



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	Active
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	110592
Number of I/O	235
Number of Gates	600000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	484-BGA
Supplier Device Package	484-FPBGA (23x23)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3p600l-fgg484

Table of Contents

Introduction	7
Contents	7
Revision History	7
Related Information	7
1 FPGA Array Architecture in Low Power Flash Devices	9
Device Architecture	9
FPGA Array Architecture Support	10
Device Overview	11
Related Documents	20
List of Changes	20
2 Flash*Freeze Technology and Low Power Modes	21
Flash*Freeze Technology and Low Power Modes	21
Flash Families Support the Flash*Freeze Feature	22
Low Power Modes Overview	23
Static (Idle) Mode	23
Flash*Freeze Mode	24
Sleep and Shutdown Modes	32
Flash*Freeze Design Guide	34
Conclusion	42
Related Documents	42
List of Changes	42
3 Global Resources in Low Power Flash Devices	47
Introduction	47
Global Architecture	47
Global Resource Support in Flash-Based Devices	48
VersaNet Global Network Distribution	49
Chip and Quadrant Global I/Os	51
Spine Architecture	57
Using Clock Aggregation	60
Design Recommendations	62
Conclusion	74
Related Documents	74
List of Changes	75
4 Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs	77
Introduction	77
Overview of Clock Conditioning Circuitry	77
CCC Support in Microsemi's Flash Devices	79
Global Buffers with No Programmable Delays	80
Global Buffer with Programmable Delay	80
Global Buffers with PLL Function	83
Global Input Selections	87

2 – Flash*Freeze Technology and Low Power Modes

Flash*Freeze Technology and Low Power Modes

Microsemi IGLOO,[®] IGLOO nano, IGLOO PLUS, ProASIC[®]3L, and Radiation-Tolerant (RT) ProASIC3 FPGAs with Flash*Freeze technology are designed to meet the most demanding power and area challenges of today's portable electronics products with a reprogrammable, small-footprint, full-featured flash FPGA. These devices offer lower power consumption in static and dynamic modes, utilizing the unique Flash*Freeze technology, than any other FPGA or CPLD.

IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 devices offer various power-saving modes that enable every system to utilize modes that achieve the lowest total system power. Low Power Active capability (static idle) allows for ultra-low power consumption while the device is operational in the system by maintaining SRAM, registers, I/Os, and logic functions.

Flash*Freeze technology provides an ultra-low power static mode (Flash*Freeze mode) that retains all SRAM and register information with rapid recovery to Active (operating) mode. IGLOO nano and IGLOO PLUS devices have an additional feature when operating in Flash*Freeze mode, allowing them to retain I/O states as well as SRAM and register states. This mechanism enables the user to quickly (within 1 μ s) enter and exit Flash*Freeze mode by activating the Flash*Freeze (FF) pin while all power supplies are kept in their original states. In addition, I/Os and clocks connected to the FPGA can still be toggled without impact on device power consumption. While in Flash*Freeze mode, the device retains all core register states and SRAM information. This mode can be configured so that no power is consumed by the I/O banks, clocks, JTAG pins, or PLLs; and the IGLOO and IGLOO PLUS devices consume as little as 5 μ W, while IGLOO nano devices consume as little as 2 μ W. Microsemi offers a state management IP core to aid users in gating clocks and managing data before entering Flash*Freeze mode.

This document will guide users in selecting the best low power mode for their applications, and introduces Microsemi's Flash*Freeze management IP core.

The following will happen during demotion of a global signal to regular nets:

- CLKBUF_x becomes INBUF_x; CLKINT is removed from the netlist.
- The essential global macro, such as the output of the Clock Conditioning Circuit, cannot be demoted.
- No automatic buffering will happen.

Since no automatic buffering happens when a signal is demoted, this net may have a high delay due to large fanout. This may have a negative effect on the quality of the results. Microsemi recommends that the automatic global demotion only be used on small-fanout nets. Use clock networks for high-fanout nets to improve timing and routability.

Spine Assignment

The low power flash device architecture allows the global networks to be segmented and used as clock spines. These spines, also called local clock networks, enable the use of PDC or MVN to assign a signal to a spine.

PDC syntax to promote a net to a spine/local clock:

```
assign_local_clock -net netname -type [quadrant|chip] Tn|Bn|Tn:Bm
```

If the net is driven by a clock macro, Designer automatically demotes the clock net to a regular net before it is assigned to a spine. Nets driven by a PLL or CLKDLY macro cannot be assigned to a local clock.

When assigning a signal to a spine or quadrant global network using PDC (pre-compile), the Designer software will legalize the shared instances. The number of shared instances to be legalized can be controlled by compile options. If these networks are created in MVN (only quadrant globals can be created), no legalization is done (as it is post-compile). Designer does not do legalization between non-clock nets.

As an example, consider two nets, net_clk and net_reset, driving the same flip-flop. The following PDC constraints are used:

```
assign_local_clock -net net_clk -type chip T3
assign_local_clock -net net_reset -type chip T1:T2
```

During Compile, Designer adds a buffer in the reset net and places it in the T1 or T2 region, and places the flip-flop in the T3 spine region (Figure 3-16).

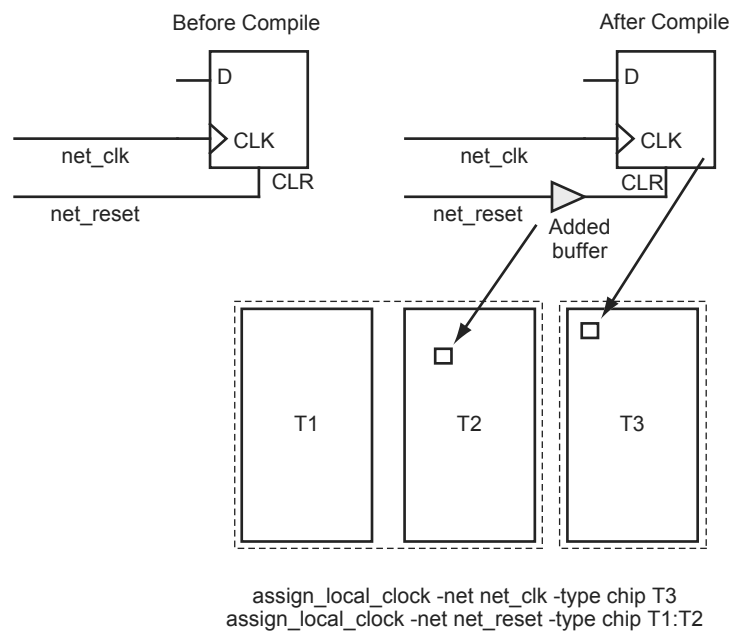
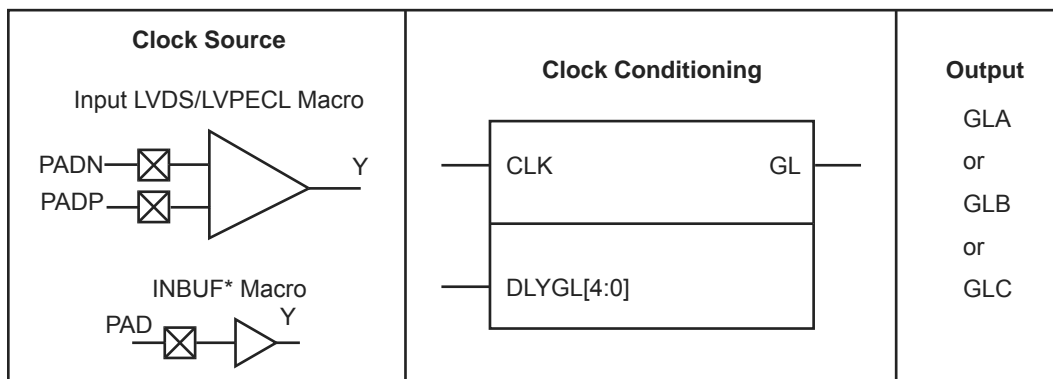


Figure 3-16 • Adding a Buffer for Shared Instances

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
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
	The "Global Architecture" section and "VersaNet Global Network Distribution" section were revised for clarity (SARs 20646, 24779).	47, 49
	The "I/O Banks and Global I/Os" section was moved earlier in the document, renamed to "Chip and Quadrant Global I/Os", and revised for clarity. Figure 3-4 • Global Connections Details, Figure 3-6 • Global Inputs, Table 3-2 • Chip Global Pin Name, and Table 3-3 • Quadrant Global Pin Name are new (SARs 20646, 24779).	51
	The "Clock Aggregation Architecture" section was revised (SARs 20646, 24779).	57
	Figure 3-7 • Chip Global Aggregation was revised (SARs 20646, 24779).	59
	The "Global Macro and Placement Selections" section is new (SARs 20646, 24779).	64
v1.4 (December 2008)	The "Global Architecture" section was updated to include 10 k devices, and to include information about VersaNet global support for IGLOO nano devices.	47
	The Table 3-1 • Flash-Based FPGAs was updated to include IGLOO nano and ProASIC3 nano devices.	48
	The "VersaNet Global Network Distribution" section was updated to include 10 k devices and to note an exception in global lines for nano devices.	49
	Figure 3-2 • Simplified VersaNet Global Network (30 k gates and below) is new.	50
	The "Spine Architecture" section was updated to clarify support for 10 k and nano devices.	57
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include IGLOO nano and ProASIC3 nano devices.	57
	The figure in the CLKBUF_LVDS/LVPECL row of Table 3-8 • Clock Macros was updated to change CLKBIBUF to CLKBUF.	62
v1.3 (October 2008)	A third bullet was added to the beginning of the "Global Architecture" section: In Fusion devices, the west CCC also contains a PLL core. In the two larger devices (AFS600 and AFS1500), the west and east CCCs each contain a PLL.	47
	The "Global Resource Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	48
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include A3PE600/L in the device column.	57
	Table note 1 was revised in Table 3-9 • I/O Standards within CLKBUF to include AFS600 and AFS1500.	63
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 3-1 • Flash-Based FPGAs: <ul style="list-style-type: none"> ProASIC3L was updated to include 1.5 V. The number of PLLs for ProASIC3E was changed from five to six. 	48



Notes:

1. For INBUF* driving a PLL macro or CLKDLY macro, the I/O will be hard-routed to the CCC; i.e., will be placed by software to a dedicated Global I/O.
2. IGLOO nano and ProASIC3 nano devices do not support differential inputs.

Figure 4-3 • CCC Options: Global Buffers with Programmable Delay

The CLKDLY macro is a pass-through clock source that does not use the PLL, but provides the ability to delay the clock input using a programmable delay. The CLKDLY macro takes the selected clock input and adds a user-defined delay element. This macro generates an output clock phase shift from the input clock.

The CLKDLY macro can be driven 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 software will automatically place the dedicated global I/O in the appropriate locations. Many specific INBUF macros support the wide variety of single-ended and differential I/O standards supported by the low power flash family. The available INBUF macros are described in the *IGLOO*, *ProASIC3*, *SmartFusion*, and *Fusion Macro Library Guide*.

The CLKDLY macro can be driven directly from the FPGA core. The CLKDLY macro can also be driven 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 the clock input driven by the hardwired I/O connection.

The visual CLKDLY configuration in the SmartGen area of the Microsemi Libero System-on-Chip (SoC) and Designer tools allows the user to select the desired amount of delay and configures the delay elements appropriately. SmartGen also allows the user to select the input clock source. SmartGen will automatically instantiate the special macro, PLLINT, when needed.

CLKDLY Macro Signal Descriptions

The CLKDLY macro supports one input and one output. Each signal is described in Table 4-2.

Table 4-2 • Input and Output Description of the CLKDLY Macro

Signal	Name	I/O	Description
CLK	Reference Clock	Input	Reference clock input
GL	Global Output	Output	Primary output clock to respective global/quadrant clock networks

Implementing EXTFB in ProASIC3/E Devices

When the external feedback (EXTFB) signal of the PLL in the ProASIC3/E devices is implemented, the phase detector of the PLL core receives the reference clock (CLKA) and EXTFB as inputs. EXTFB must be sourced as an INBUF macro and located at the global/chip clock location associated with the target PLL by Designer software. EXTFB cannot be sourced from the FPGA fabric.

The following example shows CLKA and EXTFB signals assigned to two global I/Os in the same global area of ProASIC3E device.

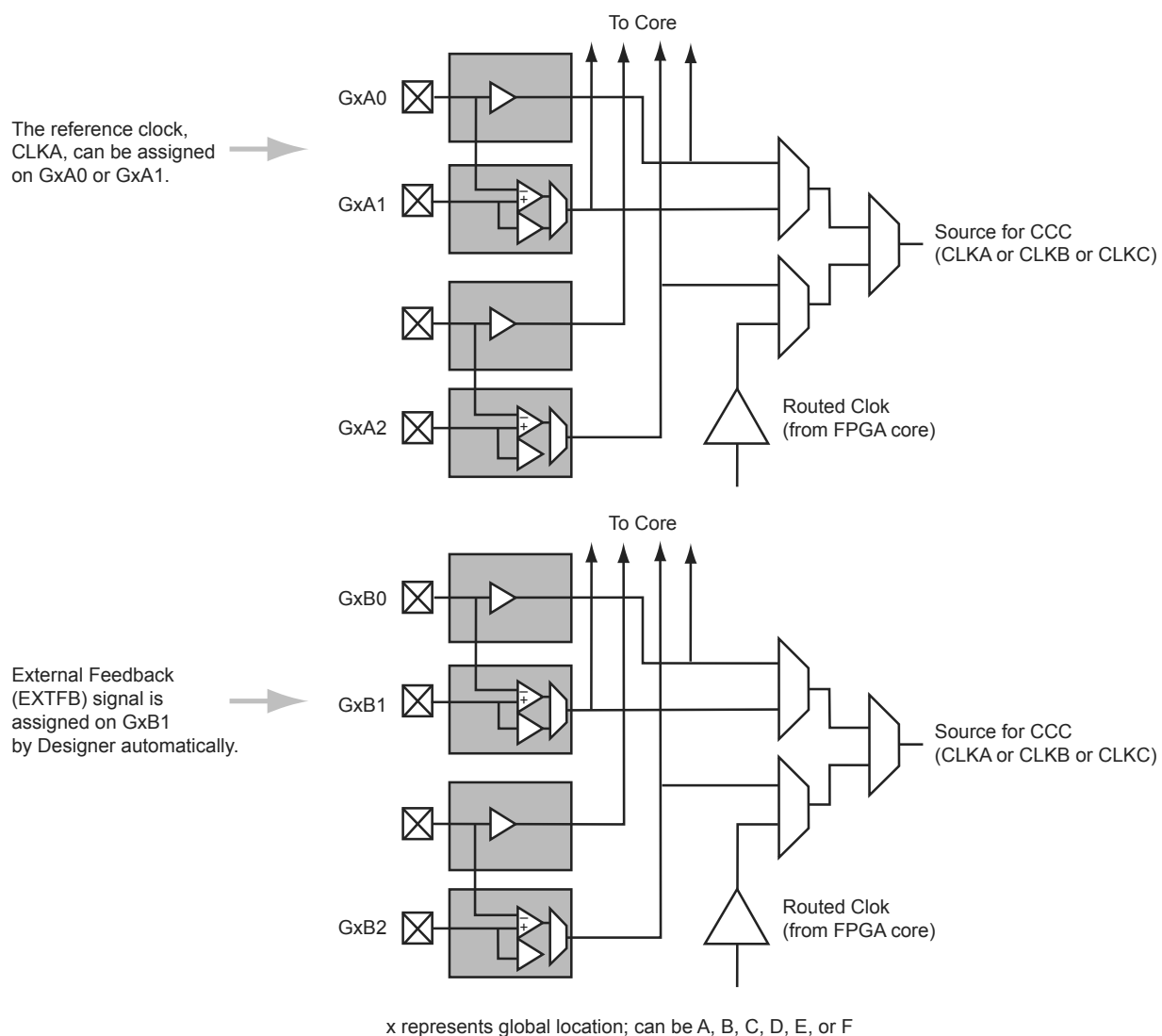


Figure 4-5 • CLKA and EXTFB Assigned to Global I/Os

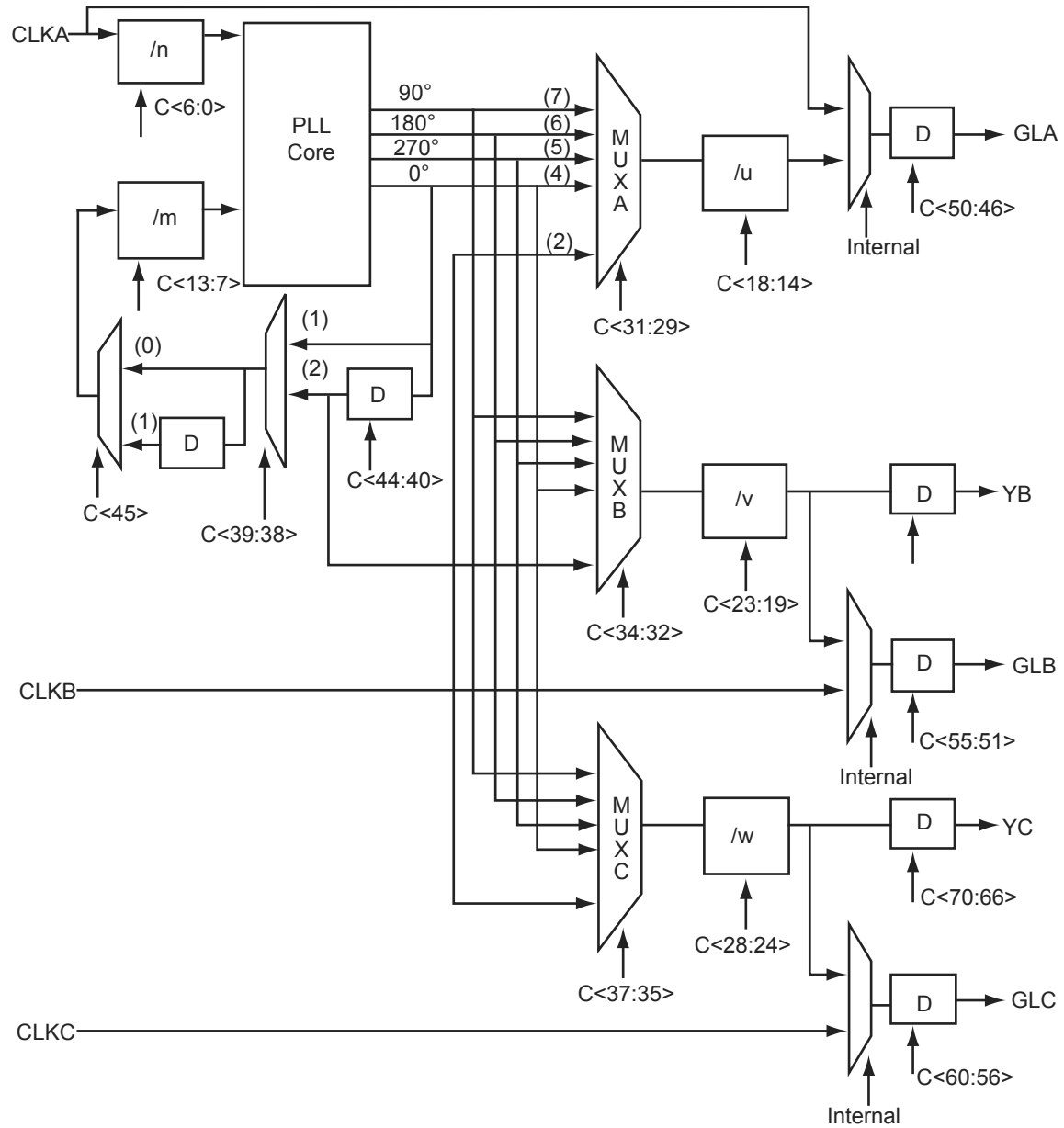


Figure 4-22 • CCC Block Control Bits – Graphical Representation of Assignments

FlashROM Design Flow

The Microsemi Libero System-on-Chip (SoC) software has extensive FlashROM support, including FlashROM generation, instantiation, simulation, and programming. Figure 5-9 shows the user flow diagram. In the design flow, there are three main steps:

1. FlashROM generation and instantiation in the design
2. Simulation of FlashROM design
3. Programming file generation for FlashROM design

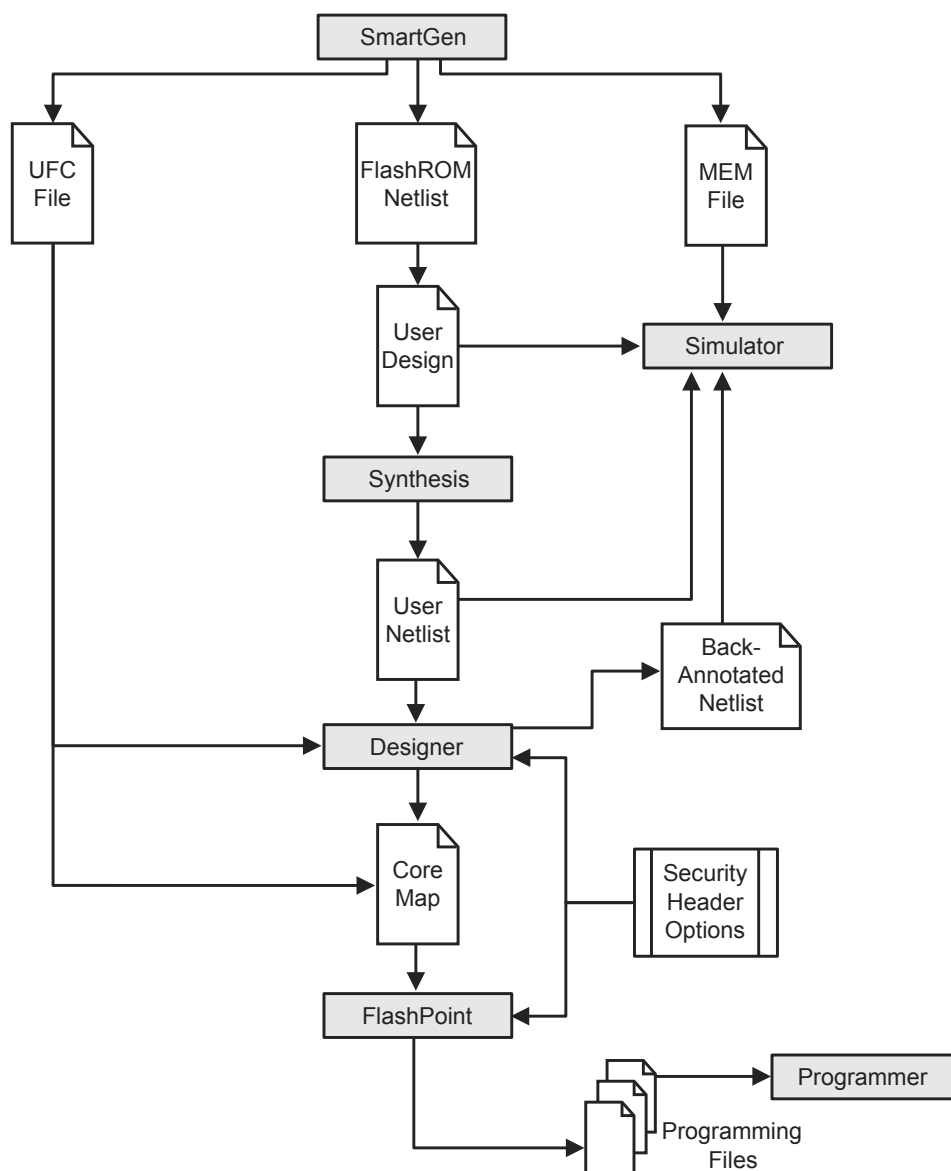


Figure 5-9 • FlashROM Design Flow

SRAM and FIFO Architecture

To meet the needs of high-performance designs, the memory blocks operate strictly in synchronous mode for both read and write operations. The read and write clocks are completely independent, and each can operate at any desired frequency up to 250 MHz.

- 4k×1, 2k×2, 1k×4, 512×9 (dual-port RAM—2 read / 2 write or 1 read / 1 write)
- 512×9, 256×18 (2-port RAM—1 read / 1 write)
- Sync write, sync pipelined / nonpipelined read

Automotive ProASIC3 devices support single-port SRAM capabilities or dual-port SRAM only under specific conditions. Dual-port mode is supported if the clocks to the two SRAM ports are the same and 180° out of phase (i.e., the port A clock is the inverse of the port B clock). The Libero SoC software macro libraries support a dual-port macro only. For use of this macro as a single-port SRAM, the inputs and clock of one port should be tied off (grounded) to prevent errors during design compile. For use in dual-port mode, the same clock with an inversion between the two clock pins of the macro should be used in the design to prevent errors during compile.

The memory block includes dedicated FIFO control logic to generate internal addresses and external flag logic (FULL, EMPTY, AFULL, AEMPTY).

Simultaneous dual-port read/write and write/write operations at the same address are allowed when certain timing requirements are met.

During RAM operation, addresses are sourced by the user logic, and the FIFO controller is ignored. In FIFO mode, the internal addresses are generated by the FIFO controller and routed to the RAM array by internal MUXes.

The low power flash device architecture enables the read and write sizes of RAMs to be organized independently, allowing for bus conversion. For example, the write size can be set to 256×18 and the read size to 512×9.

Both the write width and read width for the RAM blocks can be specified independently with the WW (write width) and RW (read width) pins. The different D×W configurations are 256×18, 512×9, 1k×4, 2k×2, and 4k×1. When widths of one, two, or four are selected, the ninth bit is unused. For example, when writing nine-bit values and reading four-bit values, only the first four bits and the second four bits of each nine-bit value are addressable for read operations. The ninth bit is not accessible.

Conversely, when writing four-bit values and reading nine-bit values, the ninth bit of a read operation will be undefined. The RAM blocks employ little-endian byte order for read and write operations.

Memory Blocks and Macros

Memory blocks can be configured with many different aspect ratios, but are generically supported in the macro libraries as one of two memory elements: RAM4K9 or RAM512X18. The RAM4K9 is configured as a true dual-port memory block, and the RAM512X18 is configured as a two-port memory block. Dual-port memory allows the RAM to both read from and write to either port independently. Two-port memory allows the RAM to read from one port and write to the other using a common clock or independent read and write clocks. If needed, the RAM4K9 blocks can be configured as two-port memory blocks. The memory block can be configured as a FIFO by combining the basic memory block with dedicated FIFO controller logic. The FIFO macro is named FIFO4KX18 (Figure 6-3 on page 152).

Clocks for the RAM blocks can be driven by the VersaNet (global resources) or by regular nets. When using local clock segments, the clock segment region that encompasses the RAM blocks can drive the RAMs. In the dual-port configuration (RAM4K9), each memory block port can be driven by either rising-edge or falling-edge clocks. Each port can be driven by clocks with different edges. Though only a rising-edge clock can drive the physical block itself, the Microsemi Designer software will automatically bubble-push the inversion to properly implement the falling-edge trigger for the RAM block.

RD

This is the output data bus and is 18 bits wide. Not all 18 bits are valid in all configurations. Like the WD bus, high-order bits become unusable if the data width is less than 18. The output data on unused pins is undefined (Table 6-7).

Table 6-7 • Input Data Signal Usage for Different Aspect Ratios

D×W	WD/RD Unused
4k×1	WD[17:1], RD[17:1]
2k×2	WD[17:2], RD[17:2]
1k×4	WD[17:4], RD[17:4]
512×9	WD[17:9], RD[17:9]
256×18	—

ESTOP, FSTOP

ESTOP is used to stop the FIFO read counter from further counting once the FIFO is empty (i.e., the EMPTY flag goes HIGH). A HIGH on this signal inhibits the counting.

FSTOP is used to stop the FIFO write counter from further counting once the FIFO is full (i.e., the FULL flag goes HIGH). A HIGH on this signal inhibits the counting.

For more information on these signals, refer to the "ESTOP and FSTOP Usage" section.

FULL, EMPTY

When the FIFO is full and no more data can be written, the FULL flag asserts HIGH. The FULL flag is synchronous to WCLK to inhibit writing immediately upon detection of a full condition and to prevent overflows. Since the write address is compared to a resynchronized (and thus time-delayed) version of the read address, the FULL flag will remain asserted until two WCLK active edges after a read operation eliminates the full condition.

When the FIFO is empty and no more data can be read, the EMPTY flag asserts HIGH. The EMPTY flag is synchronous to RCLK to inhibit reading immediately upon detection of an empty condition and to prevent underflows. Since the read address is compared to a resynchronized (and thus time-delayed) version of the write address, the EMPTY flag will remain asserted until two RCLK active edges after a write operation removes the empty condition.

For more information on these signals, refer to the "FIFO Flag Usage Considerations" section on page 161.

AFULL, AEMPTY

These are programmable flags and will be asserted on the threshold specified by AFVAL and AEVAL, respectively.

When the number of words stored in the FIFO reaches the amount specified by AEVAL while reading, the AEMPTY output will go HIGH. Likewise, when the number of words stored in the FIFO reaches the amount specified by AFVAL while writing, the AFULL output will go HIGH.

AFVAL, AEVAL

The AEVAL and AFVAL pins are used to specify the almost-empty and almost-full threshold values. They are 12-bit signals. For more information on these signals, refer to the "FIFO Flag Usage Considerations" section on page 161.

FIFO Usage

ESTOP and FSTOP Usage

The ESTOP pin is used to stop the read counter from counting any further once the FIFO is empty (i.e., the EMPTY flag goes HIGH). Likewise, the FSTOP pin is used to stop the write counter from counting any further once the FIFO is full (i.e., the FULL flag goes HIGH).

The FIFO counters in the device start the count at zero, reach the maximum depth for the configuration (e.g., 511 for a 512×9 configuration), and then restart at zero. An example application for ESTOP, where the read counter keeps counting, would be writing to the FIFO once and reading the same content over and over without doing another write.

Instantiating in HDL code

All the supported I/O macros can be instantiated in the top-level HDL code (refer to the *IGLOO*, *ProASIC3*, *SmartFusion*, and *Fusion Macro Library Guide* for a detailed list of all I/O macros). The following is an example:

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3e;

entity TOP is
    port(IN2, IN1 : in std_logic; OUT1 : out std_logic);
end TOP;

architecture DEF_ARCH of TOP is

    component INBUF_LVCMOS5U
        port(PAD : in std_logic := 'U'; Y : out std_logic);
    end component;

    component INBUF_LVCMOS5
        port(PAD : in std_logic := 'U'; Y : out std_logic);
    end component;

    component OUTBUF_SSTL3_II
        port(D : in std_logic := 'U'; PAD : out std_logic);
    end component;

    Other component ....

    signal x, y, z,.....other signals : std_logic;

begin

    I1 : INBUF_LVCMOS5U
        port map(PAD => IN1, Y => x);
    I2 : INBUF_LVCMOS5
        port map(PAD => IN2, Y => y);
    I3 : OUTBUF_SSTL3_II
        port map(D => z, PAD => OUT1);

    other port mapping...

end DEF_ARCH;
```

Synthesizing the Design

Libero SoC integrates with the Synplify® synthesis tool. Other synthesis tools can also be used with Libero SoC. Refer to the *Libero SoC User's Guide* or Libero online help for details on how to set up the Libero tool profile with synthesis tools from other vendors.

During synthesis, the following rules apply:

- Generic macros:
 - Users can instantiate generic INBUF, OUTBUF, TRIBUF, and BIBUF macros.
 - Synthesis will automatically infer generic I/O macros.
 - The default I/O technology for these macros is LVTTTL.
 - Users will need to use the I/O Attribute Editor in Designer to change the default I/O standard if needed (see Figure 9-6 on page 259).
- Technology-specific I/O macros:
 - Technology-specific I/O macros, such as INBUF_LVCMO25 and OUTBUF_GTL25, can be instantiated in the design. Synthesis will infer these I/O macros in the netlist.

Types of Programming for Flash Devices

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

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

In-System Programming

Device Type Supported: Flash

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

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

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

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

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

- Advantages: Allows local control of programming and data files for maximum security. The programming algorithms and hardware are available from Microsemi. The only hardware required on the board is a programming header.
 - Limitations: A negligible board space requirement for the programming header and JTAG signal routing
 - Microprocessor ISP—Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.
- Using a microprocessor and an external or internal memory, you can store the program in memory and use the microprocessor to perform the programming. In Microsemi documentation, this is referred to as internal ISP. Both the code for the programming algorithm and the FPGA programming file must be stored in memory on the board. Programming voltages must also be generated on the board.
- Advantages: The programming code is stored in the system memory. An external programmer is not required during programming.
 - Limitations: This is the approach that requires the most design work, since some way of getting and/or storing the data is needed; a system interface to the device must be designed; and the low-level API to the programming firmware must be written and linked into the code provided by Microsemi. While there are benefits to this methodology, serious thought and planning should go into the decision.

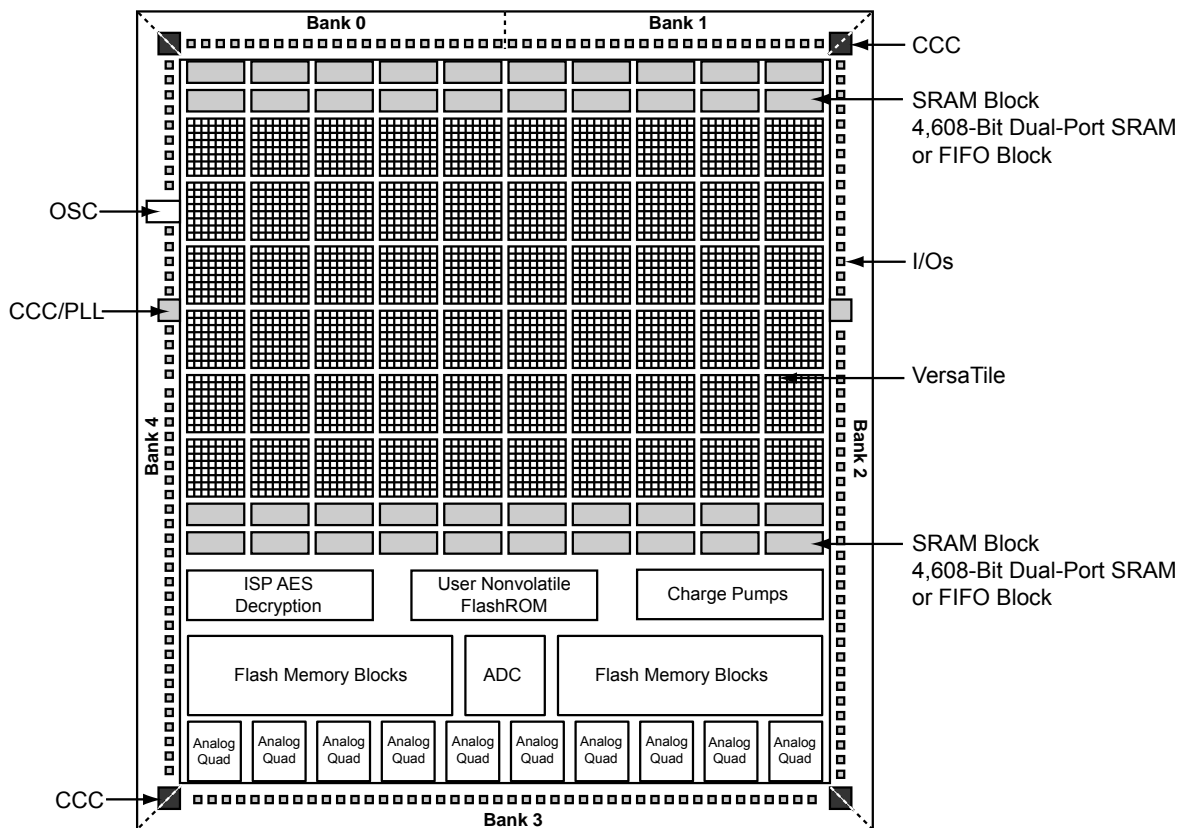


Figure 12-3 • Block Representation of the AES Decryption Core in a Fusion AFS600 FPGA

Security Features

IGLOO and ProASIC3 devices have two entities inside: FlashROM and the FPGA core fabric. Fusion devices contain three entities: FlashROM, FBs, and the FPGA core fabric. The parts can be programmed or updated independently with a STAPL programming file. The programming files can be AES-encrypted or plaintext. This allows maximum flexibility in providing security to the entire device. Refer to the "Programming Flash Devices" section on page 287 for information on the FlashROM structure.

Unlike SRAM-based FPGA devices, which require a separate boot PROM to store programming data, low power flash devices are nonvolatile, and the secured configuration data is stored in on-chip flash cells that are part of the FPGA fabric. Once programmed, this data is an inherent part of the FPGA array and does not need to be loaded at system power-up. SRAM-based FPGAs load the configuration bitstream upon power-up; therefore, the configuration is exposed and can be read easily.

The built-in FPGA core, FBs, and FlashROM support programming files encrypted with the 128-bit AES (FIPS-192) block ciphers. The AES key is stored in dedicated, on-chip flash memory and can be programmed before the device is shipped to other parties (allowing secure remote field updates).

Security in ARM-Enabled Low Power Flash Devices

There are slight differences between the regular flash devices and the ARM®-enabled flash devices, which have the M1 and M7 prefix.

The AES key is used by Microsemi and preprogrammed into the device to protect the ARM IP. As a result, the design is encrypted along with the ARM IP, according to the details below.

3. Choose the desired settings for the FlashROM configurations to be programmed (Figure 12-13). Click **Finish** to generate the STAPL programming file for the design.
-

Figure 12-13 • FlashROM Configuration Settings for Low Power Flash Devices

Generation of Security Header Programming File Only— Application 2

As mentioned in the "Application 2: Nontrusted Environment—Unsecured Location" section on page 309, the designer may employ FlashLock Pass Key protection or FlashLock Pass Key with AES encryption on the device before sending it to a nontrusted or unsecured location for device programming. To achieve this, the user needs to generate a programming file containing only the security settings desired (Security Header programming file).

Note: If AES encryption is configured, FlashLock Pass Key protection must also be configured.

The available security options are indicated in Table 12-4 and Table 12-5 on page 317.

Table 12-4 • FlashLock Security Options for IGLOO and ProASIC3

Security Option	FlashROM Only	FPGA Core Only	Both FlashROM and FPGA
No AES / no FlashLock	—	—	—
FlashLock only	✓	✓	✓
AES and FlashLock	✓	✓	✓

Table 12-6 and Table 12-7 show all available options. If you want to implement custom levels, refer to the "Advanced Options" section on page 322 for information on each option and how to set it.

3. When done, click **Finish** to generate the Security Header programming file.

Table 12-6 • All IGLOO and ProASIC3 Header File Security Options

Security Option	FlashROM Only	FPGA Core Only	Both FlashROM and FPGA
No AES / no FlashLock	✓	✓	✓
FlashLock only	✓	✓	✓
AES and FlashLock	✓	✓	✓

Note: ✓ = options that may be used

Table 12-7 • All Fusion Header File Security Options

Security Option	FlashROM Only	FPGA Core Only	FB Core Only	All
No AES / No FlashLock	✓	✓	✓	✓
FlashLock	✓	✓	✓	✓
AES and FlashLock	✓	✓	✓	✓

Generation of Programming Files with AES Encryption— Application 3

This section discusses how to generate design content programming files needed specifically at unsecured or remote locations to program devices with a Security Header (FlashLock Pass Key and AES key) already programmed ("Application 2: Nontrusted Environment—Unsecured Location" section on page 309 and "Application 3: Nontrusted Environment—Field Updates/Upgrades" section on page 310). In this case, the encrypted programming file must correspond to the AES key already programmed into the device. If AES encryption was previously selected to encrypt the FlashROM, FBs, and FPGA array, AES encryption must be set when generating the programming file for them. AES encryption can be applied to the FlashROM only, the FBs only, the FPGA array only, or all. The user must ensure both the FlashLock Pass Key and the AES key match those already programmed to the device(s), and all security settings must match what was previously programmed. Otherwise, the encryption and/or device unlocking will not be recognized when attempting to program the device with the programming file.

The generated programming file will be AES-encrypted.

In this scenario, generate the programming file as follows:

1. Deselect **Security settings** and select the portion of the device to be programmed (Figure 12-17 on page 320). Select **Programming previously secured device(s)**. Click **Next**.

ISP Programming Header Information

The FlashPro4/3/3X programming cable connector can be connected with a 10-pin, 0.1"-pitch programming header. The recommended programming headers are manufactured by AMP (103310-1) and 3M (2510-6002UB). If you have limited board space, you can use a compact programming header manufactured by Samtec (FTSH-105-01-L-D-K). Using this compact programming header, you are required to order an additional header adapter manufactured by Microsemi SoC Products Group (FP3-10PIN-ADAPTER-KIT).

Existing ProASIC^{PLUS} family customers who are using the Samtec Small Programming Header (FTSH-113-01-L-D-K) and are planning to migrate to IGLOO or ProASIC3 devices can also use FP3-10PIN-ADAPTER-KIT.

Table 13-3 • Programming Header Ordering Codes

Manufacturer	Part Number	Description
AMP	103310-1	10-pin, 0.1"-pitch cable header (right-angle PCB mount angle)
3M	2510-6002UB	10-pin, 0.1"-pitch cable header (straight PCB mount angle)
Samtec	FTSH-113-01-L-D-K	Small programming header supported by FlashPro and Silicon Sculptor
Samtec	FTSH-105-01-L-D-K	Compact programming header
Samtec	FFSD-05-D-06.00-01-N	10-pin cable with 50 mil pitch sockets; included in FP3-10PIN-ADAPTER-KIT.
Microsemi	FP3-10PIN-ADAPTER-KIT	Transition adapter kit to allow FP3 to be connected to a micro 10-pin header (50 mil pitch). Includes a 6 inch Samtec FFSD-05-D-06.00-01-N cable in the kit. The transition adapter board was previously offered as FP3-26PIN-ADAPTER and includes a 26-pin adapter for design transitions from ProASIC ^{PLUS} based boards to ProASIC3 based boards.

TCK	1	2	GND
TDO	3	4	NC (FlashPro3/3X); Prog_Mode* (FlashPro4)
TMS	5	6	VJTAG
VPUMP	7	8	TRST
TDI	9	10	GND

*Note: *Prog_Mode on FlashPro4 is an output signal that goes High during device programming and returns to Low when programming is complete. This signal can be used to drive a system to provide a 1.5 V programming signal to IGLOO nano, ProASIC3L, and RT ProASIC3 devices that can run with 1.2 V core voltage but require 1.5 V for programming. IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only), but IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V.*

Figure 13-5 • Programming Header (top view)

Microprocessor Programming Support in Flash Devices

The flash-based FPGAs listed in Table 15-1 support programming with a microprocessor 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 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 15-1. Where the information applies to only one device 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 device 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*.

Programming Algorithm

JTAG Interface

The low power flash families are fully compliant with the IEEE 1149.1 (JTAG) standard. They support all the mandatory boundary scan instructions (EXTEST, SAMPLE/PRELOAD, and BYPASS) as well as six optional public instructions (USERCODE, IDCODE, HIGHZ, and CLAMP).

IEEE 1532

The low power flash families are also fully compliant with the IEEE 1532 programming standard. The IEEE 1532 standard adds programming instructions and associated data registers to devices that comply with the IEEE 1149.1 standard (JTAG). These instructions and registers extend the capabilities of the IEEE 1149.1 standard such that the Test Access Port (TAP) can be used for configuration activities. The IEEE 1532 standard greatly simplifies the programming algorithm, reducing the amount of time needed to implement microprocessor ISP.

Implementation Overview

To implement device programming with a microprocessor, the user should first download the C-based STAPL player or DirectC code from the Microsemi SoC Products Group website. Refer to the website for future updates regarding the STAPL player and DirectC code.

http://www.microsemi.com/soc/download/program_debug/stapl/default.aspx

http://www.microsemi.com/soc/download/program_debug/directc/default.aspx

Using the easy-to-follow user's guide, create the low-level application programming interface (API) to provide the necessary basic functions. These API functions act as the interface between the programming software and the actual hardware (Figure 15-2).

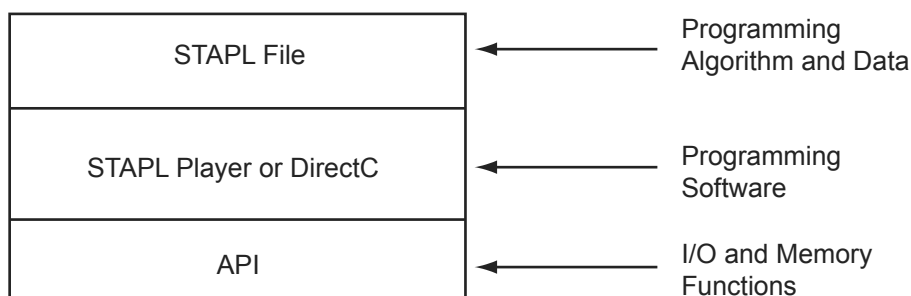


Figure 15-2 • Device Programming Code Relationship

The API is then linked with the STAPL player or DirectC and compiled using the microprocessor's compiler. Once the entire code is compiled, the user must download the resulting binary into the MCU system's program memory (such as ROM, EEPROM, or flash). The system is now ready for programming.

To program a design into the FPGA, the user creates a bitstream or STAPL file using the Microsemi Designer software, downloads it into the MCU system's volatile memory, and activates the stored programming binary file (Figure 15-3 on page 352). Once the programming is completed, the bitstream or STAPL file can be removed from the system, as the configuration profile is stored in the flash FPGA fabric and does not need to be reloaded at every system power-on.

FlashLock
 IGLOO and ProASIC devices 307
 permanent 307

FlashROM
 access using JTAG port 139
 architecture 333
 architecture of user nonvolatile 133
 configuration 136
 custom serialization 145
 design flow 140
 generation 141
 programming and accessing 138
 programming file 143
 programming files 333
 SmartGen 142
 FlashROM read-back 371

G
 global architecture 47
 global buffers
 no programmable delays 80
 with PLL function 83
 with programmable delays 80
 global macros
 Synplicity 66
 globals
 designer flow 69
 networks 74
 spines and rows 57

H
 HLD code
 instantiating 258
 hot-swapping 383

I
 I/O banks
 standards 56
 I/O standards 93
 global macros 62
 I/Os
 assigning technologies 264
 assignments defined in PDC file 259
 automatically assigning 268
 behavior at power-up/-down 377
 buffer schematic cell 257
 cell architecture 273
 configuration with SmartGen 254
 global, naming 51
 manually assigning technologies 264
 software-controlled attributes 253
 user I/O assignment flow chart 251
 idle mode 23
 INBUF_FF 39
 ISP 289, 290
 architecture 327
 board-level considerations 337

circuit 343
 microprocessor 349

J
 JTAG 1532 327
 JTAG interface 351

L
 layout
 device-specific 94
 LTC3025 linear voltage regulator 343

M
 MAC validation/authentication 354
 macros
 CLKBUF 93
 CLKBUF_LVDS/LVPECL 93
 CLKDLY 81, 89
 FIFO4KX18 157
 PLL 89
 PLL macro signal descriptions 84
 RAM4K9 153
 RAM512X18 155
 supported basic RAM macros 152
 UJTAG 365
 ULSICC 40
 MCU FPGA programming model 352
 memory availability 162
 memory blocks 151
 microprocessor programming 349
 Microsemi SoC Products Group
 email 387
 web-based technical support 387
 website 387

O
 OTP 289

P
 PDC
 global promotion and demotion 67
 place-and-route 259
 PLL
 behavior at brownout condition 381
 configuration bits 106
 core specifications 100
 dynamic PLL configuration 103
 functional description 101
 power supply decoupling scheme 128
 PLL block signals 84
 PLL macro block diagram 85
 power conservation 41
 power modes
 Flash*Freeze 24
 idle 23
 shutdown 32