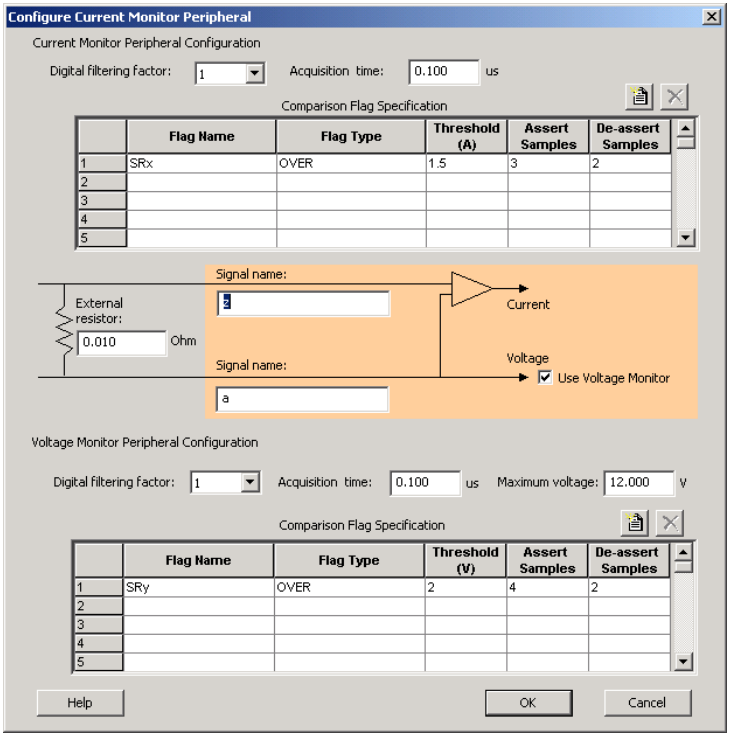


Figure 4-5. Configure Current Monitor Peripheral Dialog Box

External Resistor

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

Maximum voltage

The maximum anticipated voltage measured by this Voltage Monitor peripheral pad. The range is -12V to +12V (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. SmartGen automatically configures the Pre-scalar in the AB Analog Block for this peripheral based on the maximum voltage such that the input voltage is scaled up / down (if the voltage given to ADC is in the proper range). SmartGen also post-scales the digital result of ADC conversion so that it returns a result in your specified range.

Each C-quad has a Current Monitor Switch bit (B0[4] for AC0, B4[4] for AC1, etc). This switch needs to be 'ON' if the analog MUX selects the Current Monitor, otherwise the analog MUX output will be '0.0'. Default is Off, B0[4] = '0'.

The current monitor output is the difference between the AV and AC multiplied by a factor of 10. CMSTB-9 enables the current monitor for analog quads 0-9. Additionally, each C-quad has a Current-Monitor Switch (B0[4]) which enables you to switch the current monitor on or off. This switch needs to be 'ON' if the analog MUX selects the Current Monitor input, otherwise the analog MUX output will be '0.0'. The default setting is off.

The following requirements must be met in order to use the current monitor:

- ABS(AV) needs to be greater than ABS(AC), otherwise the Current Monitor returns a value of 0.0
- AV and AC must have the same sign and polarity. If not, they are invalid Current monitor inputs, and the current monitor output will be 0.0
- If a Polarity error occurs (e.g. AV0 > 0 and AV0-polarity = '1', or AC0 < 0 and AC0-polarity = '0'), the current monitor output will be 0.0
- If the difference between the AV and AC multiplied by a factor of 10 is greater than the internal reference voltage, the current monitor output saturates and the current monitor output is equal to the internal reference voltage (default 2.56 V).

Direct Digital Input

You can use the Direct Digital Input (Figure 4-6) 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 also be up to 12 V. but are limited to a frequency of 10 MHz.

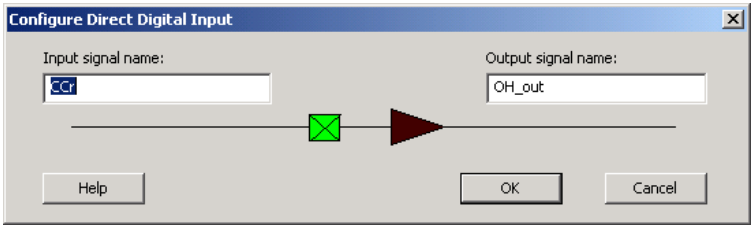


Figure 4-6. Configure Direct Digital Input Dialog Box

Direct Analog Input

Each V, C and T-quad has a Direct Analog Input Switch (B0[5] for AV0, B1[5] for AC0, and B3[5] for AT0) which enables you to switch the direct analog input ON or OFF. This switch must

be ON if the analog MUX selects the direct analog input, otherwise the analog MUX output will be 0.0. The default setting is OFF.

Gate driver

The Gate Driver (Figure 4-7) 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.

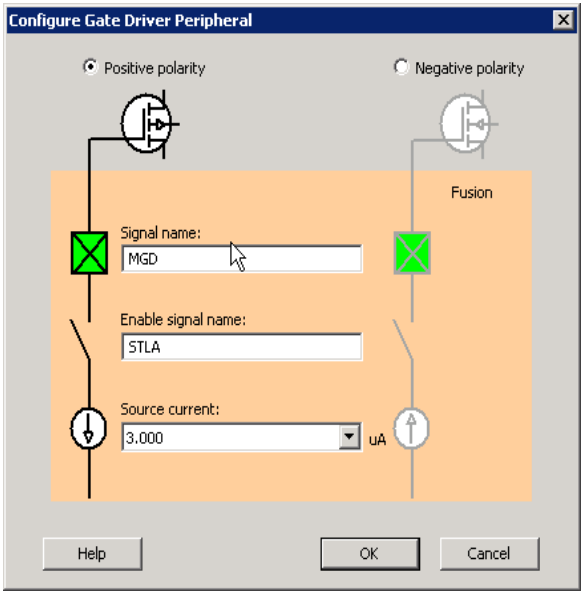


Figure 4-7. Output Gate Driver Dialog Box

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 or N-Channel type devices, by either pulling down to ground (P-channel) or up to ground (N-channel 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 enabled when the gate comes down.

Source Current

The amount of current the device is expected to source to bring the gate down. The Gate Driver supports four selectable gate drive levels: 1 µA, 3 µA, 10 µA, and 30 µA.

Temperature Monitor

When used in conjunction with an external bipolar transistor, the Temperature Monitor (Figure 4-8) is designed to measure temperature of an external location. A temperature monitor circuit can be very sensitive to system noise.

See the "Configuring Current, Temperature, and Voltage Peripherals" on page 125 for information on the Digital filtering factor, Acquisition time, and Comparison Flag Specifications.

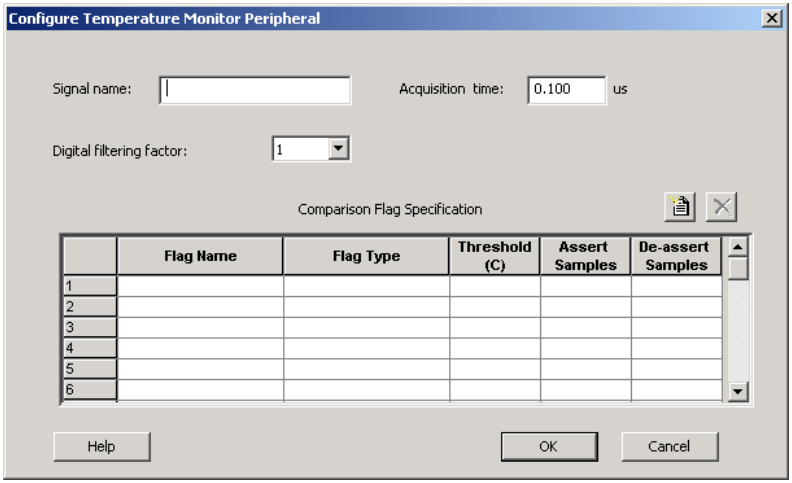


Figure 4-8. Configure Temperature Monitor Peripheral Dialog Box

The temperature monitor output is the AT-quad value multiplied by a factor of 12.5. TMSTB0-9 enables the temperature monitor for analog quads 0-9.

AT quad only accepts positive voltages, and T-pad polarity has to be set to 0 (Positive)

If the AT-quad value multiplied by 12.5 is greater than the internal reference voltage, the temperature monitor output saturates and the temperature monitor output is equal to the internal reference voltage (default 2.56V).

When using the temperature monitor, to reflect a temperature change, the value applied to AT should be a differential voltage.

AT (delta V) = T (K) * (0.0595 / 300)

AT (delta V) = T (K) * 1.983E-4

Using ADC in the Temperature Monitor

Using the previous equation, 300K (room temperature) should correspond to 59.5mV (0.0595 on AT, therefore 0.748 at the temperature monitor output / ADC input). When doing a 10-bit ADC conversion using a 2.56V reference voltage, 0.0595 on AT (T=300K) will give a RESULT of 300 (decimal). In this case, 1LSB change on RESULT corresponds to 1K temperature change.

Configuring Current, Temperature, and Voltage Peripherals

The Current, 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:

$$Filtering\_result_n = filtering\_result_{n\_1} + (ADC\_Result_n / filtering\_factor) - (filtering\_result_{n\_1} / filtering\_factor)$$

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

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.

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.

Comparison Flag Specification

Once the software calculates the average, you can run further processing on the result. Compare the result to a given threshold to determine whether it is above or below a given threshold, and decide 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.

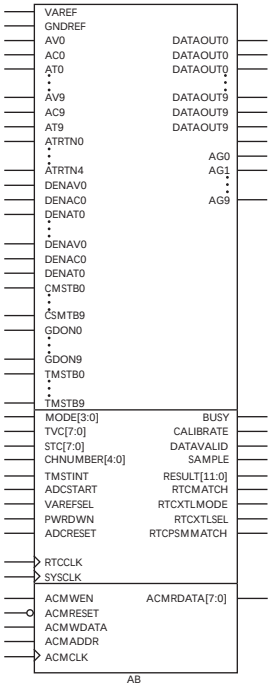
Assert Samples - The number of consecutive samples on this channel that reach or pass the threshold for the flag to assert. This can be is a glitch removal feature. If it is set to 1, the final flag is identical to the comparison result.

For example, if your Assert Sample value is 3.0 V, the channel must surpass 3.0 V five times in a row on this channel for the flag to assert. If your voltage values are less than 3.0 V, the flag will not assert.

Analog Macros

Analog Block

Fusion



Function

Analog system builder for use with SmartGen and Fusion. See the Fusion datasheet for a thorough description of the Analog System Builder.

Inputs / Outputs

Inputs are listed on the left, outputs on the right.

See the description below for an explanation of the inputs and outputs available on the Analog System Builder. For a complete description of the features in the ASB, see the Fusion datasheet.

Analog Block Pin Description

Table 4-1 • Analog Block Pin Description

Signal Name	Number of bits	Direction	Function	Location of Details
VAREF	1	Inout	External voltage reference; used as either input or output, depending on VREFSEL	ADC
GNDREF	1	Input	External ground reference	ADC
MODE[3:0]	4	Input	ADC operating mode	ADC
SYSCLK	1	Input	External system clock	
TVC[7:0]	8	Input	Clock divide control	ADC
STC[7:0]	8	Input	Sample time control	ADC
ADCSTART	1	Input	Start of conversion	ADC
PWRDWN	1	Input	Comparator power-down if 1	ADC
ADCRESET	1	Input	ADC initialize if 1	ADC
BUSY	1	Output	1 – Running conversion	ADC
CALIBRATE	1	Output	1 – Power-up calibration	ADC
DATAVALID	1	Output	1 – Valid conversion result	ADC
RESULT[11:0]	12	Output	Conversion result	ADC
TMSTBINT	1	Input	Internal temp. monitor strobe	ADC
SAMPLE	1	Output	1 – Analog input is sampled	ADC
VAREFSEL	1	Input	0 = Output internal voltage reference (2.56 V) to VAREF 1 = Input external voltage reference from VAREF and GNDREF	ADC
CHNUMBER[4:0]	5	Input	Analog input channel select	Input multiplexer
ACMCLK	1	Input	ACM clock	ACM
ACMWEN	1	Input	ACM write enable – active high	ACM
ACMRESET	1	Input	ACM reset – active low	ACM
ACMWDATA[7:0]	8	Input	ACM write data	ACM
ACMRDATA[7:0]	8	Output	ACM read data	ACM
ACMADDR[7:0]	8	Input	ACM address	ACM
CMSTB0 to CMSTB9	10	Input	Current monitor strobe – 1 per quad, active high	Analog Quad
GDON0 to GDON9	10	Input	Control to power MOS – 1 per quad	Analog Quad
TMSTB0 to TMSTB9	10	Input	Temperature monitor strobe – 1 per quad; active high	Analog Quad

Table 4-1 • Analog Block Pin Description (Continued)

Signal Name	Number of bits	Direction	Function	Location of Details
DAVOUT0, DACOUT0, DATOUT0 to DAVOUT9, DACOUT9, DATOUT9	30	Output	Digital outputs – 3 per quad	Analog Quad
DENAV0, DENAC0, DENAT0 to DENAV9, DENAC9, DENAT9	30	Input	Digital input enables – 3 per quad	Analog Quad
AV0	1	Input	Analog Quad 0 - Voltage	Analog Quad
AC0	1	Input		Analog Quad
AG0	1	Output		Analog Quad
AT0	1	Input		Analog Quad
ATRETURN01	1	Input	Temperature monitor return shared by Analog Quads 0 and 1	Analog Quad
AV1	1	Input	Analog Quad 1	Analog Quad
AC1	1	Input		Analog Quad
AG1	1	Output		Analog Quad
AT1	1	Input		Analog Quad
AV2	1	Input	Analog Quad 2	Analog Quad
AC2	1	Input		Analog Quad
AG2	1	Output		Analog Quad
AT2	1	Input		Analog Quad
ATRETURN23	1	Input	Temperature monitor return shared by Analog Quads 2 and 3	Analog Quad
AV3	1	Input	Analog Quad 3	Analog Quad
AC3	1	Input		Analog Quad
AG3	1	Output		Analog Quad
AT3	1	Input		Analog Quad
AV4	1	Input	Analog Quad 4	Analog Quad
AC4	1	Input		Analog Quad
AG4	1	Output		Analog Quad
AT4	1	Input		Analog Quad

Table 4-1 • Analog Block Pin Description (Continued)

Signal Name	Number of bits	Direction	Function	Location of Details
ATRETURN45	1	Input	Temperature monitor return shared by Analog Quads 4 and 5	Analog Quad
AV5	1	Input	Analog Quad 5	Analog Quad
AC5	1	Input		Analog Quad
AG5	1	Output		Analog Quad
AT5	1	Input		Analog Quad
AV6	1	Input	Analog Quad 6	Analog Quad
AC6	1	Input		Analog Quad
AG6	1	Output		Analog Quad
AT6	1	Input		Analog Quad
ATRETURN67	1	Input	Temperature monitor return shared by Analog Quads 6 and 7	Analog Quad
AV7	1	Input	Analog Quad 7	Analog Quad
AC7	1	Input		Analog Quad
AG7	1	Output		Analog Quad
AT7	1	Input		Analog Quad
AV8	1	Input	Analog Quad 8	Analog Quad
AC8	1	Input		Analog Quad
AG8	1	Output		Analog Quad
AT8	1	Input		Analog Quad
ATRETURN89	1	Input	Temperature monitor return shared by Analog Quads 8 and 9	Analog Quad
AV9	1	Input	Analog Quad 9	Analog Quad
AC9	1	Input		Analog Quad
AG9	1	Output		Analog Quad
AT9	1	Input		Analog Quad
RTCMATCH	1	Output	MATCH	RTC
RTCPSMMATCH	1	Output	MATCH connected to VRPSM	RTC
RTCXTLMODE[1:0]	2	Output	Drives XTLOSC RTCMODE[1:0] pins	RTC
RTCXTLSEL	1	Output	Drives XTLOSC MODESEL pin	RTC
RTCCLK	1	Input	RTC clock input	RTC

Functional Description

The Fusion datasheet, available at http://www.actel.com/techdocs/ds/default.aspx, contains a detailed functional description of the entire Analog System Builder.

Connecting Analog Ports

Each analog port must be connected to one of the following "virtual pads": INBUF_A. INBUF_DA or OUTBUF_A.

AV0, AC0, AT0, …, AV9, AC9 and AT9 are analog inputs that can be used either as analog or digital inputs. When used as an analog input, the analog input signal (e.g. AV0) must be connected to an INBUF_A, and the corresponding digital input enable (e.g. DENAV0) must be tied to 0.

When used as a digital input, the analog input must be connected to an INBUF_DA, and the corresponding digital input enable must be tied to 1.

All other analog inputs (ATRETURN01, ATRETURN23, ATRETURN45, ATRETURN67, and ATRETURN89, GND_REF) must be connected to INBUF_A.

Note: ATRETURN01, ATRETURN23, ATRETURN45, ATRETURN67, and ATRETURN89 must be connected to an INBUF_A, even though they have no function in the simulation model.

Analog outputs (AG0, …, AG9) must be connected to an OUTBUF_A instance.

VAREF is an inout pad and does not need to be connected to INBUF_A or OUTBUF_A.

Serialization

The analog ports are represented by a 1-bit wide port in both the Verilog and VHDL simulation models. Verilog modules and VHDL functions were developed to drive a real value through a 1-bit port and to read an analog value from a 1-bit port. The Analog System Builder macro contains embedded read and drive logic to read from the analog input and drive the analog output, respectively.

The drive module/function converts a real value into a 64-bit value, serializes it and streams it in zero simulation time, using delta delays. The read module/function deserializes a 64-bit stream into a 64-bit value and converts it into a real value.

Verilog

Two Verilog modules (drive_analog_io and read_analog_io) are available to drive an analog input and read an analog output. You must instantiate a drive_analog_io ror each analog, and a read_analog_io for each analog output. The read_analog_io starts as soon as there is a non 'Z' data bit on the module input pin. All read and drive operations happen in zero time (delta delays).

Example: drive_analog_io with an INBUF_A and OUTBUF_A instantiation

```
wire            AV0_stream_pad, AV0_stream_y, AG0_d, AG0_pad;
```

```
wire [63:0] AG0_VECTOR;
real            AV0_real;
drive_analog_io drive_AV0 ( $realtobits(AV0_real),  AV0_stream_pad );
INBUF_A inbuf_AV0 ( .Y(AV0_stream_y), .PAD (AV0_stream_pad) );
AB ab_inst (
    …
    .AV0   (AV0_stream_y),
    …
    .AG0   (AG0_d),
   …
   );
OUTBUF_A outbuf_AG0 ( .PAD(AG0_pad), . D (AG0_d) );
initial
begin
  AV0_real <= 1.28;
end
```

VHDL

Similarly, two VHDL functions (drive_analog_input and read_analog_input) are available to drive an analog input and read an analog output. These function are part of the analog_io VHDL package. The read_analog_io starts as soon as there is a non 'Z' data bit on the function input pin. All read and drive operations happen in zero time (delta delays).

Example: drive_analog_io and read_analog_io with an INBUF_A instantiation

```
…
use work.analog_io.all;
…
signal varef_real : real
signal AT0_real : real
signal varef_serial_out  : std_logic;
signal AT0_stream_pad : std_logic;
signal AT0_stream_y : std_logic;
component INBUF_A
   port(
   …
   );
end component;
component AB
   port(
   …
   );
```

```
end component;
read_varef : process
begin
  wait until varef_serial_out /= 'Z';
  read_analog_input( varef_serial_out, varef_real);
end process read_varef;
drive_quads : process
begin
    AT0_real <= 2.18;
    drive_analog_input( AT0_real, AT0_stream_pad );
end process drive_quads;
inbuf_a_at0 : INBUF_A
    port map (
    PAD  => AT0_stream_pad,
    Y      => AT0_stream_y,
    …
    );
ab_inst: AB
    port map (
    VAREF => varef_serial_out,
    …
    AV0     => AT0_stream_y,
    …
    );
```

Analog System Builder Signal Description

System Level Signals

Table 4-2. System Level Signals

Name	Type	Description
SYS_CLK	Input	System Clock: reference clock for all internal logic (100 MHz maximum).
SYS_RESET	Input	Active Low System Reset

Table 4-2. System Level Signals (Continued)

Name	Type	Description
VAREF	Inout	Voltage Reference. Connects to external voltage for external voltage refernce. Returns the internal Vref in the case of internal voltage refernce. Must be connected to a top-level port without any I/Os.
Analog Input Channels	Input	User specified port names. Analog Channels being used
INIT_ADDR[MAX_ADDR-1:0]	Input	Initialization Address from Flash Memory Block. Maximum address location required by the largest of the Analog system RAMs
INIT_DATA[8:0]	Input	Initialization data from Flash Memory Block
INIT_DONE	Input	Initialization done signal from Flash Memory Block
INIT_ACM_WEN	Input	Enable the ACM for Initialization from Flash Memory Block System
INIT_ASSC_WEN	Input	Enable the ASSC for Initialization from Flash Memory Block
INIT_EV_WEN	Input	Enable the SMEV for Initialization from Flash Memory Block
INIT_TR_WEN	Input	Enable the SMTR for Initialization from Flash Memory Block
Analog Output Channels	Output	Output Gate Drivers
Input Channel Compare Flags	Output	Flags required

Real Time Counter (RTC) Signals

Table 4-3. RTC Signals in Analog System Builder

Name	Type	Description
RTCCLK	Input	RTC Clock. Must be driven by the XTLOUT of the XTLOSC macro .
RTCXTLSEL	Output	Crystal Oscillator Select. Must be connected to XTLSEL on XTLOSC macro.
RTCXTLMODE[1:0]	Output	Crystal Oscillator Mode Select. Must be connected to RTCXTLMODE on XTLOSC macro.
RTCMATCH	Output	Result of Match
RTCPSMMATCH	Output	Result of Match. Must be connected to RTCMATCH of VRPSM macro. Exposed only if On Match Enable Voltage Regulator is checked.
ACMCLK	Input	Clock for writing to ACM. Used with INIT_ADDR and INIT_DATA to update the RTC Match Register and Counter. Exposed only if Enable Dynamic Update is checked.
ACMRDATA[7:0]	Output	ACM Read Data Bus. Used with ACMCLK and INIT_ADDR to read the contents of ACM. Exposed only if Enable Dynamic Update is checked.

Status Signals

Table 4-4. Status Signals in the Analog System Builder

Name	Type	Description
DATAVALID	Output	This indicates the data from ADC is valid
ASSC_DONE	Output	This indicates ASSC finished processing the current sample
SMEV_DONE	Output	This indicates SMEV finished processing the current sample
SMTR_DONE	Output	This indicates SMTR finished processing the current sample
ASSC_CHSAT	Output	Sampled Channel Saturated: This output indicates that the current channel sampled by the ADC has a value that is saturated (too high for the current 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 pre-scale value for the particular analog input pad).
ASSC_CHALTD	Output	Channel Selector Latched: This signal indicates that the ADC_CHNR[4:0] (ADC channel selector) value has been latched. This output may optionally be used external to the analog interface soft IP blocks by the user.

External Trigger Signals

These signals are exposed when there is at least one slot occupied in the Jump Slots section of the Sequencer..

Table 4-5. External Trigger Signals in Analog System Builder

Name	Type	Description
ASSC_XMODE	Input	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. 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.
ASSC_XTRIG	Input	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. If the ASSC_XMODE input is logic 0 (default operation for automated sequencing), this input is ignored. 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.
ASSC_SEQJUMP	Input	Sequence Jump Enable: Setting this signal to logic 1will jump to the sequence number indicated in the ASSC_SEQIN[TS_WIDTH-1:0] input pins after the current sequence timeslot has completed.
ASSC_SEQIN[TS_WIDTH-1:0]	Input	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.
ASSC_SEQOUT[TS_WIDTH-1:0]	Output	Sequence Number Out: These outputs denote the current sequence timeslot.
ASSC_SEQCHANGE	Output	Sequence Change: This output indicates that the ASSC_SEQOUT[TS_WIDTH-1:0] outputs are about to change after the very next rising edge of CLK.

User RAM access signals

You can monitor the contents of ASSC, SMEV and SMTR RAM to read out the ADC output, averaged signals and results of threshold comparison.

Table 4-6. ASSC RAM Access in Analog System Builder

Name	Type	Description
USER_ASSC_ADDR [8:0]	Input	User RAM Address: These address signals can be controlled by you to allow read access from the A-port of the 512x9 ASSC RAM. If unused, tie these signals off to logic 0 or logic 1 values.
USER_ASSC_RD	Input	User RAM Read Enable: This control signal can be controlled by you to allow 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 should be tied off to logic 0. **Warning: The ASSC_RAM_BUSY signal must be inactive at logic 0 while this signal is activated. If not, 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	ASSC RAM Busy: This output signal indicates that either the Flash Memory Block or the System Monitor Evaluation Phase State Machine Soft IP block is busy accessing the A-port of the ASSC RAM.
ASSC_RAM_WR_B USY_B	Output	ASSC Busy Writing: This active-high signal is for status monitoring and indicates that the ASSC block is busy writing to the B port of its dual-port RAM.

Table 4-7. SMEV RAM Access in Analog System Builder

Name	Type	Description
USER_EV_ADDR[EV_ASIZE-1:0]	Input	User RAM Address: These address signals can be controlled by you to allow read access from the 512x9 SMEV RAM(s). If unused, tie these signals off to logic 0 or logic 1 values.
USER_EV_RD	Input	User RAM Read Enable: This control signal can be controlled by the user to allow read access from the A-port of the 512x9 SMEV dual-port RAM(s) (the user will need to connect to the EV_RAM_DO_A[8:0] port for read data). If unused, this signal should be tied off to logic 0. **Warning: The USER_EV_RAM_BUSY signal must be inactive at logic 0 while this signal is 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	SMEV RAM Busy: This output signal indicates that either the Flash Memory Block or the SMTR block is busy accessing the A-port of the SMEV RAM(s).
EV_RAM_WR_BUSY_B	Output	SMEV Busy Writing: This active-high signal is for user status monitoring and indicates that the SMEV block is busy writing to the B port of its dual-port RAM.

Table 4-8. SMTR RAM Access in Analog System Builder

Name	Type	Description
USER_TR_ADDR[TR_ASIZE-1:0]	Input	User RAM Address: These address signals can be controlled by the user to allow read access from the 512x9 SMTR RAM(s). If unused, these signals should be tied off to logic 0 or logic 1 values.

Table 4-8. SMTR RAM Access in Analog System Builder (Continued)

Name	Type	Description
USER_TR_RD	Input	User RAM Read Enable: This control signal can be controlled by the user to allow read access from the A-port of the 512x9 SMTR dual-port RAM(s) (the user will need to connect to the TR_RAM_DO_A[8:0] port for read data). If unused, this signal should be tied off to logic 0. **Warning: The USER_TR_RAM_BUSY signal must be inactive at logic 0 while this signal is activated, otherwise, the data read from the A-port of the SMTR RAM(s) will not be from the USER_TR_ADDR[TR_ASIZE-1:0] address.**
USER_TR_RAM_BUSY	Output	SMTR RAM Busy: This output signal indicates that the Flash Memory Block is busy accessing the A-port of the SMTR RAM(s). This signal can optionally be used by user logic external to the analog interface soft IP blocks.
TR_RAM_WR_BUSY_B	Output	SMTR Busy Writing: This active-high signal is for status monitoring and indicates that the SMTR block is busy writing to the B port of its dual-port RAM. It can be left unconnected.

Tips and Tricks

Design Flow Considerations

For details on the design flow considerations for the Analog block, refer to the Fusion Design Flow Tutorial (http://www.actel.com/documents/FusionDesignFlow_Tutorial.pdf). It is important to note that you must generate Analog System before generating theFlash Memory system.

Design Considerations

- System Clock <100MHz - For ADC initialization, the clock freq (ACM Clock) <10MHz
- Default Settling time should work for less than 5 threshold flags per channel - If more than 4 flags per channel, should increase Settling Time and/or System Clock frequency to resolve the interlock clock issue
- Voltage channel used in Current Monitor can be used for Voltage monitor as well

More flags in each channel will demand an increase in settling time or system frequency

- Sampling the same channel back to back when there is multiple channels in the analog system, may result in interlock issue - Avoid this type of sample sequence or increase settling time or system frequency.

Comparing Direct Analog Input and Prescaler Input

You can bring analog signals into the FPGA through either direct analog inputs or prescalers.

If the input value is less than the ADC reference voltage (2.56~3.3V), and high accuracy is critical for the application, choose direct input. The prescaler OpAmp 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 prescaler input.

Consider the following scenarios

1. If the input voltage is <2.0V, and you enter the max input voltage = 2V in SmartGen Analog System Builder. The prescaler (2.046V range) will be automatically selected by SmartGen to increase the resolution (10bit, 1LSB=2mV), but the accuracy could be affected by the Prescaler OpAmp gain errors and offset errors.
2. Enter the max input voltage = 2.56V, and direct input is selected. For a 10bit ADC, 1LSB=2.5 mV (less resolution than prescaler 10bit 1LSB=2 mV). But this avoids the gain error and offset error introduced by the prescaler.
3. If you are using a prescaler input, enter the maximum input voltage in the SmartGen GUI. SmartGen selects the lowest Full Scale Voltage range that exceeds the maximum input voltage. Table 4-9 shows the available Full Scale Voltage ranges and the corresponding resolutions. The prescaler is configured automatically by SmartGen to achieve the highest possible resolution.

Table 4-9. Full Scale Voltage Ranges and Corresponding Resolution

Control Lines Bx[2:0]	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 (V)	Range Name (V)
000*	0.15625	64	16	4	16.368	16
001	0.3125	32	8	2	8.184	8
010*	0.625	16	4	1	4.092	4
011	1.25	8	2	0.5	2.046	2
100	2.5	4	1	0.25	1.023	1
101	5.0	2	0.5	0.125	0.5115	0.5

Table 4-9. Full Scale Voltage Ranges and Corresponding Resolution

Control Lines Bx[2:0]	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 (V)	Range Name (V)
110	10.0	1	0.25	0.0625	0.255575	0.25
111	20.0	0.5	0.125	0.03125	0.127875	0.125

ADC Sample Rate Sets Limit for Input Signal Frequency

In direct analog input mode, the input signal frequency should not be more than half of the sample rate. If the input signal toggles faster than half of the sample rate, then the ADC cannot capture enough samples within a cycle of the input to accurately represent the input signal.

In other words, if the sample rate is 500 ksps, then the input signal frequency should not be more than 250 kHz.

In analog input prescaler mode, the input signal frequency must be less than or equal to 100 kHz or half of the sample rate, whichever is lower.

Thus, if the sample rate is 300 ksps, then the input frequency should be 100 kHz (100 kHz < 1/2 300 ksps). If the sample rate is 150 ksps, then input frequency should be 75 kHz. (1/2 150 ksps < 100 kHz).

Filtering Analog Inputs

In some cases, where the inputs are very low frequency, and the electrical environment is not very noisy, it may be possible to get away 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 over sampling 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 by SmartGen by setting the Digital Filtering factor in the Analog System Builder (ASB) to specify how many samples are averaged (When the factor = N, 2**N 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.

Finally, 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.

How do the Assert/De-assert Samples Settings Affect the Design?

Assert or de-assert samples function as a digital state filter. It may be used to remove glitches in the threshold flags. Typically, it is desirable to have a smaller value in "Assert Samples" to catch any abnormal situation quickly and send out the alerts, then have a relatively larger value in "De-assert Samples" to indicate that the system is back to normal and stable (Figure 4-9).

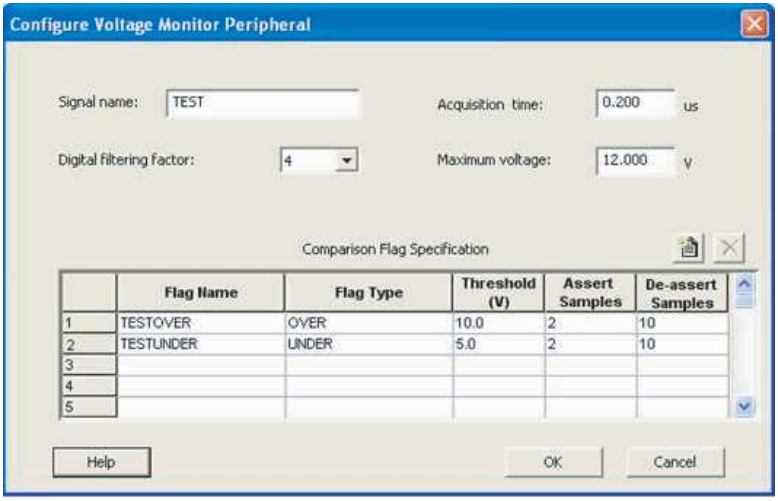


Figure 4-9. Current Voltage Monitor Peripheral

Analog Block Interconnection

- Analog Block has 30 input ports which can be configured as:
  - Analog inputs serving the 10 channels for measuring Temperature, Voltage, Current or
  - Slow digital input to the FPGA logic
- In addition, there are 6 special analog-only inputs:
  - 5 for the ATRETURN functionality
  - 1 for the GNDREF.
- There are 10 analog outputs for the gate driver of the quads.
- Analog Block has 4 dedicated ports for the RTC functionality.
- Placement: There is only one fixed position on the device. Run Compile to automatically place theAnalog Block macro. You should not move the macro to some other location.

Channel Inputs

There are 30 inputs available on the Analog Block macro. The pin names are:

- AV0 to AV9 for Voltage
- AC0 to AC9 for Current
- AT0 to AT9 for Temperature

Each of these can be used as an independent analog input to access the current/temperature/voltage quad.

You can also use these for providing a low-skew digital input to the FPGA logic.

Use Model

For digital configurations, use INBUF_DA to connect to these ports. For analog configurations, use INBUF_A to connect to these ports.

Maximum that can be used for a design: 30.

The fanout of the net connecting these buffers to the analog block should always be 1.

Placement of all these special buffers is automatically done when you run Compile. The positions are determined by the connection to the Analog Block. You cannot change the position of these macros.

To configure these inputs as digital input connections, connect the corresponding Enable (DEN…) at the input side to VCC. The corresponding output (DAVOUT) in the Analog Block must be driving some logic.

For example, if the INBUF_DA connected to the AV0 port of Analog Block is to be used for Digital purpose then DENAV0 should be connected to VCC and DAVOUT0 must be connected to some logic (Figure 4-10).

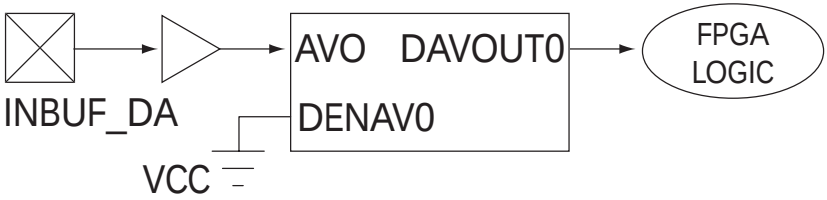


Figure 4-10. AV0 used as FPGA input

To configure as an analog input, you must connect the corresponding Enable (DEN…) at the input side to GND. The corresponding output (DAVOUT.) in the Analog Block must be left hanging.

For example, if the AV0 port of Analog Block is to be used for Analog purpose, DENAV0 must be connected to ground and DAVOUT0 must be left dangling (Figure 4-11).

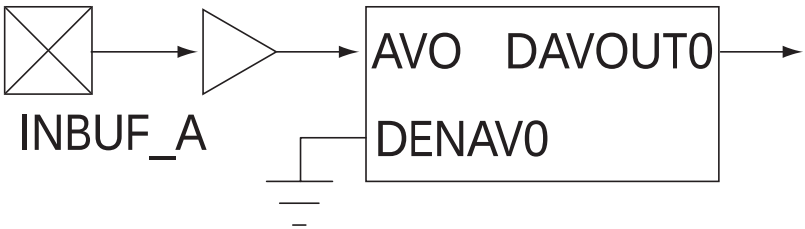


Figure 4-11. AV0 used as Analog Input

Analog-only Inputs

The following ports are analog-only:

- ATRETURN01 to ATRETURN89
- GNDREF

Use Model

Use 'INBUF_A' to connect to these ports.

Maximum that can be used for a design: 6

The fanout of the net connecting these buffers to the analog block should always be 1.

Placement of all these special buffers is automatically done by when you run Compile. The positions are determined by the connection to the Analog Block. You cannot change the position of these macros.

Analog Outputs

There are 10 analog outputs: The pin names are AG0 to AG9.

Use OUTBUF_A to connect to these ports.

Maximum that can be used for a design: 10

The fanout of the net connecting these buffers to the analog block should always be 1.

Placement of all these special buffers is automatically done when you run Compile. The positions are determined by the connection to the Analog Block. You cannot change the position of these macros.

VAREF port

The VAREF port is a special INPUT port with a PAD property on it (Figure 4-12). Synthesis tool will not put an input/output buffer in front of this port.

Use Model

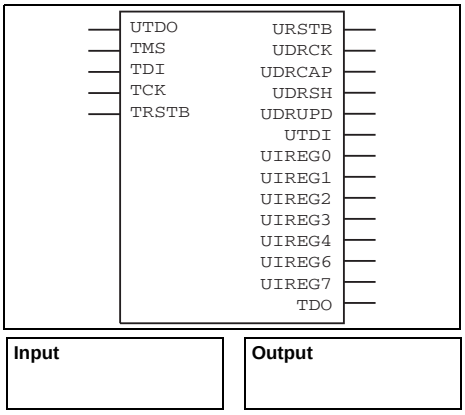
You must ensure that this port is directly connected to a design port.

Figure 4-12. VAREF Port

5 User JTAG

The UJTAG macro is a special purpose macro. It is provided to give you access to the user JTAG circuitry on board the chip. You must instantiate a UJTAG macro in your design if you plan to make use of the user JTAG feature. It is identical to the APA and A500K UJTAG macro.

UJTAG Fusion, ProASIC3, ProASIC3E



For more details on UJTAG applications and usage, refer to the following application notes:

- UJTAG Applications in ProASIC3/E Devices at http://www.actel.com/documents/PA3_E_UJTAG_AN.pdf
- How to use UJTAG at http://www.actel.com/documents/Flash_UJTAG_AN.pdf

6

Product Support

Actel backs its products with various support services including Customer Service, a Customer Technical Support Center, a web site, an FTP site, electronic mail, and worldwide sales offices. This appendix contains information about contacting Actel and using these support services.

Customer Service

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

From Northeast and North Central U.S.A., call **650.318.4480**
From Southeast and Southwest U.S.A., call **650. 318.4480**
From South Central U.S.A., call **650.318.4434**
From Northwest U.S.A., call **650.318.4434**
From Canada, call **650.318.4480**
From Europe, call **650.318.4252** or **+44 (0) 1276 401 500**
From Japan, call **650.318.4743**
From the rest of the world, call **650.318.4743**
Fax, from anywhere in the world **650.318.8044**

Actel Customer Technical Support Center

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

Actel Technical Support

Visit the Actel Customer Support website (www.actel.com/custsup/search.html) for more information and support. Many answers available on the searchable web resource include diagrams, illustrations, and links to other resources on the Actel web site.

Website

You can browse a variety of technical and non-technical information on Actel's home page, at www.actel.com.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. Several ways of contacting the Center follow:

Email

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

The technical support email address is tech@actel.com.

Phone

Our Technical Support Center answers all calls. The center retrieves information, such as your name, company name, phone number and your question, and then issues a case number. The Center then forwards the information to a queue where the first available application engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific Time, Monday through Friday. The Technical Support numbers are:

650.318.4460
800.262.1060

Customers needing assistance outside the US time zones can either contact technical support via email (tech@actel.com) or contact a local sales office. Sales office listings can be found at www.actel.com/contact/offices/index.html.

Index

For more information about Actel's products, visit our website at http://www.actel.com

Actel Corporation *• 2061 Stierlin Court • Mountain View, CA 94043 USA*
Customer Service: 650.318.1010 • Customer Applications Center: 800.262.1060

Actel Europe Ltd*. • Dunlop House, Riverside Way • Camberley, Surrey GU15 3YL • United Kingdom*
Phone +44 (0) 1276 401 450 • Fax +44 (0) 1276 401 490

Actel Japan *• EXOS Ebisu Bldg. 4F • 1-24-14 Ebisu Shibuya-ku • Tokyo 150 • Japan*
Phone +81.03.3445.7671 • Fax +81.03.3445.7668 • www.jp.actel.com

Actel Hong Kong *• Suite 2114, Two Pacific Place • 88 Queensway, Admiralty Hong Kong*
Phone +852 2185 6460 • Fax +852 2185 6488 • www.actel.com.cn

5-02-00063-1/02/06