



Welcome to [E-XFL.COM](https://www.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	516096
Number of I/O	620
Number of Gates	3000000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	896-BGA
Supplier Device Package	896-FBGA (31x31)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pe3000l-1fg896

Using Spines of Occupied Global Networks

When a signal is assigned to a global network, the flash switches are programmed to set the MUX select lines (explained in the "Clock Aggregation Architecture" section on page 61) to drive the spines of that network with the global net. However, if the global net is restricted from reaching into the scope of a spine, the MUX drivers of that spine are available for other high-fanout or critical signals (Figure 3-20).

For example, if you want to limit the CLK1_c signal to the left half of the chip and want to use the right side of the same global network for CLK2_c, you can add the following PDC commands:

```
define_region -name region1 -type inclusive 0 0 34 29
assign_net_macros region1 CLK1_c
assign_local_clock -net CLK2_c -type chip B2
```

Figure 3-20 • Design Example Using Spines of Occupied Global Networks

Conclusion

IGLOO, Fusion, and ProASIC3 devices contain 18 global networks: 6 chip global networks and 12 quadrant global networks. These global networks can be segmented into local low-skew networks called spines. The spines provide low-skew networks for the high-fanout signals of a design. These allow you up to 252 different internal/external clocks in an A3PE3000 device. This document describes the architecture for the global network, plus guidelines and methodologies in assigning signals to globals and spines.

Related Documents

User's Guides

IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide
http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

4 – Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs

Introduction

This document outlines the following device information: Clock Conditioning Circuit (CCC) features, PLL core specifications, functional descriptions, software configuration information, detailed usage information, recommended board-level considerations, and other considerations concerning clock conditioning circuits and global networks in low power flash devices or mixed signal FPGAs.

Overview of Clock Conditioning Circuitry

In Fusion, IGLOO, and ProASIC3 devices, the CCCs are used to implement frequency division, frequency multiplication, phase shifting, and delay operations. The CCCs are available in six chip locations—each of the four chip corners and the middle of the east and west chip sides. For device-specific variations, refer to the "Device-Specific Layout" section on page 94.

The CCC is composed of the following:

- PLL core
- 3 phase selectors
- 6 programmable delays and 1 fixed delay that advances/delays phase
- 5 programmable frequency dividers that provide frequency multiplication/division (not shown in Figure 4-6 on page 87 because they are automatically configured based on the user's required frequencies)
- 1 dynamic shift register that provides CCC dynamic reconfiguration capability

Figure 4-1 provides a simplified block diagram of the physical implementation of the building blocks in each of the CCCs.

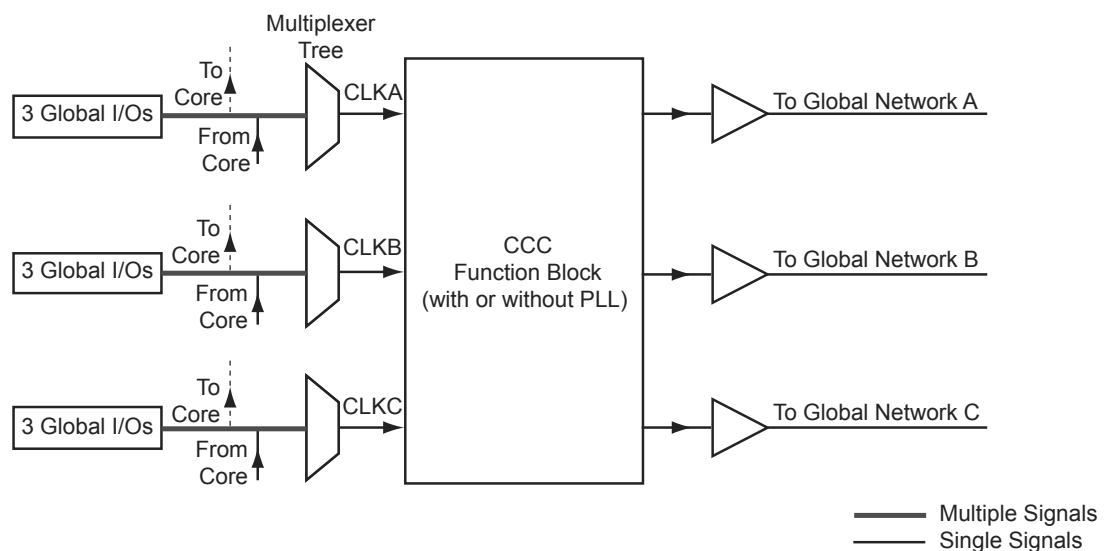


Figure 4-1 • Overview of the CCCs Offered in Fusion, IGLOO, and ProASIC3

difference will cause the VCO to increase its frequency until the output signal is phase-identical to the input after undergoing division. In other words, lock in both frequency and phase is achieved when the output frequency is M times the input. Thus, clock division in the feedback path results in multiplication at the output.

A similar argument can be made when the delay element is inserted into the feedback path. To achieve steady-state lock, the VCO output signal will be delayed by the input period less the feedback delay. For periodic signals, this is equivalent to time-advancing the output clock by the feedback delay.

Another key parameter of a PLL system is the acquisition time. Acquisition time is the amount of time it takes for the PLL to achieve lock (i.e., phase-align the feedback signal with the input reference clock). For example, suppose there is no voltage applied to the VCO, allowing it to operate at its free-running frequency. Should an input reference clock suddenly appear, a lock would be established within the maximum acquisition time.

Functional Description

This section provides detailed descriptions of PLL block functionality: clock dividers and multipliers, clock delay adjustment, phase adjustment, and dynamic PLL configuration.

Clock Dividers and Multipliers

The PLL block contains five programmable dividers. Figure 4-20 shows a simplified PLL block.

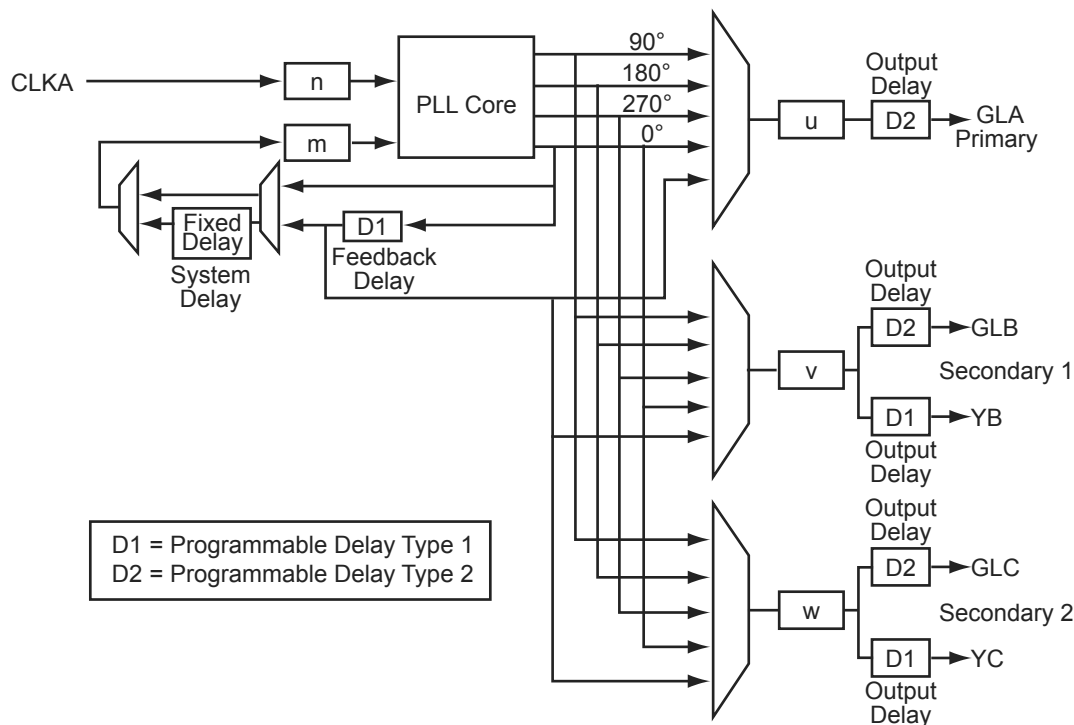


Figure 4-20 • PLL Block Diagram

Phase Adjustment

The four phases available (0, 90, 180, 270) are phases with respect to VCO (PLL output). The VCO is divided to achieve the user's CCC required output frequency (GLA, YB/GLB, YC/GLC). The division happens after the selection of the VCO phase. The effective phase shift is actually the VCO phase shift divided by the output divider. This is why the visual CCC shows both the actual achievable phase and more importantly the actual delay that is equivalent to the phase shift that can be achieved.

Dynamic PLL Configuration

The CCCs can be configured both statically and dynamically.

In addition to the ports available in the Static CCC, the Dynamic CCC has the dynamic shift register signals that enable dynamic reconfiguration of the CCC. With the Dynamic CCC, the ports CLKB and CLKC are also exposed. All three clocks (CLKA, CLKB, and CLKC) can be configured independently.

The CCC block is fully configurable. The following two sources can act as the CCC configuration bits.

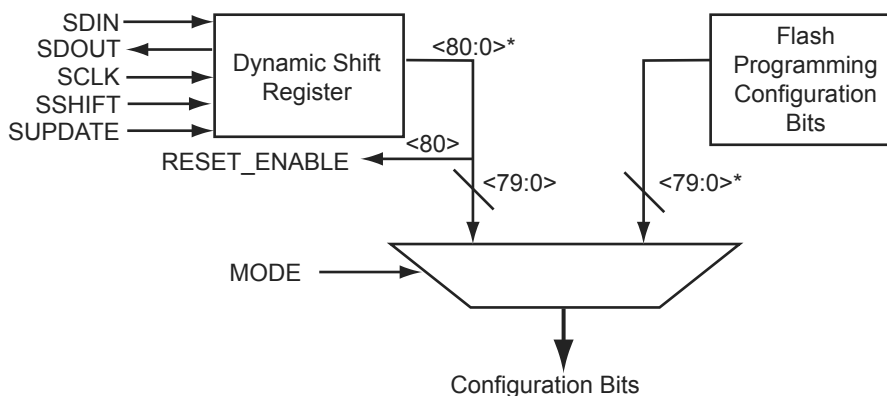
Flash Configuration Bits

The flash configuration bits are the configuration bits associated with programmed flash switches. These bits are used when the CCC is in static configuration mode. Once the device is programmed, these bits cannot be modified. They provide the default operating state of the CCC.

Dynamic Shift Register Outputs

This source does not require core reprogramming and allows core-driven dynamic CCC reconfiguration. When the dynamic register drives the configuration bits, the user-defined core circuit takes full control over SDIN, SDOUT, SCLK, SSHIFT, and SUPDATE. The configuration bits can consequently be dynamically changed through shift and update operations in the serial register interface. Access to the logic core is accomplished via the dynamic bits in the specific tiles assigned to the PLLs.

Figure 4-21 illustrates a simplified block diagram of the MUX architecture in the CCCs.



Note: *For Fusion, bit <88:81> is also needed.

Figure 4-21 • The CCC Configuration MUX Architecture

The selection between the flash configuration bits and the bits from the configuration register is made using the MODE signal shown in Figure 4-21. If the MODE signal is logic HIGH, the dynamic shift register configuration bits are selected. There are 81 control bits to configure the different functions of the CCC.

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

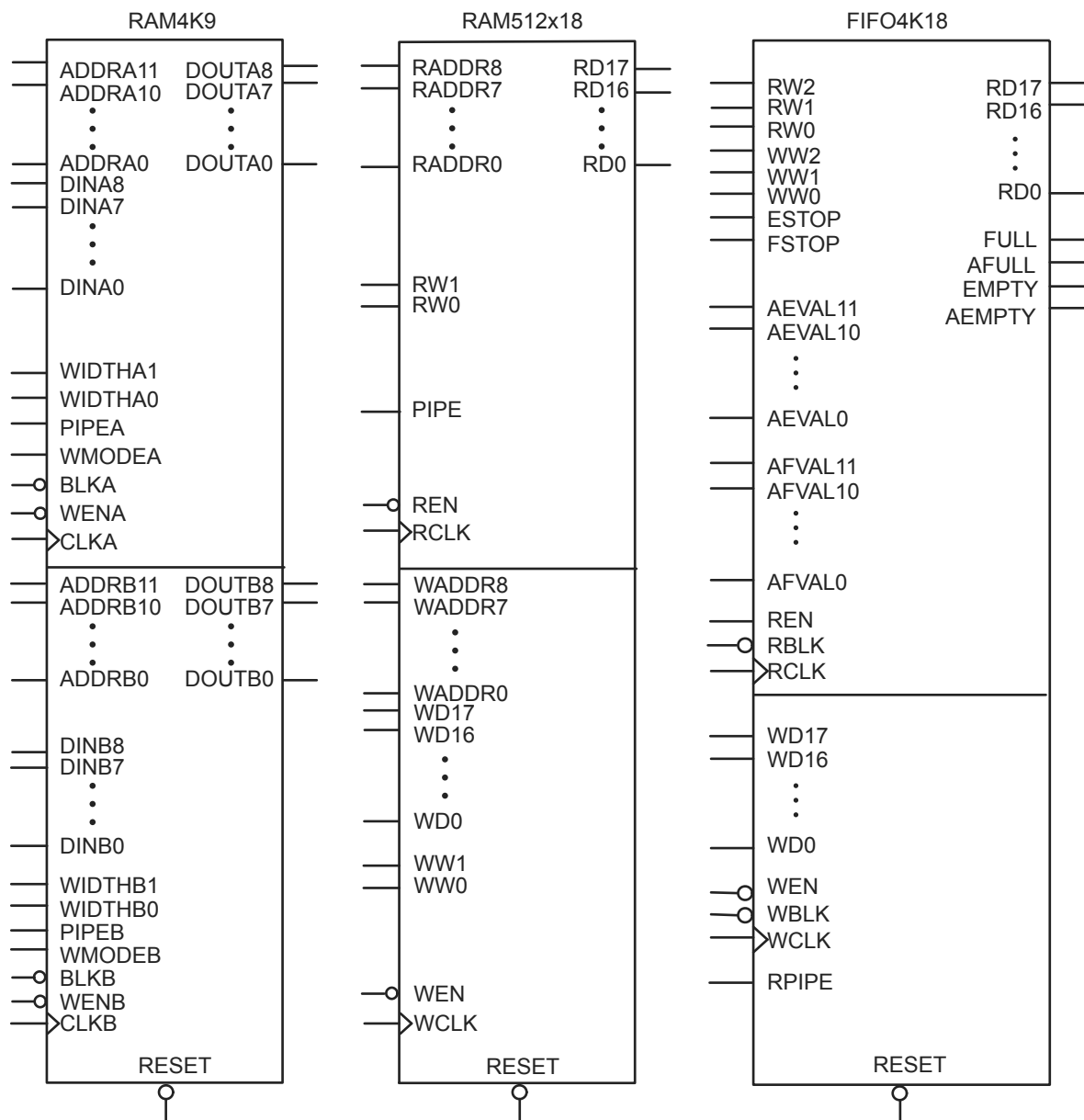
FlashPro User's Guide

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 <i>FlashPoint User's Guide</i> was removed from the "User's Guides" section, as its content is now part of the <i>FlashPro User's Guide</i> .	146
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 5-1 • Flash-Based FPGAs: <ul style="list-style-type: none"> ProASIC3L was updated to include 1.5 V. The number of PLLs for ProASIC3E was changed from five to six. 	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

Table 6-2 • Allowable Aspect Ratio Settings for WIDTHA[1:0]

WIDTHA[1:0]	WIDTHB[1:0]	D×W
00	00	4k×1
01	01	2k×2
10	10	1k×4
11	11	512×9

Note: The aspect ratio settings are constant and cannot be changed on the fly.

BLKA and BLKB

These signals are active-low and will enable the respective ports when asserted. When a BLKx signal is deasserted, that port's outputs hold the previous value.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, BLKB should be tied to ground.

WENA and WENB

These signals switch the RAM between read and write modes for the respective ports. A LOW on these signals indicates a write operation, and a HIGH indicates a read.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, WENB should be tied to ground.

CLKA and CLKB

These are the clock signals for the synchronous read and write operations. These can be driven independently or with the same driver.

Note: For Automotive ProASIC3 devices, dual-port mode is supported if the clocks to the two SRAM ports are the same and 180° out of phase (i.e., the port A clock is the inverse of the port B clock). For use of this macro as a single-port SRAM, the inputs and clock of one port should be tied off (grounded) to prevent errors during design compile.

PIPEA and PIPEB

These signals are used to specify pipelined read on the output. A LOW on PIPEA or PIPEB indicates a nonpipelined read, and the data appears on the corresponding output in the same clock cycle. A HIGH indicates a pipelined read, and data appears on the corresponding output in the next clock cycle.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, PIPEB should be tied to ground. For use in dual-port mode, the same clock with an inversion between the two clock pins of the macro should be used in the design to prevent errors during compile.

WMODEA and WMODEB

These signals are used to configure the behavior of the output when the RAM is in write mode. A LOW on these signals makes the output retain data from the previous read. A HIGH indicates pass-through behavior, wherein the data being written will appear immediately on the output. This signal is overridden when the RAM is being read.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, WMODEB should be tied to ground.

RESET

This active-low signal resets the control logic, forces the output hold state registers to zero, disables reads and writes from the SRAM block, and clears the data hold registers when asserted. It does not reset the contents of the memory array.

While the RESET signal is active, read and write operations are disabled. As with any asynchronous reset signal, care must be taken not to assert it too close to the edges of active read and write clocks.

ADDRA and ADDRb

These are used as read or write addresses, and they are 12 bits wide. When a depth of less than 4 k is specified, the unused high-order bits must be grounded (Table 6-3 on page 155).

Initializing the RAM/FIFO

The SRAM blocks can be initialized with data to use as a lookup table (LUT). Data initialization can be accomplished either by loading the data through the design logic or through the UJTAG interface. The UJTAG macro is used to allow access from the JTAG port to the internal logic in the device. By sending the appropriate initialization string to the JTAG Test Access Port (TAP) Controller, the designer can put the JTAG circuitry into a mode that allows the user to shift data into the array logic through the JTAG port using the UJTAG macro. For a more detailed explanation of the UJTAG macro, refer to the "FlashROM in Microsemi's Low Power Flash Devices" section on page 133.

A user interface is required to receive the user command, initialization data, and clock from the UJTAG macro. The interface must synchronize and load the data into the correct RAM block of the design. The main outputs of the user interface block are the following:

- Memory block chip select: Selects a memory block for initialization. The chip selects signals for each memory block that can be generated from different user-defined pockets or simple logic, such as a ring counter (see below).
- Memory block write address: Identifies the address of the memory cell that needs to be initialized.
- Memory block write data: The interface block receives the data serially from the UTDI port of the UJTAG macro and loads it in parallel into the write data ports of the memory blocks.
- Memory block write clock: Drives the WCLK of the memory block and synchronizes the write data, write address, and chip select signals.

Figure 6-8 shows the user interface between UJTAG and the memory blocks.

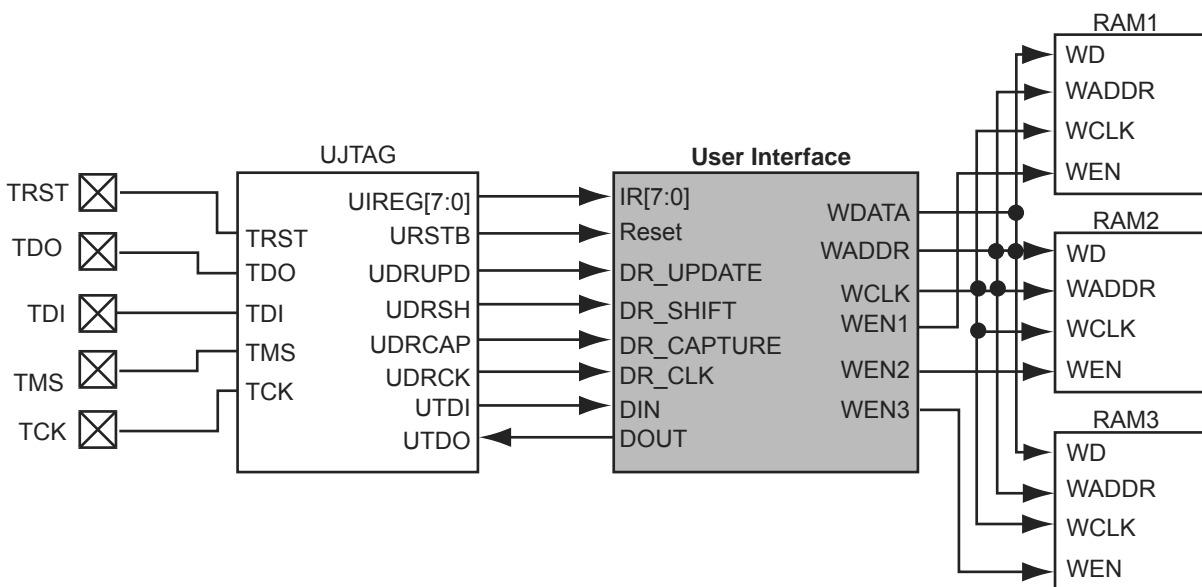


Figure 6-8 • Interfacing TAP Ports and SRAM Blocks

An important component of the interface between the UJTAG macro and the RAM blocks is a serial-in/parallel-out shift register. The width of the shift register should equal the data width of the RAM blocks. The RAM data arrives serially from the UTDI output of the UJTAG macro. The data must be shifted into a shift register clocked by the JTAG clock (provided at the UDRCK output of the UJTAG macro).

Then, after the shift register is fully loaded, the data must be transferred to the write data port of the RAM block. To synchronize the loading of the write data with the write address and write clock, the output of the shift register can be pipelined before driving the RAM block.

The write address can be generated in different ways. It can be imported through the TAP using a different instruction opcode and another shift register, or generated internally using a simple counter. Using a counter to generate the address bits and sweep through the address range of the RAM blocks is

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
```

Date	Change	Page
June 2011 (continued)	The following sentence was removed from the "LVCMOS (Low-Voltage CMOS)" section (SAR 22634): "All these versions use a 3.3 V-tolerant CMOS input buffer and a push-pull output buffer."	184
	Hot-insertion was changed to "No" for other IGLOO and all ProASIC3 devices in Table 7-12 • I/O Hot-Swap and 5 V Input Tolerance Capabilities in IGLOO and ProASIC3 Devices (SAR 24526).	193
	The "Electrostatic Discharge Protection" section was revised to remove references to tolerances (refer to the <i>Reliability Report</i> for tolerances). The Machine Model (MM) is not supported and was deleted from this section (SAR 24385).	192
	The "I/O Interfacing" section was revised to state that low power flash devices are 5 V-input- and 5 V-output-tolerant if certain I/O standards are selected, removing "without adding any extra circuitry," which was incorrect (SAR 21404).	208
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)	The terminology in the "Low Power Flash Device I/O Support" section was revised.	176
v1.3 (October 2008)	The "Low Power Flash Device I/O Support" section was revised to include new families and make the information more concise.	176
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 7-1 • Flash-Based FPGAs: <ul style="list-style-type: none"> ProASIC3L was updated to include 1.5 V. The number of PLLs for ProASIC3E was changed from five to six. 	176
v1.1 (March 2008)	Originally, this document contained information on all IGLOO and ProASIC3 families. With the addition of new families and to highlight the differences between the features, the document has been separated into 3 documents: This document contains information specific to IGLOO, ProASIC3, and ProASIC3L. "I/O Structures in IGLOOe and ProASIC3E Devices" in the <i>ProASIC3E FPGA Fabric User's Guide</i> contains information specific to IGLOOe, ProASIC3E, and ProASIC3EL I/O features. "I/O Structures in IGLOO PLUS Devices" in the <i>IGLOO PLUS FPGA Fabric User's Guide</i> contains information specific to IGLOO PLUS I/O features.	N/A

Features Supported on Every I/O

Table 8-6 lists all features supported by transmitter/receiver for single-ended and differential I/Os. Table 8-7 on page 219 lists the performance of each I/O technology.

Table 8-6 • IGLOOe and ProASIC3E I/O Features

Feature	Description
All I/O	<ul style="list-style-type: none"> High performance (Table 8-7 on page 219) Electrostatic discharge protection I/O register combining option
Single-Ended and Voltage-Referenced Transmitter Features	<ul style="list-style-type: none"> Hot-swap in every mode except PCI or 5 V–input–tolerant (these modes use clamp diodes and do not allow hot-swap) Activation of hot-insertion (disabling the clamp diode) is selectable by I/Os. Output slew rate: 2 slew rates Weak pull-up and pull-down resistors Output drive: 5 drive strengths Programmable output loading Skew between output buffer enable/disable time: 2 ns delay on rising edge and 0 ns delay on falling edge (see "Selectable Skew between Output Buffer Enable and Disable Times" section on page 236 for more information) LVTTL/LVCMOS 3.3 V outputs compatible with 5 V TTL inputs
Single-Ended Receiver Features	<ul style="list-style-type: none"> 5 V–input–tolerant receiver (Table 8-13 on page 231) Schmitt trigger option Programmable delay: 0 ns if bypassed, 0.625 ns with '000' setting, 6.575 ns with '111' setting, 0.85-ns intermediate delay increments (at 25°C, 1.5 V) Separate ground plane for GNDQ pin and power plane for VMV pin are used for input buffer to reduce output-induced noise.
Voltage-Referenced Differential Receiver Features	<ul style="list-style-type: none"> Programmable delay: 0 ns if bypassed, 0.46 ns with '000' setting, 4.66 ns with '111' setting, 0.6-ns intermediate delay increments (at 25°C, 1.5 V) Separate ground plane for GNDQ pin and power plane for VMV pin are used for input buffer to reduce output-induced noise.
CMOS-Style LVDS, B-LVDS, M-LVDS, or LVPECL Transmitter	<ul style="list-style-type: none"> Two I/Os and external resistors are used to provide a CMOS-style LVDS, DDR LVDS, B-LVDS, and M-LVDS/LVPECL transmitter solution. Activation of hot-insertion (disabling the clamp diode) is selectable by I/Os. High slew rate Weak pull-up and pull-down resistors Programmable output loading
LVDS, DDR LVDS, B-LVDS, and M-LVDS/LVPECL Differential Receiver Features	<ul style="list-style-type: none"> Programmable delay: 0 ns if bypassed, 0.46 ns with '000' setting, 4.66 ns with '111' setting, 0.6-ns intermediate delay increments (at 25°C, 1.5 V)

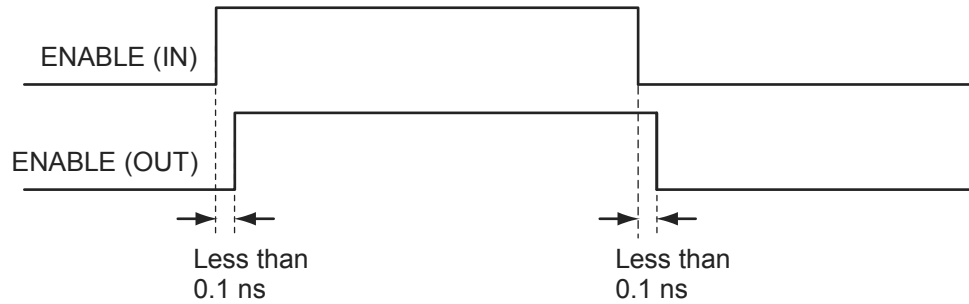


Figure 8-15 • Timing Diagram (option 1: bypasses skew circuit)

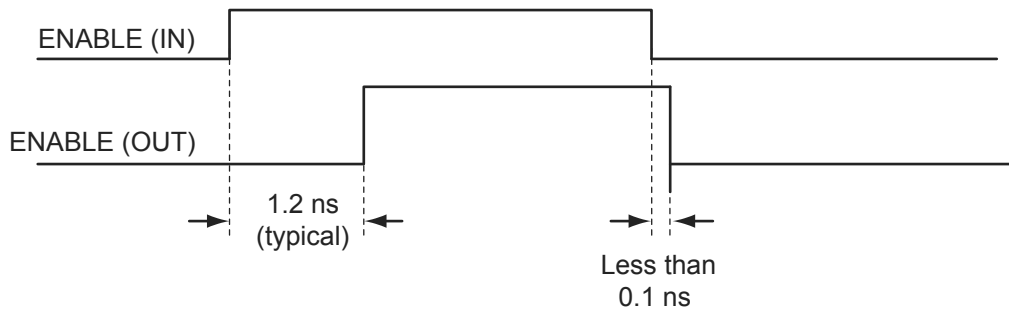


Figure 8-16 • Timing Diagram (option 2: enables skew circuit)

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 8-17 presents an example of the skew circuit implementation in a bidirectional communication system. Figure 8-18 on page 238 shows how bus contention is created, and Figure 8-19 on page 238 shows how it can be avoided with the skew circuit.

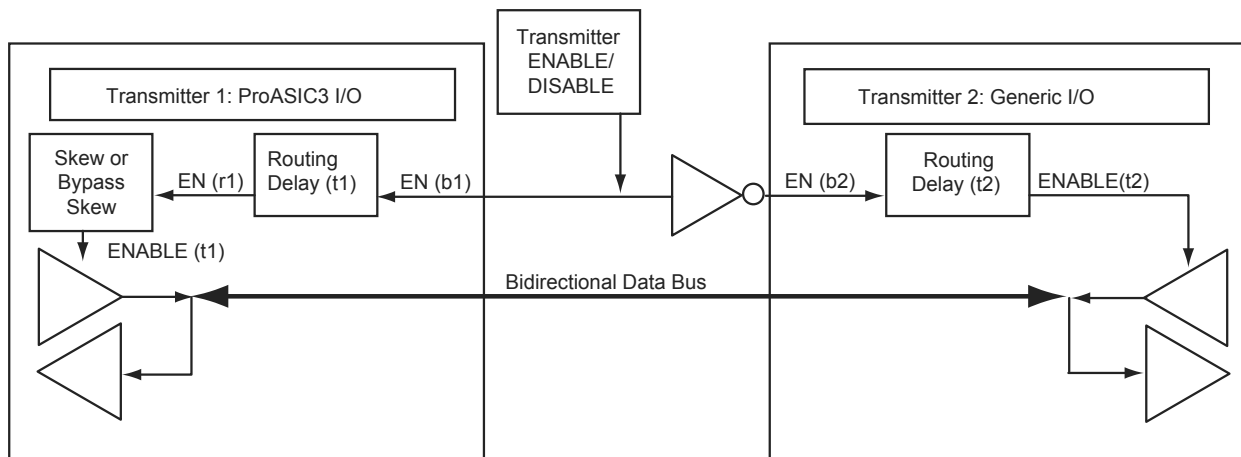


Figure 8-17 • Example of Implementation of Skew Circuits in Bidirectional Transmission Systems Using IGLOO or ProASIC3 Devices

The procedure is as follows:

1. Select the bank to which you want VCCI to be assigned from the **Choose Bank** list.
2. Select the I/O standards for that bank. If you select any standard, the tool will automatically show all compatible standards that have a common VCCI voltage requirement.
3. Click **Apply**.
4. Repeat steps 1–3 to assign VCCI voltages to other banks. Refer to Figure 9-11 on page 263 to find out how many I/O banks are needed for VCCI bank assignment.

Manually Assigning VREF Pins

Voltage-referenced inputs require an input reference voltage (VREF). The user must assign VREF pins before running Layout. Before assigning a VREF pin, the user must set a VREF technology for the bank to which the pin belongs.

VREF Rules for the Implementation of Voltage-Referenced I/O Standards

The VREF rules are as follows:

1. Any I/O (except JTAG I/Os) can be used as a V_{REF} pin.
2. One V_{REF} pin can support up to 15 I/Os. It is recommended, but not required, that eight of them be on one side and seven on the other side (in other words, all 15 can still be on one side of VREF).
3. SSTL3 (I) and (II): Up to 40 I/Os per north or south bank in any position
4. LVPECL / GTL+ 3.3 V / GTL 3.3 V: Up to 48 I/Os per north or south bank in any position (not applicable for IGLOO nano and ProASIC3 nano devices)
5. SSTL2 (I) and (II) / GTL+ 2.5 V / GTL 2.5 V: Up to 72 I/Os per north or south bank in any position
6. VREF minibanks partition rule: Each I/O bank is physically partitioned into VREF minibanks. The VREF pins within a VREF minibank are interconnected internally, and consequently, only one VREF voltage can be used within each VREF minibank. If a bank does not require a VREF signal, the VREF pins of that bank are available as user I/Os.
7. The first VREF minibank includes all I/Os starting from one end of the bank to the first power triple and eight more I/Os after the power triple. Therefore, the first VREF minibank may contain (0 + 8), (2 + 8), (4 + 8), (6 + 8), or (8 + 8) I/Os.

The second VREF minibank is adjacent to the first VREF minibank and contains eight I/Os, a power triple, and eight more I/Os after the triple. An analogous rule applies to all other VREF minibanks but the last.

The last VREF minibank is adjacent to the previous one but contains eight I/Os, a power triple, and all I/Os left at the end of the bank. This bank may also contain (8 + 0), (8 + 2), (8 + 4), (8 + 6), or (8 + 8) available I/Os.

Example:

4 I/Os → Triple → 8 I/Os, 8 I/Os → Triple → 8 I/Os, 8 I/Os → Triple → 2 I/Os

That is, minibank A = (4 + 8) I/Os, minibank B = (8 + 8) I/Os, minibank C = (8 + 2) I/Os.

8. Only minibanks that contain input or bidirectional I/Os require a VREF. A VREF is not needed for minibanks composed of output or tristated I/Os.

Assigning the VREF Voltage to a Bank

When importing the PDC file, the VREF voltage can be assigned to the I/O bank. The PDC command is as follows:

```
set_iobank -vref [value]
```

Another method for assigning VREF is by using **MVN > Edit > I/O Bank Settings** (Figure 9-13 on page 266).

DDR Support in Flash-Based Devices

The flash FPGAs listed in Table 10-1 support the DDR feature and the functions described in this document.

Table 10-1 • 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
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 10-1. 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 10-1. 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*.

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

Figure 12-11 • All Silicon Features Selected for Fusion

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

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

Generation of Security Header Programming File Only— Application 2

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

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

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

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

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

Figure 13-2 shows different applications for ISP programming.

1. In a trusted programming environment, you can program the device using the unencrypted (plaintext) programming file.
2. You can program the AES Key in a trusted programming environment and finish the final programming in an untrusted environment using the AES-encrypted (cipher text) programming file.
3. For the remote ISP updating/reprogramming, the AES Key stored in the device enables the encrypted programming bitstream to be transmitted through the untrusted network connection.

Microsemi low power flash devices also provide the unique Microsemi FlashLock feature, which protects the Pass Key and AES Key. Unless the original FlashLock Pass Key is used to unlock the device, security settings cannot be modified. Microsemi does not support read-back of FPGA core-programmed data; however, the FlashROM contents can selectively be read back (or disabled) via the JTAG port based on the security settings established by the Microsemi Designer software. Refer to the "Security in Low Power Flash Devices" section on page 301 for more information.

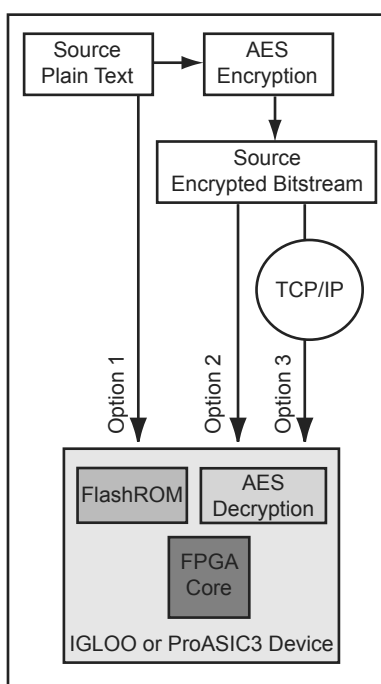
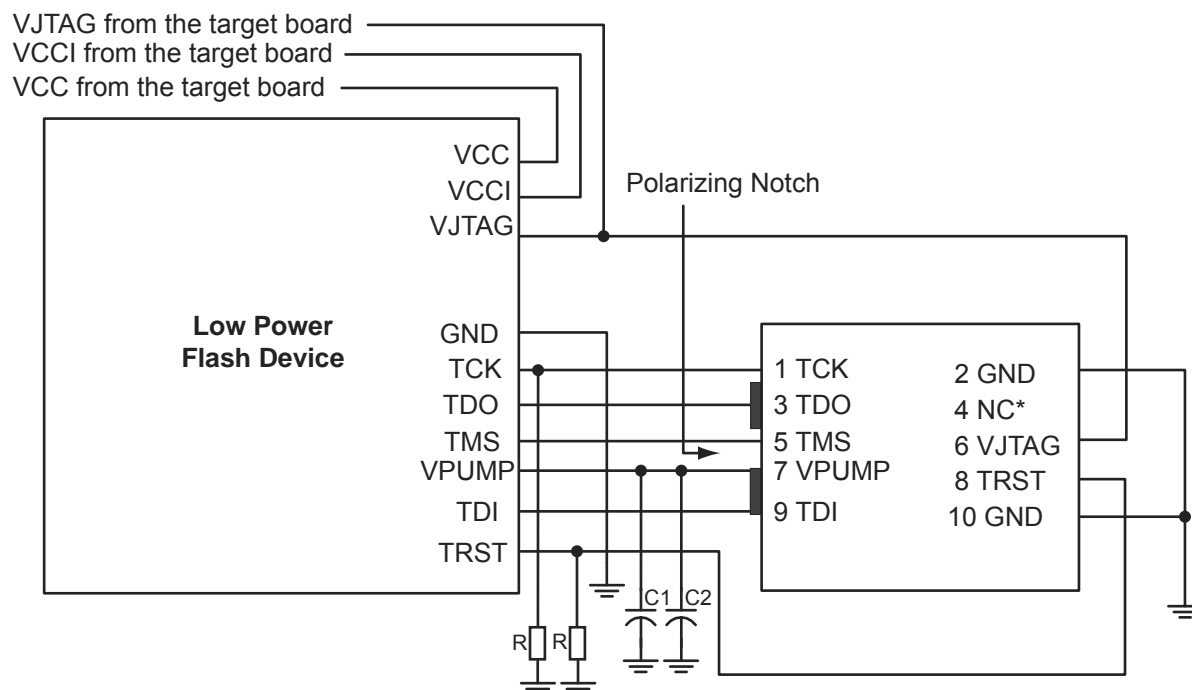


Figure 13-2 • Different ISP Use Models

Board-Level Considerations

A bypass capacitor is required from VPUMP to GND for all low power flash devices during programming. This bypass capacitor protects the devices from voltage spikes that may occur on the VPUMP supplies during the erase and programming cycles. Refer to the "Pin Descriptions and Packaging" chapter of the appropriate device datasheet for specific recommendations. For proper programming, 0.01 μF and 0.33 μF capacitors (both rated at 16 V) are to be connected in parallel across VPUMP and GND, and positioned as close to the FPGA pins as possible. The bypass capacitor must be placed within 2.5 cm of the device pins.



Note: *NC (FlashPro3/3X); Prog_Mode (FlashPro4). Prog_Mode on FlashPro4 is an output signal that goes High during device programming and returns to Low when programming is complete. This signal can be used to drive a system to provide a 1.5 V programming signal to IGLOO nano, ProASIC3L, and RT ProASIC3 devices that can run with 1.2 V core voltage but require 1.5 V for programming. IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only), but IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V.

Figure 13-6 • Board Layout and Programming Header Top View

Troubleshooting Signal Integrity

Symptoms of a Signal Integrity Problem

A signal integrity problem can manifest itself in many ways. The problem may show up as extra or dropped bits during serial communication, changing the meaning of the communication. There is a normal variation of threshold voltage and frequency response between parts even from the same lot. Because of this, the effects of signal integrity may not always affect different devices on the same board in the same way. Sometimes, replacing a device appears to make signal integrity problems go away, but this is just masking the problem. Different parts on identical boards will exhibit the same problem sooner or later. It is important to fix signal integrity problems early. Unless the signal integrity problems are severe enough to completely block all communication between the device and the programmer, they may show up as subtle problems. Some of the FlashPro4/3/3X exit codes that are caused by signal integrity problems are listed below. Signal integrity problems are not the only possible cause of these

14 – Core Voltage Switching Circuit for IGLOO and ProASIC3L In-System Programming

Introduction

The IGLOO[®] and ProASIC[®]3L families offer devices that can be powered by either 1.5 V or, in the case of V2 devices, a core supply voltage anywhere in the range of 1.2 V to 1.5 V, in 50 mV increments.

Since IGLOO and ProASIC3L devices are flash-based, they can be programmed and reprogrammed multiple times in-system using Microsemi FlashPro3. FlashPro3 uses the JTAG standard interface (IEEE 1149.1) and STAPL file (defined in JESD 71 to support programming of programmable devices using IEEE 1149.1) for in-system configuration/programming (IEEE 1532) of a device. Programming can also be executed by other methods, such as an embedded microcontroller that follows the same standards above.

All IGLOO and ProASIC3L devices must be programmed with the VCC core voltage at 1.5 V. Therefore, applications using IGLOO or ProASIC3L devices powered by a 1.2 V supply must switch the core supply to 1.5 V for in-system programming.

The purpose of this document is to describe an easy-to-use and cost-effective solution for switching the core supply voltage from 1.2 V to 1.5 V during in-system programming for IGLOO and ProASIC3L devices.

Programming Algorithm

JTAG Interface

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

IEEE 1532

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

Implementation Overview

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

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

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

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

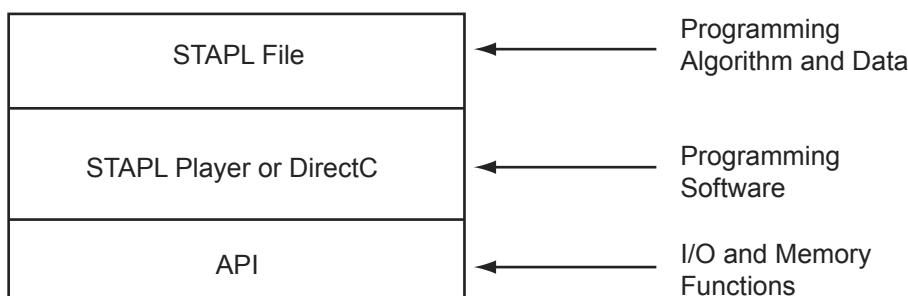


Figure 15-2 • Device Programming Code Relationship

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

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