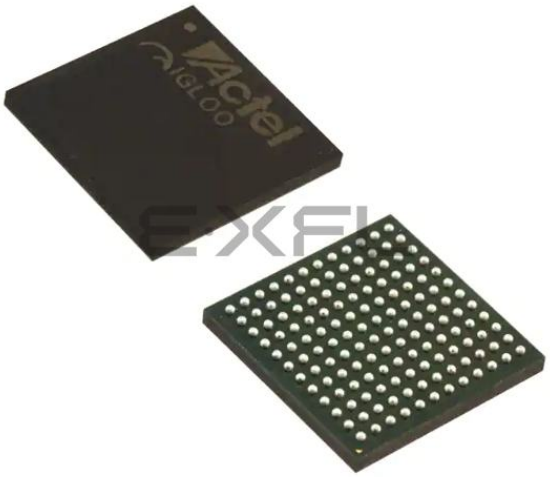


Welcome to E-XFL.COM

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	97
Number of Gates	600000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	144-LBGA
Supplier Device Package	144-FPBGA (13x13)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3p600l-1fgg144

Flash*Freeze Mode

IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 FPGAs offer an ultra-low static power mode to reduce power consumption while preserving the state of the registers, SRAM contents, and I/O states (IGLOO nano and IGLOO PLUS only) without switching off any power supplies, inputs, or input clocks.

Flash*Freeze technology enables the user to switch to Flash*Freeze mode within 1 µs, thus simplifying low power design implementation. The Flash*Freeze (FF) pin (active Low) is a dedicated pin used to enter or exit Flash*Freeze mode directly; or the pin can be routed internally to the FPGA core and state management IP to allow the user's application to decide if and when it is safe to transition to this mode. If the FF pin is not used, it can be used as a regular I/O.

The FF pin has a built-in glitch filter and optional Schmitt trigger (not available for all devices) to prevent entering or exiting Flash*Freeze mode accidentally.

There are two ways to use Flash*Freeze mode. In Flash*Freeze type 1, entering and exiting the mode is exclusively controlled by the assertion and deassertion of the FF pin. This enables an external processor or human interface device to directly control Flash*Freeze mode; however, valid data must be preserved using standard procedures (refer to the "Flash*Freeze Mode Device Behavior" section on page 30). In Flash*Freeze mode type 2, entering and exiting the mode is controlled by both the FF pin AND user-defined logic. Flash*Freeze management IP may be used in type 2 mode for clock and data management while entering and exiting Flash*Freeze mode.

Flash*Freeze Type 1: Control by Dedicated Flash*Freeze Pin

Flash*Freeze type 1 is intended for systems where either the device will be reset upon exiting Flash*Freeze mode, or data and clock are managed externally. The device enters Flash*Freeze mode 1 µs after the dedicated FF pin is asserted (active Low), and returns to normal operation when the FF pin is deasserted (High) (Figure 2-1 on page 25). In this mode, FF pin assertion or deassertion is the only condition that determines entering or exiting Flash*Freeze mode.

In Libero® System-on-Chip (SoC) software v8.2 and before, this mode is implemented by enabling Flash*Freeze mode (default setting) in the Compile options of the Microsemi Designer software. To simplify usage of Flash*Freeze mode, beginning with Libero software v8.3, an INBUF_FF I/O macro was introduced. An INBUF_FF I/O buffer must be used to identify the Flash*Freeze input. Microsemi recommends switching to the new implementation.

In Libero software v8.3 and later, the user must manually instantiate the INBUF_FF macro in the top level of the design to implement Flash*Freeze Type 1, as shown in Figure 2-1 on page 25.

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.

3 – Global Resources in Low Power Flash Devices

Introduction

IGLOO, Fusion, and ProASIC3 FPGA devices offer a powerful, low-delay VersaNet global network scheme and have extensive support for multiple clock domains. In addition to the Clock Conditioning Circuits (CCCs) and phase-locked loops (PLLs), there is a comprehensive global clock distribution network called a VersaNet global network. Each logical element (VersaTile) input and output port has access to these global networks. The VersaNet global networks can be used to distribute low-skew clock signals or high-fanout nets. In addition, these highly segmented VersaNet global networks contain spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside their region. This allows users the flexibility to create low-skew local clock networks using spines. This document describes VersaNet global networks and discusses how to assign signals to these global networks and spines in a design flow. Details concerning low power flash device PLLs are described in the "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 77. This chapter describes the low power flash devices' global architecture and uses of these global networks in designs.

Global Architecture

Low power flash devices offer powerful and flexible control of circuit timing through the use of global circuitry. Each chip has up to six CCCs, some with PLLs.

- In IGLOOe, ProASIC3EL, and ProASIC3E devices, all CCCs have PLLs—hence, 6 PLLs per device (except the PQ208 package, which has only 2 PLLs).
- In IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3, and ProASIC3L devices, the west CCC contains a PLL core (except in 10 k through 30 k devices).
- 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.

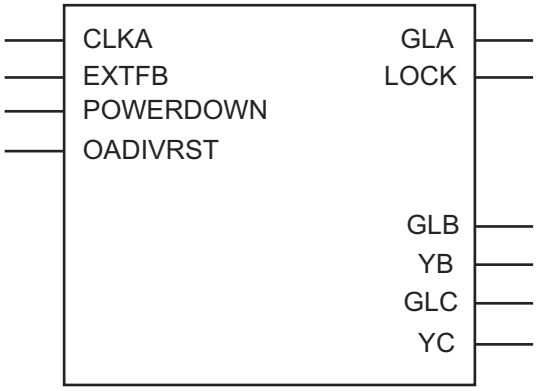
Refer to Table 4-6 on page 100 for details. Each PLL includes delay lines, a phase shifter (0°, 90°, 180°, 270°), and clock multipliers/dividers. Each CCC has all the circuitry needed for the selection and interconnection of inputs to the VersaNet global network. The east and west CCCs each have access to three chip global lines on each side of the chip (six chip global lines total). The CCCs at the four corners each have access to three quadrant global lines in each quadrant of the chip (except in 10 k through 30 k gate devices).

The nano 10 k, 15 k, and 20 k devices support four VersaNet global resources, and 30 k devices support six global resources. The 10 k through 30 k devices have simplified CCCs called CCC-GLs.

The flexible use of the VersaNet global network allows the designer to address several design requirements. User applications that are clock-resource-intensive can easily route external or gated internal clocks using VersaNet global routing networks. Designers can also drastically reduce delay penalties and minimize resource usage by mapping critical, high-fanout nets to the VersaNet global network.

Note: Microsemi recommends that you choose the appropriate global pin and use the appropriate global resource so you can realize these benefits.

The following sections give an overview of the VersaNet global network, the structure of the global network, access point for the global networks, and the clock aggregation feature that enables a design to have very low clock skew using spines.



Note: OAVDIVRST exists only in the Fusion PLL.

***Figure 3-15* • PLLs in Low Power Flash Devices**

You can use the syn_global_buffers attribute in Synplify to specify a maximum number of global macros to be inserted in the netlist. This can also be used to restrict the number of global buffers inserted. In the Synplicity 8.1 version or newer, a new attribute, syn_global_minfanout, has been added for low power flash devices. This enables you to promote only the high-fanout signal to global. However, be aware that you can only have six signals assigned to chip global networks, and the rest of the global signals should be assigned to quadrant global networks. So, if the netlist has 18 global macros, the remaining 12 global macros should have fanout that allows the instances driven by these globals to be placed inside a quadrant.

Global Promotion and Demotion Using PDC

The HDL source file or schematic is the preferred place for defining which signals should be assigned to a clock network using clock macro instantiation. This method is preferred because it is guaranteed to be honored by the synthesis tools and Designer software and stop any replication on this net by the synthesis tool. Note that a signal with fanout may have logic replication if it is not promoted to global during synthesis. In that case, the user cannot promote that signal to global using PDC. See Synplicity Help for details on using this attribute. To help you with global management, Designer allows you to promote a signal to a global network or demote a global macro to a regular macro from the user netlist using the compile options and/or PDC commands.

The following are the PDC constraints you can use to promote a signal to a global network:

1. PDC syntax to promote a regular net to a chip global clock:

   ```
   assign_global_clock –net netname
   ```

   The following will happen during promotion of a regular signal to a global network:

   – If the net is external, the net will be driven by a CLKINT inserted automatically by Compile.
   – The I/O macro will not be changed to CLKBUF macros.
   – If the net is an internal net, the net will be driven by a CLKINT inserted automatically by Compile.

2. PDC syntax to promote a net to a quadrant clock:

   ```
   assign_local_clock –net netname –type quadrant UR|UL|LR|LL
   ```

   This follows the same rule as the chip global clock network.

The following PDC command demotes the clock nets to regular nets.

```
unassign_global_clock -net netname
```

Simple Design Example

Consider a design consisting of six building blocks (shift registers) and targeted for an A3PE600-PQ208 (Figure 3-16 on page 68). The example design consists of two PLLs (PLL1 has GLA only; PLL2 has both GLA and GLB), a global reset (ACLR), an enable (EN_ALL), and three external clock domains (QCLK1, QCLK2, and QCLK3) driving the different blocks of the design. Note that the PQ208 package only has two PLLs (which access the chip global network). Because of fanout, the global reset and enable signals need to be assigned to the chip global resources. There is only one free chip global for the remaining global (QCLK1, QCLK2, QCLK3). Place two of these signals on the quadrant global resource. The design example demonstrates manually assignment of QCLK1 and QCLK2 to the quadrant global using the PDC command.

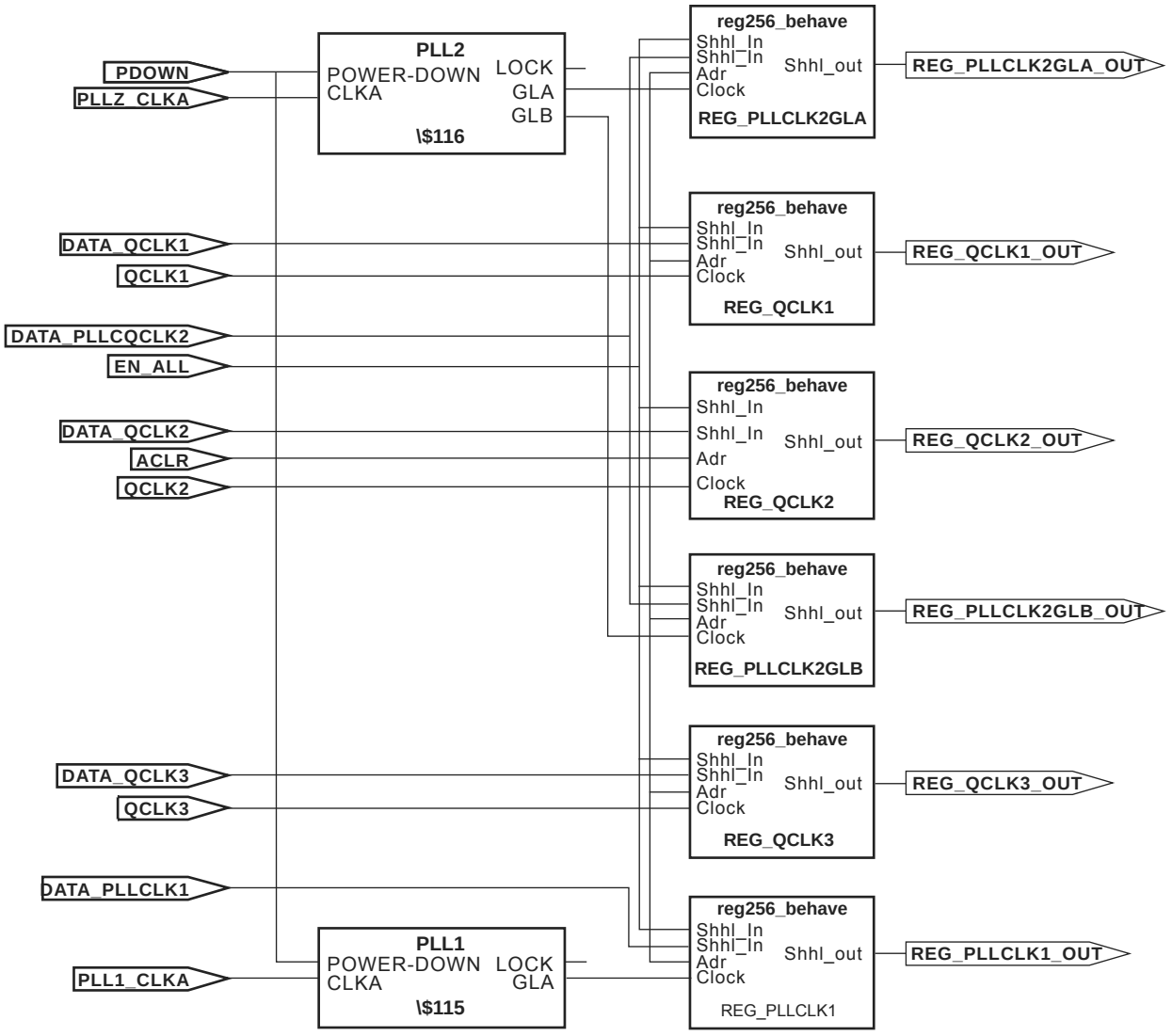


Figure 3-19 • **Block Diagram of the Global Management Example Design**

CCC Support in Microsemi's Flash Devices

The flash FPGAs listed in Table 4-1 support the CCC feature and the functions described in this document.

***Table 4-1* • 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 PLUS	IGLOO FPGAs with enhanced I/O capabilities
	IGLOO nano	The industry's lowest-power, smallest-size solution
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 4-1. 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 4-1. 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*.

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.

```
DLYGLC[4:0]     00000
DLYYB[4:0]      00000
DLYYC[4:0]      00000
VCOSEL[2:0]     100

Primary Clock Frequency 33.000
Primary Clock Phase Shift 0.000
Primary Clock Output Delay from CLKA 1.695

Secondary1 Clock Frequency 40.000
Secondary1 Clock Phase Shift 0.000
Secondary1 Clock Global Output Delay from CLKB 0.200

Secondary2 Clock Frequency 50.000
Secondary2 Clock Phase Shift 0.000
Secondary2 Clock Global Output Delay from CLKC 0.200

#####################################
# Dynamic Stream Data
#####################################
--------------------------------------
|NAME    |SDIN     |VALUE   |TYPE     |
--------------------------------------
|FINDIV  |[6:0]    |0000101 |EDIT     |
|FBDIV   |[13:7]   |0100000 |EDIT     |
|OADIV   |[18:14]  |00100   |EDIT     |
|OBDIV   |[23:19]  |00000   |EDIT     |
|OCDIV   |[28:24]  |00000   |EDIT     |
|OAMUX   |[31:29]  |100     |EDIT     |
|OBMUX   |[34:32]  |000     |EDIT     |
|OCMUX   |[37:35]  |000     |EDIT     |
|FBSEL   |[39:38]  |01      |EDIT     |
|FBDLY   |[44:40]  |00000   |EDIT     |
|XDLYSEL |[45]     |0       |EDIT     |
|DLYGLA  |[50:46]  |00000   |EDIT     |
|DLYGLB  |[55:51]  |00000   |EDIT     |
|DLYGLC  |[60:56]  |00000   |EDIT     |
|DLYYB   |[65:61]  |00000   |EDIT     |
|DLYYC   |[70:66]  |00000   |EDIT     |
|STATASEL|[71]     |X       |MASKED   |
|STATBSEL|[72]     |X       |MASKED   |
|STATCSEL|[73]     |X       |MASKED   |
|VCOSEL  |[76:74]  |100     |EDIT     |
|DYNASEL |[77]     |X       |MASKED   |
|DYNBSEL |[78]     |X       |MASKED   |
|DYNCSEL |[79]     |X       |MASKED   |
|RESETEN |[80]     |1       |READONLY |
```

Below is the resultant Verilog HDL description of a legal dynamic PLL core configuration generated by SmartGen:

```
module dyn_pll_macro(POWERDOWN, CLKA, LOCK, GLA, GLB, GLC, SDIN, SCLK, SSHIFT, SUPDATE,
  MODE, SDOUT, CLKB, CLKC);

input POWERDOWN, CLKA;
output  LOCK, GLA, GLB, GLC;
input  SDIN, SCLK, SSHIFT, SUPDATE, MODE;
output  SDOUT;
input  CLKB, CLKC;

  wire VCC, GND;

  VCC VCC_1_net(.Y(VCC));
  GND GND_1_net(.Y(GND));
```

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 143. 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 143 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 5-12 shows the programming file generator, which enables different STAPL file generation methods. When you select **Program FlashROM** and choose the UFC file, the FlashROM Settings window appears, as shown in Figure 5-13. In this window, you can select the FlashROM page you want to program and the data value for the configured regions. This enables you to use a different page for different programming files.

***Figure 5-12* • Programming File Generator**

***Figure 5-13* • Setting FlashROM during Programming File Generation**

The programming hardware and software can load the FlashROM with the appropriate STAPL file. Programming software handles the single STAPL file that contains multiple FlashROM contents for multiple devices, and programs the FlashROM in sequential order (e.g., for device serialization). This feature is supported in the programming software. After programming with the STAPL file, you can run DEVICE_INFO to check the FlashROM content.

without reprogramming the device. Dynamic flag settings are determined by register values and can be altered without reprogramming the device by reloading the register values either from the design or through the UJTAG interface described in the "Initializing the RAM/FIFO" section on page 164.

SmartGen can also configure the FIFO to continue counting after the FIFO is full. In this configuration, the FIFO write counter will wrap after the counter is full and continue to write data. With the FIFO configured to continue to read after the FIFO is empty, the read counter will also wrap and re-read data that was previously read. This mode can be used to continually read back repeating data patterns stored in the FIFO (Figure 6-15).

***Figure 6-15* • SmartGen FIFO Configuration Interface**

FIFOs configured using SmartGen can also make use of the port mapping feature to configure the names of the ports.

Limitations

Users should be aware of the following limitations when configuring SRAM blocks for low power flash devices:

- SmartGen does not track the target device in a family, so it cannot determine if a configured memory block will fit in the target device.
- Dual-port RAMs with different read and write aspect ratios are not supported.
- Cascaded memory blocks can only use a maximum of 64 blocks of RAM.
- The Full flag of the FIFO is sensitive to the maximum depth of the actual physical FIFO block, not the depth requested in the SmartGen interface.

Low Power Flash Device I/O Support

The low power flash FPGAs 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	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L

*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 7-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 7-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Example: 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 Ω, R_T = 70 Ω, given Z_O = 50 Ω (2") and Z_{stub} = 50 Ω (~1.5").

EQ 1

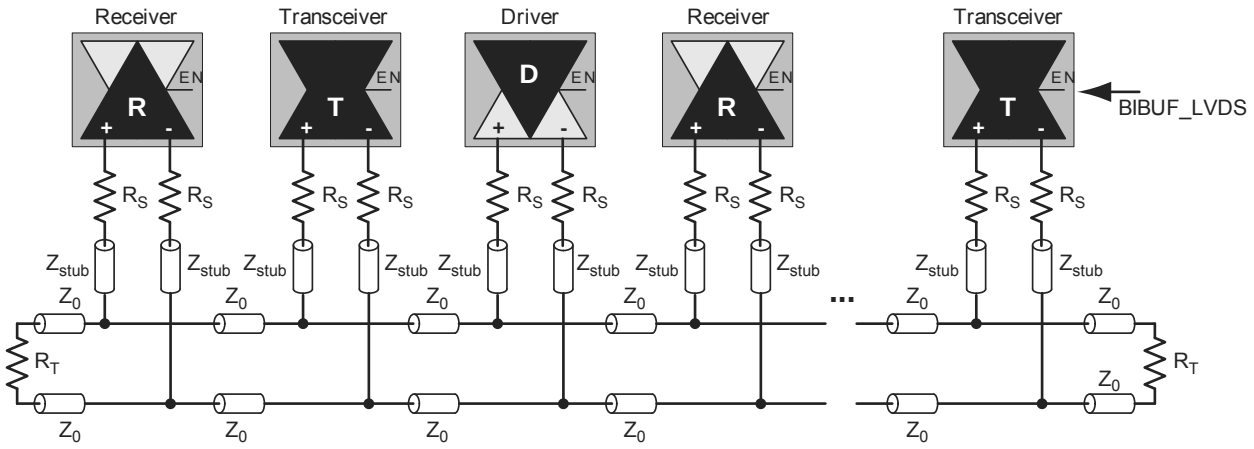


Figure 7-8 • **A B-LVDS/M-LVDS Multipoint Application Using LVDS I/O Buffers**

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

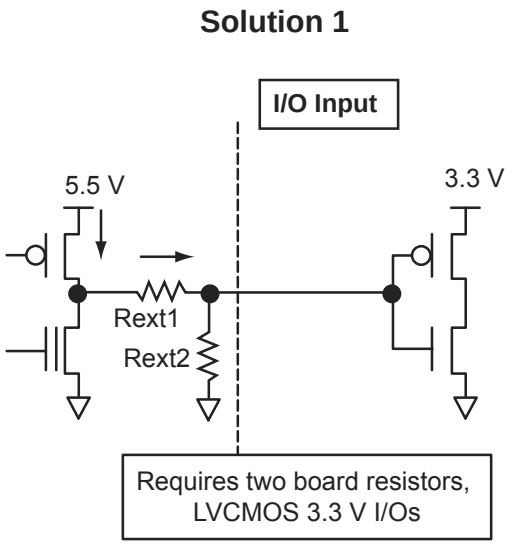


Figure 7-9 • **Solution 1**

Solution 2

The board-level design must ensure that the reflected waveform at the pad does not exceed the voltage overshoot/undershoot limits provided 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 external resistors and Zener, as shown in Figure 7-10. Relying on the diode clamping would create an excessive pad DC voltage of 3.3 V + 0.7 V = 4 V.

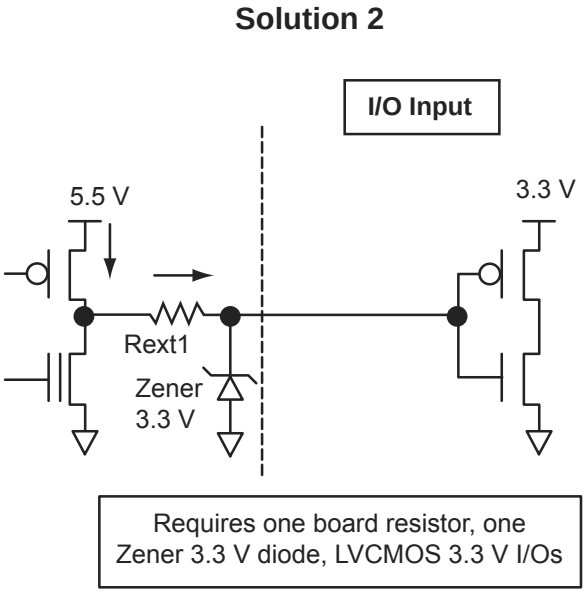


Figure 7-10 • **Solution 2**

9 – I/O Software Control in Low Power Flash Devices

Fusion, IGLOO, and ProASIC3 I/Os provide more design flexibility, allowing the user to control specific features by enabling certain I/O standards. Some features are selectable only for certain I/O standards, whereas others are available for all I/O standards. For example, slew control is not supported by differential I/O standards. Conversely, I/O register combining is supported by all I/O standards. For detailed information about which I/O standards and features are available on each device and each I/O type, refer to the I/O Structures section of the handbook for the device you are using.

Figure 9-1 shows the various points in the software design flow where a user can provide input or control of the I/O selection and parameters. A detailed description is provided throughout this document.

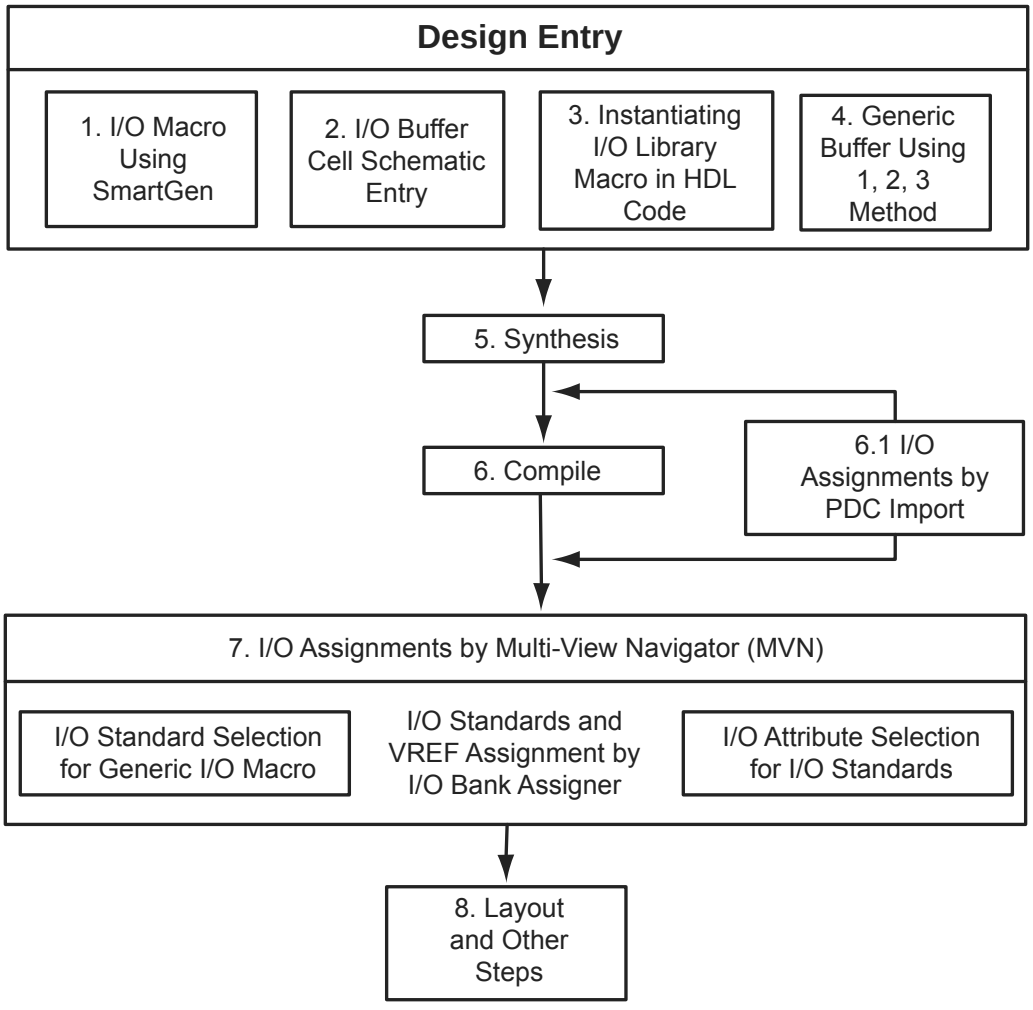


***Figure 9-1* • User I/O Assignment Flow Chart**

Flash FPGAs I/O Support

The flash FPGAs listed in Table 9-1 support I/Os and the functions described in this document.

***Table 9-1* • 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 9-1. 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 9-1. 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*.

Rules for the DDR I/O Function

- The fanout between an I/O pin (D or Y) and a DDR (DDR_REG or DDR_OUT) macro must be equal to one for the combining to happen on that pin.
- If a DDR_REG macro and a DDR_OUT macro are combined on the same bidirectional I/O, they must share the same clear signal.
- Registers will not be combined in an I/O in the presence of DDR combining on the same I/O.

Using the I/O Buffer Schematic Cell

Libero SoC software includes the ViewDraw schematic entry tool. Using ViewDraw, the user can insert any supported I/O buffer cell in the top-level schematic. Figure 9-5 shows a top-level schematic with different I/O buffer cells. When synthesized, the netlist will contain the same I/O macro.

***Figure 9-5* • I/O Buffer Schematic Cell Usage**

2. Choose the appropriate security level setting and enter a FlashLock Pass Key. The default is the **Medium** security level (Figure 12-12). Click **Next**.

   If you want to select different options for the FPGA and/or FlashROM, this can be set by clicking **Custom Level**. Refer to the "Advanced Options" section on page 322 for different custom security level options and descriptions of each.

***Figure 12-12* • Medium Security Level Selected for Low Power Flash Devices**

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

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

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

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

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

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

List of Changes

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

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.1 (October 2008)	The "Introduction" was revised to include information about the core supply voltage range of operation in V2 devices.	341
	IGLOO nano device support was added to Table 14-1 • Flash-Based FPGAs Supporting Voltage Switching Circuit.	342
	The "Circuit Description" section was updated to include IGLOO PLUS core operation from 1.2 V to 1.5 V in 50 mV increments.	343
v1.0 (August 2008)	The "Microsemi's Flash Families Support Voltage Switching Circuit" section was revised to include new families and make the information more concise.	342