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	147456
Number of I/O	300
Number of Gates	1000000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	484-BGA
Supplier Device Package	484-FPBGA (23x23)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/a3p1000l-1fgg484i

Figure 3-18 • **Globals Management GUI in Designer**

3. Occasionally, the synthesis tool assigns a global macro to clock nets, even though the fanout is significantly less than other asynchronous signals. Select **Demote global nets whose fanout is less than** and enter a reasonable value for fanouts. This frees up some global networks from the signals that have very low fanouts. This can also be done using PDC.
4. Use a local clock network for the signals that do not need to go to the whole chip but should have low skew. This local clock network assignment can only be done using PDC.
5. Assign the I/O buffer using MVN if you have fixed I/O assignment. As shown in Figure 3-10 on page 61, there are three sets of global pins that have a hardwired connection to each global network. Do not try to put multiple CLKBUF macros in these three sets of global pins. For example, do not assign two CLKBUFs to GAA0x and GAA2x pins.
6. You must click **Commit** at the end of MVN assignment. This runs the pre-layout checker and checks the validity of global assignment.
7. Always run Compile with the **Keep existing physical constraints** option on. This uses the quadrant clock network assignment in the MVN assignment and checks if you have the desired signals on the global networks.
8. Run Layout and check the timing.

Feedback Configuration

The PLL provides both internal and external feedback delays. Depending on the configuration, various combinations of feedback delays can be achieved.

Internal Feedback Configuration

This configuration essentially sets the feedback multiplexer to route the VCO output of the PLL core as the input to the feedback of the PLL. The feedback signal can be processed with the fixed system and the adjustable feedback delay, as shown in Figure 4-24. The dividers are automatically configured by SmartGen based on the user input.

Indicated below is the System Delay pull-down menu. The System Delay can be bypassed by setting it to 0. When set, it adds a 2 ns delay to the feedback path (which results in delay advancement of the output clock by 2 ns).

***Figure 4-24* • Internal Feedback with Selectable System Delay**

Figure 4-25 shows the controllable Feedback Delay. If set properly in conjunction with the fixed System Delay, the total output delay can be advanced significantly.

***Figure 4-25* • Internal Feedback with Selectable Feedback Delay**

FlashROM Generation and Instantiation in the Design

The SmartGen core generator, available in Libero SoC and Designer, is the only tool that can be used to generate the FlashROM content. SmartGen has several user-friendly features to help generate the FlashROM contents. Instead of selecting each byte and assigning values, you can create a region within a page, modify the region, and assign properties to that region. The FlashROM user interface, shown in Figure 5-10, includes the configuration grid, existing regions list, and properties field. The properties field specifies the region-specific information and defines the data used for that region. You can assign values to the following properties:

1. Static Fixed Data—Enables you to fix the data so it cannot be changed during programming time. This option is useful when you have fixed data stored in this region, which is required for the operation of the design in the FPGA. Key storage is one example.
2. Static Modifiable Data—Select this option when the data in a particular region is expected to be static data (such as a version number, which remains the same for a long duration but could conceivably change in the future). This option enables you to avoid changing the value every time you enter new data.
3. Read from File—This provides the full flexibility of FlashROM usage to the customer. If you have a customized algorithm for generating the FlashROM data, you can specify this setting. You can then generate a text file with data for as many devices as you wish to program, and load that into the FlashPoint programming file generation software to get programming files that include all the data. SmartGen will optionally pass the location of the file where the data is stored if the file is specified in SmartGen. Each text file has only one type of data format (binary, decimal, hex, or ASCII text). The length of each data file must be shorter than or equal to the selected region length. If the data is shorter than the selected region length, the most significant bits will be padded with 0s. For multiple text files for multiple regions, the first lines are for the first device. In SmartGen, **Load Sim. Value From File** allows you to load the first device data in the MEM file for simulation.
4. Auto Increment/Decrement—This scenario is useful when you specify the contents of FlashROM for a large number of devices in a series. You can specify the step value for the serial number and a maximum value for inventory control. During programming file generation, the actual number of devices to be programmed is specified and a start value is fed to the software.

Figure 5-10 • **SmartGen GUI of the FlashROM**

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.

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

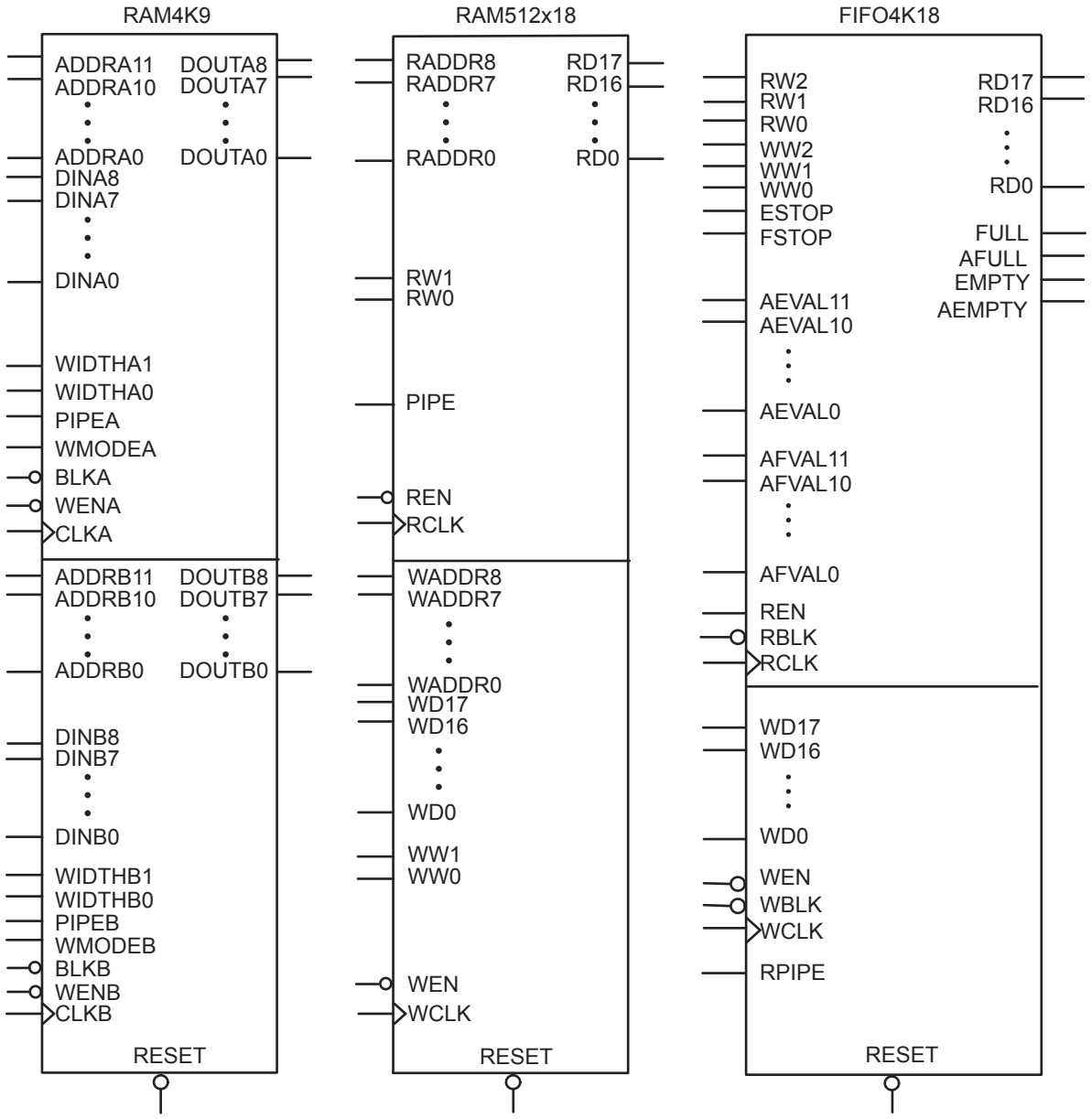
FlashPro User's Guide

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



Notes:

1. *Automotive ProASIC3 devices restrict RAM4K9 to a single port or to dual ports with the same clock 180° out of phase (inverted) between clock pins. In single-port mode, inputs to port B should be tied to ground to prevent errors during compile. This warning applies only to automotive ProASIC3 parts of certain revisions and earlier. Contact Technical Support at soc_tech@microsemi.com for information on the revision number for a particular lot and date code.*
2. *For FIFO4K18, the same clock 180° out of phase (inverted) between clock pins should be used.*

***Figure 6-3* • Supported Basic RAM Macros**

- Designer software will automatically facilitate falling-edge clocks by bubble-pushing the inversion to previous stages.

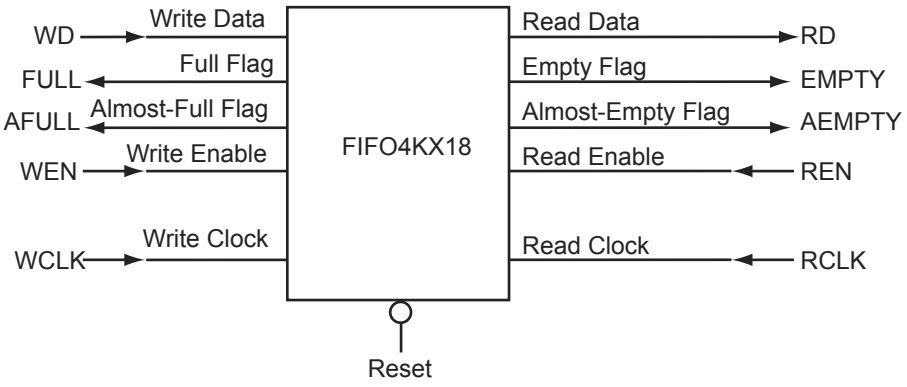


***Figure 6-6* • FIFO4KX18 Block Diagram**

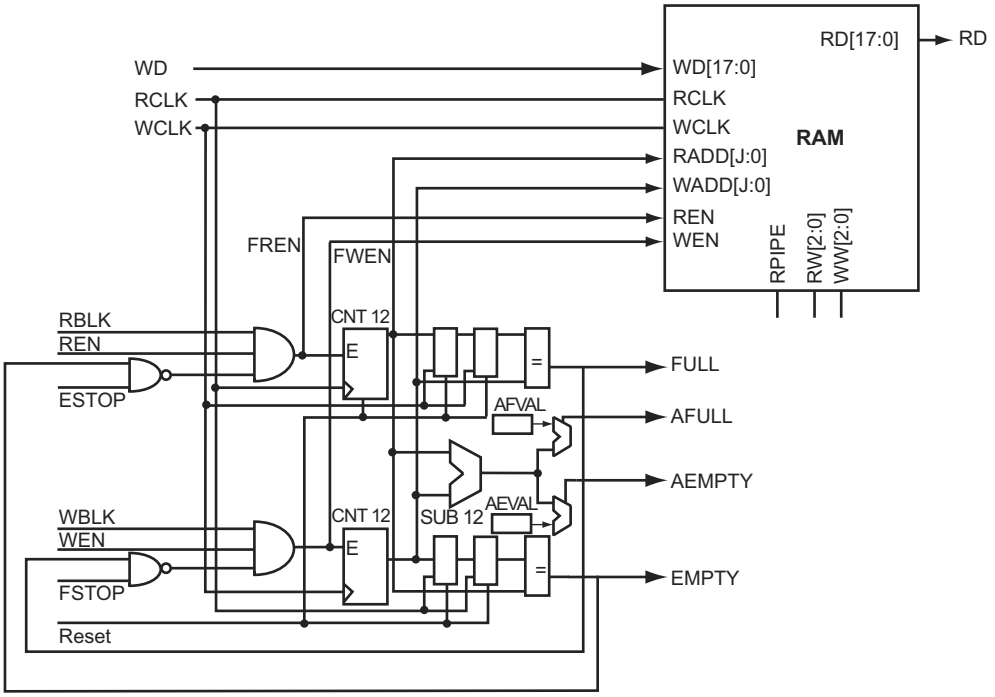


***Figure 6-7* • RAM Block with Embedded FIFO Controller**

The FIFOs maintain a separate read and write address. Whenever the difference between the write address and the read address is greater than or equal to the almost-full value (AFVAL), the Almost-Full flag is asserted. Similarly, the Almost-Empty flag is asserted whenever the difference between the write address and read address is less than or equal to the almost-empty value (AEVAL).

Due to synchronization between the read and write clocks, the Empty flag will deassert after the second read clock edge from the point that the write enable asserts. However, since the Empty flag is synchronized to the read clock, it will assert after the read clock reads the last data in the FIFO. Also, since the Full flag is dependent on the actual hardware configuration, it will assert when the actual physical implementation of the FIFO is full.

For example, when a user configures a 128×18 FIFO, the actual physical implementation will be a 256×18 FIFO element. Since the actual implementation is 256×18, the Full flag will not trigger until the

Pipeline Register

```
module D_pipeline (Data, Clock, Q);

input [3:0] Data;
input Clock;
output [3:0] Q;

reg [3:0] Q;

always @ (posedge Clock) Q <= Data;

endmodule
```

4x4 RAM Block (created by SmartGen Core Generator)

```
module mem_block(DI,DO,WADDR,RADDR,WRB,RDB,WCLOCK,RCLOCK);

input [3:0] DI;
output [3:0] DO;
input [1:0] WADDR, RADDR;
input WRB, RDB, WCLOCK, RCLOCK;

wire WEBP, WEAP, VCC, GND;

VCC VCC_1_net(.Y(VCC));
GND GND_1_net(.Y(GND));
INV WEBUBBLEB(.A(WRB), .Y(WEBP));
RAM4K9 RAMBLOCK0(.ADDRA11(GND), .ADDRA10(GND), .ADDRA9(GND), .ADDRA8(GND),
  .ADDRA7(GND), .ADDRA6(GND), .ADDRA5(GND), .ADDRA4(GND), .ADDRA3(GND), .ADDRA2(GND),
  .ADDRA1(RADDR[1]), .ADDRA0(RADDR[0]), .ADDRB11(GND), .ADDRB10(GND), .ADDRB9(GND),
  .ADDRB8(GND), .ADDRB7(GND), .ADDRB6(GND), .ADDRB5(GND), .ADDRB4(GND), .ADDRB3(GND),
  .ADDRB2(GND), .ADDRB1(WADDR[1]), .ADDRB0(WADDR[0]), .DINA8(GND), .DINA7(GND),
  .DINA6(GND), .DINA5(GND), .DINA4(GND), .DINA3(GND), .DINA2(GND), .DINA1(GND),
  .DINA0(GND), .DINB8(GND), .DINB7(GND), .DINB6(GND), .DINB5(GND), .DINB4(GND),
  .DINB3(DI[3]), .DINB2(DI[2]), .DINB1(DI[1]), .DINB0(DI[0]), .WIDTHA0(GND),
  .WIDTHA1(VCC), .WIDTHB0(GND), .WIDTHB1(VCC), .PIPEA(GND), .PIPEB(GND),
  .WMODEA(GND), .WMODEB(GND), .BLKA(WEAP), .BLKB(WEBP), .WENA(VCC), .WENB(GND),
  .CLKA(RCLOCK), .CLKB(WCLOCK), .RESET(VCC), .DOUTA8(), .DOUTA7(), .DOUTA6(),
  .DOUTA5(), .DOUTA4(), .DOUTA3(DO[3]), .DOUTA2(DO[2]), .DOUTA1(DO[1]),
  .DOUTA0(DO[0]), .DOUTB8(), .DOUTB7(), .DOUTB6(), .DOUTB5(), .DOUTB4(), .DOUTB3(),
  .DOUTB2(), .DOUTB1(), .DOUTB0());
INV WEBUBBLEA(.A(RDB), .Y(WEAP));

endmodule
```

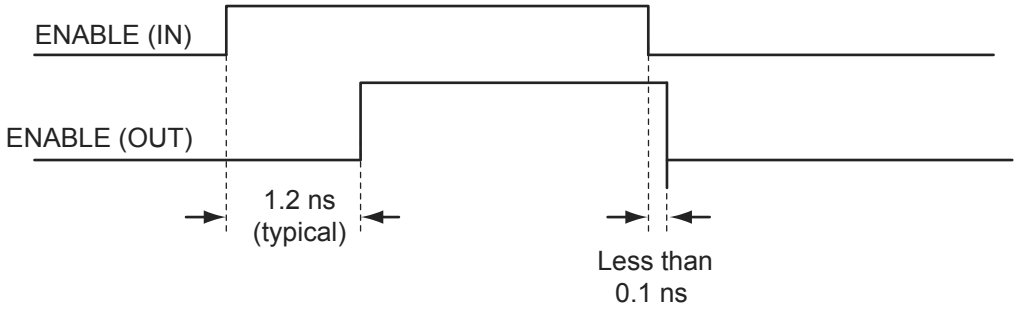


Figure 7-15 • **Timing Diagram (option 2: enables skew circuit)**

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.

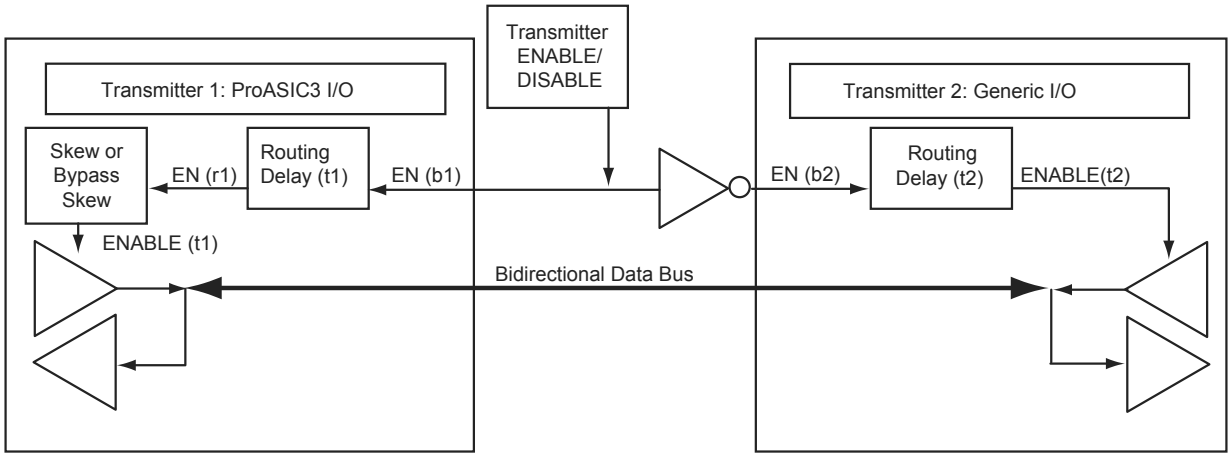


Figure 7-16 • **Example of Implementation of Skew Circuits in Bidirectional Transmission Systems Using IGLOO or ProASIC3 Devices**

I/O Software Support

In Microsemi's Libero software, default settings have been defined for the various I/O standards supported. Changes can be made to the default settings via the use of attributes; however, not all I/O attributes are applicable for all I/O standards. Table 7-17 list the valid I/O attributes that can be manipulated by the user for each I/O standard.

Single-ended I/O standards in low power flash devices support up to five different drive strengths.

***Table 7-17* • IGLOO and ProASIC3 I/O Attributes vs. I/O Standard Applications**

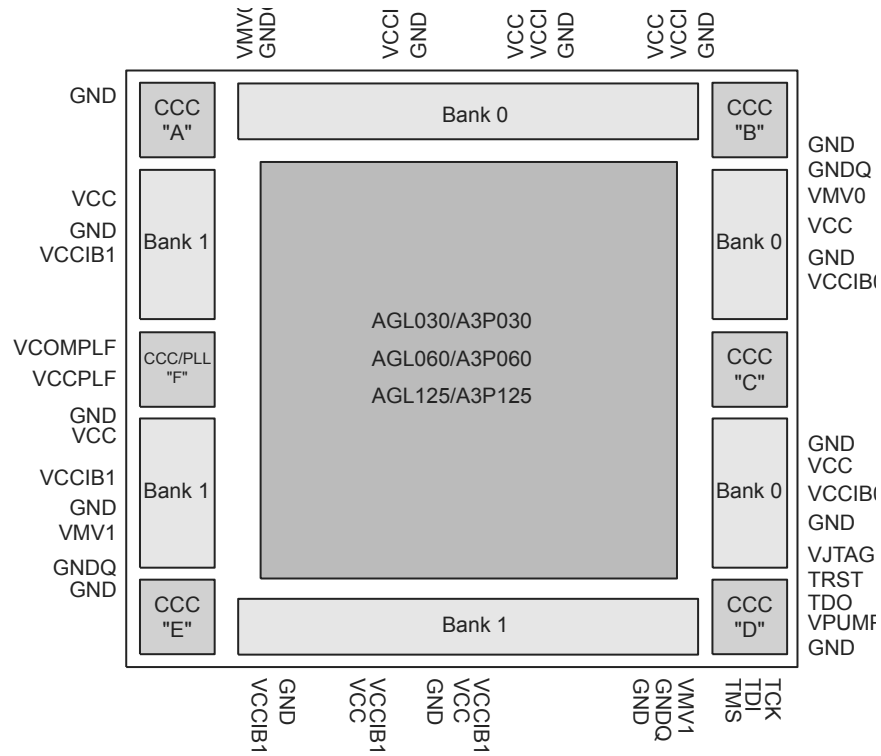
I/O Standard	SLEW (output only)	OUT_DRIVE (output only)	SKEW (all macros with OE)	RES_PULL	OUT_LOAD (output only)	COMBINE_REGISTER
LVTTL/LVCMOS 3.3 V	✓	✓	✓	✓	✓	✓
LVCMOS 2.5 V	✓	✓	✓	✓	✓	✓
LVCMOS 2.5/5.0 V	✓	✓	✓	✓	✓	✓
LVCMOS 1.8 V	✓	✓	✓	✓	✓	✓
LVCMOS 1.5 V	✓	✓	✓	✓	✓	✓
PCI (3.3 V)			✓		✓	✓
PCI-X (3.3 V)	✓		✓		✓	✓
LVDS, B-LVDS, M-LVDS			✓			✓
LVPECL						✓

Note: Applies to all 30 k gate devices.

Table 7-18 lists the default values for the above selectable I/O attributes as well as those that are preset for that I/O standard. See Table 7-14 on page 203 to Table 7-16 on page 203 for SLEW and OUT_DRIVE settings.

***Table 7-18* • IGLOO and ProASIC3 I/O Default Attributes**

I/O Standards	SLEW (output only)	OUT_DRIVE (output only)	SKEW (tribuf and bibuf only)	RES_PULL	OUT_LOAD (output only)	COMBINE_REGISTER
LVTTL/LVCMOS 3.3 V	See Table 7-14 on page 203 to Table 7-16 on page 203.	See Table 7-14 on page 203 to Table 7-16 on page 203.	Off	None	35 pF	–
LVCMOS 2.5 V			Off	None	35 pF	–
LVCMOS 2.5/5.0 V			Off	None	35 pF	–
LVCMOS 1.8 V			Off	None	35 pF	–
LVCMOS 1.5 V			Off	None	35 pF	–
PCI (3.3 V)			Off	None	10 pF	–
PCI-X (3.3 V)			Off	None	10 pF	–
LVDS, B-LVDS, M-LVDS			Off	None	0 pF	–
LVPECL			Off	None	0 pF	–



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**

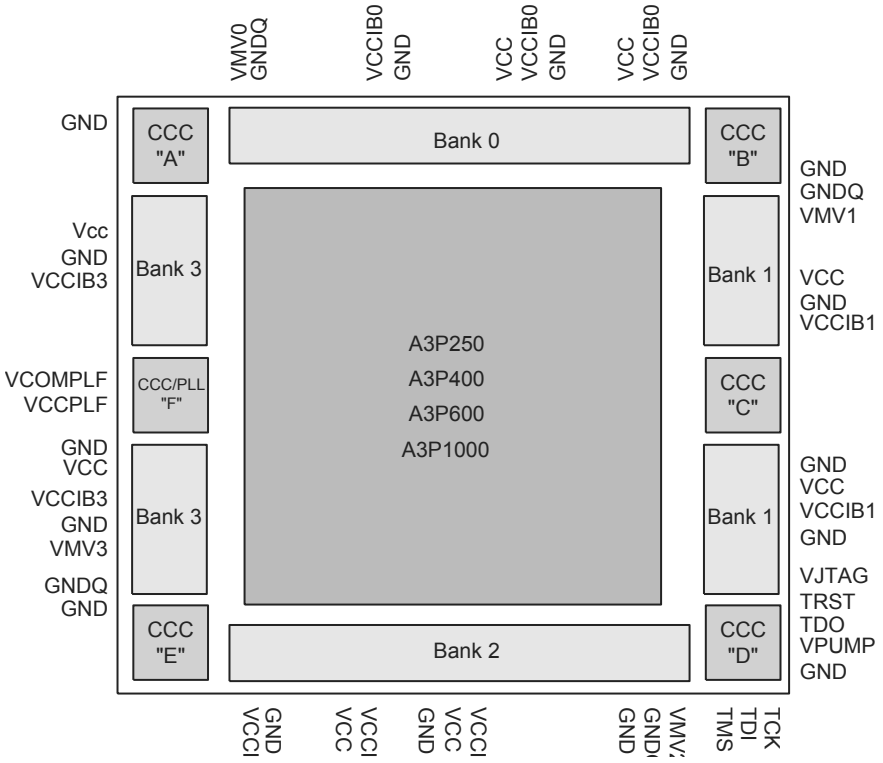


***Figure 7-20* • Naming Conventions of IGLOO and ProASIC3 Devices with Four I/O Banks – Top View**

***Table 8-7* • Maximum I/O Frequency for Single-Ended and Differential I/Os in All Banks in ProASIC3E Devices (maximum drive strength and high slew selected)**

Specification	Maximum Performance		
	ProASIC3E	IGLOOe V2 or V5 Devices, 1.5 V DC Core Supply Voltage	IGLOOe V2, 1.2 V DC Core Supply Voltage
LVTTL/LVCMOS 3.3 V	200 MHz	180 MHz	TBD
LVCMOS 2.5 V	250 MHz	230 MHz	TBD
LVCMOS 1.8 V	200 MHz	180 MHz	TBD
LVCMOS 1.5 V	130 MHz	120 MHz	TBD
PCI	200 MHz	180 MHz	TBD
PCI-X	200 MHz	180 MHz	TBD
HSTL-I	300 MHz	275 MHz	TBD
HSTL-II	300 MHz	275 MHz	TBD
SSTL2-I	300 MHz	275 MHz	TBD
SSTL2-II	300 MHz	275 MHz	TBD
SSTL3-I	300 MHz	275 MHz	TBD
SSTL3-II	300 MHz	275 MHz	TBD
GTL+ 3.3 V	300 MHz	275 MHz	TBD
GTL+ 2.5 V	300 MHz	275 MHz	TBD
GTL 3.3 V	300 MHz	275 MHz	TBD
GTL 2.5 V	300 MHz	275 MHz	TBD
LVDS	350 MHz	300 MHz	TBD
M-LVDS	200 MHz	180 MHz	TBD
B LVDS	200 MHz	180 MHz	TBD
LVPECL	350 MHz	300 MHz	TBD

compatible, which means devices can operate at conventional PCI frequencies (33 MHz and 66 MHz). PCI-X is more fault-tolerant than PCI. It also does not have programmable drive strength.

Voltage-Referenced Standards

I/Os using these standards are referenced to an external reference voltage (VREF) and are supported on E devices only.

HSTL Class I and II (High-Speed Transceiver Logic)

These are general-purpose, high-speed 1.5 V bus standards (EIA/JESD 8-6) for signaling between integrated circuits. The signaling range is 0 V to 1.5 V, and signals can be either single-ended or differential. HSTL requires a differential amplifier input buffer and a push-pull output buffer. The reference voltage (VREF) is 0.75 V. These standards are used in the memory bus interface with data switching capability of up to 400 MHz. The other advantages of these standards are low power and fewer EMI concerns.

HSTL has four classes, of which low power flash devices support Class I and II. These classes are defined by standard EIA/JESD 8-6 from the Electronic Industries Alliance (EIA):

- Class I – Unterminated or symmetrically parallel-terminated
- Class II – Series-terminated
- Class III – Asymmetrically parallel-terminated
- Class IV – Asymmetrically double-parallel-terminated

SSTL2 Class I and II (Stub Series Terminated Logic 2.5 V)

These are general-purpose 2.5 V memory bus standards (JESD 8-9) for driving transmission lines, designed specifically for driving the DDR SDRAM modules used in computer memory. SSTL2 requires a differential amplifier input buffer and a push-pull output buffer. The reference voltage (VREF) is 1.25 V.

SSTL3 Class I and II (Stub Series Terminated Logic 3.3 V)

These are general-purpose 3.3 V memory bus standards (JESD 8-8) for driving transmission lines. SSTL3 requires a differential amplifier input buffer and a push-pull output buffer. The reference voltage (VREF) is 1.5 V.

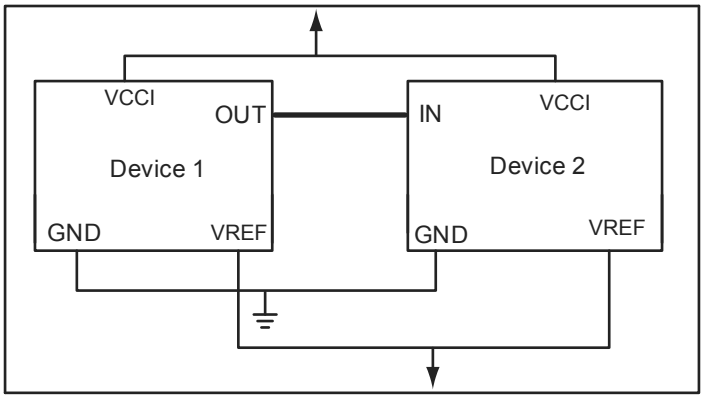


***Figure 8-7* • SSTL and HSTL Topology**

Software-Controlled I/O Attributes

Users may modify these programmable I/O attributes using the I/O Attribute Editor. Modifying an I/O attribute may result in a change of state in Designer. Table 9-2 details which steps have to be re-run as a function of modified I/O attribute.

***Table 9-2* • Designer State (resulting from I/O attribute modification)**

	Designer States[1]				
I/O Attribute	**Compile**	**Layout**	**Fuse**	**Timing**	**Power**
Slew Control[2]	No	No	Yes	Yes	Yes
Output Drive (mA)	No	No	Yes	Yes	Yes
Skew Control	No	No	Yes	Yes	Yes
Resistor Pull	No	No	Yes	Yes	Yes
Input Delay	No	No	Yes	Yes	Yes
Schmitt Trigger	No	No	Yes	Yes	Yes
OUT_LOAD	No	No	No	Yes	Yes
COMBINE_REGISTER	Yes	Yes	N/A	N/A	N/A

Notes:

1. *No = Remains the same, Yes = Re-run the step, N/A = Not applicable*
2. *Skew control does not apply to IGLOO nano, IGLOO PLUS, and ProASIC3 nano devices.*
3. *Programmable input delay is applicable only for ProASIC3E, ProASIC3EL, RT ProASIC3, and IGLOOe devices.*

Compiling the Design

During Compile, a PDC I/O constraint file can be imported along with the netlist file. If only the netlist file is compiled, certain I/O assignments need to be completed before proceeding to Layout. All constraints that can be entered in PDC can also be entered using ChipPlanner, I/O Attribute Editor, and PinEditor.

There are certain rules that must be followed in implementing I/O register combining and the I/O DDR macro (refer to the I/O Registers section of the handbook for the device that you are using and the "DDR" section on page 256 for details). Provided these rules are met, the user can enable or disable I/O register combining by using the PDC command `set_io portname –register yes|no` in the I/O Attribute Editor or selecting a check box in the Compile Options dialog box (see Figure 9-7). The Compile Options dialog box appears when the design is compiled for the first time. It can also be accessed by choosing **Options** > **Compile** during successive runs. I/O register combining is off by default. The PDC command overrides the setting in the Compile Options dialog box.

***Figure 9-7* • Setting Register Combining During Compile**

Understanding the Compile Report

The I/O bank report is generated during Compile and displayed in the log window. This report lists the I/O assignments necessary before Layout can proceed.

When Designer is started, the I/O Bank Assigner tool is run automatically if the Layout command is executed. The I/O Bank Assigner takes care of the necessary I/O assignments. However, these assignments can also be made manually with MVN or by importing the PDC file. Refer to the "Assigning Technologies and VREF to I/O Banks" section on page 264 for further description.

The I/O bank report can also be extracted from Designer by choosing **Tools** > **Report** and setting the Report Type to **IOBank**.

This report has the following tables: I/O Function, I/O Technology, I/O Bank Resource Usage, and I/O Voltage Usage. This report is useful if the user wants to do I/O assignments manually.

Input Support for DDR

The basic structure to support a DDR input is shown in Figure 10-2. Three input registers are used to capture incoming data, which is presented to the core on each rising edge of the I/O register clock. Each I/O tile supports DDR inputs.

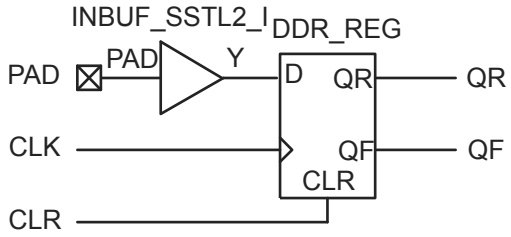


***Figure 10-2* • DDR Input Register Support in Low Power Flash Devices**

Output Support for DDR

The basic DDR output structure is shown in Figure 10-1 on page 271. New data is presented to the output every half clock cycle.

Note: DDR macros and I/O registers do not require additional routing. The combiner automatically recognizes the DDR macro and pushes its registers to the I/O register area at the edge of the chip. The routing delay from the I/O registers to the I/O buffers is already taken into account in the DDR macro.

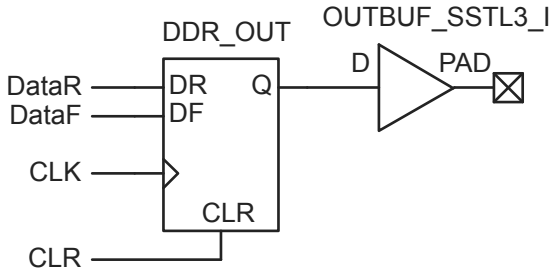


***Figure 10-3* • DDR Output Register (SSTL3 Class I)**

Volume Programming Services

Device Type Supported: Flash and Antifuse

Once the design is stable for applications with large production volumes, preprogrammed devices can be purchased. Table 11-2 describes the volume programming services.

***Table 11-2* • Volume Programming Services**

Programmer	Vendor	Availability
In-House Programming	Microsemi	Contact Microsemi Sales
Distributor Programming Centers	Memec Unique	Contact Distribution
Independent Programming Centers	Various	Contact Vendor

Advantages: As programming is outsourced, this solution is easier to implement than creating a substantial in-house programming capability. As programming houses specialize in large-volume programming, this is often the most cost-effective solution.

Limitations: There are some logistical issues with the use of a programming service provider, such as the transfer of programming files and the approval of First Articles. By definition, the programming file must be released to a third-party programming house. Nondisclosure agreements (NDAs) can be signed to help ensure data protection; however, for extremely security-conscious designs, this may not be an option.

- Microsemi In-House Programming

  When purchasing Microsemi devices in volume, IHP can be requested as part of the purchase. If this option is chosen, there is a small cost adder for each device programmed. Each device is marked with a special mark to distinguish it from blank parts. Programming files for the design will be sent to Microsemi. Sample parts with the design programmed, First Articles, will be returned for customer approval. Once approval of First Articles has been received, Microsemi will proceed with programming the remainder of the order. To request Microsemi IHP, contact your local Microsemi representative.

- Distributor Programming Centers

  If purchases are made through a distributor, many distributors will provide programming for their customers. Consult with your preferred distributor about this option.

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

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

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";
============================================
```

Brownout Voltage

Brownout is a condition in which the voltage supplies are lower than normal, causing the device to malfunction as a result of insufficient power. In general, Microsemi does not guarantee the functionality of the design inside the flash FPGA if voltage supplies are below their minimum recommended operating condition. Microsemi has performed measurements to characterize the brownout levels of FPGA power supplies. Refer to Table 18-3 for device-specific brownout deactivation levels. For the purpose of characterization, a direct path from the device input to output is monitored while voltage supplies are lowered gradually. The brownout point is defined as the voltage level at which the output stops following the input. Characterization tests performed on several IGLOO, ProASIC3L, and ProASIC3 devices in typical operating conditions showed the brownout voltage levels to be within the specification.

During device power-down, the device I/Os become tristated once the first supply in the power-down sequence drops below its brownout deactivation voltage.

***Table 18-3* • Brownout Deactivation Levels for VCC and VCCI**

Devices	VCC B**rownout Deactivation Level (V)**	VCCI Br**ownout Deactivation Level (V)**
ProASIC3, ProASIC3 nano, IGLOO, IGLOO nano, IGLOO PLUS and ProASIC3L devices running at VCC = 1.5 V	0.75 V ± 0.25 V	0.8 V ± 0.3 V
IGLOO, IGLOO nano, IGLOO PLUS, and ProASIC3L devices running at VCC = 1.2 V	0.75 V ± 0.2 V	0.8 V ± 0.15 V

PLL Behavior at Brownout Condition

When PLL power supply voltage and/or V_{CC} levels drop below the V_{CC} brownout levels mentioned above for 1.5 V and 1.2 V devices, the PLL output lock signal goes LOW and/or the output clock is lost. The following sections explain PLL behavior during and after the brownout condition.

VCCPLL and VCC **Tied Together**

In this condition, both VCC and VCCPLL drop below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level. During the brownout recovery, once VCCPLL and VCC reach the activation point (0.85 ± 0.25 V or ± 0.2 V) again, the PLL output lock signal may still remain LOW with the PLL output clock signal toggling. If this condition occurs, there are two ways to recover the PLL output lock signal:

1. Cycle the power supplies of the PLL (power off and on) by using the PLL POWERDOWN signal.
2. Turn off the input reference clock to the PLL and then turn it back on.

Only VCCPLL **Is at Brownout**

In this case, only VCCPLL drops below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level and the VCC supply remains at nominal recommended operating voltage (1.5 V ± 0.075 V for 1.5 V devices and 1.2 V ± 0.06 V for 1.2 V devices). In this condition, the PLL behavior after brownout recovery is similar to initial power-up condition, and the PLL will regain lock automatically after VCCPLL is ramped up above the activation level (0.85 ± 0.25 V or ± 0.2 V). No intervention is necessary in this case.

Only VCC **Is at Brownout**

In this condition, VCC drops below the 0.75 V (± 0.25 V or ± 0.2 V) brownout level and VCCPLL remains at nominal recommended operating voltage (1.5 V ± 0.075 V for 1.5 V devices and 1.2 V ± 0.06 V for 1.2 V devices). During the brownout recovery, once VCC reaches the activation point again (0.85 ± 0.25 V or ± 0.2 V), the PLL output lock signal may still remain LOW with the PLL output clock signal toggling. If this condition occurs, there are two ways to recover the PLL output lock signal:

1. Cycle the power supplies of the PLL (power off and on) by using the PLL POWERDOWN signal.
2. Turn off the input reference clock to the PLL and then turn it back on.

It is important to note that Microsemi recommends using a monotonic power supply or voltage regulator to ensure proper power-up behavior.