

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

E·XFI

Product Status	Obsolete
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	110592
Number of I/O	177
Number of Gates	600000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	256-LBGA
Supplier Device Package	256-FPBGA (17x17)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/m1a3p600l-1fg256

Email: info@E-XFL.COM

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

### IGLOO nano and IGLOO PLUS I/O State in Flash\*Freeze Mode

In IGLOO nano and IGLOO PLUS devices, users have multiple options in how to configure I/Os during Flash\*Freeze mode:

- 1. Hold the previous state
- 2. Set I/O pad to weak pull-up or pull-down
- 3. Tristate I/O pads

The I/O configuration must be configured by the user in the I/O Attribute Editor or in a PDC constraint file, and can be done on a pin-by-pin basis. The output hold feature will hold the output in the last registered state, using the I/O pad weak pull-up or pull-down resistor when the FF pin is asserted. When inputs are configured with the hold feature enabled, the FPGA core side of the input will hold the last valid state of the input pad before the device entered Flash\*Freeze mode. The input pad can be driven to any value, configured as tristate, or configured with the weak pull-up or pull-down I/O pad feature during Flash\*Freeze mode without affecting the hold state. If the weak pull-up or pull-down feature is used without the output hold feature, the input and output pads will maintain the configured weak pull-up or pull-down is defined on an output buffer or as bidirectional in output mode, and a hold state is also defined for the same pin, the pin will be configured with the predefined weak pull-up or pull-down. Any I/Os that do not use the hold state or I/O pad weak pull-up or pull-down features will be tristated during Flash\*Freeze mode and the FPGA core will be driven High by inputs. Inputs that are tristated during Flash\*Freeze mode may be left floating without any reliability concern or impact to power consumption.

Table 2-6 shows the I/O pad state based on the configuration and buffer type.

Note that configuring weak pull-up or pull-down for the FF pin is not allowed.

Buffer Type		Hold State	I/O Pad Weak Pull-Up/-Down	I/O Pad State in Flash*Freeze Mode
Input		Enabled	Enabled	Weak pull-up/pull-down <sup>1</sup>
		Disabled	Enabled	Weak pull-up/pull-down <sup>2</sup>
		Enabled	Disabled	Tristate <sup>1</sup>
		Disabled	Disabled	Tristate <sup>2</sup>
Output		Enabled	"Don't care"	Weak pull to hold state
		Disabled	Enabled	Weak pull-up/pull-down
		Disabled	Disabled	Tristate
Bidirectional / Tristate Buffer	E = 0 (input/tristate)	Enabled	Enabled	Weak pull-up/pull-down <sup>1</sup>
		Disabled	Enabled	Weak pull-up/pull-down <sup>2</sup>
		Enabled	Disabled	Tristate <sup>1</sup>
		Disabled	Disabled	Tristate <sup>2</sup>
	E = 1 (output)	Enabled	"Don't care"	Weak pull to hold state <sup>3</sup>
		Disabled	Enabled	Weak pull-up/pull-down
		Disabled	Disabled	Tristate

#### Table 2-6 • IGLOO nano and IGLOO PLUS Flash\*Freeze Mode (type 1 and type 2)—I/O Pad State

Notes:

- 1. Internal core logic driven by this input buffer will be set to the value this I/O had when entering Flash\*Freeze mode.
- 2. Internal core logic driven by this input buffer will be tied High as long as the device is in Flash\*Freeze mode.
- 3. For bidirectional buffers: Internal core logic driven by the input portion of the bidirectional buffer will be set to the hold state.

ProASIC3L FPGA Fabric User's Guide



Figure 3-12 • Chip Global Region



Figure 3-13 • Quadrant Global Region



Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs

### Core Logic Clock Source

*Core logic* refers to internal routed nets. Internal routed signals access the CCC via the FPGA Core Fabric. Similar to the External I/O option, whenever the clock source comes internally from the core itself, the routed signal is instantiated with a PLLINT macro before connecting to the CCC clock input (see Figure 4-12 for an example illustration of the connections, shown in red).



#### Figure 4-12 • Illustration of Core Logic Usage

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

#### Figure 4-31 • Static Timing Analysis Using SmartTime

#### Place-and-Route Stage Considerations

Several considerations must be noted to properly place the CCC macros for layout. For CCCs with clock inputs configured with the Hardwired I/O–Driven option:

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

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

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

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

Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs

## **Recommended Board-Level Considerations**

The power to the PLL core is supplied by VCCPLA/B/C/D/E/F (VCCPLx), and the associated ground connections are supplied by VCOMPLA/B/C/D/E/F (VCOMPLx). When the PLLs are not used, the Designer place-and-route tool automatically disables the unused PLLs to lower power consumption. The user should tie unused VCCPLx and VCOMPLx pins to ground. Optionally, the PLL can be turned on/off during normal device operation via the POWERDOWN port (see Table 4-3 on page 84).

## PLL Power Supply Decoupling Scheme

The PLL core is designed to tolerate noise levels on the PLL power supply as specified in the datasheets. When operated within the noise limits, the PLL will meet the output peak-to-peak jitter specifications specified in the datasheets. User applications should always ensure the PLL power supply is powered from a noise-free or low-noise power source.

However, in situations where the PLL power supply noise level is higher than the tolerable limits, various decoupling schemes can be designed to suppress noise to the PLL power supply. An example is provided in Figure 4-38. The VCCPLx and VCOMPLx pins correspond to the PLL analog power supply and ground.

Microsemi strongly recommends that two ceramic capacitors (10 nF in parallel with 100 nF) be placed close to the power pins (less than 1 inch away). A third generic 10  $\mu$ F electrolytic capacitor is recommended for low-frequency noise and should be placed farther away due to its large physical size. Microsemi recommends that a 6.8  $\mu$ H inductor be placed between the supply source and the capacitors to filter out any low-/medium- and high-frequency noise. In addition, the PCB layers should be controlled so the VCCPLx and VCOMPLx planes have the minimum separation possible, thus generating a good-quality RF capacitor.

For more recommendations, refer to the Board-Level Considerations application note.

Recommended 100 nF capacitor:

- Producer BC Components, type X7R, 100 nF, 16 V
- BC Components part number: 0603B104K160BT
- Digi-Key part number: BC1254CT-ND
- Digi-Key part number: BC1254TR-ND

Recommended 10 nF capacitor:

- Surface-mount ceramic capacitor
- Producer BC Components, type X7R, 10 nF, 50 V
- BC Components part number: 0603B103K500BT
- Digi-Key part number: BC1252CT-ND
- Digi-Key part number: BC1252TR-ND



Figure 4-38 • Decoupling Scheme for One PLL (should be replicated for each PLL used)

# 6 – SRAM and FIFO Memories in Microsemi's Low Power Flash Devices

## Introduction

As design complexity grows, greater demands are placed upon an FPGA's embedded memory. Fusion, IGLOO, and ProASIC3 devices provide the flexibility of true dual-port and two-port SRAM blocks. The embedded memory, along with built-in, dedicated FIFO control logic, can be used to create cascading RAM blocks and FIFOs without using additional logic gates.

IGLOO, IGLOO PLUS, and ProASIC3L FPGAs contain an additional feature that allows the device to be put in a low power mode called Flash\*Freeze. In this mode, the core draws minimal power (on the order of 2 to 127  $\mu$ W) and still retains values on the embedded SRAM/FIFO and registers. Flash\*Freeze technology allows the user to switch to Active mode on demand, thus simplifying power management and the use of SRAM/FIFOs.

## **Device Architecture**

The low power flash devices feature up to 504 kbits of RAM in 4,608-bit blocks (Figure 6-1 on page 148 and Figure 6-2 on page 149). The total embedded SRAM for each device can be found in the datasheets. These memory blocks are arranged along the top and bottom of the device to allow better access from the core and I/O (in some devices, they are only available on the north side of the device). Every RAM block has a flexible, hardwired, embedded FIFO controller, enabling the user to implement efficient FIFOs without sacrificing user gates.

In the IGLOO and ProASIC3 families of devices, the following memories are supported:

- 30 k gate devices and smaller do not support SRAM and FIFO.
- 60 k and 125 k gate devices support memories on the north side of the device only.
- 250 k devices and larger support memories on the north and south sides of the device.

In Fusion devices, the following memories are supported:

- AFS090 and AFS250 support memories on the north side of the device only.
- AFS600 and AFS1500 support memories on the north and south sides of the device.

SRAM and FIFO Memories in Microsemi's Low Power Flash Devices

```
//
addr_counter counter_1 (.Clock(data_update), .Q(wr_addr), .Aset(rst_n),
    .Enable(enable));
addr_counter counter_2 (.Clock(test_clk), .Q(rd_addr), .Aset(rst_n),
    .Enable( test_active));
```

endmodule

### Interface Block / UJTAG Wrapper

This example is a sample wrapper, which connects the interface block to the UJTAG and the memory blocks.

```
// WRAPPER
module top_init (TDI, TRSTB, TMS, TCK, TDO, test, test_clk, test_ out);
input TDI, TRSTB, TMS, TCK;
output TDO;
input test, test_clk;
output [3:0] test_out;
wire [7:0] IR;
wire reset, DR_shift, DR_cap, init_clk, DR_update, data_in, data_out;
wire clk out, wen, ren;
wire [3:0] word_in, word_out;
wire [1:0] write_addr, read_addr;
UJTAG UJTAG_U1 (.UIREG0(IR[0]), .UIREG1(IR[1]), .UIREG2(IR[2]), .UIREG3(IR[3]),
  .UIREG4(IR[4]), .UIREG5(IR[5]), .UIREG6(IR[6]), .UIREG7(IR[7]), .URSTB(reset),
  .UDRSH(DR_shift), .UDRCAP(DR_cap), .UDRCK(init_clk), .UDRUPD(DR_update),
  .UT-DI(data_in), .TDI(TDI), .TMS(TMS), .TCK(TCK), .TRSTB(TRSTB), .TDO(TDO),
  .UT-DO(data_out));
mem_block RAM_block (.DO(word_out), .RCLOCK(clk_out), .WCLOCK(clk_out), .DI(word_in),
  .WRB(wen), .RDB(ren), .WAD-DR(write_addr), .RADDR(read_addr));
interface init_block (.IR(IR), .rst_n(reset), .data_shift(DR_shift), .clk_in(init_clk),
  .data_update(DR_update), .din_ser(data_in), .dout_ser(data_out), .test(test),
  .test_out(test_out), .test_clk(test_clk), .clk_out(clk_out), .wr_en(wen),
  .rd_en(ren), .write_word(word_in), .read_word(word_out), .rd_addr(read_addr),
  .wr_addr(write_addr));
```

endmodule

### Address Counter

module addr\_counter (Clock, Q, Aset, Enable);

```
input Clock;
output [1:0] Q;
input Aset;
input Enable;
reg [1:0] Qaux;
always @(posedge Clock or negedge Aset)
begin
    if (!Aset) Qaux <= 2'b11;
    else if (Enable) Qaux <= Qaux + 1;
end
assign Q = Qaux;
endmodule
```

# 7 – I/O Structures in IGLOO and ProASIC3 Devices

## Introduction

Low power flash devices feature a flexible I/O structure, supporting a range of mixed voltages (1.2 V, 1.5 V, 1.8 V, 2.5 V, and 3.3 V) through bank-selectable voltages. IGLOO,<sup>®</sup> ProASIC3<sup>®</sup>L, and ProASIC3 families support Standard, Standard Plus, and Advanced I/Os.

Users designing I/O solutions are faced with a number of implementation decisions and configuration choices that can directly impact the efficiency and effectiveness of their final design. The flexible I/O structure, supporting a wide variety of voltages and I/O standards, enables users to meet the growing challenges of their many diverse applications. Libero SoC software provides an easy way to implement I/Os that will result in robust I/O design.

This document first describes the two different I/O types in terms of the standards and features they support. It then explains the individual features and how to implement them in Libero SoC.



Figure 7-1 • DDR Configured I/O Block Logical Representation



I/O Structures in IGLOO and ProASIC3 Devices

## I/O Register Combining

Every I/O has several embedded registers in the I/O tile that are close to the I/O pads. Rather than using the internal register from the core, the user has the option of using these registers for faster clock-to-out timing, and external hold and setup. When combining these registers at the I/O buffer, some architectural rules must be met. Provided these rules are met, the user can enable register combining globally during Compile (as shown in the "Compiling the Design" section on page 261).

This feature is supported by all I/O standards.

### Rules for Registered I/O Function

- 1. The fanout between an I/O pin (D, Y, or E) and a register must be equal to one for combining to be considered on that pin.
- All registers (Input, Output, and Output Enable) connected to an I/O must share the same clear or preset function:
  - If one of the registers has a CLR pin, all the other registers that are candidates for combining in the I/O must have a CLR pin.
  - If one of the registers has a PRE pin, all the other registers that are candidates for combining in the I/O must have a PRE pin.
  - If one of the registers has neither a CLR nor a PRE pin, all the other registers that are candidates for combining must have neither a CLR nor a PRE pin.
  - If the clear or preset pins are present, they must have the same polarity.
  - If the clear or preset pins are present, they must be driven by the same signal (net).
- 3. Registers connected to an I/O on the Output and Output Enable pins must have the same clock and enable function:
  - Both the Output and Output Enable registers must have an E pin (clock enable), or none at all.
  - If the E pins are present, they must have the same polarity. The CLK pins must also have the same polarity.

In some cases, the user may want registers to be combined with the input of a bibuf while maintaining the output as-is. This can be achieved by using PDC commands as follows:

```
set_io <signal name> -REGISTER yes -----register will combine
set_preserve <signal name> ----register will not combine
```

### Weak Pull-Up and Weak Pull-Down Resistors

IGLOO and ProASIC3 devices support optional weak pull-up and pull-down resistors on each I/O pin. When the I/O is pulled up, it is connected to the VCCI of its corresponding I/O bank. When it is pulled down, it is connected to GND. Refer to the datasheet for more information.

For low power applications, configuration of the pull-up or pull-down of the I/O can be used to set the I/O to a known state while the device is in Flash\*Freeze mode. Refer to the "Flash\*Freeze Technology and Low Power Modes in IGLOO and ProASIC3L Devices" chapter of the *IGLOO FPGA Fabric User's Guide* or *ProASIC3L FPGA Fabric User's Guide* for more information.

The Flash\*Freeze (FF) pin cannot be configured with a weak pull-down or pull-up I/O attribute, as the signal needs to be driven at all times.

### **Output Slew Rate Control**

The slew rate is the amount of time an input signal takes to get from logic Low to logic High or vice versa.

It is commonly defined as the propagation delay between 10% and 90% of the signal's voltage swing. Slew rate control is available for the output buffers of low power flash devices. The output buffer has a programmable slew rate for both HIGH-to-LOW and LOW-to-HIGH transitions. Slew rate control is available for LVTTL, LVCMOS, and PCI-X I/O standards. The other I/O standards have a preset slew value.

The slew rate can be implemented by using a PDC command (Table 7-5 on page 179), setting it "High" or "Low" in the I/O Attribute Editor in Designer, or instantiating a special I/O macro. The default slew rate value is "High."

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 Software Control in Low Power Flash Devices

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

Command	Action	Example	Comment		
I/O Attribute Cons	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] [-out_drive value] [-slew value] [-res_pull value] [-schmitt_trigger value] [-in_delay value] [-out_load 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>	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. 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. Example: OUTBUF_S_24 (low slew, output drive 24 mA) Slew and output drive cannot be changed.		
I/O Region Placer	nent 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	assign_region [region name] [macro_name] assign_region test U12	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 \\\$1119\ the correct use of escape characters is \\\\\\$119\\\.		

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

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.	254
	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)	IGLOO nano and ProASIC3 nano devices were added to Table 9-1 • Flash-Based FPGAs.	252
	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)	<ul> <li>The following changes were made to the family descriptions in Table 9-1 • Flash-Based FPGAs:</li> <li>ProASIC3L was updated to include 1.5 V.</li> <li>The number of PLLs for ProASIC3E was changed from five to six.</li> </ul>	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

# 10 – DDR for Microsemi's Low Power Flash Devices

## Introduction

The I/Os in Fusion, IGLOO, and ProASIC3 devices support Double Data Rate (DDR) mode. In this mode, new data is present on every transition (or clock edge) of the clock signal. This mode doubles the data transfer rate compared with Single Data Rate (SDR) mode, where new data is present on one transition (or clock edge) of the clock signal. Low power flash devices have DDR circuitry built into the I/O tiles. I/Os are configured to be DDR receivers or transmitters by instantiating the appropriate special macros (examples shown in Figure 10-4 on page 276 and Figure 10-5 on page 277) and buffers (DDR\_OUT or DDR\_REG) in the RTL design. This document discusses the options the user can choose to configure the I/Os in this mode and how to instantiate them in the design.

## Double Data Rate (DDR) Architecture

Low power flash devices support 350 MHz DDR inputs and outputs. In DDR mode, new data is present on every transition of the clock signal. Clock and data lines have identical bandwidths and signal integrity requirements, making them very efficient for implementing very high-speed systems. High-speed DDR interfaces can be implemented using LVDS (not applicable for IGLOO nano and ProASIC3 nano devices). In IGLOOe, ProASIC3E, AFS600, and AFS1500 devices, DDR interfaces can also be implemented using the HSTL, SSTL, and LVPECL I/O standards. The DDR feature is primarily implemented in the FPGA core periphery and is not tied to a specific I/O technology or limited to any I/O standard.



Figure 10-1 • DDR Support in Low Power Flash Devices

ProASIC3L FPGA Fabric User's Guide



Figure 12-5 • Example Application Scenario Using AES in Fusion Devices

### FlashLock

#### Additional Options for IGLOO and ProASIC3 Devices

The user also has the option of prohibiting Write operations to the FPGA array but allowing Verify operations on the FPGA array and/or Read operations on the FlashROM without the use of the FlashLock Pass Key. This option provides the user the freedom of verifying the FPGA array and/or reading the FlashROM contents after the device is programmed, without having to provide the FlashLock Pass Key. The user can incorporate AES encryption on the programming files to better enhance the level of security used.

### **Permanent Security Setting Options**

In applications where a permanent lock is not desired, yet the security settings should not be modifiable, IGLOO and ProASIC3 devices can accommodate this requirement.

This application is particularly useful in cases where a device is located at a remote location and must be reprogrammed with a design or data update. Refer to the "Application 3: Nontrusted Environment—Field Updates/Upgrades" section on page 310 for further discussion and examples of how this can be achieved.

The user must be careful when considering the Permanent FlashLock or Permanent Security Settings option. Once the design is programmed with the permanent settings, it is not possible to reconfigure the security settings already employed on the device. Therefore, exercise careful consideration before programming permanent settings.

### Permanent FlashLock

The purpose of the permanent lock feature is to provide the benefits of the highest level of security to IGLOO and ProASIC3 devices. If selected, the permanent FlashLock feature will create a permanent barrier, preventing any access to the contents of the device. This is achieved by permanently disabling Write and Verify access to the array, and Write and Read access to the FlashROM. After permanently locking the device, it has been effectively rendered one-time-programmable. This feature is useful if the intended applications do not require design or system updates to the device.

## 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. 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

In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X

Date	Changes	Page
July 2010 (continued)	The "Chain Integrity Test Error Analyze Chain Failure" section was renamed to the "Scan Chain Failure" section, and the Analyze Chain command was changed to Scan Chain. It was noted that occasionally a faulty programmer can cause scan chain failures.	338
v1.5 (August 2009)	The "CoreMP7 Device Security" section was removed from "Security in ARM- Enabled Low Power Flash Devices", since M7-enabled devices are no longer supported.	331
v1.4 (December 2008)	The "ISP Architecture" section was revised to include information about core voltage for IGLOO V2 and ProASIC3L devices, as well as 50 mV increments allowable in Designer software.	327
	IGLOO nano and ProASIC3 nano devices were added to Table 13-1 • Flash-Based FPGAs Supporting ISP.	328
	A second capacitor was added to Figure 13-6 • Board Layout and Programming Header Top View.	337
v1.3 (October 2008)	The "ISP Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	328
v1.2 (June 2008)	<ul> <li>The following changes were made to the family descriptions in Table 13-1 • Flash-Based FPGAs Supporting ISP:</li> <li>ProASIC3L was updated to include 1.5 V.</li> <li>The number of PLLs for ProASIC3E was changed from five to six.</li> </ul>	328
v1.1 (March 2008)	The "ISP Architecture" section was updated to included the IGLOO PLUS family in the discussion of family-specific support. The text, "When 1.2 V is used, the device can be reprogrammed in-system at 1.5 V only," was revised to state, "Although the device can operate at 1.2 V core voltage, the device can only be reprogrammed when all supplies (VCC, VCCI, and VJTAG) are at 1.5 V."	327
	The "ISP Support in Flash-Based Devices" section and Table 13-1 • Flash-Based FPGAs Supporting ISP were updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	328
	The "Security" section was updated to mention that 15 k gate devices do not have a built-in 128-bit decryption core.	330
	Table 13-2 • Power Supplies was revised to remove the Normal Operation column and add a table note stating, "All supply voltages should be at 1.5 V or higher, regardless of the setting during normal operation."	329
	The "ISP Programming Header Information" section was revised to change FP3-26PIN-ADAPTER to FP3-10PIN-ADAPTER-KIT. Table 13-3 • Programming Header Ordering Codes was updated with the same change, as well as adding the part number FFSD-05-D-06.00-01-N, a 10-pin cable with 50-mil-pitch sockets.	335
	The "Board-Level Considerations" section was updated to describe connecting two capacitors in parallel across VPUMP and GND for proper programming.	337
v1.0 (January 2008)	Information was added to the "Programming Voltage (VPUMP) and VJTAG" section about the JTAG interface pin.	329
51900055-2/7.06	ACTgen was changed to SmartGen.	N/A
	In Figure 13-6 • Board Layout and Programming Header Top View, the order of the text was changed to: VJTAG from the target board VCCI from the target board VCC from the target board	337

Microprocessor Programming of Microsemi's Low Power Flash Devices

# List of Changes

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

Date	Changes	Page
September 2012	The "Security" section was modified to clarify that Microsemi does not support read-back of FPGA core-programmed data (SAR 41235).	354
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 15-1 • Flash-Based FPGAs.	350
v1.3 (October 2008)	The "Microprocessor Programming Support in Flash Devices" section was revised to include new families and make the information more concise.	350
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 15-1 • Flash-Based FPGAs:	350
	<ul> <li>ProASIC3L was updated to include 1.5 V.</li> </ul>	
	The number of PLLs for ProASIC3E was changed from five to six.	
v1.1 (March 2008)	The "Microprocessor Programming Support in Flash Devices" section was updated to include information on the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	350

## Fine Tuning

In some applications, design constants or parameters need to be modified after programming the original design. The tuning process can be done using the UJTAG tile without reprogramming the device with new values. If the parameters or constants of a design are stored in distributed registers or embedded SRAM blocks, the new values can be shifted onto the JTAG TAP Controller pins, replacing the old values. The UJTAG tile is used as the "bridge" for data transfer between the JTAG pins and the FPGA VersaTiles or SRAM logic. Figure 17-5 shows a flow chart example for fine-tuning application steps using the UJTAG tile.

In Figure 17-5, the TMS signal sets the TAP Controller state machine to the appropriate states. The flow mainly consists of two steps: a) shifting the defined instruction and b) shifting the new data. If the target parameter is constantly used in the design, the new data can be shifted into a temporary shift register from UTDI. The UDRSH output of UJTAG can be used as a shift-enable signal, and UDRCK is the shift clock to the shift register. Once the shift process is completed and the TAP Controller state is moved to the Update\_DR state, the UDRUPD output of the UJTAG can latch the new parameter value from the temporary register into a permanent location. This avoids any interruption or malfunctioning during the serial shift of the new value.



Figure 17-5 • Flow Chart Example of Fine-Tuning an Application Using UJTAG

## **SRAM** Initialization

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

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

## FlashROM Read-Back Using JTAG

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

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



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