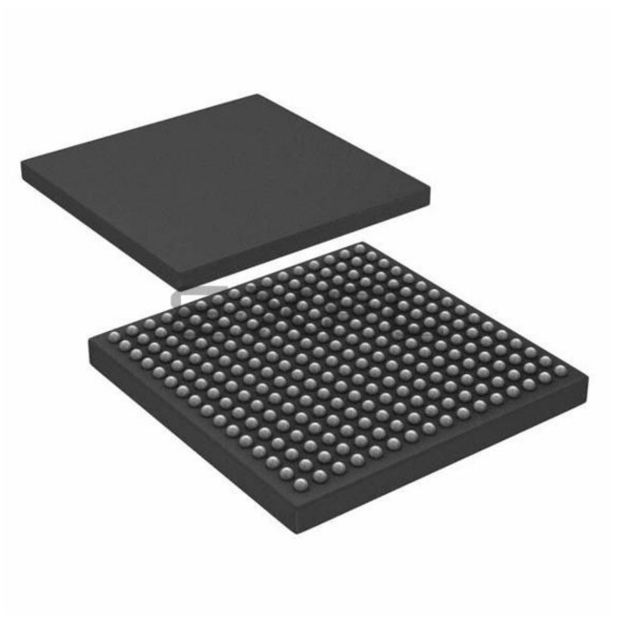


Welcome to E-XFL.COM

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	157
Number of Gates	250000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	256-LBGA
Supplier Device Package	256-FPBGA (17x17)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/a3p250l-1fg256

1 – FPGA Array Architecture in Low Power Flash Devices

Device Architecture

Advanced Flash Switch

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

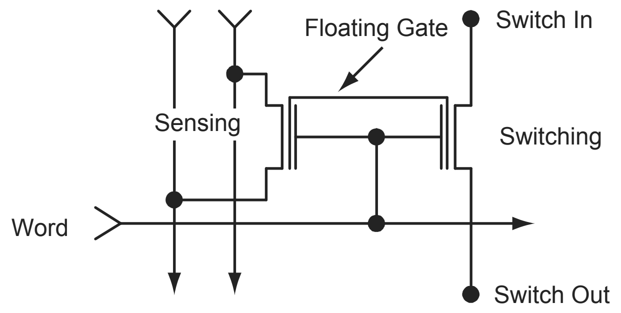


Figure 1-1 **• Flash-Based Switch**

Low Power Modes Overview

Table 2-2 summarizes the low power modes that achieve power consumption reduction when the FPGA or system is idle.

Table 2-2 • **Power Modes Summary**

Mode		**VCCI**	**VCC**	**Core**	**Clocks**	**ULSICC Macro**	**To Enter Mode**	**To Resume Operation**	**Trigger**
Active		On	On	On	On	N/A	Initiate clock	None	–
Static	Idle	On	On	On	Off	N/A	Stop clock	Initiate clock	External
	Flash*Freeze type 1	On	On	On	On*	N/A	Assert FF pin	Deassert FF pin	External
	Flash*Freeze type 2	On	On	On	On*	Used to enter Flash*Freeze mode	Assert FF pin and assert LSICC	Deassert FF pin	External
Sleep		On	Off	Off	Off	N/A	Shut down VCC	Turn on VCC supply	External
Shutdown		Off	Off	Off	Off	N/A	Shut down VCC and VCCI supplies	Turn on VCC and VCCI supplies	External

* *External clocks can be left toggling while the device is in Flash*Freeze mode. Clocks generated by the embedded PLL will be turned off automatically.*

Static (Idle) Mode

In Static (Idle) mode, none of the clock inputs is switching, and static power is the only power consumed by the device. This mode can be achieved by switching off the incoming clocks to the FPGA, thus benefitting from reduced power consumption. In addition, I/Os draw only minimal leakage current. In this mode, embedded SRAM, I/Os, and registers retain their values so the device can enter and exit this mode just by switching the clocks on or off.

If the device-embedded PLL is used as the clock source, Static (Idle) mode can easily be entered by pulling the PLL POWERDOWN pin LOW (active Low), which will turn off the PLL.

Flash*Freeze Mode Device Behavior

*Entering Flash*Freeze Mode*

- IGLOO, IGLOO nano, IGLOO PLUS, ProASCI3L, and RT ProASIC3 devices are designed and optimized to enter Flash*Freeze mode only when power supplies are stable. If the device is being powered up while the FF pin is asserted (Flash*Freeze mode type 1), or while both FF pin and LSICC signal are asserted (Flash*Freeze mode type 2), the device is expected to enter Flash*Freeze mode within 5 µs after the I/Os and FPGA core have reached their activation levels.
- If the device is already powered up when the FF pin is asserted, the device will enter Flash*Freeze mode within 1 µs (type 1). In Flash*Freeze mode type 2 operation, entering Flash*Freeze mode is completed within 1 µs after both FF pin and LSICC signal are asserted. Exiting Flash*Freeze mode is completed within 1 µs after deasserting the FF pin only.

PLLs

- If an embedded PLL is used, entering Flash*Freeze mode will automatically power down the PLL.
- The PLL output clocks will stop toggling within 1 µs after the assertion of the FF pin in type 1, or after both FF pin and LSICC signal are asserted in type 2. At the same time, I/Os will transition into the state specified in Table 2-6 on page 29. The user design must ensure it is safe to enter Flash*Freeze mode.

I/Os and Globals

- While entering Flash*Freeze mode, inputs, globals, and PLLs will enter their Flash*Freeze state asynchronously to each other. As a result, clock and data glitches and narrow pulses may be generated while entering Flash*Freeze mode, as shown in Figure 2-5.

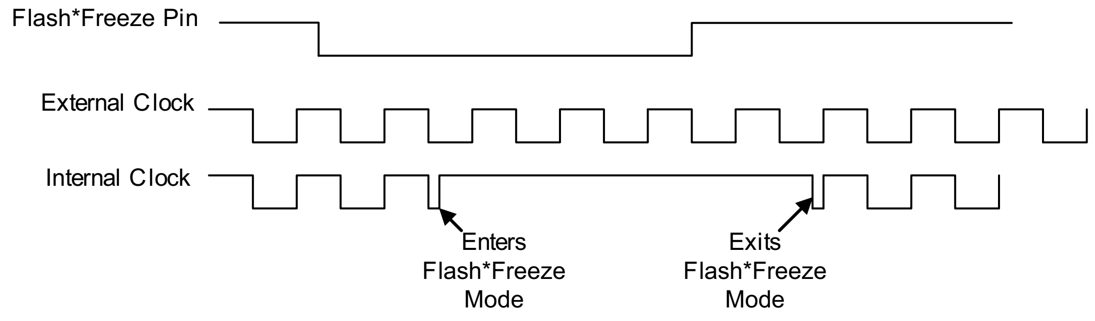


***Figure 2-5* • Narrow Clock Pulses During Flash*Freeze Entrance and Exit**

- I/O banks are not all deactivated simultaneously when entering Flash*Freeze mode. This can cause clocks and inputs to become disabled at different times, resulting in unexpected data being captured.
- Upon entering Flash*Freeze mode, all inputs and globals become tied High internally (except when an input hold state is used on IGLOO nano or IGLOO PLUS devices). If any of these signals are driven Low or tied Low externally, they will experience a Low to High transition internally when entering Flash*Freeze mode.
- Upon entering type 2 Flash*Freeze mode, ensure the LSICC signal (active High) does not de-assert. This can prevent the device from entering Flash*Freeze mode.
- Asynchronous input to output paths may experience output glitches. For example, on a direct in-to-out path, if the current state is '0' and the input bank turns off first, the input and then the output will transition to '1' before the output enters its Flash*Freeze state. This can be prevented by using latches in asynchronous in-to-out paths.
- The above situations can cause glitches or invalid data to be clocked into and preserved in the device. Refer to the "Flash*Freeze Design Guide" section on page 34 for solutions.

During Flash*Freeze Mode

- PLLs are turned off during Flash*Freeze mode.
- I/O pads are configured according to Table 2-5 on page 28 and Table 2-6 on page 29.
- Inputs and input clocks to the FPGA can toggle without any impact on static power consumption, assuming weak pull-up or pull-down is not selected.
- If weak pull-up or pull-down is selected and the input is driven to the opposite direction, power dissipation will occur.
- Any toggling signals will be charging and discharging the package pin capacitance.
- IGLOO and ProASIC3L outputs will be tristated unless the I/O is configured with weak pull-up or pull-down. The output of the I/O to the FPGA core is logic High regardless of whether the I/O pin is configured with a weak pull-up or pull-down. Refer to Table 2-5 on page 28 for more information.
- IGLOO nano and IGLOO PLUS output behavior will be based on the configuration defined by the user. Refer to Table 2-6 on page 29 for a description of output behavior during Flash*Freeze mode.
- The JTAG circuit is active; however, JTAG operations, such as JTAG commands, JTAG bypass, programming, and authentication, cannot be executed. The device must exit Flash*Freeze mode before JTAG commands can be sent. TCK should be static to avoid extra power consumption from the JTAG state machine.
- The FF pin must be externally asserted for the device to stay in Flash*Freeze mode.
- The FF pin is still active; i.e., the pin is used to exit Flash*Freeze mode when deasserted.

Exiting Flash*Freeze Mode

I/Os and Globals

- While exiting Flash*Freeze mode, inputs and globals will exit their Flash*Freeze state asynchronously to each other. As a result, clock and data glitches and narrow pulses may be generated while exiting Flash*Freeze mode, unless clock gating schemes are used.
- I/O banks are not all activated simultaneously when exiting Flash*Freeze mode. This can cause clocks and inputs to become enabled at different times, resulting in unexpected data being captured.
- Upon exiting Flash*Freeze mode, inputs and globals will no longer be tied High internally (does not apply to input hold state on IGLOO nano and IGLOO PLUS). If any of these signals are driven Low or tied Low externally, they will experience a High-to-Low transition internally when exiting Flash*Freeze mode.
- Applies only to IGLOO nano and IGLOO PLUS: Output hold state is asynchronously controlled by the signal driving the output buffer (output signal). This ensures a clean, glitch-free transition from hold state to output drive. However, any glitches on the output signal during exit from Flash*Freeze mode may result in glitches on the output pad.
- The above situations can cause glitches or invalid data to be clocked into and preserved in the device. Refer to the "Flash*Freeze Design Guide" on page 34 for solutions.

PLLs

- If the embedded PLL is used, the design must allow maximum acquisition time (per device datasheet) for the PLL to acquire the lock signal.

Flash*Freeze Pin Locations

Refer to the Pin Descriptions and Packaging chapter of specific device datasheets for information regarding Flash*Freeze pin location on the available packages. The Flash*Freeze pin location is independent of the device, allowing migration to larger or smaller devices while maintaining the same pin location on the board.

Date	Changes	Page
v1.2 (continued)	Figure 2-3 • Flash*Freeze Mode Type 2 – Controlled by Flash*Freeze Pin and Internal Logic (LSICC signal) was updated.	27
	Figure 2-4 • Flash*Freeze Mode Type 2 – Timing Diagram was revised to show deasserting LSICC after the device has exited Flash*Freeze mode.	27
	The "IGLOO nano and IGLOO PLUS I/O State in Flash*Freeze Mode" section was added to include information for IGLOO PLUS devices. Table 2-6 • IGLOO nano and IGLOO PLUS Flash*Freeze Mode (type 1 and type 2)—I/O Pad State is new.	28, 29
	The "During Flash*Freeze Mode" section was revised to include a new bullet pertaining to output behavior for IGLOO PLUS. The bullet on JTAG operation was revised to provide more detail.	31
	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 updated to include IGLOO PLUS.	33, 33
	The first sentence of the "Shutdown Mode" section was updated to list the devices for which it is supported.	32
	The first paragraph of the "Power-Up/-Down Behavior" section was revised. The second sentence was changed to, "The I/Os remain tristated until the last voltage supply (V_{CC} or V_{CCI}) is powered to its activation level." The word "activation" replaced the word "functional." The sentence, "During power-down, device I/Os become tristated once the first power supply (V_{CC} or V_{CCI}) drops below its deactivation voltage level" was revised. The word "deactivation" replaced the word "brownout."	33
	The "Prototyping for IGLOO and ProASIC3L Devices Using ProASIC3" section was revised to state that prototyping in ProASIC3 does not apply for the IGLOO PLUS family.	2-21
	Table 2-8 • Prototyping/Migration Solutions, Table 2-9 • Device Migration—IGLOO Supported Packages in ProASIC3 Devices, and Table 2-10 • Device Migration—ProASIC3L Supported Packages in ProASIC3 Devices were updated with a table note stating that device migration is not supported for IGLOO PLUS devices.	2-21, 2-23
	The text following Table 2-10 • Device Migration—ProASIC3L Supported Packages in ProASIC3 Devices was moved to a new section: the "Flash*Freeze Design Guide" section.	34
v1.1 (February 2008)	Table 2-1 • Flash-Based FPGAs was updated to remove the ProASIC3, ProASIC3E, and Automotive ProASIC3 families, which were incorrectly included.	22
v1.0 (January 2008)	Detailed descriptions of low power modes are described in the advanced datasheets. This application note was updated to describe how to use the features in an IGLOO/e application.	N/A
	Figure 2-1 • Flash*Freeze Mode Type 1 – Controlled by the Flash*Freeze Pin was updated.	25
	Figure 2-2 • Flash*Freeze Mode Type 1 – Timing Diagram is new.	25
	Steps 4 and 5 are new in the "Flash*Freeze Type 2: Control by Dedicated Flash*Freeze Pin and Internal Logic" section.	26

During Layout, Designer will assign two of the signals to quadrant global locations.

Step 3 (optional)

You can also assign the QCLK1_c and QCLK2_c nets to quadrant regions using the following PDC commands:

```
assign_local_clock –net QCLK1_c  –type quadrant UL
assign_local_clock –net QCLK2_c  –type quadrant LL
```

Step 4

Import this PDC with the netlist and run Compile again. You will see the following in the Compile report:

```
The following nets have been assigned to a global resource:
Fanout  Type          Name
--------------------------
1536    INT_NET       Net   : EN_ALL_c
                      Driver: EN_ALL_pad_CLKINT
                      Source: AUTO PROMOTED
1536    SET/RESET_NET Net   : ACLR_c
                      Driver: ACLR_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : QCLK3_c
                      Driver: QCLK3_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : $1N14
                      Driver: $1I5/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N12
                      Driver: $1I6/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N10
                      Driver: $1I6/Core
                      Source: ESSENTIAL
The following nets have been assigned to a quadrant clock resource using PDC:
Fanout  Type          Name
--------------------------
256     CLK_NET       Net   : QCLK1_c
                      Driver: QCLK1_pad_CLKINT
                      Region: quadrant_UL
256     CLK_NET       Net   : QCLK2_c
                      Driver: QCLK2_pad_CLKINT
                      Region: quadrant_LL
```

Step 5

Run Layout.

Global Management in PLL Design

This section describes the legal global network connections to PLLs in the low power flash devices. For detailed information on using PLLs, refer to "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 77. Microsemi recommends that you use the dedicated global pins to directly drive the reference clock input of the associated PLL for reduced propagation delays and clock distortion. However, low power flash devices offer the flexibility to connect other signals to reference clock inputs. Each PLL is associated with three global networks (Figure 3-5 on page 52). There are some limitations, such as when trying to use the global and PLL at the same time:

- If you use a PLL with only primary output, you can still use the remaining two free global networks.
- If you use three globals associated with a PLL location, you cannot use the PLL on that location.
- If the YB or YC output is used standalone, it will occupy one global, even though this signal does not go to the global network.

This section outlines the following device information: CCC features, PLL core specifications, functional descriptions, software configuration information, detailed usage information, recommended board-level considerations, and other considerations concerning global networks in low power flash devices.

Clock Conditioning Circuits with Integrated PLLs

Each of the CCCs with integrated PLLs includes the following:

- 1 PLL core, which consists of a phase detector, a low-pass filter, and a four-phase voltage-controlled oscillator
- 3 global multiplexer blocks that steer signals from the global pads and the PLL core onto the global networks
- 6 programmable delays and 1 fixed delay for time advance/delay adjustments
- 5 programmable frequency divider blocks to provide frequency synthesis (automatically configured by the SmartGen macro builder tool)

Clock Conditioning Circuits without Integrated PLLs

There are two types of simplified CCCs without integrated PLLs in low power flash devices.

1. The simplified CCC with programmable delays, which is composed of the following:
   - 3 global multiplexer blocks that steer signals from the global pads and the programmable delay elements onto the global networks
   - 3 programmable delay elements to provide time delay adjustments
2. The simplified CCC (referred to as CCC-GL) without programmable delay elements, which is composed of the following:
   - A global multiplexer block that steer signals from the global pads onto the global networks

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.

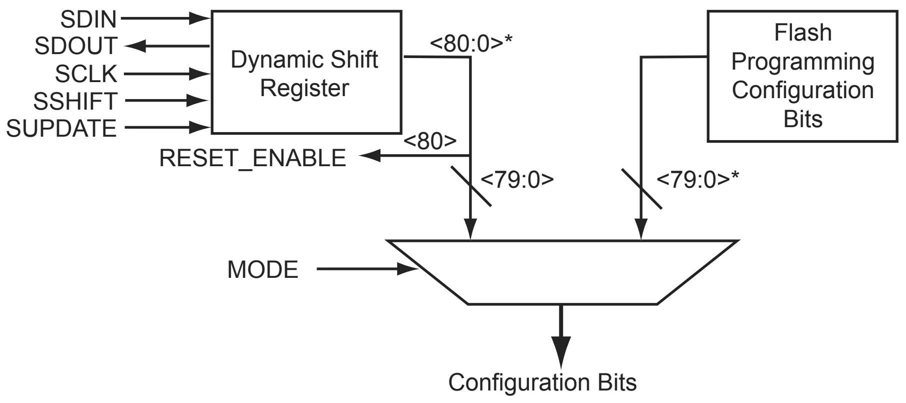
Flash Configuration Bits

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.

Recommended Board-Level Considerations

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

PLL Power Supply Decoupling Scheme

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

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

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

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

Recommended 100 nF capacitor:

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

Recommended 10 nF capacitor:

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

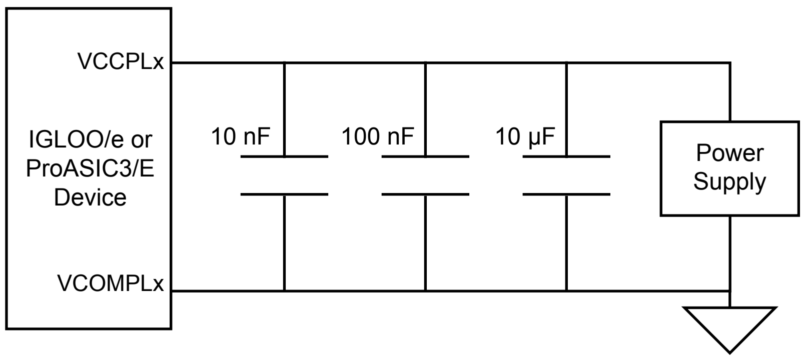


Figure 4-38 • **Decoupling Scheme for One PLL (should be replicated for each PLL used)**

Example of RAM Initialization

This section of the document presents a sample design in which a 4×4 RAM block is being initialized through the JTAG port. A test feature has been implemented in the design to read back the contents of the RAM after initialization to verify the procedure.

The interface block of this example performs two major functions: initialization of the RAM block and running a test procedure to read back the contents. The clock output of the interface is either the write clock (for initialization) or the read clock (for reading back the contents). The Verilog code for the interface block is included in the "Sample Verilog Code" section on page 167.

For simulation purposes, users can declare the input ports of the UJTAG macro for easier assignment in the testbench. However, the UJTAG input ports should not be declared on the top level during synthesis. If the input ports of the UJTAG are declared during synthesis, the synthesis tool will instantiate input buffers on these ports. The input buffers on the ports will cause Compile to fail in Designer.

Figure 6-10 shows the simulation results for the initialization step of the example design.

The CLK_OUT signal, which is the clock output of the interface block, is the inverted DR_UPDATE output of the UJTAG macro. It is clear that it gives sufficient time (while the TAP Controller is in the Data Register Update state) for the write address and data to become stable before loading them into the RAM block.

Figure 6-11 presents the test procedure of the example. The data read back from the memory block matches the written data, thus verifying the design functionality.

Figure 6-10 **• Simulation of Initialization Step**

Figure 6-11 **• Simulation of the Test Procedure of the Example**

The ROM emulation application is based on RAM block initialization. If the user's main design has access only to the read ports of the RAM block (RADDR, RD, RCLK, and REN), and the contents of the RAM are already initialized through the TAP, then the memory blocks will emulate ROM functionality for the core design. In this case, the write ports of the RAM blocks are accessed only by the user interface block, and the interface is activated only by the TAP Instruction Register contents.

Users should note that the contents of the RAM blocks are lost in the absence of applied power. However, the 1 kbit of flash memory, FlashROM, in low power flash devices can be used to retain data after power is removed from the device. Refer to the "SRAM and FIFO Memories in Microsemi's Low Power Flash Devices" section on page 147 for more information.

Sample Verilog Code

Interface Block

```
`define Initialize_start 8'h22 //INITIALIZATION START COMMAND VALUE
`define Initialize_stop 8'h23 //INITIALIZATION START COMMAND VALUE

module interface(IR, rst_n, data_shift, clk_in, data_update, din_ser, dout_ser, test,
  test_out,test_clk,clk_out,wr_en,rd_en,write_word,read_word,rd_addr, wr_addr);

input [7:0] IR;
input [3:0] read_word; //RAM DATA READ BACK
input rst_n, data_shift, clk_in, data_update, din_ser; //INITIALIZATION SIGNALS
input test, test_clk; //TEST PROCEDURE CLOCK AND COMMAND INPUT
output [3:0] test_out; //READ DATA
output [3:0] write_word; //WRITE DATA
output [1:0] rd_addr; //READ ADDRESS
output [1:0] wr_addr; //WRITE ADDRESS
output dout_ser; //TDO DRIVER
output clk_out, wr_en, rd_en;

wire [3:0] write_word;
wire [1:0] rd_addr;
wire [1:0] wr_addr;
wire [3:0] Q_out;
wire enable, test_active;

reg clk_out;

//SELECT CLOCK FOR INITIALIZATION OR READBACK TEST
always @(enable or test_clk or data_update)
begin
  case ({test_active})
    1 : clk_out = test_clk ;
    0 : clk_out = !data_update;
    default : clk_out = 1'b1;
  endcase
end

assign test_active = test && (IR == 8'h23);
assign enable = (IR == 8'h22);
assign wr_en = !enable;
assign rd_en = !test_active;
assign test_out = read_word;
assign dout_ser = Q_out[3];

//4-bit SIN/POUT SHIFT REGISTER
shift_reg data_shift_reg (.Shiften(data_shift), .Shiftin(din_ser), .Clock(clk_in),
  .Q(Q_out));

//4-bit PIPELINE REGISTER
D_pipeline pipeline_reg (.Data(Q_out), .Clock(data_update), .Q(write_word));
```

I/O Banks

Advanced I/Os are divided into multiple technology banks. Each device has two to four banks, and the number of banks is device-dependent as described above. The bank types have different characteristics, such as drive strength, the I/O standards supported, and timing and power differences.

There are three types of banks: Advanced I/O banks, Standard Plus I/O banks, and Standard I/O banks.

Advanced I/O banks offer single-ended and differential capabilities. These banks are available on the east and west sides of 250K, 400K, 600K, and 1M gate devices.

Standard Plus I/O banks offer LVTTL/LVCMOS and PCI single-ended I/O standards. These banks are available on the north and south sides of 250K, 400K, 600K, and 1M gate devices as well as all sides of 125K and 60K devices.

Standard I/O banks offer LVTTL/LVCMOS single-ended I/O standards. These banks are available on all sides of 30K gate devices.

Table 7-4 shows the I/O bank types, devices and bank locations supported, drive strength, slew rate control, and supported standards.

All inputs and disabled outputs are voltage-tolerant up to 3.3 V.

For more information about I/O and global assignments to I/O banks in a device, refer to the specific pin table for the device in the packaging section of the datasheet and the "User I/O Naming Convention" section on page 206.

Table 7-4 **• IGLOO and ProASIC3 Bank Type Definitions and Differences**

			I/O Standards Supported		
I/O Bank Type	**Device and Bank Location**	**Drive Strength**	**LVTTL/ LVCMOS**	**PCI/PCI-X**	**LVPECL, LVDS, B-LVDS, M-LVDS**
Standard	30 k gate devices (all banks)	Refer to Table 7-14 on page 203	✓	Not Supported	Not Supported
Standard Plus	60 k and 125 k gate devices (all banks)	Refer to Table 7-15 on page 203	✓	✓	Not Supported
	North and south banks of 250 k and 1 M gate devices	Refer to Table 7-15 on page 203	✓	✓	Not Supported
Advanced	East and west banks of 250 k and 1 M gate devices	Refer to Table 7-16 on page 203	✓	✓	✓

Example: For a bus consisting of 20 equidistant loads, the terminations given in EQ 1 provide the required differential voltage, in worst-case industrial operating conditions, at the farthest receiver:

$$R_S = 60\ \Omega,\ R_T = 70\ \Omega,\ \text{given } Z_O = 50\ \Omega\ (2") \text{ and } Z_{stub} = 50\ \Omega\ (\sim 1.5").$$

EQ 1

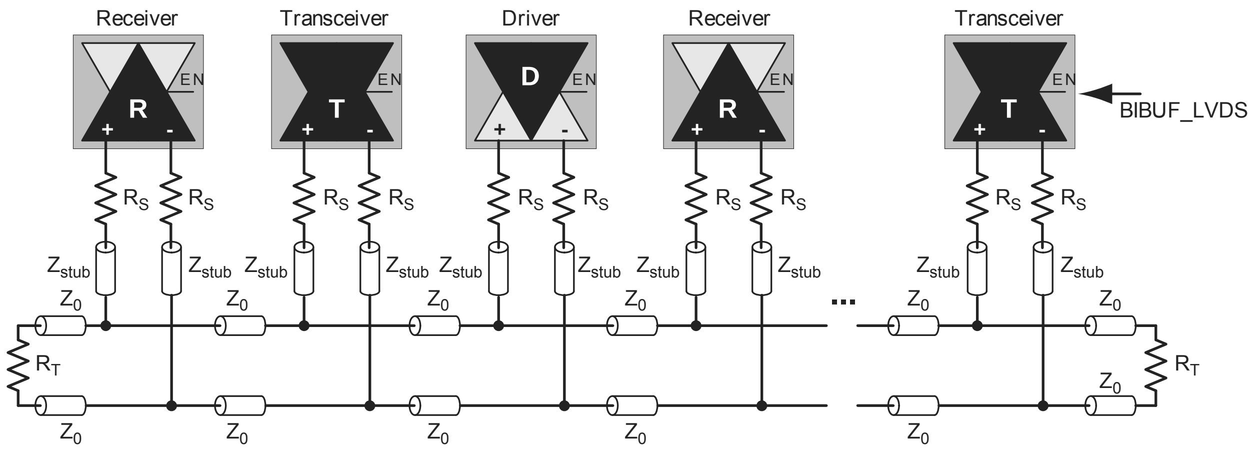
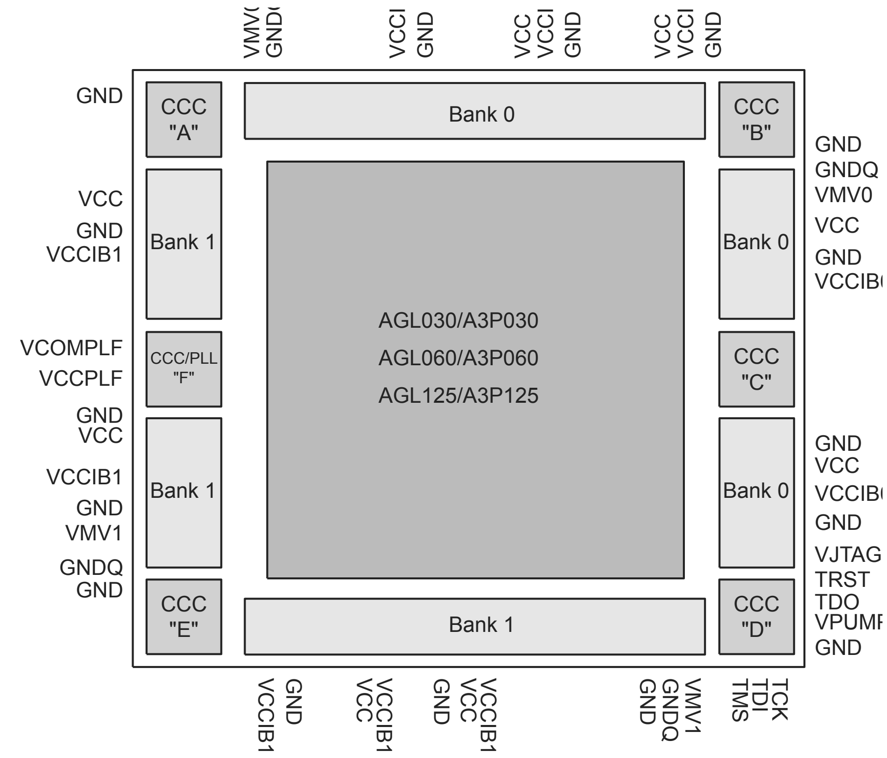


***Figure 7-8* • A B-LVDS/M-LVDS Multipoint Application Using LVDS I/O Buffers**



Note: *The 30 k gate devices do not support a PLL (V_{COMPLF} and V_{CCPLF} pins).*

Figure 7-19 **• Naming Conventions of IGLOO and ProASIC3 Devices with Two I/O Banks – Top View**

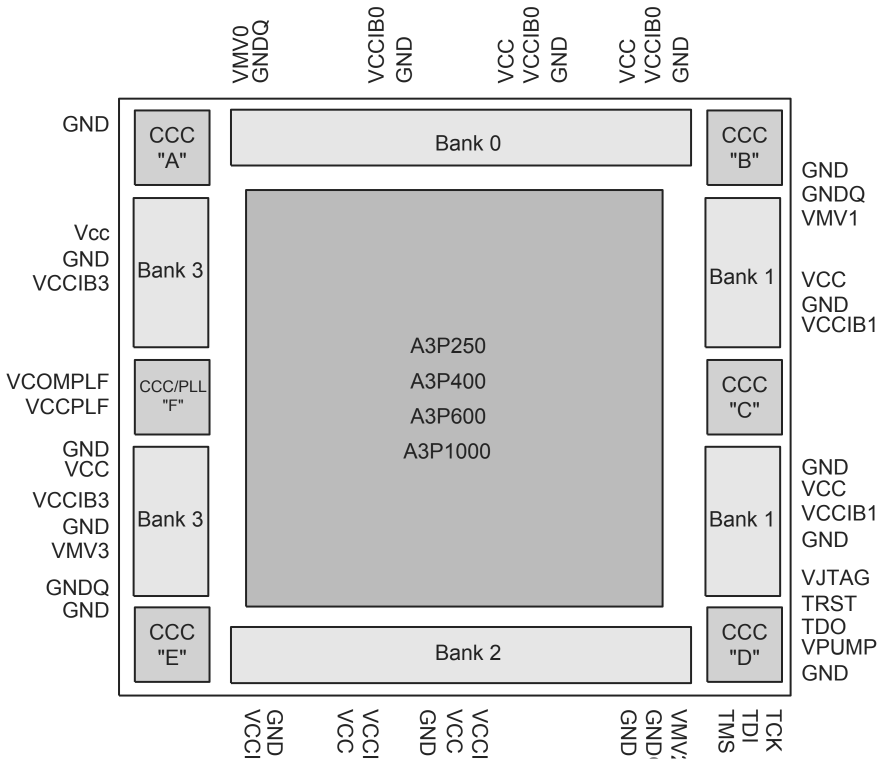


Figure 7-20 **• Naming Conventions of IGLOO and ProASIC3 Devices with Four I/O Banks – Top View**

Table 7-19 shows some high-level interfacing examples using low power flash devices.

***Table 7-19* • High-Level Interface Examples**

	Clock		**I/O**			
Interface	**Type**	**Frequency**	**Type**	**Signals In**	**Signals Out**	**Data I/O**
GM	Src Sync	125 MHz	LVTTL	8	8	125 Mbps
TBI	Src Sync	125 MHz	LVTTL	10	10	125 Mbps
XSBI	Src Sync	644 MHz	LVDS	16	16	644 Mbps
XGMI	Src Sync DDR	156 MHz	HSTL1	32	32	312 Mbps
FlexBus 3	Sys Sync	104 MHz	LVTTL	≤ 32	≤ 32	≤ 104
Pos-PHY3/SPI-3	Sys Sync	104	LVTTL	8, 16, 32	8, 16, 32	≤ 104 Mbps
FlexBus 4/SPI-4.1	Src Sync	200 MHz	HSTL1	16,64	16,64	200 Mbps
Pos-PHY4/SPI-4.2	Src Sync DDR	≥ 311 MHz	LVDS	16	16	≥ 622 Mbps
SFI-4.1	Src Sync	622 MHz	LVDS	16	16	622 Mbps
CSIX L1	Sys Sync	≤ 250 MHz	HSTL1	32,64,96,128	32,64,96,128	≤ 250 Mbps
Hyper Transport	Sys Sync DDR	≤ 800 MHz	LVDS	2,4,8,16	2,4,8,16	≤ 1.6 Gbps
Rapid I/O Parallel	Sys Sync DDR	250 MHz – 1 GHz	LVDS	8,16	8,16	≤ 2 Gbps
Star Fabric	CDR		LVDS	4	4	622 Mbps

Note: Sys Sync = System Synchronous Clocking, Src Sync = Source Synchronous Clocking, and CDR = Clock and Data Recovery.

Conclusion

IGLOO and ProASIC3 support for multiple I/O standards minimizes board-level components and makes possible a wide variety of applications. The Microsemi Designer software, integrated with Libero SoC, presents a clear visual display of I/O assignments, allowing users to verify I/O and board-level design requirements before programming the device. The IGLOO and ProASIC3 device I/O features and functionalities ensure board designers can produce low-cost and low power FPGA applications fulfilling the complexities of contemporary design needs.

I/O Standards

Single-Ended Standards

These I/O standards use a push-pull CMOS output stage with a voltage referenced to system ground to designate logical states. The input buffer configuration, output drive, and I/O supply voltage (V_{CCI}) vary among the I/O standards (Figure 8-6).

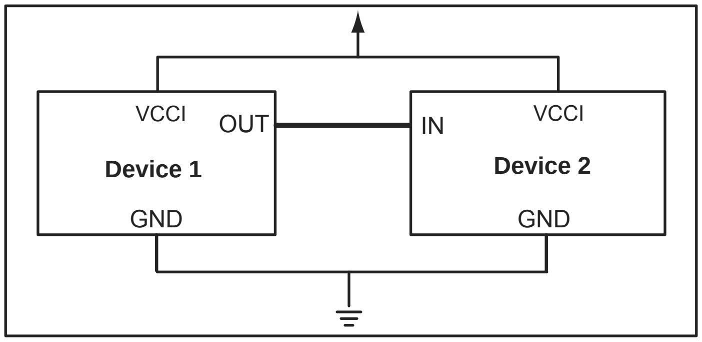


***Figure 8-6* • Single-Ended I/O Standard Topology**

The advantage of these standards is that a common ground can be used for multiple I/Os. This simplifies board layout and reduces system cost. Their low-edge-rate (*dv*/*dt*) data transmission causes less electromagnetic interference (EMI) on the board. However, they are not suitable for high-frequency (>200 MHz) switching due to noise impact and higher power consumption.

LVTTL (Low-Voltage TTL)

This is a general-purpose standard (EIA/JESD8-B) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. The LVTTL output buffer can have up to six different programmable drive strengths. The default drive strength is 12 mA. VCCI is 3.3 V. Refer to "I/O Programmable Features" on page 227 for details.

LVCMOS (Low-Voltage CMOS)

The low power flash devices provide four different kinds of LVCMOS: LVCMOS 3.3 V, LVCMOS 2.5 V, LVCMOS 1.8 V, and LVCMOS 1.5 V. LVCMOS 3.3 V is an extension of the LVCMOS standard (JESD8-B–compliant) used for general-purpose 3.3 V applications. LVCMOS 2.5 V is an extension of the LVCMOS standard (JESD8-5–compliant) used for general-purpose 2.5 V applications. LVCMOS 2.5 V for the 30 k gate devices has a clamp diode to VCCI, but for all other devices there is no clamp diode.

There is yet another standard supported by IGLOO and ProASIC3 devices (except A3P030): LVCMOS 2.5/5.0 V. This standard is similar to LVCMOS 2.5 V, with the exception that it can support up to 3.3 V on the input side (2.5 V output drive).

LVCMOS 1.8 V is an extension of the LVCMOS standard (JESD8-7–compliant) used for general-purpose 1.8 V applications. LVCMOS 1.5 V is an extension of the LVCMOS standard (JESD8-11–compliant) used for general-purpose 1.5 V applications.

The VCCI values for these standards are 3.3 V, 2.5 V, 1.8 V, and 1.5 V, respectively. Like LVTTL, the output buffer has up to seven different programmable drive strengths (2, 4, 6, 8, 12, 16, and 24 mA). Refer to "I/O Programmable Features" on page 227 for details.

3.3 V PCI (Peripheral Component Interface)

This standard specifies support for both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5 V–compliant for low power flash devices. It does not have programmable drive strength.

3.3 V PCI-X (Peripheral Component Interface Extended)

An enhanced version of the PCI specification, 3.3 V PCI-X can support higher average bandwidths; it increases the speed that data can move within a computer from 66 MHz to 133 MHz. It is backward-

5 V Input and Output Tolerance

IGLOO and ProASIC3 devices are both 5 V-input– and 5 V–output–tolerant if certain I/O standards are selected. Table 8-6 on page 218 shows the I/O standards that support 5 V input tolerance. Only 3.3 V LVTTL/LVCMOS standards support 5 V output tolerance. Refer to the appropriate family datasheet for detailed description and configuration information.

This feature is not shown in the I/O Attribute Editor.

5 V Input Tolerance

I/Os can support 5 V input tolerance when LVTTL 3.3 V, LVCMOS 3.3 V, LVCMOS 2.5 V, and LVCMOS 2.5 V / 5.0 V configurations are used (see Table 8-13 on page 231). There are four recommended solutions for achieving 5 V receiver tolerance (see Figure 8-10 on page 233 to Figure 8-13 on page 235 for details of board and macro setups). All the solutions meet a common requirement of limiting the voltage at the input to 3.6 V or less. In fact, the I/O absolute maximum voltage rating is 3.6 V, and any voltage above 3.6 V may cause long-term gate oxide failures.

Solution 1

The board-level design must ensure that the reflected waveform at the pad does not exceed the limits provided in the recommended operating conditions in the datasheet. This is a requirement to ensure long-term reliability.

This scheme will also work for a 3.3 V PCI/PCI-X configuration, but the internal diode should not be used for clamping, and the voltage must be limited by the two external resistors as explained below. Relying on the diode clamping would create an excessive pad DC voltage of 3.3 V + 0.7 V = 4 V.

This solution requires two board resistors, as demonstrated in Figure 8-10 on page 233. Here are some examples of possible resistor values (based on a simplified simulation model with no line effects and 10 Ω transmitter output resistance, where Rtx_out_high = [VCCI – VOH] / I_{OH} and Rtx_out_low = VOL / I_{OL}).

Example 1 (high speed, high current):

Rtx_out_high = Rtx_out_low = 10 Ω

R1 = 36 Ω (±5%), P(r1)min = 0.069 Ω

R2 = 82 Ω (±5%), P(r2)min = 0.158 Ω

Imax_tx = 5.5 V / (82 × 0.95 + 36 × 0.95 + 10) = 45.04 mA

t_{RISE} = t_{FALL} = 0.85 ns at C_pad_load = 10 pF (includes up to 25% safety margin)

t_{RISE} = t_{FALL} = 4 ns at C_pad_load = 50 pF (includes up to 25% safety margin)

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.

FlashROM Security Use Models

Each of the subsequent sections describes in detail the available selections in Microsemi Designer as an aid to understanding security applications and generating appropriate programming files for those applications. Before proceeding, it is helpful to review Figure 12-7 on page 309, which gives a general overview of the programming file generation flow within the Designer software as well as what occurs during the device programming stage. Specific settings are discussed in the following sections.

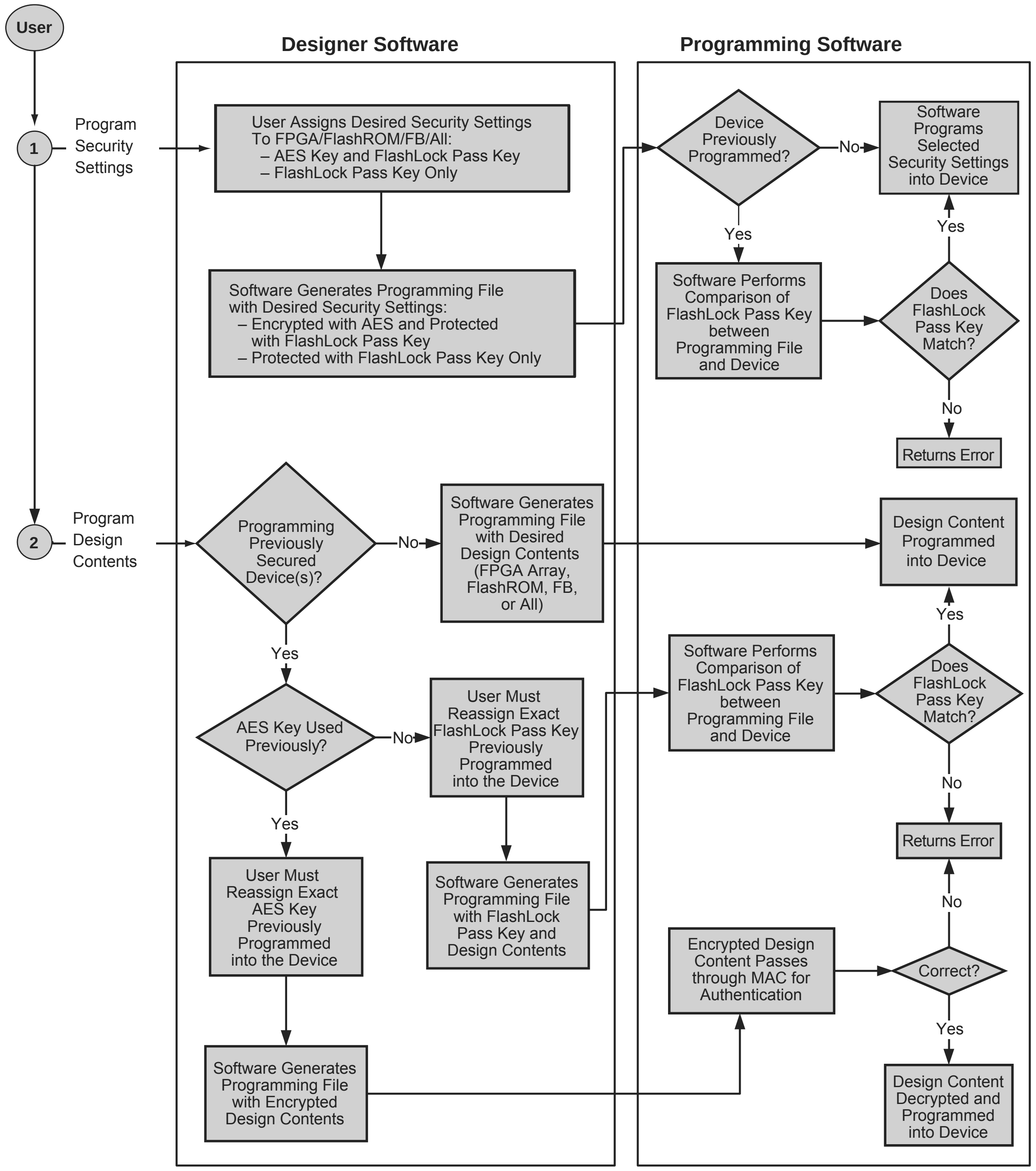
In Figure 12-7 on page 309, the flow consists of two sub-flows. Sub-flow 1 describes programming security settings to the device only, and sub-flow 2 describes programming the design contents only.

In Application 1, described in the "Application 1: Trusted Environment" section on page 309, the user does not need to generate separate files but can generate one programming file containing both security settings and design contents. Then programming of the security settings and design contents is done in one step. Both sub-flow 1 and sub-flow 2 are used.

In Application 2, described in the "Application 2: Nontrusted Environment—Unsecured Location" section on page 309, the trusted site should follow sub-flows 1 and 2 separately to generate two separate programming files. The programming file from sub-flow 1 will be used at the trusted site to program the device(s) first. The programming file from sub-flow 2 will be sent off-site for production programming.

In Application 3, described in the "Application 3: Nontrusted Environment—Field Updates/Upgrades" section on page 310, typically only sub-flow 2 will be used, because only updates to the design content portion are needed and no security settings need to be changed.

In the event that update of the security settings is necessary, see the "Reprogramming Devices" section on page 321 for details. For more information on programming low power flash devices, refer to the "In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" section on page 327.


User
Designer Software
Programming Software
1
Program Security Settings
User Assigns Desired Security Settings To FPGA/FlashROM/FB/All: – AES Key and FlashLock Pass Key – FlashLock Pass Key Only
Software Generates Programming File with Desired Security Settings: – Encrypted with AES and Protected with FlashLock Pass Key – Protected with FlashLock Pass Key Only
Device Previously Programmed?
No
Software Programs Selected Security Settings into Device
Yes
Software Performs Comparison of FlashLock Pass Key between Programming File and Device
Does FlashLock Pass Key Match?
Yes
No
Returns Error
2
Program Design Contents
Programming Previously Secured Device(s)?
No
Software Generates Programming File with Desired Design Contents (FPGA Array, FlashROM, FB, or All)
Design Content Programmed into Device
Yes
AES Key Used Previously?
No
User Must Reassign Exact FlashLock Pass Key Previously Programmed into the Device
Software Generates Programming File with FlashLock Pass Key and Design Contents
Software Performs Comparison of FlashLock Pass Key between Programming File and Device
Does FlashLock Pass Key Match?
Yes
No
Returns Error
Yes
User Must Reassign Exact AES Key Previously Programmed into the Device
Software Generates Programming File with Encrypted Design Contents
Encrypted Design Content Passes through MAC for Authentication
Correct?
No
Yes
Design Content Decrypted and Programmed into Device

Note: *If programming the Security Header only, just perform sub-flow 1.*
If programming design content only, just perform sub-flow 2.

Figure 12-9 **• Security Programming Flows**

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	✓	✓	✓