



Welcome to E-XFL.COM

Understanding <u>Embedded - FPGAs (Field</u> <u>Programmable Gate Array)</u>

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	<u>-</u>
Total RAM Bits	147456
Number of I/O	97
Number of Gates	1000000
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/microchip-technology/m1a3p1000l-1fg144

Email: info@E-XFL.COM

Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

Core Architecture

VersaTile

The proprietary IGLOO and ProASIC3 device architectures provide granularity comparable to gate arrays. The device core consists of a sea-of-VersaTiles architecture.

As illustrated in Figure 1-8, there are four inputs in a logic VersaTile cell, and each VersaTile can be configured using the appropriate flash switch connections:

- Any 3-input logic function
- Latch with clear or set
- D-flip-flop with clear or set
- Enable D-flip-flop with clear or set (on a 4th input)

VersaTiles can flexibly map the logic and sequential gates of a design. The inputs of the VersaTile can be inverted (allowing bubble pushing), and the output of the tile can connect to high-speed, very-long-line routing resources. VersaTiles and larger functions can be connected with any of the four levels of routing hierarchy.

When the VersaTile is used as an enable D-flip-flop, SET/CLR is supported by a fourth input. The SET/CLR signal can only be routed to this fourth input over the VersaNet (global) network. However, if, in the user's design, the SET/CLR signal is not routed over the VersaNet network, a compile warning message will be given, and the intended logic function will be implemented by two VersaTiles instead of one.

The output of the VersaTile is F2 when the connection is to the ultra-fast local lines, or YL when the connection is to the efficient long-line or very-long-line resources.



Figure 1-8 • Low Power Flash Device Core VersaTile

During Flash*Freeze Mode

- PLLs are turned off during Flash*Freeze mode.
- I/O pads are configured according to Table 2-5 on page 28 and Table 2-6 on page 29.
- Inputs and input clocks to the FPGA can toggle without any impact on static power consumption, assuming weak pull-up or pull-down is not selected.
- If weak pull-up or pull-down is selected and the input is driven to the opposite direction, power dissipation will occur.
- Any toggling signals will be charging and discharging the package pin capacitance.
- IGLOO and ProASIC3L outputs will be tristated unless the I/O is configured with weak pull-up or pull-down. The output of the I/O to the FPGA core is logic High regardless of whether the I/O pin is configured with a weak pull-up or pull-down. Refer to Table 2-5 on page 28 for more information.
- IGLOO nano and IGLOO PLUS output behavior will be based on the configuration defined by the user. Refer to Table 2-6 on page 29 for a description of output behavior during Flash*Freeze mode.
- The JTAG circuit is active; however, JTAG operations, such as JTAG commands, JTAG bypass, programming, and authentication, cannot be executed. The device must exit Flash*Freeze mode before JTAG commands can be sent. TCK should be static to avoid extra power consumption from the JTAG state machine.
- The FF pin must be externally asserted for the device to stay in Flash*Freeze mode.
- The FF pin is still active; i.e., the pin is used to exit Flash*Freeze mode when deasserted.

Exiting Flash*Freeze Mode

I/Os and Globals

- While exiting Flash*Freeze mode, inputs and globals will exit their Flash*Freeze state asynchronously to each other. As a result, clock and data glitches and narrow pulses may be generated while exiting Flash*Freeze mode, unless clock gating schemes are used.
- I/O banks are not all activated simultaneously when exiting Flash*Freeze mode. This can cause clocks and inputs to become enabled at different times, resulting in unexpected data being captured.
- Upon exiting Flash*Freeze mode, inputs and globals will no longer be tied High internally (does not apply to input hold state on IGLOO nano and IGLOO PLUS). If any of these signals are driven Low or tied Low externally, they will experience a High-to-Low transition internally when exiting Flash*Freeze mode.
- Applies only to IGLOO nano and IGLOO PLUS: Output hold state is asynchronously controlled by the signal driving the output buffer (output signal). This ensures a clean, glitch-free transition from hold state to output drive. However, any glitches on the output signal during exit from Flash*Freeze mode may result in glitches on the output pad.
- The above situations can cause glitches or invalid data to be clocked into and preserved in the device. Refer to the "Flash*Freeze Design Guide" on page 34 for solutions.

PLLs

• If the embedded PLL is used, the design must allow maximum acquisition time (per device datasheet) for the PLL to acquire the lock signal.

Flash*Freeze Pin Locations

Refer to the Pin Descriptions and Packaging chapter of specific device datasheets for information regarding Flash*Freeze pin location on the available packages. The Flash*Freeze pin location is independent of the device, allowing migration to larger or smaller devices while maintaining the same pin location on the board.

standard for CLKBUF is LVTTL in the current Microsemi Libero $^{\ensuremath{\mathbb{R}}}$ System-on-Chip (SoC) and Designer software.

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			

Table 3-9 • I/O Standards within CLKBUF

Notes:

- 1. Supported in only the IGLOOe, ProASIC3E, AFS600, and AFS1500 devices
- 2. By default, the CLKBUF macro uses the 3.3 V LVTTL 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

Figure 4-34 • Cascade PLL Configuration

Using internal feedback, we know from EQ 4-1 on page 102 that the maximum achievable output frequency from the primary output is

 $f_{GLA} = f_{CLKA} \times m / (n \times u) = 2 MHz \times 128 / (1 \times 1) = 256 MHz$

EQ 4-5

Figure 4-35 shows the settings of the initial PLL. When configuring the initial PLL, specify the input to be either Hardwired I/O–Driven or External I/O–Driven. This generates a netlist with the initial PLL routed from an I/O. Do not specify the input to be Core Logic–Driven, as this prohibits the connection from the I/O pin to the input of the PLL.



Figure 4-35 • First-Stage PLL Showing Input of 2 MHz and Output of 256 MHz

A second PLL can be connected serially to achieve the required frequency. EQ 4-1 on page 102 to EQ 4-3 on page 102 are extended as follows:

 $f_{GLA2} = f_{GLA} \times m_2 / (n_2 \times u_2) = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times u_1 \times n_2 \times u_2) - Primary PLL Output Clock$

EQ 4-6

$$f_{GLB2} = f_{YB2} = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times n_2 \times v_1 \times v_2) - \text{Secondary 1 PLL Output Clock(s)}$$

EQ 4-7

$$f_{GLC2} = f_{YC2} = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times n_2 \times w_1 \times w_2) - \text{Secondary 2 PLL Output Clock(s)}$$

EQ 4-8

In the example, the final output frequency (f_{output}) from the primary output of the second PLL will be as follows (EQ 4-9):

$$f_{output} = f_{GLA2} = f_{GLA} \times m_2 / (n_2 \times u_2) = 256 \text{ MHz} \times 70 / (64 \times 1) = 280 \text{ MHz}$$

EQ 4-9

Figure 4-36 on page 127 shows the settings of the second PLL. When configuring the second PLL (or any subsequent-stage PLLs), specify the input to be Core Logic–Driven. This generates a netlist with the second PLL routed internally from the core. Do not specify the input to be Hardwired I/O–Driven or External I/O–Driven, as these options prohibit the connection from the output of the first PLL to the input of the second PLL.

ProASIC3L FPGA Fabric User's Guide



Figure 5-2 • Fusion Device Architecture Overview (AFS600)



Figure 5-3 • ProASIC3 and IGLOO Device Architecture

Conclusion

Fusion, IGLOO, and ProASIC3 devices provide users with extremely flexible SRAM blocks for most design needs, with the ability to choose between an easy-to-use dual-port memory or a wide-word two-port memory. Used with the built-in FIFO controllers, these memory blocks also serve as highly efficient FIFOs that do not consume user gates when implemented. The SmartGen core generator provides a fast and easy way to configure these memory elements for use in designs.

List of Changes

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

Date	Changes	Page
August 2012	The note connected with Figure 6-3 • Supported Basic RAM Macros, regarding RAM4K9, was revised to explain that it applies only to part numbers of certain revisions and earlier (SAR 29574).	152
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 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 6-1 • Flash-Based per 2008) FPGAs.	
	IGLOO nano and ProASIC3 nano devices were added to Figure 6-8 • Interfacing TAP Ports and SRAM Blocks.	164
v1.4 (October 2008)	The "SRAM/FIFO Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	150
	The "SRAM and FIFO Architecture" section was modified to remove "IGLOO and ProASIC3E" from the description of what the memory block includes, as this statement applies to all memory blocks.	151
	Wording in the "Clocking" section was revised to change "IGLOO and ProASIC3 devices support inversion" to "Low power flash devices support inversion." The reference to IGLOO and ProASIC3 development tools in the last paragraph of the section was changed to refer to development tools in general.	157
	The "ESTOP and FSTOP Usage" section was updated to refer to FIFO counters in devices in general rather than only IGLOO and ProASIC3E devices.	160
v1.3 (August 2008)	The note was removed from Figure 6-7 • RAM Block with Embedded FIFO Controller and placed in the WCLK and RCLK description.	158
	The "WCLK and RCLK" description was revised.	159
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 6-1 • Flash- Based FPGAs:	150
	ProASIC3L was updated to include 1.5 V.	
	The number of PLLs for ProASIC3E was changed from five to six.	
v1.1 (March 2008)	The "Introduction" section was updated to include the IGLOO PLUS family.	147
	The "Device Architecture" section was updated to state that 15 k gate devices do not support SRAM and FIFO.	147
	The first note in Figure 6-1 • IGLOO and ProASIC3 Device Architecture Overview was updated to include mention of 15 k gate devices, and IGLOO PLUS was added to the second note.	149

I/O Structures in IGLOO and ProASIC3 Devices

Low Power Flash Device I/O Support

The low power flash FPGAs listed in Table 7-1 support I/Os and the functions described in this document.

Table 7-1 • Flash-Based FPGAs

Series	Family [*]	Description		
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology		
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs		
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L		
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L		

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 7-1. 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 7-1. 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*.

I/O Structures in IGLOO and ProASIC3 Devices

Simultaneously Switching Outputs (SSOs) and Printed Circuit Board Layout

Each I/O voltage bank has a separate ground and power plane for input and output circuits (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.

Since voltage bounce originates on the package inductance, the VMV and VCCI supplies have separate package pin assignments. For the same reason, GND and GNDQ also have separate pin assignments.

The VMV and VCCI pins must be shorted to each other on the board. Also, the GND and GNDQ pins must be shorted to each other on the board. This will prevent unwanted current draw from the power supply.

SSOs can cause signal integrity problems on adjacent signals that are not part of the SSO bus. Both inductive and capacitive coupling parasitics of bond wires inside packages and of traces on PCBs will transfer noise from SSO busses onto signals adjacent to those busses. Additionally, SSOs can produce ground bounce noise and VCCI dip noise. These two noise types are caused by rapidly changing currents through GND and VCCI package pin inductances during switching activities (EQ 2 and EQ 3).

Ground bounce noise voltage = L(GND) × di/dt

VCCI dip noise voltage = L(VCCI) × di/dt

EQ 3

EQ 2

Any group of four or more input pins switching on the same clock edge is considered an SSO bus. The shielding should be done both on the board and inside the package unless otherwise described.

In-package shielding can be achieved in several ways; the required shielding will vary depending on whether pins next to the SSO bus are LVTTL/LVCMOS inputs, LVTTL/LVCMOS outputs, or GTL/SSTL/HSTL/LVDS/LVPECL inputs and outputs. Board traces in the vicinity of the SSO bus have to be adequately shielded from mutual coupling and inductive noise that can be generated by the SSO bus. Also, noise generated by the SSO bus needs to be reduced inside the package.

PCBs perform an important function in feeding stable supply voltages to the IC and, at the same time, maintaining signal integrity between devices.

Key issues that need to be considered are as follows:

- · Power and ground plane design and decoupling network design
- Transmission line reflections and terminations

For extensive data per package on the SSO and PCB issues, refer to the "ProASIC3/E SSO and Pin Placement and Guidelines" chapter of the *ProASIC3 FPGA Fabric User's Guide*.

ProASIC3L FPGA Fabric User's Guide







Figure 7-20 • Naming Conventions of IGLOO and ProASIC3 Devices with Four I/O Banks – Top View

Revision 4

I/O Structures in IGLOOe and ProASIC3E Devices

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 V_{CCI} /GND for output buffers). Because of these dedicated supplies, only I/Os with compatible standards can be assigned to the same I/O voltage bank. Table 8-3 on page 217 shows the required voltage compatibility values for each of these voltages.

There are eight I/O banks (two per side).

Every I/O bank is divided into minibanks. Any user I/O in a VREF minibank (a minibank is the region of scope of a VREF pin) can be configured as a VREF pin (Figure 8-2). Only one V_{REF} pin is needed to control the entire V_{REF} minibank. The location and scope of the V_{REF} minibanks can be determined by the I/O name. For details, see the user I/O naming conventions for "IGLOOe and ProASIC3E" on page 245. Table 8-5 on page 217 shows the I/O standards supported by IGLOOe and ProASIC3E devices, and the corresponding voltage levels.

I/O standards are compatible if they comply with the following:

- Their VCCI and VMV values are identical.
- Both of the standards need a VREF, and their VREF values are identical.
- All inputs and disabled outputs are voltage tolerant up to 3.3 V.

For more information about I/O and global assignments to I/O banks in a device, refer to the specific pin table for the device in the packaging section of the datasheet, and see the user I/O naming conventions for "IGLOOe and ProASIC3E" on page 245.



Figure 8-2 • Typical IGLOOe and ProASIC3E I/O Bank Detail Showing VREF Minibanks

I/O Structures in IGLOOe and ProASIC3E Devices

Table 8-18 • Supported IGLOOe, ProASIC3L, and ProASIC3E I/O Standards and Corresponding VREF and VTT Voltages

I/O Standard	Input/Output Supply Voltage (VMV _{TYP} /V _{CCI_TYP})	Input Reference Voltage (V _{REF_TYP})	Board Termination Voltage (V _{TT_TYP})
LVTTL/ L VCMOS 3.3 V	3.30 V	-	-
LVCMOS 2.5 V	2.50 V	-	_
LVCMOS 2.5/5.0 V Input	2.50 V	-	-
LVCMOS 1.8 V	1.80 V	-	-
LVCMOS 1.5 V	1.50 V	-	_
PCI 3.3 V	3.30 V	-	-
PCI-X 3.3 V	3.30 V	-	-
GTL+ 3.3 V	3.30 V	1.00 V	1.50 V
GTL+ 2.5 V	2.50 V	1.00 V	1.50 V
GTL 3.3 V	3.30 V	0.80 V	1.20 V
GTL 2.5 V	2.50 V	0.80 V	1.20 V
HSTL Class I	1.50 V	0.75 V	0.75 V
HSTL Class II	1.50 V	0.75 V	0.75 V
SSTL3 Class I	3.30 V	1.50 V	1.50 V
SSTL3 Class II	3.30 V	1.50 V	1.50 V
SSTL2 Class I	2.50 V	1.25 V	1.25 V
SSTL2 Class II	2.50 V	1.25 V	1.25 V
LVDS, DDR LVDS, B-LVDS, M-LVDS	2.50 V	-	-
LVPECL	3.30 V	-	-

9 – I/O Software Control in Low Power Flash Devices

Fusion, IGLOO, and ProASIC3 I/Os provide more design flexibility, allowing the user to control specific features by enabling certain I/O standards. Some features are selectable only for certain I/O standards, whereas others are available for all I/O standards. For example, slew control is not supported by differential I/O standards. Conversely, I/O register combining is supported by all I/O standards. For detailed information about which I/O standards and features are available on each device and each I/O type, refer to the I/O Structures section of the handbook for the device you are using.

Figure 9-1 shows the various points in the software design flow where a user can provide input or control of the I/O selection and parameters. A detailed description is provided throughout this document.



Figure 9-1 • User I/O Assignment Flow Chart

Compiling the Design

During Compile, a PDC I/O constraint file can be imported along with the netlist file. If only the netlist file is compiled, certain I/O assignments need to be completed before proceeding to Layout. All constraints that can be entered in PDC can also be entered using ChipPlanner, I/O Attribute Editor, and PinEditor.

There are certain rules that must be followed in implementing I/O register combining and the I/O DDR macro (refer to the I/O Registers section of the handbook for the device that you are using and the "DDR" section on page 256 for details). Provided these rules are met, the user can enable or disable I/O register combining by using the PDC command set_io portname -register yes |no in the I/O Attribute Editor or selecting a check box in the Compile Options dialog box (see Figure 9-7). The Compile Options dialog box appears when the design is compiled for the first time. It can also be accessed by choosing **Options** > **Compile** during successive runs. I/O register combining is off by default. The PDC command overrides the setting in the Compile Options dialog box.

Figure 9-7 • Setting Register Combining During Compile

Understanding the Compile Report

The I/O bank report is generated during Compile and displayed in the log window. This report lists the I/O assignments necessary before Layout can proceed.

When Designer is started, the I/O Bank Assigner tool is run automatically if the Layout command is executed. The I/O Bank Assigner takes care of the necessary I/O assignments. However, these assignments can also be made manually with MVN or by importing the PDC file. Refer to the "Assigning Technologies and VREF to I/O Banks" section on page 264 for further description.

The I/O bank report can also be extracted from Designer by choosing **Tools** > **Report** and setting the Report Type to **IOBank**.

This report has the following tables: I/O Function, I/O Technology, I/O Bank Resource Usage, and I/O Voltage Usage. This report is useful if the user wants to do I/O assignments manually.

I/O Software Control in Low Power Flash Devices

List of Changes

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

Date	Changes	Page		
August 2012	The notes in Table 9-2 • Designer State (resulting from I/O attribute modification) were revised to clarify which device families support programmable input delay (SAR 39666).	253		
June 2011	Figure 9-2 • SmartGen Catalog was updated (SAR 24310). Figure 8-3 • Expanded I/O Section and the step associated with it were deleted to reflect changes in the software.			
	The following rule was added to the "VREF Rules for the Implementation of Voltage-Referenced I/O Standards" section:	265		
	Only minibanks that contain input or bidirectional I/Os require a VREF. A VREF is not needed for minibanks composed of output or tristated I/Os (SAR 24310).	I		
July 2010	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)	.4 IGLOO nano and ProASIC3 nano devices were added to Table 9-1 • Flash-Based ecember 2008) FPGAs.			
	The notes for Table 9-2 • Designer State (resulting from I/O attribute modification) were revised to indicate that skew control and input delay do not apply to nano devices.	253		
v1.3 (October 2008)	The "Flash FPGAs I/O Support" section was revised to include new families and make the information more concise.	252		
v1.2 (June 2008)	 The following changes were made to the family descriptions in Table 9-1 • Flash-Based FPGAs: ProASIC3L was updated to include 1.5 V. The number of PLLs for ProASIC3E was changed from five to six. 	252		
v1.1 (March 2008)	This document was previously part of the <i>I/O Structures in IGLOO and ProASIC3 Devices</i> document. The content was separated and made into a new document.	N/A		
	Table 9-2 • Designer State (resulting from I/O attribute modification) was updated to include note 2 for IGLOO PLUS.	253		



Types of Programming for Flash Devices

The number of devices to be programmed will influence the optimal programming methodology. Those available are listed below:

- In-system programming
 - Using a programmer
 - Using a microprocessor or microcontroller
- Device programmers
 - Single-site programmers
 - Multi-site programmers, batch programmers, or gang programmers
 - Automated production (robotic) programmers
- Volume programming services
 - Microsemi in-house programming
 - Programming centers

In-System Programming

Device Type Supported: Flash

٠

ISP refers to programming the FPGA after it has been mounted on the system printed circuit board. The FPGA may be preprogrammed and later reprogrammed using ISP.

The advantage of using ISP is the ability to update the FPGA design many times without any changes to the board. This eliminates the requirement of using a socket for the FPGA, saving cost and improving reliability. It also reduces programming hardware expenses, as the ISP methodology is die-/package-independent.

There are two methods of in-system programming: external and internal.

Programmer ISP—Refer to the "In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" section on page 327 for more information.

Using an external programmer and a cable, the device can be programmed through a header on the system board. In Microsemi SoC Products Group documentation, this is referred to as external ISP. Microsemi provides FlashPro4, FlashPro3, FlashPro Lite, or Silicon Sculptor 3 to perform external ISP. Note that Silicon Sculptor II and Silicon Sculptor 3 can only provide ISP for ProASIC and ProASIC^{PLUS®} families, not for SmartFusion, Fusion, IGLOO, or ProASIC3. Silicon Sculptor II and Silicon Sculptor 3 can be used for programming ProASIC and ProASIC^{PLUS®} devices by using an adapter module (part number SMPA-ISP-ACTEL-3).

- Advantages: Allows local control of programming and data files for maximum security. The programming algorithms and hardware are available from Microsemi. The only hardware required on the board is a programming header.
- Limitations: A negligible board space requirement for the programming header and JTAG signal routing
- Microprocessor ISP—Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

Using a microprocessor and an external or internal memory, you can store the program in memory and use the microprocessor to perform the programming. In Microsemi documentation, this is referred to as internal ISP. Both the code for the programming algorithm and the FPGA programming file must be stored in memory on the board. Programming voltages must also be generated on the board.

- Advantages: The programming code is stored in the system memory. An external programmer is not required during programming.
- Limitations: This is the approach that requires the most design work, since some way of getting and/or storing the data is needed; a system interface to the device must be designed; and the low-level API to the programming firmware must be written and linked into the code provided by Microsemi. While there are benefits to this methodology, serious thought and planning should go into the decision.

Programming Solutions

Details for the available programmers can be found in the programmer user's guides listed in the "Related Documents" section on page 297.

All the programmers except FlashPro4, FlashPro3, FlashPro Lite, and FlashPro require adapter modules, which are designed to support device packages. All modules are listed on the Microsemi SoC Products Group website at

http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx. They are not listed in this document, since this list is updated frequently with new package options and any upgrades required to improve programming yield or support new families.

Programmer	Vendor	ISP	Single Device	Multi-Device	Availability
FlashPro4	Microsemi	Only	Yes	Yes ¹	Available
FlashPro3	Microsemi	Only	Yes	Yes ¹	Available
FlashPro Lite ²	Microsemi	Only	Yes	Yes ¹	Available
FlashPro	Microsemi	Only	Yes	Yes ¹	Discontinued
Silicon Sculptor 3	Microsemi	Yes ³	Yes	Cascade option (up to two)	Available
Silicon Sculptor II	Microsemi	Yes ³	Yes	Cascade option (up to two)	Available
Silicon Sculptor	Microsemi	Yes	Yes	Cascade option (up to four)	Discontinued
Sculptor 6X	Microsemi	No	Yes	Yes	Discontinued
BP MicroProgrammers	BP Microsystems	No	Yes	Yes	Contact BP Microsystems at www.bpmicro.com

Table 11-3 • Programming Solutions

Notes:

1. Multiple devices can be connected in the same JTAG chain for programming.

2. If FlashPro Lite is used for programming, the programmer derives all of its power from the target pc board's VDD supply. The FlashPro Lite's VPP and VPN power supplies use the target pc board's VDD as a power source. The target pc board must supply power to both the VDDP and VDD power pins of the ProASIC^{PLUS} device in addition to supplying VDD to the FlashPro Lite. The target pc board needs to provide at least 500 mA of current to the FlashPro Lite VDD connection for programming.

3. Silicon Sculptor II and Silicon Sculptor 3 can only provide ISP for ProASIC and ProASIC^{PLUS} families, not for Fusion, IGLOO, or ProASIC3 devices.

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

Microprocessor Programming of Microsemi's Low Power Flash Devices

Remote Upgrade via TCP/IP

Transmission Control Protocol (TCP) provides a reliable bitstream transfer service between two endpoints on a network. TCP depends on Internet Protocol (IP) to move packets around the network on its behalf. TCP protects against data loss, data corruption, packet reordering, and data duplication by adding checksums and sequence numbers to transmitted data and, on the receiving side, sending back packets and acknowledging the receipt of data.

The system containing the low power flash device can be assigned an IP address when deployed in the field. When the device requires an update (core or FlashROM), the programming instructions along with the new programming data (AES-encrypted cipher text) can be sent over the Internet to the target system via the TCP/IP protocol. Once the MCU receives the instruction and data, it can proceed with the FPGA update. Low power flash devices support Message Authentication Code (MAC), which can be used to validate data for the target device. More details are given in the "Message Authentication Code (MAC) Validation/Authentication" section.

Hardware Requirement

To facilitate the programming of the low power flash families, the system must have a microprocessor (with access to the device JTAG pins) to process the programming algorithm, memory to store the programming algorithm, programming data, and the necessary programming voltage. Refer to the relevant datasheet for programming voltages.

Security

Encrypted Programming

As an additional security measure, the devices are equipped with AES decryption. AES works in two steps. The first step is to program a key into the devices in a secure or trusted programming center (such as Microsemi SoC Products Group In-House Programming (IHP) center). The second step is to encrypt any programming files with the same encryption key. The encrypted programming file will only work with the devices that have the same key. The AES used in the low power flash families is the 128-bit AES decryption engine (Rijndael algorithm).

Message Authentication Code (MAC) Validation/Authentication

As part of the AES decryption flow, the devices are equipped with a MAC validation/authentication system. MAC is an authentication tag, also called a checksum, derived by applying an on-chip authentication scheme to a STAPL file as it is loaded into the FPGA. MACs are computed and verified with the same key so they can only be verified by the intended recipient. When the MCU system receives the AES-encrypted programming data (cipher text), it can validate the data by loading it into the FPGA and performing a MAC verification prior to loading the data, via a second programming pass, into the FPGA core cells. This prevents erroneous or corrupt data from getting into the FPGA.

Low power flash devices with AES and MAC are superior to devices with only DES or 3DES encryption. Because the MAC verifies the correctness of the data, the FPGA is protected from erroneous loading of invalid programming data that could damage a device (Figure 15-5 on page 355).

The AES with MAC enables field updates over public networks without fear of having the design stolen. An encrypted programming file can only work on devices with the correct key, rendering any stolen files

SRAM Initialization

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

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

FlashROM Read-Back Using JTAG

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

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



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

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 ± 0.25 V	0.8 V ± 0.3 V
IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices running at VCC = 1.2 V	0.75 V ± 0.2 V	0.8 V ± 0.15 V

PLL Behavior at Brownout Condition

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

VCCPLL and VCC Tied Together

In this condition, both VCC and VCCPLL drop below the 0.75 V (\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.