



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

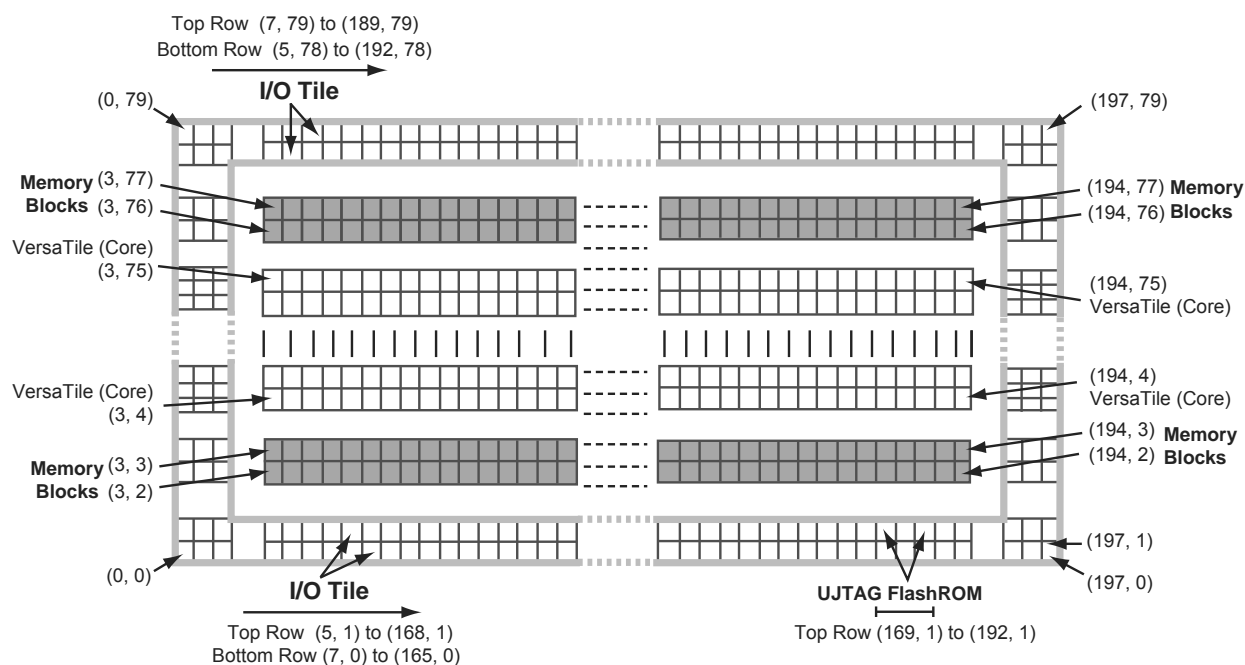
The versatility of Embedded - FPGAs makes them indispensable in numerous fields. In telecommunications.

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/m1a3pe3000l-fg484

Table 1-4 • IGLOO nano and ProASIC3 nano Array Coordinates

Device		VersaTiles		Memory Rows		Entire Die	
		Min.	Max.	Bottom	Top	Min.	Max.
IGLOO nano	ProASIC3 nano	(x, y)	(x, y)	(x, y)	(x, y)	(x, y)	(x, y)
AGLN010	A3P010	(0, 2)	(32, 5)	None	None	(0, 0)	(34, 5)
AGLN015	A3PN015	(0, 2)	(32, 9)	None	None	(0, 0)	(34, 9)
AGLN020	A3PN020	(0, 2)	32, 13)	None	None	(0, 0)	(34, 13)
AGLN060	A3PN060	(3, 2)	(66, 25)	None	(3, 26)	(0, 0)	(69, 29)
AGLN125	A3PN125	(3, 2)	(130, 25)	None	(3, 26)	(0, 0)	(133, 29)
AGLN250	A3PN250	(3, 2)	(130, 49)	None	(3, 50)	(0, 0)	(133, 49)



Note: The vertical I/O tile coordinates are not shown. West-side coordinates are {(0, 2) to (2, 2)} to {(0, 77) to (2, 77)}; east-side coordinates are {(195, 2) to (197, 2)} to {(195, 77) to (197, 77)}.

Figure 1-9 • Array Coordinates for AGL600, AGL600, A3P600, and A3PE600

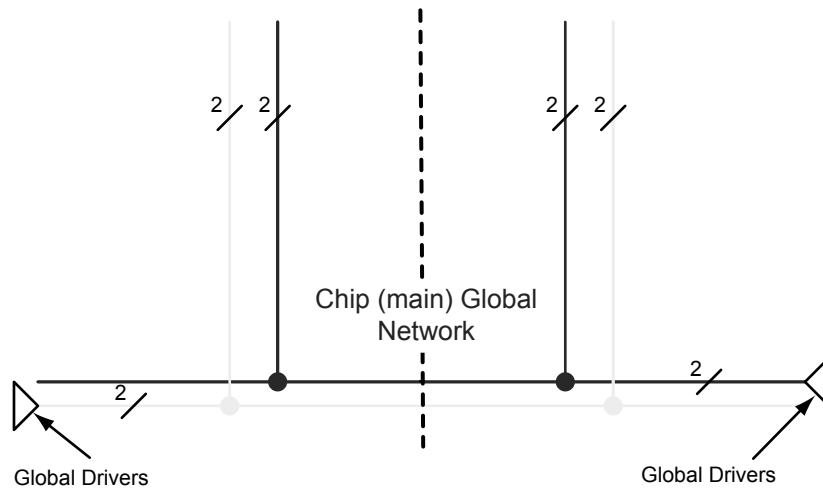


Figure 3-2 • Simplified VersaNet Global Network (30 k gates and below)

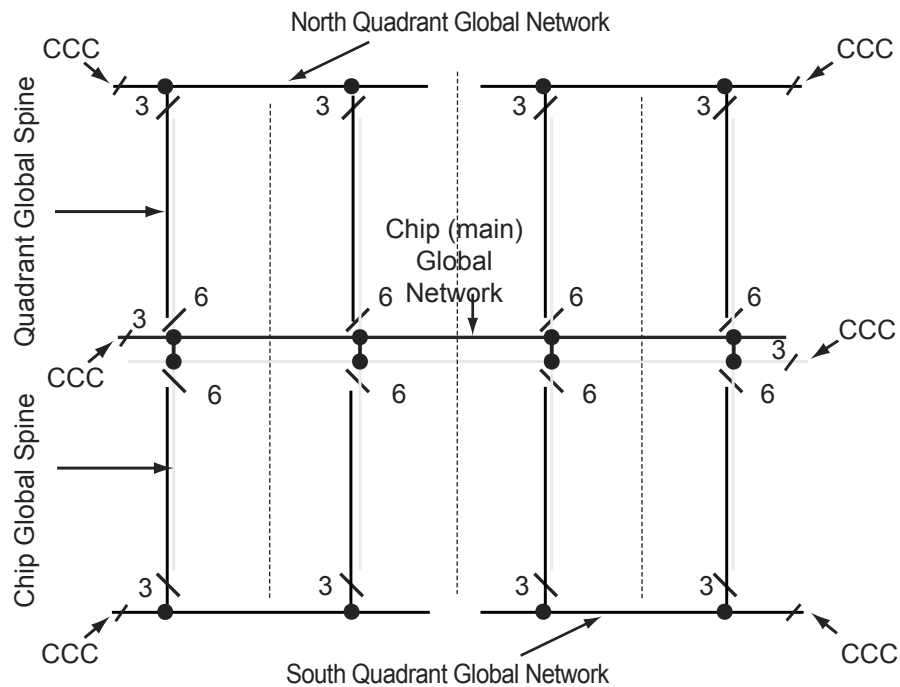


Figure 3-3 • Simplified VersaNet Global Network (60 k gates and above)

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
	The "Global Architecture" section and "VersaNet Global Network Distribution" section were revised for clarity (SARs 20646, 24779).	47, 49
	The "I/O Banks and Global I/Os" section was moved earlier in the document, renamed to "Chip and Quadrant Global I/Os", and revised for clarity. Figure 3-4 • Global Connections Details, Figure 3-6 • Global Inputs, Table 3-2 • Chip Global Pin Name, and Table 3-3 • Quadrant Global Pin Name are new (SARs 20646, 24779).	51
	The "Clock Aggregation Architecture" section was revised (SARs 20646, 24779).	57
	Figure 3-7 • Chip Global Aggregation was revised (SARs 20646, 24779).	59
	The "Global Macro and Placement Selections" section is new (SARs 20646, 24779).	64
v1.4 (December 2008)	The "Global Architecture" section was updated to include 10 k devices, and to include information about VersaNet global support for IGLOO nano devices.	47
	The Table 3-1 • Flash-Based FPGAs was updated to include IGLOO nano and ProASIC3 nano devices.	48
	The "VersaNet Global Network Distribution" section was updated to include 10 k devices and to note an exception in global lines for nano devices.	49
	Figure 3-2 • Simplified VersaNet Global Network (30 k gates and below) is new.	50
	The "Spine Architecture" section was updated to clarify support for 10 k and nano devices.	57
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include IGLOO nano and ProASIC3 nano devices.	57
	The figure in the CLKBUF_LVDS/LVPECL row of Table 3-8 • Clock Macros was updated to change CLKBIBUF to CLKBUF.	62
v1.3 (October 2008)	A third bullet was added to the beginning of the "Global Architecture" section: 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.	47
	The "Global Resource Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	48
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include A3PE600/L in the device column.	57
	Table note 1 was revised in Table 3-9 • I/O Standards within CLKBUF to include AFS600 and AFS1500.	63
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 3-1 • 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. 	48

Feedback Configuration

The PLL provides both internal and external feedback delays. Depending on the configuration, various combinations of feedback delays can be achieved.

Internal Feedback Configuration

This configuration essentially sets the feedback multiplexer to route the VCO output of the PLL core as the input to the feedback of the PLL. The feedback signal can be processed with the fixed system and the adjustable feedback delay, as shown in Figure 4-24. The dividers are automatically configured by SmartGen based on the user input.

Indicated below is the System Delay pull-down menu. The System Delay can be bypassed by setting it to 0. When set, it adds a 2 ns delay to the feedback path (which results in delay advancement of the output clock by 2 ns).

Figure 4-24 • Internal Feedback with Selectable System Delay

Figure 4-25 shows the controllable Feedback Delay. If set properly in conjunction with the fixed System Delay, the total output delay can be advanced significantly.

Figure 4-25 • Internal Feedback with Selectable Feedback Delay

Conclusion

The Fusion, IGLOO, and ProASIC3 families are the only FPGAs that offer on-chip FlashROM support. This document presents information on the FlashROM architecture, possible applications, programming, access through the JTAG and UJTAG interface, and integration into your design. In addition, the Libero tool set enables easy creation and modification of the FlashROM content.

The nonvolatile FlashROM block in the FPGA can be customized, enabling multiple applications.

Additionally, the security offered by the low power flash devices keeps both the contents of FlashROM and the FPGA design safe from system over-builders, system cloners, and IP thieves.

Related Documents

User's Guides

FlashPro User's Guide

http://www.microsemi.com/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
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 5-1 • Flash-Based FPGAs.	134
v1.3 (October 2008)	The "FlashROM Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	134
	Figure 5-2 • Fusion Device Architecture Overview (AFS600) was replaced. Figure 5-5 • Programming FlashROM Using AES was revised to change "Fusion" to "Flash Device."	135, 137
	The <i>FlashPoint User's Guide</i> was removed from the "User's Guides" section, as its content is now part of the <i>FlashPro User's Guide</i> .	146
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 5-1 • 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. 	134
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	N/A

Table 6-2 • Allowable Aspect Ratio Settings for WIDTHA[1:0]

WIDTHA[1:0]	WIDTHB[1:0]	D×W
00	00	4k×1
01	01	2k×2
10	10	1k×4
11	11	512×9

Note: The aspect ratio settings are constant and cannot be changed on the fly.

BLKA and BLKB

These signals are active-low and will enable the respective ports when asserted. When a BLKx signal is deasserted, that port's outputs hold the previous value.

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

WENA and WENB

These signals switch the RAM between read and write modes for the respective ports. A LOW on these signals indicates a write operation, and a HIGH indicates a read.

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

CLKA and CLKB

These are the clock signals for the synchronous read and write operations. These can be driven independently or with the same driver.

Note: For Automotive ProASIC3 devices, dual-port mode is supported if the clocks to the two SRAM ports are the same and 180° out of phase (i.e., the port A clock is the inverse of the port B clock). For use of this macro as a single-port SRAM, the inputs and clock of one port should be tied off (grounded) to prevent errors during design compile.

PIPEA and PIPEB

These signals are used to specify pipelined read on the output. A LOW on PIPEA or PIPEB indicates a nonpipelined read, and the data appears on the corresponding output in the same clock cycle. A HIGH indicates a pipelined read, and data appears on the corresponding output in the next clock cycle.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, PIPEB should be tied to ground. For use in dual-port mode, the same clock with an inversion between the two clock pins of the macro should be used in the design to prevent errors during compile.

WMODEA and WMODEB

These signals are used to configure the behavior of the output when the RAM is in write mode. A LOW on these signals makes the output retain data from the previous read. A HIGH indicates pass-through behavior, wherein the data being written will appear immediately on the output. This signal is overridden when the RAM is being read.

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

RESET

This active-low signal resets the control logic, forces the output hold state registers to zero, disables reads and writes from the SRAM block, and clears the data hold registers when asserted. It does not reset the contents of the memory array.

While the RESET signal is active, read and write operations are disabled. As with any asynchronous reset signal, care must be taken not to assert it too close to the edges of active read and write clocks.

ADDRA and ADDRb

These are used as read or write addresses, and they are 12 bits wide. When a depth of less than 4 k is specified, the unused high-order bits must be grounded (Table 6-3 on page 155).

Software Support

The SmartGen core generator is the easiest way to select and configure the memory blocks (Figure 6-12). SmartGen automatically selects the proper memory block type and aspect ratio, and cascades the memory blocks based on the user's selection. SmartGen also configures any additional signals that may require tie-off.

SmartGen will attempt to use the minimum number of blocks required to implement the desired memory. When cascading, SmartGen will configure the memory for width before configuring for depth. For example, if the user requests a 256×8 FIFO, SmartGen will use a 512×9 FIFO configuration, not 256×18.

Figure 6-12 • SmartGen Core Generator Interface

I/O Features

Low power flash devices support multiple I/O features that make board design easier. For example, an I/O feature like Schmitt Trigger in the ProASIC3E input buffer saves the board space that would be used by an external Schmitt trigger for a slow or noisy input signal. These features are also programmable for each I/O, which in turn gives flexibility in interfacing with other components. The following is a detailed description of all available features in low power flash devices.

I/O Programmable Features

Low power flash devices offer many flexible I/O features to support a wide variety of board designs. Some of the features are programmable, with a range for selection. Table 7-7 lists programmable I/O features and their ranges.

Table 7-7 • Programmable I/O Features (user control via I/O Attribute Editor)

Feature ¹	Description	Range
Slew Control	Output slew rate	HIGH, LOW
Output Drive (mA)	Output drive strength	2, 4, 6, 8, 12, 16, 24
Skew Control	Output tristate enable delay option	ON, OFF
Resistor Pull	Resistor pull circuit	Up, Down, None
Input Delay ²	Input delay	OFF, 0–7
Schmitt Trigger	Schmitt trigger for input only	ON, OFF

Notes:

1. Limitations of these features with respect to different devices are discussed in later sections.
2. Programmable input delay is applicable only to ProASIC3EL and RT ProASIC3 devices.

Hot-Swap Support

A pull-up clamp diode must not be present in the I/O circuitry if the hot-swap feature is used. The 3.3 V PCI standard requires a pull-up clamp diode on the I/O, so it cannot be selected if hot-swap capability is required. The A3P030 device does not support 3.3 V PCI, so it is the only device in the ProASIC3 family that supports the hot-swap feature. All devices in the ProASIC3E family are hot-swappable. All standards except LVCMOS 2.5/5.0 V and 3.3 V PCI/PCI-X support the hot-swap feature.

The hot-swap feature appears as a read-only check box in the I/O Attribute Editor that shows whether an I/O is hot-swappable or not. Refer to the *"Power-Up/Down Behavior of Low Power Flash Devices"* section on page 373 for details on hot-swapping.

Hot-swapping (also called hot-plugging) is the operation of hot insertion or hot removal of a card in a powered-up system. The levels of hot-swap support and examples of related applications are described in Table 7-8 on page 189 to Table 7-11 on page 190. The I/Os also need to be configured in hot-insertion mode if hot-plugging compliance is required. The AGL030 and A3P030 devices have an I/O structure that allows the support of Level 3 and Level 4 hot-swap with only two levels of staging.

Table 9-3 • PDC I/O Constraints (continued)

Command	Action	Example	Comment
I/O Attribute Constraint			
set_io	Sets the attributes of an I/O	<pre>set_io portname [-pinname value] [-fixed value] [-iostd value] [-out_drive value] [-slew value] [-res_pull value] [-schmitt_trigger value] [-in_delay value] [-skew value] [-out_load value] [-register value] set_io IN2 -pinname 28 -fixed yes -iostd LVCMOS15 -out_drive 12 -slew high -RES_PULL None -SCHMITT_TRIGGER Off -IN_DELAY Off -skew off -REGISTER No</pre>	<p>If the I/O macro is generic (e.g., INBUF) or technology-specific (INBUF_LVCMOS25), then all I/O attributes can be assigned using this constraint.</p> <p>If the netlist has an I/O macro that specifies one of its attributes, that attribute cannot be changed using this constraint, though other attributes can be changed.</p> <p>Example: OUTBUF_S_24 (low slew, output drive 24 mA)</p> <p>Slew and output drive cannot be changed.</p>
I/O Region Placement Constraints			
define_region	Defines either a rectangular region or a rectilinear region	<pre>define_region -name [region_name] -type [region_type] x1 y1 x2 y2 define_region -name test -type inclusive 0 15 2 29</pre>	If any number of I/Os must be assigned to a particular I/O region, such a region can be created with this constraint.
assign_region	Assigns a set of macros to a specified region	<pre>assign_region [region name] [macro_name...] assign_region test U12</pre>	This constraint assigns I/O macros to the I/O regions. When assigning an I/O macro, PDC naming conventions must be followed if the macro name contains special characters; e.g., if the macro name is \\\$1I19\\, the correct use of escape characters is \\\$1I19\\.

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

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**.

4. Right-click and then choose **Highlight VREF range**. All the pins covered by that VREF pin will be highlighted (Figure 9-14).
-

Figure 9-14 • VREF Range

Using PinEditor or ChipPlanner, VREF pins can also be assigned (Figure 9-15).

Figure 9-15 • Assigning VREF from PinEditor

To unassign a VREF pin:

1. Select the pin to unassign.
2. Right-click and choose **Use Pin for VREF**. The check mark next to the command disappears. The VREF pin is now a regular pin.

Resetting the pin may result in unassigning I/O cores, even if they are locked. In this case, a warning message appears so you can cancel the operation.

After you assign the VREF pins, right-click a VREF pin and choose **Highlight VREF Range** to see how many I/Os are covered by that pin. To unhighlight the range, choose **Unhighlight All** from the **Edit** menu.

DDR Support in Flash-Based Devices

The flash FPGAs listed in Table 10-1 support the DDR feature and the functions described in this document.

Table 10-1 • 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
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 10-1. 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 10-1. 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*.

DDR Output Register

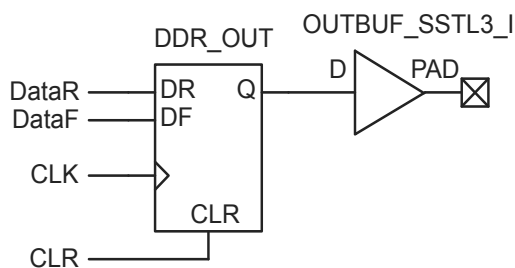


Figure 10-6 • DDR Output Register (SSTL3 Class I)

Verilog

```
module DDR_OutBuf_SSTL3_I(DataR,DataF,CLR,CLK,PAD);

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

wire Q, VCC;

    VCC VCC_1_net(.Y(VCC));
    DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
    OUTBUF_SSTL3_I OUTBUF_SSTL3_I_0_inst(.D(Q),.PAD(PAD));

endmodule
```

VHDL

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

entity DDR_OutBuf_SSTL3_I is
    port(DataR, DataF, CLR, CLK : in std_logic;  PAD : out std_logic) ;
end DDR_OutBuf_SSTL3_I;

architecture DEF_ARCH of  DDR_OutBuf_SSTL3_I is

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

    component OUTBUF_SSTL3_I
        port(D : in std_logic := 'U'; PAD : out std_logic) ;
    end component;

    component VCC
        port( Y : out std_logic);
    end component;

    signal Q, VCC_1_net : std_logic ;

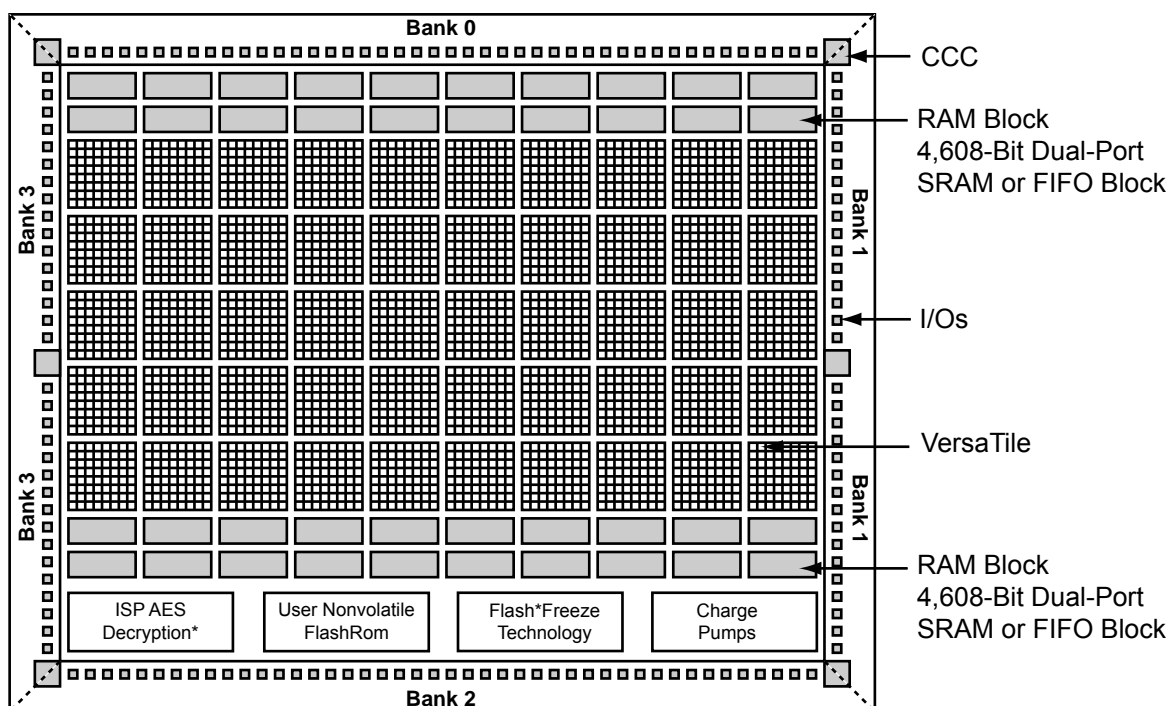
begin

    VCC_2_net : VCC port map(Y => VCC_1_net);
    DDR_OUT_0_inst : DDR_OUT
        port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
    OUTBUF_SSTL3_I_0_inst : OUTBUF_SSTL3_I
        port map(D => Q, PAD => PAD);

end DEF_ARCH;
```

Security Architecture

Fusion, IGLOO, and ProASIC3 devices have been designed with the most comprehensive programming logic design security in the industry. In the architecture of these devices, security has been designed into the very fabric. The flash cells are located beneath seven metal layers, and the use of many device design and layout techniques makes invasive attacks difficult. Since device layers cannot be removed without disturbing the charge on the programmed (or erased) flash gates, devices cannot be easily deconstructed to decode the design. Low power flash devices are unique in being reprogrammable and having inherent resistance to both invasive and noninvasive attacks on valuable IP. Secure, remote ISP is now possible with AES encryption capability for the programming file during electronic transfer. Figure 12-2 shows a view of the AES decryption core inside an IGLOO device; Figure 12-3 on page 304 shows the AES decryption core inside a Fusion device. The AES core is used to decrypt the encrypted programming file when programming.



Note: *ISP AES Decryption is not supported by 30 k gate devices and smaller. For details of other architecture features by device, refer to the appropriate family datasheet.

Figure 12-2 • Block Representation of the AES Decryption Core in IGLOO and ProASIC3 Devices

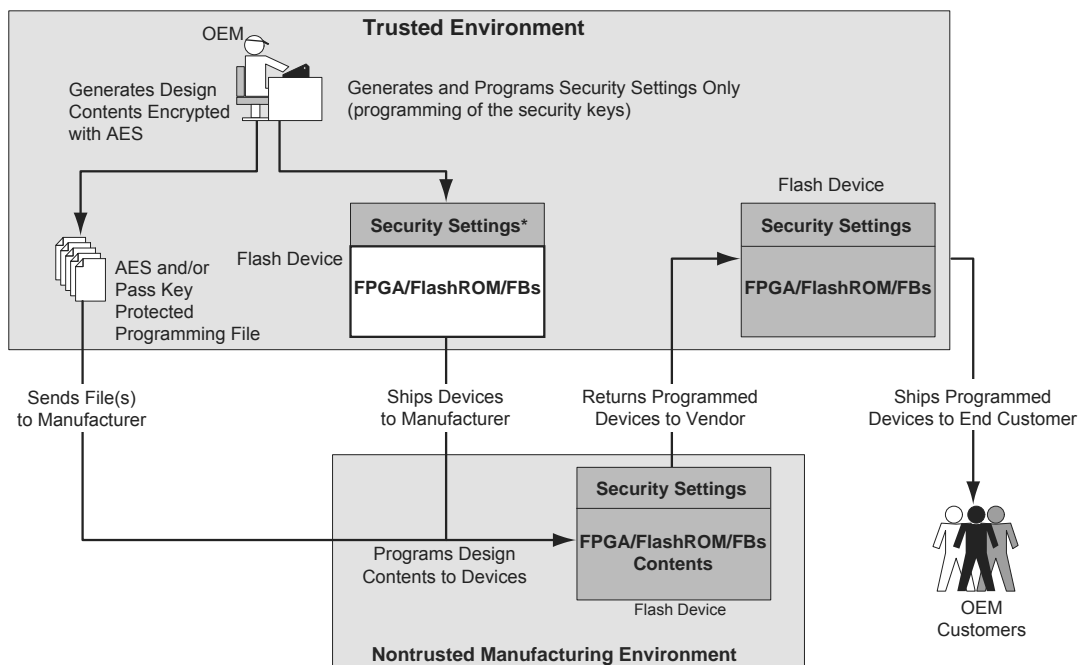
Application 1: Trusted Environment

As illustrated in Figure 12-7, this application allows the programming of devices at design locations where research and development take place. Therefore, encryption is not necessary and is optional to the user. This is often a secure way to protect the design, since the design program files are not sent elsewhere. In situations where production programming is not available at the design location, programming centers (such as Microsemi In-House Programming) provide a way of programming designs at an alternative, secure, and trusted location. In this scenario, the user generates a STAPL programming file from the Designer software in plaintext format, containing information on the entire design or the portion of the design to be programmed. The user can choose to employ the FlashLock Pass Key feature with the design. Once the design is programmed to unprogrammed devices, the design is protected by this FlashLock Pass Key. If no future programming is needed, the user can consider permanently securing the IGLOO and ProASIC3 device, as discussed in the "Permanent FlashLock" section on page 307.

Application 2: Nontrusted Environment—Unsecured Location

Often, programming of devices is not performed in the same location as actual design implementation, to reduce manufacturing cost. Overseas programming centers and contract manufacturers are examples of this scenario.

To achieve security in this case, the AES key and the FlashLock Pass Key can be initially programmed in-house (trusted environment). This is done by generating a programming file with only the security settings and no design contents. The design FPGA core, FlashROM, and (for Fusion) FB contents are generated in a separate programming file. This programming file must be set with the same AES key that was used to program to the device previously so the device will correctly decrypt this encrypted programming file. As a result, the encrypted design content programming file can be safely sent off-site to nontrusted programming locations for design programming. Figure 12-7 shows a more detailed flow for this application.



Notes:

1. Programmed portion indicated with dark gray.
2. Programming of FBs applies to Fusion only.

Figure 12-7 • Application 2: Device Programming in a Nontrusted Environment

Generating Programming Files

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

As discussed in the "Application 1: Trusted Environment" section on page 309, in a trusted environment, the user can choose to program the device with plaintext bitstream content. It is possible to use plaintext for programming even when the FlashLock Pass Key option has been selected. In this application, it is not necessary to employ AES encryption protection. For AES encryption settings, refer to the next sections.

The generated programming file will include the security setting (if selected) and the plaintext programming file content for the FPGA array, FlashROM, and/or FBs. These options are indicated in Table 12-2 and Table 12-3.

Table 12-2 • IGLOO and ProASIC3 Plaintext Security Options, No AES

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

Table 12-3 • Fusion Plaintext Security Options

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

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

For this scenario, generate the programming file as follows:

1. Select the **Silicon features to be programmed** (Security Settings, FPGA Array, FlashROM, Flash Memory Blocks), as shown in Figure 12-10 on page 314 and Figure 12-11 on page 314. Click **Next**.

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

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

Introduction

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

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

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

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

Programming Algorithm

JTAG Interface

The low power flash families are fully compliant with the IEEE 1149.1 (JTAG) standard. They support all the mandatory boundary scan instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS) as well as six optional public instructions (USERCODE, IDCODE, HIGHZ, and CLAMP).

IEEE 1532

The low power flash families are also fully compliant with the IEEE 1532 programming standard. The IEEE 1532 standard adds programming instructions and associated data registers to devices that comply with the IEEE 1149.1 standard (JTAG). These instructions and registers extend the capabilities of the IEEE 1149.1 standard such that the Test Access Port (TAP) can be used for configuration activities. The IEEE 1532 standard greatly simplifies the programming algorithm, reducing the amount of time needed to implement microprocessor ISP.

Implementation Overview

To implement device programming with a microprocessor, the user should first download the C-based STAPL player or DirectC code from the Microsemi SoC Products Group website. Refer to the website for future updates regarding the STAPL player and DirectC code.

http://www.microsemi.com/soc/download/program_debug/stapl/default.aspx

http://www.microsemi.com/soc/download/program_debug/directc/default.aspx

Using the easy-to-follow user's guide, create the low-level application programming interface (API) to provide the necessary basic functions. These API functions act as the interface between the programming software and the actual hardware (Figure 15-2).

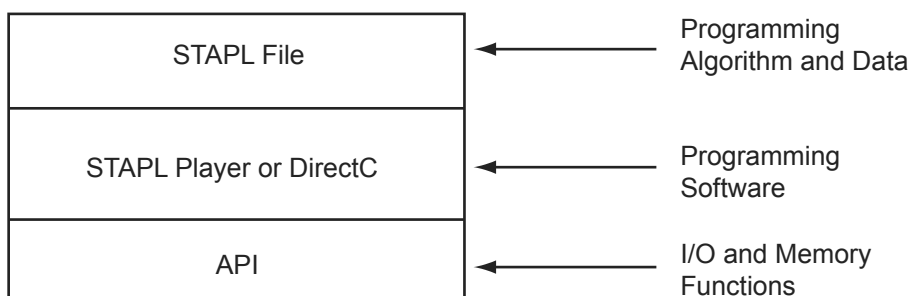


Figure 15-2 • Device Programming Code Relationship

The API is then linked with the STAPL player or DirectC and compiled using the microprocessor's compiler. Once the entire code is compiled, the user must download the resulting binary into the MCU system's program memory (such as ROM, EEPROM, or flash). The system is now ready for programming.

To program a design into the FPGA, the user creates a bitstream or STAPL file using the Microsemi Designer software, downloads it into the MCU system's volatile memory, and activates the stored programming binary file (Figure 15-3 on page 352). Once the programming is completed, the bitstream or STAPL file can be removed from the system, as the configuration profile is stored in the flash FPGA fabric and does not need to be reloaded at every system power-on.

STAPL vs. DirectC

Programming the low power flash devices is performed using DirectC or the STAPL player. Both tools use the STAPL file as an input. DirectC is a compiled language, whereas STAPL is an interpreted language. Microprocessors will be able to load the FPGA using DirectC much more quickly than STAPL. This speed advantage becomes more apparent when lower clock speeds of 8- or 16-bit microprocessors are used. DirectC also requires less memory than STAPL, since the programming algorithm is directly implemented. STAPL does have one advantage over DirectC—the ability to upgrade. When a new programming algorithm is required, the STAPL user simply needs to regenerate a STAPL file using the latest version of the Designer software and download it to the system. The DirectC user must download the latest version of DirectC from Microsemi, compile everything, and download the result into the system (Figure 15-4).

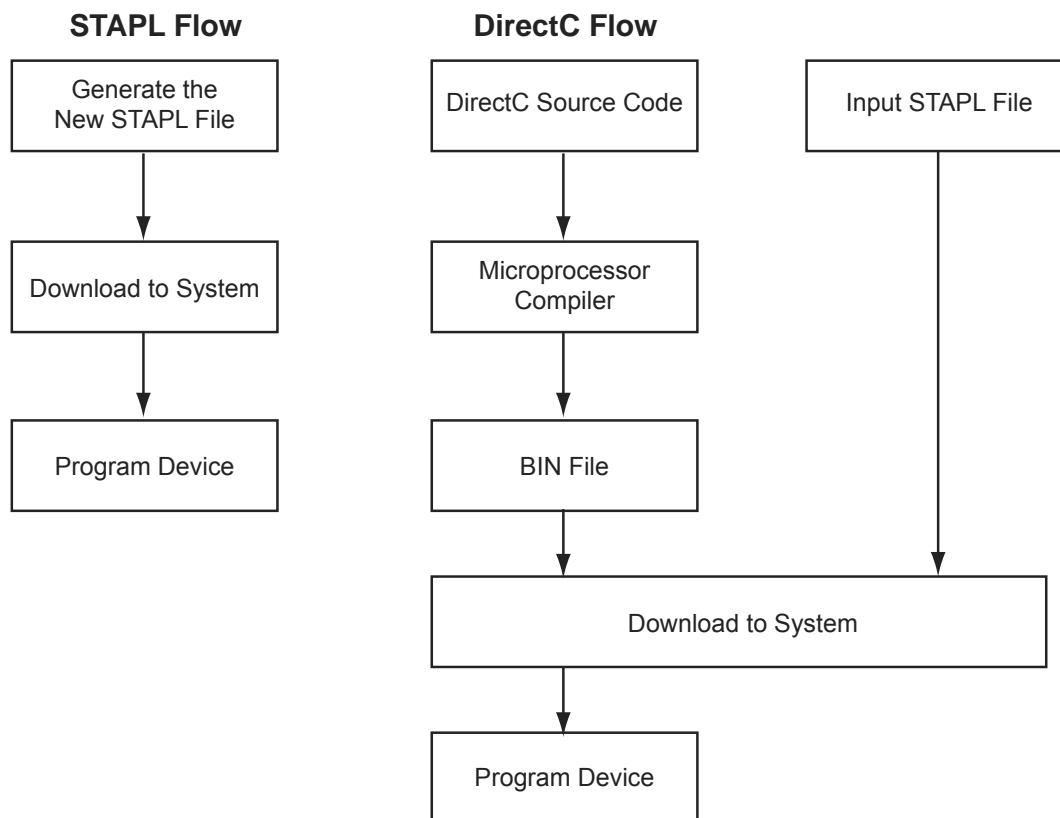


Figure 15-4 • STAPL vs. DirectC

Brownout Voltage

Brownout is a condition in which the voltage supplies are lower than normal, causing the device to malfunction as a result of insufficient power. In general, Microsemi does not guarantee the functionality of the design inside the flash FPGA if voltage supplies are below their minimum recommended operating condition. Microsemi has performed measurements to characterize the brownout levels of FPGA power supplies. Refer to Table 18-3 for device-specific brownout deactivation levels. For the purpose of characterization, a direct path from the device input to output is monitored while voltage supplies are lowered gradually. The brownout point is defined as the voltage level at which the output stops following the input. Characterization tests performed on several IGLOO, ProASIC3L, and ProASIC3 devices in typical operating conditions showed the brownout voltage levels to be within the specification.

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

Table 18-3 • Brownout Deactivation Levels for VCC and VCCI

Devices	VCC Brownout Deactivation Level (V)	VCCI Brownout Deactivation Level (V)
ProASIC3, ProASIC3 nano, IGLOO, IGLOO nano, IGLOO PLUS and ProASIC3L devices running at VCC = 1.5 V	0.75 V \pm 0.25 V	0.8 V \pm 0.3 V
IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices running at VCC = 1.2 V	0.75 V \pm 0.2 V	0.8 V \pm 0.15 V

PLL Behavior at Brownout Condition

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

VCCPLL and VCC Tied Together

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

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

Only VCCPLL Is at Brownout

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

Only VCC Is at Brownout

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

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

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