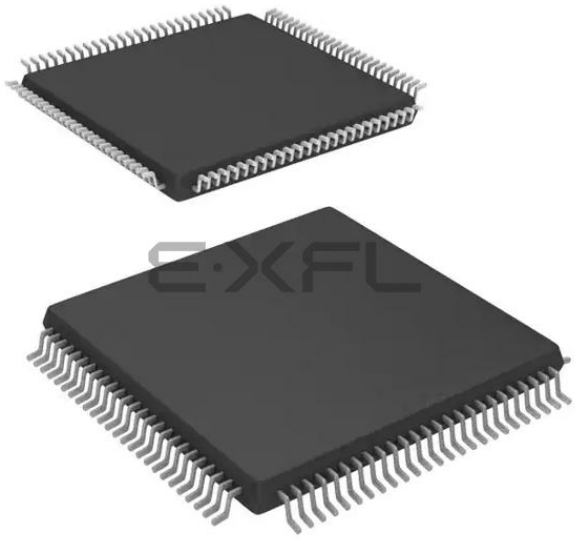

Welcome to E-XFL.COM

Understanding Embedded - FPGAs (Field Programmable Gate Array)

Embedded - FPGAs, or Field Programmable Gate Arrays, are advanced integrated circuits that offer unparalleled flexibility and performance for digital systems. Unlike traditional fixed-function logic devices, FPGAs can be programmed and reprogrammed to execute a wide array of logical operations, enabling customized functionality tailored to specific applications. This reprogrammability allows developers to iterate designs quickly and implement complex functions without the need for custom hardware.

Applications of Embedded - FPGAs

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

Details	
Product Status	Obsolete
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	36864
Number of I/O	68
Number of Gates	250000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	100-TQFP
Supplier Device Package	100-VQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/a3p250l-vqg100i

1 – FPGA Array Architecture in Low Power Flash Devices

Device Architecture

Advanced Flash Switch

Unlike SRAM FPGAs, the low power flash devices use a live-at-power-up ISP flash switch as their programming element. Flash cells are distributed throughout the device to provide nonvolatile, reconfigurable programming to connect signal lines to the appropriate VersaTile inputs and outputs. In the flash switch, two transistors share the floating gate, which stores the programming information (Figure 1-1). One is the sensing transistor, which is only used for writing and verification of the floating gate voltage. The other is the switching transistor. The latter is used to connect or separate routing nets, or to configure VersaTile logic. It is also used to erase the floating gate. Dedicated high-performance lines are connected as required using the flash switch for fast, low-skew, global signal distribution throughout the device core. Maximum core utilization is possible for virtually any design. The use of the flash switch technology also removes the possibility of firm errors, which are increasingly common in SRAM-based FPGAs.

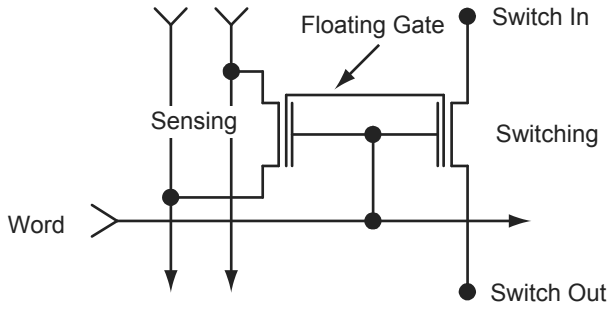

Figure 1-1 • **Flash-Based Switch**

Sleep and Shutdown Modes

Sleep Mode

IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 FPGAs support Sleep mode when device functionality is not required. In Sleep mode, V_{CC} (core voltage), V_{JTAG} (JTAG DC voltage), and VPUMP (programming voltage) are grounded, resulting in the FPGA core being turned off to reduce power consumption. While the device is in Sleep mode, the rest of the system can still be operating and driving the input buffers of the device. The driven inputs do not pull up the internal power planes, and the current draw is limited to minimal leakage current.

Table 2-7 shows the power supply status in Sleep mode.

Table 2-7 • **Sleep Mode—Power Supply Requirement for IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 Devices**

Power Supplies	Power Supply State
VCC	Powered off
VCCI = VMV	Powered on
VJTAG	Powered off
VPUMP	Powered off

Refer to the "Power-Up/-Down Behavior" section on page 33 for more information about I/O states during Sleep mode and the timing diagram for entering and exiting Sleep mode.

Shutdown Mode

Shutdown mode is supported for all IGLOO nano and IGLOO PLUS devices as well the following IGLOO/e devices: AGL015, AGL030, AGLE600, AGLE3000, and A3PE3000L. Shutdown mode can be used by turning off all power supplies when the device function is not needed. Cold-sparing and hot-insertion features enable these devices to be powered down without turning off the entire system. When power returns, the live-at-power-up feature enables operation of the device after reaching the voltage activation point.

Each CCC can implement up to three independent global buffers (with or without programmable delay) or a PLL function (programmable frequency division/multiplication, phase shift, and delays) with up to three global outputs. Unused global outputs of a PLL can be used to implement independent global buffers, up to a maximum of three global outputs for a given CCC.

CCC Programming

The CCC block is fully configurable, either via flash configuration bits set in the programming bitstream or through an asynchronous interface. This asynchronous dedicated shift register interface is dynamically accessible from inside the low power flash devices to permit parameter changes, such as PLL divide ratios and delays, during device operation.

To increase the versatility and flexibility of the clock conditioning system, the CCC configuration is determined either by the user during the design process, with configuration data being stored in flash memory as part of the device programming procedure, or by writing data into a dedicated shift register during normal device operation.

This latter mode allows the user to dynamically reconfigure the CCC without the need for core programming. The shift register is accessed through a simple serial interface. Refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 363 or the application note *Using Global Resources in Actel Fusion Devices*.

Global Resources

Low power flash and mixed signal devices provide three global routing networks (GLA, GLB, and GLC) for each of the CCC locations. There are potentially many I/O locations; each global I/O location can be chosen from only one of three possibilities. This is controlled by the multiplexer tree circuitry in each global network. Once the I/O location is selected, the user has the option to utilize the CCCs before the signals are connected to the global networks. The CCC in each location (up to six) has the same structure, so generating the CCC macros is always done with an identical software GUI. The CCCs in the corner locations drive the quadrant global networks, and the CCCs in the middle of the east and west chip sides drive the chip global networks. The quadrant global networks span only a quarter of the device, while the chip global networks span the entire device. For more details on global resources offered in low power flash devices, refer to the "Global Resources in Low Power Flash Devices" section on page 47.

A global buffer can be placed in any of the three global locations (CLKA-GLA, CLKB-GLB, or CLKC-GLC) of a given CCC. A PLL macro uses the CLKA CCC input to drive its reference clock. It uses the GLA and, optionally, the GLB and GLC global outputs to drive the global networks. A PLL macro can also drive the YB and YC regular core outputs. The GLB (or GLC) global output cannot be reused if the YB (or YC) output is used. Refer to the "PLL Macro Signal Descriptions" section on page 84 for more information.

Each global buffer, as well as the PLL reference clock, can be driven from one of the following:

- 3 dedicated single-ended I/Os using a hardwired connection
- 2 dedicated differential I/Os using a hardwired connection (not supported for IGLOO nano or ProASIC3 nano devices)
- The FPGA core

Dividers *n* and *m* (the input divider and feedback divider, respectively) provide integer frequency division factors from 1 to 128. The output dividers *u*, *v*, and *w* provide integer division factors from 1 to 32. Frequency scaling of the reference clock CLKA is performed according to the following formulas:

$$f_{GLA} = f_{CLKA} \times m / (n \times u) \text{ – GLA Primary PLL Output Clock}$$

EQ 4-1

$$f_{GLB} = f_{YB} = f_{CLKA} \times m / (n \times v) \text{ – GLB Secondary 1 PLL Output Clock(s)}$$

EQ 4-2

$$f_{GLC} = f_{YC} = f_{CLKA} \times m / (n \times w) \text{ – GLC Secondary 2 PLL Output Clock(s)}$$

EQ 4-3

SmartGen provides a user-friendly method of generating the configured PLL netlist, which includes automatically setting the division factors to achieve the closest possible match to the requested frequencies. Since the five output clocks share the *n* and *m* dividers, the achievable output frequencies are interdependent and related according to the following formula:

$$f_{GLA} = f_{GLB} \times (v / u) = f_{GLC} \times (w / u)$$

EQ 4-4

Clock Delay Adjustment

There are a total of seven configurable delay elements implemented in the PLL architecture.

Two of the delays are located in the feedback path, entitled System Delay and Feedback Delay. System Delay provides a fixed delay of 2 ns (typical), and Feedback Delay provides selectable delay values from 0.6 ns to 5.56 ns in 160 ps increments (typical). For PLLs, delays in the feedback path will effectively advance the output signal from the PLL core with respect to the reference clock. Thus, the System and Feedback delays generate negative delay on the output clock. Additionally, each of these delays can be independently bypassed if necessary.

The remaining five delays perform traditional time delay and are located at each of the outputs of the PLL. Besides the fixed global driver delay of 0.755 ns for each of the global networks, the global multiplexer outputs (GLA, GLB, and GLC) each feature an additional selectable delay value, as given in Table 4-7.

***Table 4-7* • Delay Values in Libero SoC Software per Device Family**

Device	Typical	Starting Values	Increments	Ending Value
ProASIC3	200 ps	0 to 735 ps	200 ps	6.735 ns
IGLOO/ProASIC3L 1.5 V	360 ps	0 to 1.610 ns	360 ps	12.410 ns
IGLOO/ProASIC3L 1.2 V	580 ps	0 to 2.880 ns	580 ps	20.280 ns

The additional YB and YC signals have access to a selectable delay from 0.6 ns to 5.56 ns in 160 ps increments (typical). This is the same delay value as the CLKDLY macro. It is similar to CLKDLY, which bypasses the PLL core just to take advantage of the phase adjustment option with the delay value.

The following parameters must be taken into consideration to achieve minimum delay at the outputs (GLA, GLB, GLC, YB, and YC) relative to the reference clock: routing delays from the PLL core to CCC outputs, core outputs and global network output delays, and the feedback path delay. The feedback path delay acts as a time advance of the input clock and will offset any delays introduced beyond the PLL core output. The routing delays are determined from back-annotated simulation and are configuration-dependent.

The following is an example of a PLL configuration utilizing the clock frequency synthesis and clock delay adjustment features. The steps include generating the PLL core with SmartGen, performing simulation for verification with Model*Sim*, and performing static timing analysis with SmartTime in Designer.

Parameters of the example PLL configuration:

Input Frequency – 20 MHz

Primary Output Requirement – 20 MHz with clock advancement of 3.02 ns

Secondary 1 Output Requirement – 40 MHz with clock delay of 2.515 ns

Figure 4-29 shows the SmartGen settings. Notice that the overall delays are calculated automatically, allowing the user to adjust the delay elements appropriately to obtain the desired delays.

Figure 4-29 • **SmartGen Settings**

After confirming the correct settings, generate a structural netlist of the PLL and verify PLL core settings by checking the log file:

```
Name                          : test_pll_delays
Family                        : ProASIC3E
Output Format                 : VHDL
Type                          : Static PLL
Input Freq(MHz)               : 20.000
CLKA Source                   : Hardwired I/O
Feedback Delay Value Index    : 21
Feedback Mux Select           : 2
XDLY Mux Select               : No
Primary Freq(MHz)             : 20.000
Primary PhaseShift            : 0
Primary Delay Value Index     : 1
Primary Mux Select            : 4
Secondary1 Freq(MHz)          : 40.000
Use GLB                       : YES
Use YB                        : NO
…
…
…
Primary Clock frequency 20.000
Primary Clock Phase Shift 0.000
```

Figure 4-36 • **Second-Stage PLL Showing Input of 256 MHz from First Stage and Final Output of 280 MHz**

Figure 4-37 shows the simulation results, where the first PLL's output period is 3.9 ns (~256 MHz), and the stage 2 (final) output period is 3.56 ns (~280 MHz).

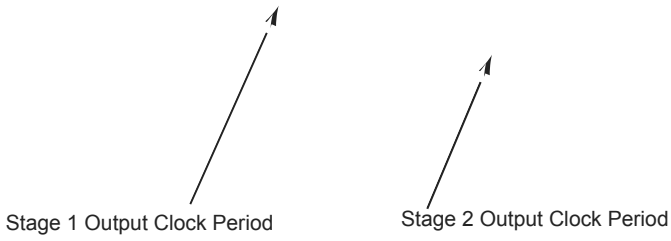

Figure 4-37 • **Model*Sim* Simulation Results**

SmartGen enables the user to configure the desired RAM element to use either a single clock for read and write, or two independent clocks for read and write. The user can select the type of RAM as well as the width/depth and several other parameters (Figure 6-13).

***Figure 6-13* • SmartGen Memory Configuration Interface**

SmartGen also has a Port Mapping option that allows the user to specify the names of the ports generated in the memory block (Figure 6-14).

***Figure 6-14* • Port Mapping Interface for SmartGen-Generated Memory**

SmartGen also configures the FIFO according to user specifications. Users can select no flags, static flags, or dynamic flags. Static flag settings are configured using configuration flash and cannot be altered

- In Active and Static modes:
  - Input buffers with pull-up, driven Low
  - Input buffers with pull-down, driven High
  - Bidirectional buffers with pull-up, driven Low
  - Bidirectional buffers with pull-down, driven High
  - Output buffers with pull-up, driven Low
  - Output buffers with pull-down, driven High
  - Tristate buffers with pull-up, driven Low
  - Tristate buffers with pull-down, driven High
- In Flash*Freeze mode:
  - Input buffers with pull-up, driven Low
  - Input buffers with pull-down, driven High
  - Bidirectional buffers with pull-up, driven Low
  - Bidirectional buffers with pull-down, driven High

Electrostatic Discharge Protection

Low power flash devices are tested per JEDEC Standard JESD22-A114-B.

These devices contain clamp diodes at every I/O, global, and power pad. Clamp diodes protect all device pads against damage from ESD as well as from excessive voltage transients.

All IGLOO and ProASIC3 devices are tested to the Human Body Model (HBM) and the Charged Device Model (CDM).

Each I/O has two clamp diodes. One diode has its positive (P) side connected to the pad and its negative (N) side connected to VCCI. The second diode has its P side connected to GND and its N side connected to the pad. During operation, these diodes are normally biased in the off state, except when transient voltage is significantly above VCCI or below GND levels.

In 30K gate devices, the first diode is always off. In other devices, the clamp diode is always on and cannot be switched off.

By selecting the appropriate I/O configuration, the diode is turned on or off. Refer to Table 7-12 on page 193 for more information about the I/O standards and the clamp diode.

The second diode is always connected to the pad, regardless of the I/O configuration selected.

Board-Level Considerations

Low power flash devices have robust I/O features that can help in reducing board-level components. The devices offer single-chip solutions, which makes the board layout simpler and more immune to signal integrity issues. Although, in many cases, these devices resolve board-level issues, special attention should always be given to overall signal integrity. This section covers important board-level considerations to facilitate optimum device performance.

Termination

Proper termination of all signals is essential for good signal quality. Nonterminated signals, especially clock signals, can cause malfunctioning of the device.

For general termination guidelines, refer to the *Board-Level Considerations* application note for Microsemi FPGAs. Also refer to the "Pin Descriptions" chapter of the appropriate datasheet for termination requirements for specific pins.

Low power flash I/Os are equipped with on-chip pull-up/-down resistors. The user can enable these resistors by instantiating them either in the top level of the design (refer to the *IGLOO, Fusion, and ProASIC3 Macro Library Guide* for the available I/O macros with pull-up/-down) or in the I/O Attribute Editor in Designer if generic input or output buffers are instantiated in the top level. Unused I/O pins are configured as inputs with pull-up resistors.

As mentioned earlier, low power flash devices have multiple programmable drive strengths, and the user can eliminate unwanted overshoot and undershoot by adjusting the drive strengths.

Power-Up Behavior

Low power flash devices are power-up/-down friendly; i.e., no particular sequencing is required for power-up and power-down. This eliminates extra board components for power-up sequencing, such as a power-up sequencer.

During power-up, all I/Os are tristated, irrespective of I/O macro type (input buffers, output buffers, I/O buffers with weak pull-ups or weak pull-downs, etc.). Once I/Os become activated, they are set to the user-selected I/O macros. Refer to the "Power-Up/-Down Behavior of Low Power Flash Devices" section on page 373 for details.

Drive Strength

Low power flash devices have up to seven programmable output drive strengths. The user can select the drive strength of a particular output in the I/O Attribute Editor or can instantiate a specialized I/O macro, such as OUTBUF_S_12 (slew = low, out_drive = 12 mA).

The maximum available drive strength is 24 mA per I/O. Though no I/O should be forced to source or sink more than 24 mA indefinitely, I/Os may handle a higher amount of current (refer to the device IBIS model for maximum source/sink current) during signal transition (AC current). Every device package has its own power dissipation limit; hence, power calculation must be performed accurately to determine how much current can be tolerated per I/O within that limit.

I/O Interfacing

Low power flash devices are 5 V–input– and 5 V–output–tolerant if certain I/O standards are selected (refer to the "5 V Input and Output Tolerance" section on page 194). Along with other low-voltage I/O macros, this 5 V tolerance makes these devices suitable for many types of board component interfacing.

***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	`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`
			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 Placement Constraints			

define_region	Defines either a rectangular region or a rectilinear region	`define_region`
		`-name [region_name]`
		`-type [region_type] x1 y1 x2 y2`
		
		`define_region -name test`
		`-type inclusive 0 15 2 29`
			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 \\$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.*

I/O Function

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

***Figure 9-8* • I/O Function Table**

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

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

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

I/O Technology

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

***Figure 9-9* • I/O Technology Table**

those banks, the user does not need to assign the same VCCI voltage to another bank. The user needs to assign the other three VCCI voltages to three more banks.

Assigning Technologies and VREF to I/O Banks

Low power flash devices offer a wide variety of I/O standards, including voltage-referenced standards. Before proceeding to Layout, each bank must have the required VCCI voltage assigned for the corresponding I/O technologies used for that bank. The voltage-referenced standards require the use of a reference voltage (VREF). This assignment can be done manually or automatically. The following sections describe this in detail.

Manually Assigning Technologies to I/O Banks

The user can import the PDC at this point and resolve this requirement. The PDC command is

```
set_iobank [bank name] –vcci [vcci value]
```

Another method is to use the I/O Bank Settings dialog box (**MVN** > **Edit** > **I/O Bank Settings**) to set up the V_{CCI} voltage for the bank (Figure 9-12).

Figure 9-12 **• Setting VCCI for a Bank**

Related Documents

Below is a list of related documents, their location on the Microsemi SoC Products Group website, and a brief summary of each document.

Application Notes

Programming Antifuse Devices

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

Implementation of Security in Actel's ProASIC and ProASIC^{PLUS} Flash-Based FPGAs

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

User's Guides

FlashPro Programmers

FlashPro4,[1] FlashPro3, FlashPro Lite, and FlashPro[2]

http://www.microsemi.com/soc/products/hardware/program_debug/flashpro/default.aspx

FlashPro User's Guide

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

The FlashPro User's Guide includes hardware and software setup, self-test instructions, use instructions, and a troubleshooting / error message guide.

Silicon Sculptor 3 and Silicon Sculptor II

http://www.microsemi.com/soc/products/hardware/program_debug/ss/default.aspx

Other Documents

http://www.microsemi.com/soc/products/solutions/security/default.aspx#flashlock

The security resource center describes security in Microsemi Flash FPGAs.

Quality and Reliability Guide

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

Programming and Functional Failure Guidelines

http://www.microsemi.com/soc/documents/FA_Policies_Guidelines_5-06-00002.pdf

1. *FlashPro4 replaced FlashPro3 in Q1 2010.*
2. *FlashPro is no longer available.*

Security Support in Flash-Based Devices

The flash FPGAs listed in Table 12-1 support the security feature and the functions described in this document.

***Table 12-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM Cortex™-M1 soft processors, and flash memory into a monolithic device

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

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 12-1. 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 12-1. 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*.

The AES key is securely stored on-chip in dedicated low power flash device flash memory and cannot be read out. In the first step, the AES key is generated and programmed into the device (for example, at a secure or trusted programming site). The Microsemi Designer software tool provides AES key generation capability. After the key has been programmed into the device, the device will only correctly decrypt programming files that have been encrypted with the same key. If the individual programming file content is incorrect, a Message Authentication Control (MAC) mechanism inside the device will fail in authenticating the programming file. In other words, when an encrypted programming file is being loaded into a device that has a different programmed AES key, the MAC will prevent this incorrect data from being loaded, preventing possible device damage. See Figure 12-3 on page 304 and Figure 12-4 on page 306 for graphical representations of this process.

It is important to note that the user decides what level of protection will be implemented for the device. When AES protection is desired, the FlashLock Pass Key must be set. The AES key is a content protection mechanism, whereas the FlashLock Pass Key is a device protection mechanism. When the AES key is programmed into the device, the device still needs the Pass Key to protect the FPGA and FlashROM contents and the security settings, including the AES key. Using the FlashLock Pass Key prevents modification of the design contents by means of simply programming the device with a different AES key.

AES Decryption and MAC Authentication

Low power flash devices have a built-in 128-bit AES decryption core, which decrypts the encrypted programming file and performs a MAC check that authenticates the file prior to programming.

MAC authenticates the entire programming data stream. After AES decryption, the MAC checks the data to make sure it is valid programming data for the device. This can be done while the device is still operating. If the MAC validates the file, the device will be erased and programmed. If the MAC fails to validate, then the device will continue to operate uninterrupted.

This will ensure the following:

- Correct decryption of the encrypted programming file
- Prevention of erroneous or corrupted data being programmed during the programming file transfer
- Correct bitstream passed to the device for decryption

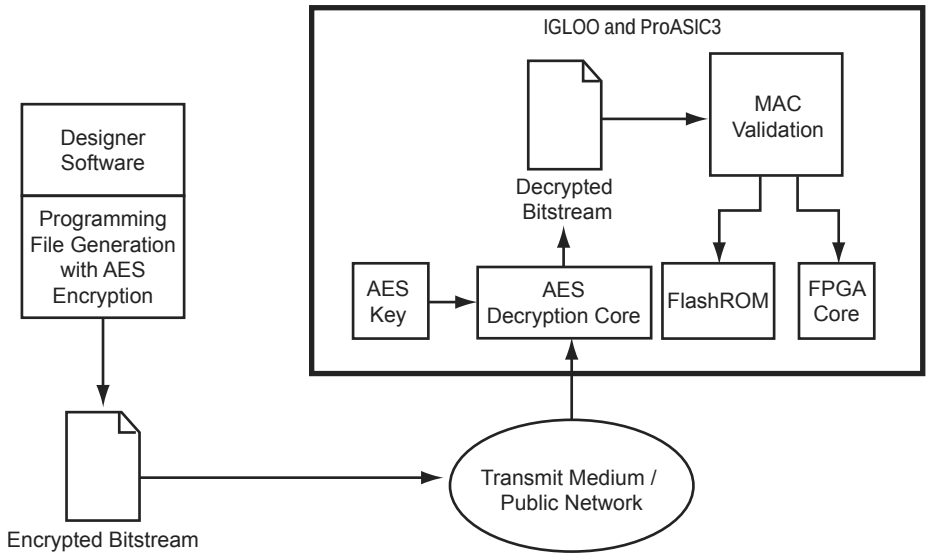

***Figure 12-4* • Example Application Scenario Using AES in IGLOO and ProASIC3 Devices**

1. *National Institute of Standards and Technology, "ADVANCED ENCRYPTION STANDARD (AES) Questions and Answers," 28 January 2002 (10 January 2005). See http://csrc.nist.gov/archive/aes/index1.html for more information.*

Note: The settings in this figure are used to show the generation of an AES-encrypted programming file for the FPGA array, FlashROM, and FB contents. One or all locations may be selected for encryption.

Figure 12-17 • **Settings to Program a Device Secured with FlashLock and using AES Encryption**

Choose the **High** security level to reprogram devices using both the FlashLock Pass Key and AES key protection (Figure 12-18 on page 321). Enter the AES key and click **Next**.

A device that has already been secured with FlashLock and has an AES key loaded must recognize the AES key to program the device and generate a valid bitstream in authentication. The FlashLock Key is only required to unlock the device and change the security settings.

This is what makes it possible to program in an untrusted environment. The AES key is protected inside the device by the FlashLock Key, so you can only program if you have the correct AES key. In fact, the AES key is not in the programming file either. It is the key used to encrypt the data in the file. The same key previously programmed with the FlashLock Key matches to decrypt the file.

An AES-encrypted file programmed to a device without FlashLock would not be secure, since without FlashLock to protect the AES key, someone could simply reprogram the AES key first, then program with any AES key desired or no AES key at all. This option is therefore not available in the software.

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

Introduction

Microsemi's low power flash devices are all in-system programmable. This document describes the general requirements for programming a device and specific requirements for the FlashPro4/3/3X programmers[1].

IGLOO, ProASIC3, SmartFusion, and Fusion devices offer a low power, single-chip, live-at-power-up solution with the ASIC advantages of security and low unit cost through nonvolatile flash technology. Each device contains 1 kbit of on-chip, user-accessible, nonvolatile FlashROM. The FlashROM can be used in diverse system applications such as Internet Protocol (IP) addressing, user system preference storage, device serialization, or subscription-based business models. IGLOO, ProASIC3, SmartFusion, and Fusion devices offer the best in-system programming (ISP) solution, FlashLock® security features, and AES-decryption-based ISP.

ISP Architecture

Low power flash devices support ISP via JTAG and require a single VPUMP voltage of 3.3 V during programming. In addition, programming via a microcontroller in a target system is also supported.

Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide.

Family-specific support:

- ProASIC3, ProASIC3E, SmartFusion, and Fusion devices support ISP.
- ProASIC3L devices operate using a 1.2 V core voltage; however, programming can be done only at 1.5 V. Voltage switching is required in-system to switch from a 1.2 V core to 1.5 V core for programming.
- IGLOO and IGLOOe V5 devices can be programmed in-system when the device is using a 1.5 V supply voltage to the FPGA core.
- IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only) or 1.5 V. IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V. Voltage switching is required in-system to switch from a 1.2 V supply (VCC,VCCI, and VJTAG) to 1.5 V for programming. The exception is that V2 devices can be programmed at 1.2 V VCC with FlashPro4.

IGLOO devices cannot be programmed in-system when the device is in Flash*Freeze mode. The device should exit Flash*Freeze mode and be in normal operation for programming to start. Programming operations in IGLOO devices can be achieved when the device is in normal operating mode and a 1.5 V core voltage is used.

JTAG 1532

IGLOO, ProASIC3, SmartFusion, and Fusion devices support the JTAG-based IEEE 1532 standard for ISP. To start JTAG operations, the IGLOO device must exit Flash*Freeze mode and be in normal operation before starting to send JTAG commands to the device. As part of this support, when a device is in an unprogrammed state, all user I/O pins are disabled. This is achieved by keeping the global IO_EN

1. *FlashPro4 replaced FlashPro3/3X in 2010 and is backward compatible with FlashPro3/3X as long as there is no connection to pin 4 on the JTAG header on the board. On FlashPro3/3X, there is no connection to pin 4 on the JTAG header; however, pin 4 is used for programming mode (Prog_Mode) on FlashPro4. When converting from FlashPro3/3X to FlashPro4, users should make sure that JTAG connectors on system boards do not have any connection to pin 4. FlashPro3X supports discrete TCK toggling that is needed to support non-JTAG compliant devices in the chain. This feature is included in FlashPro4.*

Microsemi's Flash Families Support Voltage Switching Circuit

The flash FPGAs listed in Table 14-1 support the voltage switching circuit feature and the functions described in this document.

***Table 14-1* • Flash-Based FPGAs Supporting Voltage Switching Circuit**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	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

*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 14-1. 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 14-1. 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*.

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

Silicon Testing and Debugging

In many applications, the design needs to be tested, debugged, and verified on real silicon or in the final embedded application. To debug and test the functionality of designs, users may need to monitor some internal logic (or nets) during device operation. The approach of adding design test pins to monitor the critical internal signals has many disadvantages, such as limiting the number of user I/Os. Furthermore, adding external I/Os for test purposes may require additional or dedicated board area for testing and debugging.

The UJTAG tiles of low power flash devices offer a flexible and cost-effective solution for silicon test and debug applications. In this solution, the signals under test are shifted out to the TDO pin of the TAP Controller. The main advantage is that all the test signals are monitored from the TDO pin; no pins or additional board-level resources are required. Figure 17-6 illustrates this technique. Multiple test nets are brought into an internal MUX architecture. The selection of the MUX is done using the contents of the TAP Controller instruction register, where individual instructions (values from 16 to 127) correspond to different signals under test. The selected test signal can be synchronized with the rising or falling edge of TCK (optional) and sent out to UTDO to drive the TDO output of JTAG.

For flash devices, TDO (the output) is configured as low slew and the highest drive strength available in the technology and/or device. Here are some examples:

1. If the device is A3P1000 and VCCI is 3.3 V, TDO will be configured as LVTTL 3.3 V output, 24 mA, low slew.
2. If the device is AGLN020 and VCCI is 1.8 V, TDO will be configured as LVCMOS 1.8 V output, 4 mA, low slew.
3. If the device is AGLE300 and VCCI is 2.5 V, TDO will be configured as LVCMOS 2.5 V output, 24 mA, low slew.

The test and debug procedure is not limited to the example in Figure 17-5 on page 369. Users can customize the debug and test interface to make it appropriate for their applications. For example, multiple test signals can be registered and then sent out through UTDO, each at a different edge of TCK. In other words, *n* signals are sampled with an F_{TCK} / *n* sampling rate. The bandwidth of the information sent out to TDO is always proportional to the frequency of TCK.

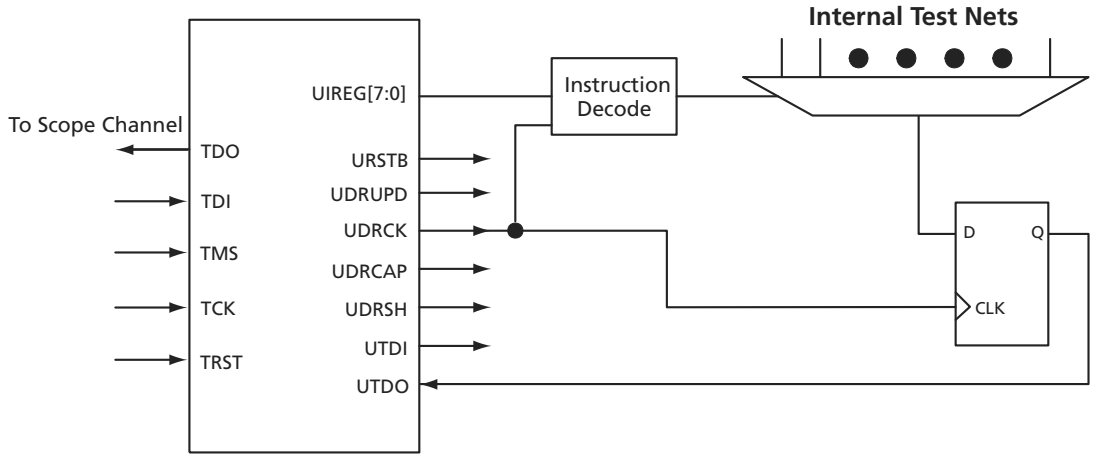

Figure 17-6 • **UJTAG Usage Example in Test and Debug Applications**