



Welcome to [E-XFL.COM](https://www.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**

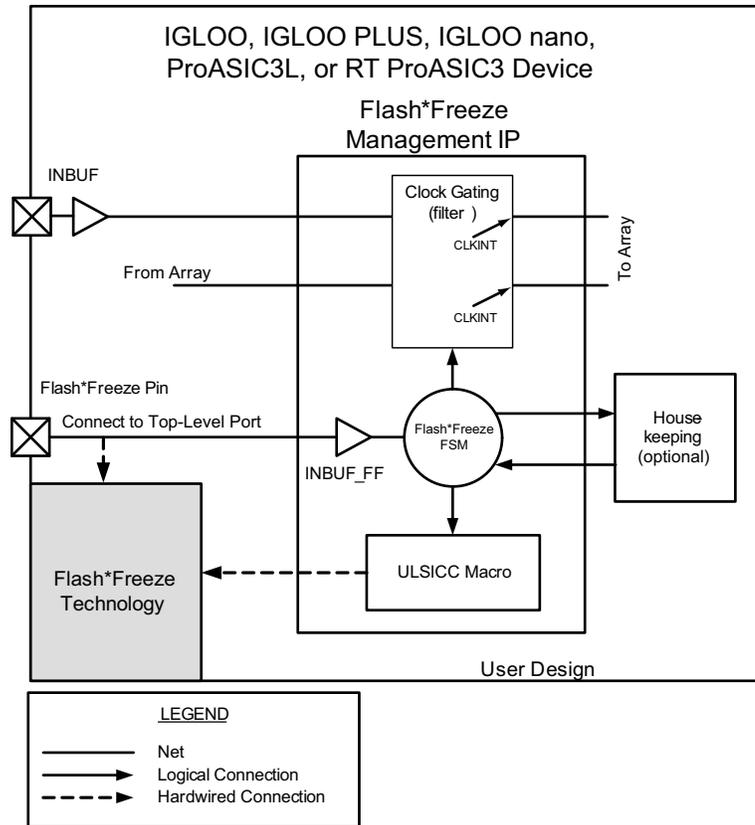
The versatility of Embedded - FPGAs makes them indispensable in numerous fields. In telecommunications.

#### **Details**

Product Status	Active
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	516096
Number of I/O	221
Number of Gates	3000000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	324-BGA
Supplier Device Package	324-FBGA (19x19)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/microchip-technology/a3pe3000l-1fg324">https://www.e-xfl.com/product-detail/microchip-technology/a3pe3000l-1fg324</a>

Flash\*Freeze management IP. Additional information on this IP core can be found in the Libero online help.

The Flash\*Freeze management IP is comprised of three blocks: the Flash\*Freeze finite state machine (FSM), the clock gating (filter) block, and the ULSICC macro, as shown in Figure 2-10.



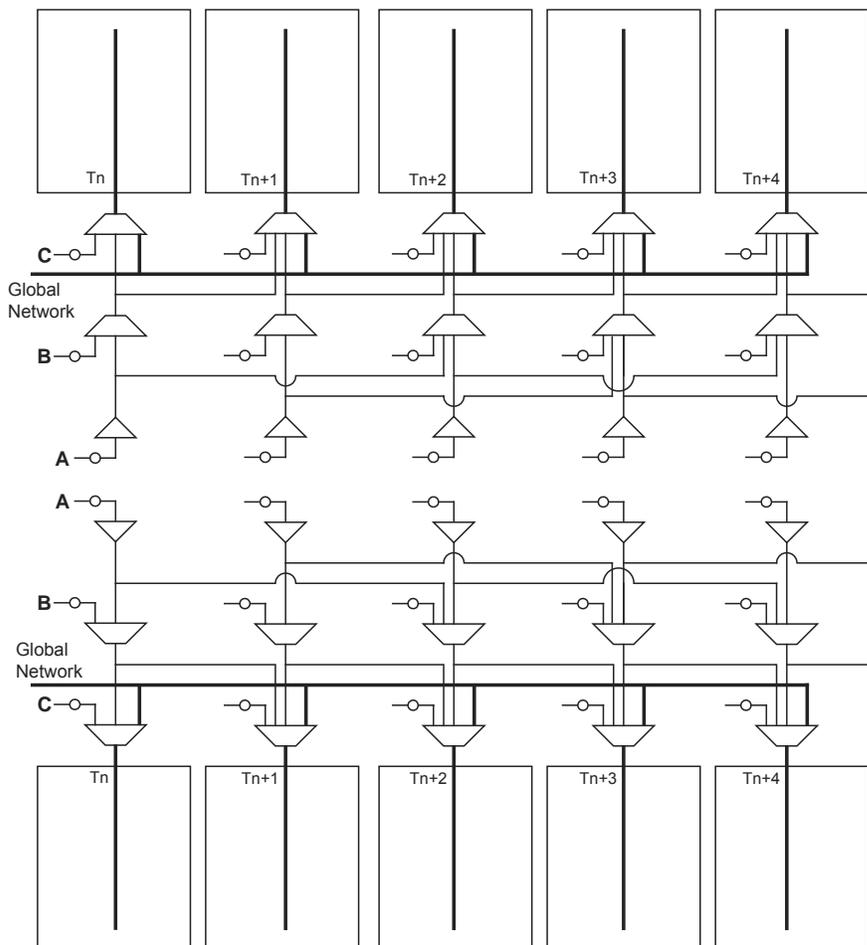
**Figure 2-10 • Flash\*Freeze Management IP Block Diagram**

### **Flash\*Freeze Management FSM**

The Flash\*Freeze FSM block is a simple, robust, fully encoded 3-bit state machine that ensures clean entrance to and exit from Flash\*Freeze mode by controlling activities of the clock gating, ULSICC, and optional housekeeping blocks. The state diagram for the FSM is shown in Figure 2-11 on page 38. In normal operation, the state machine waits for Flash\*Freeze pin assertion, and upon detection of a request, it waits for a short period of time to ensure the assertion persists; then it asserts WAIT\_HOUSEKEEPING (active High) synchronous to the user's designated system clock. This flag can be used by user logic to perform any needed shutdown processes prior to entering Flash\*Freeze mode, such as storing data into SRAM, notifying other system components of the request, or timing/validating the Flash\*Freeze request. The FSM also asserts Flash\_Freeze\_Enabled whenever the device enters Flash\*Freeze mode. This occurs after all housekeeping and clock gating functions have completed. The Flash\_Freeze\_Enabled signal remains asserted, even during Flash\*Freeze mode, until the Flash\*Freeze pin is deasserted. Use the Flash\_Freeze\_Enabled signal to drive any logic in the design that needs to be in a particular state during Flash\*Freeze mode. The DONE\_HOUSEKEEPING (active High) signal should be asserted to notify the FSM when all the housekeeping tasks are completed. If the user chooses not to use housekeeping, the Flash\*Freeze management IP core generator in Libero SoC will connect WAIT\_HOUSEKEEPING to DONE\_HOUSEKEEPING.

## Spine Access

The physical location of each spine is identified by the letter T (top) or B (bottom) and an accompanying number ( $Tn$  or  $Bn$ ). 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 60 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.

During Layout, Designer will assign two of the signals to quadrant global locations.

### Step 3 (optional)

You can also assign the QCLK1\_c and QCLK2\_c nets to quadrant regions using the following PDC commands:

```
assign_local_clock -net QCLK1_c -type quadrant UL
assign_local_clock -net QCLK2_c -type quadrant LL
```

### Step 4

Import this PDC with the netlist and run Compile again. You will see the following in the Compile report:

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 : 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

The following nets have been assigned to a quadrant clock resource using PDC:

Fanout	Type	Name
256	CLK_NET	Net : QCLK1_c Driver: QCLK1_pad_CLKINT Region: quadrant_UL
256	CLK_NET	Net : QCLK2_c Driver: QCLK2_pad_CLKINT Region: quadrant_LL

### Step 5

Run Layout.

## Global Management in PLL Design

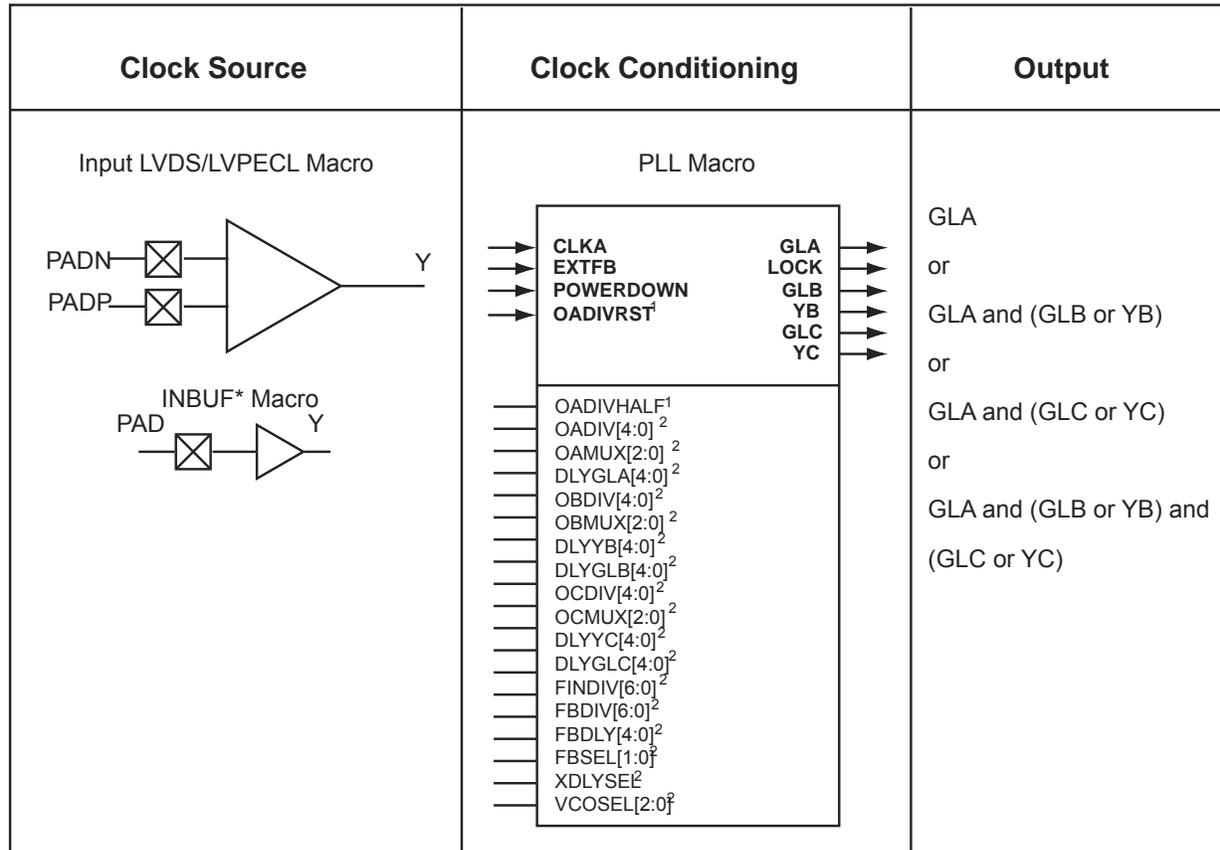
This section describes the legal global network connections to PLLs in the low power flash devices. For detailed information on using PLLs, refer to "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 77. Microsemi recommends that you use the dedicated global pins to directly drive the reference clock input of the associated PLL for reduced propagation delays and clock distortion. However, low power flash devices offer the flexibility to connect other signals to reference clock inputs. Each PLL is associated with three global networks (Figure 3-5 on page 52). There are some limitations, such as when trying to use the global and PLL at the same time:

- If you use a PLL with only primary output, you can still use the remaining two free global networks.
- If you use three globals associated with a PLL location, you cannot use the PLL on that location.
- If the YB or YC output is used standalone, it will occupy one global, even though this signal does not go to the global network.

## Global Buffers with PLL Function

Clocks requiring frequency synthesis or clock adjustments can utilize the PLL core before connecting to the global / quadrant global networks. A maximum of 18 CCC global buffers can be instantiated in a device—three per CCC and up to six CCCs per device. Each PLL core can generate up to three global/quadrant clocks, while a clock delay element provides one.

The PLL functionality of the clock conditioning block is supported by the PLL macro.



**Notes:**

1. For Fusion only.
2. Refer to the IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide for more information.
3. For INBUF\* driving a PLL macro or CLKDLY macro, the I/O will be hard-routed to the CCC; i.e., will be placed by software to a dedicated Global I/O.
4. IGLOO nano and ProASIC3 nano devices do not support differential inputs.

**Figure 4-4 • CCC Options: Global Buffers with PLL**

The PLL macro provides five derived clocks (three independent) from a single reference clock. The PLL macro also provides power-down input and lock output signals. The additional inputs shown on the macro are configuration settings, which are configured through the use of SmartGen. For manual setting of these bits refer to the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide* for details.

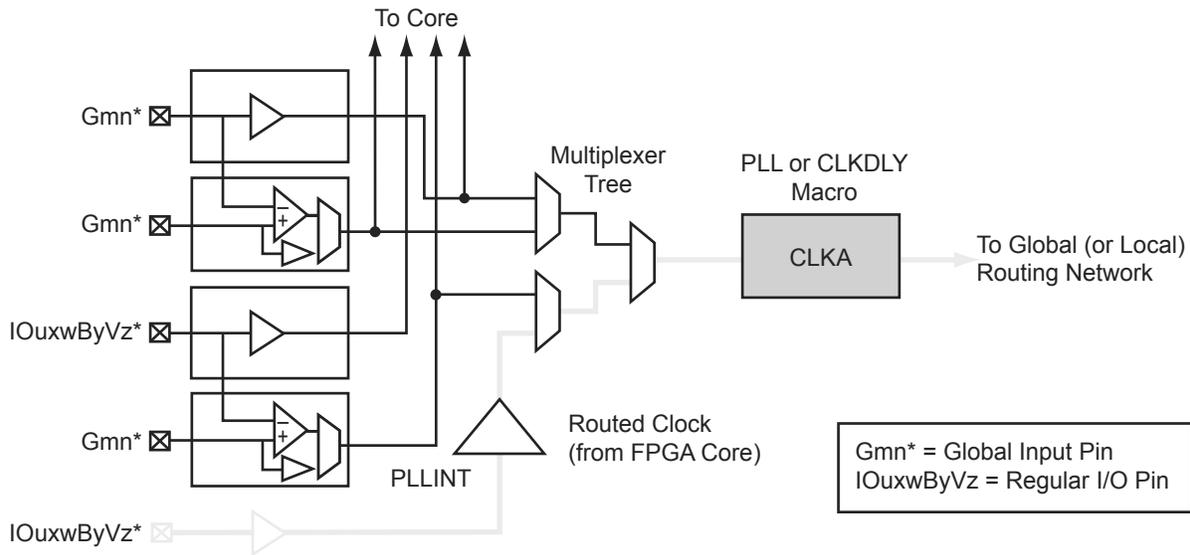
Figure 4-6 on page 87 illustrates the various clock output options and delay elements.

### External I/O Clock Source

External I/O refers to regular I/O pins. The clock source is instantiated with one of the various INBUF options and accesses the CCCs via internal routing. The user has the option of assigning this input to any of the I/Os labeled with the I/O convention *IOuxwByVz*. Refer to the "User I/O Naming Conventions in I/O Structures" chapter of the appropriate device user's guide, and for Fusion, refer to the *Fusion Family of Mixed Signal FPGAs* datasheet for more information. Figure 4-11 gives a brief explanation of external I/O usage. Choosing this option provides the freedom of selecting any user I/O location but introduces additional delay because the signal connects to the routed clock input through internal routing before connecting to the CCC reference clock input.

For the External I/O option, the routed signal would be instantiated with a PLLINT macro before connecting to the CCC reference clock input. This instantiation is conveniently done automatically by SmartGen when this option is selected. Microsemi recommends using the SmartGen tool to generate the CCC macro. The instantiation of the PLLINT macro results in the use of the routed clock input of the I/O to connect to the PLL clock input. If not using SmartGen, manually instantiate a PLLINT macro before the PLL reference clock to indicate that the regular I/O driving the PLL reference clock should be used (see Figure 4-11 for an example illustration of the connections, shown in red).

In the above two options, the clock source must be instantiated with one of the various INBUF macros. The reference clock pins of the CCC functional block core macros must be driven by regular input macros (INBUFs), not clock input macros.



**Figure 4-11 • Illustration of External I/O Usage**

For Fusion devices, the input reference clock can also be from the embedded RC oscillator and crystal oscillator. In this case, the CCC configuration is the same as the hardwired I/O clock source, and users are required to instantiate the RC oscillator or crystal oscillator macro and connect its output to the input reference clock of the CCC block.

## Available I/O Standards

**Table 4-4 • Available I/O Standards within CLKBUF and CLKBUF\_LVDS/LVPECL Macros**

CLKBUF_LVCMOS5
CLKBUF_LVCMOS33 <sup>1</sup>
CLKBUF_LVCMOS25 <sup>2</sup>
CLKBUF_LVCMOS18
CLKBUF_LVCMOS15
CLKBUF_PCI
CLKBUF_PCIX <sup>3</sup>
CLKBUF_GTL25 <sup>2,3</sup>
CLKBUF_GTL33 <sup>2,3</sup>
CLKBUF_GTLP25 <sup>2,3</sup>
CLKBUF_GTLP33 <sup>2,3</sup>
CLKBUF_HSTL_I <sup>2,3</sup>
CLKBUF_HSTL_II <sup>2,3</sup>
CLKBUF_SSTL3_I <sup>2,3</sup>
CLKBUF_SSTL3_II <sup>2,3</sup>
CLKBUF_SSTL2_I <sup>2,3</sup>
CLKBUF_SSTL2_II <sup>2,3</sup>
CLKBUF_LVDS <sup>4,5</sup>
CLKBUF_LVPECL <sup>5</sup>

*Notes:*

1. By default, the CLKBUF macro uses 3.3 V LVTTTL I/O technology. For more details, refer to the IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide.
2. I/O standards only supported in ProASIC3E and IGLOOe families.
3. I/O standards only supported in the following Fusion devices: AFS600 and AFS1500.
4. B-LVDS and M-LVDS standards are supported by CLKBUF\_LVDS.
5. Not supported for IGLOO nano and ProASIC3 nano devices.

## Global Synthesis Constraints

The Synplify® synthesis tool, by default, allows six clocks in a design for Fusion, IGLOO, and ProASIC3. When more than six clocks are needed in the design, a user synthesis constraint attribute, `syn_global_buffers`, can be used to control the maximum number of clocks (up to 18) that can be inferred by the synthesis engine.

High-fanout nets will be inferred with clock buffers and/or internal clock buffers. If the design consists of CCC global buffers, they are included in the count of clocks in the design.

The subsections below discuss the clock input source (global buffers with no programmable delays) and the clock conditioning functional block (global buffers with programmable delays and/or PLL function) in detail.

## Phase Adjustment

The four phases available (0, 90, 180, 270) are phases with respect to VCO (PLL output). The VCO is divided to achieve the user's CCC required output frequency (GLA, YB/GLB, YC/GLC). The division happens after the selection of the VCO phase. The effective phase shift is actually the VCO phase shift divided by the output divider. This is why the visual CCC shows both the actual achievable phase and more importantly the actual delay that is equivalent to the phase shift that can be achieved.

## Dynamic PLL Configuration

The CCCs can be configured both statically and dynamically.

In addition to the ports available in the Static CCC, the Dynamic CCC has the dynamic shift register signals that enable dynamic reconfiguration of the CCC. With the Dynamic CCC, the ports CLKB and CLKC are also exposed. All three clocks (CLKA, CLKB, and CLKC) can be configured independently. The CCC block is fully configurable. The following two sources can act as the CCC configuration bits.

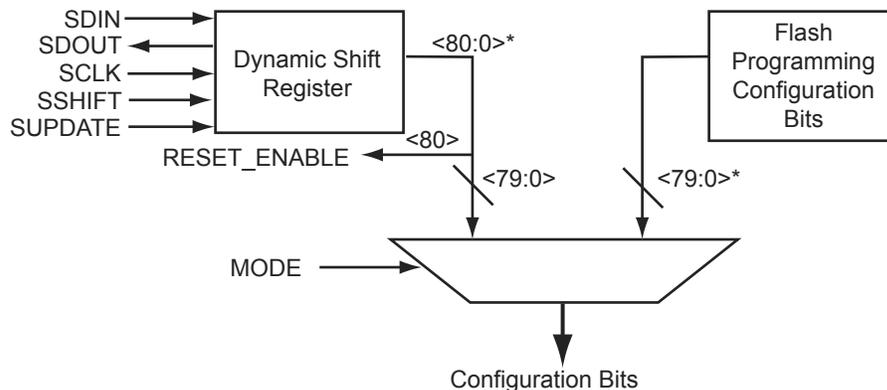
### Flash Configuration Bits

The flash configuration bits are the configuration bits associated with programmed flash switches. These bits are used when the CCC is in static configuration mode. Once the device is programmed, these bits cannot be modified. They provide the default operating state of the CCC.

### Dynamic Shift Register Outputs

This source does not require core reprogramming and allows core-driven dynamic CCC reconfiguration. When the dynamic register drives the configuration bits, the user-defined core circuit takes full control over SDIN, SDOUT, SCLK, SSHIFT, and SUPDATE. The configuration bits can consequently be dynamically changed through shift and update operations in the serial register interface. Access to the logic core is accomplished via the dynamic bits in the specific tiles assigned to the PLLs.

Figure 4-21 illustrates a simplified block diagram of the MUX architecture in the CCCs.



Note: \*For Fusion, bit <88:81> is also needed.

**Figure 4-21 • The CCC Configuration MUX Architecture**

The selection between the flash configuration bits and the bits from the configuration register is made using the MODE signal shown in Figure 4-21. If the MODE signal is logic HIGH, the dynamic shift register configuration bits are selected. There are 81 control bits to configure the different functions of the CCC.

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

Config. Bits	Signal	Name	Description
<31:29>	OAMUX[2:0]	GLA Output Select	Selects from the VCO's four phase outputs for GLA.
<28:24>	OCDIV[4:0]	Secondary 2 Output Divider	Sets the divider value for the GLC/YC outputs. Also known as divider <i>w</i> in Figure 4-20 on page 101. The divider value will be $OCDIV[4:0] + 1$ .
<23:19>	OBDIV[4:0]	Secondary 1 Output Divider	Sets the divider value for the GLB/YB outputs. Also known as divider <i>v</i> in Figure 4-20 on page 101. The divider value will be $OBDIV[4:0] + 1$ .
<18:14>	OADIV[4:0]	Primary Output Divider	Sets the divider value for the GLA output. Also known as divider <i>u</i> in Figure 4-20 on page 101. The divider value will be $OADIV[4:0] + 1$ .
<13:7>	FBDIV[6:0]	Feedback Divider	Sets the divider value for the PLL core feedback. Also known as divider <i>m</i> in Figure 4-20 on page 101. The divider value will be $FBDIV[6:0] + 1$ .
<6:0>	FINDIV[6:0]	Input Divider	Input Clock Divider ( <i>n</i> ). Sets the divider value for the input delay on CLK_A. The divider value will be $FINDIV[6:0] + 1$ .

*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 4-31 • Static Timing Analysis Using SmartTime**

**Place-and-Route Stage Considerations**

Several considerations must be noted to properly place the CCC macros for layout.

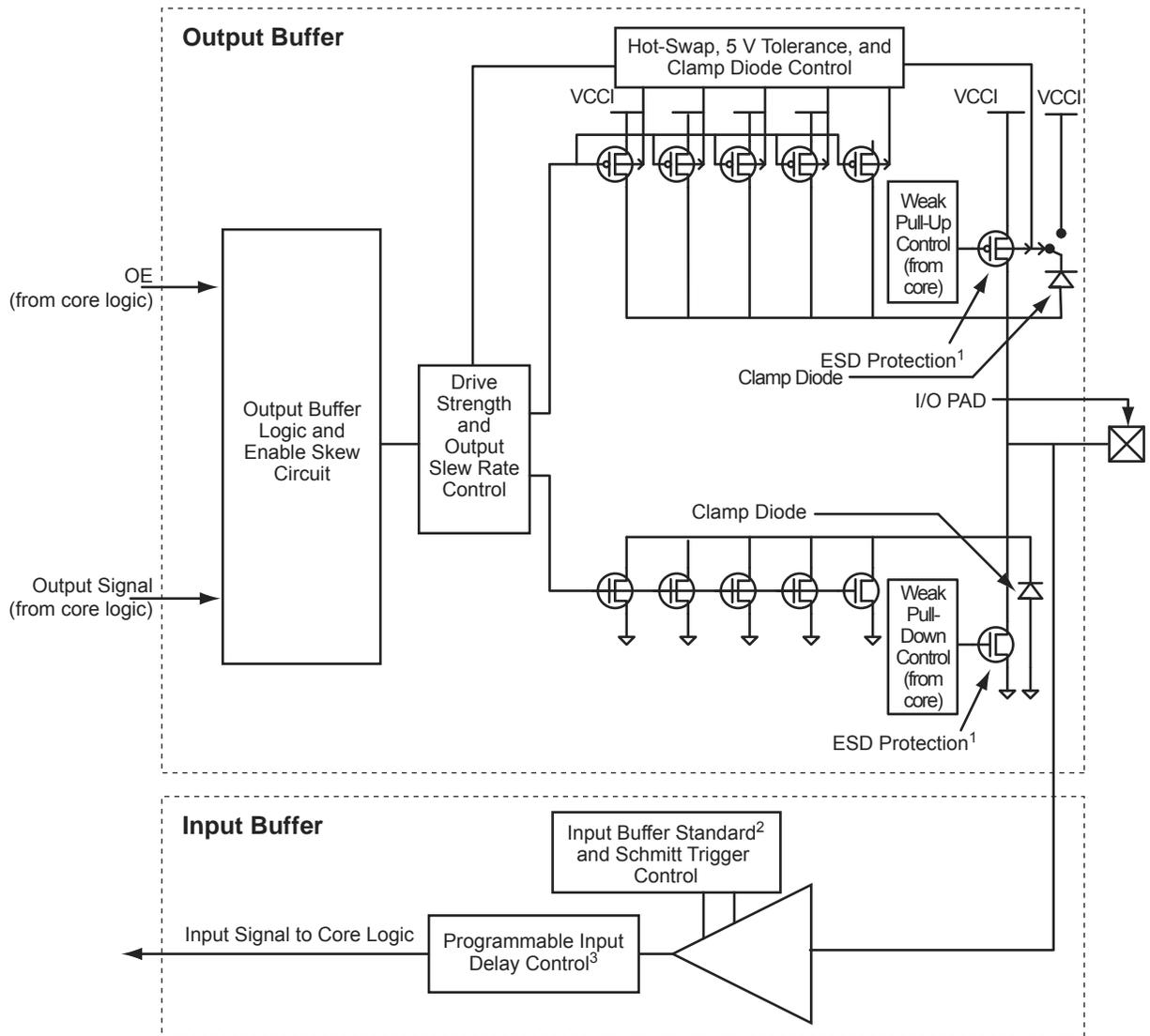
For CCCs with clock inputs configured with the Hardwired I/O–Driven option:

- PLL macros must have the clock input pad coming from one of the GmA\* locations.
- CLKDLY macros must have the clock input pad coming from one of the Global I/Os.

If a PLL with a Hardwired I/O input is used at a CCC location and a Hardwired I/O–Driven CLKDLY macro is used at the same CCC location, the clock input of the CLKDLY macro must be chosen from one of the GmB\* or GmC\* pin locations. If the PLL is not used or is an External I/O–Driven or Core Logic–Driven PLL, the clock input of the CLKDLY macro can be sourced from the GmA\*, GmB\*, or GmC\* pin locations.

For CCCs with clock inputs configured with the External I/O–Driven option, the clock input pad can be assigned to any regular I/O location (IO\*\*\*\*\* pins). Note that since global I/O pins can also be used as regular I/Os, regardless of CCC function (CLKDLY or PLL), clock inputs can also be placed in any of these I/O locations.

By default, the Designer layout engine will place global nets in the design at one of the six chip globals. When the number of globals in the design is greater than six, the Designer layout engine will automatically assign additional globals to the quadrant global networks of the low power flash devices. If the user wishes to decide which global signals should be assigned to chip globals (six available) and which to the quadrant globals (three per quadrant for a total of 12 available), the assignment can be achieved with PinEditor, ChipPlanner, or by importing a placement constraint file. Layout will fail if the



**Notes:**

1. All NMOS transistors connected to the I/O pad serve as ESD protection.
2. See Table 8-2 on page 215 for available I/O standards.
3. Programmable input delay is applicable only to ProASIC3E, IGLOOe, ProASIC3EL, and RT ProASIC3 devices.

**Figure 8-5 • Simplified I/O Buffer Circuitry**

**I/O Registers**

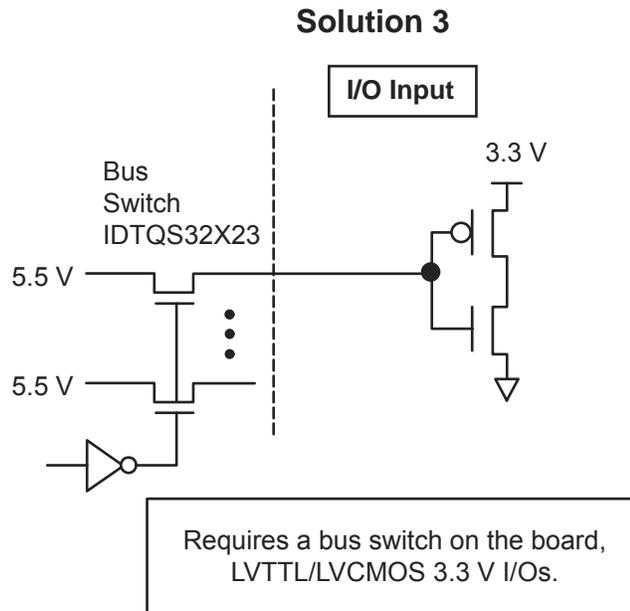
Each I/O module contains several input, output, and enable registers. Refer to Figure 8-5 for a simplified representation of the I/O block. The number of input registers is selected by a set of switches (not shown in Figure 8-3 on page 220) between registers to implement single-ended or differential data transmission to and from the FPGA core. The Designer software sets these switches for the user. A common CLR/PRE signal is employed by all I/O registers when I/O register combining is used. Input Register 2 does not have a CLR/PRE pin, as this register is used for DDR implementation. The I/O register combining must satisfy certain rules.



### Solution 3

The board-level design must ensure that the reflected waveform at the pad does not exceed the voltage overshoot/undershoot limits provided in the datasheet. This is a requirement to ensure long-term reliability.

This scheme will also work for a 3.3 V PCI/PCI-X configuration, but the internal diode should not be used for clamping, and the voltage must be limited by the bus switch, as shown in Figure 8-12. Relying on the diode clamping would create an excessive pad DC voltage of  $3.3\text{ V} + 0.7\text{ V} = 4\text{ V}$ .



**Figure 8-12 • Solution 3**

- The I/O standard of technology-specific I/O macros cannot be changed in the I/O Attribute Editor (see Figure 9-6).
- The user MUST instantiate differential I/O macros (LVDS/LVPECL) in the design. This is the only way to use these standards in the design (IGLOO nano and ProASIC3 nano devices do not support differential inputs).
- To implement the DDR I/O function, the user must instantiate a DDR\_REG or DDR\_OUT macro. This is the only way to use a DDR macro in the design.

**Figure 9-6 • Assigning a Different I/O Standard to the Generic I/O Macro**

## Performing Place-and-Route on the Design

The netlist created by the synthesis tool should now be imported into Designer and compiled. During Compile, the user can specify the I/O placement and attributes by importing the PDC file. The user can also specify the I/O placement and attributes using ChipPlanner and the I/O Attribute Editor under MVN.

### Defining I/O Assignments in the PDC File

A PDC file is a Tcl script file specifying physical constraints. This file can be imported to and exported from Designer.

Table 9-3 shows I/O assignment constraints supported in the PDC file.

**Table 9-3 • PDC I/O Constraints**

Command	Action	Example	Comment
<b>I/O Banks Setting Constraints</b>			
set_iobank	Sets the I/O supply voltage, $V_{CCI}$ , and the input reference voltage, $V_{REF}$ , for the specified I/O bank.	<pre>set_iobank bankname [-vcci vcci_voltage] [-vref vref_voltage]  set_iobank Bank7 -vcci 1.50 -vref 0.75</pre>	Must use in case of mixed I/O voltage ( $V_{CCI}$ ) design
set_vref	Assigns a $V_{REF}$ pin to a bank.	<pre>set_vref -bank [bankname] [pinnum]  set_vref -bank Bank0 685 704 723 742 761</pre>	Must use if voltage-referenced I/Os are used
set_vref_defaults	Sets the default $V_{REF}$ pins for the specified bank. This command is ignored if the bank does not need a $V_{REF}$ pin.	<pre>set_vref_defaults bankname  set_vref_defaults bank2</pre>	

*Note:* Refer to the Libero SoC User's Guide for detailed rules on PDC naming and syntax conventions.

## Related Documents

### User's Guides

*FlashPro User's Guide*

[http://www.microsemi.com/soc/documents/flashpro\\_ug.pdf](http://www.microsemi.com/soc/documents/flashpro_ug.pdf)

## List of Changes

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

Date	Changes	Page
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
v1.5 (August 2009)	The "CoreMP7 Device Security" section was removed from "Security in ARM-Enabled Low Power Flash Devices", since M7-enabled devices are no longer supported.	304
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 12-1 • Flash-Based FPGAs.	302
v1.3 (October 2008)	The "Security Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	302
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 12-1 • Flash-Based FPGAs: <ul style="list-style-type: none"> <li>• ProASIC3L was updated to include 1.5 V.</li> <li>• The number of PLLs for ProASIC3E was changed from five to six.</li> </ul>	302
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices.	N/A
	The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	302

signal deactivated, which also has the effect of disabling the input buffers. The SAMPLE/PRELOAD instruction captures the status of pads in parallel and shifts them out as new data is shifted in for loading into the Boundary Scan Register (BSR). When the device is in an unprogrammed state, the OE and output BSR will be undefined; however, the input BSR will be defined as long as it is connected and being used. For JTAG timing information on setup, hold, and fall times, refer to the *FlashPro User's Guide*.

## ISP Support in Flash-Based Devices

The flash FPGAs listed in Table 13-1 support the ISP feature and the functions described in this document.

**Table 13-1 • Flash-Based FPGAs Supporting ISP**

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 nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
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
SmartFusion	SmartFusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable microcontroller subsystem (MSS) which includes programmable analog and an ARM® Cortex™-M3 hard processor and flash memory in a monolithic device
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
ProASIC	ProASIC	First generation ProASIC devices
	ProASIC <sup>PLUS</sup>	Second generation ProASIC devices

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 devices as listed in Table 13-1. 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 13-1. 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*.

Date	Changes	Page
July 2010 (continued)	The "Chain Integrity Test Error Analyze Chain Failure" section was renamed to the "Scan Chain Failure" section, and the Analyze Chain command was changed to Scan Chain. It was noted that occasionally a faulty programmer can cause scan chain failures.	338
v1.5 (August 2009)	The "CoreMP7 Device Security" section was removed from "Security in ARM-Enabled Low Power Flash Devices", since M7-enabled devices are no longer supported.	331
v1.4 (December 2008)	The "ISP Architecture" section was revised to include information about core voltage for IGLOO V2 and ProASIC3L devices, as well as 50 mV increments allowable in Designer software.	327
	IGLOO nano and ProASIC3 nano devices were added to Table 13-1 • Flash-Based FPGAs Supporting ISP.	328
	A second capacitor was added to Figure 13-6 • Board Layout and Programming Header Top View.	337
v1.3 (October 2008)	The "ISP Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	328
v1.2 (June 2008)	<p>The following changes were made to the family descriptions in Table 13-1 • Flash-Based FPGAs Supporting ISP:</p> <ul style="list-style-type: none"> <li>• ProASIC3L was updated to include 1.5 V.</li> <li>• The number of PLLs for ProASIC3E was changed from five to six.</li> </ul>	328
v1.1 (March 2008)	The "ISP Architecture" section was updated to included the IGLOO PLUS family in the discussion of family-specific support. The text, "When 1.2 V is used, the device can be reprogrammed in-system at 1.5 V only," was revised to state, "Although the device can operate at 1.2 V core voltage, the device can only be reprogrammed when all supplies (VCC, VCCI, and VJTAG) are at 1.5 V."	327
	The "ISP Support in Flash-Based Devices" section and Table 13-1 • Flash-Based FPGAs Supporting ISP were updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	328
	The "Security" section was updated to mention that 15 k gate devices do not have a built-in 128-bit decryption core.	330
	Table 13-2 • Power Supplies was revised to remove the Normal Operation column and add a table note stating, "All supply voltages should be at 1.5 V or higher, regardless of the setting during normal operation."	329
	The "ISP Programming Header Information" section was revised to change FP3-26PIN-ADAPTER to FP3-10PIN-ADAPTER-KIT. Table 13-3 • Programming Header Ordering Codes was updated with the same change, as well as adding the part number FFSD-05-D-06.00-01-N, a 10-pin cable with 50-mil-pitch sockets.	335
	The "Board-Level Considerations" section was updated to describe connecting two capacitors in parallel across VPUMP and GND for proper programming.	337
v1.0 (January 2008)	Information was added to the "Programming Voltage (VPUMP) and VJTAG" section about the JTAG interface pin.	329
51900055-2/7.06	ACTgen was changed to SmartGen.	N/A
	<p>In Figure 13-6 • Board Layout and Programming Header Top View, the order of the text was changed to:</p> <p>VJTAG from the target board VCCI from the target board VCC from the target board</p>	337

## **14 – Core Voltage Switching Circuit for IGLOO and ProASIC3L In-System Programming**

---

### **Introduction**

The IGLOO<sup>®</sup> and ProASIC<sup>®</sup>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.

## Circuit Verification

The power switching circuit recommended above is implemented on Microsemi's Icicle board (Figure 14-2). On the Icicle board, VJTAGENB is used to control the N-Channel Digital FET; however, this circuit was modified to use TRST instead of VJTAGENB in this application. There are three important aspects of this circuit that were verified:

1. The rise on VCC from 1.2 V to 1.5 V when TRST is HIGH
2. VCC rises to 1.5 V before programming begins.
3. VCC switches from 1.5 V to 1.2 V when TRST is LOW.

## Verification Steps

1. The rise on VCC from 1.2 V to 1.5 V when TRST is HIGH.
- 

---

### **Figure 14-2 • Core Voltage on the IGLOO AGL125-QNG132 Device**

In the oscilloscope plots (Figure 14-2), the TRST from FlashPro3 and the VCC core voltage of the IGLOO device are labeled. This plot shows the rise characteristic of the TRST signal from FlashPro3. Once the TRST signal is asserted HIGH, the LTC3025 shown in Figure 14-1 on page 343 senses the increase in voltage and changes the output from 1.2 V to 1.5 V. It takes the circuit approximately 100  $\mu$ s to respond to TRST and change the voltage to 1.5 V on the VCC core.

## SRAM Initialization

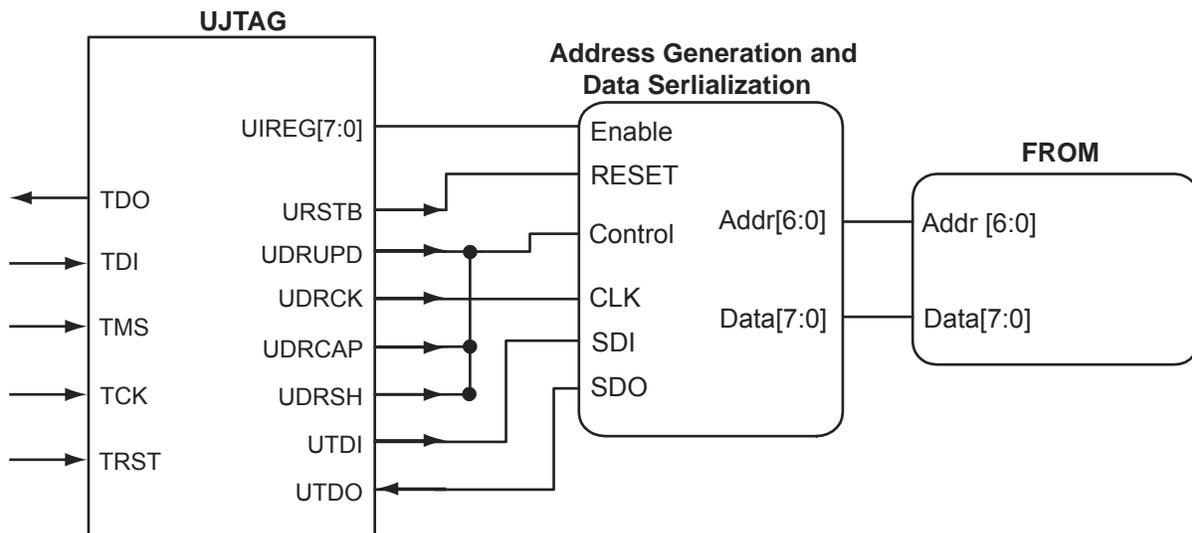
Users can also initialize embedded SRAMs of the low power flash devices. The initialization of the embedded SRAM blocks of the design can be done using UJTAG tiles, where the initialization data is imported using the TAP Controller. Similar functionality is available in ProASIC<sup>PLUS</sup> devices using JTAG. The guidelines for implementation and design examples are given in the *RAM Initialization and ROM Emulation in ProASIC<sup>PLUS</sup> Devices* application note.

SRAMs are volatile by nature; data is lost in the absence of power. Therefore, the initialization process should be done at each power-up if necessary.

## FlashROM Read-Back Using JTAG

The low power flash architecture contains a dedicated nonvolatile FlashROM block, which is formatted into eight 128-bit pages. For more information on FlashROM, refer to the "FlashROM in Microsemi's Low Power Flash Devices" section on page 133. The contents of FlashROM are available to the VersaTiles during normal operation through a read operation. As a result, the UJTAG macro can be used to provide the FlashROM contents to the JTAG port during normal operation. Figure 17-7 illustrates a simple block diagram of using UJTAG to read the contents of FlashROM during normal operation.

The FlashROM read address can be provided from outside the FPGA through the TDI input or can be generated internally using the core logic. In either case, data serialization logic is required (Figure 17-7) and should be designed using the VersaTile core logic. FlashROM contents are read asynchronously in parallel from the flash memory and shifted out in a synchronous serial format to TDO. Shifting the serial data out of the serialization block should be performed while the TAP is in UDRSH mode. The coordination between TCK and the data shift procedure can be done using the TAP state machine by monitoring UDRSH, UDRCAP, and UDRUPD.



**Figure 17-7 • Block Diagram of Using UJTAG to Read FlashROM Contents**

FlashLock  
  IGLOO and ProASIC devices 307  
  permanent 307

FlashROM  
  access using JTAG port 139  
  architecture 333  
  architecture of user nonvolatile 133  
  configuration 136  
  custom serialization 145  
  design flow 140  
  generation 141  
  programming and accessing 138  
  programming file 143  
  programming files 333  
  SmartGen 142  
FlashROM read-back 371

## G

global architecture 47  
global buffers  
  no programmable delays 80  
  with PLL function 83  
  with programmable delays 80  
global macros  
  Synplicity 66  
globals  
  designer flow 69  
  networks 74  
  spines and rows 57

## H

HLD code  
  instantiating 258  
hot-swapping 383

## I

I/O banks  
  standards 56  
I/O standards 93  
  global macros 62  
I/Os  
  assigning technologies 264  
  assignments defined in PDC file 259  
  automatically assigning 268  
  behavior at power-up/-down 377  
  buffer schematic cell 257  
  cell architecture 273  
  configuration with SmartGen 254  
  global, naming 51  
  manually assigning technologies 264  
  software-controlled attributes 253  
  user I/O assignment flow chart 251  
idle mode 23  
INBUF\_FF 39  
ISP 289, 290  
  architecture 327  
  board-level considerations 337

  circuit 343  
  microprocessor 349

## J

JTAG 1532 327  
JTAG interface 351

## L

layout  
  device-specific 94  
LTC3025 linear voltage regulator 343

## M

MAC validation/authentication 354  
macros  
  CLKBUF 93  
  CLKBUF\_LVDS/LVPECL 93  
  CLKDLY 81, 89  
  FIFO4KX18 157  
  PLL 89  
  PLL macro signal descriptions 84  
  RAM4K9 153  
  RAM512X18 155  
  supported basic RAM macros 152  
  UJTAG 365  
  ULSICC 40  
MCU FPGA programming model 352  
memory availability 162  
memory blocks 151  
microprocessor programming 349  
Microsemi SoC Products Group  
  email 387  
  web-based technical support 387  
  website 387

## O

OTP 289

## P

PDC  
  global promotion and demotion 67  
  place-and-route 259  
PLL  
  behavior at brownout condition 381  
  configuration bits 106  
  core specifications 100  
  dynamic PLL configuration 103  
  functional description 101  
  power supply decoupling scheme 128  
PLL block signals 84  
PLL macro block diagram 85  
power conservation 41  
power modes  
  Flash\*Freeze 24  
  idle 23  
  shutdown 32