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

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

Applications of Embedded - FPGAs

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

Details

Product Status	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-z1vqg100i

Spine Architecture

The low power flash device architecture allows the VersaNet global networks to be segmented. Each of these networks contains spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside its region. The nine spines available in a vertical column reside in global networks with two separate regions of scope: the quadrant global network, which has three spines, and the chip (main) global network, which has six spines. Note that the number of quadrant globals and globals/spines per tree varies depending on the specific device. Refer to Table 3-4 for the clocking resources available for each device. The spines are the vertical branches of the global network tree, shown in Figure 3-3 on page 34. Each spine in a vertical column of a chip (main) global network is further divided into two spine segments of equal lengths: one in the top and one in the bottom half of the die (except in 10 k through 30 k gate devices).

Top and bottom spine segments radiating from the center of a device have the same height. However, just as in the ProASIC$^{\underline{PLUS}®}$ family, signals assigned only to the top and bottom spine cannot access the middle two rows of the die. The spines for quadrant clock networks do not cross the middle of the die and cannot access the middle two rows of the architecture.

Each spine and its associated ribs cover a certain area of the device (the "scope" of the spine; see Figure 3-3 on page 34). Each spine is accessed by the dedicated global network MUX tree architecture, which defines how a particular spine is driven—either by the signal on the global network from a CCC, for example, or by another net defined by the user. Details of the chip (main) global network spine-selection MUX are presented in Figure 3-8 on page 44. The spine drivers for each spine are located in the middle of the die.

Quadrant spines can be driven from user I/Os or an internal signal from the north and south sides of the die. The ability to drive spines in the quadrant global networks can have a significant effect on system performance for high-fanout inputs to a design. Access to the top quadrant spine regions is from the top of the die, and access to the bottom quadrant spine regions is from the bottom of the die. The A3PE3000 device has 28 clock trees and each tree has nine spines; this flexible global network architecture enables users to map up to 252 different internal/external clocks in an A3PE3000 device.

Table 3-4 • **Globals/Spines/Rows for IGLOO and ProASIC3 Devices**

ProASIC3/ ProASIC3L Devices	IGLOO Devices	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
A3PN010	AGLN010	4	0	1	0	0	260	260	4
A3PN015	AGLN015	4	0	1	0	0	384	384	6
A3PN020	AGLN020	4	0	1	0	0	520	520	6
A3PN060	AGLN060	6	12	4	9	36	384	1,536	12
A3PN125	AGLN125	6	12	8	9	72	384	3,072	12
A3PN250	AGLN250	6	12	8	9	72	768	6,144	24
A3P015	AGL015	6	0	1	9	9	384	384	12
A3P030	AGL030	6	0	2	9	18	384	768	12
A3P060	AGL060	6	12	4	9	36	384	1,536	12
A3P125	AGL125	6	12	8	9	72	384	3,072	12
A3P250/L	AGL250	6	12	8	9	72	768	6,144	24
A3P400	AGL400	6	12	12	9	108	768	9,216	24
A3P600/L	AGL600	6	12	12	9	108	1,152	13,824	36
A3P1000/L	AGL1000	6	12	16	9	144	1,536	24,576	48
A3PE600/L	AGLE600	6	12	12	9	108	1,120	13,440	35
A3PE1500		6	12	20	9	180	1,888	37,760	59
A3PE3000/L	AGLE3000	6	12	28	9	252	2,656	74,368	83
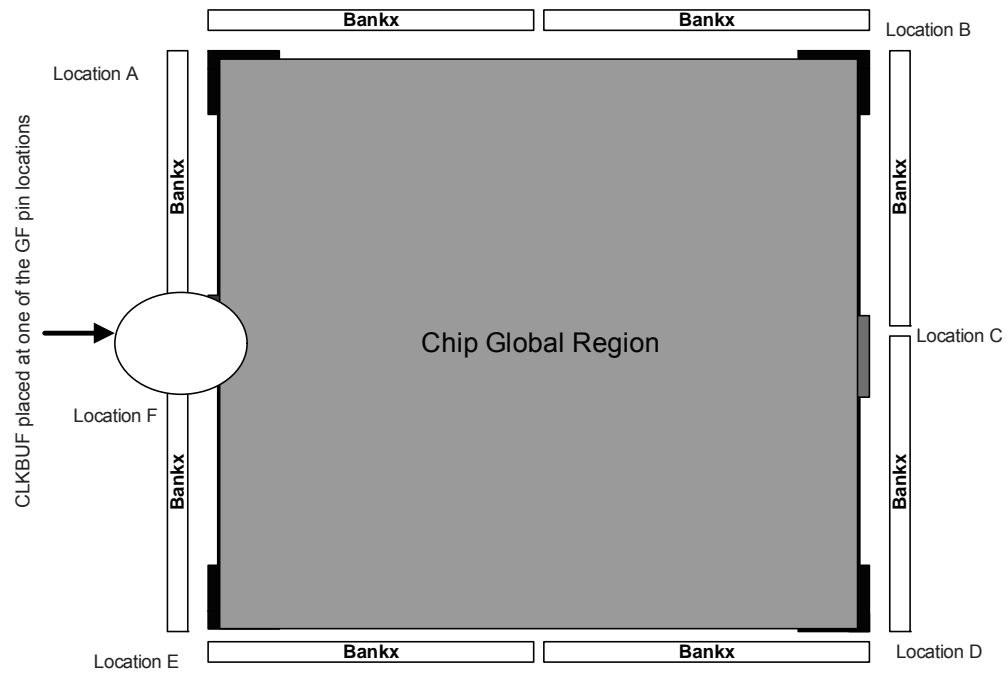
Figure 3-12 • **Chip Global Region**
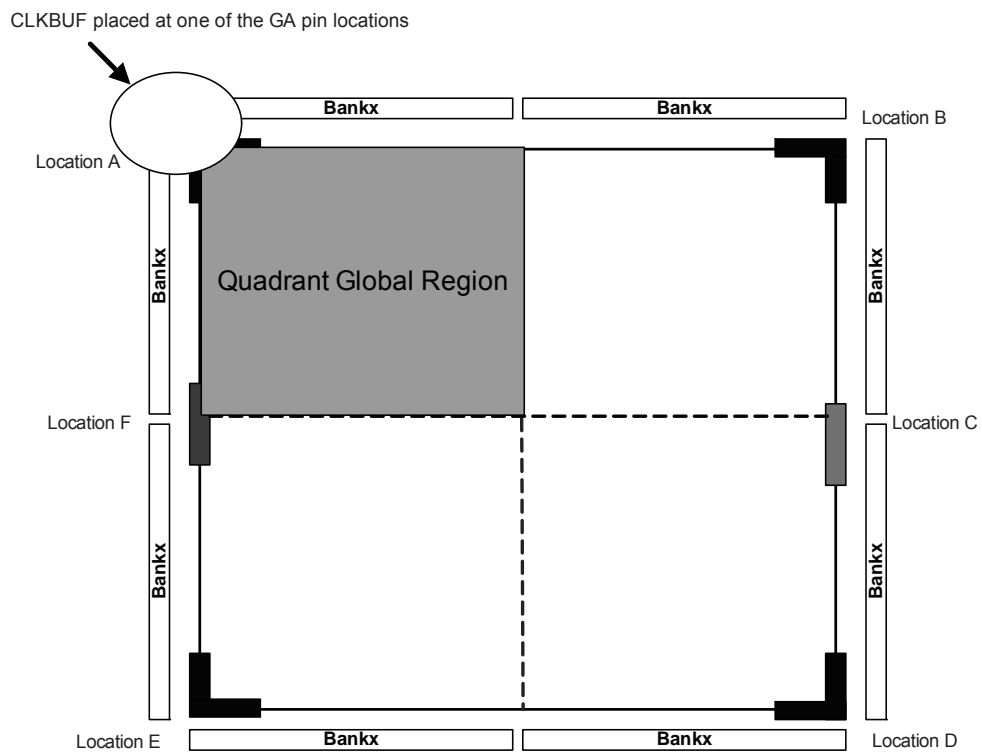
Figure 3-13 • **Quadrant Global Region**

YB and YC are identical to GLB and GLC, respectively, with the exception of a higher selectable final output delay. The SmartGen PLL Wizard will configure these outputs according to user specifications and can enable these signals with or without the enabling of Global Output Clocks.

The above signals can be enabled in the following output groupings in both internal and external feedback configurations of the static PLL:

- One output – GLA only
- Two outputs – GLA + (GLB and/or YB)
- Three outputs – GLA + (GLB and/or YB) + (GLC and/or YC)

PLL Macro Block Diagram

As illustrated, the PLL supports three distinct output frequencies from a given input clock. Two of these (GLB and GLC) can be routed to the B and C global network access, respectively, and/or routed to the device core (YB and YC).

There are five delay elements to support phase control on all five outputs (GLA, GLB, GLC, YB, and YC).

There are delay elements in the feedback loop that can be used to advance the clock relative to the reference clock.

The PLL macro reference clock can be driven in the following ways:

1. By an INBUF* macro to create a composite macro, where the I/O macro drives the global buffer (with programmable delay) using a hardwired connection. In this case, the I/O must be placed in one of the dedicated global I/O locations.
2. Directly from the FPGA core.
3. From an I/O that is routed through the FPGA regular routing fabric. In this case, users must instantiate a special macro, PLLINT, to differentiate from the hardwired I/O connection described earlier.

During power-up, the PLL outputs will toggle around the maximum frequency of the voltage-controlled oscillator (VCO) gear selected. Toggle frequencies can range from 40 MHz to 250 MHz. This will continue as long as the clock input (CLKA) is constant (HIGH or LOW). This can be prevented by LOW assertion of the POWERDOWN signal.

The visual PLL configuration in SmartGen, a component of the Libero SoC and Designer tools, will derive the necessary internal divider ratios based on the input frequency and desired output frequencies selected by the user.

Phase Adjustment

The four phases available (0, 90, 180, 270) are phases with respect to VCO (PLL output). The VCO is divided to achieve the user's CCC required output frequency (GLA, YB/GLB, YC/GLC). The division happens after the selection of the VCO phase. The effective phase shift is actually the VCO phase shift divided by the output divider. This is why the visual CCC shows both the actual achievable phase and more importantly the actual delay that is equivalent to the phase shift that can be achieved.

Dynamic PLL Configuration

The CCCs can be configured both statically and dynamically.

In addition to the ports available in the Static CCC, the Dynamic CCC has the dynamic shift register signals that enable dynamic reconfiguration of the CCC. With the Dynamic CCC, the ports CLKB and CLKC are also exposed. All three clocks (CLKA, CLKB, and CLKC) can be configured independently.

The CCC block is fully configurable. The following two sources can act as the CCC configuration bits.

Flash Configuration Bits

The flash configuration bits are the configuration bits associated with programmed flash switches. These bits are used when the CCC is in static configuration mode. Once the device is programmed, these bits cannot be modified. They provide the default operating state of the CCC.

Dynamic Shift Register Outputs

This source does not require core reprogramming and allows core-driven dynamic CCC reconfiguration. When the dynamic register drives the configuration bits, the user-defined core circuit takes full control over SDIN, SDOUT, SCLK, SSHIFT, and SUPDATE. The configuration bits can consequently be dynamically changed through shift and update operations in the serial register interface. Access to the logic core is accomplished via the dynamic bits in the specific tiles assigned to the PLLs.

Figure 4-21 illustrates a simplified block diagram of the MUX architecture in the CCCs.
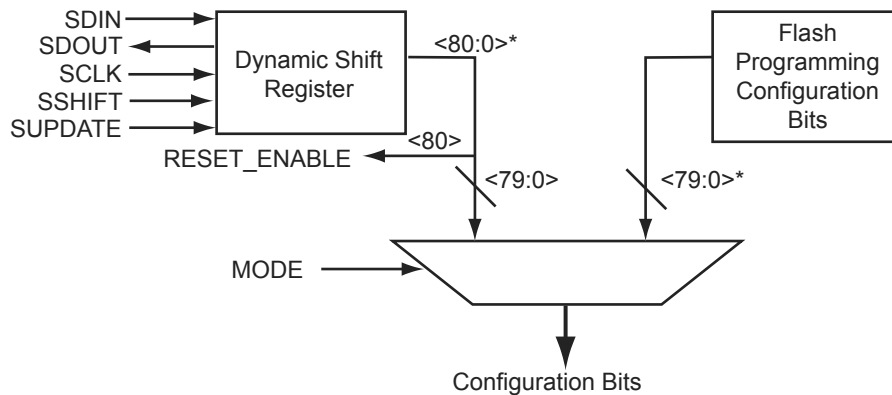
Note: *For Fusion, bit <88:81> is also needed.

Figure 4-21 • The CCC Configuration MUX Architecture

The selection between the flash configuration bits and the bits from the configuration register is made using the MODE signal shown in Figure 4-21. If the MODE signal is logic HIGH, the dynamic shift register configuration bits are selected. There are 81 control bits to configure the different functions of the CCC.

SmartGen allows you to generate the FlashROM netlist in VHDL, Verilog, or EDIF format. After the FlashROM netlist is generated, the core can be instantiated in the main design like other SmartGen cores. Note that the macro library name for FlashROM is UFROM. The following is a sample FlashROM VHDL netlist that can be instantiated in the main design:

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

entity FROM_a is
  port( ADDR : in std_logic_vector(6 downto 0); DOUT : out std_logic_vector(7 downto 0));
end FROM_a;

architecture DEF_ARCH of  FROM_a is

  component UFROM
    generic (MEMORYFILE:string);
    port(DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7 : out std_logic;
      ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, ADDR5, ADDR6 : in std_logic := 'U') ;
  end component;

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

signal U_7_PIN2 : std_logic ;

begin

  GND_1_net : GND port map(Y => U_7_PIN2);
  UFROM0 : UFROM
  generic map(MEMORYFILE => "FROM_a.mem")
  port map(DO0 => DOUT(0), DO1 => DOUT(1), DO2 => DOUT(2), DO3 => DOUT(3), DO4 => DOUT(4),
    DO5 => DOUT(5), DO6 => DOUT(6), DO7 => DOUT(7), ADDR0 => ADDR(0), ADDR1 => ADDR(1),
    ADDR2 => ADDR(2), ADDR3 => ADDR(3), ADDR4 => ADDR(4), ADDR5 => ADDR(5),
    ADDR6 => ADDR(6));

end DEF_ARCH;
```

SmartGen generates the following files along with the netlist. These are located in the SmartGen folder for the Libero SoC project.

1. MEM (Memory Initialization) file
2. UFC (User Flash Configuration) file
3. Log file

The MEM file is used for simulation, as explained in the "Simulation of FlashROM Design" section on page 127. The UFC file, generated by SmartGen, has the FlashROM configuration for single or multiple devices and is used during STAPL generation. It contains the region properties and simulation values. Note that any changes in the MEM file will not be reflected in the UFC file. Do not modify the UFC to change FlashROM content. Instead, use the SmartGen GUI to modify the FlashROM content. See the "Programming File Generation for FlashROM Design" section on page 127 for a description of how the UFC file is used during the programming file generation. The log file has information regarding the file type and file location.
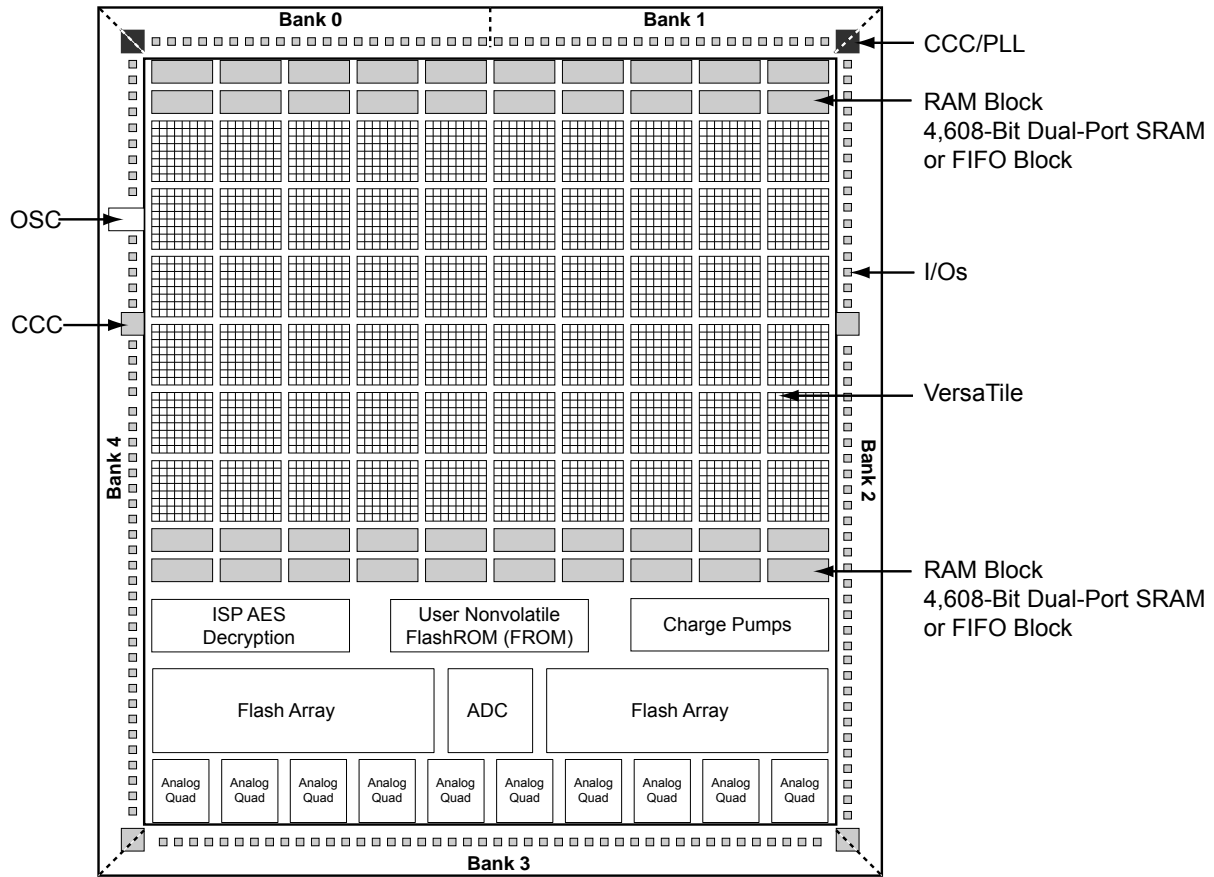
Figure 6-2 • **Fusion Device Architecture Overview (AFS600)**

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*.

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 8-12).

Figure 8-12 • **Setting VCCI for a Bank**

Automatically Assigning Technologies to I/O Banks

The I/O Bank Assigner (IOBA) tool runs automatically when you run Layout. You can also use this tool from within the MultiView Navigator (Figure 8-17). The IOBA tool automatically assigns technologies and VREF pins (if required) to every I/O bank that does not currently have any technologies assigned to it. This tool is available when at least one I/O bank is unassigned.

To automatically assign technologies to I/O banks, choose I/O Bank Assigner from the **Tools** menu (or click the I/O Bank Assigner's toolbar button, shown in Figure 8-16).

Figure 8-16 • **I/O Bank Assigner's Toolbar Button**

Messages will appear in the Output window informing you when the automatic I/O bank assignment begins and ends. If the assignment is successful, the message "I/O Bank Assigner completed successfully" appears in the Output window, as shown in Figure 8-17.

Figure 8-17 • **I/O Bank Assigner Displays Messages in Output Window**

Device Programmers

Single Device Programmer

Single device programmers are used to program a device before it is mounted on the system board.

The advantage of using device programmers is that no programming hardware is required on the system board. Therefore, no additional components or board space are required.

Adapter modules are purchased with single device programmers to support the FPGA packages used. The FPGA is placed in the adapter module and the programming software is run from a PC. Microsemi supplies the programming software for all of the Microsemi programmers. The software allows for the selection of the correct die/package and programming files. It will then program and verify the device.

- Single-site programmers

  A single-site programmer programs one device at a time. Microsemi offers Silicon Sculptor 3, built by BP Microsystems, as a single-site programmer. Silicon Sculptor 3 and associated software are available only from Microsemi.

  – Advantages: Lower cost than multi-site programmers. No additional overhead for programming on the system board. Allows local control of programming and data files for maximum security. Allows on-demand programming on-site.

  – Limitations: Only programs one device at a time.

- Multi-site programmers

Often referred to as batch or gang programmers, multi-site programmers can program multiple devices at the same time using the same programming file. This is often used for large volume programming and by programming houses. The sites often have independent processors and memory enabling the sites to operate concurrently, meaning each site may start programming the same file independently. This enables the operator to change one device while the other sites continue programming, which increases throughput. Multiple adapter modules for the same package are required when using a multi-site programmer. Silicon Sculptor I, II, and 3 programmers can be cascaded to program multiple devices in a chain. Multi-site programmers, such as the BP2610 and BP2710, can also be purchased from BP Microsystems. When using BP Microsystems multi-site programmers, users must use programming adapter modules available only from Microsemi. Visit the Microsemi SoC Products Group website to view the part numbers of the desired adapter module:

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

Also when using BP Microsystems programmers, customers must use Microsemi programming software to ensure the best programming result will occur.

  – Advantages: Provides the capability of programming multiple devices at the same time. No additional overhead for programming on the system board. Allows local control of programming and data files for maximum security.

  – Limitations: More expensive than a single-site programmer

- Automated production (robotic) programmers

Automated production programmers are based on multi-site programmers. They consist of a large input tray holding multiple parts and a robotic arm to select and place parts into appropriate programming sockets automatically. When the programming of the parts is complete, the parts are removed and placed in a finished tray. The automated programmers are often used in volume programming houses to program parts for which the programming time is small. BP Microsystems part number BP4710, BP4610, BP3710 MK2, and BP3610 are available for this purpose. Auto programmers cannot be used to program RTAX-S devices.

Where an auto-programmer is used, the appropriate open-top adapter module from BP Microsystems must be used.

Application 3: Nontrusted Environment—Field Updates/Upgrades

Programming or reprogramming of devices may occur at remote locations. Reconfiguration of devices in consumer products/equipment through public networks is one example. Typically, the remote system is already programmed with particular design contents. When design update (FPGA array contents update) and/or data upgrade (FlashROM and/or FB contents upgrade) is necessary, an updated programming file with AES encryption can be generated, sent across public networks, and transmitted to the remote system. Reprogramming can then be done using this AES-encrypted programming file, providing easy and secure field upgrades. Low power flash devices support this secure ISP using AES. The detailed flow for this application is shown in Figure 11-8. Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

To prepare devices for this scenario, the user can initially generate a programming file with the available security setting options. This programming file is programmed into the devices before shipment. During the programming file generation step, the user has the option of making the security settings permanent or not. In situations where no changes to the security settings are necessary, the user can select this feature in the software to generate the programming file with permanent security settings. Microsemi recommends that the programming file use encryption with an AES key, especially when ISP is done via public domain.

For example, if the designer wants to use an AES key for the FPGA array and the FlashROM, **Permanent** needs to be chosen for this setting. At first, the user chooses the options to use an AES key for the FPGA array and the FlashROM, and then chooses **Permanently lock the security settings**. A unique AES key is chosen. Once this programming file is generated and programmed to the devices, the AES key is permanently stored in the on-chip memory, where it is secured safely. The devices are sent to distant locations for the intended application. When an update is needed, a new programming file must be generated. The programming file must use the same AES key for encryption; otherwise, the authentication will fail and the file will not be programmed in the device.
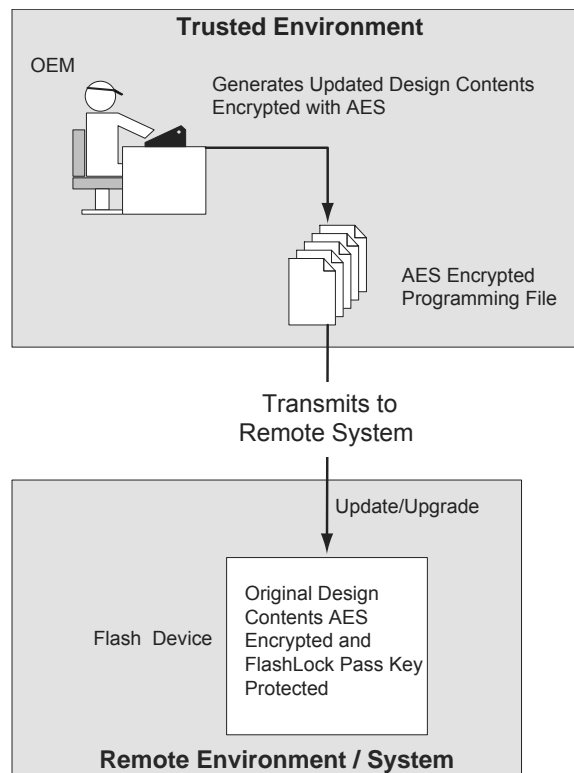
Figure 11-8 • **Application 3: Nontrusted Environment—Field Updates/Upgrades**

Figure 11-19 • **FlashLock Pass Key, Previously Programmed Devices**

It is important to note that when the security settings need to be updated, the user also needs to select the **Security settings** check box in Step 1, as shown in Figure 11-10 on page 248 and Figure 11-11 on page 248, to modify the security settings. The user must consider the following:

- If only a new AES key is necessary, the user must re-enter the same Pass Key previously programmed into the device in Designer and then generate a programming file with the same Pass Key and a different AES key. This ensures the programming file can be used to access and program the device and the new AES key.

- If a new Pass Key is necessary, the user can generate a new programming file with a new Pass Key (with the same or a new AES key if desired). However, for programming, the user must first load the original programming file with the Pass Key that was previously used to unlock the device. Then the new programming file can be used to program the new security settings.

Advanced Options

As mentioned, there may be applications where more complicated security settings are required. The "Custom Security Levels" section in the *FlashPro User's Guide* describes different advanced options available to aid the user in obtaining the best available security settings.

Microsemi's Flash Families Support Voltage Switching Circuit

The flash FPGAs listed in Table 13-1 support the voltage switching circuit feature and the functions described in this document.

Table 13-1 • **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 13-1. 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 13-1. 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*.

Microsemi's Flash Devices Support the JTAG Feature

The flash-based FPGAs listed in Table 15-1 support the JTAG feature and the functions described in this document.

Table 15-1 • **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 ProASIC®3 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 15-1. 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 15-1. 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*.

Microsemi

16 – UJTAG Applications in Microsemi's Low Power Flash Devices

Introduction

In Fusion, IGLOO, and ProASIC3 devices, there is bidirectional access from the JTAG port to the core VersaTiles during normal operation of the device (Figure 16-1). User JTAG (UJTAG) is the ability for the design to use the JTAG ports for access to the device for updates, etc. While regular JTAG is used, the UJTAG tiles, located at the southeast area of the die, are directly connected to the JTAG Test Access Port (TAP) Controller in normal operating mode. As a result, all the functional blocks of the device, such as Clock Conditioning Circuits (CCCs) with PLLs, SRAM blocks, embedded FlashROM, flash memory blocks, and I/O tiles, can be reached via the JTAG ports. The UJTAG functionality is available by instantiating the UJTAG macro directly in the source code of a design. Access to the FPGA core VersaTiles from the JTAG ports enables users to implement different applications using the TAP Controller (JTAG port). This document introduces the UJTAG tile functionality and discusses a few application examples. However, the possible applications are not limited to what is presented in this document. UJTAG can serve different purposes in many designs as an elementary or auxiliary part of the design. For detailed usage information, refer to the "Boundary Scan in Low Power Flash Devices" section on page 291.
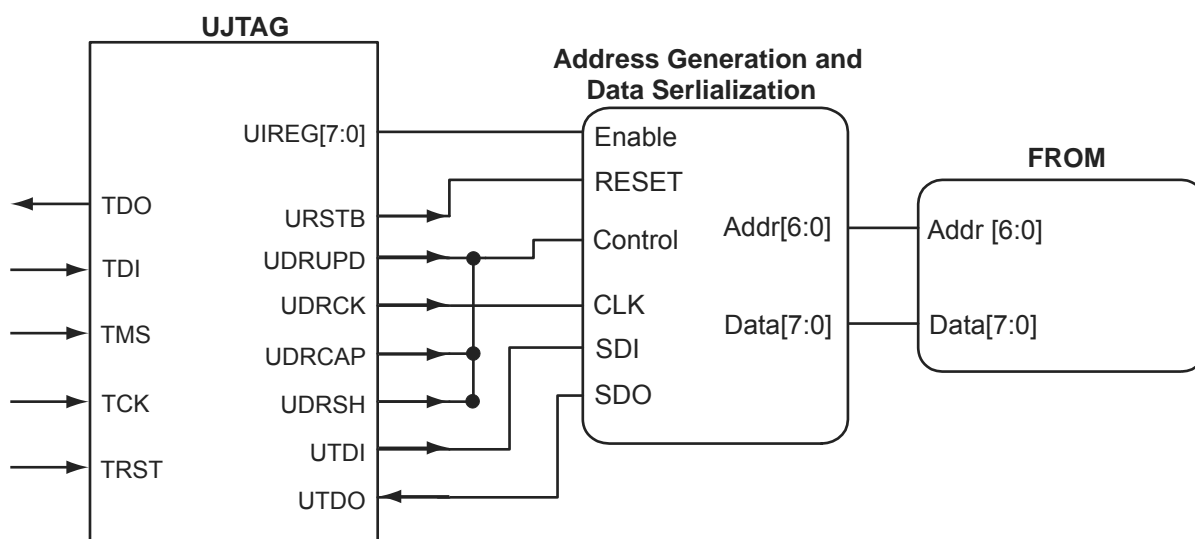

Figure 16-1 • **Block Diagram of Using UJTAG to Read FlashROM Contents**

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 16-5 shows a flow chart example for fine-tuning application steps using the UJTAG tile.

In Figure 16-5, 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.
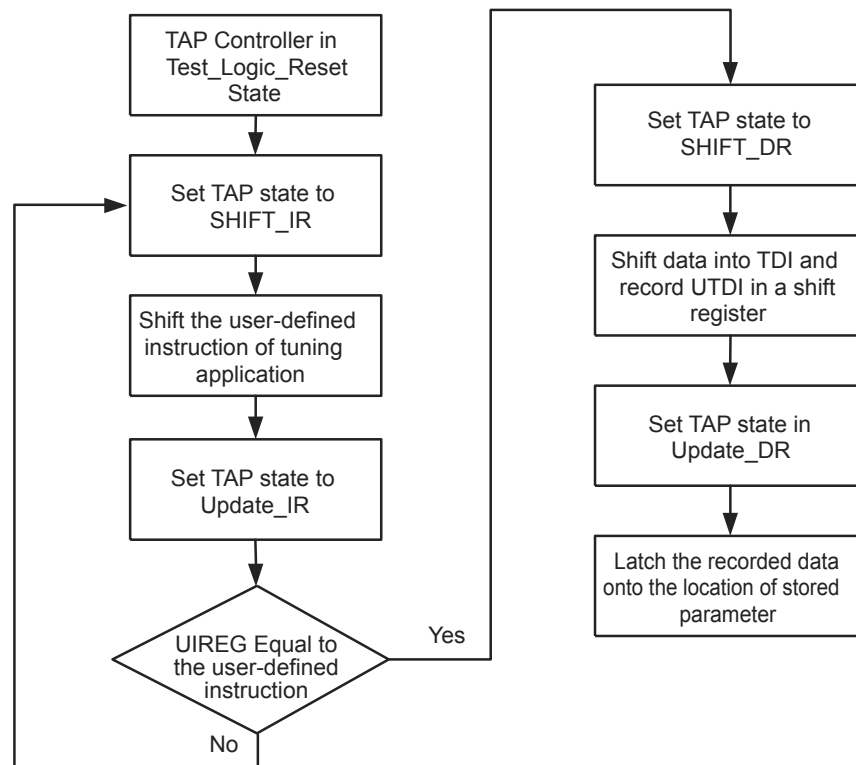
Figure 16-5 • **Flow Chart Example of Fine-Tuning an Application Using UJTAG**

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 16-6 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 16-5 on page 303. 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.
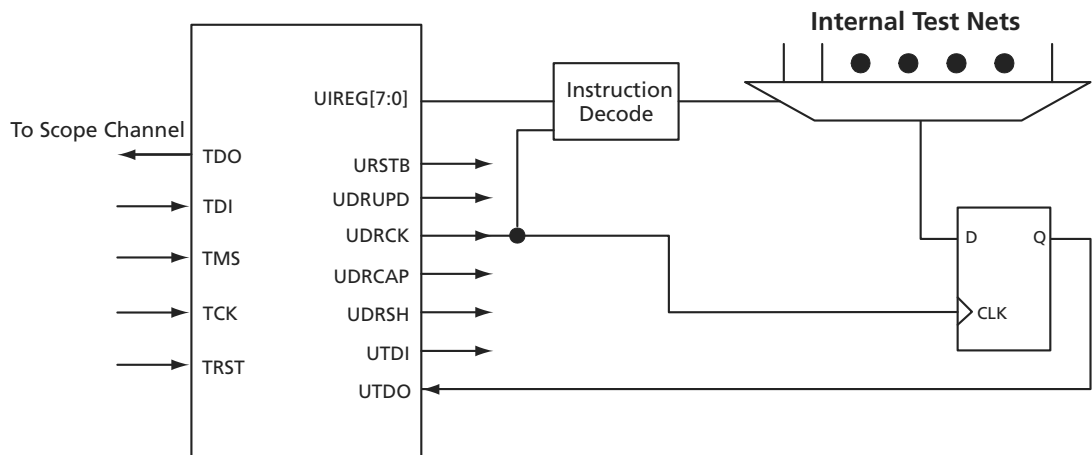
Figure 16-6 • **UJTAG Usage Example in Test and Debug Applications**

Power-Up/-Down Sequence and Transient Current

Microsemi's low power flash devices use the following main voltage pins during normal operation:[2]

- VCCPLX
- VJTAG
- VCC: Voltage supply to the FPGA core
  - VCC is 1.5 V ± 0.075 V for IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3 devices operating at 1.5 V.
  - VCC is 1.2 V ± 0.06 V for IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices operating at 1.2 V.
  - V5 devices will require a 1.5 V VCC supply, whereas V2 devices can utilize either a 1.2 V or 1.5 V VCC.
- VCCIBx: Supply voltage to the bank's I/O output buffers and I/O logic. Bx is the I/O bank number.
- VMVx: Quiet supply voltage to the input buffers of each I/O bank. x is the bank number. (Note: IGLOO nano, IGLOO PLUS, and ProASIC3 nano devices do not have VMVx supply pins.)

The I/O bank VMV pin must be tied to the VCCI pin within the same bank. Therefore, the supplies that need to be powered up/down during normal operation are VCC and VCCI. These power supplies can be powered up/down in any sequence during normal operation of IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, ProASIC3, and ProASIC3 nano FPGAs. During power-up, I/Os in each bank will remain tristated until the last supply (either VCCIBx or VCC) reaches its functional activation voltage. Similarly, during power-down, I/Os of each bank are tristated once the first supply reaches its brownout deactivation voltage.

Although Microsemi's low power flash devices have no power-up or power-down sequencing requirements, Microsemi identifies the following power conditions that will result in higher than normal transient current. Use this information to help maximize power savings:

Microsemi recommends tying VCCPLX to VCC and using proper filtering circuits to decouple VCC noise from the PLL.

a. If VCCPLX is powered up before VCC, a static current of up to 5 mA (typical) per PLL may be measured on VCCPLX.

   The current vanishes as soon as VCC reaches the VCCPLX voltage level.

   The same current is observed at power-down (VCC before VCCPLX).

b. If VCCPLX is powered up simultaneously or after VCC:

   i. Microsemi's low power flash devices exhibit very low transient current on VCC. For ProASIC3 devices, the maximum transient current on V_{CC} does not exceed the maximum standby current specified in the device datasheet.

The source of transient current, also known as inrush current, varies depending on the FPGA technology. Due to their volatile technology, the internal registers in SRAM FPGAs must be initialized before configuration can start. This initialization is the source of significant inrush current in SRAM FPGAs during power-up. Due to the nonvolatile nature of flash technology, low power flash devices do not require any initialization at power-up, and there is very little or no crossbar current through PMOS and NMOS devices. Therefore, the transient current at power-up is significantly less than for SRAM FPGAs. Figure 17-1 on page 310 illustrates the types of power consumption by SRAM FPGAs compared to Microsemi's antifuse and flash FPGAs.

---

2. *For more information on Microsemi FPGA voltage supplies, refer to the appropriate datasheet located at http://www.microsemi.com/soc/techdocs/ds.*

A – Summary of Changes

History of Revision to Chapters

The following table lists chapters that were affected in each revision of this document. Each chapter includes its own change history because it may appear in other device family user's guides. Refer to the individual chapter for a list of specific changes.

Revision (month/year)	Chapter Affected	List of Changes (page number)
Revision 5 (September 2012)	"Microprocessor Programming of Microsemi's Low Power Flash Devices" was revised.	290
Revision 4 (August 2012)	"FPGA Array Architecture in Low Power Flash Devices" was revised.	20
	The "Low Power Modes in ProASIC3/E and ProASIC3 nano FPGAs" chapter was added (SAR 32020).	21
	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	113
	"SRAM and FIFO Memories in Microsemi's Low Power Flash Devices" was revised.	157
	"I/O Structures in nano Devices" was revised.	183
	The "Pin Descriptions" and "Packaging" chapters were removed. This information is now published in the datasheet for each product line (SAR 34772).	N/A
	"In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" was revised.	273
	"Boundary Scan in Low Power Flash Devices" was revised.	296
Revision 3 (December 2011)	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	113
	"UJTAG Applications in Microsemi's Low Power Flash Devices" was revised.	306
Revision 2 (June 2011)	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	113
	"I/O Structures in nano Devices" was revised.	183
	"I/O Software Control in Low Power Flash Devices" was revised.	204
	"In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" was revised.	273
Revision 1 (July 2010)	"Global Resources in Low Power Flash Devices" was revised.	59
	"Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" was revised.	113
	"I/O Software Control in Low Power Flash Devices" was revised.	204
	"DDR for Microsemi's Low Power Flash Devices" was revised.	219
	"Programming Flash Devices" was revised.	232

Microsemi

Index