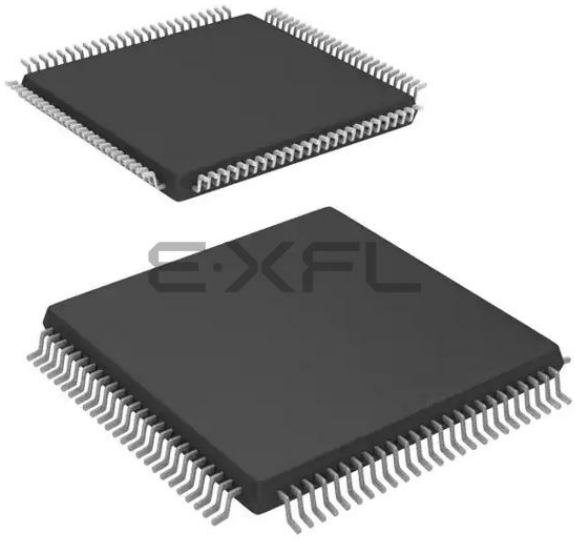


Welcome to E-XFL.COM

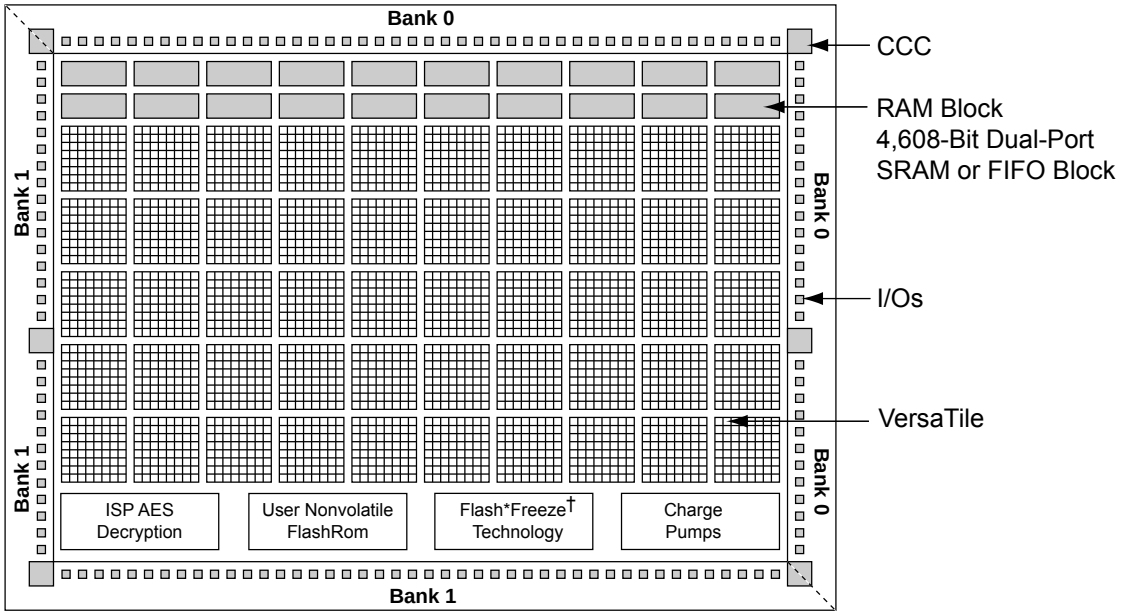
Understanding Embedded - FPGAs (Field Programmable Gate Array)

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

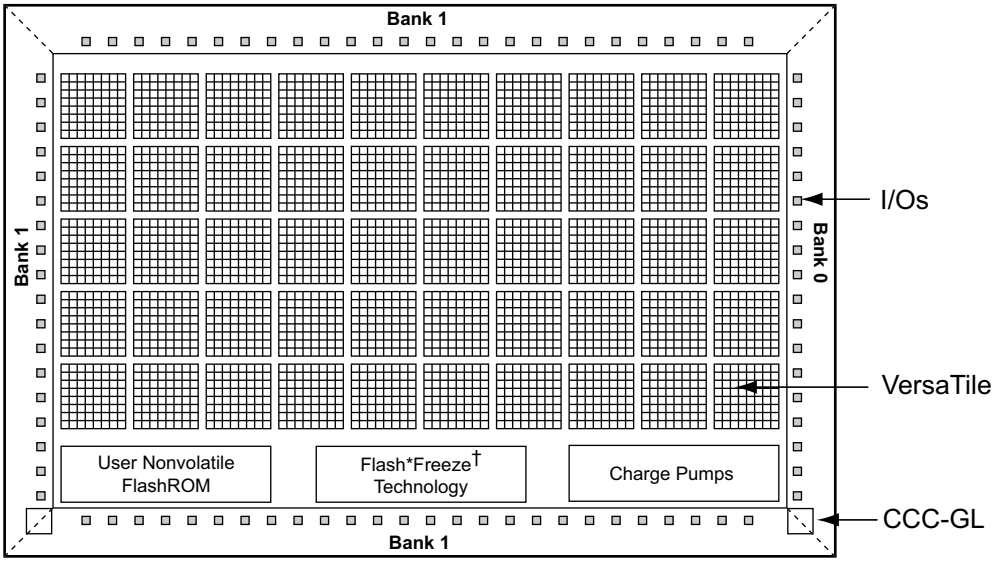
The versatility of Embedded - FPGAs makes them indispensable in numerous fields. In telecommunications.

Details	
Product Status	Obsolete
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	-
Number of I/O	77
Number of Gates	30000
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/a3pn030-z2vqg100i



*Note: † Flash*Freeze mode is supported on IGLOO devices.*

Figure 1-3 • **IGLOO Device Architecture Overview with Two I/O Banks with RAM and PLL (60 k and 125 k gate densities)**



*Note: † Flash*Freeze mode is supported on IGLOO devices.*

Figure 1-4 • **IGLOO Device Architecture Overview with Three I/O Banks (AGLN015, AGLN020, A3PN015, and A3PN020)**

***Table 3-2* • Chip Global Pin Name**

I/O Type	Beginning of I/O Name	Notes
Single-Ended	GFAO/IOuxwByVz GFA1/IOuxwByVz GFA2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GFBO/IOuxwByVz GFB1/IOuxwByVz GFB2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GFC0/IOuxwByVz GFC1/IOuxwByVz GFC2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GCAO/IOuxwByVz GCA1/IOuxwByVz GCA2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GCBO/IOuxwByVz GCB1/IOuxwByVz GCB2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
	GCC0/IOuxwByVz GCC1/IOuxwByVz GCC2/IOuxwByVz	Only one of the I/Os can be directly connected to a chip global at a time.
Differential I/O Pairs	GFAO/IOuxwByVz GFA1/IOuxwByVz	The output of the different pair will drive the chip global.
	GFBO/IOuxwByVz GFB1/IOuxwByVz	The output of the different pair will drive the chip global.
	GFCO/IOuxwByVz GFC1/IOuxwByVz	The output of the different pair will drive the chip global.
	GCAO/IOuxwByVz GCA1/IOuxwByVz	The output of the different pair will drive the chip global.
	GCBO/IOuxwByVz GCB1/IOuxwByVz	The output of the different pair will drive the chip global.
	GCCO/IOuxwByVz GCC1/IOuxwByVz	The output of the different pair will drive the chip global.

Note: Only one of the I/Os can be directly connected to a quadrant at a time.

During Layout, Designer will assign two of the signals to quadrant global locations.

Step 3 (optional)

You can also assign the QCLK1_c and QCLK2_c nets to quadrant regions using the following PDC commands:

```
assign_local_clock –net QCLK1_c  –type quadrant UL
assign_local_clock –net QCLK2_c  –type quadrant LL
```

Step 4

Import this PDC with the netlist and run Compile again. You will see the following in the Compile report:

```
The following nets have been assigned to a global resource:
Fanout  Type          Name
--------------------------
1536    INT_NET       Net   : EN_ALL_c
                      Driver: EN_ALL_pad_CLKINT
                      Source: AUTO PROMOTED
1536    SET/RESET_NET Net   : ACLR_c
                      Driver: ACLR_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : QCLK3_c
                      Driver: QCLK3_pad_CLKINT
                      Source: AUTO PROMOTED
256     CLK_NET       Net   : $1N14
                      Driver: $1I5/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N12
                      Driver: $1I6/Core
                      Source: ESSENTIAL
256     CLK_NET       Net   : $1N10
                      Driver: $1I6/Core
                      Source: ESSENTIAL
The following nets have been assigned to a quadrant clock resource using PDC:
Fanout  Type          Name
--------------------------
256     CLK_NET       Net   : QCLK1_c
                      Driver: QCLK1_pad_CLKINT
                      Region: quadrant_UL
256     CLK_NET       Net   : QCLK2_c
                      Driver: QCLK2_pad_CLKINT
                      Region: quadrant_LL
```

Step 5

Run Layout.

Global Management in PLL Design

This section describes the legal global network connections to PLLs in the low power flash devices. For detailed information on using PLLs, refer to "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 61. Microsemi recommends that you use the dedicated global pins to directly drive the reference clock input of the associated PLL for reduced propagation delays and clock distortion. However, low power flash devices offer the flexibility to connect other signals to reference clock inputs. Each PLL is associated with three global networks (Figure 3-5 on page 36). There are some limitations, such as when trying to use the global and PLL at the same time:

- If you use a PLL with only primary output, you can still use the remaining two free global networks.
- If you use three globals associated with a PLL location, you cannot use the PLL on that location.
- If the YB or YC output is used standalone, it will occupy one global, even though this signal does not go to the global network.

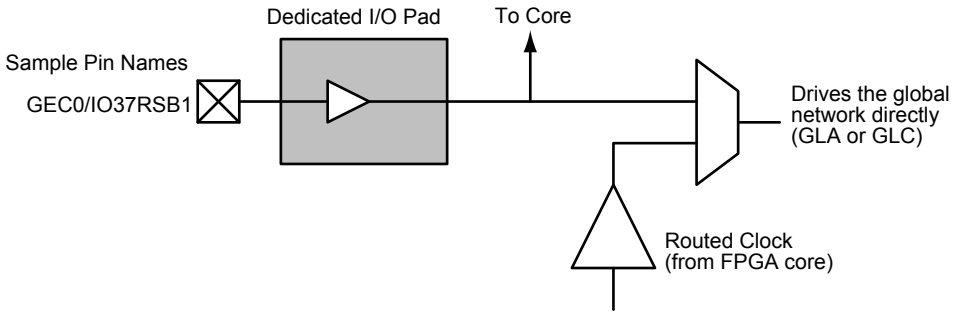
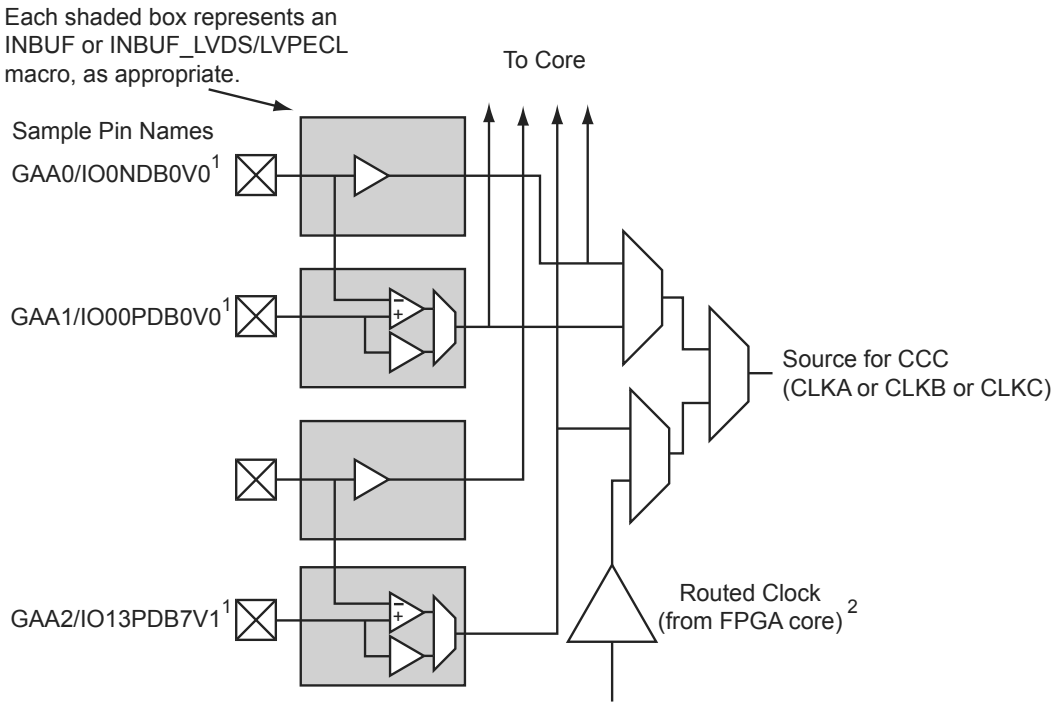


***Figure 4-7* • Clock Input Sources (30 k gates devices and below)**



GAA[0:2]: GA represents global in the northwest corner of the device. A[0:2]: designates specific A clock source.

Notes:

1. *Represents the global input pins. Globals have direct access to the clock conditioning block and are not routed via the FPGA fabric. Refer to the "User I/O Naming Conventions in I/O Structures" chapter of the appropriate device user's guide.*
2. *Instantiate the routed clock source input as follows:*
   a) *Connect the output of a logic element to the clock input of a PLL, CLKDLY, or CLKINT macro.*
   b) *Do not place a clock source I/O (INBUF or INBUF_LVPECL/LVDS/B-LVDS/M-LVDS/DDR) in a relevant global pin location.*
3. *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

***Figure 4-8* • Clock Input Sources Including CLKBUF, CLKBUF_LVDS/LVPECL, and CLKINT (60 k gates devices and above)**

Software Configuration

SmartGen automatically generates the desired CCC functional block by configuring the control bits, and allows the user to select two CCC modes: Static PLL and Delayed Clock (CLKDLY).

Static PLL Configuration

The newly implemented Visual PLL Configuration Wizard feature provides the user a quick and easy way to configure the PLL with the desired settings (Figure 4-23). The user can invoke SmartGen to set the parameters and generate the netlist file with the appropriate flash configuration bits set for the CCCs. As mentioned in "PLL Macro Block Diagram" on page 69, the input reference clock CLKA can be configured to be driven by Hardwired I/O, External I/O, or Core Logic. The user enters the desired settings for all the parameters (output frequency, output selection, output phase adjustment, clock delay, feedback delay, and system delay). Notice that the actual values (divider values, output frequency, delay values, and phase) are shown to aid the user in reaching the desired design frequency in real time. These values are typical-case data. Best- and worst-case data can be observed through static timing analysis in SmartTime within Designer.

For dynamic configuration, the CCC parameters are defined using either the external JTAG port or an internally defined serial interface via the built-in dynamic shift register. This feature provides the ability to compensate for changes in the external environment.

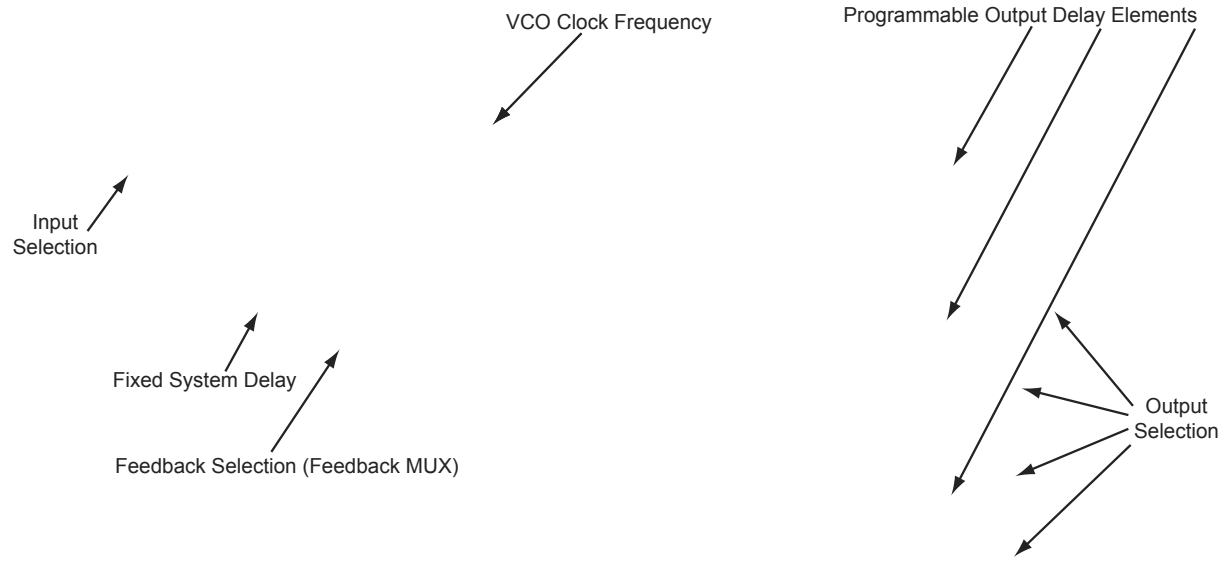


Figure 4-23 • **Visual PLL Configuration Wizard**

Dynamic PLL Configuration

To generate a dynamically reconfigurable CCC, the user should select **Dynamic CCC** in the configuration section of the SmartGen GUI (Figure 4-26). This will generate both the CCC core and the configuration shift register / control bit MUX.

***Figure 4-26* • SmartGen GUI**

Even if dynamic configuration is selected in SmartGen, the user must still specify the static configuration data for the CCC (Figure 4-27). The specified static configuration is used whenever the MODE signal is set to LOW and the CCC is required to function in the static mode. The static configuration data can be used as the default behavior of the CCC where required.

***Figure 4-27* • Dynamic CCC Configuration in SmartGen**

```
  DYNCCC Core(.CLKA(CLKA), .EXTFB(GND), .POWERDOWN(POWERDOWN), .GLA(GLA), .LOCK(LOCK),
    .CLKB(CLKB), .GLB(GLB), .YB(), .CLKC(CLKC), .GLC(GLC), .YC(), .SDIN(SDIN),
    .SCLK(SCLK), .SSHIFT(SSHIFT), .SUPDATE(SUPDATE), .MODE(MODE), .SDOUT(SDOUT),
    .OADIV0(GND), .OADIV1(GND), .OADIV2(VCC), .OADIV3(GND), .OADIV4(GND), .OAMUX0(GND),
    .OAMUX1(GND), .OAMUX2(VCC), .DLYGLA0(GND), .DLYGLA1(GND), .DLYGLA2(GND),
    .DLYGLA3(GND), .DLYGLA4(GND), .OBDIV0(GND), .OBDIV1(GND), .OBDIV2(GND),
    .OBDIV3(GND), .OBDIV4(GND), .OBMUX0(GND), .OBMUX1(GND), .OBMUX2(GND), .DLYYB0(GND),
    .DLYYB1(GND), .DLYYB2(GND), .DLYYB3(GND), .DLYYB4(GND), .DLYGLB0(GND),
    .DLYGLB1(GND), .DLYGLB2(GND), .DLYGLB3(GND), .DLYGLB4(GND), .OCDIV0(GND),
    .OCDIV1(GND), .OCDIV2(GND), .OCDIV3(GND), .OCDIV4(GND), .OCMUX0(GND), .OCMUX1(GND),
    .OCMUX2(GND), .DLYYC0(GND), .DLYYC1(GND), .DLYYC2(GND), .DLYYC3(GND), .DLYYC4(GND),
    .DLYGLC0(GND), .DLYGLC1(GND), .DLYGLC2(GND), .DLYGLC3(GND), .DLYGLC4(GND),
    .FINDIV0(VCC), .FINDIV1(GND), .FINDIV2(VCC), .FINDIV3(GND), .FINDIV4(GND),
    .FINDIV5(GND), .FINDIV6(GND), .FBDIV0(GND), .FBDIV1(GND), .FBDIV2(GND),
    .FBDIV3(GND), .FBDIV4(GND), .FBDIV5(VCC), .FBDIV6(GND), .FBDLY0(GND), .FBDLY1(GND),
    .FBDLY2(GND), .FBDLY3(GND), .FBDLY4(GND), .FBSEL0(VCC), .FBSEL1(GND),
    .XDLYSEL(GND), .VCOSEL0(GND), .VCOSEL1(GND), .VCOSEL2(VCC));
  defparam Core.VCOFREQUENCY = 165.000;

endmodule
```

Delayed Clock Configuration

The CLKDLY macro can be generated with the desired delay and input clock source (Hardwired I/O, External I/O, or Core Logic), as in Figure 4-28.

Figure 4-28 • **Delayed Clock Configuration Dialog Box**

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

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

Name                            : delay_macro
Family                          : ProASIC3
Output Format                   : Verilog
Type                            : Delayed Clock
Delay Index                     : 2
CLKA Source                     : Hardwired I/O

Total Clock Delay = 0.935 ns.

The resultant CLKDLY macro Verilog netlist is as follows:

module delay_macro(GL,CLK);

output GL;
input  CLK;
```

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

Recommended Board-Level Considerations

The power to the PLL core is supplied by VCCPLA/B/C/D/E/F (VCCPLx), and the associated ground connections are supplied by VCOMPLA/B/C/D/E/F (VCOMPLx). When the PLLs are not used, the Designer place-and-route tool automatically disables the unused PLLs to lower power consumption. The user should tie unused VCCPLx and VCOMPLx pins to ground. Optionally, the PLL can be turned on/off during normal device operation via the POWERDOWN port (see Table 4-3 on page 68).

PLL Power Supply Decoupling Scheme

The PLL core is designed to tolerate noise levels on the PLL power supply as specified in the datasheets. When operated within the noise limits, the PLL will meet the output peak-to-peak jitter specifications specified in the datasheets. User applications should always ensure the PLL power supply is powered from a noise-free or low-noise power source.

However, in situations where the PLL power supply noise level is higher than the tolerable limits, various decoupling schemes can be designed to suppress noise to the PLL power supply. An example is provided in Figure 4-38. The VCCPLx and VCOMPLx pins correspond to the PLL analog power supply and ground.

Microsemi strongly recommends that two ceramic capacitors (10 nF in parallel with 100 nF) be placed close to the power pins (less than 1 inch away). A third generic 10 µF electrolytic capacitor is recommended for low-frequency noise and should be placed farther away due to its large physical size. Microsemi recommends that a 6.8 µH inductor be placed between the supply source and the capacitors to filter out any low-/medium- and high-frequency noise. In addition, the PCB layers should be controlled so the VCCPLx and VCOMPLx planes have the minimum separation possible, thus generating a good-quality RF capacitor.

For more recommendations, refer to the *Board-Level Considerations* application note.

Recommended 100 nF capacitor:

- Producer BC Components, type X7R, 100 nF, 16 V
- BC Components part number: 0603B104K160BT
- Digi-Key part number: BC1254CT-ND
- Digi-Key part number: BC1254TR-ND

Recommended 10 nF capacitor:

- Surface-mount ceramic capacitor
- Producer BC Components, type X7R, 10 nF, 50 V
- BC Components part number: 0603B103K500BT
- Digi-Key part number: BC1252CT-ND
- Digi-Key part number: BC1252TR-ND

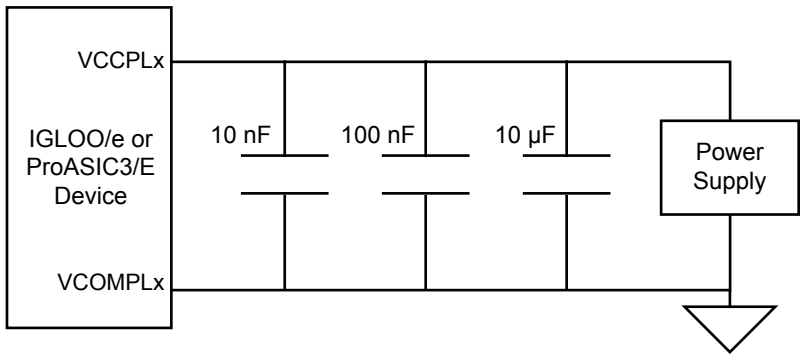


***Figure 4-38* • Decoupling Scheme for One PLL (should be replicated for each PLL used)**

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.

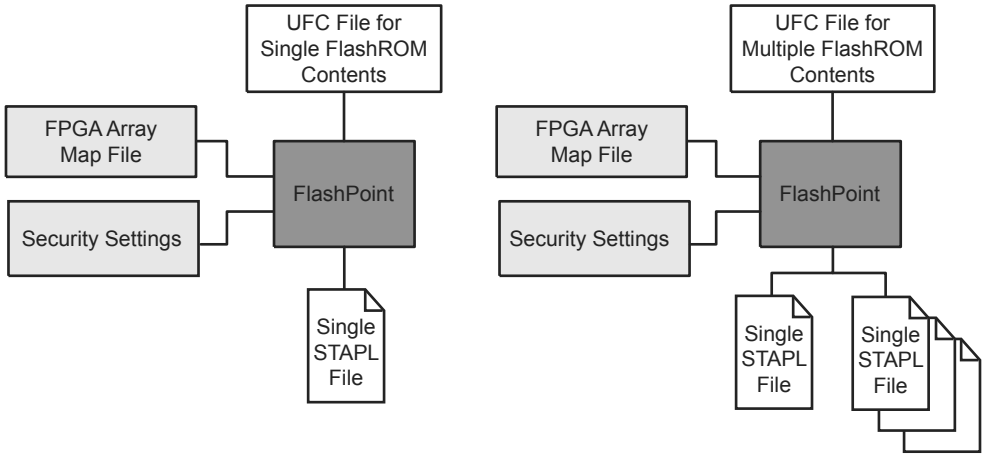


***Figure 5-11* • Single or Multiple Programming File Generation**

6 – SRAM and FIFO Memories in Microsemi's Low Power Flash Devices

Introduction

As design complexity grows, greater demands are placed upon an FPGA's embedded memory. Fusion, IGLOO, and ProASIC3 devices provide the flexibility of true dual-port and two-port SRAM blocks. The embedded memory, along with built-in, dedicated FIFO control logic, can be used to create cascading RAM blocks and FIFOs without using additional logic gates.

IGLOO, IGLOO PLUS, and ProASIC3L FPGAs contain an additional feature that allows the device to be put in a low power mode called Flash*Freeze. In this mode, the core draws minimal power (on the order of 2 to 127 µW) and still retains values on the embedded SRAM/FIFO and registers. Flash*Freeze technology allows the user to switch to Active mode on demand, thus simplifying power management and the use of SRAM/FIFOs.

Device Architecture

The low power flash devices feature up to 504 kbits of RAM in 4,608-bit blocks (Figure 6-1 on page 132 and Figure 6-2 on page 133). The total embedded SRAM for each device can be found in the datasheets. These memory blocks are arranged along the top and bottom of the device to allow better access from the core and I/O (in some devices, they are only available on the north side of the device). Every RAM block has a flexible, hardwired, embedded FIFO controller, enabling the user to implement efficient FIFOs without sacrificing user gates.

In the IGLOO and ProASIC3 families of devices, the following memories are supported:

- 30 k gate devices and smaller do not support SRAM and FIFO.
- 60 k and 125 k gate devices support memories on the north side of the device only.
- 250 k devices and larger support memories on the north and south sides of the device.

In Fusion devices, the following memories are supported:

- AFS090 and AFS250 support memories on the north side of the device only.
- AFS600 and AFS1500 support memories on the north and south sides of the device.

SRAM/FIFO Support in Flash-Based Devices

The flash FPGAs listed in Table 6-1 support SRAM and FIFO blocks and the functions described in this document.

***Table 6-1* • Flash-Based FPGAs**

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

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

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 6-1. 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 6-1. 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*.

FIFO Flag Usage Considerations

The AEVAL and AFVAL pins are used to specify the 12-bit AEMPTY and AFULL threshold values. The FIFO contains separate 12-bit write address (WADDR) and read address (RADDR) counters. WADDR is incremented every time a write operation is performed, and RADDR is incremented every time a read operation is performed. Whenever the difference between WADDR and RADDR is greater than or equal to AFVAL, the AFULL output is asserted. Likewise, whenever the difference between WADDR and RADDR is less than or equal to AEVAL, the AEMPTY output is asserted. To handle different read and write aspect ratios, AFVAL and AEVAL are expressed in terms of total data bits instead of total data words. When users specify AFVAL and AEVAL in terms of read or write words, the SmartGen tool translates them into bit addresses and configures these signals automatically. SmartGen configures the AFULL flag to assert when the write address exceeds the read address by at least a predefined value. In a 2k×8 FIFO, for example, a value of 1,500 for AFVAL means that the AFULL flag will be asserted after a write when the difference between the write address and the read address reaches 1,500 (there have been at least 1,500 more writes than reads). It will stay asserted until the difference between the write and read addresses drops below 1,500.

The AEMPTY flag is asserted when the difference between the write address and the read address is less than a predefined value. In the example above, a value of 200 for AEVAL means that the AEMPTY flag will be asserted when a read causes the difference between the write address and the read address to drop to 200. It will stay asserted until that difference rises above 200. Note that the FIFO can be configured with different read and write widths; in this case, the AFVAL setting is based on the number of write data entries, and the AEVAL setting is based on the number of read data entries. For aspect ratios of 512×9 and 256×18, only 4,096 bits can be addressed by the 12 bits of AFVAL and AEVAL. The number of words must be multiplied by 8 and 16 instead of 9 and 18. The SmartGen tool automatically uses the proper values. To avoid halfwords being written or read, which could happen if different read and write aspect ratios were specified, the FIFO will assert FULL or EMPTY as soon as at least one word cannot be written or read. For example, if a two-bit word is written and a four-bit word is being read, the FIFO will remain in the empty state when the first word is written. This occurs even if the FIFO is not completely empty, because in this case, a complete word cannot be read. The same is applicable in the full state. If a four-bit word is written and a two-bit word is read, the FIFO is full and one word is read. The FULL flag will remain asserted because a complete word cannot be written at this point.

Variable Aspect Ratio and Cascading

Variable aspect ratio and cascading allow users to configure the memory in the width and depth required. The memory block can be configured as a FIFO by combining the basic memory block with dedicated FIFO controller logic. The FIFO macro is named FIFO4KX18. Low power flash device RAM can be configured as 1, 2, 4, 9, or 18 bits wide. By cascading the memory blocks, any multiple of those widths can be created. The RAM blocks can be from 256 to 4,096 bits deep, depending on the aspect ratio, and the blocks can also be cascaded to create deeper areas. Refer to the aspect ratios available for each macro cell in the "SRAM Features" section on page 137. The largest continuous configurable memory area is equal to half the total memory available on the device, because the RAM is separated into two groups, one on each side of the device.

The SmartGen core generator will automatically configure and cascade both RAM and FIFO blocks. Cascading is accomplished using dedicated memory logic and does not consume user gates for depths up to 4,096 bits deep and widths up to 18, depending on the configuration. Deeper memory will utilize some user gates to multiplex the outputs.

Generated RAM and FIFO macros can be created as either structural VHDL or Verilog for easy instantiation into the design. Users of Libero SoC can create a symbol for the macro and incorporate it into a design schematic.

Table 6-10 on page 147 shows the number of memory blocks required for each of the supported depth and width memory configurations, and for each depth and width combination. For example, a 256-bit deep by 32-bit wide two-port RAM would consist of two 256×18 RAM blocks. The first 18 bits would be stored in the first RAM block, and the remaining 14 bits would be implemented in the other 256×18 RAM block. This second RAM block would have four bits of unused storage. Similarly, a dual-port memory block that is 8,192 bits deep and 8 bits wide would be implemented using 16 memory blocks. The dual-port memory would be configured in a 4,096×1 aspect ratio. These blocks would then be cascaded two deep to achieve 8,192 bits of depth, and eight wide to achieve the eight bits of width.

Automatically Assigning Technologies to I/O Banks

The I/O Bank Assigner (IOBA) tool runs automatically when you run Layout. You can also use this tool from within the MultiView Navigator (Figure 8-17). The IOBA tool automatically assigns technologies and VREF pins (if required) to every I/O bank that does not currently have any technologies assigned to it. This tool is available when at least one I/O bank is unassigned.

To automatically assign technologies to I/O banks, choose I/O Bank Assigner from the **Tools** menu (or click the I/O Bank Assigner's toolbar button, shown in Figure 8-16).

***Figure 8-16* • I/O Bank Assigner's Toolbar Button**

Messages will appear in the Output window informing you when the automatic I/O bank assignment begins and ends. If the assignment is successful, the message "I/O Bank Assigner completed successfully" appears in the Output window, as shown in Figure 8-17.

***Figure 8-17* • I/O Bank Assigner Displays Messages in Output Window**

Signal Integrity While Using ISP

For ISP of flash devices, customers are expected to follow the board-level guidelines provided on the Microsemi SoC Products Group website. These guidelines are discussed in the datasheets and application notes (refer to the “Related Documents” section of the datasheet for application note links). Customers are also expected to troubleshoot board-level signal integrity issues by measuring voltages and taking oscilloscope plots.

Programming Failure Allowances

Microsemi has strict policies regarding programming failure allowances. Please refer to *Programming and Functional Failure Guidelines* on the Microsemi SoC Products Group website for details.

Contacting the Customer Support Group

Highly skilled engineers staff the Customer Applications Center from 7:00 A.M. to 6:00 P.M., Pacific time, Monday through Friday. You can contact the center by one of the following methods:

Electronic Mail

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. Microsemi monitors the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and contact information for efficient processing of your request. The technical support email address is soc_tech@microsemi.com.

Telephone

Our Technical Support Hotline answers all calls. The center retrieves information, such as your name, company name, telephone number, and question. Once this is done, a case number is assigned. Then the center forwards the information to a queue where the first available applications engineer receives the data and returns your call. The phone hours are from 7:00 A.M. to 6:00 P.M., Pacific time, Monday through Friday.

The Customer Applications Center number is (800) 262-1060.

European customers can call +44 (0) 1256 305 600.

Remote Upgrade via TCP/IP

Transmission Control Protocol (TCP) provides a reliable bitstream transfer service between two endpoints on a network. TCP depends on Internet Protocol (IP) to move packets around the network on its behalf. TCP protects against data loss, data corruption, packet reordering, and data duplication by adding checksums and sequence numbers to transmitted data and, on the receiving side, sending back packets and acknowledging the receipt of data.

The system containing the low power flash device can be assigned an IP address when deployed in the field. When the device requires an update (core or FlashROM), the programming instructions along with the new programming data (AES-encrypted cipher text) can be sent over the Internet to the target system via the TCP/IP protocol. Once the MCU receives the instruction and data, it can proceed with the FPGA update. Low power flash devices support Message Authentication Code (MAC), which can be used to validate data for the target device. More details are given in the "Message Authentication Code (MAC) Validation/Authentication" section.

Hardware Requirement

To facilitate the programming of the low power flash families, the system must have a microprocessor (with access to the device JTAG pins) to process the programming algorithm, memory to store the programming algorithm, programming data, and the necessary programming voltage. Refer to the relevant datasheet for programming voltages.

Security

Encrypted Programming

As an additional security measure, the devices are equipped with AES decryption. AES works in two steps. The first step is to program a key into the devices in a secure or trusted programming center (such as Microsemi SoC Products Group In-House Programming (IHP) center). The second step is to encrypt any programming files with the same encryption key. The encrypted programming file will only work with the devices that have the same key. The AES used in the low power flash families is the 128-bit AES decryption engine (Rijndael algorithm).

Message Authentication Code (MAC) Validation/Authentication

As part of the AES decryption flow, the devices are equipped with a MAC validation/authentication system. MAC is an authentication tag, also called a checksum, derived by applying an on-chip authentication scheme to a STAPL file as it is loaded into the FPGA. MACs are computed and verified with the same key so they can only be verified by the intended recipient. When the MCU system receives the AES-encrypted programming data (cipher text), it can validate the data by loading it into the FPGA and performing a MAC verification prior to loading the data, via a second programming pass, into the FPGA core cells. This prevents erroneous or corrupt data from getting into the FPGA.

Low power flash devices with AES and MAC are superior to devices with only DES or 3DES encryption. Because the MAC verifies the correctness of the data, the FPGA is protected from erroneous loading of invalid programming data that could damage a device (Figure 14-5 on page 289).

The AES with MAC enables field updates over public networks without fear of having the design stolen. An encrypted programming file can only work on devices with the correct key, rendering any stolen files

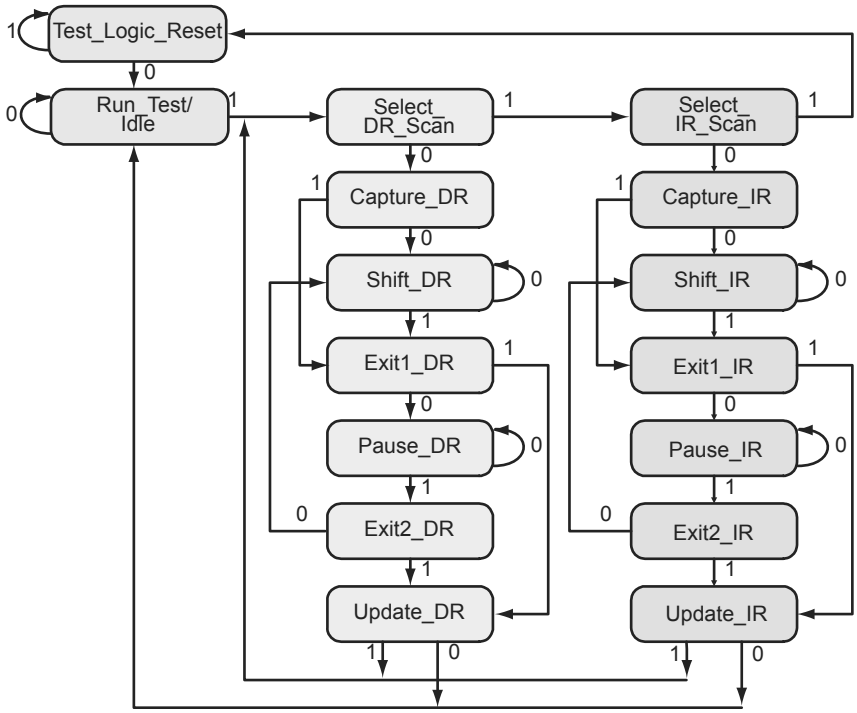


Figure 16-4 • **TAP Controller State Diagram**

UJTAG Port Usage

UIREG[7:0] hold the contents of the JTAG instruction register. The UIREG vector value is updated when the TAP Controller state machine enters the Update_IR state. Instructions 16 to 127 are user-defined and can be employed to encode multiple applications and commands within an application. Loading new instructions into the UIREG vector requires users to send appropriate logic to TMS to put the TAP Controller in a full IR cycle starting from the Select IR_Scan state and ending with the Update_IR state.

UTDI, UTDO, and UDRCK are directly connected to the JTAG TDI, TDO, and TCK ports, respectively. The TDI input can be used to provide either data (TAP Controller in the Shift_DR state) or the new contents of the instruction register (TAP Controller in the Shift_IR state).

UDRSH, UDRUPD, and UDRCAP are HIGH when the TAP Controller state machine is in the Shift_DR, Update_DR, and Capture_DR states, respectively. Therefore, they act as flags to indicate the stages of the data shift process. These flags are useful for applications in which blocks of data are shifted into the design from JTAG pins. For example, an active UDRSH can indicate that UTDI contains the data bitstream, and UDRUPD is a candidate for the end-of-data-stream flag.

As mentioned earlier, users should not connect the TDI, TDO, TCK, TMS, and TRST ports of the UJTAG macro to any port or net of the design netlist. The Designer software will automatically handle the port connection.

17 – Power-Up/-Down Behavior of Low Power Flash Devices

Introduction

Microsemi’s low power flash devices are flash-based FPGAs manufactured on a 0.13 µm process node. These devices offer a single-chip, reprogrammable solution and support Level 0 live at power-up (LAPU) due to their nonvolatile architecture.

Microsemi's low power flash FPGA families are optimized for logic area, I/O features, and performance. IGLOO® devices are optimized for power, making them the industry's lowest power programmable solution. IGLOO PLUS FPGAs offer enhanced I/O features beyond those of the IGLOO ultra-low power solution for I/O-intensive low power applications. IGLOO nano devices are the industry's lowest-power cost-effective solution. ProASIC3®L FPGAs balance low power with high performance. The ProASIC3 family is Microsemi's high-performance flash FPGA solution. ProASIC3 nano devices offer the lowest-cost solution with enhanced I/O capabilities.

Microsemi’s low power flash devices exhibit very low transient current on each power supply during power-up. The peak value of the transient current depends on the device size, temperature, voltage levels, and power-up sequence.

The following devices can have inputs driven in while the device is not powered:

- IGLOO (AGL015 and AGL030)
- IGLOO nano (all devices)
- IGLOO PLUS (AGLP030, AGLP060, AGLP125)
- IGLOOe (AGLE600, AGLE3000)
- ProASIC3L (A3PE3000L)
- ProASIC3 (A3P015, A3P030)
- ProASIC3 nano (all devices)
- ProASIC3E (A3PE600, A3PE1500, A3PE3000)
- Military ProASIC3EL (A3PE600L, A3PE3000L, but not A3P1000)
- RT ProASIC3 (RT3PE600L, RT3PE3000L)

The driven I/Os do not pull up power planes, and the current draw is limited to very small leakage current, making them suitable for applications that require cold-sparing. These devices are hot-swappable, meaning they can be inserted in a live power system.[1]

1. *For more details on the levels of hot-swap compatibility in Microsemi’s low power flash devices, refer to the "Hot-Swap Support" section in the I/O Structures chapter of the FPGA fabric user’s guide for the device you are using.*

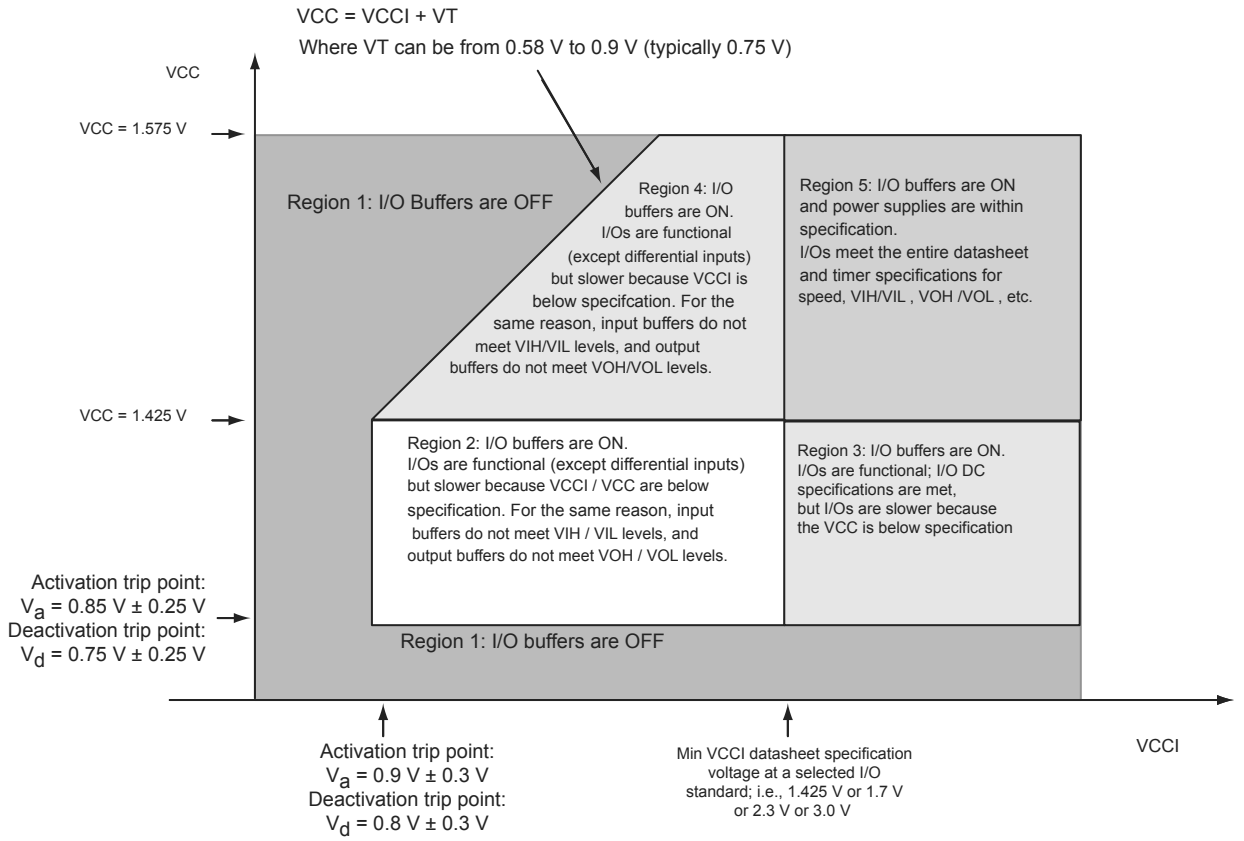


Figure 17-4 • **I/O State as a Function of VCCI and VCC Voltage Levels for IGLOO V5, IGLOO nano V5, IGLOO PLUS V5, ProASIC3L, and ProASIC3 Devices Running at VCC = 1.5 V ± 0.075 V**

Related Documents

Datasheets

ProASIC3 Flash Family FPGAs

http://www.microsemi.com/soc/documents/PA3_DS.pdf

ProASIC3E Flash Family FPGAs

http://www.microsemi.com/soc/documents/PA3E_DS.pdf

List of Changes

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

Date	Changes	Page
v1.2 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to the document as supported device types.	
v1.1 (October 2008)	The "Introduction" section was updated to add Military ProASIC3EL and RT ProASIC3 devices to the list of devices that can have inputs driven in while the device is not powered.	307
	The "Flash Devices Support Power-Up Behavior" section was revised to include new families and make the information more concise.	308
	The "Cold-Sparing" section was revised to add Military ProASIC3/EL and RT ProASIC3 devices to the lists of devices with and without cold-sparing support.	316
	The "Hot-Swapping" section was revised to add Military ProASIC3/EL and RT ProASIC3 devices to the lists of devices with and without hot-swap support. AGL400 was added to the list of devices that do not support hot-swapping.	317
v1.0 (August 2008)	This document was revised, renamed, and assigned a new part number. It now includes data for the IGLOO and ProASIC3L families.	N/A
v1.3 (March 2008)	The "List of Changes" section was updated to include the three different I/O Structure handbook chapters.	318
v1.2 (February 2008)	The first sentence of the "PLL Behavior at Brownout Condition" section was updated to read, "When PLL power supply voltage and/or V_{CC} levels drop below the VCC brownout levels (0.75 V ± 0.25 V), the PLL output lock signal goes low and/or the output clock is lost."	315
v1.1 (January 2008)	The "PLL Behavior at Brownout Condition" section was added.	315