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	-
Number of I/O	34
Number of Gates	10000
Voltage - Supply	1.425V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-20°C ~ 85°C (TJ)
Package / Case	48-VFQFN Exposed Pad
Supplier Device Package	48-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pn010-1qng48

Global Resource Support in Flash-Based Devices

The flash FPGAs listed in Table 3-1 support the global resources and the functions described in this document.

Table 3-1 • **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 PLUS	IGLOO FPGAs with enhanced I/O capabilities
	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 products as listed in Table 3-1. 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 3-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.
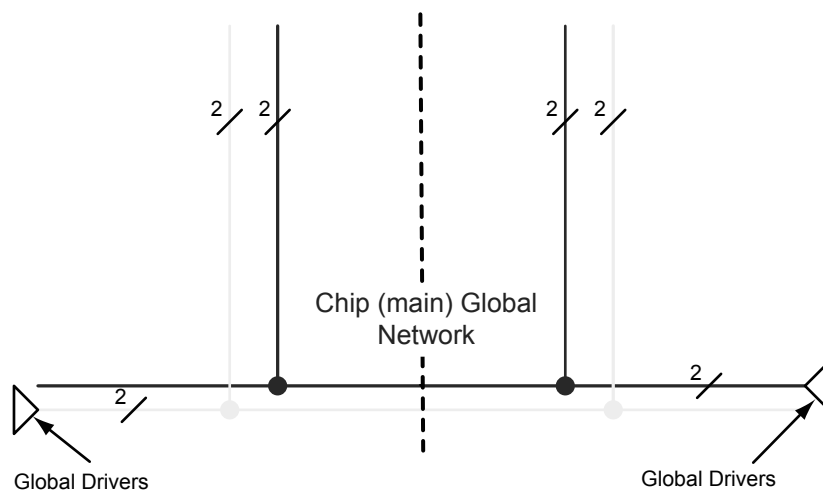
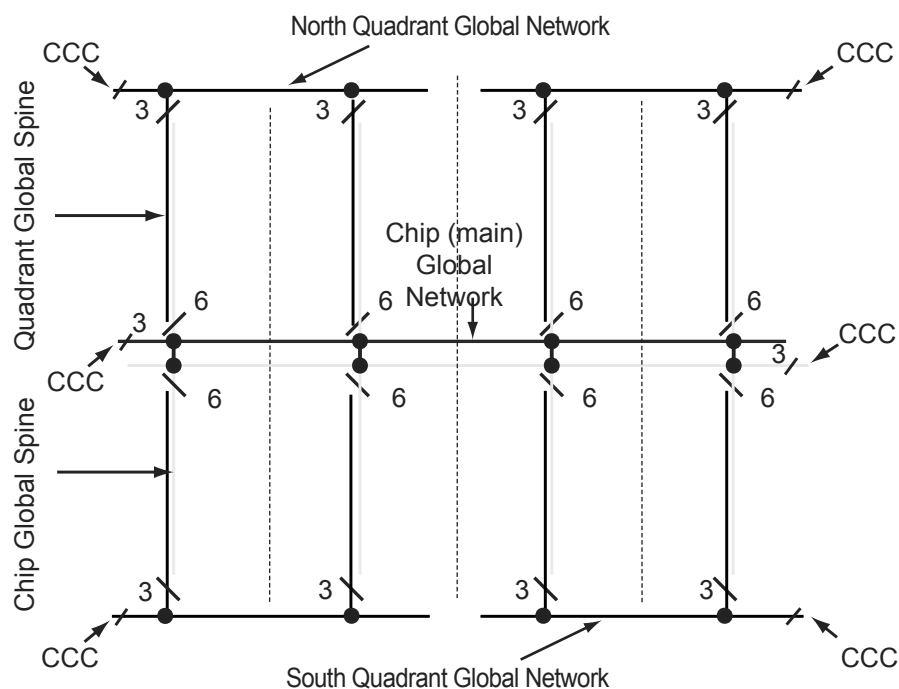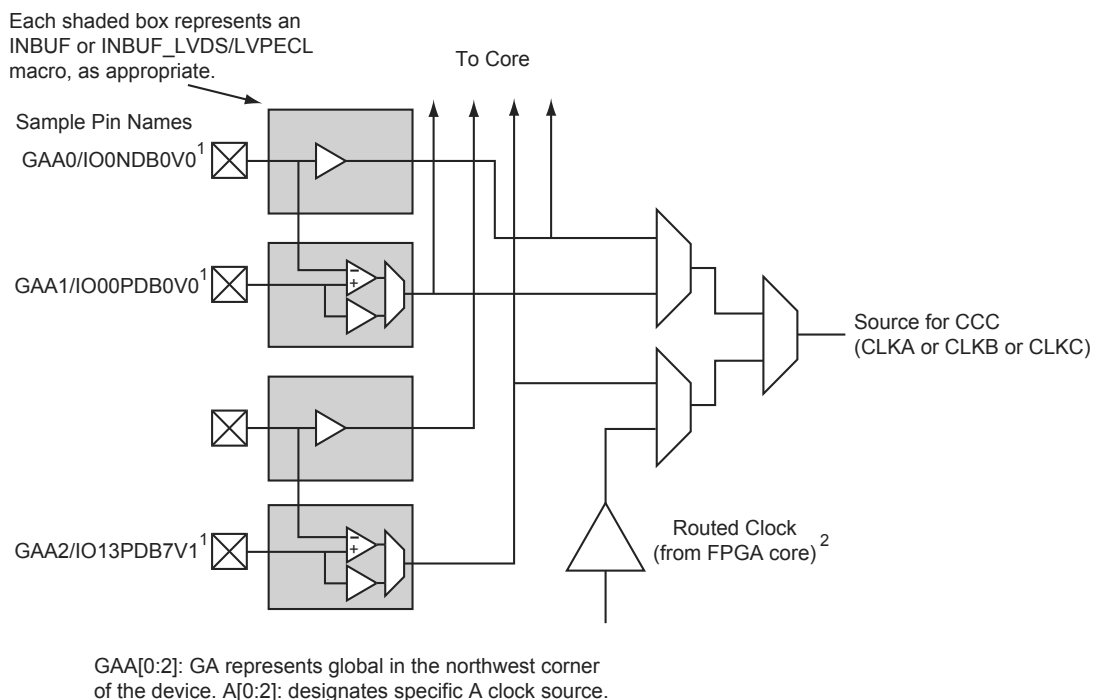*Figure 3-2 •* **Simplified VersaNet Global Network (30 k gates and below)**

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

Figure 3-5 shows more detailed global input connections. It shows the global input pins connection to the northwest quadrant global networks. 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

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

Note: Differential inputs are not supported for IGLOO nano or ProASIC3 nano devices.

***Figure 3-5 •* Global I/O Overview**

Table 3-5 • **Globals/Spines/Rows for IGLOO PLUS Devices**

IGLOO PLUS Devices	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
AGLP030	6	0	2	9	18	384*	792	12
AGLP060	6	12	4	9	36	384*	1,584	12
AGLP125	6	12	8	9	72	384*	3,120	12

*Note: *Clock trees that are located at far left and far right will support more VersaTiles.*

Table 3-6 • **Globals/Spines/Rows for Fusion Devices**

Fusion Device	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
AFS090	6	12	6	9	54	384	2,304	12
AFS250	6	12	8	9	72	768	6,144	24
AFS600	6	12	12	9	108	1,152	13,824	36
AFS1500	6	12	20	9	180	1,920	38,400	60

You can control the maximum number of shared instances allowed for the legalization to take place using the Compile Option dialog box shown in Figure 3-17. Refer to Libero SoC / Designer online help for details on the Compile Option dialog box. A large number of shared instances most likely indicates a floorplanning problem that you should address.

Figure 3-17 • **Shared Instances in the Compile Option Dialog Box**

Designer Flow for Global Assignment

To achieve the desired result, pay special attention to global management during synthesis and place-and-route. The current Synplify tool does not insert more than six global buffers in the netlist by default. Thus, the default flow will not assign any signal to the quadrant global network. However, you can use attributes in Synplify and increase the default global macro assignment in the netlist. Designer v6.2 supports automatic quadrant global assignment, which was not available in Designer v6.1. Layout will make the choice to assign the correct signals to global. However, you can also utilize PDC and perform manual global assignment to overwrite any automatic assignment. The following step-by-step suggestions guide you in the layout of your design and help you improve timing in Designer:

1. Run Compile and check the Compile report. The Compile report has global information in the "Device Utilization" section that describes the number of chip and quadrant signals in the design. A "Net Report" section describes chip global nets, quadrant global nets, local clock nets, a list of nets listed by fanout, and net candidates for local clock assignment. Review this information. Note that YB or YC are counted as global only when they are used in isolation; if you use YB only and not GLB, this net is not shown in the global/quadrant nets report. Instead, it appears in the Global Utilization report.

2. If some signals have a very high fanout and are candidates for global promotion, promote those signals to global using the compile options or PDC commands. Figure 3-18 on page 54 shows the Globals Management section of the compile options. Select **Promote regular nets whose fanout is greater than** and enter a reasonable value for fanouts.

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).

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.*

When SmartGen is used to define the configuration that will be shifted in via the serial interface, SmartGen prints out the values of the 81 configuration bits. For ease of use, several configuration bits are automatically inferred by SmartGen when the dynamic PLL core is generated; however, <71:73> (STATASEL, STATBSEL, STATCSEL) and <77:79> (DYNASEL, DYNBSEL, DYNCSEL) depend on the input clock source of the corresponding CCC. Users must first run Layout in Designer to determine the exact setting for these ports. After Layout is complete, generate the "CCC_Configuration" report by choosing **Tools** > **Reports** > **CCC_Configuration** in the Designer software. Refer to "PLL Configuration Bits Description" on page 90 for descriptions of the PLL configuration bits. For simulation purposes, bits <71:73> and <78:80> are "don't care." Therefore, it is strongly suggested that SmartGen be used to generate the correct configuration bit settings for the dynamic PLL core.

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                          : dyn_pll_hardio
Family                        : ProASIC3E
Output Format                 : VERILOG
Type                          : Dynamic CCC
Input Freq(MHz)               : 30.000
CLKA Source                   : Hardwired I/O
Feedback Delay Value Index    : 1
Feedback Mux Select           : 1
XDLY Mux Select               : No
Primary Freq(MHz)             : 33.000
Primary PhaseShift            : 0
Primary Delay Value Index     : 1
Primary Mux Select            : 4
Secondary1 Freq(MHz)          : 40.000
Use GLB                       : YES
Use YB                        : NO
GLB Delay Value Index         : 1
YB Delay Value Index          : 1
Secondary1 PhaseShift         : 0
Secondary1 Mux Select         : 0
Secondary1 Input Freq(MHz)    : 40.000
CLKB Source                   : Hardwired I/O
Secondary2 Freq(MHz)          : 50.000
Use GLC                       : YES
Use YC                        : NO
GLC Delay Value Index         : 1
YC Delay Value Index          : 1
Secondary2 PhaseShift         : 0
Secondary2 Mux Select         : 0
Secondary2 Input Freq(MHz)    : 50.000
CLKC Source                   : Hardwired I/O

Configuration Bits:
FINDIV[6:0]     0000101
FBDIV[6:0]      0100000
OADIV[4:0]      00100
OBDIV[4:0]      00000
OCDIV[4:0]      00000
OAMUX[2:0]      100
OBMUX[2:0]      000
OCMUX[2:0]      000
FBSEL[1:0]      01
FBDLY[4:0]      00000
XDLYSEL         0
DLYGLA[4:0]     00000
DLYGLB[4:0]     00000
```

FlashROM Generation and Instantiation in the Design

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

1. Static Fixed Data—Enables you to fix the data so it cannot be changed during programming time. This option is useful when you have fixed data stored in this region, which is required for the operation of the design in the FPGA. Key storage is one example.

2. Static Modifiable Data—Select this option when the data in a particular region is expected to be static data (such as a version number, which remains the same for a long duration but could conceivably change in the future). This option enables you to avoid changing the value every time you enter new data.

3. Read from File—This provides the full flexibility of FlashROM usage to the customer. If you have a customized algorithm for generating the FlashROM data, you can specify this setting. You can then generate a text file with data for as many devices as you wish to program, and load that into the FlashPoint programming file generation software to get programming files that include all the data. SmartGen will optionally pass the location of the file where the data is stored if the file is specified in SmartGen. Each text file has only one type of data format (binary, decimal, hex, or ASCII text). The length of each data file must be shorter than or equal to the selected region length. If the data is shorter than the selected region length, the most significant bits will be padded with 0s. For multiple text files for multiple regions, the first lines are for the first device. In SmartGen, **Load Sim. Value From File** allows you to load the first device data in the MEM file for simulation.

4. Auto Increment/Decrement—This scenario is useful when you specify the contents of FlashROM for a large number of devices in a series. You can specify the step value for the serial number and a maximum value for inventory control. During programming file generation, the actual number of devices to be programmed is specified and a start value is fed to the software.

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

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.
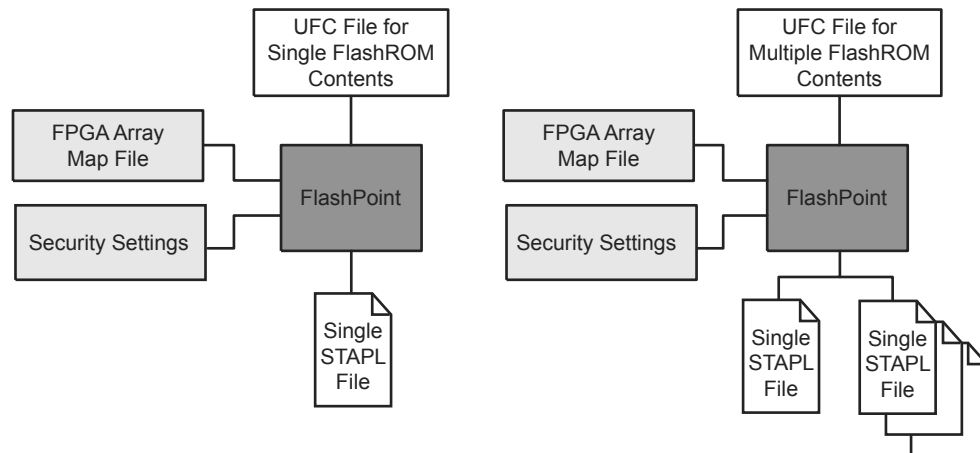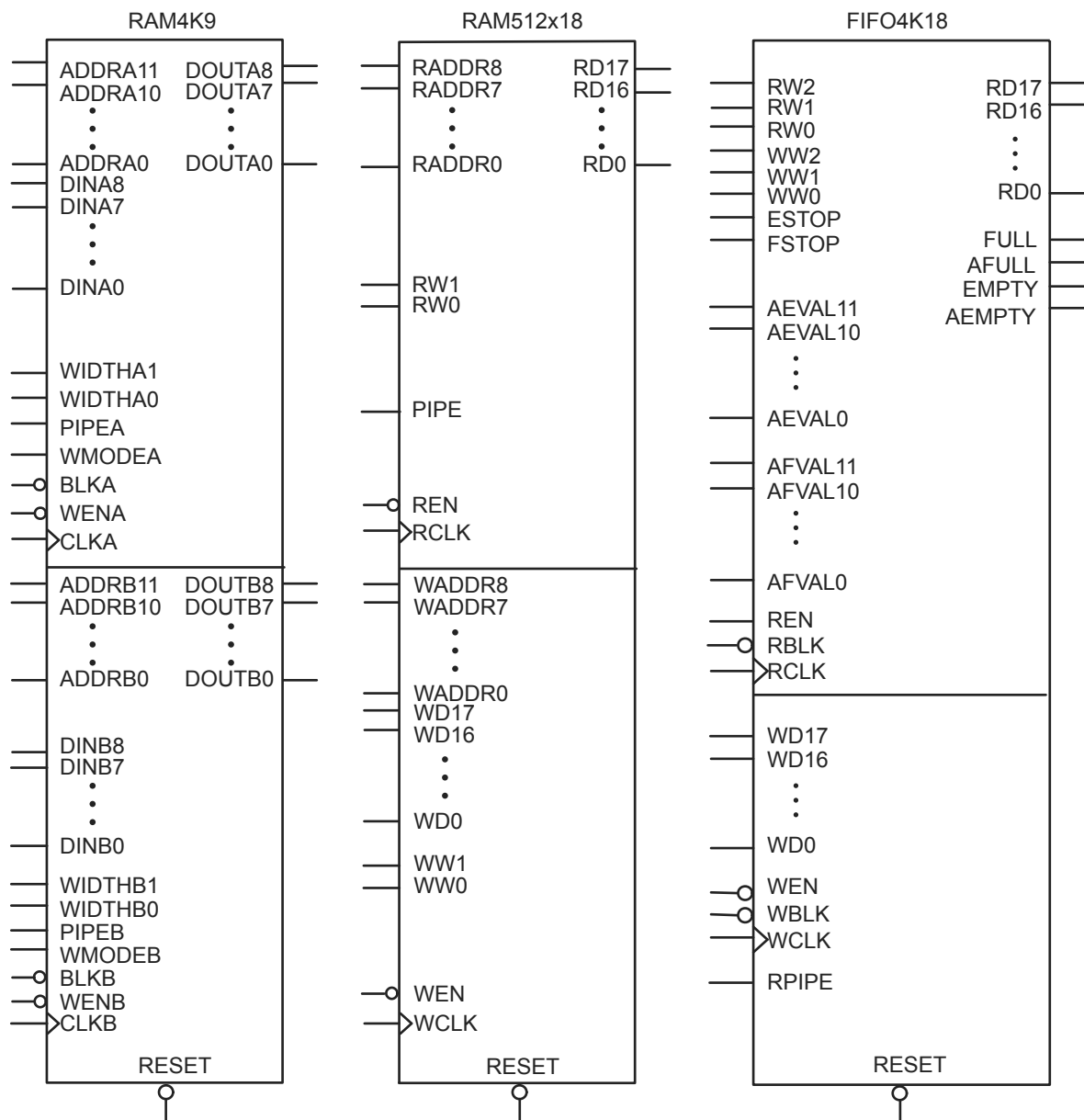

Figure 5-11 • **Single or Multiple Programming File Generation**

RAM4K9	RAM512x18	FIFO4K18
ADDRA11 DOUTA8 ADDRA10 DOUTA7 ⋮ ⋮ ADDRA0 DOUTA0 DINA8 DINA7 ⋮ DINA0 WIDTHA1 WIDTHA0 PIPEA WMODEA BLKA WENA CLKA ADDRB11 DOUTB8 ADDRB10 DOUTB7 ⋮ ⋮ ADDRB0 DOUTB0 DINB8 DINB7 ⋮ DINB0 WIDTHB1 WIDTHB0 PIPEB WMODEB BLKB WENB CLKB RESET	RADDR8 RD17 RADDR7 RD16 ⋮ ⋮ RADDR0 RD0 RW1 RW0 PIPE REN RCLK WADDR8 WADDR7 ⋮ WADDR0 WD17 WD16 ⋮ WD0 WW1 WW0 WEN WCLK RESET	RW2 RD17 RW1 RD16 RW0 ⋮ WW2 RD0 WW1 WW0 FULL ESTOP AFULL FSTOP EMPTY AEMPTY AEVAL11 AEVAL10 ⋮ AEVAL0 AFVAL11 AFVAL10 ⋮ AFVAL0 REN RBLK RCLK WD17 WD16 ⋮ WD0 WEN WBLK WCLK RPIPE RESET

Notes:

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

2. *For FIFO4K18, the same clock 180° out of phase (inverted) between clock pins should be used.*

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

The ROM emulation application is based on RAM block initialization. If the user's main design has access only to the read ports of the RAM block (RADDR, RD, RCLK, and REN), and the contents of the RAM are already initialized through the TAP, then the memory blocks will emulate ROM functionality for the core design. In this case, the write ports of the RAM blocks are accessed only by the user interface block, and the interface is activated only by the TAP Instruction Register contents.

Users should note that the contents of the RAM blocks are lost in the absence of applied power. However, the 1 kbit of flash memory, FlashROM, in low power flash devices can be used to retain data after power is removed from the device. Refer to the "SRAM and FIFO Memories in Microsemi's Low Power Flash Devices" section on page 131 for more information.

Sample Verilog Code

Interface Block

```
`define Initialize_start 8'h22 //INITIALIZATION START COMMAND VALUE
`define Initialize_stop 8'h23 //INITIALIZATION START COMMAND VALUE

module interface(IR, rst_n, data_shift, clk_in, data_update, din_ser, dout_ser, test,
  test_out,test_clk,clk_out,wr_en,rd_en,write_word,read_word,rd_addr, wr_addr);

input [7:0] IR;
input [3:0] read_word; //RAM DATA READ BACK
input rst_n, data_shift, clk_in, data_update, din_ser; //INITIALIZATION SIGNALS
input test, test_clk; //TEST PROCEDURE CLOCK AND COMMAND INPUT
output [3:0] test_out; //READ DATA
output [3:0] write_word; //WRITE DATA
output [1:0] rd_addr; //READ ADDRESS
output [1:0] wr_addr; //WRITE ADDRESS
output dout_ser; //TDO DRIVER
output clk_out, wr_en, rd_en;

wire [3:0] write_word;
wire [1:0] rd_addr;
wire [1:0] wr_addr;
wire [3:0] Q_out;
wire enable, test_active;

reg clk_out;

//SELECT CLOCK FOR INITIALIZATION OR READBACK TEST
always @(enable or test_clk or data_update)
begin
  case ({test_active})
    1 : clk_out = test_clk ;
    0 : clk_out = !data_update;
    default : clk_out = 1'b1;
  endcase
end

assign test_active = test && (IR == 8'h23);
assign enable = (IR == 8'h22);
assign wr_en = !enable;
assign rd_en = !test_active;
assign test_out = read_word;
assign dout_ser = Q_out[3];

//4-bit SIN/POUT SHIFT REGISTER
shift_reg data_shift_reg (.Shiften(data_shift), .Shiftin(din_ser), .Clock(clk_in),
  .Q(Q_out));

//4-bit PIPELINE REGISTER
D_pipeline pipeline_reg (.Data(Q_out), .Clock(data_update), .Q(write_word));
```

nano Standard I/Os

Table 7-2 and Table 7-3 show the voltages and compatible I/O standards for ProASIC3 nano and IGLOO nano devices.

I/Os provide programmable slew rates, drive strengths, and weak pull-up and pull-down circuits. Selectable Schmitt trigger and 5 V tolerant receivers are offered. See the "5 V Input Tolerance" section on page 171 for possible implementations of 5 V tolerance.

All I/Os are in a known state during power-up, and any power-up sequence is allowed without current impact. Refer to the "I/O Power-Up and Supply Voltage Thresholds for Power-On Reset (Commercial and Industrial)" section in the datasheet for more information. During power-up, before reaching activation levels, the I/O input and output buffers are disabled while the weak pull-up is enabled. Activation levels are described in the datasheet.

Table 7-2 • **Supported I/O Standards for IGLOO nano Devices**

IGLOO nano	AGLN010	AGLN015	AGLN020	AGLN060	AGLN125	AGLN250
Single-Ended						
LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V / 1.8 V / 1.5 V / 1.2 V	✓	✓	✓	✓	✓	✓

Table 7-3 • **Supported I/O Standards for ProASIC3 Devices**

ProASIC3 nano	A3PN010	A3PN015	A3PN020	A3PN060	A3PN125	A3PN250
Single-Ended						
LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V / 1.8 V / 1.5 V	✓	✓	✓	✓	✓	✓

I/O Banks and I/O Standards Compatibility

I/Os are grouped into I/O voltage banks. nano devices have two, three, or four I/O banks.

Each I/O voltage bank has dedicated I/O supply and ground voltages. This isolation is necessary to minimize simultaneous switching noise from the input and output (SSI and SSO). The switching noise (ground bounce and power bounce) is generated by the output buffers and transferred into input buffer circuits, and vice versa. Because of these dedicated supplies, only I/Os with compatible standards can be assigned to the same I/O voltage bank. Table 7-4 shows the required voltage compatibility values for each of these voltages.

There are four I/O banks on the 250 k gate device.

There are three I/O banks on the 15 k and 20 k gate devices.

There are two I/O banks on the 10 k, 60 k, and 125 k gate devices.

I/O standards are compatible if their VCCI values are identical. For more information about I/O and global assignments to I/O banks in a device, refer to the specific pin table for the device in the packaging section of the datasheet and the "User I/O Naming Convention" section on page 178.

Table 7-4 • **VCCI Voltages and Compatible Standards**

VCCI (typical)	Compatible Standards
3.3 V	LVTTL/LVCMOS 3.3
2.5 V	LVCMOS 2.5
1.8 V	LVCMOS 1.8
1.5 V	LVCMOS 1.5
1.2 V	LVCMOS 1.2

Features Supported on Every I/O

Table 7-5 lists all features supported by transmitter/receiver for single-ended I/Os. Table 7-6 lists the performance of each I/O technology.

Table 7-5 • **I/O Features**

Feature	Description
All I/O	• High performance (Table 7-6) • Electrostatic discharge (ESD) protection • I/O register combining option
Single-Ended Transmitter Features	• Hot-swap • I/Os can be configured to behave in Flash*Freeze mode as tristate, HIGH, LOW, or to hold the previous state (not supported on ProASIC3 nano devices). • Programmable output slew rate: high and low • Optional weak pull-up and pull-down resistors • Output drive: 2 drive strengths (except for LVCMOS 1.2 V) • LVTTL/LVCMOS 3.3 V outputs compatible with 5 V TTL inputs
Single-Ended Receiver Features	• Selectable Schmitt trigger • 5 V–input–tolerant receiver (Table 7-12 on page 171) • Separate ground plane for GNDQ pin and power plane for V_{CCI} pin are used for input buffer to reduce output-induced noise.
DDR	• DDR is supported for 60 k gate devices and above.

Table 7-6 • **Maximum I/O Frequency**

Specification	Maximum Performance		
	ProASIC3 nano 1.5 V DC Core Supply Voltage	**IGLOO nano V2 or V5 Devices, 1.5 V DC Core Supply Voltage**	**IGLOO nano V2, 1.2 V DC Core Supply Voltage**
LVTTL/LVCMOS 3.3 V	200 MHz	180 MHz	TBD
LVCMOS 2.5 V	250 MHz	230 MHz	TBD
LVCMOS 1.8 V	200 MHz	180 MHz	TBD
LVCMOS 1.5 V	130 MHz	120 MHz	TBD
LVCMOS 1.2 V	Not supported	TBD	TBD

Flash FPGAs I/O Support

The flash FPGAs listed in Table 8-1 support I/Os and the functions described in this document.

Table 8-1 • **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 8-1. 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 8-1. 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*.
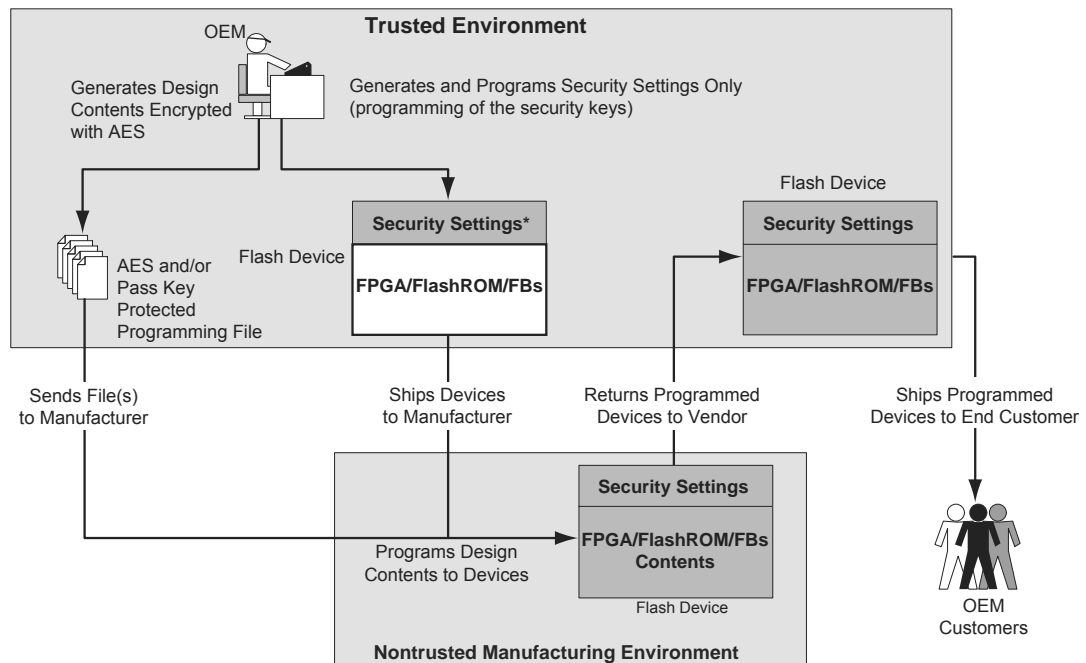
Application 1: Trusted Environment

As illustrated in Figure 11-7, this application allows the programming of devices at design locations where research and development take place. Therefore, encryption is not necessary and is optional to the user. This is often a secure way to protect the design, since the design program files are not sent elsewhere. In situations where production programming is not available at the design location, programming centers (such as Microsemi In-House Programming) provide a way of programming designs at an alternative, secure, and trusted location. In this scenario, the user generates a STAPL programming file from the Designer software in plaintext format, containing information on the entire design or the portion of the design to be programmed. The user can choose to employ the FlashLock Pass Key feature with the design. Once the design is programmed to unprogrammed devices, the design is protected by this FlashLock Pass Key. If no future programming is needed, the user can consider permanently securing the IGLOO and ProASIC3 device, as discussed in the "Permanent FlashLock" section on page 241.

Application 2: Nontrusted Environment—Unsecured Location

Often, programming of devices is not performed in the same location as actual design implementation, to reduce manufacturing cost. Overseas programming centers and contract manufacturers are examples of this scenario.

To achieve security in this case, the AES key and the FlashLock Pass Key can be initially programmed in-house (trusted environment). This is done by generating a programming file with only the security settings and no design contents. The design FPGA core, FlashROM, and (for Fusion) FB contents are generated in a separate programming file. This programming file must be set with the same AES key that was used to program to the device previously so the device will correctly decrypt this encrypted programming file. As a result, the encrypted design content programming file can be safely sent off-site to nontrusted programming locations for design programming. Figure 11-7 shows a more detailed flow for this application.

Notes:

1. Programmed portion indicated with dark gray.

2. Programming of FBs applies to Fusion only.

Figure 11-7 • Application 2: Device Programming in a Nontrusted Environment

Programming File Header Definition

In each STAPL programming file generated, there will be information about how the AES key and FlashLock Pass Key are configured. Table 11-8 shows the header definitions in STAPL programming files for different security levels.

Table 11-8 • **STAPL Programming File Header Definitions by Security Level**

Security Level	STAPL File Header Definition
No security (no FlashLock Pass Key or AES key)	`NOTE "SECURITY" "Disable";`
FlashLock Pass Key with no AES key	`NOTE "SECURITY" "KEYED ";`
FlashLock Pass Key with AES key	`NOTE "SECURITY" "KEYED ENCRYPT ";`
Permanent Security Settings option enabled	`NOTE "SECURITY" "PERMLOCK ENCRYPT ";`
AES-encrypted FPGA array (for programming updates)	`NOTE "SECURITY" "ENCRYPT CORE ";`
AES-encrypted FlashROM (for programming updates)	`NOTE "SECURITY" "ENCRYPT FROM ";`
AES-encrypted FPGA array and FlashROM (for programming updates)	`NOTE "SECURITY" "ENCRYPT FROM CORE ";`

Example File Headers

STAPL Files Generated with FlashLock Key and AES Key Containing Key Information

- FlashLock Key / AES key indicated in STAPL file header definition
- Intended ONLY for secured/trusted environment programming applications

```
==============================================
NOTE "CREATOR" "Designer Version: 6.1.1.108";
NOTE "DEVICE" "A3PE600";
NOTE "PACKAGE" "208 PQFP";
NOTE "DATE" "2005/04/08";
NOTE "STAPL_VERSION" "JESD71";
NOTE "IDCODE" "$123261CF";
NOTE "DESIGN" "counter32";
NOTE "CHECKSUM" "$EDB9";
NOTE "SAVE_DATA" "FRomStream";
NOTE "SECURITY" "KEYED ENCRYPT ";
NOTE "ALG_VERSION" "1";
NOTE "MAX_FREQ" "20000000";
NOTE "SILSIG" "$00000000";
NOTE "PASS_KEY" "$00123456789012345678901234567890";
NOTE "AES_KEY" "$ABCDEFABCDEFABCDEFABCDEFABCDEFAB";
==============================================
```

Microsemi's Flash Families Support Voltage Switching Circuit

The flash FPGAs listed in Table 13-1 support the voltage switching circuit feature and the functions described in this document.

Table 13-1 • **Flash-Based FPGAs Supporting Voltage Switching Circuit**

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	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

*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 13-1. 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 13-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.
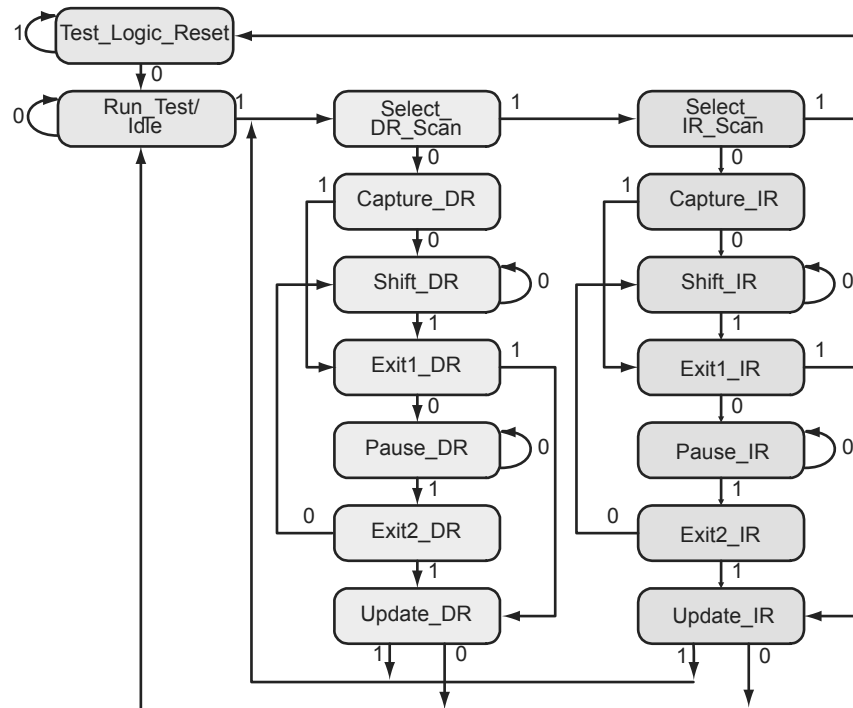
Figure 16-4 • **TAP Controller State Diagram**

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.

I/O Behavior at Power-Up/-Down

This section discusses the behavior of device I/Os, used and unused, during power-up/-down of V_{CC} and V_{CCI}. As mentioned earlier, VMVx and V_{CCI}Bx are tied together, and therefore, inputs and outputs are powered up/down at the same time.

I/O State during Power-Up/-Down

This section discusses the characteristics of I/O behavior during device power-up and power-down. Before the start of power-up, all I/Os are in tristate mode. The I/Os will remain tristated during power-up until the last voltage supply (VCC or VCCI) is powered to its functional level (power supply functional levels are discussed in the "Power-Up to Functional Time" section on page 312). After the last supply reaches the functional level, the outputs will exit the tristate mode and drive the logic at the input of the output buffer. Similarly, the input buffers will pass the external logic into the FPGA fabric once the last supply reaches the functional level. The behavior of user I/Os is independent of the VCC and VCCI sequence or the state of other voltage supplies of the FPGA (VPUMP and VJTAG). Figure 17-2 shows the output buffer driving HIGH and its behavior during power-up with 10 kΩ external pull-down. In Figure 17-2, VCC is powered first, and VCCI is powered 5 ms after VCC. Figure 17-3 on page 312 shows the state of the I/O when VCCI is powered about 5 ms before VCC. In the circuitry shown in Figure 17-3 on page 312, the output is externally pulled down.

During power-down, device I/Os become tristated once the first power supply (VCC or VCCI) drops below its brownout voltage level. The I/O behavior during power-down is also independent of voltage supply sequencing.

Figure 17-2 • **I/O State when VCC Is Powered before VCCI**