



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	-
Number of I/O	49
Number of Gates	20000
Voltage - Supply	1.425V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	68-VFQFN Exposed Pad
Supplier Device Package	68-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pn020-qng68i

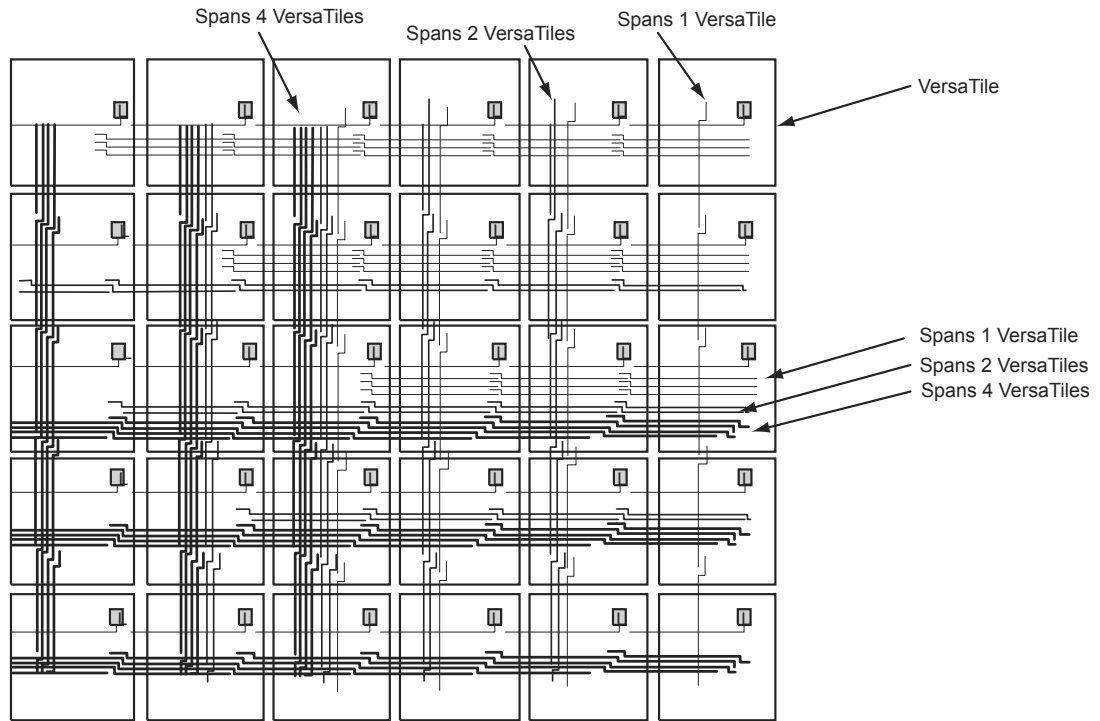


Figure 1-11 • Efficient Long-Line Resources

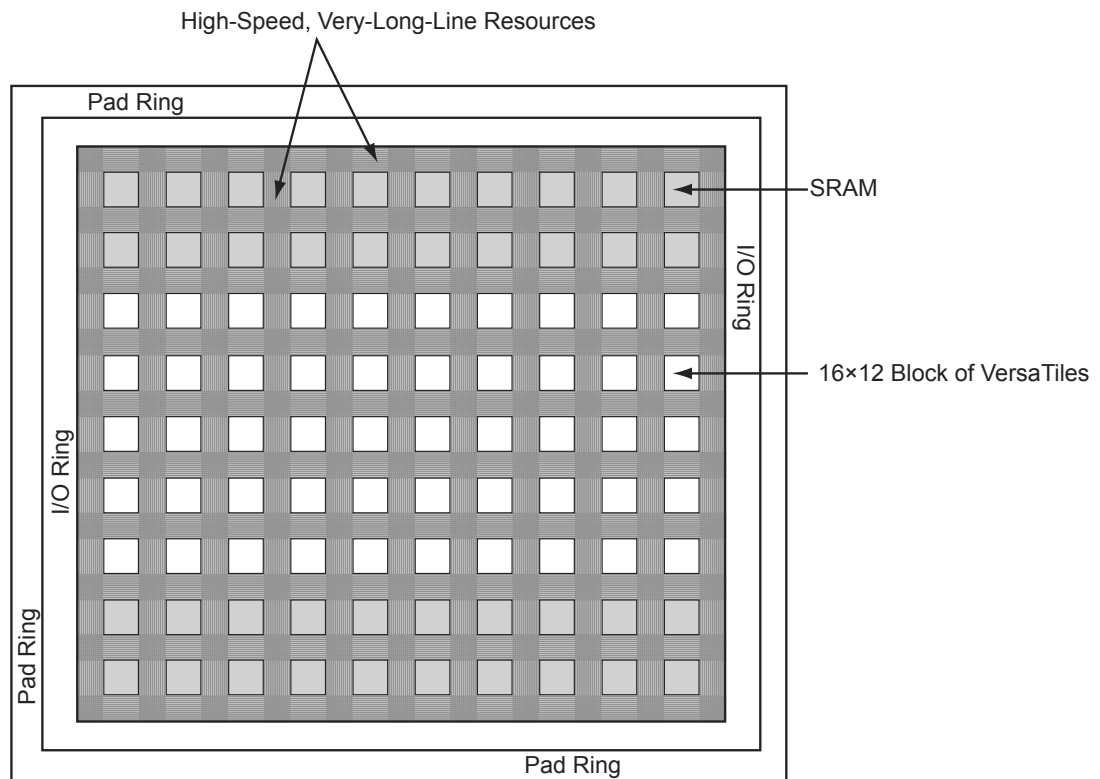


Figure 1-12 • Very-Long-Line Resources

Figure 3-6 shows all nine global inputs for the location A connected to the top left quadrant global network via CCC.

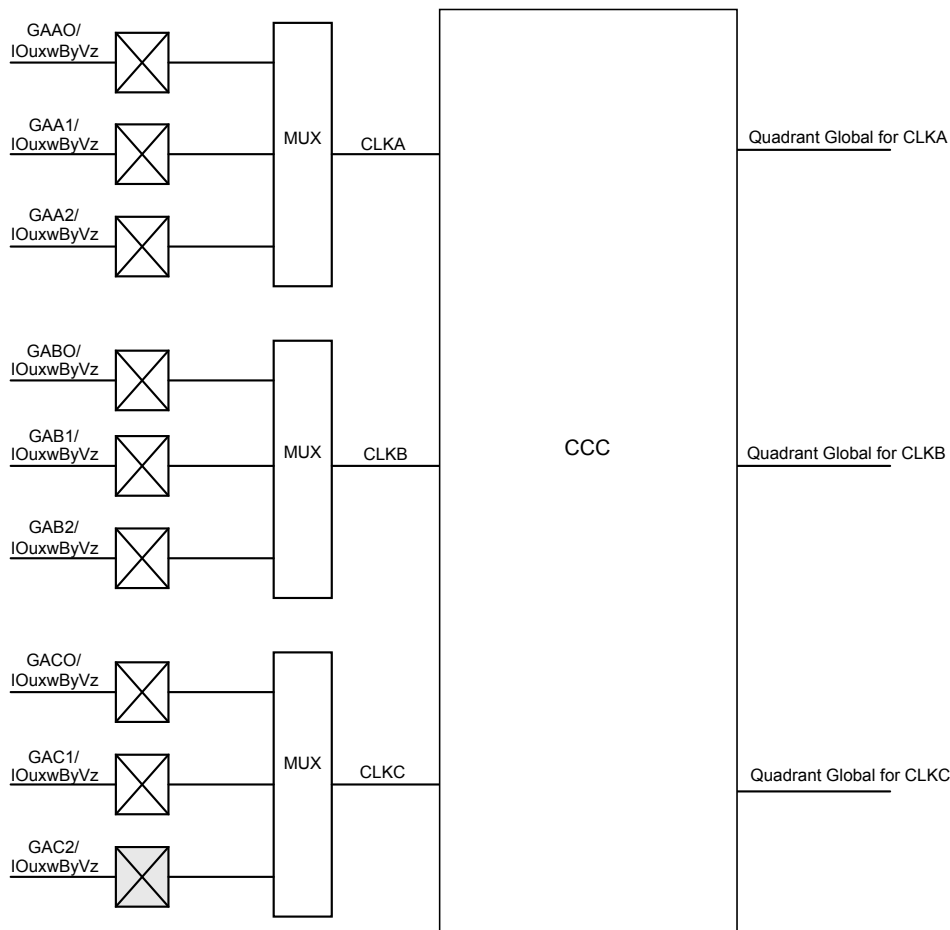


Figure 3-6 • Global Inputs

Since each bank can have a different I/O standard, the user should be careful to choose the correct global I/O for the design. There are 54 global pins available to access 18 global networks. For the single-ended and voltage-referenced I/O standards, you can use any of these three available I/Os to access the global network. For differential I/O standards such as LVDS and LVPECL, the I/O macro needs to be placed on (A0, A1), (B0, B1), (C0, C1), or a similar location. The unassigned global I/Os can be used as regular I/Os. Note that pin names starting with GF and GC are associated with the chip global networks, and GA, GB, GD, and GE are used for quadrant global networks. Table 3-2 on page 38 and Table 3-3 on page 39 show the general chip and quadrant global pin names.

Using Clock Aggregation

Clock aggregation allows for multi-spine clock domains to be assigned using hardwired connections, without adding any extra skew. A MUX tree, shown in Figure 3-8, provides the necessary flexibility to allow long lines, local resources, or I/Os to access domains of one, two, or four global spines. Signal access to the clock aggregation system is achieved through long-line resources in the central rib in the center of the die, and also through local resources in the north and south ribs, allowing I/Os to feed directly into the clock system. As Figure 3-9 indicates, this access system is contiguous.

There is no break in the middle of the chip for the north and south I/O VersaNet access. This is different from the quadrant clocks located in these ribs, which only reach the middle of the rib.

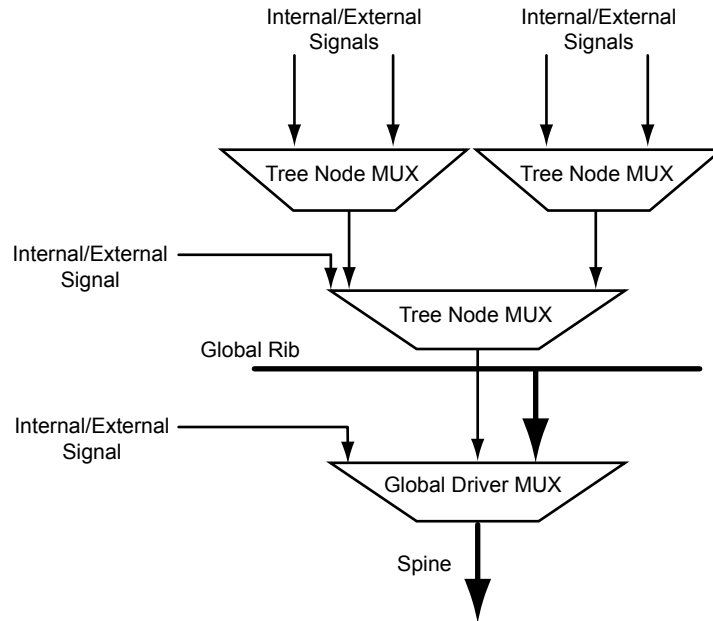


Figure 3-8 • Spine Selection MUX of Global Tree

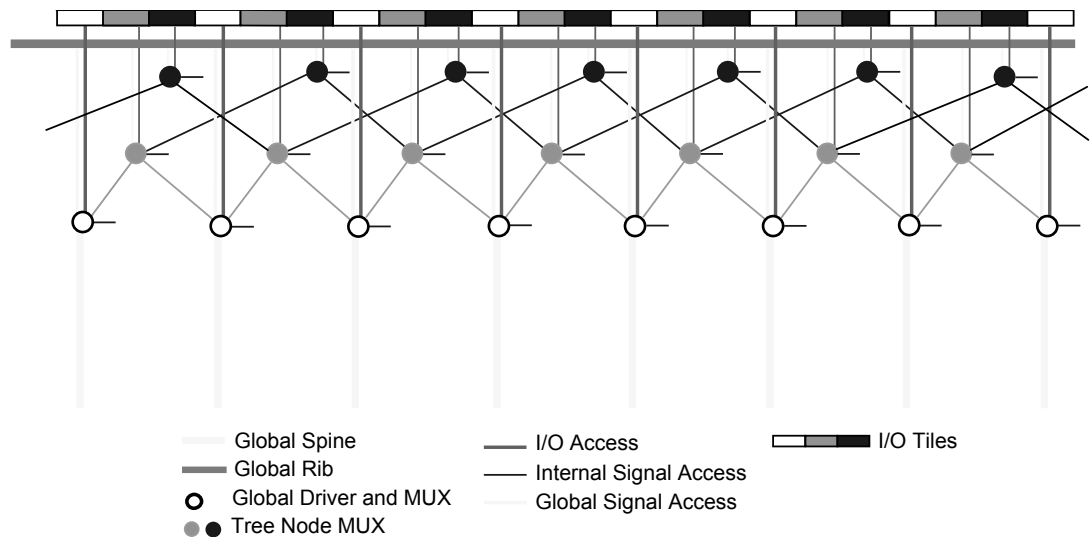


Figure 3-9 • Clock Aggregation Tree Architecture

The following will happen during demotion of a global signal to regular nets:

- CLKBUF_x becomes INBUF_x; CLKINT is removed from the netlist.
- The essential global macro, such as the output of the Clock Conditioning Circuit, cannot be demoted.
- No automatic buffering will happen.

Since no automatic buffering happens when a signal is demoted, this net may have a high delay due to large fanout. This may have a negative effect on the quality of the results. Microsemi recommends that the automatic global demotion only be used on small-fanout nets. Use clock networks for high-fanout nets to improve timing and routability.

Spine Assignment

The low power flash device architecture allows the global networks to be segmented and used as clock spines. These spines, also called local clock networks, enable the use of PDC or MVN to assign a signal to a spine.

PDC syntax to promote a net to a spine/local clock:

```
assign_local_clock -net netname -type [quadrant|chip] Tn|Bn|Tn:Bm
```

If the net is driven by a clock macro, Designer automatically demotes the clock net to a regular net before it is assigned to a spine. Nets driven by a PLL or CLKDLY macro cannot be assigned to a local clock.

When assigning a signal to a spine or quadrant global network using PDC (pre-compile), the Designer software will legalize the shared instances. The number of shared instances to be legalized can be controlled by compile options. If these networks are created in MVN (only quadrant globals can be created), no legalization is done (as it is post-compile). Designer does not do legalization between non-clock nets.

As an example, consider two nets, net_clk and net_reset, driving the same flip-flop. The following PDC constraints are used:

```
assign_local_clock -net net_clk -type chip T3
assign_local_clock -net net_reset -type chip T1:T2
```

During Compile, Designer adds a buffer in the reset net and places it in the T1 or T2 region, and places the flip-flop in the T3 spine region (Figure 3-16).

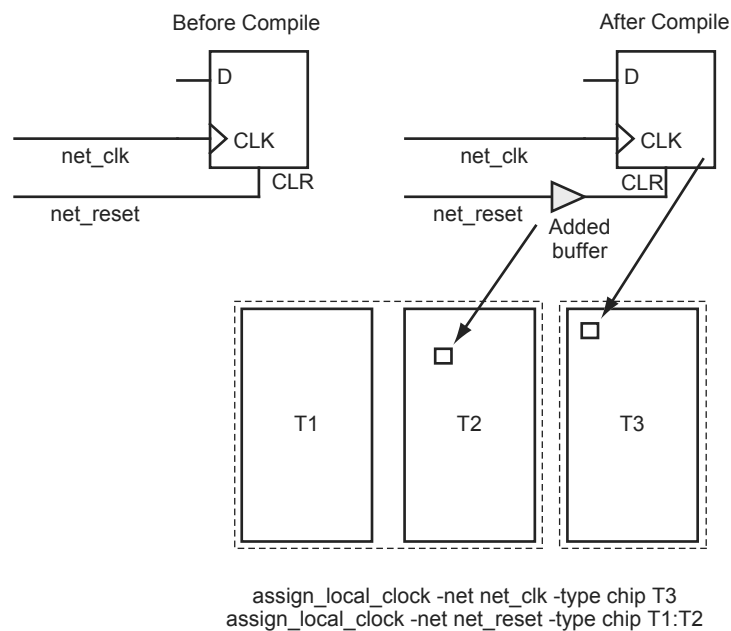


Figure 3-16 • Adding a Buffer for Shared Instances

Each CCC can implement up to three independent global buffers (with or without programmable delay) or a PLL function (programmable frequency division/multiplication, phase shift, and delays) with up to three global outputs. Unused global outputs of a PLL can be used to implement independent global buffers, up to a maximum of three global outputs for a given CCC.

CCC Programming

The CCC block is fully configurable, either via flash configuration bits set in the programming bitstream or through an asynchronous interface. This asynchronous dedicated shift register interface is dynamically accessible from inside the low power flash devices to permit parameter changes, such as PLL divide ratios and delays, during device operation.

To increase the versatility and flexibility of the clock conditioning system, the CCC configuration is determined either by the user during the design process, with configuration data being stored in flash memory as part of the device programming procedure, or by writing data into a dedicated shift register during normal device operation.

This latter mode allows the user to dynamically reconfigure the CCC without the need for core programming. The shift register is accessed through a simple serial interface. Refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 297 or the application note *Using Global Resources in Actel Fusion Devices*.

Global Resources

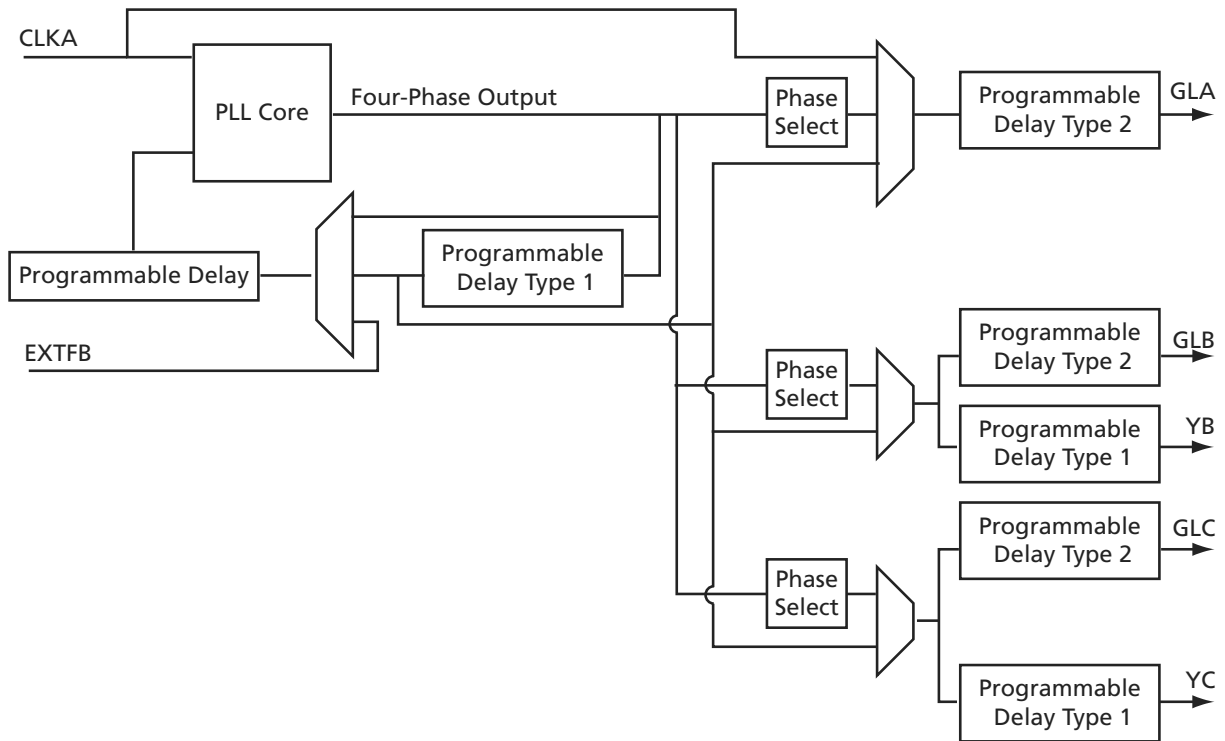
Low power flash and mixed signal devices provide three global routing networks (GLA, GLB, and GLC) for each of the CCC locations. There are potentially many I/O locations; each global I/O location can be chosen from only one of three possibilities. This is controlled by the multiplexer tree circuitry in each global network. Once the I/O location is selected, the user has the option to utilize the CCCs before the signals are connected to the global networks. The CCC in each location (up to six) has the same structure, so generating the CCC macros is always done with an identical software GUI. The CCCs in the corner locations drive the quadrant global networks, and the CCCs in the middle of the east and west chip sides drive the chip global networks. The quadrant global networks span only a quarter of the device, while the chip global networks span the entire device. For more details on global resources offered in low power flash devices, refer to the "Global Resources in Low Power Flash Devices" section on page 31.

A global buffer can be placed in any of the three global locations (CLKA-GLA, CLKB-GLB, or CLKC-GLC) of a given CCC. A PLL macro uses the CLKA CCC input to drive its reference clock. It uses the GLA and, optionally, the GLB and GLC global outputs to drive the global networks. A PLL macro can also drive the YB and YC regular core outputs. The GLB (or GLC) global output cannot be reused if the YB (or YC) output is used. Refer to the "PLL Macro Signal Descriptions" section on page 68 for more information.

Each global buffer, as well as the PLL reference clock, can be driven from one of the following:

- 3 dedicated single-ended I/Os using a hardwired connection
- 2 dedicated differential I/Os using a hardwired connection (not supported for IGLOO nano or ProASIC3 nano devices)
- The FPGA core

SmartGen also allows the user to select the various delays and phase shift values necessary to adjust the phases between the reference clock (CLKA) and the derived clocks (GLA, GLB, GLC, YB, and YC). SmartGen allows the user to select the input clock source. SmartGen automatically instantiates the special macro, PLLINT, when needed.



Note: Clock divider and clock multiplier blocks are not shown in this figure or in SmartGen. They are automatically configured based on the user's required frequencies.

Figure 4-6 • CCC with PLL Block

Global Input Selections

Low power flash devices provide the flexibility of choosing one of the three global input pad locations available to connect to a CCC functional block or to a global / quadrant global network. Figure 4-7 on page 72 and Figure 4-8 on page 72 show the detailed architecture of each global input structure for 30 k gate devices and below, as well as 60 k gate devices and above, respectively. For 60 k gate devices and above (Figure 4-7 on page 72), if the single-ended I/O standard is chosen, there is flexibility to choose one of the global input pads (the first, second, and fourth input). Once chosen, the other I/O locations are used as regular I/Os. If the differential I/O standard is chosen (not applicable for IGLOO nano and ProASIC3 nano devices), the first and second inputs are considered as paired, and the third input is paired with a regular I/O.

The user then has the choice of selecting one of the two sets to be used as the clock input source to the CCC functional block. There is also the option to allow an internal clock signal to feed the global network or the CCC functional block. A multiplexer tree selects the appropriate global input for routing to the desired location. Note that the global I/O pads do not need to feed the global network; they can also be used as regular I/O pads.

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

Figure 4-36 • Second-Stage PLL Showing Input of 256 MHz from First Stage and Final Output of 280 MHz

Figure 4-37 shows the simulation results, where the first PLL's output period is 3.9 ns (~256 MHz), and the stage 2 (final) output period is 3.56 ns (~280 MHz).

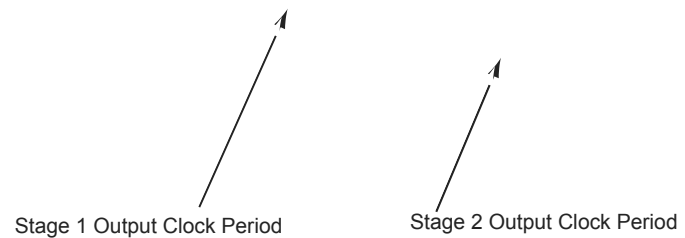


Figure 4-37 • ModelSim Simulation Results

I/O Features

Both IGLOO nano and ProASIC3 nano devices support multiple I/O features that make board design easier. For example, an I/O feature like Schmitt Trigger in the input buffer saves the board space that would be used by an external Schmitt trigger for a slow or noisy input signal. These features are also programmable for each I/O, which in turn gives flexibility in interfacing with other components. The following is a detailed description of all available features in nano devices.

I/O Programmable Features

Low power flash devices offer many flexible I/O features to support a wide variety of board designs. Some of the features are programmable, with a range for selection. Table 7-7 lists programmable I/O features and their ranges.

Table 7-7 • Programmable I/O Features (user control via I/O Attribute Editor)

Feature	Description	Range
Slew Control	Output slew rate	HIGH, LOW
Output Drive (mA)	Output drive strength	Depends on I/O type
Resistor Pull	Weak resistor pull circuit	Up, Down, None
Schmitt Trigger	Schmitt trigger for input only	ON, OFF

Hot-Swap Support

All nano devices are hot-swappable.

The hot-swap feature appears as a read-only check box in the I/O Attribute Editor that shows whether an I/O is hot-swappable or not. Refer to the "Power-Up/-Down Behavior of Low Power Flash Devices" section on page 307 for details on hot-swapping.

Hot-swapping is the operation of hot insertion or hot removal of a card in a powered-up system. The levels of hot-swap support and examples of related applications are described in Table 7-8 on page 168 to Table 7-11 on page 169. The I/Os also need to be configured in hot-insertion mode if hot-plugging compliance is required. nano devices have an I/O structure that allows the support of Level 3 and Level 4 hot-swap with only two levels of staging.

Table 7-8 • Hot-Swap Level 1

Description	Cold-swap
Power Applied to Device	No
Bus State	–
Card Ground Connection	–
Device Circuitry Connected to Bus Pins	–
Example Application	System and card with Microsemi FPGA chip are powered down, and the card is plugged into the system. Then the power supplies are turned on for the system but not for the FPGA on the card.
Compliance of nano Devices	Compliant

Table 7-9 • Hot-Swap Level 2

Description	Hot-swap while reset
Power Applied to Device	Yes
Bus State	Held in reset state
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	–
Example Application	In the PCI hot-plug specification, reset control circuitry isolates the card busses until the card supplies are at their nominal operating levels and stable.
Compliance of nano Devices	Compliant

Table 7-13 • Comparison Table for 5 V–Compliant Receiver Solutions

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

Notes:

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

5 V Output Tolerance

nano Standard I/Os must be set to 3.3 V LVTTTL or 3.3 V LVCMOS mode to reliably drive 5 V TTL receivers. It is also critical that there be NO external I/O pull-up resistor to 5 V, since this resistor would pull the I/O pad voltage beyond the 3.6 V absolute maximum value and consequently cause damage to the I/O.

When set to 3.3 V LVTTTL or 3.3 V LVCMOS mode, the I/Os can directly drive signals into 5 V TTL receivers. In fact, $V_{OL} = 0.4$ V and $V_{OH} = 2.4$ V in both 3.3 V LVTTTL and 3.3 V LVCMOS modes exceeds the $V_{IL} = 0.8$ V and $V_{IH} = 2$ V level requirements of 5 V TTL receivers. Therefore, level 1 and level 0 will be recognized correctly by 5 V TTL receivers.

Schmitt Trigger

A Schmitt trigger is a buffer used to convert a slow or noisy input signal into a clean one before passing it to the FPGA. Using Schmitt trigger buffers guarantees a fast, noise-free input signal to the FPGA.

nano devices have Schmitt triggers built into their I/O circuitry. Schmitt Trigger is available on all I/O configurations.

This feature can be implemented by using a Physical Design Constraints (PDC) command (Table 7-5 on page 163) or by selecting a check box in the I/O Attribute Editor in Designer. The check box is cleared by default.

I/O Register Combining

Every I/O has several embedded registers in the I/O tile that are close to the I/O pads. Rather than using the internal register from the core, the user has the option of using these registers for faster clock-to-out timing, and external hold and setup. When combining these registers at the I/O buffer, some architectural rules must be met. Provided these rules are met, the user can enable register combining globally during Compile (as shown in the "Compiling the Design" section in the "I/O Software Control in Low Power Flash Devices" section on page 185).

This feature is supported by all I/O standards.

Rules for Registered I/O Function:

1. The fanout between an I/O pin (D, Y, or E) and a register must be equal to one for combining to be considered on that pin.
2. All registers (Input, Output, and Output Enable) connected to an I/O must share the same clear or preset function:
 - If one of the registers has a CLR pin, all the other registers that are candidates for combining in the I/O must have a CLR pin.

VHDL

```

library ieee;
use ieee.std_logic_1164.all;
library proasic3; use proasic3.all;

entity DDR_BiDir_HSTL_I_LowEnb is
    port(DataR, DataF, CLR, CLK, Trien : in std_logic; QR, QF : out std_logic;
        PAD : inout std_logic) ;
end DDR_BiDir_HSTL_I_LowEnb;

architecture DEF_ARCH of  DDR_BiDir_HSTL_I_LowEnb is

    component INV
        port(A : in std_logic := 'U'; Y : out std_logic) ;
    end component;

    component DDR_OUT
        port(DR, DF, CLK, CLR : in std_logic := 'U'; Q : out std_logic) ;
    end component;

    component DDR_REG
        port(D, CLK, CLR : in std_logic := 'U'; QR, QF : out std_logic) ;
    end component;

    component BIBUF_HSTL_I
        port(PAD : inout std_logic := 'U'; D, E : in std_logic := 'U'; Y : out std_logic) ;
    end component;

    signal TrienAux, D, Q : std_logic ;

begin

    Inv_Tri : INV
    port map(A => Trien, Y => TrienAux);
    DDR_OUT_0_inst : DDR_OUT
    port map(DR => DataR, DF => DataF, CLK => CLK, CLR => CLR, Q => Q);
    DDR_REG_0_inst : DDR_REG
    port map(D => D, CLK => CLK, CLR => CLR, QR => QR, QF => QF);
    BIBUF_HSTL_I_0_inst : BIBUF_HSTL_I
    port map(PAD => PAD, D => Q, E => TrienAux, Y => D);

end DEF_ARCH;

```

Types of Programming for Flash Devices

The number of devices to be programmed will influence the optimal programming methodology. Those available are listed below:

- In-system programming
 - Using a programmer
 - Using a microprocessor or microcontroller
- Device programmers
 - Single-site programmers
 - Multi-site programmers, batch programmers, or gang programmers
 - Automated production (robotic) programmers
- Volume programming services
 - Microsemi in-house programming
 - Programming centers

In-System Programming

Device Type Supported: Flash

ISP refers to programming the FPGA after it has been mounted on the system printed circuit board. The FPGA may be preprogrammed and later reprogrammed using ISP.

The advantage of using ISP is the ability to update the FPGA design many times without any changes to the board. This eliminates the requirement of using a socket for the FPGA, saving cost and improving reliability. It also reduces programming hardware expenses, as the ISP methodology is die-/package-independent.

There are two methods of in-system programming: external and internal.

- Programmer ISP—Refer to the "In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" section on page 261 for more information.

Using an external programmer and a cable, the device can be programmed through a header on the system board. In Microsemi SoC Products Group documentation, this is referred to as external ISP. Microsemi provides FlashPro4, FlashPro3, FlashPro Lite, or Silicon Sculptor 3 to perform external ISP. Note that Silicon Sculptor II and Silicon Sculptor 3 can only provide ISP for ProASIC and ProASIC^{PLUS}® families, not for SmartFusion, Fusion, IGLOO, or ProASIC3. Silicon Sculptor II and Silicon Sculptor 3 can be used for programming ProASIC and ProASIC^{PLUS} devices by using an adapter module (part number SMPA-ISP-ACTEL-3).

 - Advantages: Allows local control of programming and data files for maximum security. The programming algorithms and hardware are available from Microsemi. The only hardware required on the board is a programming header.
 - Limitations: A negligible board space requirement for the programming header and JTAG signal routing
- Microprocessor ISP—Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

Using a microprocessor and an external or internal memory, you can store the program in memory and use the microprocessor to perform the programming. In Microsemi documentation, this is referred to as internal ISP. Both the code for the programming algorithm and the FPGA programming file must be stored in memory on the board. Programming voltages must also be generated on the board.

 - Advantages: The programming code is stored in the system memory. An external programmer is not required during programming.
 - Limitations: This is the approach that requires the most design work, since some way of getting and/or storing the data is needed; a system interface to the device must be designed; and the low-level API to the programming firmware must be written and linked into the code provided by Microsemi. While there are benefits to this methodology, serious thought and planning should go into the decision.

Cortex-M1 Device Security

Cortex-M1-enabled devices are shipped with the following security features:

- FPGA array enabled for AES-encrypted programming and verification
- FlashROM enabled for AES-encrypted Write and Verify
- Fusion Embedded Flash Memory enabled for AES-encrypted Write

AES Encryption of Programming Files

Low power flash devices employ AES as part of the security mechanism that prevents invasive and noninvasive attacks. The mechanism entails encrypting the programming file with AES encryption and then passing the programming file through the AES decryption core, which is embedded in the device. The file is decrypted there, and the device is successfully programmed. The AES master key is stored in on-chip nonvolatile memory (flash). The AES master key can be preloaded into parts in a secure programming environment (such as the Microsemi In-House Programming center), and then "blank" parts can be shipped to an untrusted programming or manufacturing center for final personalization with an AES-encrypted bitstream. Late-stage product changes or personalization can be implemented easily and securely by simply sending a STAPL file with AES-encrypted data. Secure remote field updates over public networks (such as the Internet) are possible by sending and programming a STAPL file with AES-encrypted data.

The AES key protects the programming data for file transfer into the device with 128-bit AES encryption. If AES encryption is used, the AES key is stored or preprogrammed into the device. To program, you must use an AES-encrypted file, and the encryption used on the file must match the encryption key already in the device.

The AES key is protected by a FlashLock security Pass Key that is also implemented in each device. The AES key is always protected by the FlashLock Key, and the AES-encrypted file does NOT contain the FlashLock Key. This FlashLock Pass Key technology is exclusive to the Microsemi flash-based device families. FlashLock Pass Key technology can also be implemented without the AES encryption option, providing a choice of different security levels.

In essence, security features can be categorized into the following three options:

- AES encryption with FlashLock Pass Key protection
- FlashLock protection only (no AES encryption)
- No protection

Each of the above options is explained in more detail in the following sections with application examples and software implementation options.

Advanced Encryption Standard

The 128-bit AES standard (FIPS-192) block cipher is the NIST (National Institute of Standards and Technology) replacement for DES (Data Encryption Standard FIPS46-2). AES has been designed to protect sensitive government information well into the 21st century. It replaces the aging DES, which NIST adopted in 1977 as a Federal Information Processing Standard used by federal agencies to protect sensitive, unclassified information. The 128-bit AES standard has 3.4×10^{38} possible 128-bit key variants, and it has been estimated that it would take 1,000 trillion years to crack 128-bit AES cipher text using exhaustive techniques. Keys are stored (securely) in low power flash devices in nonvolatile flash memory. All programming files sent to the device can be authenticated by the part prior to programming to ensure that bad programming data is not loaded into the part that may possibly damage it. All programming verification is performed on-chip, ensuring that the contents of low power flash devices remain secure.

Microsemi has implemented the 128-bit AES (Rijndael) algorithm in low power flash devices. With this key size, there are approximately 3.4×10^{38} possible 128-bit keys. DES has a 56-bit key size, which provides approximately 7.2×10^{16} possible keys. In their AES fact sheet, the National Institute of Standards and Technology uses the following hypothetical example to illustrate the theoretical security provided by AES. If one were to assume that a computing system existed that could recover a DES key in a second, it would take that same machine approximately 149 trillion years to crack a 128-bit AES key. NIST continues to make their point by stating the universe is believed to be less than 20 billion years old.¹

3. Choose the desired settings for the FlashROM configurations to be programmed (Figure 11-13). Click **Finish** to generate the STAPL programming file for the design.
-

Figure 11-13 • 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 243, 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 11-4 and Table 11-5 on page 251.

Table 11-4 • 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	✓	✓	✓

Programming Voltage (VPUMP) and VJTAG

Low-power flash devices support on-chip charge pumps, and therefore require only a single 3.3 V programming voltage for the VPUMP pin during programming. When the device is not being programmed, the VPUMP pin can be left floating or can be tied (pulled up) to any voltage between 0 V and 3.6 V². During programming, the target board or the FlashPro4/3/3X programmer can provide VPUMP. FlashPro4/3/3X is capable of supplying VPUMP to a single device. If more than one device is to be programmed using FlashPro4/3/3X on a given board, FlashPro4/3/3X should not be relied on to supply the VPUMP voltage. A FlashPro4/3/3X programmer is not capable of providing reliable VJTAG voltage. The board must supply VJTAG voltage to the device and the VJTAG pin of the programmer header must be connected to the device VJTAG pin. Microsemi recommends that VPUMP³ and VJTAG power supplies be kept separate with independent filtering capacitors rather than supplying them from a common rail. Refer to the "Board-Level Considerations" section on page 271 for capacitor requirements.

Low power flash device I/Os support a bank-based, voltage-supply architecture that simultaneously supports multiple I/O voltage standards (Table 12-2). By isolating the JTAG power supply in a separate bank from the user I/Os, low power flash devices provide greater flexibility with supply selection and simplify power supply and printed circuit board (PCB) design. The JTAG pins can be run at any voltage from 1.5 V to 3.3 V (nominal). Microsemi recommends that TCK be tied to GND through a 200 ohm to 1 Kohm resistor. This prevents a possible totempole current on the input buffer stage. For TDI, TMS, and TRST pins, the devices provide an internal nominal 10 Kohm pull-up resistor. During programming, all I/O pins, except for JTAG interface pins, are tristated and weakly pulled up to VCCI. This isolates the part and prevents the signals from floating. The JTAG interface pins are driven by the FlashPro4/3/3X during programming, including the TRST pin, which is driven HIGH.

Table 12-2 • Power Supplies

Power Supply	Programming Mode	Current during Programming
VCC	1.2 V / 1.5 V	< 70 mA
VCCI	1.2 V / 1.5 V / 1.8 V / 2.5 V / 3.3 V (bank-selectable)	I/Os are weakly pulled up.
VJTAG	1.2 V / 1.5 V / 1.8 V / 2.5 V / 3.3 V	< 20 mA
VPUMP	3.15 V to 3.45 V	< 80 mA

Note: All supply voltages should be at 1.5 V or higher, regardless of the setting during normal operation, except for IGLOO nano, where 1.2 V VCC and VJTAG programming is allowed.

Nonvolatile Memory (NVM) Programming Voltage

SmartFusion and Fusion devices need stable VCCNVM/VCCENV³ (1.5 V power supply to the embedded nonvolatile memory blocks) and VCCOSC/VCCROSC⁴ (3.3 V power supply to the integrated RC oscillator). The tolerance of VCCNVM/VCCENV is $\pm 5\%$ and VCCOSC/VCCROSC is $\pm 5\%$.

Unstable supply voltage on these pins can cause an NVM programming failure due to NVM page corruption. The NVM page can also be corrupted if the NVM reset pin has noise. This signal must be tied off properly.

Microsemi recommends installing the following capacitors⁵ on the VCCNVM/VCCENV and VCCOSC/VCCROSC pins:

- Add one bypass capacitor of 10 μF for each power supply plane followed by an array of decoupling capacitors of 0.1 μF .
- Add one 0.1 μF capacitor near each pin.

2. During sleep mode in IGLOO devices connect VPUMP to GND.
3. VPUMP has to be quiet for successful programming. Therefore VPUMP must be separate and required capacitors must be installed close to the FPGA VPUMP pin.
4. VCCROSC is for SmartFusion.
5. The capacitors cannot guarantee reliable operation of the device if the board layout is not done properly.

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.

UJTAG Macro

The UJTAG tiles can be instantiated in a design using the UJTAG macro from the Fusion, IGLOO, or ProASIC3 macro library. Note that "UJTAG" is a reserved name and cannot be used for any other user-defined blocks. A block symbol of the UJTAG tile macro is presented in Figure 16-2. In this figure, the ports on the left side of the block are connected to the JTAG TAP Controller, and the right-side ports are accessible by the FPGA core VersaTiles. The TDI, TMS, TDO, TCK, and TRST ports of UJTAG are only provided for design simulation purposes and should be treated as external signals in the design netlist. However, these ports must NOT be connected to any I/O buffer in the netlist. Figure 16-3 on page 300 illustrates the correct connection of the UJTAG macro to the user design netlist. Microsemi Designer software will automatically connect these ports to the TAP during place-and-route. Table 16-2 gives the port descriptions for the rest of the UJTAG ports:

Table 16-2 • UJTAG Port Descriptions

Port	Description
UIREG [7:0]	This 8-bit bus carries the contents of the JTAG Instruction Register of each device. Instruction Register values 16 to 127 are not reserved and can be employed as user-defined instructions.
URSTB	URSTB is an active-low signal and will be asserted when the TAP Controller is in Test-Logic-Reset mode. URSTB is asserted at power-up, and a power-on reset signal resets the TAP Controller. URSTB will stay asserted until an external TAP access changes the TAP Controller state.
UTDI	This port is directly connected to the TAP's TDI signal.
UTDO	This port is the user TDO output. Inputs to the UTDO port are sent to the TAP TDO output MUX when the IR address is in user range.
UDRSH	Active-high signal enabled in the ShiftDR TAP state
UDRCAP	Active-high signal enabled in the CaptureDR TAP state
UDRCK	This port is directly connected to the TAP's TCK signal.
UDRUPD	Active-high signal enabled in the UpdateDR TAP state

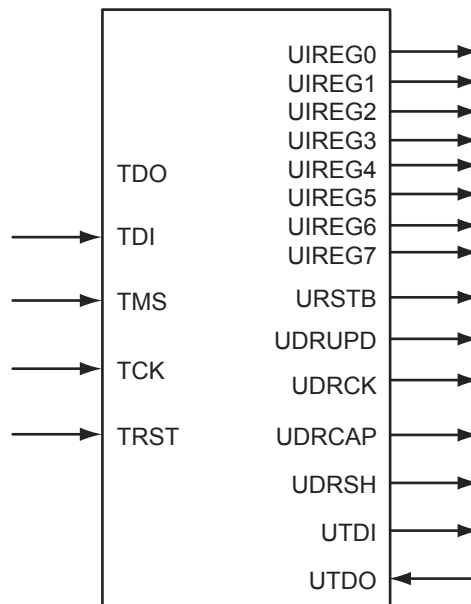


Figure 16-2 • UJTAG Tile Block Symbol

Typical UJTAG Applications

Bidirectional access to the JTAG port from VersaTiles—without putting the device into test mode—creates flexibility to implement many different applications. This section describes a few of these. All are based on importing/exporting data through the UJTAG tiles.

Clock Conditioning Circuitry—Dynamic Reconfiguration

In low power flash devices, CCCs, which include PLLs, can be configured dynamically through either an 81-bit embedded shift register or static flash programming switches. These 81 bits control all the characteristics of the CCC: routing MUX architectures, delay values, divider values, etc. Table 16-3 lists the 81 configuration bits in the CCC.

Table 16-3 • Configuration Bits of Fusion, IGLOO, and ProASIC3 CCC Blocks

Bit Number(s)	Control Function
80	RESET ENABLE
79	DYNCSEL
78	DYNBSEL
77	DYNASEL
<76:74>	VCOSSEL [2:0]
73	STATCSEL
72	STATBSEL
71	STATASEL
<70:66>	DLYC [4:0]
<65:61>	DLYB [4:0]
<60:56>	DLYGLC [4:0]
<55:51>	DLYGLB [4:0]
<50:46>	DLYGLA [4:0]
45	XDLYSEL
<44:40>	FBDLY [4:0]
<39:38>	FBSEL
<37:35>	OCMUX [2:0]
<34:32>	OBMUX [2:0]
<31:29>	OAMUX [2:0]
<28:24>	OCDIV [4:0]
<23:19>	OBDIV [4:0]
<18:14>	OADIV [4:0]
<13:7>	FBDIV [6:0]
<6:0>	FINDIV [6:0]

The embedded 81-bit shift register (for the dynamic configuration of the CCC) is accessible to the VersaTiles, which, in turn, have access to the UJTAG tiles. Therefore, the CCC configuration shift register can receive and load the new configuration data stream from JTAG.

Dynamic reconfiguration eliminates the need to reprogram the device when reconfiguration of the CCC functional blocks is needed. The CCC configuration can be modified while the device continues to operate. Employing the UJTAG core requires the user to design a module to provide the configuration data and control the CCC configuration shift register. In essence, this is a user-designed TAP Controller requiring chip resources.

Similar reconfiguration capability exists in the ProASIC^{PLUS}® family. The only difference is the number of shift register bits controlling the CCC (27 in ProASIC^{PLUS} and 81 in IGLOO, ProASIC3, and Fusion).

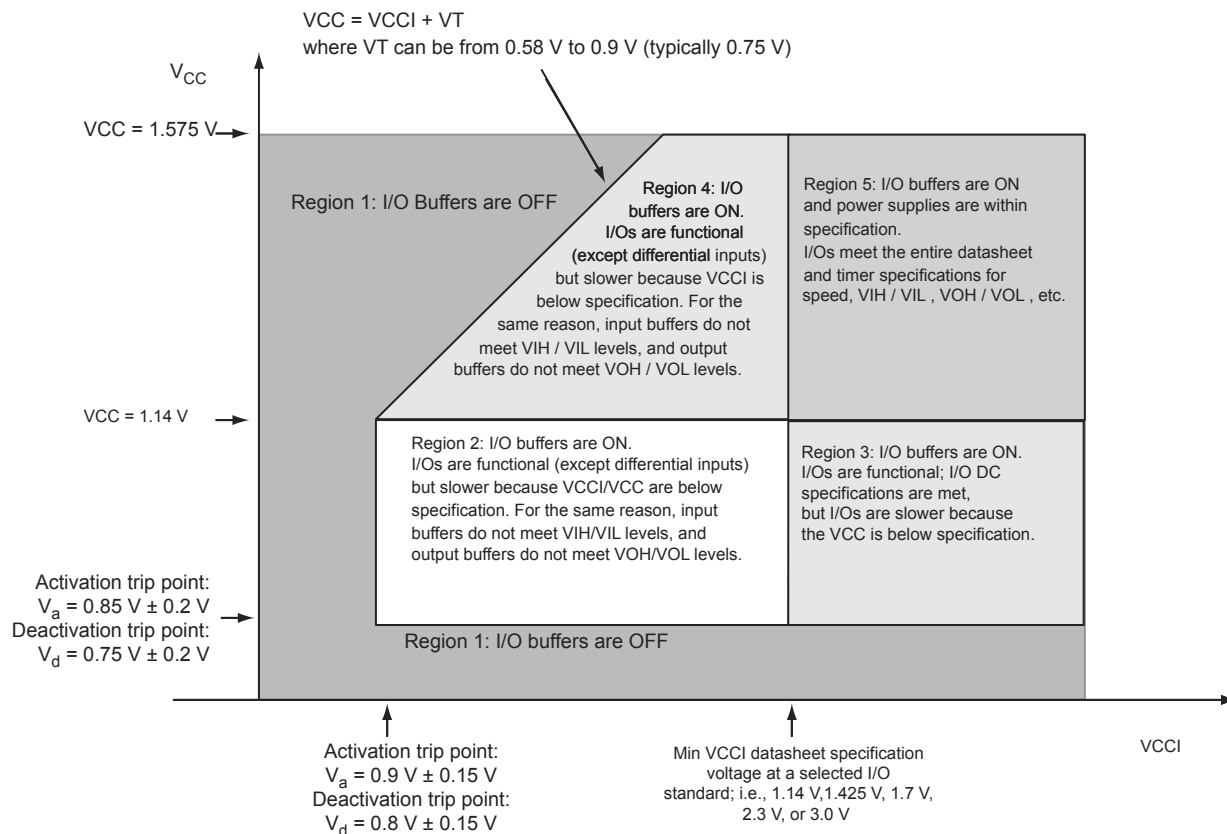


Figure 17-5 • I/O State as a Function of V_{CCI} and V_{CC} Voltage Levels for IGLOO V2, IGLOO nano V2, IGLOO PLUS V2, and ProASIC3L Devices Running at $V_{CC} = 1.2 \text{ V} \pm 0.06 \text{ V}$