



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	Obsolete
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	36864
Number of I/O	97
Number of Gates	250000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	144-LBGA
Supplier Device Package	144-FPBGA (13x13)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/a3p250l-1fg144

Flash*Freeze Mode

IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 FPGAs offer an ultra-low static power mode to reduce power consumption while preserving the state of the registers, SRAM contents, and I/O states (IGLOO nano and IGLOO PLUS only) without switching off any power supplies, inputs, or input clocks.

Flash*Freeze technology enables the user to switch to Flash*Freeze mode within 1 μ s, thus simplifying low power design implementation. The Flash*Freeze (FF) pin (active Low) is a dedicated pin used to enter or exit Flash*Freeze mode directly; or the pin can be routed internally to the FPGA core and state management IP to allow the user's application to decide if and when it is safe to transition to this mode. If the FF pin is not used, it can be used as a regular I/O.

The FF pin has a built-in glitch filter and optional Schmitt trigger (not available for all devices) to prevent entering or exiting Flash*Freeze mode accidentally.

There are two ways to use Flash*Freeze mode. In Flash*Freeze type 1, entering and exiting the mode is exclusively controlled by the assertion and deassertion of the FF pin. This enables an external processor or human interface device to directly control Flash*Freeze mode; however, valid data must be preserved using standard procedures (refer to the "Flash*Freeze Mode Device Behavior" section on page 30). In Flash*Freeze mode type 2, entering and exiting the mode is controlled by both the FF pin AND user-defined logic. Flash*Freeze management IP may be used in type 2 mode for clock and data management while entering and exiting Flash*Freeze mode.

Flash*Freeze Type 1: Control by Dedicated Flash*Freeze Pin

Flash*Freeze type 1 is intended for systems where either the device will be reset upon exiting Flash*Freeze mode, or data and clock are managed externally. The device enters Flash*Freeze mode 1 μ s after the dedicated FF pin is asserted (active Low), and returns to normal operation when the FF pin is deasserted (High) (Figure 2-1 on page 25). In this mode, FF pin assertion or deassertion is the only condition that determines entering or exiting Flash*Freeze mode.

In Libero[®] System-on-Chip (SoC) software v8.2 and before, this mode is implemented by enabling Flash*Freeze mode (default setting) in the Compile options of the Microsemi Designer software. To simplify usage of Flash*Freeze mode, beginning with Libero software v8.3, an INBUF_FF I/O macro was introduced. An INBUF_FF I/O buffer must be used to identify the Flash*Freeze input. Microsemi recommends switching to the new implementation.

In Libero software v8.3 and later, the user must manually instantiate the INBUF_FF macro in the top level of the design to implement Flash*Freeze Type 1, as shown in Figure 2-1 on page 25.

standard for CLKBUF is LVTTTL in the current Microsemi Libero® System-on-Chip (SoC) and Designer software.

Table 3-9 • I/O Standards within CLKBUF

Name	Description
CLKBUF_LVCMOS5	LVCMOS clock buffer with 5.0 V CMOS voltage level
CLKBUF_LVCMOS33	LVCMOS clock buffer with 3.3 V CMOS voltage level
CLKBUF_LVCMOS25	LVCMOS clock buffer with 2.5 V CMOS voltage level ¹
CLKBUF_LVCMOS18	LVCMOS clock buffer with 1.8 V CMOS voltage level
CLKBUF_LVCMOS15	LVCMOS clock buffer with 1.5 V CMOS voltage level
CLKBUF_LVCMOS12	LVCMOS clock buffer with 1.2 V CMOS voltage level
CLKBUF_PCI	PCI clock buffer
CLKBUF_PCIX	PCIX clock buffer
CLKBUF_GTL25	GTL clock buffer with 2.5 V CMOS voltage level ¹
CLKBUF_GTL33	GTL clock buffer with 3.3 V CMOS voltage level ¹
CLKBUF_GTLP25	GTL+ clock buffer with 2.5 V CMOS voltage level ¹
CLKBUF_GTLP33	GTL+ clock buffer with 3.3 V CMOS voltage level ¹
CLKBUF_HSTL_I	HSTL Class I clock buffer ¹
CLKBUF_HSTL_II	HSTL Class II clock buffer ¹
CLKBUF_SSTL2_I	SSTL2 Class I clock buffer ¹
CLKBUF_SSTL2_II	SSTL2 Class II clock buffer ¹
CLKBUF_SSTL3_I	SSTL3 Class I clock buffer ¹
CLKBUF_SSTL3_II	SSTL3 Class II clock buffer ¹

Notes:

1. Supported in only the IGLOOe, ProASIC3E, AFS600, and AFS1500 devices
2. By default, the CLKBUF macro uses the 3.3 V LVTTTL I/O technology.

The current synthesis tool libraries only infer the CLKBUF or CLKINT macros in the netlist. All other global macros must be instantiated manually into your HDL code. The following is an example of CLKBUF_LVCMOS25 global macro instantiations that you can copy and paste into your code:

VHDL

```
component clkbuf_lvcmos25
  port (pad : in std_logic; y : out std_logic);
end component

begin
  -- concurrent statements
  u2 : clkbuf_lvcmos25 port map (pad => ext_clk, y => int_clk);
end
```

Verilog

```
module design (____);

  input ____;
  output ____;

  clkbuf_lvcmos25 u2 (.y(int_clk), .pad(ext_clk));

endmodule
```


You can control the maximum number of shared instances allowed for the legalization to take place using the Compile Option dialog box shown in Figure 3-17. Refer to Libero SoC / Designer online help for details on the Compile Option dialog box. A large number of shared instances most likely indicates a floorplanning problem that you should address.

Figure 3-17 • Shared Instances in the Compile Option Dialog Box

Designer Flow for Global Assignment

To achieve the desired result, pay special attention to global management during synthesis and place-and-route. The current Synplify tool does not insert more than six global buffers in the netlist by default. Thus, the default flow will not assign any signal to the quadrant global network. However, you can use attributes in Synplify and increase the default global macro assignment in the netlist. Designer v6.2 supports automatic quadrant global assignment, which was not available in Designer v6.1. Layout will make the choice to assign the correct signals to global. However, you can also utilize PDC and perform manual global assignment to overwrite any automatic assignment. The following step-by-step suggestions guide you in the layout of your design and help you improve timing in Designer:

1. Run Compile and check the Compile report. The Compile report has global information in the "Device Utilization" section that describes the number of chip and quadrant signals in the design. A "Net Report" section describes chip global nets, quadrant global nets, local clock nets, a list of nets listed by fanout, and net candidates for local clock assignment. Review this information. Note that YB or YC are counted as global only when they are used in isolation; if you use YB only and not GLB, this net is not shown in the global/quadrant nets report. Instead, it appears in the Global Utilization report.
2. If some signals have a very high fanout and are candidates for global promotion, promote those signals to global using the compile options or PDC commands. Figure 3-18 on page 70 shows the Globals Management section of the compile options. Select **Promote regular nets whose fanout is greater than** and enter a reasonable value for fanouts.

difference will cause the VCO to increase its frequency until the output signal is phase-identical to the input after undergoing division. In other words, lock in both frequency and phase is achieved when the output frequency is M times the input. Thus, clock division in the feedback path results in multiplication at the output.

A similar argument can be made when the delay element is inserted into the feedback path. To achieve steady-state lock, the VCO output signal will be delayed by the input period less the feedback delay. For periodic signals, this is equivalent to time-advancing the output clock by the feedback delay.

Another key parameter of a PLL system is the acquisition time. Acquisition time is the amount of time it takes for the PLL to achieve lock (i.e., phase-align the feedback signal with the input reference clock). For example, suppose there is no voltage applied to the VCO, allowing it to operate at its free-running frequency. Should an input reference clock suddenly appear, a lock would be established within the maximum acquisition time.

Functional Description

This section provides detailed descriptions of PLL block functionality: clock dividers and multipliers, clock delay adjustment, phase adjustment, and dynamic PLL configuration.

Clock Dividers and Multipliers

The PLL block contains five programmable dividers. Figure 4-20 shows a simplified PLL block.

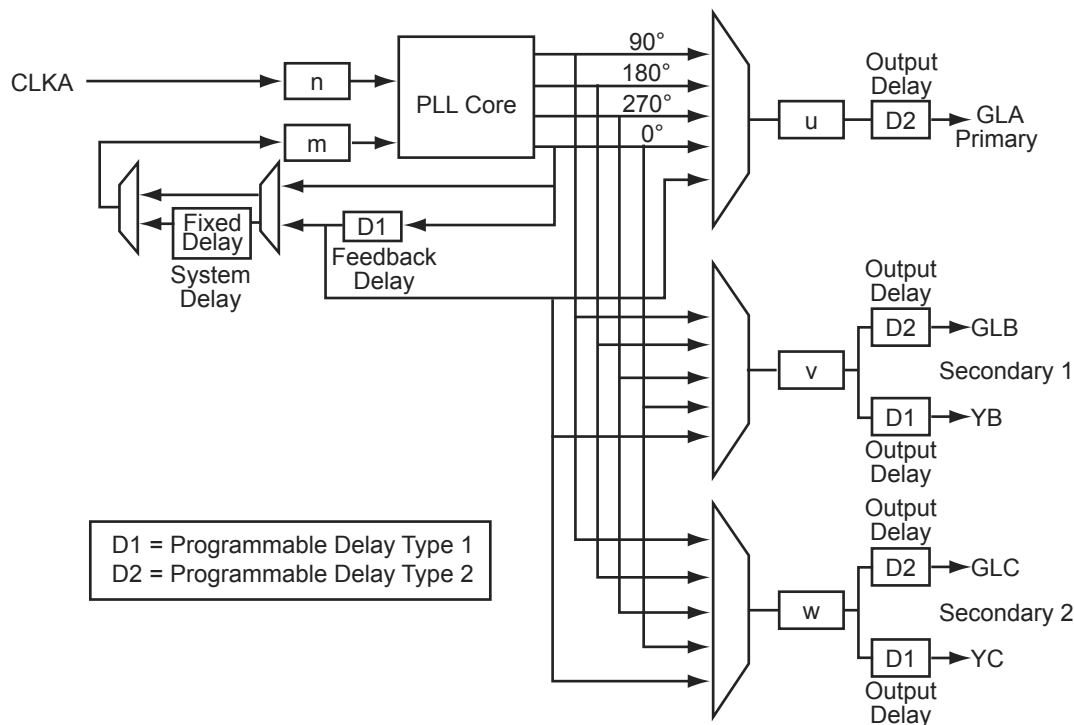


Figure 4-20 • PLL Block Diagram

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 CLKA. 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.

Advanced I/Os—IGLOO, ProASIC3L, and ProASIC3

Table 7-2 and Table 7-3 show the voltages and compatible I/O standards for the IGLOO, ProASIC3L, and ProASIC3 families.

I/Os provide programmable slew rates (except 30 K gate devices), drive strengths, and weak pull-up and pull-down circuits. 3.3 V PCI and 3.3 V PCI-X can be configured to be 5 V-tolerant. See the "5 V Input Tolerance" section on page 194 for possible implementations of 5 V tolerance.

All I/Os are in a known state during power-up, and any power-up sequence is allowed without current impact. Refer to the "I/O Power-Up and Supply Voltage Thresholds for Power-On Reset (Commercial and Industrial)" section in the datasheet for more information. During power-up, before reaching activation levels, the I/O input and output buffers are disabled while the weak pull-up is enabled. Activation levels are described in the datasheet.

Table 7-2 • Supported I/O Standards

IGLOO	AGL015	AGL030	AGL060	AGL125	AGL250		AGL600	AGL1000
ProASIC3	A3P015	A3P030	A3P060	A3P125	A3P250/ A3P250L	A3P400	A3P600/ A3P600L	A3P1000/ A3P1000L
Single-Ended								
LVTTTL/LVCMOS 3.3 V, LVCMOS 2.5 V / 1.8 V / 1.5 V / 1.2 V LVCMOS 2.5 V / 5.0 V	✓	✓	✓	✓	✓	✓	✓	✓
3.3 V PCI/PCI-X	–	–	✓	✓	✓	✓	✓	✓
Differential								
LVPECL, LVDS, B-LVDS, M-LVDS	–	–	–	–	✓	✓	✓	✓

I/O Banks and I/O Standards Compatibility

I/Os are grouped into I/O voltage banks.

Each I/O voltage bank has dedicated I/O supply and ground voltages (VMV/GNDQ for input buffers and VCCI/GND for output buffers). This isolation is necessary to minimize simultaneous switching noise from the input and output (SSI and SSO). The switching noise (ground bounce and power bounce) is generated by the output buffers and transferred into input buffer circuits, and vice versa. Because of these dedicated supplies, only I/Os with compatible standards can be assigned to the same I/O voltage bank. Table 7-3 shows the required voltage compatibility values for each of these voltages.

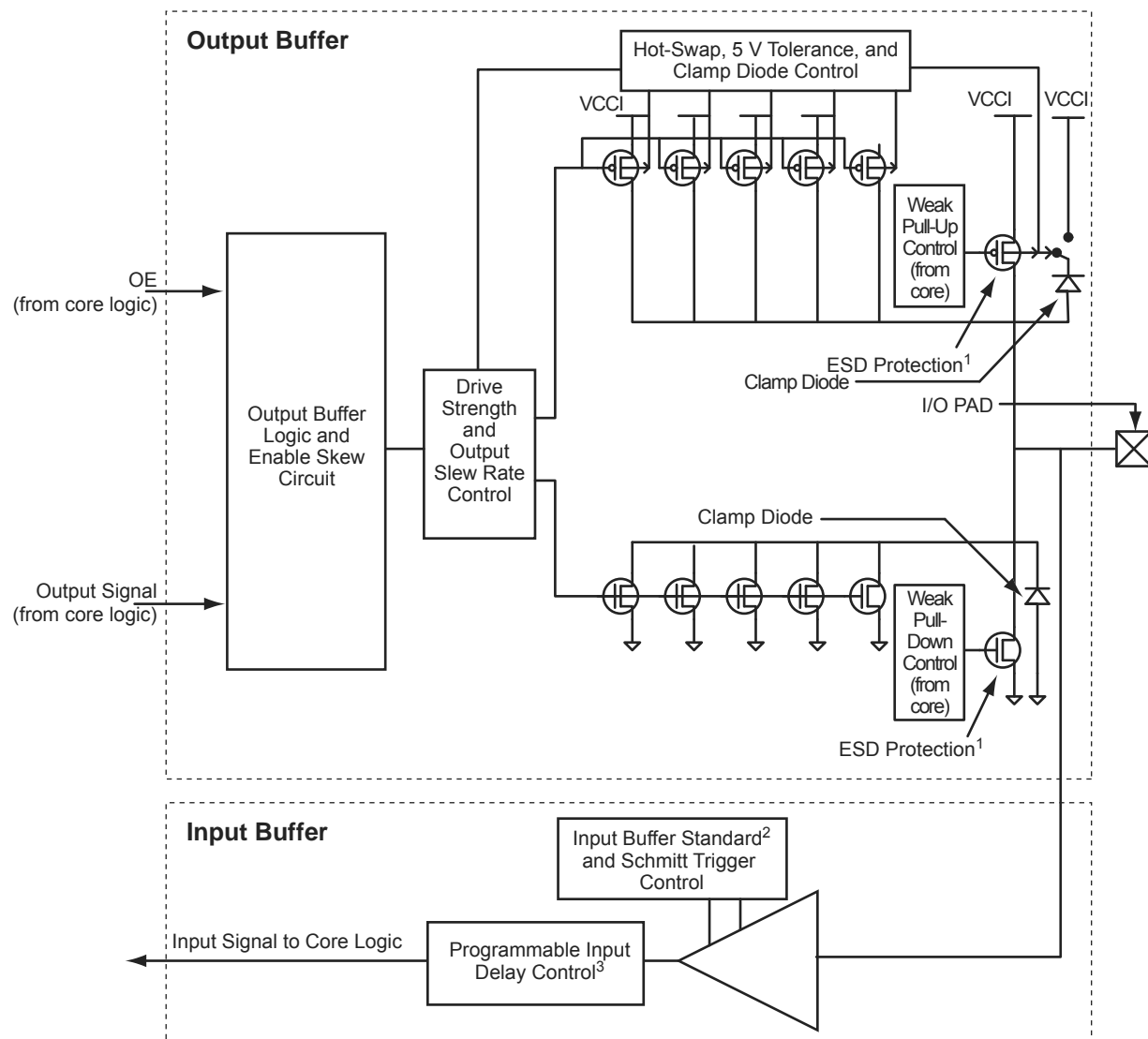
There are four I/O banks on the 250K gate through 1M gate devices.

There are two I/O banks on the 30K, 60K, and 125K gate devices.

I/O standards are compatible if their VCCI and VMV values are identical. VMV and GNDQ are "quiet" input power supply pins and are not used on 30K gate devices (Table 7-3).

Table 7-3 • VCCI Voltages and Compatible IGLOO and ProASIC3 Standards

VCCI and VMV (typical)	Compatible Standards
3.3 V	LVTTTL/LVCMOS 3.3, PCI 3.3, PCI-X 3.3 LVPECL
2.5 V	LVCMOS 2.5, LVCMOS 2.5/5.0, LVDS, B-LVDS, M-LVDS
1.8 V	LVCMOS 1.8
1.5 V	LVCMOS 1.5
1.2 V	LVCMOS 1.2



Notes:

1. All NMOS transistors connected to the I/O pad serve as ESD protection.
2. See Table 7-2 on page 177 for available I/O standards.
3. Programmable input delay is applicable only to ProASIC3EL and RT ProASIC3 devices.

Figure 7-4 • Simplified I/O Buffer Circuitry

I/O Registers

Each I/O module contains several input, output, and enable registers. Refer to Figure 7-4 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 7-2 on page 181) 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.

I/O Architecture

I/O Tile

The I/O tile provides a flexible, programmable structure for implementing a large number of I/O standards. In addition, the registers available in the I/O tile can be used to support high-performance register inputs and outputs, with register enable if desired (Figure 8-3). The registers can also be used to support the JESD-79C Double Data Rate (DDR) standard within the I/O structure (see the "DDR for Microsemi's Low Power Flash Devices" section on page 271 for more information).

As depicted in Figure 8-3, all I/O registers share one CLR port. The output register and output enable register share one CLK port.

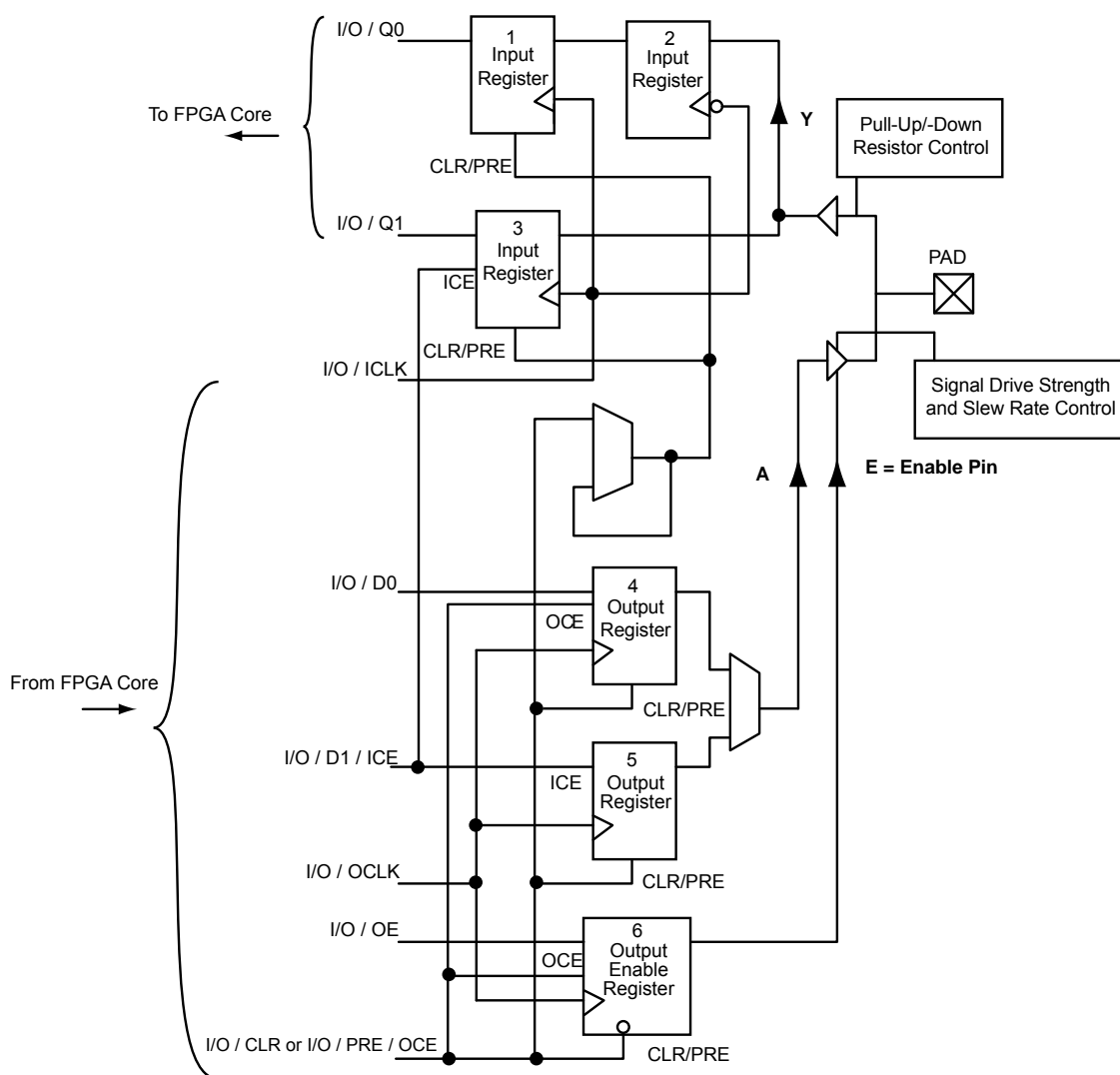


Figure 8-3 • DDR Configured I/O Block Logical Representation

Flash FPGAs I/O Support

The flash FPGAs listed in Table 9-1 support I/Os and the functions described in this document.

Table 9-1 • Flash-Based FPGAs

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO 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
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

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

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 9-1. 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 9-1. 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*.

VREF for GTL+ 3.3 V



Figure 9-13 • Selecting VREF Voltage for the I/O Bank

Assigning VREF Pins for a Bank

The user can use default pins for VREF. In this case, select the **Use default pins for VREFs** check box (Figure 9-13). This option guarantees full VREF coverage of the bank. The equivalent PDC command is as follows:

```
set_vref_default [bank name]
```

To be able to choose VREF pins, adequate VREF pins must be created to allow legal placement of the compatible voltage-referenced I/Os.

To assign VREF pins manually, the PDC command is as follows:

```
set_vref -bank [bank name] [package pin numbers]
```

For ChipPlanner/PinEditor to show the range of a VREF pin, perform the following steps:

1. Assign VCCI to a bank using **MVN > Edit > I/O Bank Settings**.
2. Open **ChipPlanner**. Zoom in on an I/O package pin in that bank.
3. Highlight the pin and then right-click. Choose **Use Pin for VREF**.

Instantiating DDR Registers

Using SmartGen is the simplest way to generate the appropriate RTL files for use in the design. Figure 10-4 shows an example of using SmartGen to generate a DDR SSTL2 Class I input register. SmartGen provides the capability to generate all of the DDR I/O cells as described. The user, through the graphical user interface, can select from among the many supported I/O standards. The output formats supported are Verilog, VHDL, and EDIF.

Figure 10-5 on page 277 through Figure 10-8 on page 280 show the I/O cell configured for DDR using SSTL2 Class I technology. For each I/O standard, the I/O pad is buffered by a special primitive that indicates the I/O standard type.

Figure 10-4 • Example of Using SmartGen to Generate a DDR SSTL2 Class I Input Register

DDR Tristate Output Register

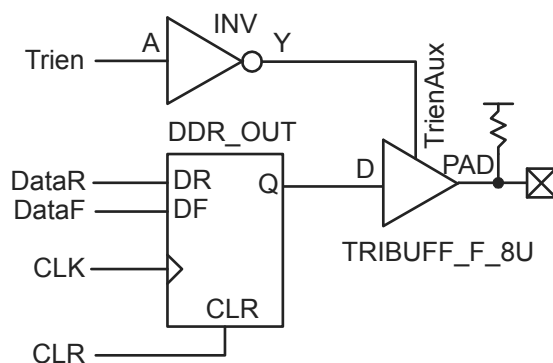


Figure 10-7 • DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTTL)

Verilog

```
module DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp(DataR, DataF, CLR, CLK, Trien,
    PAD);

    input  DataR, DataF, CLR, CLK, Trien;
    output PAD;

    wire TrienAux, Q;

    INV Inv_Tri(.A(Trien),.Y(TrienAux));
    DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
    TRIBUFF_F_8U TRIBUFF_F_8U_0_inst(.D(Q),.E(TrienAux),.PAD(PAD));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3; use proasic3.all;

entity DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is
    port(DataR, DataF, CLR, CLK, Trien : in std_logic; PAD : out std_logic) ;
end DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp;

architecture DEF_ARCH of DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is

    component INV
        port(A : in std_logic := 'U'; Y : out std_logic) ;
    end component;

    component DDR_OUT
        port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
    end component;

    component TRIBUFF_F_8U
        port(D, E : in std_logic := 'U'; PAD : out std_logic) ;
    end component;

    signal TrienAux, Q : std_logic ;

begin

    Inv_Tri : INV
        port map(A => Trien, Y => TrienAux);
```

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
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 10-1 • Flash-Based FPGAs.	272
	The "I/O Cell Architecture" section was updated with information applicable to nano devices.	273
	The output buffer (OUTBUF_SSTL3_I) input was changed to D, instead of Q, in Figure 10-1 • DDR Support in Low Power Flash Devices, Figure 10-3 • DDR Output Register (SSTL3 Class I), Figure 10-6 • DDR Output Register (SSTL3 Class I), Figure 10-7 • DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTTL), and the output from the DDR_OUT macro was connected to the input of the TRIBUFF macro in Figure 10-7 • DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTTL).	271, 275, 278, 279
v1.3 (October 2008)	The "Double Data Rate (DDR) Architecture" section was updated to include mention of the AFS600 and AFS1500 devices.	271
	The "DDR Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	272
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 10-1 • Flash-Based FPGAs: <ul style="list-style-type: none"> ProASIC3L was updated to include 1.5 V. The number of PLLs for ProASIC3E was changed from five to six. 	272
v1.1 (March 2008)	The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	272

FlashROM Security Use Models

Each of the subsequent sections describes in detail the available selections in Microsemi Designer as an aid to understanding security applications and generating appropriate programming files for those applications. Before proceeding, it is helpful to review Figure 12-7 on page 309, which gives a general overview of the programming file generation flow within the Designer software as well as what occurs during the device programming stage. Specific settings are discussed in the following sections.

In Figure 12-7 on page 309, the flow consists of two sub-flows. Sub-flow 1 describes programming security settings to the device only, and sub-flow 2 describes programming the design contents only.

In Application 1, described in the "Application 1: Trusted Environment" section on page 309, the user does not need to generate separate files but can generate one programming file containing both security settings and design contents. Then programming of the security settings and design contents is done in one step. Both sub-flow 1 and sub-flow 2 are used.

In Application 2, described in the "Application 2: Nontrusted Environment—Unsecured Location" section on page 309, the trusted site should follow sub-flows 1 and 2 separately to generate two separate programming files. The programming file from sub-flow 1 will be used at the trusted site to program the device(s) first. The programming file from sub-flow 2 will be sent off-site for production programming.

In Application 3, described in the "Application 3: Nontrusted Environment—Field Updates/Upgrades" section on page 310, typically only sub-flow 2 will be used, because only updates to the design content portion are needed and no security settings need to be changed.

In the event that update of the security settings is necessary, see the "Reprogramming Devices" section on page 321 for details. For more information on programming low power flash devices, refer to the "In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" section on page 327.

Related Documents

User's Guides

FlashPro User's Guide

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"> ProASIC3L was updated to include 1.5 V. The number of PLLs for ProASIC3E was changed from five to six. 	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

3. A single STAPL file or multiple STAPL files with multiple FlashROM contents. A single STAPL file will be generated if the device serialization feature is not used. You can program the whole FlashROM or selectively program individual pages.
4. A single STAPL file to configure the security settings for the device, such as the AES Key and/or Pass Key.

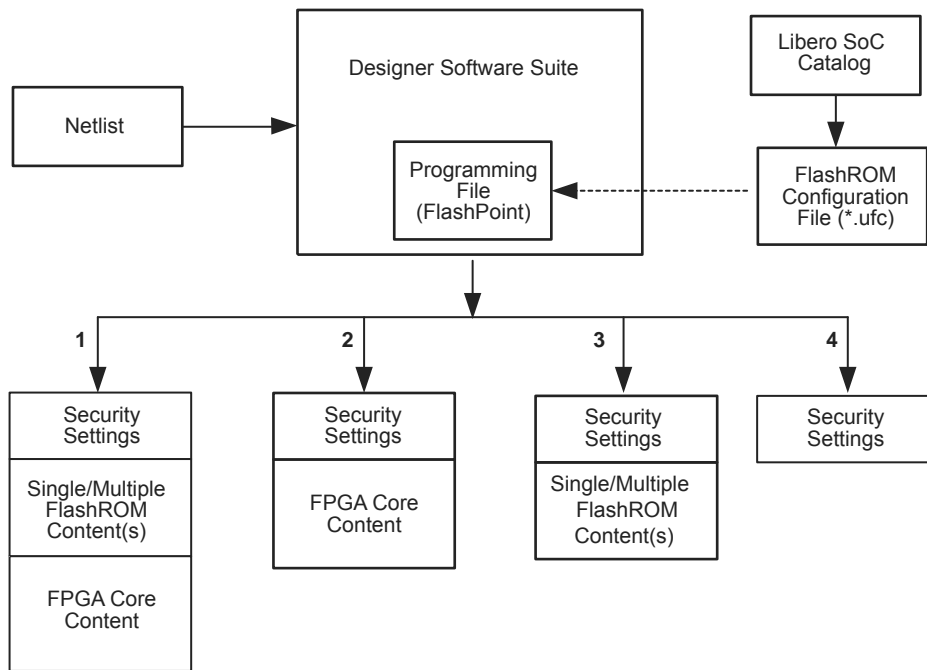


Figure 13-4 • Flexible Programming File Generation for Different Applications

Programming Solution

For device programming, any IEEE 1532-compliant programmer can be used; however, the FlashPro4/3/3X programmer must be used to control the low power flash device's rich security features and FlashROM programming options. The FlashPro4/3/3X programmer is a low-cost portable programmer for the Microsemi flash families. It can also be used with a powered USB hub for parallel programming. General specifications for the FlashPro4/3/3X programmer are as follows:

- Programming clock – TCK is used with a maximum frequency of 20 MHz, and the default frequency is 4 MHz.
- Programming file – STAPL
- Daisy chain – Supported. You can use the ChainBuilder software to build the programming file for the chain.
- Parallel programming – Supported. Multiple FlashPro4/3/3X programmers can be connected together using a powered USB hub or through the multiple USB ports on the PC.
- Power supply – The target board must provide VCC, VCCI, VPUMP, and VJTAG during programming. However, if there is only one device on the target board, the FlashPro4/3/3X programmer can generate the required VPUMP voltage from the USB port.

15 – Microprocessor Programming of Microsemi's Low Power Flash Devices

Introduction

The Fusion, IGLOO, and ProASIC3 families of flash FPGAs support in-system programming (ISP) with the use of a microprocessor. Flash-based FPGAs store their configuration information in the actual cells within the FPGA fabric. SRAM-based devices need an external configuration memory, and hybrid nonvolatile devices store the configuration in a flash memory inside the same package as the SRAM FPGA. Since the programming of a true flash FPGA is simpler, requiring only one stage, it makes sense that programming with a microprocessor in-system should be simpler than with other SRAM FPGAs. This reduces bill-of-materials costs and printed circuit board (PCB) area, and increases system reliability.

Nonvolatile flash technology also gives the low power flash devices the advantage of a secure, low power, live-at-power-up, and single-chip solution. Low power flash devices are reprogrammable and offer time-to-market benefits at an ASIC-level unit cost. These features enable engineers to create high-density systems using existing ASIC or FPGA design flows and tools.

This document is an introduction to microprocessor programming only. To explain the difference between the options available, user's guides for DirectC and STAPL provide more detail on implementing each style.

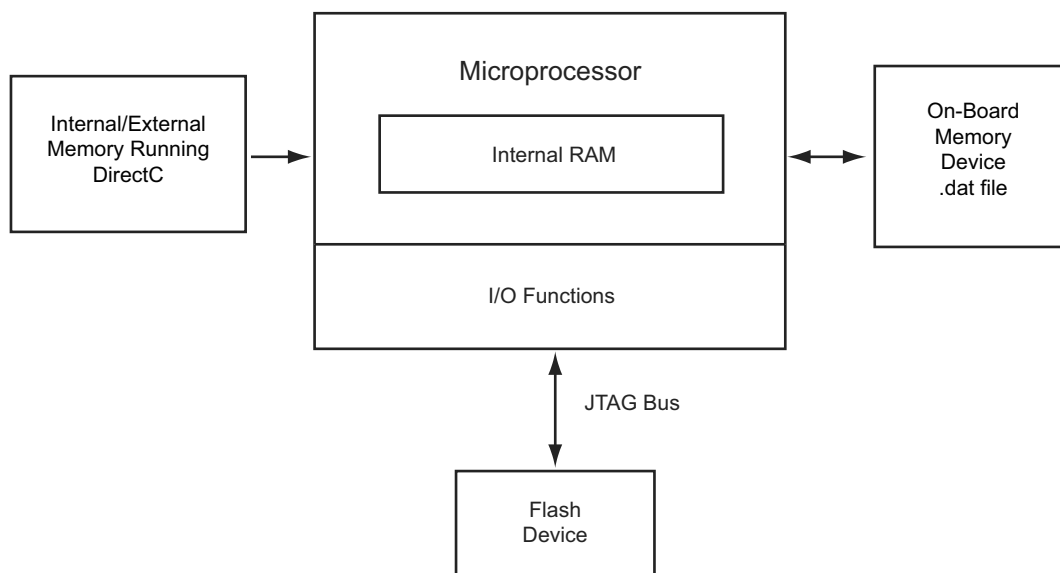


Figure 15-1 • ISP Using Microprocessor

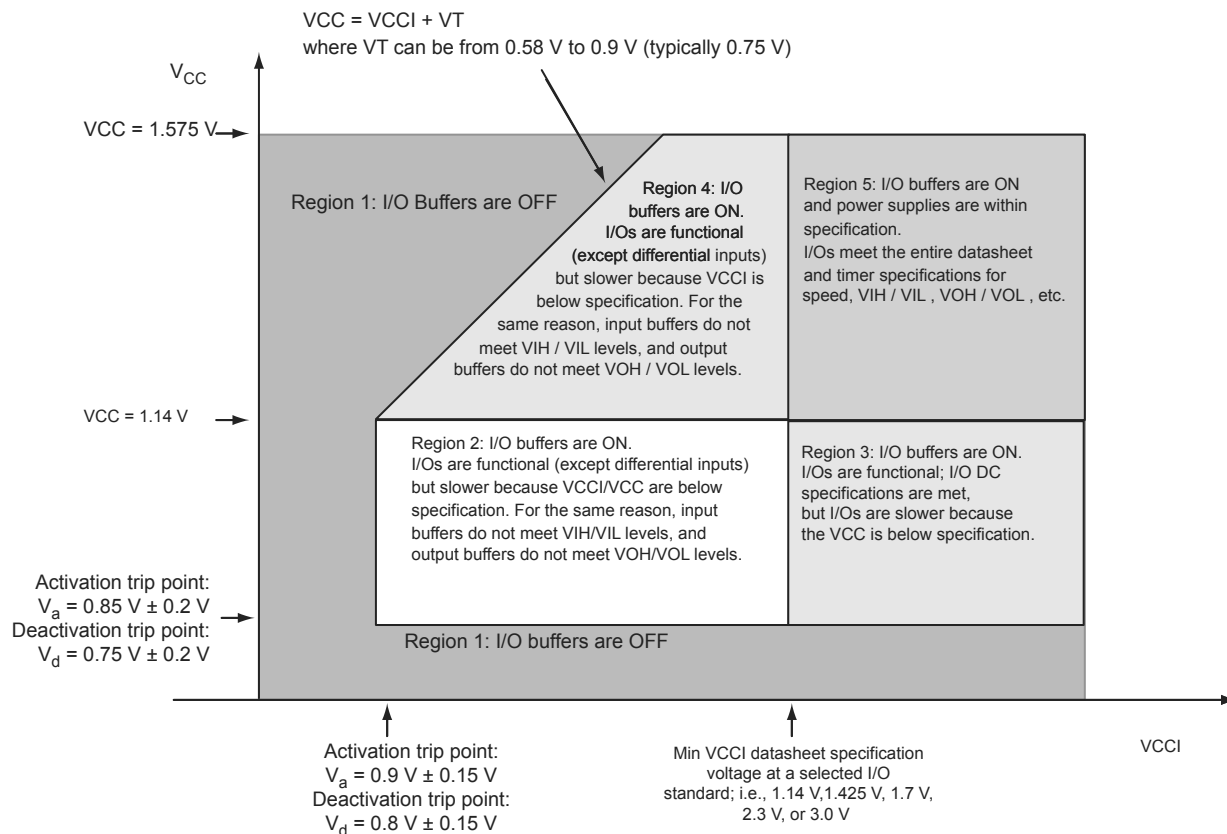


Figure 18-5 • I/O State as a Function of V_{CCI} and V_{CC} Voltage Levels for IGLOO V2, IGLOO nano V2, IGLOO PLUS V2, and ProASIC3L Devices Running at $V_{CC} = 1.2 \text{ V} \pm 0.06 \text{ V}$

Internal Pull-Up and Pull-Down

Low power flash device I/Os are equipped with internal weak pull-up/-down resistors that can be used by designers. If used, these internal pull-up/-down resistors will be activated during power-up, once both VCC and VCCI are above their functional activation level. Similarly, during power-down, these internal pull-up/-down resistors will turn off once the first supply voltage falls below its brownout deactivation level.

Cold-Sparing

In cold-sparing applications, voltage can be applied to device I/Os before and during power-up. Cold-sparing applications rely on three important characteristics of the device:

1. I/Os must be tristated before and during power-up.
2. Voltage applied to the I/Os must not power up any part of the device.
3. VCCI should not exceed 3.6 V, per datasheet specifications.

As described in the "Power-Up to Functional Time" section on page 378, Microsemi's low power flash I/Os are tristated before and during power-up until the last voltage supply (VCC or VCCI) is powered up past its functional level. Furthermore, applying voltage to the FPGA I/Os does not pull up VCC or VCCI and, therefore, does not partially power up the device. Table 18-4 includes the cold-sparing test results on A3PE600-PQ208 devices. In this test, leakage current on the device I/O and residual voltage on the power supply rails were measured while voltage was applied to the I/O before power-up.

Table 18-4 • Cold-Sparing Test Results for A3PE600 Devices

Device I/O	Residual Voltage (V)		Leakage Current
	VCC	VCCI	
Input	0	0.003	<1 μ A
Output	0	0.003	<1 μ A

VCCI must not exceed 3.6 V, as stated in the datasheet specification. Therefore, ProASIC3E devices meet all three requirements stated earlier in this section and are suitable for cold-sparing applications.

The following devices and families support cold-sparing:

- IGLOO: AGL015 and AGL030
- All IGLOO nano
- All IGLOO PLUS
- All IGLOOe
- ProASIC3L: A3PE3000L
- ProASIC3: A3P015 and A3P030
- All ProASIC3 nano
- All ProASIC3E
- Military ProASIC3EL: A3PE600L and A3PE3000L
- RT ProASIC3: RT3PE600L and RT3PE3000L