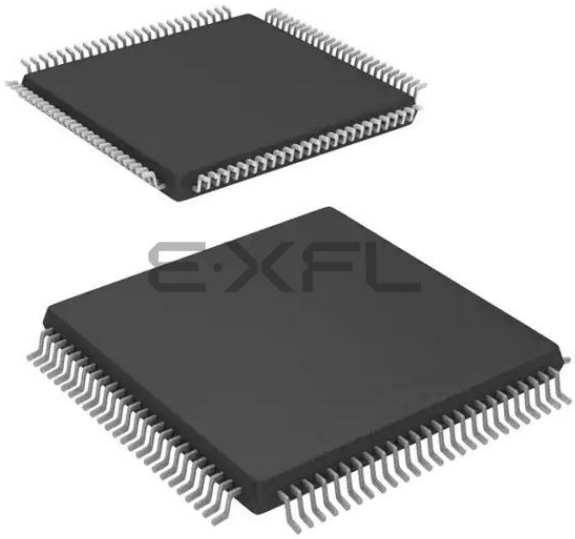


Welcome to E-XFL.COM

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	71
Number of Gates	125000
Voltage - Supply	1.425V ~ 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/microchip-technology/a3pn125-zvq100i

Routing Architecture

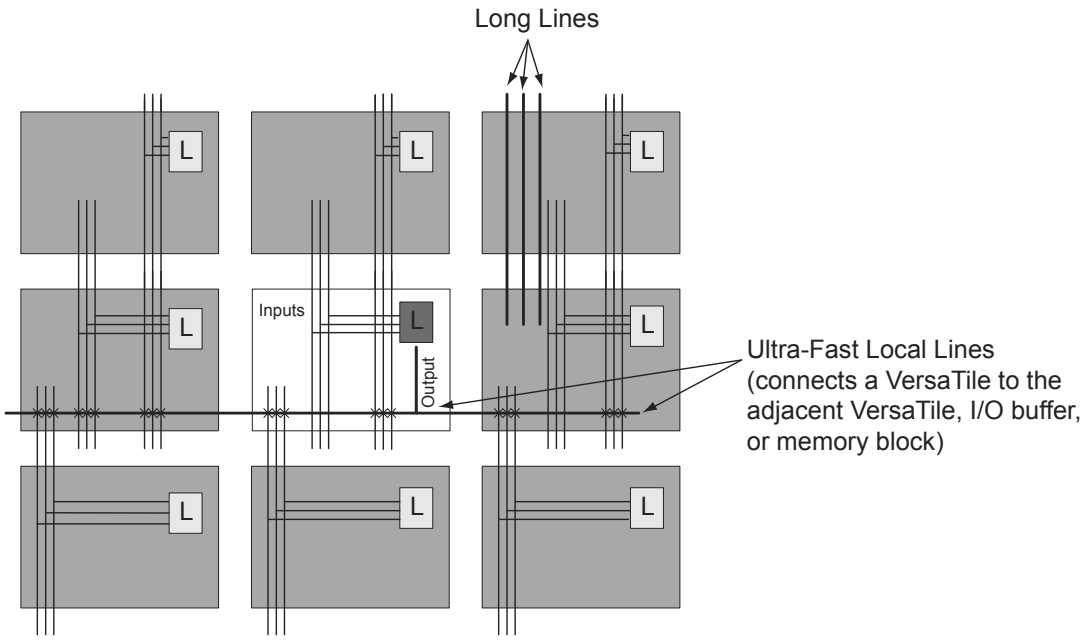
The routing structure of low power flash devices is designed to provide high performance through a flexible four-level hierarchy of routing resources: ultra-fast local resources; efficient long-line resources; high-speed, very-long-line resources; and the high-performance VersaNet networks.

The ultra-fast local resources are dedicated lines that allow the output of each VersaTile to connect directly to every input of the eight surrounding VersaTiles (Figure 1-10). The exception to this is that the SET/CLR input of a VersaTile configured as a D-flip-flop is driven only by the VersaTile global network.

The efficient long-line resources provide routing for longer distances and higher-fanout connections. These resources vary in length (spanning one, two, or four VersaTiles), run both vertically and horizontally, and cover the entire device (Figure 1-11 on page 19). Each VersaTile can drive signals onto the efficient long-line resources, which can access every input of every VersaTile. Routing software automatically inserts active buffers to limit loading effects.

The high-speed, very-long-line resources, which span the entire device with minimal delay, are used to route very long or high-fanout nets: length ±12 VersaTiles in the vertical direction and length ±16 in the horizontal direction from a given core VersaTile (Figure 1-12 on page 19). Very long lines in low power flash devices have been enhanced over those in previous ProASIC families. This provides a significant performance boost for long-reach signals.

The high-performance VersaNet global networks are low-skew, high-fanout nets that are accessible from external pins or internal logic. These nets are typically used to distribute clocks, resets, and other high-fanout nets requiring minimum skew. The VersaNet networks are implemented as clock trees, and signals can be introduced at any junction. These can be employed hierarchically, with signals accessing every input of every VersaTile. For more details on VersaNets, refer to the "Global Resources in Low Power Flash Devices" section on page 31.



Note: Input to the core cell for the D-flip-flop set and reset is only available via the VersaNet global network connection.

Figure 1-10 • **Ultra-Fast Local Lines Connected to the Eight Nearest Neighbors**

List of Changes

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

Date	Changes	Page
June 2011	Table 2-1 • ProASIC3/E/nano Low Power Modes Summary and the "Shutdown Mode" section were revised to remove reference to ProASIC3/E devices (SAR 24526).	22, 27
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
v1.2 (August 2008)	References to ProASIC3 nano devices were added to the document where appropriate.	N/A
	VJTAG and VPUMP were noted as "Off" in the Sleep Mode section of Table 2-1 • ProASIC3/E/nano Low Power Modes Summary.	22
	The "Sleep Mode" section, including Table 2-3 • Sleep Mode—Power Supply Requirements for ProASIC3/E/nano Devices, was revised to state that VJTAG and VPUMP are powered off during Sleep mode.	25
	The text above Table 2-4 • A3P250 Current Draw in Sleep Mode and Table 2-5 • A3PE600 Current Draw in Sleep Mode was revised to state "VCC = VJTAG = VPUMP = GND."	26
	Figure 2-6 • Controlling Power On/Off State Using Microprocessor and Power FET and Figure 2-7 • Controlling Power On/Off State Using Microprocessor and Voltage Regulator were revised to show shutdown of VJTAG and VPUMP during Sleep mode.	27, 28
v1.1 (March 2008)	The part number for this document was changed from 51700094-002-0 to 51700094-003-1.	N/A
v1.0 (January 2008)	The Power Supplies / Clock Status description was updated for Static (Idle) in Table 2-1 • ProASIC3/E/nano Low Power Modes Summary.	22
	Programming information was updated in the "User Low Static (Idle) Mode" section.	23
51900138-2/10.06	The "User Low Static (Idle) Mode" section was updated to include information about allowing programming in the ULSICC mode.	23
	Figure 2-2 • User Low Static (Idle) Mode Application—Internal Control Signal was updated.	24
	Figure 2-3 • User Low Static (Idle) Mode Application—External Control Signal was updated.	25
51900138-1/6.06	In Table 2-4 • A3P250 Current Draw in Sleep Mode, "VCCI = 1.5 V" was changed from 3.6158 to 3.62.	26
	In Table 2-5 • A3PE600 Current Draw in Sleep Mode, "VCCI = 2.5 V" was changed from 5.6875 to 3.69.	26

***Table 3-3* • Quadrant Global Pin Name**

I/O Type	Beginning of I/O Name	Notes
Single-Ended	GAAO/IOuxwByVz GAA1/IOuxwByVz GAA2/IOuxwByVz	Only one of the I/Os can be directly connected to a quadrant global at a time
	GABO/IOuxwByVz GAB1/IOuxwByVz GAB2/IOuxwByVz	Only one of the I/Os can be directly connected to a quadrant global at a time.
	GAC0/IOuxwByVz GAC1/IOuxwByVz GAC2/IOuxwByVz	Only one of the I/Os can be directly connected to a quadrant global at a time.
	GBAO/IOuxwByVz GBA1/IOuxwByVz GBA2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GBBO/IOuxwByVz GBB1/IOuxwByVz GBB2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GBC0/IOuxwByVz GBC1/IOuxwByVz GBC2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GDAO/IOuxwByVz GDA1/IOuxwByVz GDA2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GDBO/IOuxwByVz GDB1/IOuxwByVz GDB2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GDC0/IOuxwByVz GDC1/IOuxwByVz GDC2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GEAO/IOuxwByVz GEA1/IOuxwByVz GEA2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GEBO/IOuxwByVz GEB1/IOuxwByVz GEB2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.
	GEC0/IOuxwByVz GEC1/IOuxwByVz GEC2/IOuxwByVz	Only one of the I/Os can be directly connected to a global at a time.

Note: Only one of the I/Os can be directly connected to a quadrant at a time.

Clock Aggregation Architecture

This clock aggregation feature allows a balanced clock tree, which improves clock skew. The physical regions for clock aggregation are defined from left to right and shift by one spine. For chip global networks, there are three types of clock aggregation available, as shown in Figure 3-10:

- Long lines that can drive up to four adjacent spines (A)
- Long lines that can drive up to two adjacent spines (B)
- Long lines that can drive one spine (C)

There are three types of clock aggregation available for the quadrant spines, as shown in Figure 3-10:

- I/Os or local resources that can drive up to four adjacent spines
- I/Os or local resources that can drive up to two adjacent spines
- I/Os or local resources that can drive one spine

As an example, A3PE600 and AFS600 devices have twelve spine locations: T1, T2, T3, T4, T5, T6, B1, B2, B3, B4, B5, and B6. Table 3-7 shows the clock aggregation you can have in A3PE600 and AFS600.

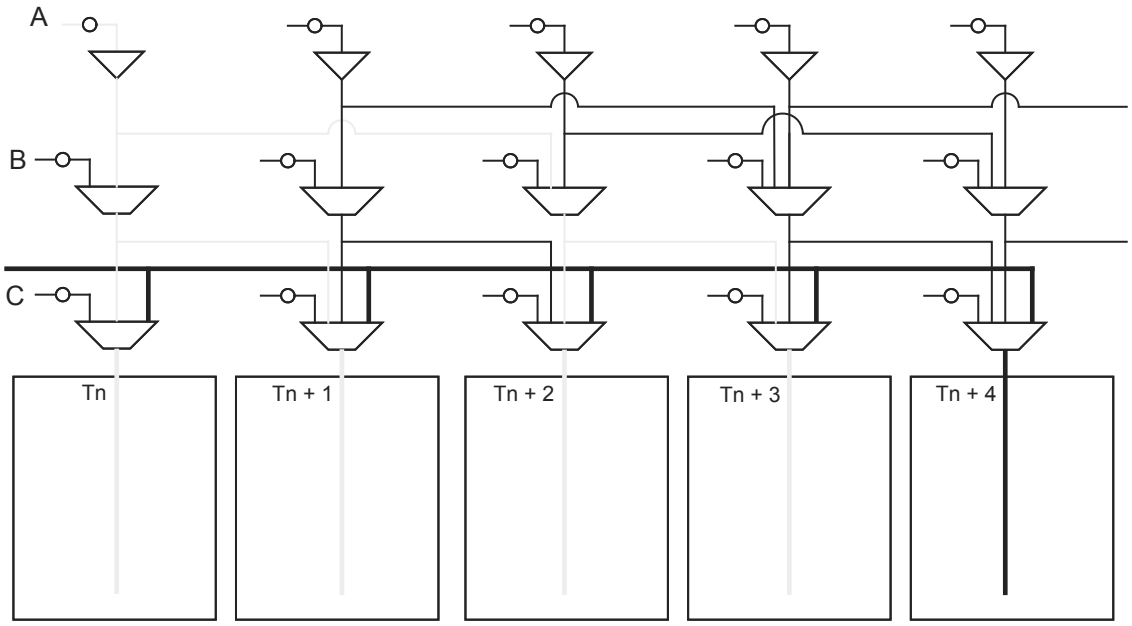


***Figure 3-10* • Four Spines Aggregation**

***Table 3-7* • Spine Aggregation in A3PE600 or AFS600**

Clock Aggregation	Spine
1 spine	T1, T2, T3, T4, T5, T6, B1, B2, B3, B4, B5, B6
2 spines	T1:T2, T2:T3, T3:T4, T4:T5, T5:T6, B1:B2, B2:B3, B3:B4, B4:B5, B5:B6
4 spines	B1:B4, B2:B5, B3:B6, T1:T4, T2:T5, T3:T6

The clock aggregation for the quadrant spines can cross over from the left to right quadrant, but not from top to bottom. The quadrant spine assignment T1:T4 is legal, but the quadrant spine assignment T1:B1 is not legal. Note that this clock aggregation is hardwired. You can always assign signals to spine T1 and B2 by instantiating a buffer, but this may add skew in the signal.

Date	Changes	Page
v1.1 (March 2008)	The "Global Architecture" section was updated to include the IGLOO PLUS family. The bullet was revised to include that the west CCC does not contain a PLL core in 15 k and 30 k devices. Instances of "A3P030 and AGL030 devices" were replaced with "15 k and 30 k gate devices."	31
v1.1 (continued)	Table 3-1 • Flash-Based FPGAs and the accompanying text was updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	32
	The "VersaNet Global Network Distribution" section, "Spine Architecture" section, the note in Figure 3-1 • Overview of VersaNet Global Network and Device Architecture, and the note in Figure 3-3 • Simplified VersaNet Global Network (60 k gates and above) were updated to include mention of 15 k gate devices.	33, 34
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to add the A3P015 device, and to revise the values for clock trees, globals/spines per tree, and globals/spines per device for the A3P030 and AGL030 devices.	41
	Table 3-5 • Globals/Spines/Rows for IGLOO PLUS Devices is new.	42
	CLKBUF_LVCMOS12 was added to Table 3-9 • I/O Standards within CLKBUF.	47
	The "User's Guides" section was updated to include the three different I/O Structures chapters for ProASIC3 and IGLOO device families.	58
v1.0 (January 2008)	Figure 3-3 • Simplified VersaNet Global Network (60 k gates and above) was updated.	34
	The "Naming of Global I/Os" section was updated.	35
	The "Using Global Macros in Synplicity" section was updated.	50
	The "Global Promotion and Demotion Using PDC" section was updated.	51
	The "Designer Flow for Global Assignment" section was updated.	53
	The "Simple Design Example" section was updated.	55
51900087-0/1.05 (January 2005)	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated.	41

5 – FlashROM in Microsemi's Low Power Flash Devices

Introduction

The Fusion, IGLOO, and ProASIC3 families of low power flash-based devices have a dedicated nonvolatile FlashROM memory of 1,024 bits, which provides a unique feature in the FPGA market. The FlashROM can be read, modified, and written using the JTAG (or UJTAG) interface. It can be read but not modified from the FPGA core. Only low power flash devices contain on-chip user nonvolatile memory (NVM).

Architecture of User Nonvolatile FlashROM

Low power flash devices have 1 kbit of user-accessible nonvolatile flash memory on-chip that can be read from the FPGA core fabric. The FlashROM is arranged in eight banks of 128 bits (16 bytes) during programming. The 128 bits in each bank are addressable as 16 bytes during the read-back of the FlashROM from the FPGA core. Figure 5-1 shows the FlashROM logical structure.

The FlashROM can only be programmed via the IEEE 1532 JTAG port. It cannot be programmed directly from the FPGA core. When programming, each of the eight 128-bit banks can be selectively reprogrammed. The FlashROM can only be reprogrammed on a bank boundary. Programming involves an automatic, on-chip bank erase prior to reprogramming the bank. The FlashROM supports synchronous read. The address is latched on the rising edge of the clock, and the new output data is stable after the falling edge of the same clock cycle. For more information, refer to the timing diagrams in the DC and Switching Characteristics chapter of the appropriate datasheet. The FlashROM can be read on byte boundaries. The upper three bits of the FlashROM address from the FPGA core define the bank being accessed. The lower four bits of the FlashROM address from the FPGA core define which of the 16 bytes in the bank is being accessed.

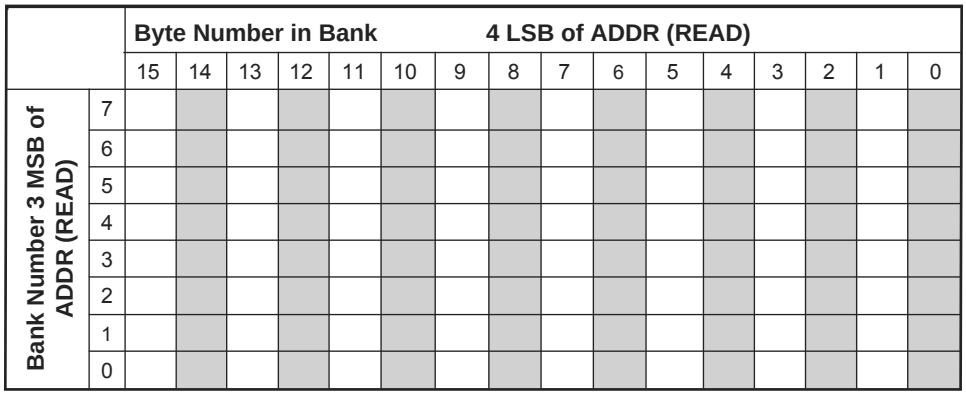


***Figure 5-1* • FlashROM Architecture**

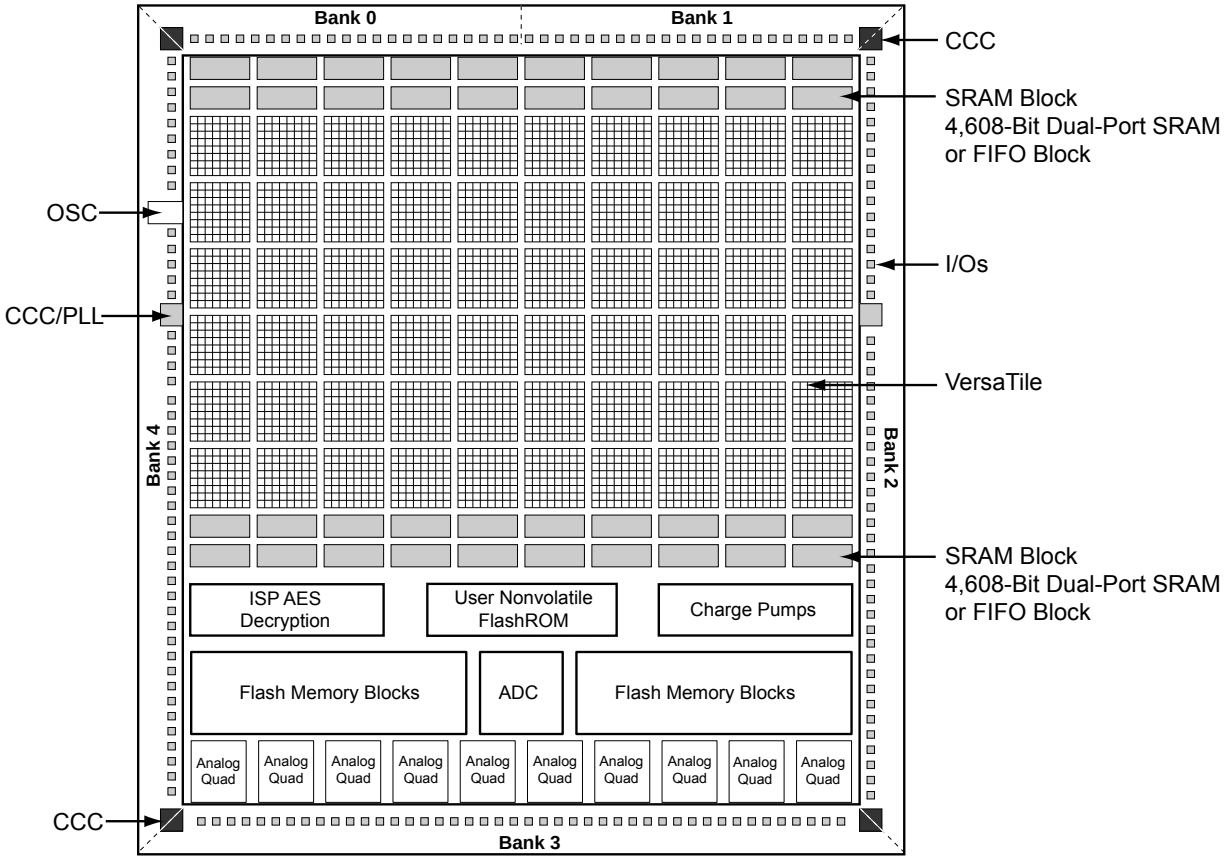


***Figure 5-2* • Fusion Device Architecture Overview (AFS600)**

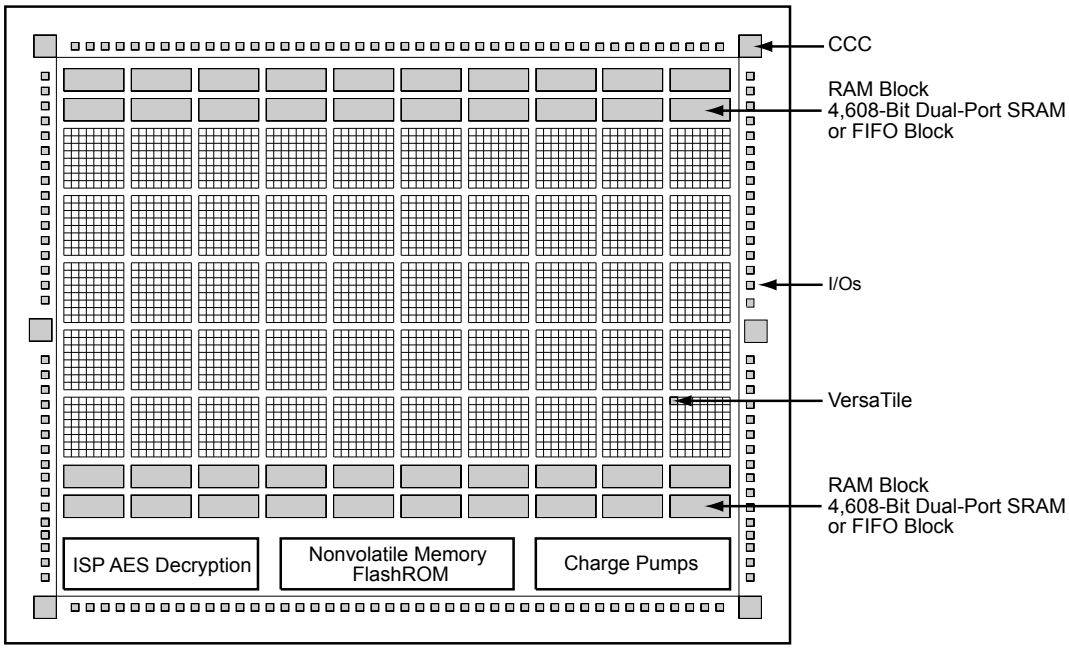


***Figure 5-3* • ProASIC3 and IGLOO Device Architecture**

DEVICE_INFO displays the FlashROM content, serial number, Design Name, and checksum, as shown below:

```
EXPORT IDCODE[32] = 123261CF
EXPORT SILSIG[32] = 00000000
User information :
CHECKSUM: 61A0
Design Name:        TOP
Programming Method: STAPL
Algorithm Version: 1
Programmer: UNKNOWN
=======================================
FlashROM Information :
EXPORT Region_7_0[128] = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
=======================================
Security Setting :
Encrypted FlashROM Programming Enabled.
Encrypted FPGA Array Programming Enabled.
=======================================
```

The Libero SoC file manager recognizes the UFC and MEM files and displays them in the appropriate view. Libero SoC also recognizes the multiple programming files if you choose the option to generate multiple files for multiple FlashROM contents in Designer. These features enable a user-friendly flow for the FlashROM generation and programming in Libero SoC.

Custom Serialization Using FlashROM

You can use FlashROM for device serialization or inventory control by using the Auto Inc region or Read From File region. FlashPoint will automatically generate the serial number sequence for the Auto Inc region with the **Start Value**, **Max Value**, and **Step Value** provided. If you have a unique serial number generation scheme that you prefer, the Read From File region allows you to import the file with your serial number scheme programmed into the region. See the *FlashPro User's Guide* for custom serialization file format information.

The following steps describe how to perform device serialization or inventory control using FlashROM:

1. Generate FlashROM using SmartGen. From the Properties section in the FlashROM Settings dialog box, select the **Auto Inc** or **Read From File** region. For the Auto Inc region, specify the desired step value. You will not be able to modify this value in the FlashPoint software.
2. Go through the regular design flow and finish place-and-route.
3. Select **Programming File in Designer** and open **Generate Programming File** (Figure 5-12 on page 128).
4. Click **Program FlashROM**, browse to the UFC file, and click **Next**. The FlashROM Settings window appears, as shown in Figure 5-13 on page 128.
5. Select the FlashROM page you want to program and the data value for the configured regions. The STAPL file generated will contain only the data that targets the selected FlashROM page.
6. Modify properties for the serialization.
   – For the Auto Inc region, specify the **Start** and **Max** values.
   – For the Read From File region, select the file name of the custom serialization file.
7. Select the FlashROM programming file type you want to generate from the two options below:
   – Single programming file for all devices: generates one programming file with all FlashROM values.
   – One programming file per device: generates a separate programming file for each FlashROM value.
8. Enter the number of devices you want to program and generate the required programming file.
9. Open the programming software and load the programming file. The programming software, FlashPro3 and Silicon Sculptor II, supports the device serialization feature. If, for some reason, the device fails to program a part during serialization, the software allows you to reuse or skip the serial data. Refer to the *FlashPro User's Guide* for details.

Conclusion

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

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

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

Related Documents

User's Guides

FlashPro User's Guide

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

List of Changes

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

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 5-1 • Flash-Based FPGAs.	118
v1.3 (October 2008)	The "FlashROM Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	118
	Figure 5-2 • Fusion Device Architecture Overview (AFS600) was replaced. Figure 5-5 • Programming FlashROM Using AES was revised to change "Fusion" to "Flash Device."	119, 121
	The *FlashPoint User's Guide* was removed from the "User's Guides" section, as its content is now part of the *FlashPro User's Guide*.	130
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 5-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	118
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	N/A

Low Power Flash Device I/O Support

The low power flash families 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 nano	Lowest power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
ProASIC3	ProASIC3 nano	Lowest cost 1.5 V FPGAs with balanced performance

*Note: *The device name links 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*.

Example 2 (low–medium speed, medium current):

Rtx_out_high = Rtx_out_low = 10 Ω

R1 = 220 Ω (±5%), P(r1)min = 0.018 Ω

R2 = 390 Ω (±5%), P(r2)min = 0.032 Ω

Imax_tx = 5.5 V / (220 × 0.95 + 390 × 0.95 + 10) = 9.17 mA

t_{RISE} = t_{FALL} = 4 ns at C_pad_load = 10 pF (includes up to 25% safety margin)

t_{RISE} = t_{FALL} = 20 ns at C_pad_load = 50 pF (includes up to 25% safety margin)

Other values of resistors are also allowed as long as the resistors are sized appropriately to limit the voltage at the receiving end to 2.5 V < Vin (rx) < 3.6 V when the transmitter sends a logic 1. This range of Vin_dc(rx) must be assured for any combination of transmitter supply (5 V ± 0.5 V), transmitter output resistance, and board resistor tolerances.

Temporary overshoots are allowed according to the overshoot and undershoot table in the datasheet.

Solution 1

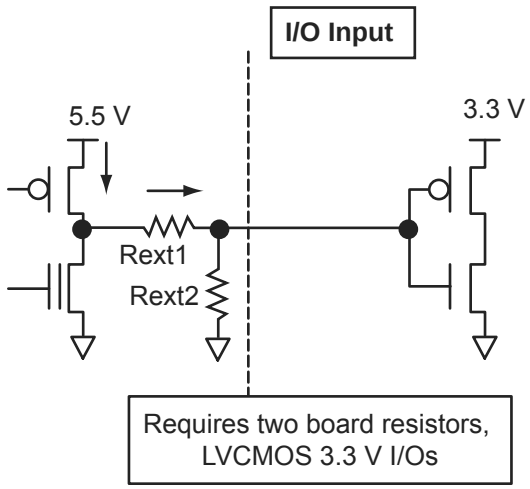


Figure 7-5 **• Solution 1**

Flash FPGAs I/O Support

The flash FPGAs listed in Table 8-1 support I/Os and the functions described in this document.

***Table 8-1* • 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 8-1. 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 8-1. 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*.

Implementing I/Os in Microsemi Software

Microsemi Libero SoC software is integrated with design entry tools such as the SmartGen macro builder, the ViewDraw schematic entry tool, and an HDL editor. It is also integrated with the synthesis and Designer tools. In this section, all necessary steps to implement the I/Os are discussed.

Design Entry

There are three ways to implement I/Os in a design:

1. Use the SmartGen macro builder to configure I/Os by generating specific I/O library macros and then instantiating them in top-level code. This is especially useful when creating I/O bus structures.
2. Use an I/O buffer cell in a schematic design.
3. Manually instantiate specific I/O macros in the top-level code.

If technology-specific macros, such as INBUF_LVCMOS33 and OUTBUF_PCI, are used in the HDL code or schematic, the user will not be able to change the I/O standard later on in Designer. If generic I/O macros are used, such as INBUF, OUTBUF, TRIBUF, CLKBUF, and BIBUF, the user can change the I/O standard using the Designer I/O Attribute Editor tool.

Using SmartGen for I/O Configuration

The SmartGen tool in Libero SoC provides a GUI-based method of configuring the I/O attributes. The user can select certain I/O attributes while configuring the I/O macro in SmartGen. The steps to configure an I/O macro with specific I/O attributes are as follows:

1. Open Libero SoC.
2. On the left-hand side of the Catalog View, select **I/O**, as shown in Figure 8-2.

Figure 8-2 **• SmartGen Catalog**

Output Buffers

There are two variations: Regular and Special.

If the **Regular** variation is selected, only the Width (1 to 128) needs to be entered. The default value for Width is 1.

The **Special** variation has Width, Technology, Output Drive, and Slew Rate options.

Bidirectional Buffers

There are two variations: Regular and Special.

The **Regular** variation has Enable Polarity (Active High, Active Low) in addition to the Width option.

The **Special** variation has Width, Technology, Output Drive, Slew Rate, and Resistor Pull-Up/-Down options.

Tristate Buffers

Same as Bidirectional Buffers.

DDR

There are eight variations: DDR with Regular Input Buffers, Special Input Buffers, Regular Output Buffers, Special Output Buffers, Regular Tristate Buffers, Special Tristate Buffers, Regular Bidirectional Buffers, and Special Bidirectional Buffers.

These variations resemble the options of the previous I/O macro. For example, the Special Input Buffers variation has Width, Technology, Voltage Level, and Resistor Pull-Up/-Down options. DDR is not available on IGLOO PLUS devices.

4. Once the desired configuration is selected, click the **Generate** button. The Generate Core window opens (Figure 8-4).
5. Enter a name for the macro. Click **OK**. The core will be generated and saved to the appropriate location within the project files (Figure 8-5 on page 191).

Figure 8-4 • **Generate Core Window**

6. Instantiate the I/O macro in the top-level code.

   The user must instantiate the DDR_REG or DDR_OUT macro in the design. Use SmartGen to generate both these macros and then instantiate them in your top level. To combine the DDR macros with the I/O, the following rules must be met:

– The I/O standard of technology-specific I/O macros cannot be changed in the I/O Attribute Editor (see Figure 8-6).
– The user MUST instantiate differential I/O macros (LVDS/LVPECL) in the design. This is the only way to use these standards in the design (IGLOO nano and ProASIC3 nano devices do not support differential inputs).
– To implement the DDR I/O function, the user must instantiate a DDR_REG or DDR_OUT macro. This is the only way to use a DDR macro in the design.

***Figure 8-6* • Assigning a Different I/O Standard to the Generic I/O Macro**

Performing Place-and-Route on the Design

The netlist created by the synthesis tool should now be imported into Designer and compiled. During Compile, the user can specify the I/O placement and attributes by importing the PDC file. The user can also specify the I/O placement and attributes using ChipPlanner and the I/O Attribute Editor under MVN.

Defining I/O Assignments in the PDC File

A PDC file is a Tcl script file specifying physical constraints. This file can be imported to and exported from Designer.

Table 8-3 shows I/O assignment constraints supported in the PDC file.

***Table 8-3* • PDC I/O Constraints**

Command	Action	Example	Comment
I/O Banks Setting Constraints			

set_iobank	Sets the I/O supply voltage, V_{CCI}, and the input reference voltage, V_{REF}, for the specified I/O bank.	`set_iobank bankname [-vcci vcci_voltage] [-vref vref_voltage]`
		
		`set_iobank Bank7 -vcci 1.50 -vref 0.75`
			Must use in case of mixed I/O voltage (V_{CCI}) design

set_vref	Assigns a V_{REF} pin to a bank.	`set_vref -bank [bankname] [pinnum]`
		
		`set_vref -bank Bank0 685 704 723 742 761`
			Must use if voltage-referenced I/Os are used

set_vref_defaults	Sets the default V_{REF} pins for the specified bank. This command is ignored if the bank does not need a V_{REF} pin.	`set_vref_defaults bankname`
		
		`set_vref_defaults bank2`	

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

Programming File Header Definition

In each STAPL programming file generated, there will be information about how the AES key and FlashLock Pass Key are configured. Table 11-8 shows the header definitions in STAPL programming files for different security levels.

***Table 11-8* • STAPL Programming File Header Definitions by Security Level**

Security Level	STAPL File Header Definition
No security (no FlashLock Pass Key or AES key)	`NOTE "SECURITY" "Disable";`
FlashLock Pass Key with no AES key	`NOTE "SECURITY" "KEYED ";`
FlashLock Pass Key with AES key	`NOTE "SECURITY" "KEYED ENCRYPT ";`
Permanent Security Settings option enabled	`NOTE "SECURITY" "PERMLOCK ENCRYPT ";`
AES-encrypted FPGA array (for programming updates)	`NOTE "SECURITY" "ENCRYPT CORE ";`
AES-encrypted FlashROM (for programming updates)	`NOTE "SECURITY" "ENCRYPT FROM ";`
AES-encrypted FPGA array and FlashROM (for programming updates)	`NOTE "SECURITY" "ENCRYPT FROM CORE ";`

Example File Headers

STAPL Files Generated with FlashLock Key and AES Key Containing Key Information

- FlashLock Key / AES key indicated in STAPL file header definition
- Intended ONLY for secured/trusted environment programming applications

```
============================================
NOTE "CREATOR" "Designer Version: 6.1.1.108";
NOTE "DEVICE" "A3PE600";
NOTE "PACKAGE" "208 PQFP";
NOTE "DATE" "2005/04/08";
NOTE "STAPL_VERSION" "JESD71";
NOTE "IDCODE" "$123261CF";
NOTE "DESIGN" "counter32";
NOTE "CHECKSUM" "$EDB9";
NOTE "SAVE_DATA" "FRomStream";
NOTE "SECURITY" "KEYED ENCRYPT ";
NOTE "ALG_VERSION" "1";
NOTE "MAX_FREQ" "20000000";
NOTE "SILSIG" "$00000000";
NOTE "PASS_KEY" "$00123456789012345678901234567890";
NOTE "AES_KEY" "$ABCDEFABCDEFABCDEFABCDEFABCDEFAB";
============================================
```

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 14-5 on page 289).

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

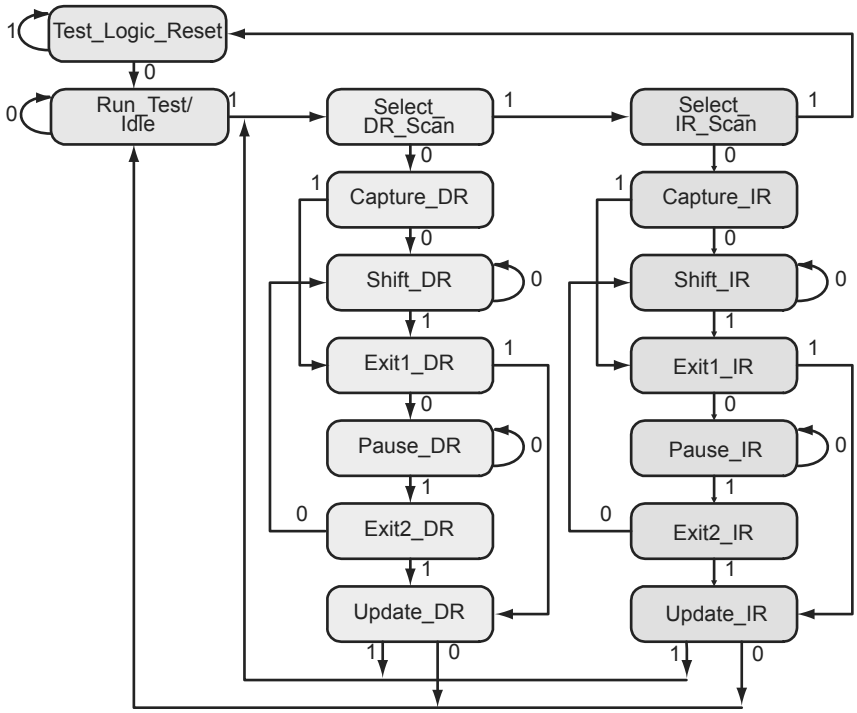


Figure 16-4 • **TAP Controller State Diagram**

UJTAG Port Usage

UIREG[7:0] hold the contents of the JTAG instruction register. The UIREG vector value is updated when the TAP Controller state machine enters the Update_IR state. Instructions 16 to 127 are user-defined and can be employed to encode multiple applications and commands within an application. Loading new instructions into the UIREG vector requires users to send appropriate logic to TMS to put the TAP Controller in a full IR cycle starting from the Select IR_Scan state and ending with the Update_IR state.

UTDI, UTDO, and UDRCK are directly connected to the JTAG TDI, TDO, and TCK ports, respectively. The TDI input can be used to provide either data (TAP Controller in the Shift_DR state) or the new contents of the instruction register (TAP Controller in the Shift_IR state).

UDRSH, UDRUPD, and UDRCAP are HIGH when the TAP Controller state machine is in the Shift_DR, Update_DR, and Capture_DR states, respectively. Therefore, they act as flags to indicate the stages of the data shift process. These flags are useful for applications in which blocks of data are shifted into the design from JTAG pins. For example, an active UDRSH can indicate that UTDI contains the data bitstream, and UDRUPD is a candidate for the end-of-data-stream flag.

As mentioned earlier, users should not connect the TDI, TDO, TCK, TMS, and TRST ports of the UJTAG macro to any port or net of the design netlist. The Designer software will automatically handle the port connection.