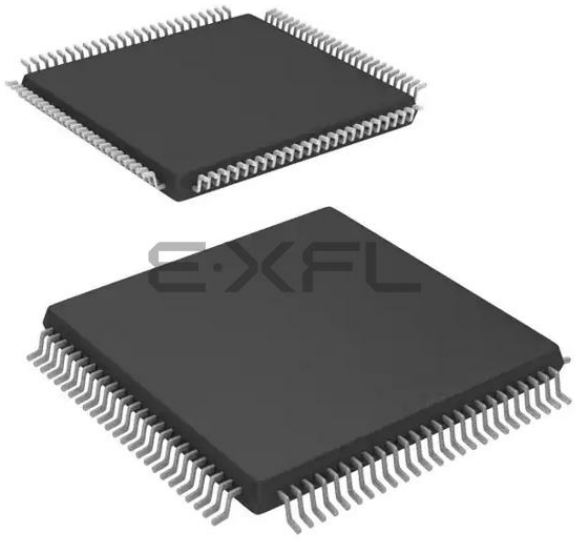


Welcome to E-XFL.COM

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	18432
Number of I/O	71
Number of Gates	60000
Voltage - Supply	1.425V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	100-TQFP
Supplier Device Package	100-VQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pn060-vqg100i

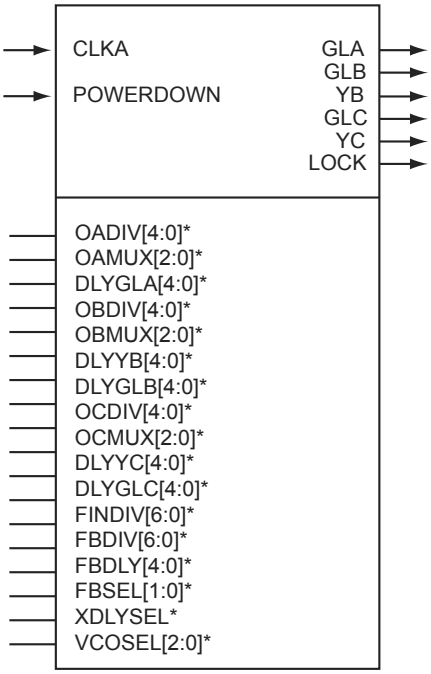


***Figure 2-1* • CCC/PLL Macro**

User Low Static (Idle) Mode

User Low Static (Idle) mode is an advanced feature supported by ProASIC3/E devices to reduce static (idle) power consumption. Entering and exiting this mode is made possible using the ULSICC macro by setting its value to disable/enable the User Low Static (Idle) mode. Under typical operating conditions, characterization results show up to 25% reduction of the static (idle) power consumption. The greatest power savings in terms of percentage are seen in the smaller members of the ProASIC3 family. The active-high control signal for User Low Static (Idle) mode can be generated by internal or external logic. When the device is operating in User Low Static (Idle) mode, FlashROM functionality is temporarily disabled to save power. If FlashROM functionality is needed, the device can exit User Low Static mode temporarily and re-enter the mode once the functionality is no longer needed.

To utilize User Low Static (Idle) mode, simply instantiate the ULSICC macro (Table 2-2 on page 24) in your design, and connect the input port to either an internal logic signal or a device package pin, as illustrated in Figure 2-2 on page 24 or Figure 2-3 on page 25, respectively. The attribute is used so the Synplify® synthesis tool will not optimize the instance with no output port.

This mode can be used to lower standard static (idle) power consumption when the FlashROM feature is not needed. Configuring the device to enter User Low Static (Idle) mode is beneficial when the FPGA enters and exits static mode frequently and lowering power consumption as much as possible is desired. The device is still functional, and data is retained in this state so the device can enter and exit this mode quickly, resulting in reduced total power consumption. The device can also stay in User Low Static mode when the FlashROM feature is not used in the device.

Alternatively, Figure 2-7 shows how a microprocessor can be used with a voltage regulator's shutdown pin to turn the power supplies connected to the device on or off.

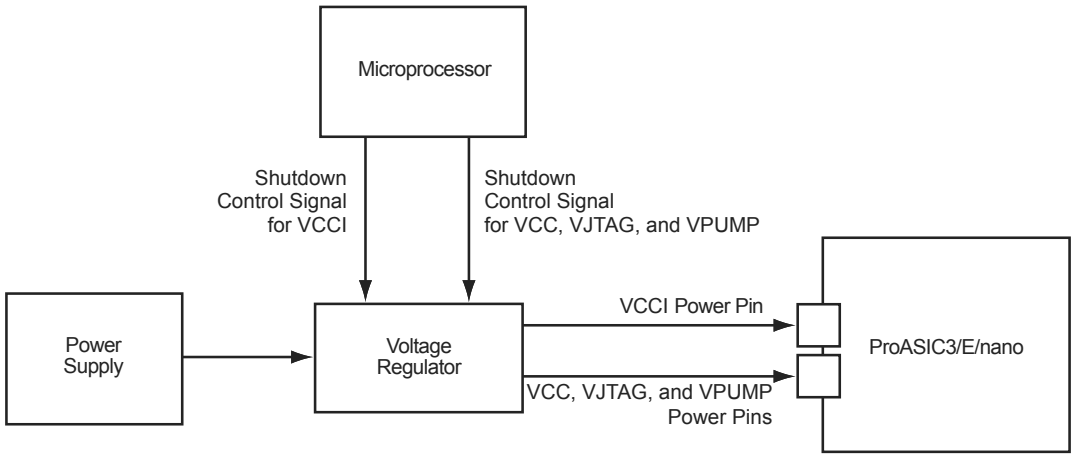


Figure 2-7 • **Controlling Power On/Off State Using Microprocessor and Voltage Regulator**

Though Sleep mode or Shutdown mode can be used to save power, the content of the SRAM and the state of the registers is lost when power is turned off if no other measure is taken. To keep the original contents of the device, a low-cost external serial EEPROM can be used to save and restore the device contents when entering and exiting Sleep mode. In the *Embedded SRAM Initialization Using External Serial EEPROM* application note, detailed information and a reference design are provided to initialize the embedded SRAM using an external serial EEPROM. The user can easily customize the reference design to save and restore the FPGA state when entering and exiting Sleep mode. The microcontroller will need to manage this activity, so before powering down VCC, the data must be read from the FPGA and stored externally. Similarly, after the FPGA is powered up, the microcontroller must allow the FPGA to load the data from external memory and restore its original state.

Conclusion

Microsemi ProASIC3/E and ProASIC3 nano FPGAs inherit low power consumption capability from their nonvolatile and live-at-power-up flash-based technology. Power consumption can be reduced further using the Static (Idle), User Low Static (Idle), Sleep, or Shutdown power modes. All these features result in a low-power, cost-effective, single-chip solution designed specifically for power-sensitive electronics applications.

Related Documents

Application Notes

Embedded SRAM Initialization Using External Serial EEPROM

http://www.microsemi.com/soc/documents/EmbeddedSRAMInit_AN.pdf

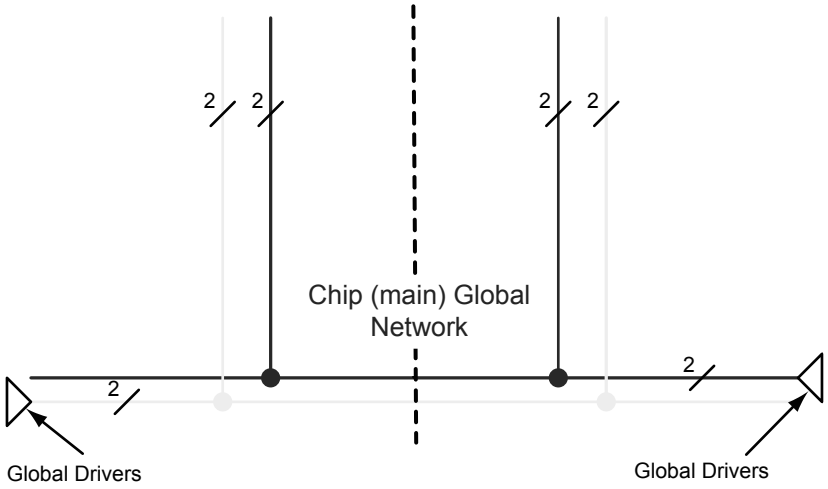


***Figure 3-2* • Simplified VersaNet Global Network (30 k gates and below)**

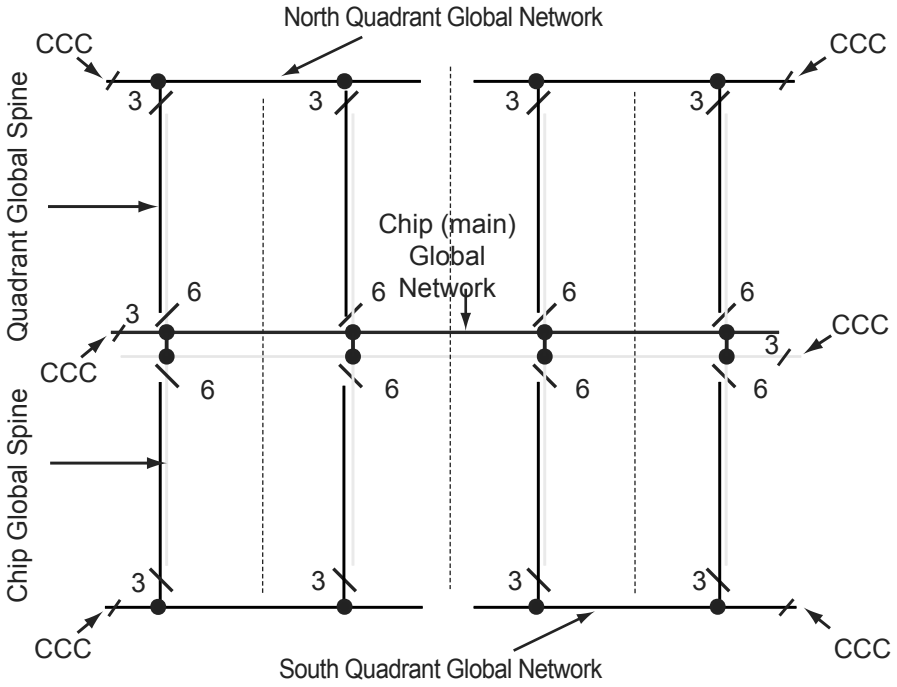


***Figure 3-3* • Simplified VersaNet Global Network (60 k gates and above)**

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
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
	The "Global Architecture" section and "VersaNet Global Network Distribution" section were revised for clarity (SARs 20646, 24779).	31, 33
	The "I/O Banks and Global I/Os" section was moved earlier in the document, renamed to "Chip and Quadrant Global I/Os", and revised for clarity. Figure 3-4 • Global Connections Details, Figure 3-6 • Global Inputs, Table 3-2 • Chip Global Pin Name, and Table 3-3 • Quadrant Global Pin Name are new (SARs 20646, 24779).	35
	The "Clock Aggregation Architecture" section was revised (SARs 20646, 24779).	41
	Figure 3-7 • Chip Global Aggregation was revised (SARs 20646, 24779).	43
	The "Global Macro and Placement Selections" section is new (SARs 20646, 24779).	48
v1.4 (December 2008)	The "Global Architecture" section was updated to include 10 k devices, and to include information about VersaNet global support for IGLOO nano devices.	31
	The Table 3-1 • Flash-Based FPGAs was updated to include IGLOO nano and ProASIC3 nano devices.	32
	The "VersaNet Global Network Distribution" section was updated to include 10 k devices and to note an exception in global lines for nano devices.	33
	Figure 3-2 • Simplified VersaNet Global Network (30 k gates and below) is new.	34
	The "Spine Architecture" section was updated to clarify support for 10 k and nano devices.	41
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include IGLOO nano and ProASIC3 nano devices.	41
	The figure in the CLKBUF_LVDS/LVPECL row of Table 3-8 • Clock Macros was updated to change CLKBIBUF to CLKBUF.	46
v1.3 (October 2008)	A third bullet was added to the beginning of the "Global Architecture" section: In Fusion devices, the west CCC also contains a PLL core. In the two larger devices (AFS600 and AFS1500), the west and east CCCs each contain a PLL.	31
	The "Global Resource Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	32
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include A3PE600/L in the device column.	41
	Table note 1 was revised in Table 3-9 • I/O Standards within CLKBUF to include AFS600 and AFS1500.	47
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 3-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	32

CLKDLY Macro Usage

When a CLKDLY macro is used in a CCC location, the programmable delay element is used to allow the clock delays to go to the global network. In addition, the user can bypass the PLL in a CCC location integrated with a PLL, but use the programmable delay that is associated with the global network by instantiating the CLKDLY macro. The same is true when using programmable delay elements in a CCC location with no PLLs (the user needs to instantiate the CLKDLY macro). There is no difference between the programmable delay elements used for the PLL and the CLKDLY macro. The CCC will be configured to use the programmable delay elements in accordance with the macro instantiated by the user.

As an example, if the PLL is not used in a particular CCC location, the designer is free to specify up to three CLKDLY macros in the CCC, each of which can have its own input frequency and delay adjustment options. If the PLL core is used, assuming output to only one global clock network, the other two global clock networks are free to be used by either connecting directly from the global inputs or connecting from one or two CLKDLY macros for programmable delay.

The programmable delay elements are shown in the block diagram of the PLL block shown in Figure 4-6 on page 71. Note that any CCC locations with no PLL present contain only the programmable delay blocks going to the global networks (labeled "Programmable Delay Type 2"). Refer to the "Clock Delay Adjustment" section on page 86 for a description of the programmable delay types used for the PLL. Also refer to Table 4-14 on page 94 for Programmable Delay Type 1 step delay values, and Table 4-15 on page 94 for Programmable Delay Type 2 step delay values. CCC locations with a PLL present can be configured to utilize only the programmable delay blocks (Programmable Delay Type 2) going to the global networks A, B, and C.

Global network A can be configured to use only the programmable delay element (bypassing the PLL) if the PLL is not used in the design. Figure 4-6 on page 71 shows a block diagram of the PLL, where the programmable delay elements are used for the global networks (Programmable Delay Type 2).

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.

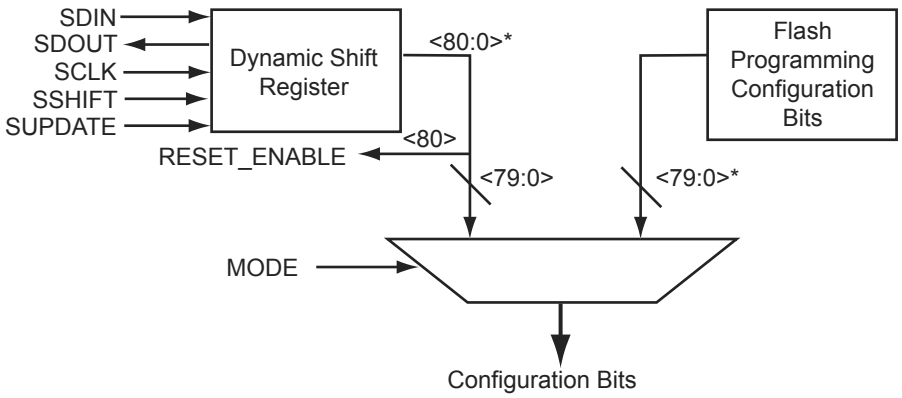
Flash Configuration Bits

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.

Table 4-8 • **Configuration Bit Descriptions for the CCC Blocks (continued)**

Config. Bits	Signal	Name	Description
83	RXCSEL[1]	CLKC input selection	Select the CLKC input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 94).[2]
82	RXBSEL[1]	CLKB input selection	Select the CLKB input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 94).[2]
81	RXASEL[1]	CLKA input selection	Select the CLKA input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 94).[2]
80	RESETEN	Reset Enable	Enables (active high) the synchronization of PLL output dividers after dynamic reconfiguration (SUPDATE). The Reset Enable signal is READ-ONLY.
79	DYNCSEL	Clock Input C Dynamic Select	Configures clock input C to be sent to GLC for dynamic control.[2]
78	DYNBSEL	Clock Input B Dynamic Select	Configures clock input B to be sent to GLB for dynamic control.[2]
77	DYNASEL	Clock Input A Dynamic Select	Configures clock input A for dynamic PLL configuration.[2]
<76:74>	VCOSEL[2:0]	VCO Gear Control	Three-bit VCO Gear Control for four frequency ranges (refer to Table 4-19 on page 95 and Table 4-20 on page 95).
73	STATCSEL	MUX Select on Input C	MUX selection for clock input C[2]
72	STATBSEL	MUX Select on Input B	MUX selection for clock input B[2]
71	STATASEL	MUX Select on Input A	MUX selection for clock input A[2]
<70:66>	DLYC[4:0]	YC Output Delay	Sets the output delay value for YC.
<65:61>	DLYB[4:0]	YB Output Delay	Sets the output delay value for YB.
<60:56>	DLYGLC[4:0]	GLC Output Delay	Sets the output delay value for GLC.
<55:51>	DLYGLB[4:0]	GLB Output Delay	Sets the output delay value for GLB.
<50:46>	DLYGLA[4:0]	Primary Output Delay	Primary GLA output delay
45	XDLYSEL	System Delay Select	When selected, inserts System Delay in the feedback path in Figure 4-20 on page 85.
<44:40>	FBDLY[4:0]	Feedback Delay	Sets the feedback delay value for the feedback element in Figure 4-20 on page 85.
<39:38>	FBSEL[1:0]	Primary Feedback Delay Select	Controls the feedback MUX: no delay, include programmable delay element, or use external feedback.
<37:35>	OCMUX[2:0]	Secondary 2 Output Select	Selects from the VCO's four phase outputs for GLC/YC.
<34:32>	OBMUX[2:0]	Secondary 1 Output Select	Selects from the VCO's four phase outputs for GLB/YB.

Notes:

1. *The <88:81> configuration bits are only for the Fusion dynamic CCC.*
2. *This value depends on the input clock source, so Layout must complete before these bits can be set. After completing Layout in Designer, generate the "CCC_Configuration" report by choosing* ***Tools*** *>* ***Report*** *>* ***CCC_Configuration****. The report contains the appropriate settings for these bits.*

Figure 4-31 • **Static Timing Analysis Using SmartTime**

Place-and-Route Stage Considerations

Several considerations must be noted to properly place the CCC macros for layout.

For CCCs with clock inputs configured with the Hardwired I/O–Driven option:

- PLL macros must have the clock input pad coming from one of the GmA* locations.
- CLKDLY macros must have the clock input pad coming from one of the Global I/Os.

If a PLL with a Hardwired I/O input is used at a CCC location and a Hardwired I/O–Driven CLKDLY macro is used at the same CCC location, the clock input of the CLKDLY macro must be chosen from one of the GmB* or GmC* pin locations. If the PLL is not used or is an External I/O–Driven or Core Logic–Driven PLL, the clock input of the CLKDLY macro can be sourced from the GmA*, GmB*, or GmC* pin locations.

For CCCs with clock inputs configured with the External I/O–Driven option, the clock input pad can be assigned to any regular I/O location (IO******** pins). Note that since global I/O pins can also be used as regular I/Os, regardless of CCC function (CLKDLY or PLL), clock inputs can also be placed in any of these I/O locations.

By default, the Designer layout engine will place global nets in the design at one of the six chip globals. When the number of globals in the design is greater than six, the Designer layout engine will automatically assign additional globals to the quadrant global networks of the low power flash devices. If the user wishes to decide which global signals should be assigned to chip globals (six available) and which to the quadrant globals (three per quadrant for a total of 12 available), the assignment can be achieved with PinEditor, ChipPlanner, or by importing a placement constraint file. Layout will fail if the

Date	Changes	Page
v1.2 (June 2008)	The following changes were made to the family descriptions in Figure 4-1 • Overview of the CCCs Offered in Fusion, IGLOO, and ProASIC3: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	61
v1.1 (March 2008)	Table 4-1 • Flash-Based FPGAs and the associated text were updated to include the IGLOO PLUS family. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	63
	The "Global Input Selections" section was updated to include 15 k gate devices as supported I/O types for globals, for CCC only.	71
	Table 4-5 • Number of CCCs by Device Size and Package was revised to include ProASIC3L, IGLOO PLUS, A3P015, AGL015, AGLP030, AGLP060, and AGLP125.	78
	The "IGLOO and ProASIC3 CCC Locations" section was revised to include 15 k gate devices in the exception statements, as they do not contain PLLs.	81
v1.0 (January 2008)	Information about unlocking the PLL was removed from the "Dynamic PLL Configuration" section.	87
	In the "Dynamic PLL Configuration" section, information was added about running Layout and determining the exact setting of the ports.	100
	In Table 4-8 • Configuration Bit Descriptions for the CCC Blocks, the following bits were updated to delete "transport to the user" and reference the footnote at the bottom of the table: 79 to 71.	90

Simulation of FlashROM Design

The MEM file has 128 rows of 8 bits, each representing the contents of the FlashROM used for simulation. For example, the first row represents page 0, byte 0; the next row is page 0, byte 1; and so the pattern continues. Note that the three MSBs of the address define the page number, and the four LSBs define the byte number. So, if you send address 0000100 to FlashROM, this corresponds to the page 0 and byte 4 location, which is the fifth row in the MEM file. SmartGen defaults to 0s for any unspecified location of the FlashROM. Besides using the MEM file generated by SmartGen, you can create a binary file with 128 rows of 8 bits each and use this as a MEM file. Microsemi recommends that you use different file names if you plan to generate multiple MEM files. During simulation, Libero SoC passes the MEM file used as the generic file in the netlist, along with the design files and testbench. If you want to use different MEM files during simulation, you need to modify the generic file reference in the netlist.

```
……..
UFROM0: UFROM
--generic map(MEMORYFILE => "F:\Appsnotes\FROM\test_designs\testa\smartgen\FROM_a.mem")
--generic map(MEMORYFILE => "F:\Appsnotes\FROM\test_designs\testa\smartgen\FROM_b.mem")
……..
```

The VITAL and Verilog simulation models accept the generics passed by the netlist, read the MEM file, and perform simulation with the data in the file.

Programming File Generation for FlashROM Design

FlashPoint is the programming software used to generate the programming files for flash devices. Depending on the applications, you can use the FlashPoint software to generate a STAPL file with different FlashROM contents. In each case, optional AES decryption is available. To generate a STAPL file that contains the same FPGA core content and different FlashROM contents, the FlashPoint software needs an Array Map file for the core and UFC file(s) for the FlashROM. This final STAPL file represents the combination of the logic of the FPGA core and FlashROM content.

FlashPoint generates the STAPL files you can use to program the desired FlashROM page and/or FPGA core of the FPGA device contents. FlashPoint supports the encryption of the FlashROM content and/or FPGA Array configuration data. In the case of using the FlashROM for device serialization, a sequence of unique FlashROM contents will be generated. When generating a programming file with multiple unique FlashROM contents, you can specify in FlashPoint whether to include all FlashROM content in a single STAPL file or generate a different STAPL file for each FlashROM (Figure 5-11). The programming software (FlashPro) handles the single STAPL file that contains the FlashROM content from multiple devices. It enables you to program the FlashROM content into a series of devices sequentially (Figure 5-11). See the *FlashPro User's Guide* for information on serial programming.

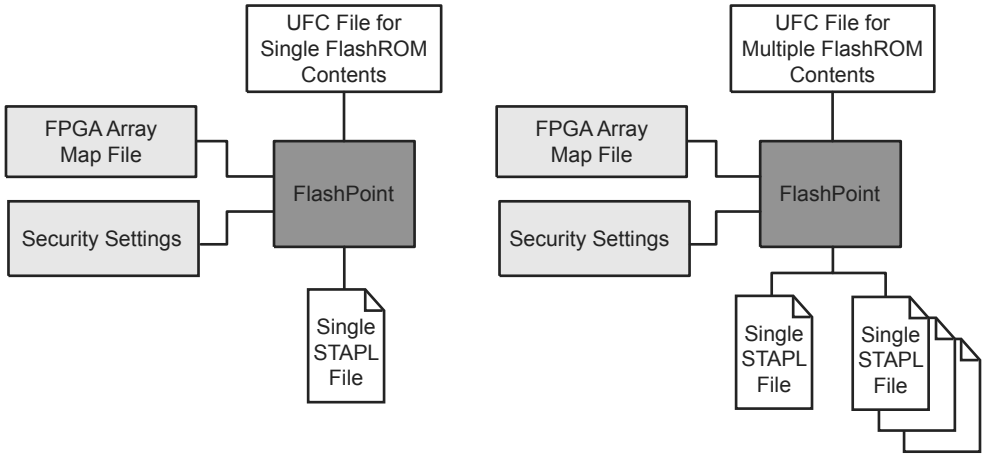


***Figure 5-11* • Single or Multiple Programming File Generation**

Power-Up Behavior

Low power flash devices are power-up/-down friendly; i.e., no particular sequencing is required for power-up and power-down. This eliminates extra board components for power-up sequencing, such as a power-up sequencer.

During power-up, all I/Os are tristated, irrespective of I/O macro type (input buffers, output buffers, I/O buffers with weak pull-ups or weak pull-downs, etc.). Once I/Os become activated, they are set to the user-selected I/O macros. Refer to the "Power-Up/-Down Behavior of Low Power Flash Devices" section on page 307 for details.

Drive Strength

Low power flash devices have up to four programmable output drive strengths. The user can select the drive strength of a particular output in the I/O Attribute Editor or can instantiate a specialized I/O macro, such as OUTBUF_S_8 (slew = low, out_drive = 8 mA).

The maximum available drive strength is 8 mA per I/O. Though no I/O should be forced to source or sink more than 8 mA indefinitely, I/Os may handle a higher amount of current (refer to the device IBIS model for maximum source/sink current) during signal transition (AC current). Every device package has its own power dissipation limit; hence, power calculation must be performed accurately to determine how much current can be tolerated per I/O within that limit.

I/O Interfacing

Low power flash devices are 5 V–input– and 5 V–output–tolerant without adding any extra circuitry. Along with other low-voltage I/O macros, this 5 V tolerance makes these devices suitable for many types of board component interfacing.

Table 7-17 shows some high-level interfacing examples using low power flash devices.

***Table 7-17* • nano High-Level Interface**

Interface	Clock		I/O			
	Type	Frequency	Type	Signals In	Signals Out	Data I/O
GM	Src Sync	125 MHz	LVTTL	8	8	125 Mbps
TBI	Src Sync	125 MHz	LVTTL	10	10	125 Mbps

Conclusion

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

If the assignment is not successful, an error message appears in the Output window.

To undo the I/O bank assignments, choose **Undo** from the **Edit** menu. Undo removes the I/O technologies assigned by the IOBA. It does not remove the I/O technologies previously assigned.

To redo the changes undone by the Undo command, choose **Redo** from the **Edit** menu.

To clear I/O bank assignments made before using the Undo command, manually unassign or reassign I/O technologies to banks. To do so, choose **I/O Bank Settings** from the **Edit** menu to display the I/O Bank Settings dialog box.

Conclusion

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

Related Documents

User's Guides

Libero SoC User's Guide

http://www.microsemi.com/soc/documents/libero_ug.pdf

IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide

http://www.microsemi.com/soc/documents/pa3_libguide_ug.pdf

SmartGen Core Reference Guide

http://www.microsemi.com/soc/documents/genguide_ug.pdf

DDR Support in Flash-Based Devices

The flash FPGAs listed in Table 9-1 support the DDR feature and the functions described in this document.

***Table 9-1* • Flash-Based FPGAs**

Series	Family*	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device

*Note: *The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.*

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 9-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 9-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

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

10 – Programming Flash Devices

Introduction

This document provides an overview of the various programming options available for the Microsemi flash families. The electronic version of this document includes active links to all programming resources, which are available at http://www.microsemi.com/soc/products/hardware/default.aspx. For Microsemi antifuse devices, refer to the *Programming Antifuse Devices* document.

Summary of Programming Support

FlashPro4 and FlashPro3 are high-performance in-system programming (ISP) tools targeted at the latest generation of low power flash devices offered by the SmartFusion,® Fusion, IGLOO,® and ProASIC®3 families, including ARM-enabled devices. FlashPro4 and FlashPro3 offer extremely high performance through the use of USB 2.0, are high-speed compliant for full use of the 480 Mbps bandwidth, and can program ProASIC3 devices in under 30 seconds. Powered exclusively via USB, FlashPro4 and FlashPro3 provide a VPUMP voltage of 3.3 V for programming these devices.

FlashPro4 replaced FlashPro3 in 2010. FlashPro4 supports SmartFusion, Fusion, ProASIC3,and IGLOO devices as well as future generation flash devices. FlashPro4 also adds 1.2 V programming for IGLOO nano V2 devices. FlashPro4 is compatible with FlashPro3; however it adds a programming mode (PROG_MODE) signal to the previously unused pin 4 of the JTAG connector. The PROG_MODE goes high during programming and can be used to turn on a 1.5 V external supply for those devices that require 1.5 V for programming. If both FlashPro3 and FlashPro4 programmers are used for programming the same boards, pin 4 of the JTAG connector must not be connected to anything on the board because FlashPro4 uses pin 4 for PROG_MODE.

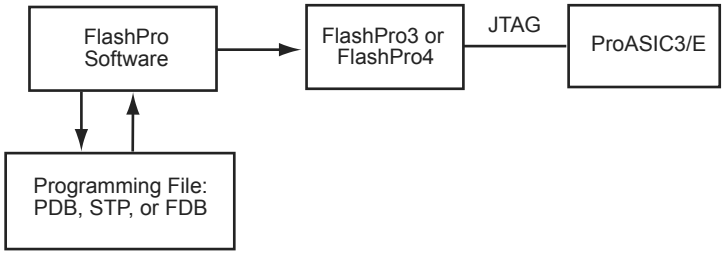


***Figure 10-1* • FlashPro Programming Setup**

13 – 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.

3. VCC switches from 1.5 V to 1.2 V when TRST is LOW.

***Figure 13-4* • TRST Toggled LOW**

In Figure 13-4, the TRST signal and the VCC core voltage signal are labeled. As TRST is pulled to ground, the core voltage is observed to switch from 1.5 V to 1.2 V. The observed fall time is approximately 2 ms.

DirectC

The above analysis is based on FlashPro3, but there are other solutions to ISP, such as DirectC. DirectC is a microprocessor program that can be run in-system to program Microsemi flash devices. For FlashPro3, TRST is the most convenient control signal to use for the recommended circuit. However, for DirectC, users may use any signal to control the FET. For example, the DirectC code can be edited so that a separate non-JTAG signal can be asserted from the microcontroller that signals the board that it is about to start programming the device. After asserting the N-Channel Digital FET control signal, the programming algorithm must allow sufficient time for the supply to rise to 1.5 V before initiating DirectC programming. As seen in Figure 13-3 on page 279, 50 ms is adequate time. Depending on the size of the PCB and the capacitance on the VCC supply, results may vary from system to system. Microsemi recommends using a conservative value for the wait time to make sure that the VCC core voltage is at the right level.

Conclusion

For applications using IGLOO and ProASIC3L low power FPGAs and taking advantage of the low core voltage power supplies with less than 1.5 V operation, there must be a way for the core voltage to switch from 1.2 V (or other voltage) to 1.5 V, which is required during in-system programming. The circuit explained in this document illustrates one simple, cost-effective way of handling this requirement. A JTAG signal from the FlashPro3 programmer allows the circuit to sense when programming is in progress, enabling it to switch to the correct core voltage.

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 14-2).

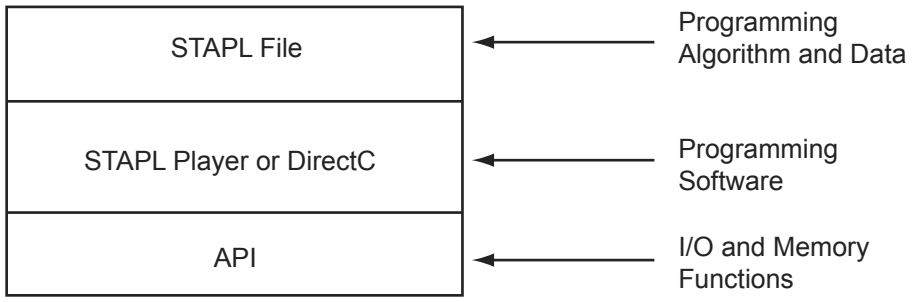


***Figure 14-2* • Device Programming Code Relationship**

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 14-3 on page 286). 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.

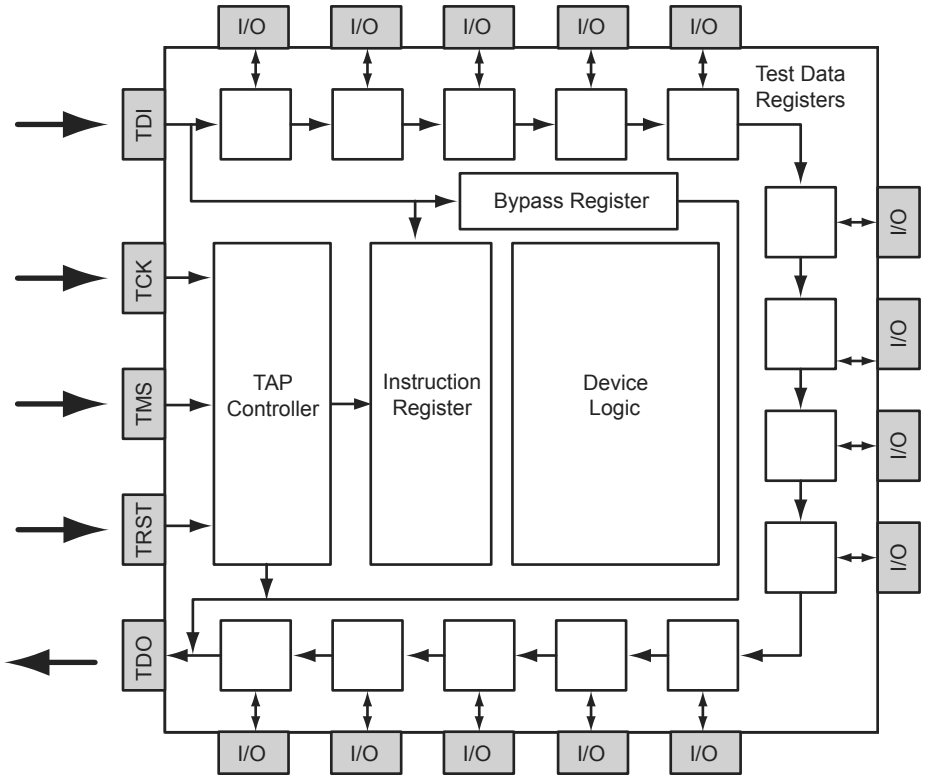


***Figure 15-2* • Boundary Scan Chain**

Board-Level Recommendations

Table 15-3 gives pull-down recommendations for the TRST and TCK pins.

***Table 15-3* • TRST and TCK Pull-Down Recommendations**

VJTAG	Tie-Off Resistance*
VJTAG at 3.3 V	200 Ω to 1 kΩ
VJTAG at 2.5 V	200 Ω to 1 kΩ
VJTAG at 1.8 V	500 Ω to 1 kΩ
VJTAG at 1.5 V	500 Ω to 1 kΩ
VJTAG at 1.2 V	TBD

Note: Equivalent parallel resistance if more than one device is on JTAG chain (Figure 15-3)