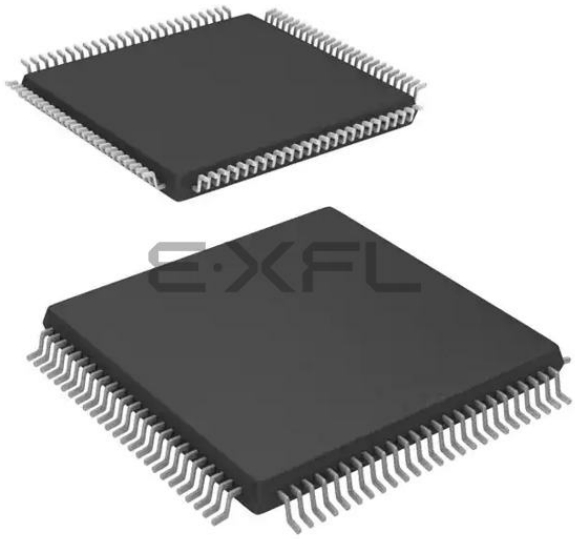


Welcome to E-XFL.COM

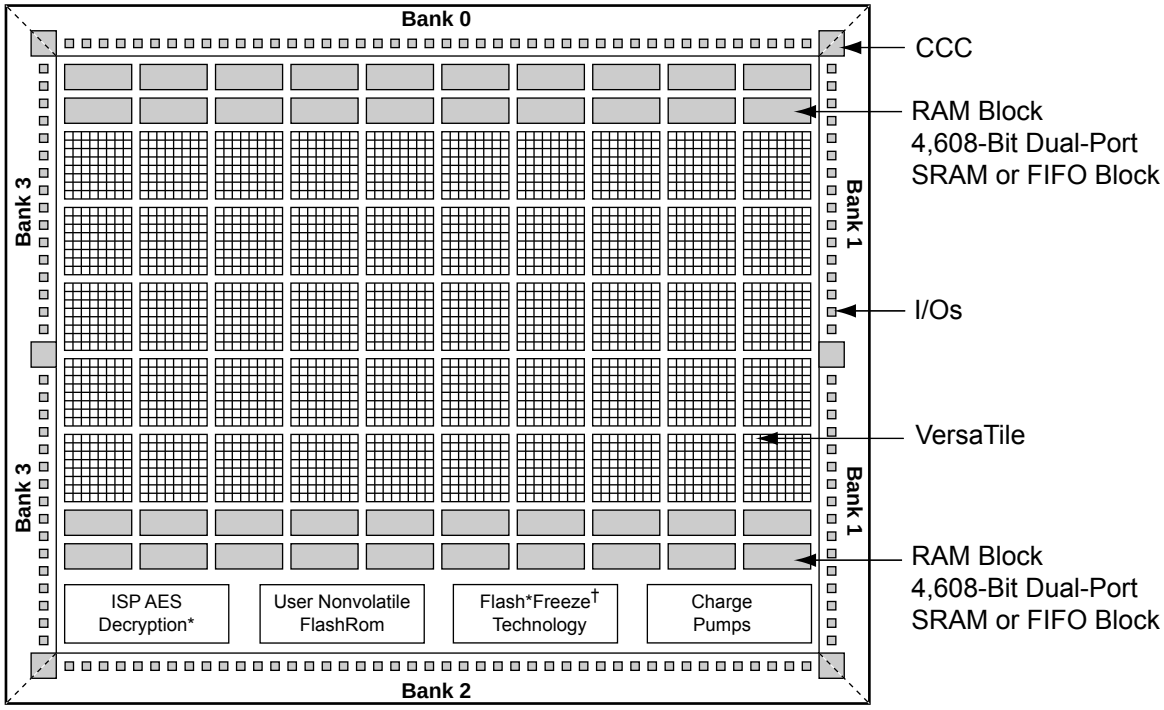
Understanding Embedded - FPGAs (Field Programmable Gate Array)

Embedded - FPGAs, or Field Programmable Gate Arrays, are advanced integrated circuits that offer unparalleled flexibility and performance for digital systems. Unlike traditional fixed-function logic devices, FPGAs can be programmed and reprogrammed to execute a wide array of logical operations, enabling customized functionality tailored to specific applications. This reprogrammability allows developers to iterate designs quickly and implement complex functions without the need for custom hardware.

Applications of Embedded - FPGAs

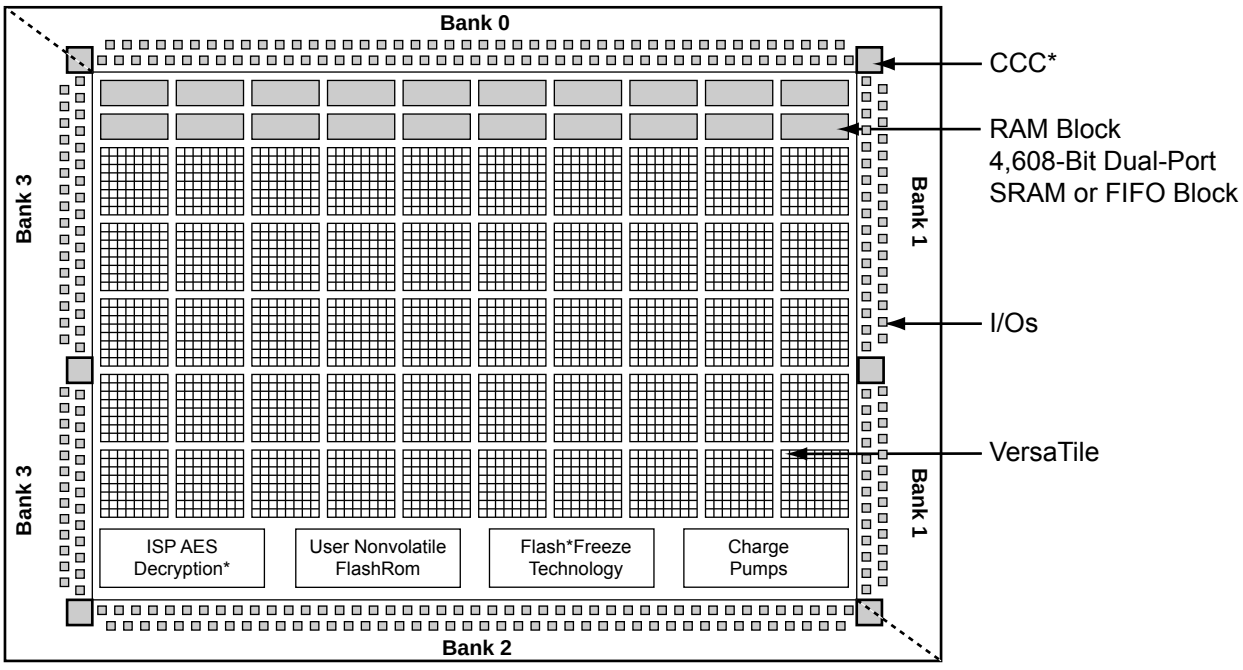
The versatility of Embedded - FPGAs makes them indispensable in numerous fields. In telecommunications.

Details	
Product Status	Active
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	18432
Number of I/O	71
Number of Gates	60000
Voltage - Supply	1.425V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	100-TQFP
Supplier Device Package	100-VQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pn060-1vq100i



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

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



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

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

Related Documents

User's Guides

Designer User's Guide

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

List of Changes

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

Date	Changes	Page
August 2012	The "I/O State of Newly Shipped Devices" section is new (SAR 39542).	14
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 1-1 • Flash-Based FPGAs.	10
	Figure 1-2 • IGLOO and ProASIC3 nano Device Architecture Overview with Two I/O Banks (applies to 10 k and 30 k device densities, excluding IGLOO PLUS devices) through Figure 1-5 • IGLOO, IGLOO nano, ProASIC3 nano, and ProASIC3/L Device Architecture Overview with Four I/O Banks (AGL600 device is shown) are new.	11, 12
	Table 1-4 • IGLOO nano and ProASIC3 nano Array Coordinates is new.	17
v1.3 (October 2008)	The title of this document was changed from "Core Architecture of IGLOO and ProASIC3 Devices" to "FPGA Array Architecture in Low Power Flash Devices."	9
	The "FPGA Array Architecture Support" section was revised to include new families and make the information more concise.	10
	Table 1-2 • IGLOO and ProASIC3 Array Coordinates was updated to include Military ProASIC3/EL and RT ProASIC3 devices.	16
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 1-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	10
v1.1 (March 2008)	Table 1-1 • Flash-Based FPGAs and the accompanying text was updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "Device Overview" section are new.	10
	The "Device Overview" section was updated to note that 15 k devices do not support SRAM or FIFO.	11
	Figure 1-6 • IGLOO PLUS Device Architecture Overview with Four I/O Banks is new.	13
	Table 1-2 • IGLOO and ProASIC3 Array Coordinates was updated to add A3P015 and AGL015.	16
	Table 1-3 • IGLOO PLUS Array Coordinates is new.	16

Global Resource Support in Flash-Based Devices

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

***Table 3-1* • Flash-Based FPGAs**

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

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

IGLOO Terminology

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

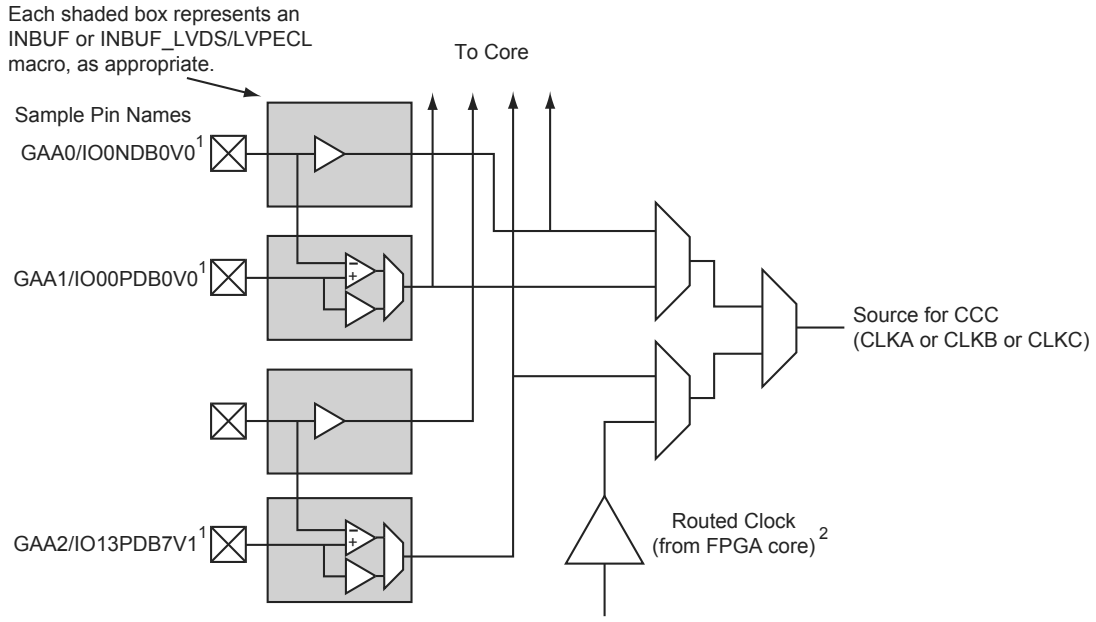
ProASIC3 Terminology

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

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

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

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



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

Figure 3-5 • **Global I/O Overview**

***Table 3-2* • Chip Global Pin Name**

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

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

Spine Architecture

The low power flash device architecture allows the VersaNet global networks to be segmented. Each of these networks contains spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside its region. The nine spines available in a vertical column reside in global networks with two separate regions of scope: the quadrant global network, which has three spines, and the chip (main) global network, which has six spines. Note that the number of quadrant globals and globals/spines per tree varies depending on the specific device. Refer to Table 3-4 for the clocking resources available for each device. The spines are the vertical branches of the global network tree, shown in Figure 3-3 on page 34. Each spine in a vertical column of a chip (main) global network is further divided into two spine segments of equal lengths: one in the top and one in the bottom half of the die (except in 10 k through 30 k gate devices).

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

Each spine and its associated ribs cover a certain area of the device (the "scope" of the spine; see Figure 3-3 on page 34). Each spine is accessed by the dedicated global network MUX tree architecture, which defines how a particular spine is driven—either by the signal on the global network from a CCC, for example, or by another net defined by the user. Details of the chip (main) global network spine-selection MUX are presented in Figure 3-8 on page 44. The spine drivers for each spine are located in the middle of the die.

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

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

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

Spine Access

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

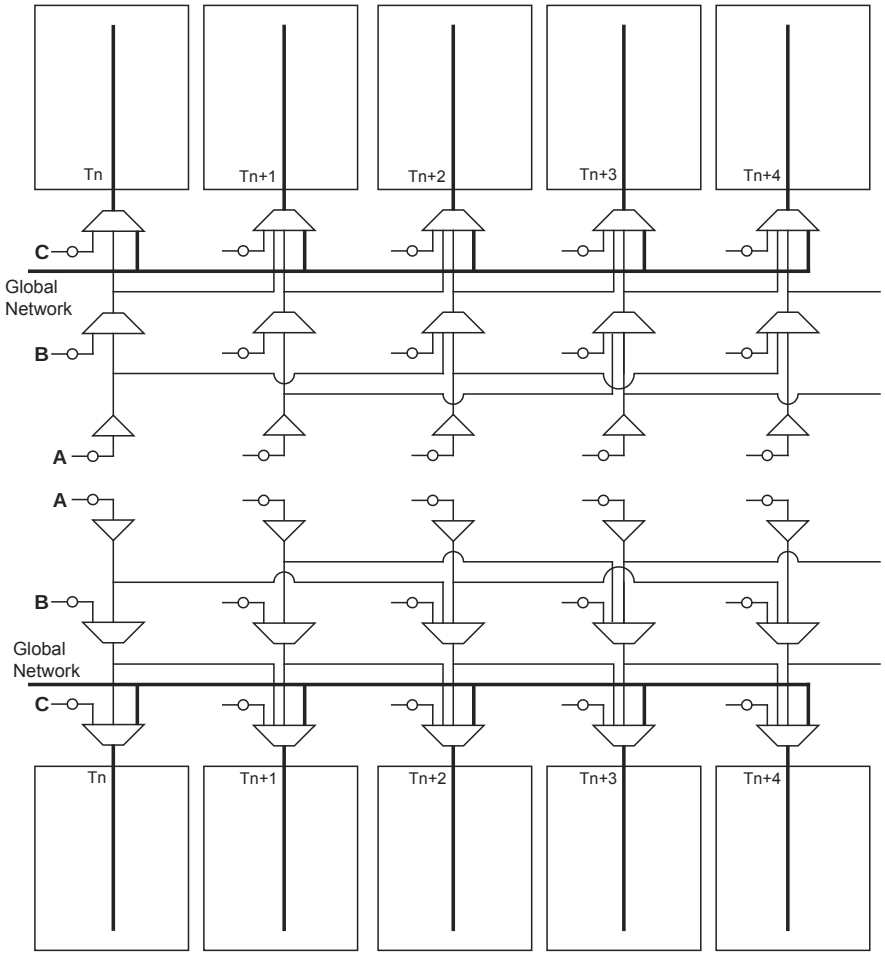


***Figure 3-7* • Chip Global Aggregation**

A spine is also called a local clock network, and is accessed by the dedicated global MUX architecture. These MUXes define how a particular spine is driven. Refer to Figure 3-8 on page 44 for the global MUX architecture. The MUXes for each chip global spine are located in the middle of the die. Access to the top and bottom chip global spine is available from the middle of the die. There is no control dependency between the top and bottom spines. If a top spine, T1, of a chip global network is assigned to a net, B1 is not wasted and can be used by the global clock network. The signal assigned only to the top or bottom spine cannot access the middle two rows of the architecture. However, if a spine is using the top and bottom at the same time (T1 and B1, for instance), the previous restriction is lifted.

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

Step 1

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

Cell usage:

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

Step 2

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

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

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

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

Introduction

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

Overview of Clock Conditioning Circuitry

In Fusion, IGLOO, and ProASIC3 devices, the CCCs are used to implement frequency division, frequency multiplication, phase shifting, and delay operations. The CCCs are available in six chip locations—each of the four chip corners and the middle of the east and west chip sides. For device-specific variations, refer to the "Device-Specific Layout" section on page 78.

The CCC is composed of the following:

- PLL core
- 3 phase selectors
- 6 programmable delays and 1 fixed delay that advances/delays phase
- 5 programmable frequency dividers that provide frequency multiplication/division (not shown in Figure 4-6 on page 71 because they are automatically configured based on the user's required frequencies)
- 1 dynamic shift register that provides CCC dynamic reconfiguration capability

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

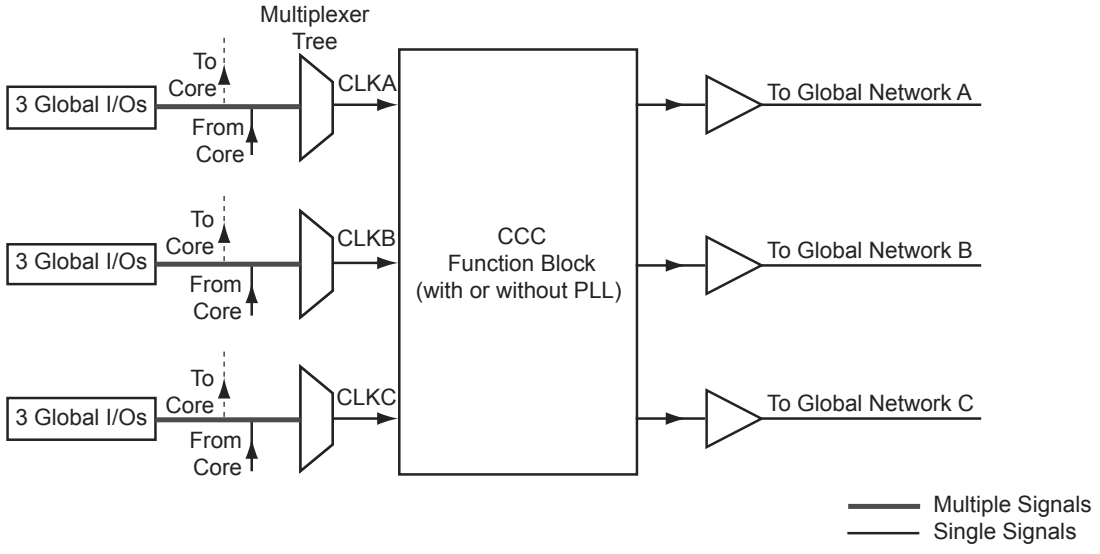


***Figure 4-1* • Overview of the CCCs Offered in Fusion, IGLOO, and ProASIC3**

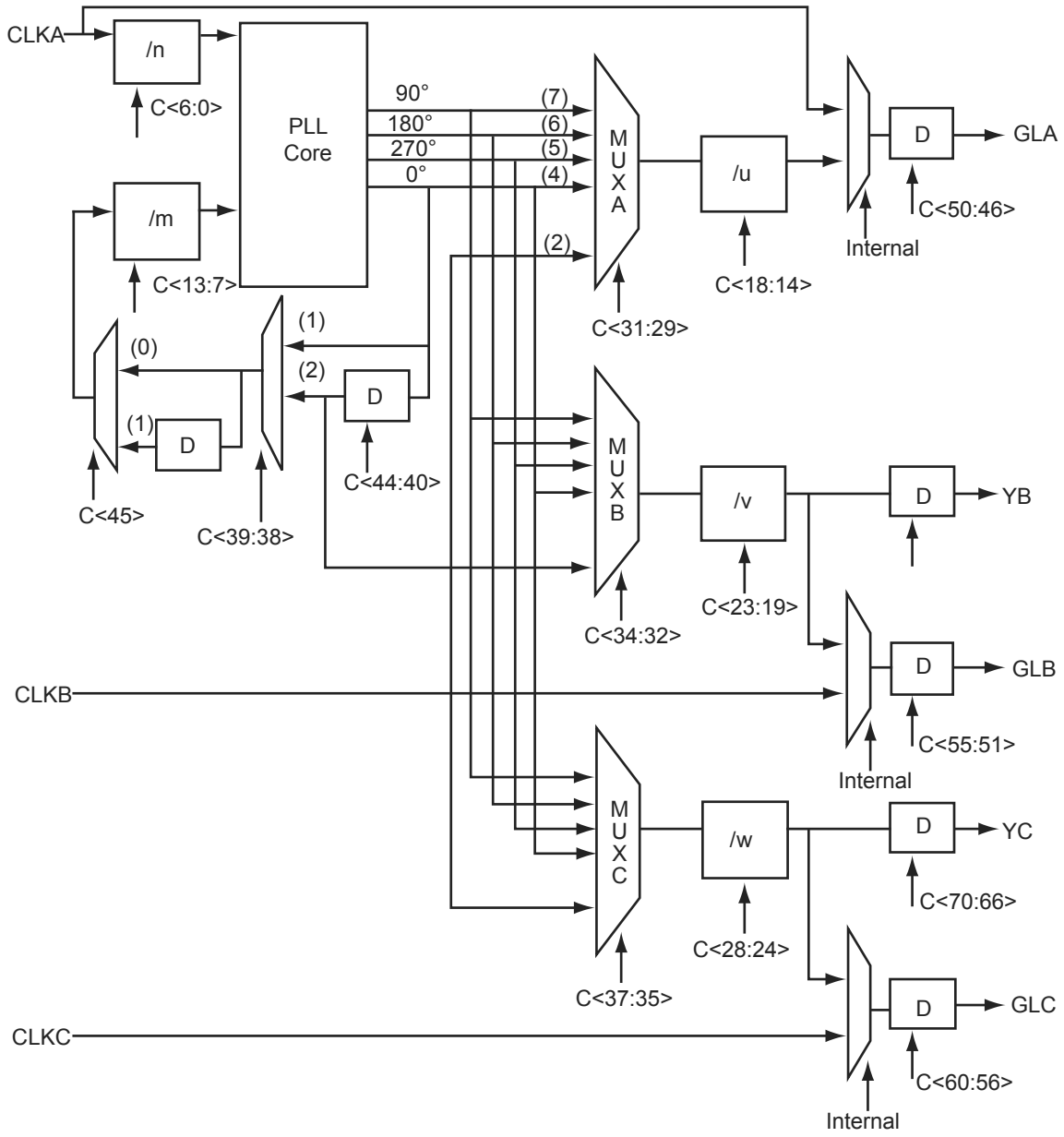


***Figure 4-22* • CCC Block Control Bits – Graphical Representation of Assignments**

Software Configuration

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

Static PLL Configuration

The newly implemented Visual PLL Configuration Wizard feature provides the user a quick and easy way to configure the PLL with the desired settings (Figure 4-23). The user can invoke SmartGen to set the parameters and generate the netlist file with the appropriate flash configuration bits set for the CCCs. As mentioned in "PLL Macro Block Diagram" on page 69, the input reference clock CLKA can be configured to be driven by Hardwired I/O, External I/O, or Core Logic. The user enters the desired settings for all the parameters (output frequency, output selection, output phase adjustment, clock delay, feedback delay, and system delay). Notice that the actual values (divider values, output frequency, delay values, and phase) are shown to aid the user in reaching the desired design frequency in real time. These values are typical-case data. Best- and worst-case data can be observed through static timing analysis in SmartTime within Designer.

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

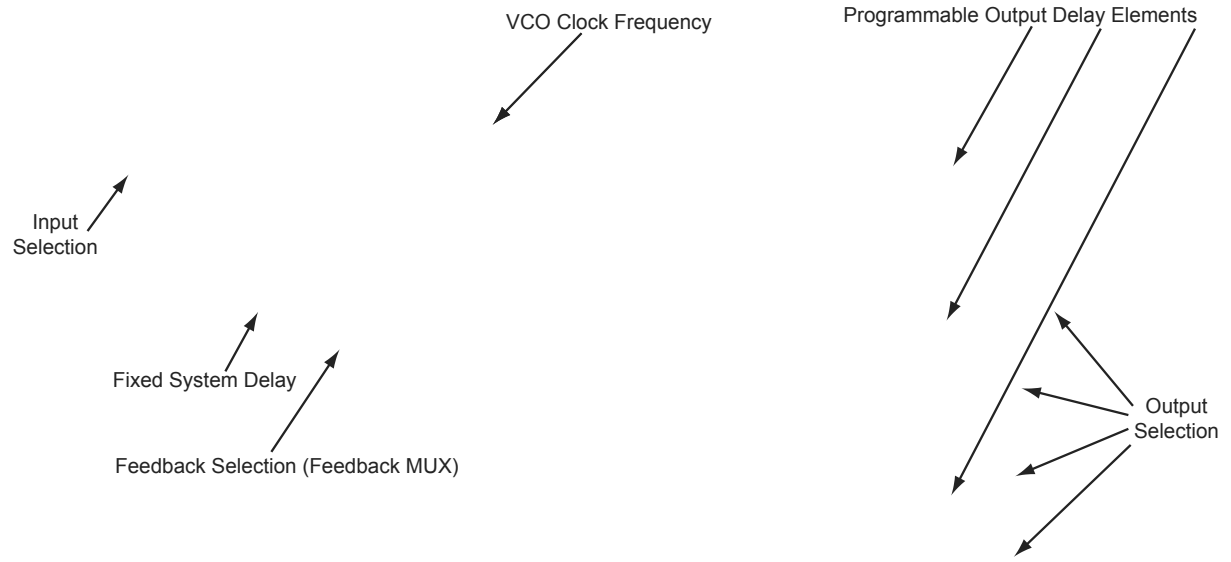


Figure 4-23 • **Visual PLL Configuration Wizard**

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

Using internal feedback, we know from EQ 4-1 on page 86 that the maximum achievable output frequency from the primary output is

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

EQ 4-5

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

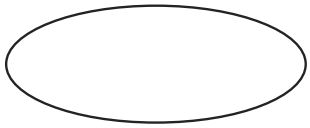


Figure 4-35 • **First-Stage PLL Showing Input of 2 MHz and Output of 256 MHz**

A second PLL can be connected serially to achieve the required frequency. EQ 4-1 on page 86 to EQ 4-3 on page 86 are extended as follows:

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

EQ 4-6

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

EQ 4-7

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

EQ 4-8

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

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

EQ 4-9

Figure 4-36 on page 111 shows the settings of the second PLL. When configuring the second PLL (or any subsequent-stage PLLs), specify the input to be Core Logic–Driven. This generates a netlist with the second PLL routed internally from the core. Do not specify the input to be Hardwired I/O–Driven or External I/O–Driven, as these options prohibit the connection from the output of the first PLL to the input of the second PLL.

FlashROM Design Flow

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

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

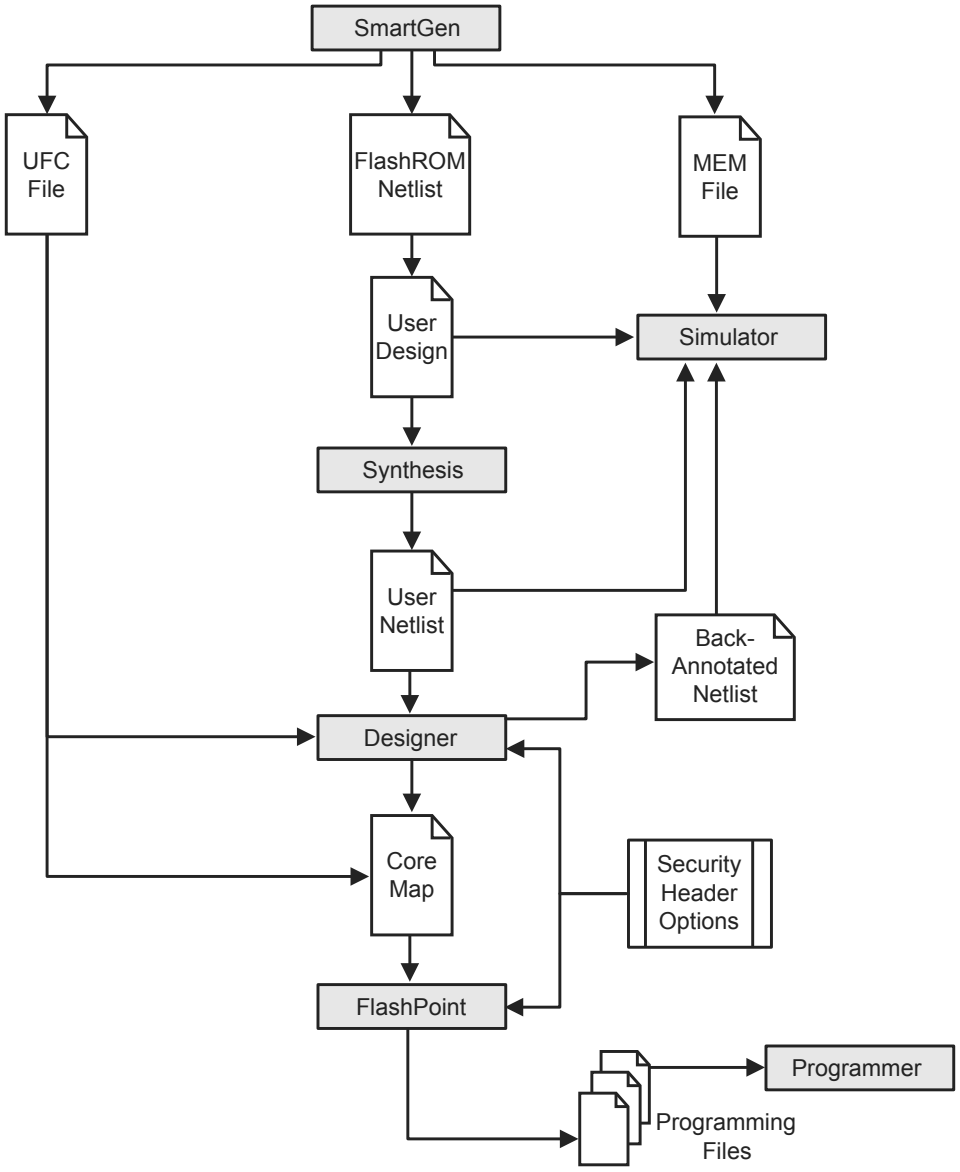


***Figure 5-9* • FlashROM Design Flow**

FlashROM Generation and Instantiation in the Design

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

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

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

Table 6-8 and Table 6-9 show the maximum potential width and depth configuration for each device. Note that 15 k and 30 k gate devices do not support RAM or FIFO.

***Table 6-8* • Memory Availability per IGLOO and ProASIC3 Device**

Device		RAM Blocks	Maximum Potential Width[1]		Maximum Potential Depth[2]	
IGLOO IGLOO nano IGLOO PLUS	ProASIC3 ProASIC3 nano ProASIC3L		Depth	Width	Depth	Width
AGL060 AGLN060 AGLP060	A3P060 A3PN060	4	256	72 (4×18)	16,384 (4,096×4)	1
AGL125 AGLN125 AGLP125	A3P125 A3PN125	8	256	144 (8×18)	32,768 (4,094×8)	1
AGL250 AGLN250	A3P250/L A3PN250	8	256	144 (8×18)	32,768 (4,096×8)	1
AGL400	A3P400	12	256	216 (12×18)	49,152 (4,096×12)	1
AGL600	A3P600/L	24	256	432 (24×18)	98,304 (4,096×24)	1
AGL1000	A3P1000/L	32	256	576 (32×18)	131,072 (4,096×32)	1
AGLE600	A3PE600	24	256	432 (24×18)	98,304 (4,096×24)	1
	A3PE1500	60	256	1,080 (60×18)	245,760 (4,096×60)	1
AGLE3000	A3PE3000/L	112	256	2,016 (112×18)	458,752 (4,096×112)	1

Notes:

1. *Maximum potential width uses the two-port configuration.*
2. *Maximum potential depth uses the dual-port configuration.*

***Table 6-9* • Memory Availability per Fusion Device**

Device	RAM Blocks	Maximum Potential Width[1]		Maximum Potential Depth[2]	
		Depth	Width	Depth	Width
AFS090	6	256	108 (6×18)	24,576 (4,094×6)	1
AFS250	8	256	144 (8×18)	32,768 (4,094×8)	1
AFS600	24	256	432 (24×18)	98,304 (4,096×24)	1
AFS1500	60	256	1,080 (60×18)	245,760 (4,096×60)	1

Notes:

1. *Maximum potential width uses the two-port configuration.*
2. *Maximum potential depth uses the dual-port configuration.*

Date	Changes	Page
v1.1 (continued)	Table 6-1 • Flash-Based FPGAs and associated text were updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	134
	The text introducing Table 6-8 • Memory Availability per IGLOO and ProASIC3 Device was updated to replace "A3P030 and AGL030" with "15 k and 30 k gate devices." Table 6-8 • Memory Availability per IGLOO and ProASIC3 Device was updated to remove AGL400 and AGLE1500 and include IGLOO PLUS and ProASIC3L devices.	146

***Figure 11-10* • All Silicon Features Selected for IGLOO and ProASIC3 Devices**

***Figure 11-11* • All Silicon Features Selected for Fusion**

Figure 12-2 shows different applications for ISP programming.

1. In a trusted programming environment, you can program the device using the unencrypted (plaintext) programming file.
2. You can program the AES Key in a trusted programming environment and finish the final programming in an untrusted environment using the AES-encrypted (cipher text) programming file.
3. For the remote ISP updating/reprogramming, the AES Key stored in the device enables the encrypted programming bitstream to be transmitted through the untrusted network connection.

Microsemi low power flash devices also provide the unique Microsemi FlashLock feature, which protects the Pass Key and AES Key. Unless the original FlashLock Pass Key is used to unlock the device, security settings cannot be modified. Microsemi does not support read-back of FPGA core-programmed data; however, the FlashROM contents can selectively be read back (or disabled) via the JTAG port based on the security settings established by the Microsemi Designer software. Refer to the "Security in Low Power Flash Devices" section on page 235 for more information.

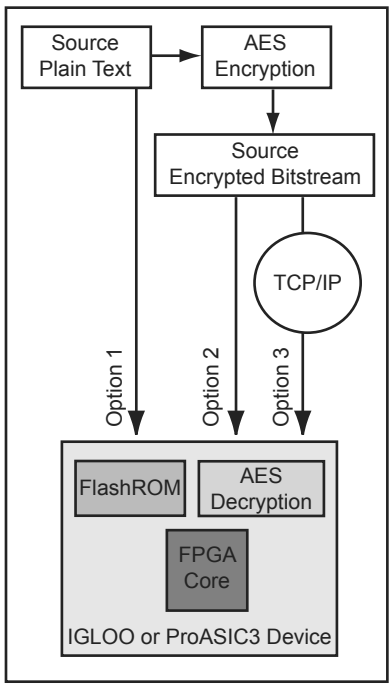


***Figure 12-2* • Different ISP Use Models**

13 – Core Voltage Switching Circuit for IGLOO and ProASIC3L In-System Programming

Introduction

The IGLOO® and ProASIC®3L families offer devices that can be powered by either 1.5 V or, in the case of V2 devices, a core supply voltage anywhere in the range of 1.2 V to 1.5 V, in 50 mV increments.

Since IGLOO and ProASIC3L devices are flash-based, they can be programmed and reprogrammed multiple times in-system using Microsemi FlashPro3. FlashPro3 uses the JTAG standard interface (IEEE 1149.1) and STAPL file (defined in JESD 71 to support programming of programmable devices using IEEE 1149.1) for in-system configuration/programming (IEEE 1532) of a device. Programming can also be executed by other methods, such as an embedded microcontroller that follows the same standards above.

All IGLOO and ProASIC3L devices must be programmed with the VCC core voltage at 1.5 V. Therefore, applications using IGLOO or ProASIC3L devices powered by a 1.2 V supply must switch the core supply to 1.5 V for in-system programming.

The purpose of this document is to describe an easy-to-use and cost-effective solution for switching the core supply voltage from 1.2 V to 1.5 V during in-system programming for IGLOO and ProASIC3L devices.

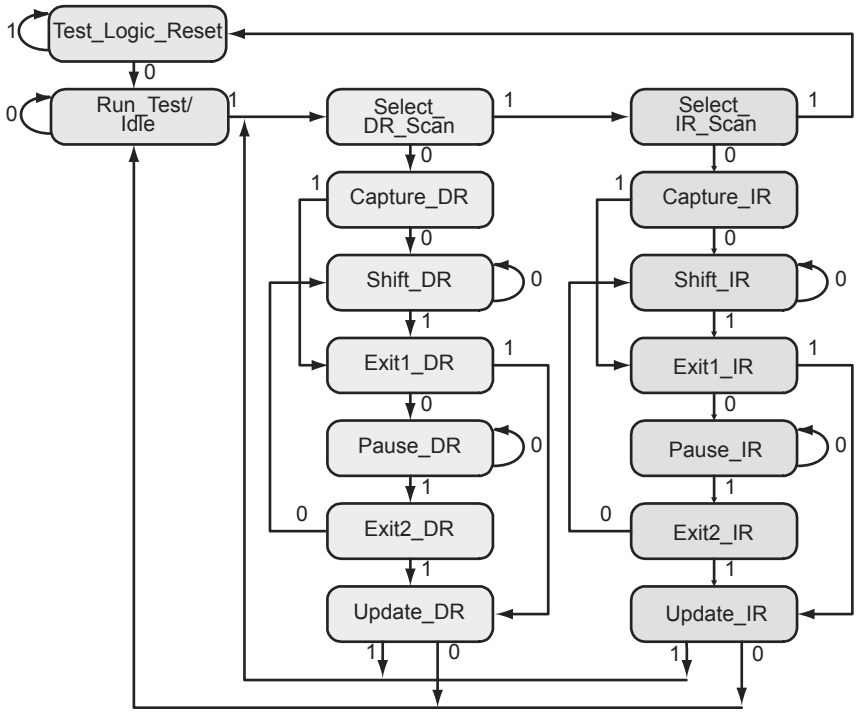


Figure 16-4 • **TAP Controller State Diagram**

UJTAG Port Usage

UIREG[7:0] hold the contents of the JTAG instruction register. The UIREG vector value is updated when the TAP Controller state machine enters the Update_IR state. Instructions 16 to 127 are user-defined and can be employed to encode multiple applications and commands within an application. Loading new instructions into the UIREG vector requires users to send appropriate logic to TMS to put the TAP Controller in a full IR cycle starting from the Select IR_Scan state and ending with the Update_IR state.

UTDI, UTDO, and UDRCK are directly connected to the JTAG TDI, TDO, and TCK ports, respectively. The TDI input can be used to provide either data (TAP Controller in the Shift_DR state) or the new contents of the instruction register (TAP Controller in the Shift_IR state).

UDRSH, UDRUPD, and UDRCAP are HIGH when the TAP Controller state machine is in the Shift_DR, Update_DR, and Capture_DR states, respectively. Therefore, they act as flags to indicate the stages of the data shift process. These flags are useful for applications in which blocks of data are shifted into the design from JTAG pins. For example, an active UDRSH can indicate that UTDI contains the data bitstream, and UDRUPD is a candidate for the end-of-data-stream flag.

As mentioned earlier, users should not connect the TDI, TDO, TCK, TMS, and TRST ports of the UJTAG macro to any port or net of the design netlist. The Designer software will automatically handle the port connection.