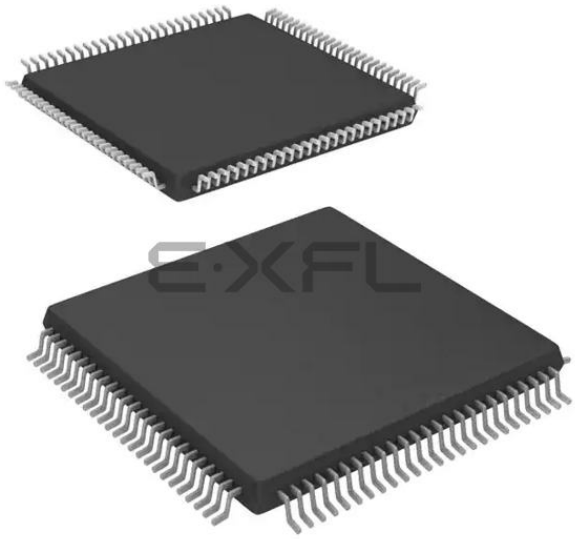


Welcome to E-XFL.COM

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	36864
Number of I/O	68
Number of Gates	250000
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/a3pn250-vq100i

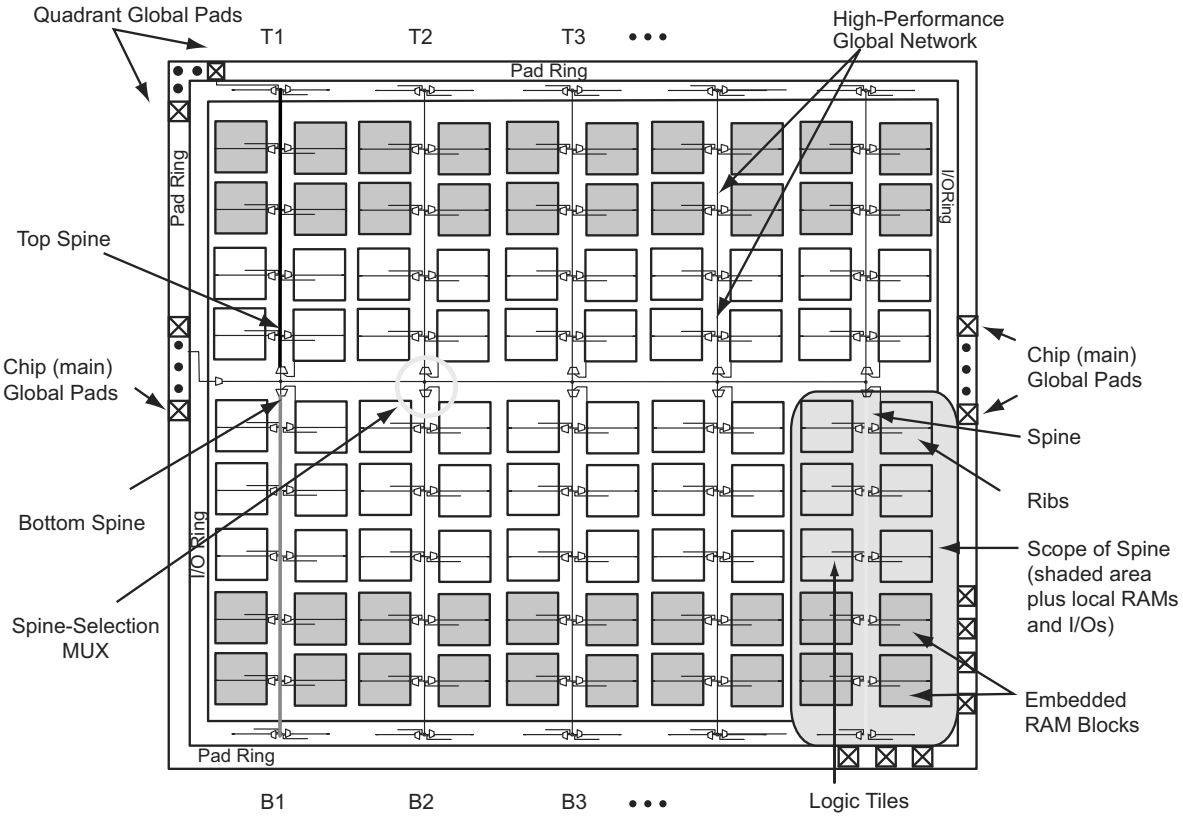
VersaNet Global Network Distribution

One of the architectural benefits of low power flash architecture is the set of powerful, low-delay VersaNet global networks that can access the VersaTiles, SRAM, and I/O tiles of the device. Each device offers a chip global network with six global lines (except for nano 10 k, 15 k, and 20 k gate devices) that are distributed from the center of the FPGA array. In addition, each device (except the 10 k through 30 k gate device) has four quadrant global networks, each consisting of three quadrant global net resources. These quadrant global networks can only drive a signal inside their own quadrant. Each VersaTile has access to nine global line resources—three quadrant and six chip-wide (main) global networks—and a total of 18 globals are available on the device (3 × 4 regional from each quadrant and 6 global).

Figure 3-1 shows an overview of the VersaNet global network and device architecture for devices 60 k and above. Figure 3-2 and Figure 3-3 on page 34 show simplified VersaNet global networks.

The VersaNet global networks are segmented and consist of spines, global ribs, and global multiplexers (MUXes), as shown in Figure 3-1. The global networks are driven from the global rib at the center of the die or quadrant global networks at the north or south side of the die. The global network uses the MUX trees to access the spine, and the spine uses the clock ribs to access the VersaTile. Access is available to the chip or quadrant global networks and the spines through the global MUXes. Access to the spine using the global MUXes is explained in the "Spine Architecture" section on page 41.

These VersaNet global networks offer fast, low-skew routing resources for high-fanout nets, including clock signals. In addition, these highly segmented global networks offer users the flexibility to create low-skew local clock networks using spines for up to 252 internal/external clocks or other high-fanout nets in low power flash devices. Optimal usage of these low-skew networks can result in significant improvement in design performance.



Note: Not applicable to 10 k through 30 k gate devices

***Figure 3-1* • Overview of VersaNet Global Network and Device Architecture**

Date	Changes	Page
v1.1 (March 2008)	The "Global Architecture" section was updated to include the IGLOO PLUS family. The bullet was revised to include that the west CCC does not contain a PLL core in 15 k and 30 k devices. Instances of "A3P030 and AGL030 devices" were replaced with "15 k and 30 k gate devices."	31
v1.1 (continued)	Table 3-1 • Flash-Based FPGAs and the accompanying text was updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	32
	The "VersaNet Global Network Distribution" section, "Spine Architecture" section, the note in Figure 3-1 • Overview of VersaNet Global Network and Device Architecture, and the note in Figure 3-3 • Simplified VersaNet Global Network (60 k gates and above) were updated to include mention of 15 k gate devices.	33, 34
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to add the A3P015 device, and to revise the values for clock trees, globals/spines per tree, and globals/spines per device for the A3P030 and AGL030 devices.	41
	Table 3-5 • Globals/Spines/Rows for IGLOO PLUS Devices is new.	42
	CLKBUF_LVCMOS12 was added to Table 3-9 • I/O Standards within CLKBUF.	47
	The "User's Guides" section was updated to include the three different I/O Structures chapters for ProASIC3 and IGLOO device families.	58
v1.0 (January 2008)	Figure 3-3 • Simplified VersaNet Global Network (60 k gates and above) was updated.	34
	The "Naming of Global I/Os" section was updated.	35
	The "Using Global Macros in Synplicity" section was updated.	50
	The "Global Promotion and Demotion Using PDC" section was updated.	51
	The "Designer Flow for Global Assignment" section was updated.	53
	The "Simple Design Example" section was updated.	55
51900087-0/1.05 (January 2005)	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated.	41

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.

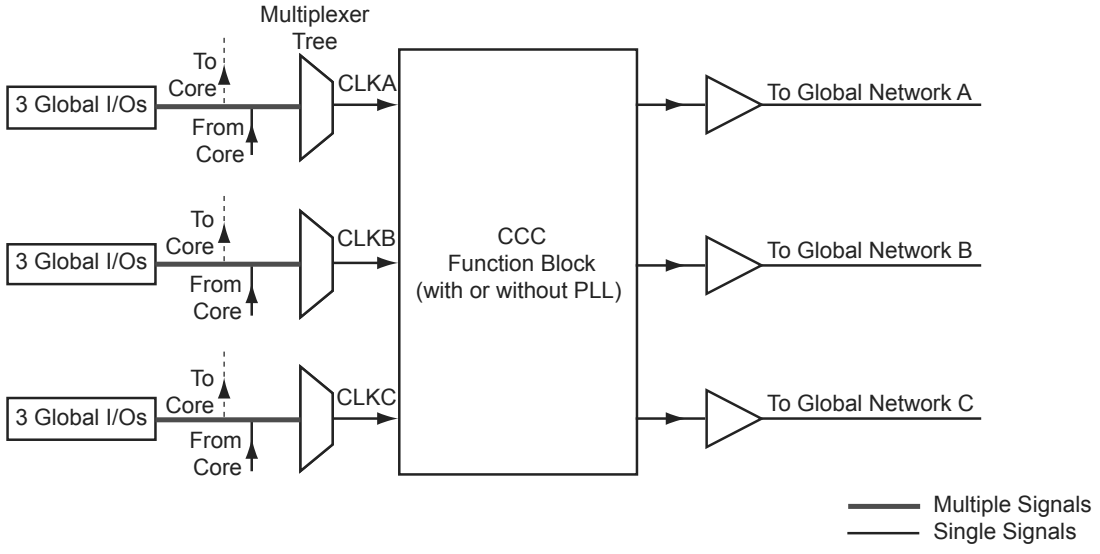
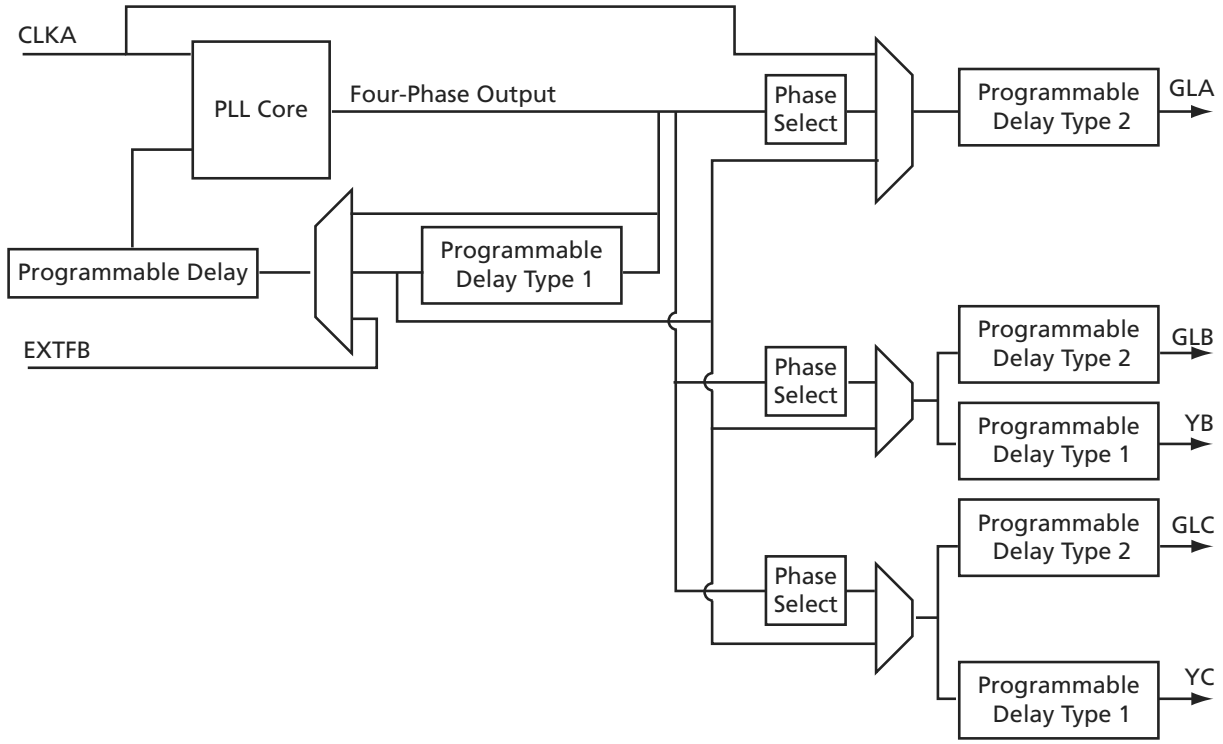


***Figure 4-1* • Overview of the CCCs Offered in Fusion, IGLOO, and ProASIC3**

SmartGen also allows the user to select the various delays and phase shift values necessary to adjust the phases between the reference clock (CLKA) and the derived clocks (GLA, GLB, GLC, YB, and YC). SmartGen allows the user to select the input clock source. SmartGen automatically instantiates the special macro, PLLINT, when needed.



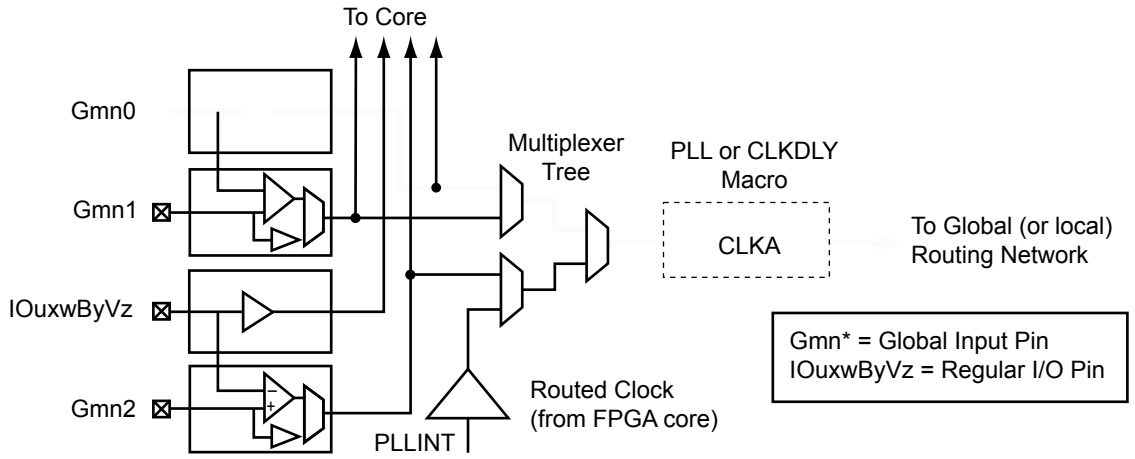
Note: Clock divider and clock multiplier blocks are not shown in this figure or in SmartGen. They are automatically configured based on the user's required frequencies.

***Figure 4-6* • CCC with PLL Block**

Global Input Selections

Low power flash devices provide the flexibility of choosing one of the three global input pad locations available to connect to a CCC functional block or to a global / quadrant global network. Figure 4-7 on page 72 and Figure 4-8 on page 72 show the detailed architecture of each global input structure for 30 k gate devices and below, as well as 60 k gate devices and above, respectively. For 60 k gate devices and above (Figure 4-7 on page 72), if the single-ended I/O standard is chosen, there is flexibility to choose one of the global input pads (the first, second, and fourth input). Once chosen, the other I/O locations are used as regular I/Os. If the differential I/O standard is chosen (not applicable for IGLOO nano and ProASIC3 nano devices), the first and second inputs are considered as paired, and the third input is paired with a regular I/O.

The user then has the choice of selecting one of the two sets to be used as the clock input source to the CCC functional block. There is also the option to allow an internal clock signal to feed the global network or the CCC functional block. A multiplexer tree selects the appropriate global input for routing to the desired location. Note that the global I/O pads do not need to feed the global network; they can also be used as regular I/O pads.



Note: Fusion CCCs have additional source selections (RCOSC, XTAL).

***Figure 4-9* • Illustration of Hardwired I/O (global input pins) Usage for IGLOO and ProASIC3 devices 60 k Gates and Larger**

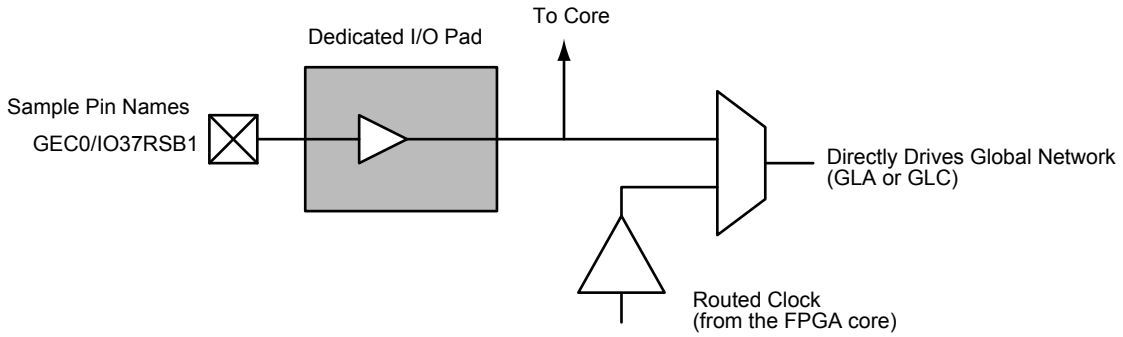


***Figure 4-10* • Illustration of Hardwired I/O (global input pins) Usage for IGLOO and ProASIC3 devices 30 k Gates and Smaller**

CCC Locations

CCCs located in the middle of the east and west sides of the device access the three VersaNet global networks on each side (six total networks), while the four CCCs located in the four corners access three quadrant global networks (twelve total networks). See Figure 4-13.

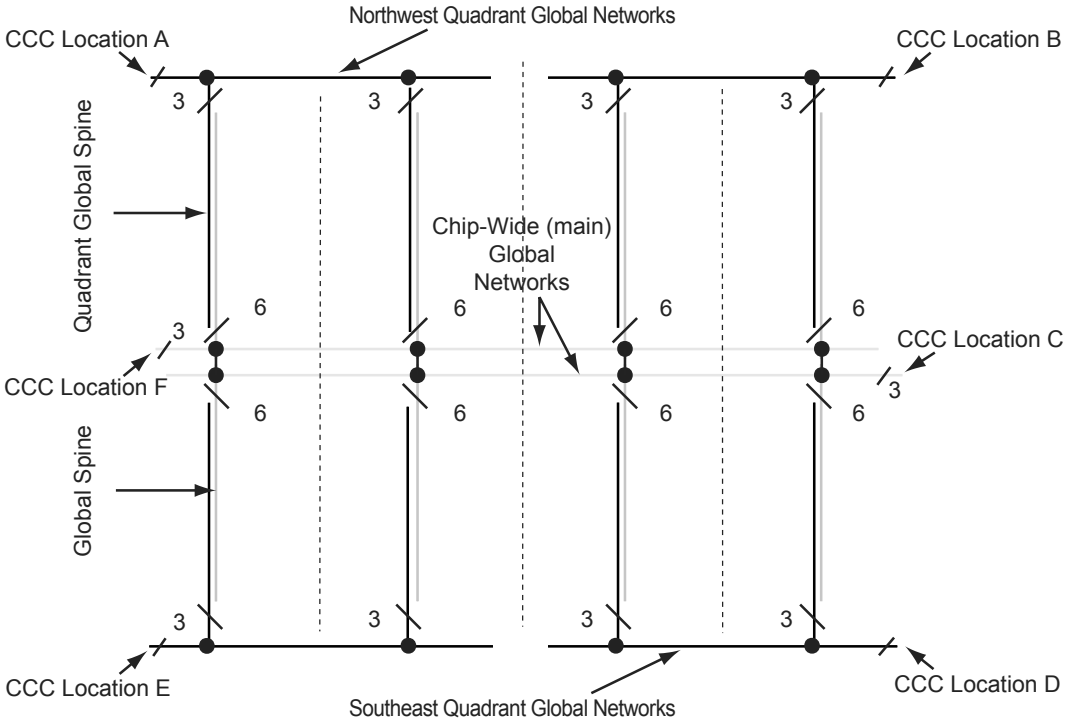


***Figure 4-13* • Global Network Architecture for 60 k Gate Devices and Above**

The following explains the locations of the CCCs in IGLOO and ProASIC3 devices:

In Figure 4-15 on page 82 through Figure 4-16 on page 82, CCCs with integrated PLLs are indicated in red, and simplified CCCs are indicated in yellow. There is a letter associated with each location of the CCC, in clockwise order. The upper left corner CCC is named "A," the upper right is named "B," and so on. These names finish up at the middle left with letter "F."

Loading the Configuration Register

The most important part of CCC dynamic configuration is to load the shift register properly with the configuration bits. There are different ways to access and load the configuration shift register:

- JTAG interface
- Logic core
- Specific I/O tiles

JTAG Interface

The JTAG interface requires no additional I/O pins. The JTAG TAP controller is used to control the loading of the CCC configuration shift register.

Low power flash devices provide a user interface macro between the JTAG pins and the device core logic. This macro is called UJTAG. A user should instantiate the UJTAG macro in his design to access the configuration register ports via the JTAG pins.

For more information on CCC dynamic reconfiguration using UJTAG, refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 297.

Logic Core

If the logic core is employed, the user must design a module to provide the configuration data and control the shifting and updating of the CCC configuration shift register. In effect, this is a user-designed TAP controller, which requires additional chip resources.

Specific I/O Tiles

If specific I/O tiles are used for configuration, the user must provide the external equivalent of a TAP controller. This does not require additional core resources but does use pins.

Shifting the Configuration Data

To enter a new configuration, all 81 bits must shift in via SDIN. After all bits are shifted, SSHIFT must go LOW and SUPDATE HIGH to enable the new configuration. For simulation purposes, bits <71:73> and <77:80> are "don't care."

The SUPDATE signal must be LOW during any clock cycle where SSHIFT is active. After SUPDATE is asserted, it must go back to the LOW state until a new update is required.

PLL Configuration Bits Description

***Table 4-8* • Configuration Bit Descriptions for the CCC Blocks**

Config. Bits	Signal	Name	Description
<88:87>	GLMUXCFG [1:0][1]	NGMUX configuration	The configuration bits specify the input clocks to the NGMUX (refer to Table 4-17 on page 94).[2]
86	OCDIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by the divider factor in Table 4-18 on page 95.
85	OBDIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by a 0.5 factor (refer to Table 4-18 on page 95).
84	OADIVHALF[1]	Division by half	When the PLL is bypassed, the 100 MHz RC oscillator can be divided by certain 0.5 factor (refer to Table 4-16 on page 94).

Notes:

1. *The <88:81> configuration bits are only for the Fusion dynamic CCC.*
2. *This value depends on the input clock source, so Layout must complete before these bits can be set. After completing Layout in Designer, generate the "CCC_Configuration" report by choosing **Tools** > **Report** > **CCC_Configuration**. The report contains the appropriate settings for these bits.*

Figure 5-12 shows the programming file generator, which enables different STAPL file generation methods. When you select **Program FlashROM** and choose the UFC file, the FlashROM Settings window appears, as shown in Figure 5-13. In this window, you can select the FlashROM page you want to program and the data value for the configured regions. This enables you to use a different page for different programming files.

***Figure 5-12* • Programming File Generator**

***Figure 5-13* • Setting FlashROM during Programming File Generation**

The programming hardware and software can load the FlashROM with the appropriate STAPL file. Programming software handles the single STAPL file that contains multiple FlashROM contents for multiple devices, and programs the FlashROM in sequential order (e.g., for device serialization). This feature is supported in the programming software. After programming with the STAPL file, you can run DEVICE_INFO to check the FlashROM content.

Initializing the RAM/FIFO

The SRAM blocks can be initialized with data to use as a lookup table (LUT). Data initialization can be accomplished either by loading the data through the design logic or through the UJTAG interface. The UJTAG macro is used to allow access from the JTAG port to the internal logic in the device. By sending the appropriate initialization string to the JTAG Test Access Port (TAP) Controller, the designer can put the JTAG circuitry into a mode that allows the user to shift data into the array logic through the JTAG port using the UJTAG macro. For a more detailed explanation of the UJTAG macro, refer to the "FlashROM in Microsemi's Low Power Flash Devices" section on page 117.

A user interface is required to receive the user command, initialization data, and clock from the UJTAG macro. The interface must synchronize and load the data into the correct RAM block of the design. The main outputs of the user interface block are the following:

- Memory block chip select: Selects a memory block for initialization. The chip selects signals for each memory block that can be generated from different user-defined pockets or simple logic, such as a ring counter (see below).
- Memory block write address: Identifies the address of the memory cell that needs to be initialized.
- Memory block write data: The interface block receives the data serially from the UTDI port of the UJTAG macro and loads it in parallel into the write data ports of the memory blocks.
- Memory block write clock: Drives the WCLK of the memory block and synchronizes the write data, write address, and chip select signals.

Figure 6-8 shows the user interface between UJTAG and the memory blocks.

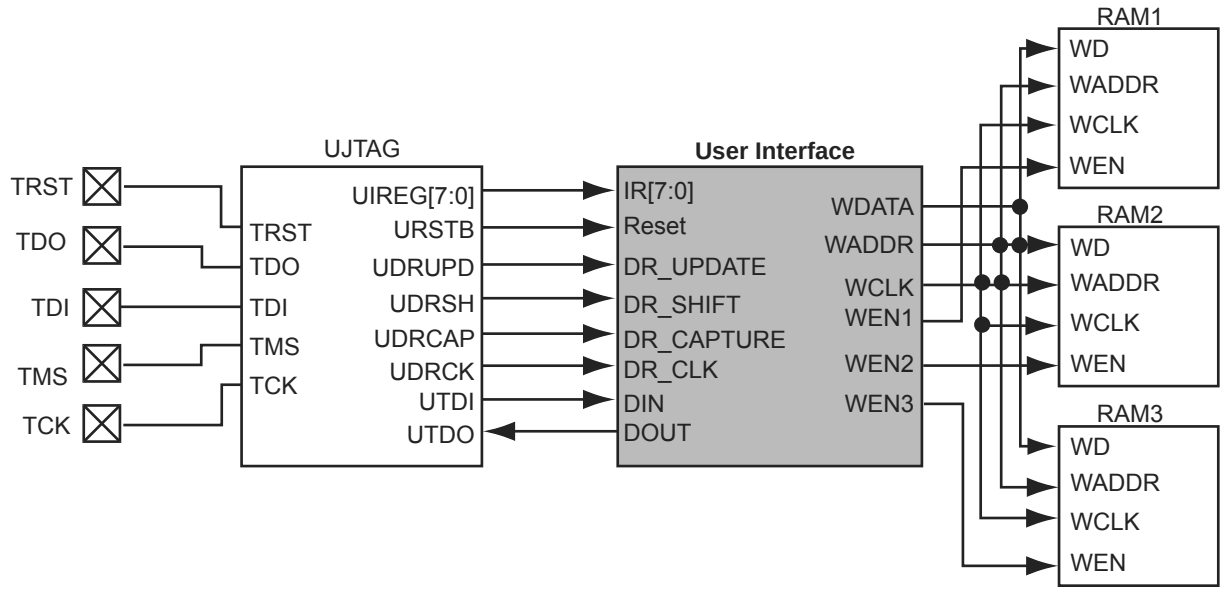


Figure 6-8 • **Interfacing TAP Ports and SRAM Blocks**

An important component of the interface between the UJTAG macro and the RAM blocks is a serial-in/parallel-out shift register. The width of the shift register should equal the data width of the RAM blocks. The RAM data arrives serially from the UTDI output of the UJTAG macro. The data must be shifted into a shift register clocked by the JTAG clock (provided at the UDRCK output of the UJTAG macro).

Then, after the shift register is fully loaded, the data must be transferred to the write data port of the RAM block. To synchronize the loading of the write data with the write address and write clock, the output of the shift register can be pipelined before driving the RAM block.

The write address can be generated in different ways. It can be imported through the TAP using a different instruction opcode and another shift register, or generated internally using a simple counter. Using a counter to generate the address bits and sweep through the address range of the RAM blocks is

***Table 7-8* • Hot-Swap Level 1**

Description	Cold-swap
Power Applied to Device	No
Bus State	–
Card Ground Connection	–
Device Circuitry Connected to Bus Pins	–
Example Application	System and card with Microsemi FPGA chip are powered down, and the card is plugged into the system. Then the power supplies are turned on for the system but not for the FPGA on the card.
Compliance of nano Devices	Compliant

***Table 7-9* • Hot-Swap Level 2**

Description	Hot-swap while reset
Power Applied to Device	Yes
Bus State	Held in reset state
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	–
Example Application	In the PCI hot-plug specification, reset control circuitry isolates the card busses until the card supplies are at their nominal operating levels and stable.
Compliance of nano Devices	Compliant

***Table 7-10* • Hot-Swap Level 3**

Description	Hot-swap while bus idle
Power Applied to Device	Yes
Bus State	Held idle (no ongoing I/O processes during insertion/removal)
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	Must remain glitch-free during power-up or power-down
Example Application	Board bus shared with card bus is "frozen," and there is no toggling activity on the bus. It is critical that the logic states set on the bus signal not be disturbed during card insertion/removal.
Compliance of nano Devices	Compliant

***Table 7-11* • Hot-Swap Level 4**

Description	Hot-swap on an active bus
Power Applied to Device	Yes
Bus State	Bus may have active I/O processes ongoing, but device being inserted or removed must be idle.
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	Must remain glitch-free during power-up or power-down
Example Application	There is activity on the system bus, and it is critical that the logic states set on the bus signal not be disturbed during card insertion/removal.
Compliance of nano Devices	Compliant

For Level 3 and Level 4 compliance with the nano devices, cards with two levels of staging should have the following sequence:

- Grounds
- Powers, I/Os, and other pins

- If one of the registers has a PRE pin, all the other registers that are candidates for combining in the I/O must have a PRE pin.
- If one of the registers has neither a CLR nor a PRE pin, all the other registers that are candidates for combining must have neither a CLR nor a PRE pin.
- If the clear or preset pins are present, they must have the same polarity.
- If the clear or preset pins are present, they must be driven by the same signal (net).

3. For single-tile devices (10 k, 15 k, and 20 k): Registers connected to an I/O on the Output and Output Enable pins must have the same clock function (both CLR and CLK are shared among all registers):
   - Both the Output and Output Enable registers must not have an E pin (clock enable).
4. For dual-tile devices (60 k, 125 k, and 250 k): Registers connected to an I/O on the Output and Output Enable pins must have the same clock and enable function:
   - Both the Output and Output Enable registers must have an E pin (clock enable), or none at all.
   - If the E pins are present, they must have the same polarity. The CLK pins must also have the same polarity.

   In some cases, the user may want registers to be combined with the input of a bibuf while maintaining the output as-is. This can be achieved by using PDC commands as follows:

```
set_io <signal name> -REGISTER yes ------register will combine
set_preserve <signal name> ----register will not combine
```

Weak Pull-Up and Weak Pull-Down Resistors

nano devices support optional weak pull-up and pull-down resistors on each I/O pin. When the I/O is pulled up, it is connected to the V_{CCI} of its corresponding I/O bank. When it is pulled down, it is connected to GND. Refer to the datasheet for more information.

For low power applications and when using IGLOO nano devices, configuration of the pull-up or pull-down of the I/O can be used to set the I/O to a known state while the device is in Flash*Freeze mode. Refer to "Flash*Freeze Technology and Low Power Modes" in an applicable FPGA fabric user's guide for more information.

The Flash*Freeze (FF) pin cannot be configured with a weak pull-down or pull-up I/O attribute, as the signal needs to be driven at all times.

Output Slew Rate Control

The slew rate is the amount of time an input signal takes to get from logic LOW to logic HIGH or vice versa.

It is commonly defined as the propagation delay between 10% and 90% of the signal's voltage swing. Slew rate control is available for the output buffers of low power flash devices. The output buffer has a programmable slew rate for both HIGH-to-LOW and LOW-to-HIGH transitions.

The slew rate can be implemented by using a PDC command (Table 7-5 on page 163), setting it "High" or "Low" in the I/O Attribute Editor in Designer, or instantiating a special I/O macro. The default slew rate value is "High."

Microsemi recommends the high slew rate option to minimize the propagation delay. This high-speed option may introduce noise into the system if appropriate signal integrity measures are not adopted. Selecting a low slew rate reduces this kind of noise but adds some delays in the system. Low slew rate is recommended when bus transients are expected.

Output Drive

The output buffers of nano devices can provide multiple drive strengths to meet signal integrity requirements. The LVTTL and LVCMOS (except 1.2 V LVCMOS) standards have selectable drive strengths.

Drive strength should also be selected according to the design requirements and noise immunity of the system.

I/O Bank Resource Usage

This is an important portion of the report. The user must meet the requirements stated in this table. Figure 8-10 shows the I/O Bank Resource Usage table included in the I/O bank report:

***Figure 8-10* • I/O Bank Resource Usage Table**

The example in Figure 8-10 shows that none of the I/O macros is assigned to the bank because more than one VCCI is detected.

I/O Voltage Usage

The I/O Voltage Usage table provides the number of VREF (E devices only) and V_{CCI} assignments required in the design. If the user decides to make I/O assignments manually (PDC or MVN), the issues listed in this table must be resolved before proceeding to Layout. As stated earlier, VREF assignments must be made if there are any voltage-referenced I/Os.

Figure 8-11 shows the I/O Voltage Usage table included in the I/O bank report.

***Figure 8-11* • I/O Voltage Usage Table**

The table in Figure 8-11 indicates that there are two voltage-referenced I/Os used in the design. Even though both of the voltage-referenced I/O technologies have the same VCCI voltage, their VREF voltages are different. As a result, two I/O banks are needed to assign the VCCI and VREF voltages.

In addition, there are six single-ended I/Os used that have the **same VCCI voltage. Since two banks are already assigned with the same VCCI voltage and there are enough u**nused bonded I/Os in

Application 3: Nontrusted Environment—Field Updates/Upgrades

Programming or reprogramming of devices may occur at remote locations. Reconfiguration of devices in consumer products/equipment through public networks is one example. Typically, the remote system is already programmed with particular design contents. When design update (FPGA array contents update) and/or data upgrade (FlashROM and/or FB contents upgrade) is necessary, an updated programming file with AES encryption can be generated, sent across public networks, and transmitted to the remote system. Reprogramming can then be done using this AES-encrypted programming file, providing easy and secure field upgrades. Low power flash devices support this secure ISP using AES. The detailed flow for this application is shown in Figure 11-8. Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

To prepare devices for this scenario, the user can initially generate a programming file with the available security setting options. This programming file is programmed into the devices before shipment. During the programming file generation step, the user has the option of making the security settings permanent or not. In situations where no changes to the security settings are necessary, the user can select this feature in the software to generate the programming file with permanent security settings. Microsemi recommends that the programming file use encryption with an AES key, especially when ISP is done via public domain.

For example, if the designer wants to use an AES key for the FPGA array and the FlashROM, **Permanent** needs to be chosen for this setting. At first, the user chooses the options to use an AES key for the FPGA array and the FlashROM, and then chooses **Permanently lock the security settings**. A unique AES key is chosen. Once this programming file is generated and programmed to the devices, the AES key is permanently stored in the on-chip memory, where it is secured safely. The devices are sent to distant locations for the intended application. When an update is needed, a new programming file must be generated. The programming file must use the same AES key for encryption; otherwise, the authentication will fail and the file will not be programmed in the device.

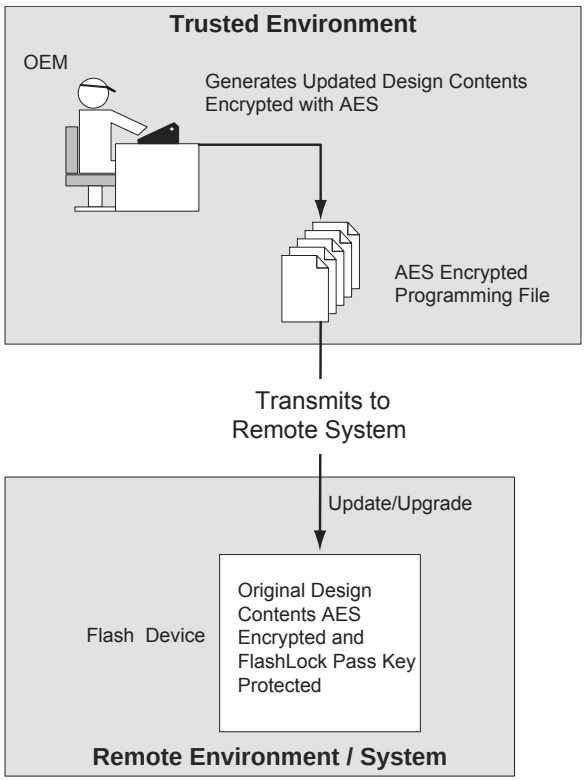


***Figure 11-8* • Application 3: Nontrusted Environment—Field Updates/Upgrades**

Generating Programming Files

Generation of the Programming File in a Trusted Environment—Application 1

As discussed in the "Application 1: Trusted Environment" section on page 243, in a trusted environment, the user can choose to program the device with plaintext bitstream content. It is possible to use plaintext for programming even when the FlashLock Pass Key option has been selected. In this application, it is not necessary to employ AES encryption protection. For AES encryption settings, refer to the next sections.

The generated programming file will include the security setting (if selected) and the plaintext programming file content for the FPGA array, FlashROM, and/or FBs. These options are indicated in Table 11-2 and Table 11-3.

***Table 11-2* • IGLOO and ProASIC3 Plaintext Security Options, No AES**

Security Protection	FlashROM Only	FPGA Core Only	Both FlashROM and FPGA
No AES / no FlashLock	✓	✓	✓
FlashLock only	✓	✓	✓
AES and FlashLock	–	–	–

***Table 11-3* • Fusion Plaintext Security Options**

Security Protection	FlashROM Only	FPGA Core Only	FB Core Only	All
No AES / no FlashLock	✓	✓	✓	✓
FlashLock	✓	✓	✓	✓
AES and FlashLock	–	–	–	–

Note: For all instructions, the programming of Flash Blocks refers to Fusion only.

For this scenario, generate the programming file as follows:

1. Select the **Silicon features to be programmed** (Security Settings, FPGA Array, FlashROM, Flash Memory Blocks), as shown in Figure 11-10 on page 248 and Figure 11-11 on page 248. Click **Next**.

   If **Security Settings** is selected (i.e., the FlashLock security Pass Key feature), an additional dialog will be displayed to prompt you to select the security level setting. If no security setting is selected, you will be directed to Step 3.

Figure 11-18 • **Security Level Set High to Reprogram Device with AES Key**

Programming with this file is intended for an unsecured environment. The AES key encrypts the programming file with the same AES key already used in the device and utilizes it to program the device.

Reprogramming Devices

Previously programmed devices can be reprogrammed using the steps in the "Generation of the Programming File in a Trusted Environment—Application 1" section on page 247 and "Generation of Security Header Programming File Only—Application 2" section on page 250. In the case where a FlashLock Pass Key has been programmed previously, the user must generate the new programming file with a FlashLock Pass Key that matches the one previously programmed into the device. The software will check the FlashLock Pass Key in the programming file against the FlashLock Pass Key in the device. The keys must match before the device can be unlocked to perform further programming with the new programming file.

Figure 11-10 on page 248 and Figure 11-11 on page 248 show the option **Programming previously secured device(s)**, which the user should select before proceeding. Upon going to the next step, the user will be notified that the same FlashLock Pass Key needs to be entered, as shown in Figure 11-19 on page 256.

List of Changes

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

Date	Changes	Page
September 2012	The "Security" section was modified to clarify that Microsemi does not support read-back of FPGA core-programmed data (SAR 41235).	288
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 14-1 • Flash-Based FPGAs.	284
v1.3 (October 2008)	The "Microprocessor Programming Support in Flash Devices" section was revised to include new families and make the information more concise.	284
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 14-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	284
v1.1 (March 2008)	The "Microprocessor Programming Support in Flash Devices" section was updated to include information on the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	284

UJTAG Macro

The UJTAG tiles can be instantiated in a design using the UJTAG macro from the Fusion, IGLOO, or ProASIC3 macro library. Note that "UJTAG" is a reserved name and cannot be used for any other user-defined blocks. A block symbol of the UJTAG tile macro is presented in Figure 16-2. In this figure, the ports on the left side of the block are connected to the JTAG TAP Controller, and the right-side ports are accessible by the FPGA core VersaTiles. The TDI, TMS, TDO, TCK, and TRST ports of UJTAG are only provided for design simulation purposes and should be treated as external signals in the design netlist. However, these ports must NOT be connected to any I/O buffer in the netlist. Figure 16-3 on page 300 illustrates the correct connection of the UJTAG macro to the user design netlist. Microsemi Designer software will automatically connect these ports to the TAP during place-and-route. Table 16-2 gives the port descriptions for the rest of the UJTAG ports:

Table 16-2 • **UJTAG Port Descriptions**

Port	Description
UIREG [7:0]	This 8-bit bus carries the contents of the JTAG Instruction Register of each device. Instruction Register values 16 to 127 are not reserved and can be employed as user-defined instructions.
URSTB	URSTB is an active-low signal and will be asserted when the TAP Controller is in Test-Logic-Reset mode. URSTB is asserted at power-up, and a power-on reset signal resets the TAP Controller. URSTB will stay asserted until an external TAP access changes the TAP Controller state.
UTDI	This port is directly connected to the TAP's TDI signal.
UTDO	This port is the user TDO output. Inputs to the UTDO port are sent to the TAP TDO output MUX when the IR address is in user range.
UDRSH	Active-high signal enabled in the ShiftDR TAP state
UDRCAP	Active-high signal enabled in the CaptureDR TAP state
UDRCK	This port is directly connected to the TAP's TCK signal.
UDRUPD	Active-high signal enabled in the UpdateDR TAP state

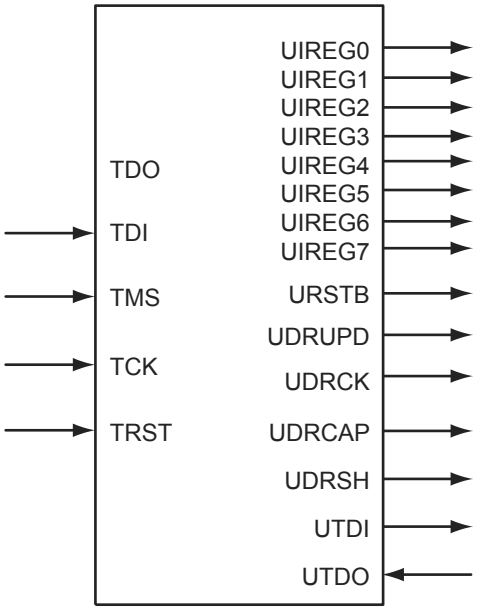


Figure 16-2 • **UJTAG Tile Block Symbol**

Brownout Voltage

Brownout is a condition in which the voltage supplies are lower than normal, causing the device to malfunction as a result of insufficient power. In general, Microsemi does not guarantee the functionality of the design inside the flash FPGA if voltage supplies are below their minimum recommended operating condition. Microsemi has performed measurements to characterize the brownout levels of FPGA power supplies. Refer to Table 17-3 for device-specific brownout deactivation levels. For the purpose of characterization, a direct path from the device input to output is monitored while voltage supplies are lowered gradually. The brownout point is defined as the voltage level at which the output stops following the input. Characterization tests performed on several IGLOO, ProASIC3L, and ProASIC3 devices in typical operating conditions showed the brownout voltage levels to be within the specification.

During device power-down, the device I/Os become tristated once the first supply in the power-down sequence drops below its brownout deactivation voltage.

***Table 17-3* • Brownout Deactivation Levels for VCC and VCCI**

Devices	VCC B**rownout Deactivation Level (V)**	VCCI Br**ownout Deactivation Level (V)**
ProASIC3, ProASIC3 nano, IGLOO, IGLOO nano, IGLOO PLUS and ProASIC3L devices running at VCC = 1.5 V	0.75 V ± 0.25 V	0.8 V ± 0.3 V
IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices running at VCC = 1.2 V	0.75 V ± 0.2 V	0.8 V ± 0.15 V

PLL Behavior at Brownout Condition

When PLL power supply voltage and/or V_{CC} levels drop below the V_{CC} brownout levels mentioned above for 1.5 V and 1.2 V devices, the PLL output lock signal goes LOW and/or the output clock is lost. The following sections explain PLL behavior during and after the brownout condition.

VCCPLL and VCC **Tied Together**

In this condition, both VCC and VCCPLL drop below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level. During the brownout recovery, once VCCPLL and VCC reach the activation point (0.85 ± 0.25 V or ± 0.2 V) again, the PLL output lock signal may still remain LOW with the PLL output clock signal toggling. If this condition occurs, there are two ways to recover the PLL output lock signal:

1. Cycle the power supplies of the PLL (power off and on) by using the PLL POWERDOWN signal.
2. Turn off the input reference clock to the PLL and then turn it back on.

Only VCCPLL **Is at Brownout**

In this case, only VCCPLL drops below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level and the VCC supply remains at nominal recommended operating voltage (1.5 V ± 0.075 V for 1.5 V devices and 1.2 V ± 0.06 V for 1.2 V devices). In this condition, the PLL behavior after brownout recovery is similar to initial power-up condition, and the PLL will regain lock automatically after VCCPLL is ramped up above the activation level (0.85 ± 0.25 V or ± 0.2 V). No intervention is necessary in this case.

Only VCC **Is at Brownout**

In this condition, VCC drops below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level and VCCPLL remains at nominal recommended operating voltage (1.5 V ± 0.075 V for 1.5 V devices and 1.2 V ± 0.06 V for 1.2 V devices). During the brownout recovery, once VCC reaches the activation point again (0.85 ± 0.25 V or ± 0.2 V), the PLL output lock signal may still remain LOW with the PLL output clock signal toggling. If this condition occurs, there are two ways to recover the PLL output lock signal:

1. Cycle the power supplies of the PLL (power off and on) by using the PLL POWERDOWN signal.
2. Turn off the input reference clock to the PLL and then turn it back on.

It is important to note that Microsemi recommends using a monotonic power supply or voltage regulator to ensure proper power-up behavior.