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	516096
Number of I/O	341
Number of Gates	3000000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	484-BGA
Supplier Device Package	484-FPBGA (23x23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pe3000l-fg484

Set/Reset

Since all I/Os and globals are tied High in Flash*Freeze mode (unless hold state is used on IGLOO nano or IGLOO PLUS), Microsemi recommends using active low set/reset at the top-level port. If needed, the signal can be inverted internally.

- If the intention is to always set/reset in Flash*Freeze mode, a self set/reset circuit may be implemented to accomplish this, as shown in Figure 2-9. Configure an active High set/reset input pin so it uses the internal pull-up during Flash*Freeze mode, and drives Low during active mode. When the device exits Flash*Freeze mode, the input will transition from High to Low, releasing the set/reset. Note that this circuit may release set/reset before all outputs become active, since outputs are enabled up to 200 ns after inputs when exiting Flash*Freeze mode.
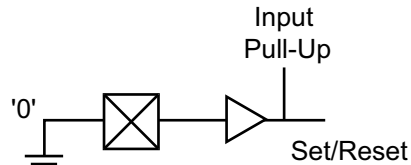

Figure 2-9 • **Flash*Freeze Self-Reset Circuit**

I/Os

- Floating inputs can cause totem pole currents on the input I/O circuitry when the device is in active mode. If inputs will be released (undriven) during Flash*Freeze mode, Microsemi recommends that they are only released after the device enters Flash*Freeze mode.

- As mentioned earlier, asynchronous input to output paths are subject to possible glitching when entering Flash*Freeze mode. For example, on a direct in-to-out path, if the current state is '0' and the input bank deactivates first, the input and then the output will transition to '1' before the output enters its Flash*Freeze state. This can be prevented by using latches along with Flash*Freeze management IP to gate asynchronous in-to-out paths prior to entering Flash*Freeze mode.

JTAG

- The JTAG state machine is powered but not active during Flash*Freeze mode.

- TCK should be held in a static state to prevent dynamic power consumption of the JTAG circuit during Flash*Freeze.

- Specific JTAG pin tie-off recommendations suitable for Flash*Freeze mode can be found in the "Pin Descriptions and Packaging" chapter of the device datasheet.

ULSICC

- The User Low Static ICC (ULSICC) macro acts as an access point to the hard Flash*Freeze technology block in the device. The ULSICC macro represents a hard, fixed location block in the device. When the LSICC input of the ULSICC macro is driven Low, the Flash*Freeze pin is blocked, and when LSICC is driven High, the Flash*Freeze pin is enabled.

- If the user decides to build his/her own Flash*Freeze type 2 clock and data management logic, note that the LSICC signal on the ULSICC macro is ANDed internally with the Flash*Freeze signal. In order to reliably enter Flash*Freeze, the LSICC signal must remain asserted High while entering and during Flash*Freeze mode.

Flash*Freeze Management IP

One of the key benefits of Microsemi's Flash*Freeze mode is the ability to preserve the state of all internal registers, SRAM content, and I/Os (IGLOO nano and IGLOO PLUS only). This feature enables seamless continuation of data processing before and after Flash*Freeze, without the need to reload or reinitialize the FPGA system. Microsemi's Flash*Freeze management IP, available for type 2 implementation, offers a robust RTL block that ensures clean clock gating of all system clocks before entering and upon exiting Flash*Freeze mode. This IP also gives users the option to perform housekeeping prior to entering Flash*Freeze mode. This section will provide an overview of the

- Avoid using pull-ups and pull-downs on I/Os because these resistors draw some current. Avoid driving resistive loads or bipolar transistors, since these draw a continuous current, thereby adding to the static current.
- When partitioning the design across multiple devices, minimize I/O usage among the devices.

Conclusion

Microsemi IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 family architectures are designed to achieve ultra-low power consumption based on enhanced nonvolatile and live-at-power-up flash-based technology. Power consumption can be reduced further by using Flash*Freeze, Static (Idle), Sleep, and Shutdown power modes. All these features result in a low power, cost-effective, single-chip solution designed specifically for power-sensitive and battery-operated electronics applications.

Related Documents

Application Notes

Embedded SRAM Initialization Using External Serial EEPROM

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

List of Changes

The following table lists critical changes that were made in each version 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
v2.3 (November 2009)	The "Sleep Mode" section was revised to state the VJTAG and VPUMP, as well as VCC, are grounded during Sleep mode (SAR 22517).	32
	Figure 2-6 • Controlling Power-On/-Off State Using Microprocessor and Power FET and Figure 2-7 • Controlling Power-On/-Off State Using Microprocessor and Voltage Regulator were revised to show that VJTAG and VPUMP are powered off during Sleep mode.	33
v2.2 (December 2008)	IGLOO nano devices were added as a supported family.	N/A
	The "Prototyping for IGLOO and ProASIC3L Devices Using ProASIC3" section was removed, as these devices are now in production.	N/A
	The "Additional Power Conservation Techniques" section was revised to add RT ProASIC3 devices.	41
v2.0 (October 2008)	The "Flash*Freeze Management FSM" section was updated with the following information: 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.	37

3 – Global Resources in Low Power Flash Devices

Introduction

IGLOO, Fusion, and ProASIC3 FPGA devices offer a powerful, low-delay VersaNet global network scheme and have extensive support for multiple clock domains. In addition to the Clock Conditioning Circuits (CCCs) and phase-locked loops (PLLs), there is a comprehensive global clock distribution network called a VersaNet global network. Each logical element (VersaTile) input and output port has access to these global networks. The VersaNet global networks can be used to distribute low-skew clock signals or high-fanout nets. In addition, these highly segmented VersaNet global networks contain spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside their region. This allows users the flexibility to create low-skew local clock networks using spines. This document describes VersaNet global networks and discusses how to assign signals to these global networks and spines in a design flow. Details concerning low power flash device PLLs are described in the "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 77. This chapter describes the low power flash devices' global architecture and uses of these global networks in designs.

Global Architecture

Low power flash devices offer powerful and flexible control of circuit timing through the use of global circuitry. Each chip has up to six CCCs, some with PLLs.

- In IGLOOe, ProASIC3EL, and ProASIC3E devices, all CCCs have PLLs—hence, 6 PLLs per device (except the PQ208 package, which has only 2 PLLs).
- In IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3, and ProASIC3L devices, the west CCC contains a PLL core (except in 10 k through 30 k devices).
- In Fusion devices, the west CCC also contains a PLL core. In the two larger devices (AFS600 and AFS1500), the west and east CCCs each contain a PLL.

Refer to Table 4-6 on page 100 for details. Each PLL includes delay lines, a phase shifter (0°, 90°, 180°, 270°), and clock multipliers/dividers. Each CCC has all the circuitry needed for the selection and interconnection of inputs to the VersaNet global network. The east and west CCCs each have access to three chip global lines on each side of the chip (six chip global lines total). The CCCs at the four corners each have access to three quadrant global lines in each quadrant of the chip (except in 10 k through 30 k gate devices).

The nano 10 k, 15 k, and 20 k devices support four VersaNet global resources, and 30 k devices support six global resources. The 10 k through 30 k devices have simplified CCCs called CCC-GLs.
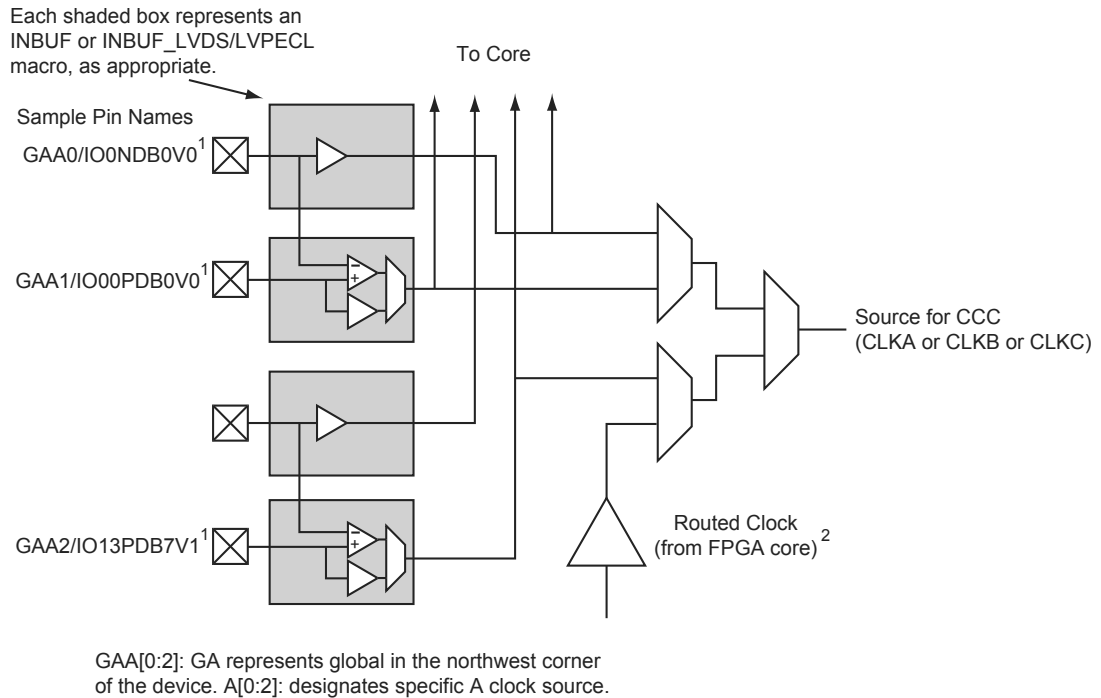
The flexible use of the VersaNet global network allows the designer to address several design requirements. User applications that are clock-resource-intensive can easily route external or gated internal clocks using VersaNet global routing networks. Designers can also drastically reduce delay penalties and minimize resource usage by mapping critical, high-fanout nets to the VersaNet global network.

Note: Microsemi recommends that you choose the appropriate global pin and use the appropriate global resource so you can realize these benefits.

The following sections give an overview of the VersaNet global network, the structure of the global network, access point for the global networks, and the clock aggregation feature that enables a design to have very low clock skew using spines.

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

GAA[0:2]: GA represents global in the northwest corner of the device. A[0:2]: designates specific A clock source.

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

Figure 3-5 • Global I/O Overview

CLKDLY Macro Usage

When a CLKDLY macro is used in a CCC location, the programmable delay element is used to allow the clock delays to go to the global network. In addition, the user can bypass the PLL in a CCC location integrated with a PLL, but use the programmable delay that is associated with the global network by instantiating the CLKDLY macro. The same is true when using programmable delay elements in a CCC location with no PLLs (the user needs to instantiate the CLKDLY macro). There is no difference between the programmable delay elements used for the PLL and the CLKDLY macro. The CCC will be configured to use the programmable delay elements in accordance with the macro instantiated by the user.

As an example, if the PLL is not used in a particular CCC location, the designer is free to specify up to three CLKDLY macros in the CCC, each of which can have its own input frequency and delay adjustment options. If the PLL core is used, assuming output to only one global clock network, the other two global clock networks are free to be used by either connecting directly from the global inputs or connecting from one or two CLKDLY macros for programmable delay.

The programmable delay elements are shown in the block diagram of the PLL block shown in Figure 4-6 on page 87. Note that any CCC locations with no PLL present contain only the programmable delay blocks going to the global networks (labeled "Programmable Delay Type 2"). Refer to the "Clock Delay Adjustment" section on page 102 for a description of the programmable delay types used for the PLL. Also refer to Table 4-14 on page 110 for Programmable Delay Type 1 step delay values, and Table 4-15 on page 110 for Programmable Delay Type 2 step delay values. CCC locations with a PLL present can be configured to utilize only the programmable delay blocks (Programmable Delay Type 2) going to the global networks A, B, and C.

Global network A can be configured to use only the programmable delay element (bypassing the PLL) if the PLL is not used in the design. Figure 4-6 on page 87 shows a block diagram of the PLL, where the programmable delay elements are used for the global networks (Programmable Delay Type 2).

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

Config. Bits	Signal	Name	Description
83	RXCSEL[1]	CLKC input selection	Select the CLKC input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 110).[2]
82	RXBSEL[1]	CLKB input selection	Select the CLKB input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 110).[2]
81	RXASEL[1]	CLKA input selection	Select the CLKA input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 110).[2]
80	RESETEN	Reset Enable	Enables (active high) the synchronization of PLL output dividers after dynamic reconfiguration (SUPDATE). The Reset Enable signal is READ-ONLY.
79	DYNCSEL	Clock Input C Dynamic Select	Configures clock input C to be sent to GLC for dynamic control.[2]
78	DYNBSEL	Clock Input B Dynamic Select	Configures clock input B to be sent to GLB for dynamic control.[2]
77	DYNASEL	Clock Input A Dynamic Select	Configures clock input A for dynamic PLL configuration.[2]
<76:74>	VCOSEL[2:0]	VCO Gear Control	Three-bit VCO Gear Control for four frequency ranges (refer to Table 4-19 on page 111 and Table 4-20 on page 111).
73	STATCSEL	MUX Select on Input C	MUX selection for clock input C[2]
72	STATBSEL	MUX Select on Input B	MUX selection for clock input B[2]
71	STATASEL	MUX Select on Input A	MUX selection for clock input A[2]
<70:66>	DLYC[4:0]	YC Output Delay	Sets the output delay value for YC.
<65:61>	DLYB[4:0]	YB Output Delay	Sets the output delay value for YB.
<60:56>	DLYGLC[4:0]	GLC Output Delay	Sets the output delay value for GLC.
<55:51>	DLYGLB[4:0]	GLB Output Delay	Sets the output delay value for GLB.
<50:46>	DLYGLA[4:0]	Primary Output Delay	Primary GLA output delay
45	XDLYSEL	System Delay Select	When selected, inserts System Delay in the feedback path in Figure 4-20 on page 101.
<44:40>	FBDLY[4:0]	Feedback Delay	Sets the feedback delay value for the feedback element in Figure 4-20 on page 101.
<39:38>	FBSEL[1:0]	Primary Feedback Delay Select	Controls the feedback MUX: no delay, include programmable delay element, or use external feedback.
<37:35>	OCMUX[2:0]	Secondary 2 Output Select	Selects from the VCO's four phase outputs for GLC/YC.
<34:32>	OBMUX[2:0]	Secondary 1 Output Select	Selects from the VCO's four phase outputs for GLB/YB.

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.*

Dynamic PLL Configuration

To generate a dynamically reconfigurable CCC, the user should select **Dynamic CCC** in the configuration section of the SmartGen GUI (Figure 4-26). This will generate both the CCC core and the configuration shift register / control bit MUX.

Figure 4-26 • **SmartGen GUI**

Even if dynamic configuration is selected in SmartGen, the user must still specify the static configuration data for the CCC (Figure 4-27). The specified static configuration is used whenever the MODE signal is set to LOW and the CCC is required to function in the static mode. The static configuration data can be used as the default behavior of the CCC where required.

Figure 4-27 • **Dynamic CCC Configuration in SmartGen**

global assignments are not allocated properly. See the "Physical Constraints for Quadrant Clocks" section for information on assigning global signals to the quadrant clock networks.

Promoted global signals will be instantiated with CLKINT macros to drive these signals onto the global network. This is automatically done by Designer when the Auto-Promotion option is selected. If the user wishes to assign the signals to the quadrant globals instead of the default chip globals, this can done by using ChipPlanner, by declaring a physical design constraint (PDC), or by importing a PDC file.

Physical Constraints for Quadrant Clocks

If it is necessary to promote global clocks (CLKBUF, CLKINT, PLL, CLKDLY) to quadrant clocks, the user can define PDCs to execute the promotion. PDCs can be created using PDC commands (pre-compile) or the MultiView Navigator (MVN) interface (post-compile). The advantage of using the PDC flow over the MVN flow is that the Compile stage is able to automatically promote any regular net to a global net before assigning it to a quadrant. There are three options to place a quadrant clock using PDC commands:

- Place a clock core (not hardwired to an I/O) into a quadrant clock location.
- Place a clock core (hardwired to an I/O) into an I/O location (set_io) or an I/O module location (set_location) that drives a quadrant clock location.
- Assign a net driven by a regular net or a clock net to a quadrant clock using the following command:

```
assign_local_clock –net <net name> -type quadrant <quadrant clock region>
```

where

`<net name>` is the name of the net assigned to the local user clock region.

`<quadrant clock region>` defines which quadrant the net should be assigned to. Quadrant clock regions are defined as UL (upper left), UR (upper right), LL (lower left), and LR (lower right).

Note: If the net is a regular net, the software inserts a CLKINT buffer on the net.

For example:

```
assign_local_clock –net localReset -type quadrant UR
```

Keep in mind the following when placing quadrant clocks using MultiView Navigator:

Hardwired I/O–Driven CCCs
- Find the associated clock input port under the Ports tab, and place the input port at one of the Gmn* locations using PinEditor or I/O Attribute Editor, as shown in Figure 4-32.

Figure 4-32 • **Port Assignment for a CCC with Hardwired I/O Clock Input**

FlashROM Applications

The SmartGen core generator is used to configure FlashROM content. You can configure each page independently. SmartGen enables you to create and modify regions within a page; these regions can be 1 to 16 bytes long (Figure 5-4).

Page Number		Byte Number in Page																
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	7																
	6																
	5																
	4																
	3																
	2																
	1																
	0																

Figure 5-4 • **FlashROM Configuration**

The FlashROM content can be changed independently of the FPGA core content. It can be easily accessed and programmed via JTAG, depending on the security settings of the device. The SmartGen core generator enables each region to be independently updated (described in the "Programming and Accessing FlashROM" section on page 138). This enables you to change the FlashROM content on a per-part basis while keeping some regions "constant" for all parts. These features allow the FlashROM to be used in diverse system applications. Consider the following possible uses of FlashROM:

- Internet protocol (IP) addressing (wireless or fixed)
- System calibration settings
- Restoring configuration after unpredictable system power-down
- Device serialization and/or inventory control
- Subscription-based business models (e.g., set-top boxes)
- Secure key storage
- Asset management tracking
- Date stamping
- Version management

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, ADDRB should be tied to ground.

Table 6-3 • **Address Pins Unused/Used for Various Supported Bus Widths**

| D×W | ADDRx | |
	Unused	Used
4k×1	None	[11:0]
2k×2	[11]	[10:0]
1k×4	[11:10]	[9:0]
512×9	[11:9]	[8:0]

Note: The "x" in ADDRx implies A or B.

DINA and DINB

These are the input data signals, and they are nine bits wide. Not all nine bits are valid in all configurations. When a data width less than nine is specified, unused high-order signals must be grounded (Table 6-4).

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, DINB should be tied to ground.

DOUTA and DOUTB

These are the nine-bit output data signals. Not all nine bits are valid in all configurations. As with DINA and DINB, high-order bits may not be used (Table 6-4). The output data on unused pins is undefined.

Table 6-4 • **Unused/Used Input and Output Data Pins for Various Supported Bus Widths**

| D×W | DINx/DOUTx | |
	Unused	Used
4k×1	[8:1]	[0]
2k×2	[8:2]	[1:0]
1k×4	[8:4]	[3:0]
512×9	None	[8:0]

Note: The "x" in DINx or DOUTx implies A or B.

RAM512X18 Macro

RAM512X18 is the two-port configuration of the same RAM block (Figure 6-5 on page 156). Like the RAM4K9 nomenclature, the RAM512X18 nomenclature refers to both the deepest possible configuration and the widest possible configuration the two-port RAM block can assume. In two-port mode, the RAM block can be configured to either the 512×9 aspect ratio or the 256×18 aspect ratio. RAM512X18 is also fully synchronous and has the following features:

- Dedicated read and write ports
- Active-low read and write enables
- Selectable pipelined or nonpipelined read
- Active-low asynchronous reset
- Designer software will automatically facilitate falling-edge clocks by bubble-pushing the inversion to previous stages.

Related Documents

Application Notes

Board-Level Considerations

http://www.microsemi.com/soc/documents/ALL_AC276_AN.pdf

User's Guides

Libero SoC User's Guide

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

IGLOO, Fusion, and ProASIC3 Macro Library Guide

http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

SmartGen Core Reference Guide

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

List of Changes

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

Date	Change	Page
August 2012	Figure 7-1 • DDR Configured I/O Block Logical Representation and Figure 7-2 • DDR Configured I/O Block Logical Representation were revised to indicate that resets on registers 1, 3, 4, and 5 are active high rather than active low. The title of the figures was revised from "I/O Block Logical Representation" (SAR 38215).	175, 181
	AGL015 and A3P015 were added to Table 7-2 • Supported I/O Standards. 1.2 V was added under single-ended I/O standards. LVCMOS 1.2 was added to Table 7-3 • VCCI Voltages and Compatible IGLOO and ProASIC3 Standards (SAR 38096).	177
	Figure 7-4 • Simplified I/O Buffer Circuitry and Table 7-7 • Programmable I/O Features (user control via I/O Attribute Editor) were modified to indicate that programmable input delay control is applicable only to ProASIC3EL and RT ProASIC3 devices (SAR 39666).	183, 188
	The following sentence is incorrect and was removed from the "LVCMOS (Low-Voltage CMOS)" section (SAR 40191): LVCMOS 2.5 V for the 30 k gate devices has a clamp diode to VCCI, but for all other devices there is no clamp diode.	184
	The hyperlink for the *Board-Level Considerations* application note was corrected (SAR 36663).	208, 210
June 2011	Figure 7-1 • DDR Configured I/O Block Logical Representation and Figure 7-2 • DDR Configured I/O Block Logical Representation were revised so that the I/O_CLR and I/O_OCLK nets are no longer joined in front of Input Register 3 but instead on the branch of the CLR/PRE signal (SAR 26052).	175, 181
	Table 7-1 • Flash-Based FPGAs was revised to remove RT ProASIC3 and add Military ProASIC3/EL in its place (SAR 31824, 31825).	176
	The "Advanced I/Os—IGLOO, ProASIC3L, and ProASIC3" section was revised. Formerly it stated, "3.3 V PCI and 3.3 V PCI-X are 5 V–tolerant." This sentence now reads, "3.3 V PCI and 3.3 V PCI-X can be configured to be 5 V–tolerant" (SAR 20983).	177

5 V Input and Output Tolerance

IGLOO and ProASIC3 devices are both 5 V-input– and 5 V–output–tolerant if certain I/O standards are selected. Table 8-6 on page 218 shows the I/O standards that support 5 V input tolerance. Only 3.3 V LVTTL/LVCMOS standards support 5 V output tolerance. Refer to the appropriate family datasheet for detailed description and configuration information.

This feature is not shown in the I/O Attribute Editor.

5 V Input Tolerance

I/Os can support 5 V input tolerance when LVTTL 3.3 V, LVCMOS 3.3 V, LVCMOS 2.5 V, and LVCMOS 2.5 V / 5.0 V configurations are used (see Table 8-13 on page 231). There are four recommended solutions for achieving 5 V receiver tolerance (see Figure 8-10 on page 233 to Figure 8-13 on page 235 for details of board and macro setups). All the solutions meet a common requirement of limiting the voltage at the input to 3.6 V or less. In fact, the I/O absolute maximum voltage rating is 3.6 V, and any voltage above 3.6 V may cause long-term gate oxide failures.

Solution 1

The board-level design must ensure that the reflected waveform at the pad does not exceed the limits provided in the recommended operating conditions 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 two external resistors as explained below. Relying on the diode clamping would create an excessive pad DC voltage of 3.3 V + 0.7 V = 4 V.

This solution requires two board resistors, as demonstrated in Figure 8-10 on page 233. Here are some examples of possible resistor values (based on a simplified simulation model with no line effects and 10 Ω transmitter output resistance, where Rtx_out_high = [VCCI – VOH] / I_{OH} and Rtx_out_low = VOL / I_{OL}).

Example 1 (high speed, high current):

Rtx_out_high = Rtx_out_low = 10 Ω

R1 = 36 Ω (±5%), P(r1)min = 0.069 Ω

R2 = 82 Ω (±5%), P(r2)min = 0.158 Ω

Imax_tx = 5.5 V / (82 × 0.95 + 36 × 0.95 + 10) = 45.04 mA

t_{RISE} = t_{FALL} = 0.85 ns at C_pad_load = 10 pF (includes up to 25% safety margin)

t_{RISE} = t_{FALL} = 4 ns at C_pad_load = 50 pF (includes up to 25% safety margin)

Example 2 (low-medium speed, medium current):

Rtx_out_high = Rtx_out_low = 10 Ω

R1 = 220 Ω (±5%), P(r1)min = 0.018 Ω

R2 = 390 Ω (±5%), P(r2)min = 0.032 Ω

Imax_tx = 5.5 V / (220 × 0.95 + 390 × 0.95 + 10) = 9.17 mA

t_{RISE} = t_{FALL} = 4 ns at C_pad_load = 10 pF (includes up to 25% safety margin)

t_{RISE} = t_{FALL} = 20 ns at C_pad_load = 50 pF (includes up to 25% safety margin)

Other values of resistors are also allowed as long as the resistors are sized appropriately to limit the voltage at the receiving end to 2.5 V < Vin(rx) < 3.6 V when the transmitter sends a logic 1. This range of Vin_dc(rx) must be assured for any combination of transmitter supply (5 V ± 0.5 V), transmitter output resistance, and board resistor tolerances.

Temporary overshoots are allowed according to the overshoot and undershoot table in the datasheet.

Instantiating in HDL code

All the supported I/O macros can be instantiated in the top-level HDL code (refer to the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide* for a detailed list of all I/O macros). The following is an example:

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3e;

entity TOP is
  port(IN2, IN1 : in std_logic; OUT1 : out std_logic);
end TOP;

architecture DEF_ARCH of TOP is

  component INBUF_LVCMOS5U
    port(PAD : in std_logic := 'U'; Y : out std_logic);
  end component;

  component INBUF_LVCMOS5
    port(PAD : in std_logic := 'U'; Y : out std_logic);
  end component;

  component OUTBUF_SSTL3_II
    port(D : in std_logic := 'U'; PAD : out std_logic);
  end component;

  Other component …..

signal x, y, z…….other signals : std_logic;

begin

  I1 : INBUF_LVCMOS5U
    port map(PAD => IN1, Y =>x);
  I2 : INBUF_LVCMOS5
    port map(PAD => IN2, Y => y);
  I3 : OUTBUF_SSTL3_II
    port map(D => z, PAD => OUT1);

  other port mapping…

end DEF_ARCH;
```

Synthesizing the Design

Libero SoC integrates with the Synplify® synthesis tool. Other synthesis tools can also be used with Libero SoC. Refer to the *Libero SoC User's Guide* or Libero online help for details on how to set up the Libero tool profile with synthesis tools from other vendors.

During synthesis, the following rules apply:

- Generic macros:
  – Users can instantiate generic INBUF, OUTBUF, TRIBUF, and BIBUF macros.
  – Synthesis will automatically infer generic I/O macros.
  – The default I/O technology for these macros is LVTTL.
  – Users will need to use the I/O Attribute Editor in Designer to change the default I/O standard if needed (see Figure 9-6 on page 259).
- Technology-specific I/O macros:
  – Technology-specific I/O macros, such as INBUF_LVCMO25 and OUTBUF_GTL25, can be instantiated in the design. Synthesis will infer these I/O macros in the netlist.

I/O Function

Figure 9-8 shows an example of the I/O Function table included in the I/O bank report:

Figure 9-8 • **I/O Function Table**

This table lists the number of input I/Os, output I/Os, bidirectional I/Os, and differential input and output I/O pairs that use I/O and DDR registers.

Note: IGLOO nano and ProASIC3 nano devices do not support differential inputs.

Certain rules must be met to implement registered and DDR I/O functions (refer to the I/O Structures section of the handbook for the device you are using and the "DDR" section on page 256).

I/O Technology

The I/O Technology table (shown in Figure 9-9) gives the values of VCCI and VREF (reference voltage) for all the I/O standards used in the design. The user should assign these voltages appropriately.

Figure 9-9 • **I/O Technology Table**

```
module ddr_test(DIN, CLK, CLR, DOUT);

input  DIN, CLK, CLR;
output DOUT;

  Inbuf_ddr Inbuf_ddr (.PAD(DIN), .CLR(clr), .CLK(clk), .QR(qr), .QF(qf));
  Outbuf_ddr Outbuf_ddr (.DataR(qr),.DataF(qf), .CLR(clr), .CLK(clk),.PAD(DOUT));

  INBUF INBUF_CLR (.PAD(CLR), .Y(clr));
  INBUF INBUF_CLK (.PAD(CLK), .Y(clk));

endmodule
```

Simulation Consideration

Microsemi DDR simulation models use inertial delay modeling by default (versus transport delay modeling). As such, pulses that are shorter than the actual gate delays should be avoided, as they will not be seen by the simulator and may be an issue in post-routed simulations. The user must be aware of the default delay modeling and must set the correct delay model in the simulator as needed.

Conclusion

Fusion, IGLOO, and ProASIC3 devices support a wide range of DDR applications with different I/O standards and include built-in DDR macros. The powerful capabilities provided by SmartGen and its GUI can simplify the process of including DDR macros in designs and minimize design errors. Additional considerations should be taken into account by the designer in design floorplanning and placement of I/O flip-flops to minimize datapath skew and to help improve system timing margins. Other system-related issues to consider include PLL and clock partitioning.
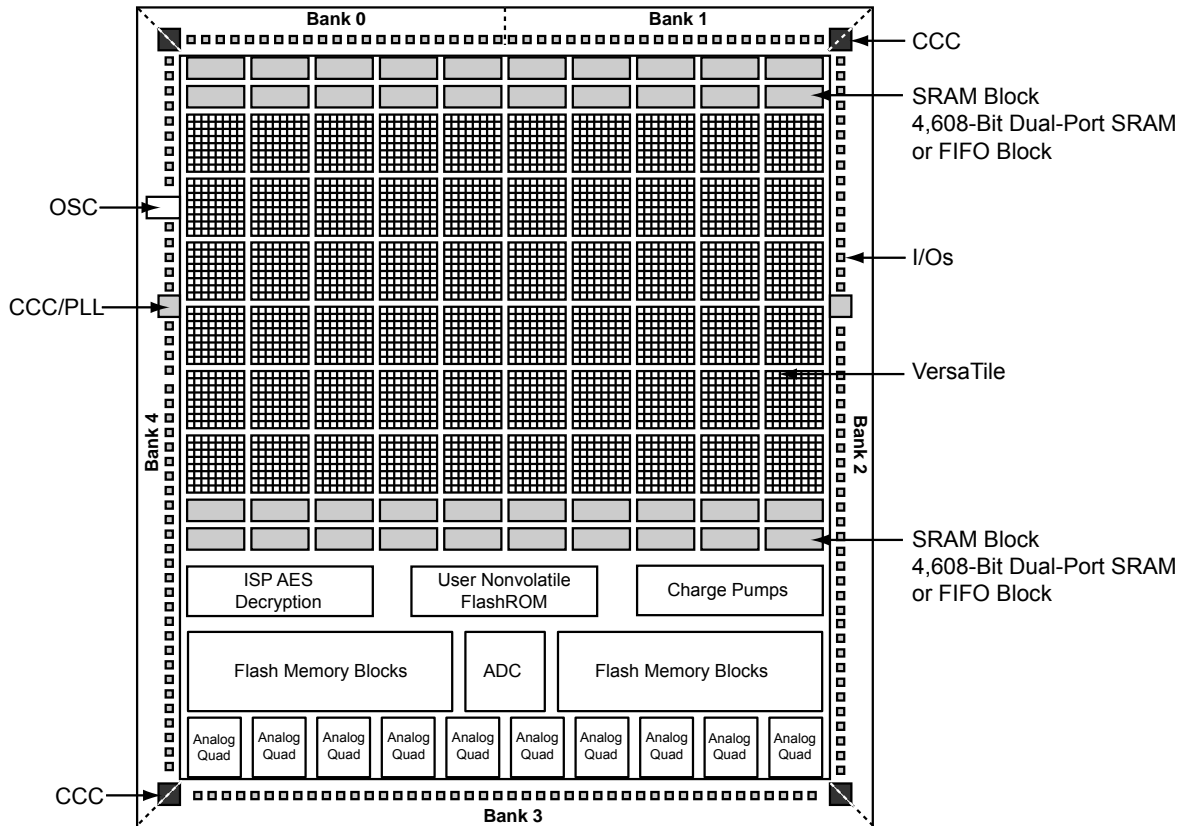
Figure 12-3 • **Block Representation of the AES Decryption Core in a Fusion AFS600 FPGA**

Security Features

IGLOO and ProASIC3 devices have two entities inside: FlashROM and the FPGA core fabric. Fusion devices contain three entities: FlashROM, FBs, and the FPGA core fabric. The parts can be programmed or updated independently with a STAPL programming file. The programming files can be AES-encrypted or plaintext. This allows maximum flexibility in providing security to the entire device. Refer to the "Programming Flash Devices" section on page 287 for information on the FlashROM structure.

Unlike SRAM-based FPGA devices, which require a separate boot PROM to store programming data, low power flash devices are nonvolatile, and the secured configuration data is stored in on-chip flash cells that are part of the FPGA fabric. Once programmed, this data is an inherent part of the FPGA array and does not need to be loaded at system power-up. SRAM-based FPGAs load the configuration bitstream upon power-up; therefore, the configuration is exposed and can be read easily.

The built-in FPGA core, FBs, and FlashROM support programming files encrypted with the 128-bit AES (FIPS-192) block ciphers. The AES key is stored in dedicated, on-chip flash memory and can be programmed before the device is shipped to other parties (allowing secure remote field updates).

Security in ARM-Enabled Low Power Flash Devices

There are slight differences between the regular flash devices and the ARM®-enabled flash devices, which have the M1 and M7 prefix.

The AES key is used by Microsemi and preprogrammed into the device to protect the ARM IP. As a result, the design is encrypted along with the ARM IP, according to the details below.

3. Choose the desired settings for the FlashROM configurations to be programmed (Figure 12-13). Click **Finish** to generate the STAPL programming file for the design.

Figure 12-13 • **FlashROM Configuration Settings for Low Power Flash Devices**

Generation of Security Header Programming File Only— Application 2

As mentioned in the "Application 2: Nontrusted Environment—Unsecured Location" section on page 309, the designer may employ FlashLock Pass Key protection or FlashLock Pass Key with AES encryption on the device before sending it to a nontrusted or unsecured location for device programming. To achieve this, the user needs to generate a programming file containing only the security settings desired (Security Header programming file).

Note: If AES encryption is configured, FlashLock Pass Key protection must also be configured.

The available security options are indicated in Table 12-4 and Table 12-5 on page 317.

Table 12-4 • **FlashLock Security Options for IGLOO and ProASIC3**

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

Figure 12-18 • **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 313 and "Generation of Security Header Programming File Only—Application 2" section on page 316. 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 12-10 on page 314 and Figure 12-11 on page 314 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 12-19 on page 322.

    3. VCC switches from 1.5 V to 1.2 V when TRST is LOW.

Figure 14-4 • **TRST Toggled LOW**

In Figure 14-4, the TRST signal and the VCC core voltage signal are labeled. As TRST is pulled to ground, the core voltage is observed to switch from 1.5 V to 1.2 V. The observed fall time is approximately 2 ms.

DirectC

The above analysis is based on FlashPro3, but there are other solutions to ISP, such as DirectC. DirectC is a microprocessor program that can be run in-system to program Microsemi flash devices. For FlashPro3, TRST is the most convenient control signal to use for the recommended circuit. However, for DirectC, users may use any signal to control the FET. For example, the DirectC code can be edited so that a separate non-JTAG signal can be asserted from the microcontroller that signals the board that it is about to start programming the device. After asserting the N-Channel Digital FET control signal, the programming algorithm must allow sufficient time for the supply to rise to 1.5 V before initiating DirectC programming. As seen in Figure 14-3 on page 345, 50 ms is adequate time. Depending on the size of the PCB and the capacitance on the VCC supply, results may vary from system to system. Microsemi recommends using a conservative value for the wait time to make sure that the VCC core voltage is at the right level.

Conclusion

For applications using IGLOO and ProASIC3L low power FPGAs and taking advantage of the low core voltage power supplies with less than 1.5 V operation, there must be a way for the core voltage to switch from 1.2 V (or other voltage) to 1.5 V, which is required during in-system programming. The circuit explained in this document illustrates one simple, cost-effective way of handling this requirement. A JTAG signal from the FlashPro3 programmer allows the circuit to sense when programming is in progress, enabling it to switch to the correct core voltage.

Microsemi

Typical UJTAG Applications

Bidirectional access to the JTAG port from VersaTiles—without putting the device into test mode—creates flexibility to implement many different applications. This section describes a few of these. All are based on importing/exporting data through the UJTAG tiles.

Clock Conditioning Circuitry—Dynamic Reconfiguration

In low power flash devices, CCCs, which include PLLs, can be configured dynamically through either an 81-bit embedded shift register or static flash programming switches. These 81 bits control all the characteristics of the CCC: routing MUX architectures, delay values, divider values, etc. Table 17-3 lists the 81 configuration bits in the CCC.

Table 17-3 • **Configuration Bits of Fusion, IGLOO, and ProASIC3 CCC Blocks**

Bit Number(s)	Control Function
80	RESET ENABLE
79	DYNCSEL
78	DYNBSEL
77	DYNASEL
<76:74>	VCOSEL [2:0]
73	STATCSEL
72	STATBSEL
71	STATASEL
<70:66>	DLYC [4:0]
<65:61>	DLYB {4:0]
<60:56>	DLYGLC [4:0]
<55:51>	DLYGLB [4:0]
<50:46>	DLYGLA [4:0]
45	XDLYSEL
<44:40>	FBDLY [4:0]
<39:38>	FBSEL
<37:35>	OCMUX [2:0]
<34:32>	OBMUX [2:0]
<31:29>	OAMUX [2:0]
<28:24>	OCDIV [4:0]
<23:19>	OBDIV [4:0]
<18:14>	OADIV [4:0]
<13:7>	FBDIV [6:0]
<6:0>	FINDIV [6:0]

The embedded 81-bit shift register (for the dynamic configuration of the CCC) is accessible to the VersaTiles, which, in turn, have access to the UJTAG tiles. Therefore, the CCC configuration shift register can receive and load the new configuration data stream from JTAG.

Dynamic reconfiguration eliminates the need to reprogram the device when reconfiguration of the CCC functional blocks is needed. The CCC configuration can be modified while the device continues to operate. Employing the UJTAG core requires the user to design a module to provide the configuration data and control the CCC configuration shift register. In essence, this is a user-designed TAP Controller requiring chip resources.

Similar reconfiguration capability exists in the ProASIC^PLUS® family. The only difference is the number of shift register bits controlling the CCC (27 in ProASIC^PLUS and 81 in IGLOO, ProASIC3, and Fusion).