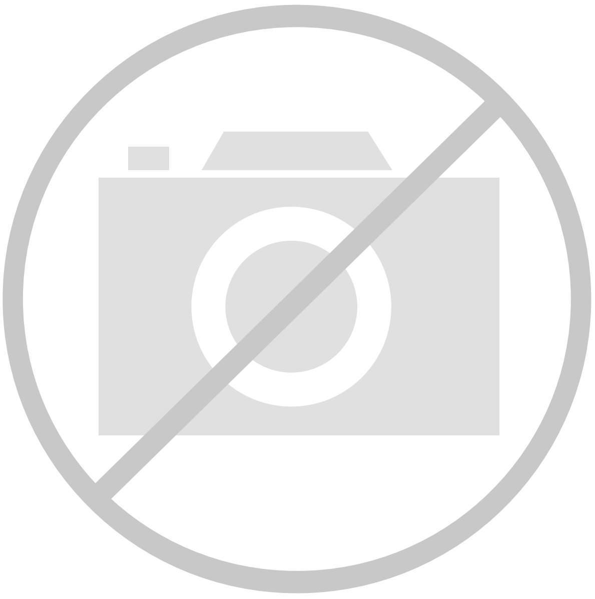
Microchip Technology - M1A3P600L-FGG484 Datasheet



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/m1a3p600l-fgg484

Set/Reset

Since all I/Os and globals are tied High in Flash*Freeze mode (unless hold state is used on IGLOO nano or IGLOO PLUS), Microsemi recommends using active low set/reset at the top-level port. If needed, the signal can be inverted internally.

- If the intention is to always set/reset in Flash*Freeze mode, a self set/reset circuit may be implemented to accomplish this, as shown in Figure 2-9. Configure an active High set/reset input pin so it uses the internal pull-up during Flash*Freeze mode, and drives Low during active mode. When the device exits Flash*Freeze mode, the input will transition from High to Low, releasing the set/reset. Note that this circuit may release set/reset before all outputs become active, since outputs are enabled up to 200 ns after inputs when exiting Flash*Freeze mode.

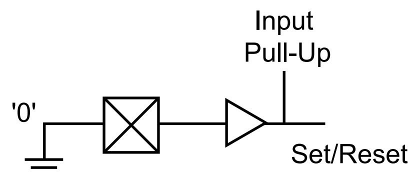


Figure 2-9 • **Flash*Freeze Self-Reset Circuit**

I/Os

- Floating inputs can cause totem pole currents on the input I/O circuitry when the device is in active mode. If inputs will be released (undriven) during Flash*Freeze mode, Microsemi recommends that they are only released after the device enters Flash*Freeze mode.
- As mentioned earlier, asynchronous input to output paths are subject to possible glitching when entering Flash*Freeze mode. For example, on a direct in-to-out path, if the current state is '0' and the input bank deactivates 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 along with Flash*Freeze management IP to gate asynchronous in-to-out paths prior to entering Flash*Freeze mode.

JTAG

- The JTAG state machine is powered but not active during Flash*Freeze mode.
- TCK should be held in a static state to prevent dynamic power consumption of the JTAG circuit during Flash*Freeze.
- Specific JTAG pin tie-off recommendations suitable for Flash*Freeze mode can be found in the "Pin Descriptions and Packaging" chapter of the device datasheet.

ULSICC

- The User Low Static ICC (ULSICC) macro acts as an access point to the hard Flash*Freeze technology block in the device. The ULSICC macro represents a hard, fixed location block in the device. When the LSICC input of the ULSICC macro is driven Low, the Flash*Freeze pin is blocked, and when LSICC is driven High, the Flash*Freeze pin is enabled.
- If the user decides to build his/her own Flash*Freeze type 2 clock and data management logic, note that the LSICC signal on the ULSICC macro is ANDed internally with the Flash*Freeze signal. In order to reliably enter Flash*Freeze, the LSICC signal must remain asserted High while entering and during Flash*Freeze mode.

Flash*Freeze Management IP

One of the key benefits of Microsemi's Flash*Freeze mode is the ability to preserve the state of all internal registers, SRAM content, and I/Os (IGLOO nano and IGLOO PLUS only). This feature enables seamless continuation of data processing before and after Flash*Freeze, without the need to reload or reinitialize the FPGA system. Microsemi's Flash*Freeze management IP, available for type 2 implementation, offers a robust RTL block that ensures clean clock gating of all system clocks before entering and upon exiting Flash*Freeze mode. This IP also gives users the option to perform housekeeping prior to entering Flash*Freeze mode. This section will provide an overview of the

Flash*Freeze management IP. Additional information on this IP core can be found in the Libero online help.

The Flash*Freeze management IP is comprised of three blocks: the Flash*Freeze finite state machine (FSM), the clock gating (filter) block, and the ULSICC macro, as shown in Figure 2-10.

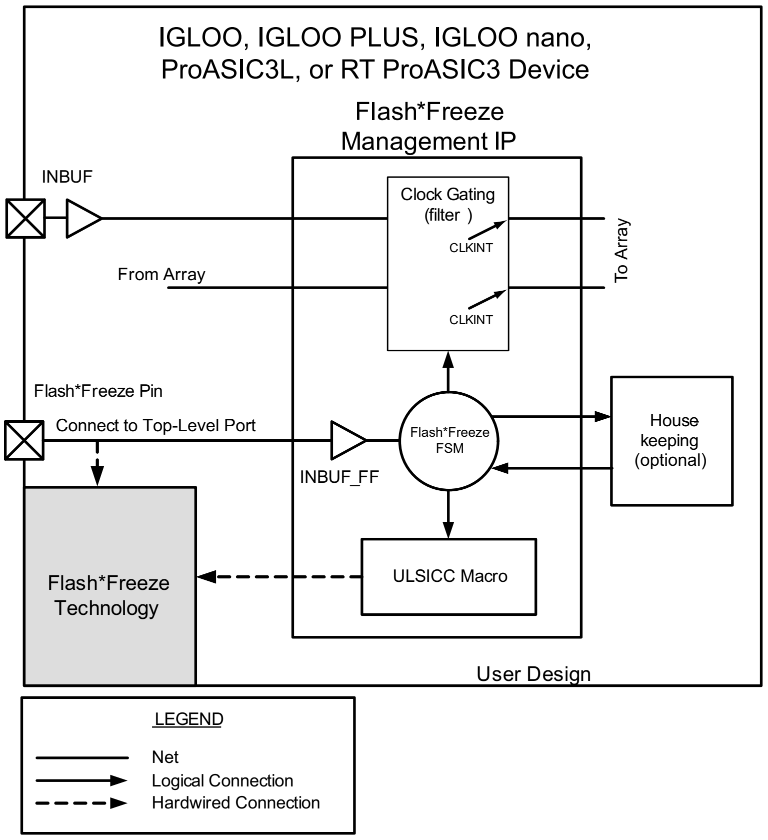


Figure 2-10 **• Flash*Freeze Management IP Block Diagram**

*Flash*Freeze Management FSM*

The Flash*Freeze FSM block is a simple, robust, fully encoded 3-bit state machine that ensures clean entrance to and exit from Flash*Freeze mode by controlling activities of the clock gating, ULSICC, and optional housekeeping blocks. The state diagram for the FSM is shown in Figure 2-11 on page 38. In normal operation, the state machine waits for Flash*Freeze pin assertion, and upon detection of a request, it waits for a short period of time to ensure the assertion persists; then it asserts WAIT_HOUSEKEEPING (active High) synchronous to the user's designated system clock. This flag can be used by user logic to perform any needed shutdown processes prior to entering Flash*Freeze mode, such as storing data into SRAM, notifying other system components of the request, or timing/validating the Flash*Freeze request. The FSM also asserts Flash_Freeze_Enabled whenever the device enters Flash*Freeze mode. This occurs after all housekeeping and clock gating functions have completed. The Flash_Freeze_Enabled signal remains asserted, even during Flash*Freeze mode, until the Flash*Freeze pin is deasserted. Use the Flash_Freeze_Enabled signal to drive any logic in the design that needs to be in a particular state during Flash*Freeze mode. The DONE_HOUSEKEEPING (active High) signal should be asserted to notify the FSM when all the housekeeping tasks are completed. If the user chooses not to use housekeeping, the Flash*Freeze management IP core generator in Libero SoC will connect WAIT_HOUSEKEEPING to DONE_HOUSEKEEPING.

Clock Gating Block

Once DONE_HOUSEKEEPING is detected, the FSM will initiate the clock gating circuit by asserting ASSERT_GATE (active Low). ASSERT_GATE is named control_user_clock_net in the IP block. Upon assertion of the ASSERT_GATE signal, the clock will be gated in less than two cycles. The clock gating circuit is comprised of a flip-flop, latch, AND gate, and CLKINT, as shown in Figure 2-12. The clock gating block can support gating of up to 17 clocks.

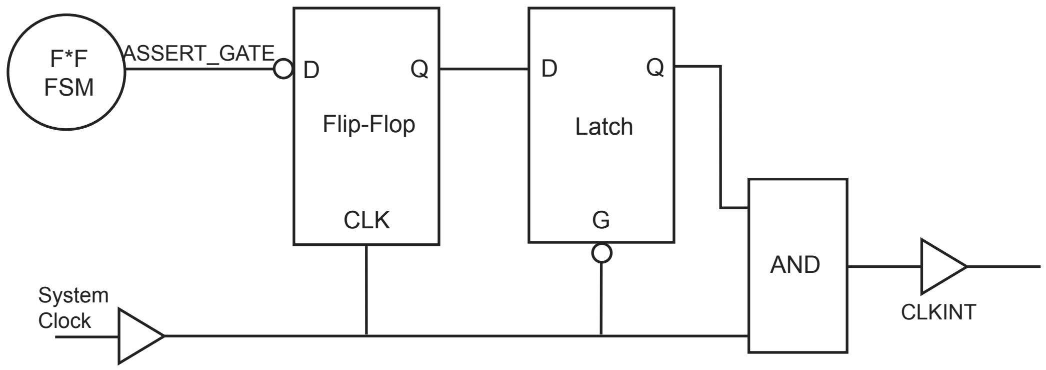


Figure 2-12 **• Clock Gating Circuit**

After initiating the clock gating circuit, the FSM will assert and hold the LSICC signal (active High), feeding the ULSICC macro. This will initiate the 1 µs entrance into Flash*Freeze mode.

Upon deassertion of the Flash*Freeze pin, the FSM will set ASSERT_GATE High. Once the I/O banks become active, the clock will enter the device and register the ASSERT_GATE signal, cleanly releasing the clock gate.

Design Flow[1]

Microsemi has developed a convenient and intuitive design flow for configuring and integrating Flash*Freeze technology into an FPGA design. Flash*Freeze type 1 is implemented by instantiating the INBUF_FF macro in the top level of a design. Flash*Freeze type 2 with management IP can be generated by the Libero core generator or SmartGen and instantiated as a single block in the user's design. This single block will include an INBUF_FF macro and the optional Flash*Freeze management IP, which includes the ULSICC macro. If designers do not wish to use this core generator, the INBUF_FF macro and the optional ULSICC macro may be instantiated in the design, and custom Flash*Freeze management IP can be developed by the user. The remainder of this section will cover configuration details of the INBUF_FF macro, the ULSICC macro, and the Flash*Freeze management IP.

Additional information on the tools discussed within this section may be found in the Libero online help.

INBUF_FF

The INBUF_FF macro is a special-purpose input buffer macro that is interpreted downstream in the design flow by Microsemi's Designer software. When this macro is used, the top-level port will be forced to the dedicated FF pin in the FPGA, and Flash*Freeze mode will be available for use in the device. The following are the design rules for INBUF_FF:

- If INBUF_FF is not used in the design, the device will not be configured to support Flash*Freeze mode.
- When the INBUF_FF macro is used, the FF pin will establish a hardwired connection to the Flash*Freeze technology circuit in the device, as shown in Figure 2-1 on page 25, Figure 2-3 on page 27, and Figure 2-10 on page 37, and described in the "Flash*Freeze Type 1: Control by Dedicated Flash*Freeze Pin" section on page 24.

1. *This section applies to Libero / Designer software v8.3 and later. Microsemi recommends that designs created in earlier versions of the software be modified to accommodate this flow by instantiating the INBUF_FF macro or the Flash*Freeze management IP. Refer to the Libero / Designer software v8.3 release notes and the Libero online help for more information on migrating designs from older software versions.*

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

External Feedback Configuration

For certain applications, such as those requiring generation of PCB clocks that must be matched with existing board delays, it is useful to implement an external feedback, EXTFB. The Phase Detector of the PLL core will receive CLKA and EXTFB as inputs. EXTFB may be processed by the fixed System Delay element as well as the *M* divider element. The EXTFB option is currently not supported.

After setting all the required parameters, users can generate one or more PLL configurations with HDL or EDIF descriptions by clicking the **Generate** button. SmartGen gives the option of saving session results and messages in a log file:

```
****************
Macro Parameters
****************

Name                            : test_pll
Family                          : ProASIC3E
Output Format                   : VHDL
Type                            : Static PLL
Input Freq(MHz)                 : 10.000
CLKA Source                     : Hardwired I/O
Feedback Delay Value Index      : 1
Feedback Mux Select             : 2
XDLY Mux Select                 : No
Primary Freq(MHz)               : 33.000
Primary PhaseShift              : 0
Primary Delay Value Index       : 1
Primary Mux Select              : 4
Secondary1 Freq(MHz)            : 66.000
Use GLB                         : YES
Use YB                          : YES
GLB Delay Value Index           : 1
YB Delay Value Index            : 1
Secondary1 PhaseShift           : 0
Secondary1 Mux Select           : 4
Secondary2 Freq(MHz)            : 101.000
Use GLC                         : YES
Use YC                          : NO
GLC Delay Value Index           : 1
YC Delay Value Index            : 1
Secondary2 PhaseShift           : 0
Secondary2 Mux Select           : 4

…
…
…

Primary Clock frequency 33.333
Primary Clock Phase Shift 0.000
Primary Clock Output Delay from CLKA 0.180

Secondary1 Clock frequency 66.667
Secondary1 Clock Phase Shift 0.000
Secondary1 Clock Global Output Delay from CLKA 0.180
Secondary1 Clock Core Output Delay from CLKA 0.625

Secondary2 Clock frequency 100.000
Secondary2 Clock Phase Shift 0.000
Secondary2 Clock Global Output Delay from CLKA 0.180
```

Below is an example Verilog HDL description of a legal PLL core configuration generated by SmartGen:

```
module test_pll(POWERDOWN,CLKA,LOCK,GLA);
input POWERDOWN, CLKA;
output  LOCK, GLA;
```

Conclusion

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

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

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

Related Documents

User's Guides

FlashPro User's Guide

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

List of Changes

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

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 5-1 • Flash-Based FPGAs.	134
v1.3 (October 2008)	The "FlashROM Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	134
	Figure 5-2 • Fusion Device Architecture Overview (AFS600) was replaced. Figure 5-5 • Programming FlashROM Using AES was revised to change "Fusion" to "Flash Device."	135, 137
	The *FlashPoint User's Guide* was removed from the "User's Guides" section, as its content is now part of the *FlashPro User's Guide*.	146
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 5-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	134
v1.1 (March 2008)	The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	N/A

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**

```
//
addr_counter counter_1 (.Clock(data_update), .Q(wr_addr), .Aset(rst_n),
  .Enable(enable));
addr_counter counter_2 (.Clock(test_clk), .Q(rd_addr), .Aset(rst_n),
  .Enable( test_active));

endmodule
```

Interface Block / UJTAG Wrapper

This example is a sample wrapper, which connects the interface block to the UJTAG and the memory blocks.

```
// WRAPPER
module top_init (TDI, TRSTB, TMS, TCK, TDO, test, test_clk, test_ out);

input TDI, TRSTB, TMS, TCK;
output TDO;
input test, test_clk;
output [3:0] test_out;

wire [7:0] IR;
wire reset, DR_shift, DR_cap, init_clk, DR_update, data_in, data_out;
wire clk_out, wen, ren;
wire [3:0] word_in, word_out;
wire [1:0] write_addr, read_addr;

UJTAG UJTAG_U1 (.UIREG0(IR[0]), .UIREG1(IR[1]), .UIREG2(IR[2]), .UIREG3(IR[3]),
  .UIREG4(IR[4]), .UIREG5(IR[5]), .UIREG6(IR[6]), .UIREG7(IR[7]), .URSTB(reset),
  .UDRSH(DR_shift), .UDRCAP(DR_cap), .UDRCK(init_clk), .UDRUPD(DR_update),
  .UT-DI(data_in), .TDI(TDI), .TMS(TMS), .TCK(TCK), .TRSTB(TRSTB), .TDO(TDO),
  .UT-DO(data_out));
mem_block RAM_block (.DO(word_out), .RCLOCK(clk_out), .WCLOCK(clk_out), .DI(word_in),
  .WRB(wen), .RDB(ren), .WAD-DR(write_addr), .RADDR(read_addr));
interface init_block (.IR(IR), .rst_n(reset), .data_shift(DR_shift), .clk_in(init_clk),
  .data_update(DR_update), .din_ser(data_in), .dout_ser(data_out), .test(test),
  .test_out(test_out), .test_clk(test_clk), .clk_out(clk_out), .wr_en(wen),
  .rd_en(ren), .write_word(word_in), .read_word(word_out), .rd_addr(read_addr),
  .wr_addr(write_addr));

endmodule
```

Address Counter

```
module addr_counter (Clock, Q, Aset, Enable);

input Clock;
output [1:0] Q;
input Aset;
input Enable;

reg [1:0] Qaux;

always @(posedge Clock or negedge Aset)
begin
  if (!Aset) Qaux <= 2'b11;
  else if (Enable) Qaux <= Qaux + 1;
end

assign Q = Qaux;

endmodule
```

***Table 7-12* • I/O Hot-Swap and 5 V Input Tolerance Capabilities in IGLOO and ProASIC3 Devices**

	Clamp Diode [1]		Hot Insertion		5 V Input Tolerance [2]		
I/O Assignment	**AGL030 and A3P030**	**Other IGLOO and ProASIC3 Devices**	**AGL015 and AGL030**	**Other IGLOO Devices and All ProASIC3**	**AGL030 and A3P030**	**Other IGLOO and ProASIC3 Devices**	**Input and Output Buffer**
3.3 V LVTTL/LVCMOS	No	Yes	Yes	No	Yes [2]	Yes [2]	Enabled/Disabled
3.3 V PCI, 3.3 V PCI-X	N/A	Yes	N/A	No	N/A	Yes [2]	Enabled/Disabled
LVCMOS 2.5 V [5]	No	Yes	Yes	No	Yes [2]	Yes [4]	Enabled/Disabled
LVCMOS 2.5 V/5.0 V [6]	N/A	Yes	N/A	No	N/A	Yes [4]	Enabled/Disabled
LVCMOS 1.8 V	No	Yes	Yes	No	No	No	Enabled/Disabled
LVCMOS 1.5 V	No	Yes	Yes	No	No	No	Enabled/Disabled
Differential, LVDS/ B-LVDS/M-LVDS/LVPECL	N/A	Yes	N/A	No	N/A	No	Enabled/Disabled

Notes:

1. *The clamp diode is always off for the AGL030 and A3P030 device and always active for other IGLOO and ProASIC3 devices.*
2. *Can be implemented with an external IDT bus switch, resistor divider, or Zener with resistor.*
3. *Refer to Table 7-8 on page 189 to Table 7-11 on page 190 for device-compliant information.*
4. *Can be implemented with an external resistor and an internal clamp diode.*
5. *The LVCMOS 2.5 V I/O standard is supported by the 30 k gate devices only; select the LVCMOS25 macro.*
6. *The LVCMOS 2.5 V / 5.0 V I/O standard is supported by all IGLOO and ProASIC3 devices except 30K gate devices; select the LVCMOS5 macro.*

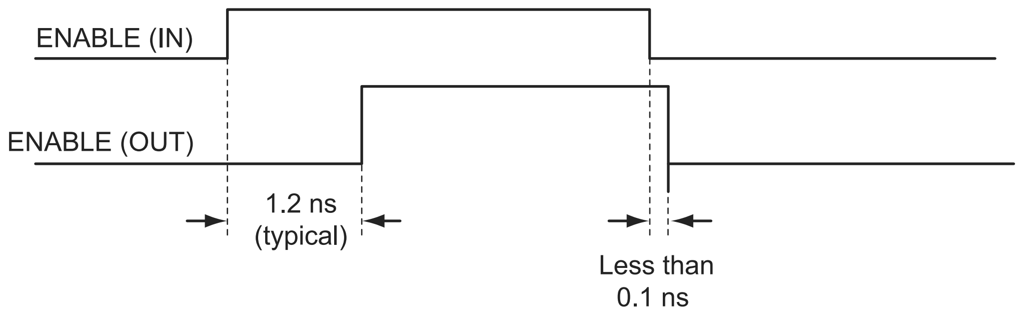


Figure 7-15 • **Timing Diagram (option 2: enables skew circuit)**

At the system level, the skew circuit can be used in applications where transmission activities on bidirectional data lines need to be coordinated. This circuit, when selected, provides a timing margin that can prevent bus contention and subsequent data loss and/or transmitter over-stress due to transmitter-to-transmitter current shorts. Figure 7-16 presents an example of the skew circuit implementation in a bidirectional communication system. Figure 7-17 on page 201 shows how bus contention is created, and Figure 7-18 on page 201 shows how it can be avoided with the skew circuit.

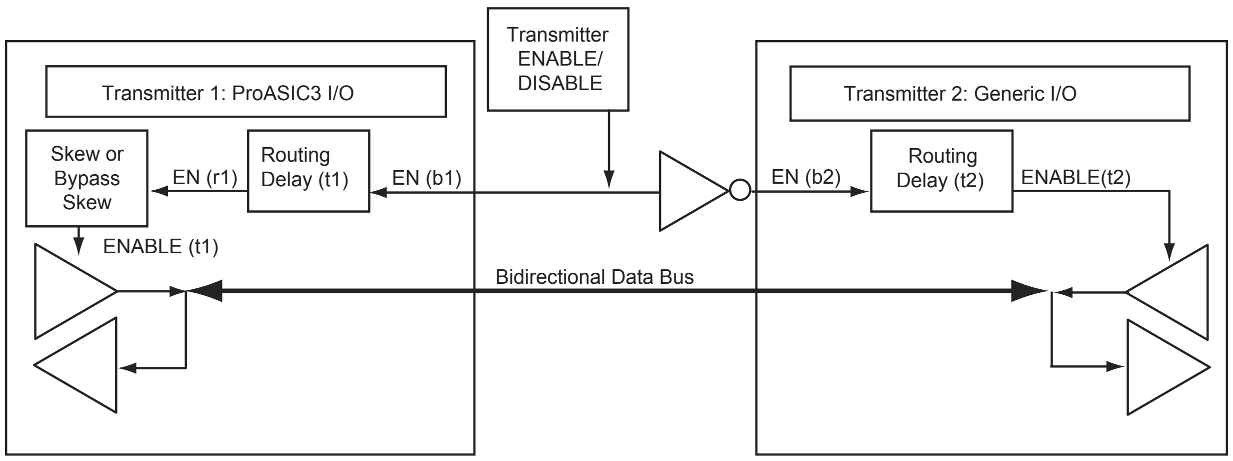


Figure 7-16 • **Example of Implementation of Skew Circuits in Bidirectional Transmission Systems Using IGLOO or ProASIC3 Devices**

Table 8-11 • **Hot-Swap Level 3**

Description	Hot-swap while bus idle
Power Applied to Device	Yes
Bus State	Held idle (no ongoing I/O processes during insertion/removal)
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	Must remain glitch-free during power-up or power-down
Example Application	Board bus shared with card bus is "frozen," and there is no toggling activity on the bus. It is critical that the logic states set on the bus signal not be disturbed during card insertion/removal.
Compliance of IGLOO and ProASIC3 Devices	30 k gate devices, all IGLOOe/ProASIC3E devices: Compliant with two levels of staging (first: GND; second: all other pins) Other IGLOO/ProASIC3 devices: Compliant: Option A – Two levels of staging (first: GND; second: all other pins) together with bus switch on the I/Os Option B – Three levels of staging (first: GND; second: supplies; third: all other pins)

Table 8-12 • **Hot-Swap Level 4**

Description	Hot-swap on an active bus
Power Applied to Device	Yes
Bus State	Bus may have active I/O processes ongoing, but device being inserted or removed must be idle.
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	Must remain glitch-free during power-up or power-down
Example Application	There is activity on the system bus, and it is critical that the logic states set on the bus signal not be disturbed during card insertion/removal.
Compliance of IGLOO and ProASIC3 Devices	30 k gate devices, all IGLOOe/ProASIC3E devices: Compliant with two levels of staging (first: GND; second: all other pins) Other IGLOO/ProASIC3 devices: Compliant: Option A – Two levels of staging (first: GND; second: all other pins) together with bus switch on the I/Os Option B – Three levels of staging (first: GND; second: supplies; third: all other pins)

Solution 4

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.

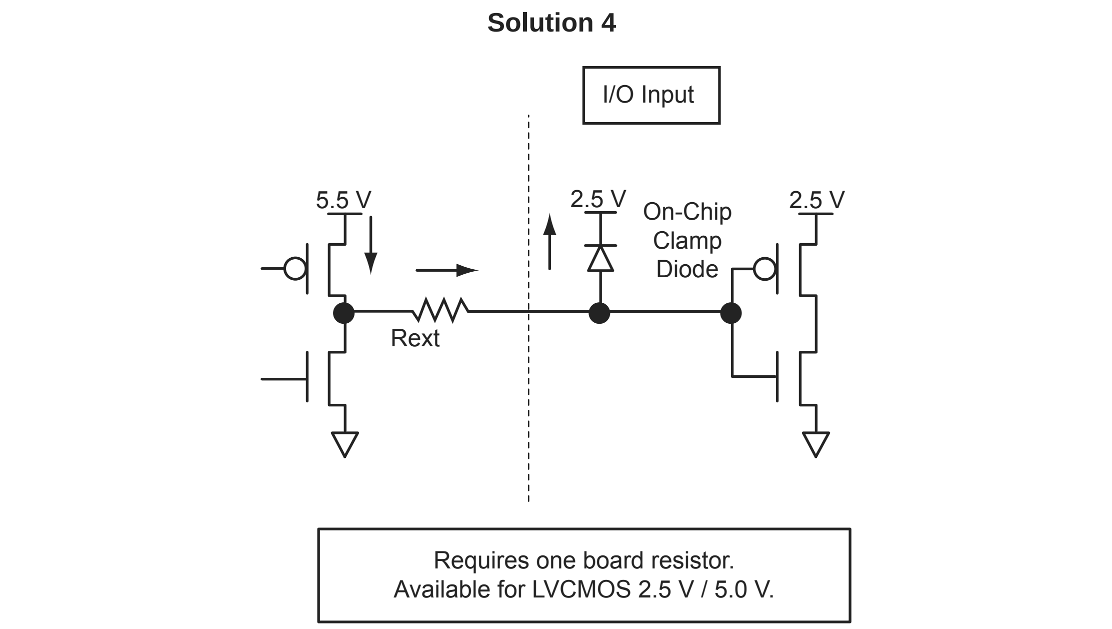

Solution 4
I/O Input
5.5 V
2.5 V
On-Chip Clamp Diode
2.5 V
Rext
Requires one board resistor.
Available for LVCMOS 2.5 V / 5.0 V.

Figure 8-13 • **Solution 4**

Table 8-14 • **Comparison Table for 5 V–Compliant Receiver Solutions**

Solution	Board Components	Speed	Current Limitations
1	Two resistors	Low to High[1]	Limited by transmitter's drive strength
2	Resistor and Zener 3.3 V	Medium	Limited by transmitter's drive strength
3	Bus switch	High	N/A

4	Minimum resistor value[2,3,4,5]
	
	R = 47 Ω at T_J = 70°C
	R = 150 Ω at T_J = 85°C
	R = 420 Ω at T_J = 100°C
		Medium	Maximum diode current at 100% duty cycle, signal constantly at 1
			52.7 mA at T_J = 70°C / 10-year lifetime
			16.5 mA at T_J = 85°C / 10-year lifetime
			5.9 mA at T_J = 100°C / 10-year lifetime
			For duty cycles other than 100%, the currents can be increased by a factor of 1 / (duty cycle).
			Example: 20% duty cycle at 70°C
			Maximum current = (1 / 0.2) × 52.7 mA = 5 × 52.7 mA = 263.5 mA

Notes:

1. *Speed and current consumption increase as the board resistance values decrease.*
2. *Resistor values ensure I/O diode long-term reliability.*
3. *At 70°C, customers could still use 420 Ω on every I/O.*
4. *At 85°C, a 5 V solution on every other I/O is permitted, since the resistance is lower (150 Ω) and the current is higher. Also, the designer can still use 420 Ω and use the solution on every I/O.*
5. *At 100°C, the 5 V solution on every I/O is permitted, since 420 Ω are used to limit the current to 5.9 mA.*

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.

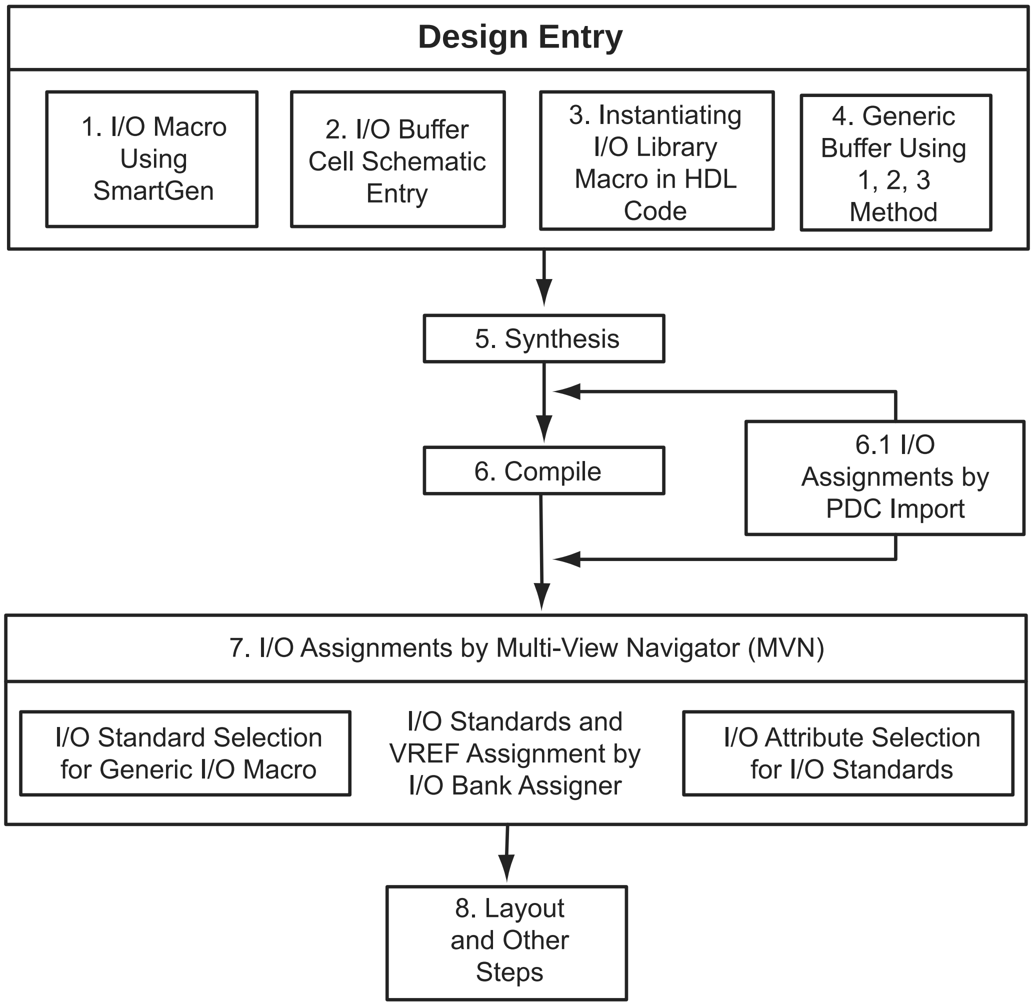

Design Entry
1. I/O Macro Using SmartGen
2. I/O Buffer Cell Schematic Entry
3. Instantiating I/O Library Macro in HDL Code
4. Generic Buffer Using 1, 2, 3 Method
5. Synthesis
6. Compile
6.1 I/O Assignments by PDC Import
7. I/O Assignments by Multi-View Navigator (MVN)
I/O Standard Selection for Generic I/O Macro
I/O Standards and VREF Assignment by I/O Bank Assigner
I/O Attribute Selection for I/O Standards
8. Layout and Other Steps

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

***Table 9-3* • PDC I/O Constraints (continued)**

Command	Action	Example	Comment
I/O Attribute Constraint			

set_io	Sets the attributes of an I/O	`set_io portname`
		`[-pinname value]`
		`[-fixed value]`
		`[-iostd value]`
		`[-out_drive value]`
		`[-slew value]`
		`[-res_pull value]`
		`[-schmitt_trigger value]`
		`[-in_delay value]`
		`[-skew value]`
		`[-out_load value]`
		`[-register value]`
		
		`set_io IN2 -pinname 28`
		`-fixed yes -iostd LVCMOS15`
		`-out_drive 12 -slew high`
		`-RES_PULL None`
		`-SCHMITT_TRIGGER Off`
		`-IN_DELAY Off –skew off`
		`-REGISTER No`
			If the I/O macro is generic (e.g., INBUF) or technology-specific (INBUF_LVCMOS25), then all I/O attributes can be assigned using this constraint.
			If the netlist has an I/O macro that specifies one of its attributes, that attribute cannot be changed using this constraint, though other attributes can be changed.
			Example: OUTBUF_S_24 (low slew, output drive 24 mA)
			Slew and output drive cannot be changed.

I/O Region Placement Constraints			

define_region	Defines either a rectangular region or a rectilinear region	`define_region`
		`-name [region_name]`
		`-type [region_type] x1 y1 x2 y2`
		
		`define_region -name test`
		`-type inclusive 0 15 2 29`
			If any number of I/Os must be assigned to a particular I/O region, such a region can be created with this constraint.

assign_region	Assigns a set of macros to a specified region	`assign_region [region name]`
		`[macro_name...]`
		
		`assign_region test U12`
			This constraint assigns I/O macros to the I/O regions. When assigning an I/O macro, PDC naming conventions must be followed if the macro name contains special characters; e.g., if the macro name is \\$1I19\\, the correct use of escape characters is \\\\$1I19\\\\.

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

4. Right-click and then choose **Highlight VREF range**. All the pins covered by that VREF pin will be highlighted (Figure 9-14).

Figure 9-14 **• VREF Range**

Using PinEditor or ChipPlanner, VREF pins can also be assigned (Figure 9-15).

Figure 9-15 **• Assigning VREF from PinEditor**

To unassign a VREF pin:

1. Select the pin to unassign.
2. Right-click and choose **Use Pin for VREF.** The check mark next to the command disappears. The VREF pin is now a regular pin.

Resetting the pin may result in unassigning I/O cores, even if they are locked. In this case, a warning message appears so you can cancel the operation.

After you assign the VREF pins, right-click a VREF pin and choose **Highlight VREF Range** to see how many I/Os are covered by that pin. To unhighlight the range, choose **Unhighlight All** from the **Edit** menu.

Generating Programming Files

Generation of the Programming File in a Trusted Environment—Application 1

As discussed in the "Application 1: Trusted Environment" section on page 309, in a trusted environment, the user can choose to program the device with plaintext bitstream content. It is possible to use plaintext for programming even when the FlashLock Pass Key option has been selected. In this application, it is not necessary to employ AES encryption protection. For AES encryption settings, refer to the next sections.

The generated programming file will include the security setting (if selected) and the plaintext programming file content for the FPGA array, FlashROM, and/or FBs. These options are indicated in Table 12-2 and Table 12-3.

***Table 12-2* • IGLOO and ProASIC3 Plaintext Security Options, No AES**

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

***Table 12-3* • Fusion Plaintext Security Options**

Security Protection	FlashROM Only	FPGA Core Only	FB Core Only	All
No AES / no FlashLock	✓	✓	✓	✓
FlashLock	✓	✓	✓	✓
AES and FlashLock	–	–	–	–

Note: For all instructions, the programming of Flash Blocks refers to Fusion only.

For this scenario, generate the programming file as follows:

1. Select the **Silicon features to be programmed** (Security Settings, FPGA Array, FlashROM, Flash Memory Blocks), as shown in Figure 12-10 on page 314 and Figure 12-11 on page 314. Click **Next**.

   If **Security Settings** is selected (i.e., the FlashLock security Pass Key feature), an additional dialog will be displayed to prompt you to select the security level setting. If no security setting is selected, you will be directed to Step 3.

Figure 12-10 • **All Silicon Features Selected for IGLOO and ProASIC3 Devices**

Figure 12-11 • **All Silicon Features Selected for Fusion**

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**.

Programming File Header Definition

In each STAPL programming file generated, there will be information about how the AES key and FlashLock Pass Key are configured. Table 12-8 shows the header definitions in STAPL programming files for different security levels.

***Table 12-8* • STAPL Programming File Header Definitions by Security Level**

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

Example File Headers

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

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

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

UJTAG Support in Flash-Based Devices

The flash-based FPGAs listed in Table 17-1 support the UJTAG feature and the functions described in this document.

***Table 17-1* • 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 17-1. 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 17-1. 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*.