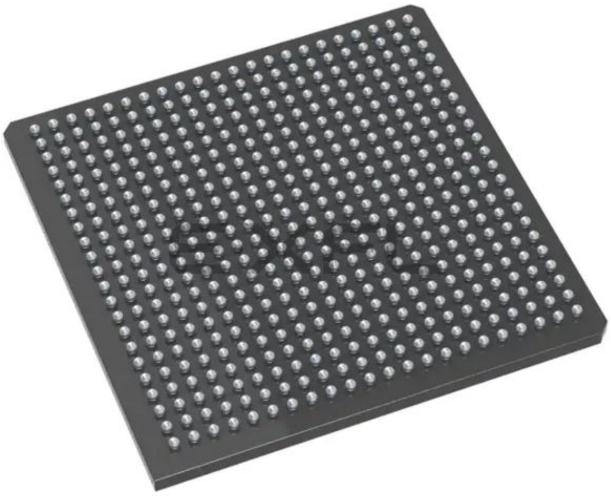

Welcome to E-XFL.COM

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	147456
Number of I/O	300
Number of Gates	1000000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	484-BGA
Supplier Device Package	484-FPBGA (23x23)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/a3p1000l-fg484i

- There will be added skew and clock insertion delay due to the clock gating circuit. The user should analyze external setup/hold times carefully. The user should also ensure the additional skew across the clock gating filter circuit is accounted for in any paths where the launch register is driven from the filter input clock and captured by a register driven by the gated clock filter output clock.

Power Analysis

SmartPower identifies static and dynamic power consumption problems quickly within a design. It provides a hierarchical view, allowing users to drill down and estimate the power consumption of individual components or events. SmartPower analyzes power consumption for nets, gates, I/Os, memories, clocks, cores, clock domains, power supply rails, peak power during a clock cycle, and switching transitions.

SmartPower generates detailed hierarchical reports of the dynamic power consumption of a design for easy inspection. These reports include design-level power summary, average switching activity, and ambient and junction temperature readings. Enter the target clock and data frequencies for a design, and let SmartPower perform a detailed and accurate power analysis. SmartPower supports importing files in the VCD (Value-Change Dump) format as specified in the IEEE 1364 standard. It also supports the Synopsys® Switching Activity Interchange Format (SAIF) standard. Support for these formats lets designers generate switching activity information in a variety of simulators and then import this information directly into SmartPower.

For portable or battery-operated applications, a power profile feature enables you to measure power and battery life, based on a sequence of operational modes of the design. In most portable and battery-operated applications, the system is seldom fully "on" 100 percent of the time. "On" is a combination of fully active, standby, sleep, or other functional modes. SmartPower allows users to create a power profile for a design by specifying operational modes and the percent of time the device will run in each of the modes. Power is calculated for each of the modes, and total power is calculated based on the weighted average of all modes.

SmartPower also provides an estimated battery life based on the power profile. The current capacity for a given battery is entered and used to estimate the life of the battery. The result is an accurate and realistic indication of battery life.

More information on SmartPower can be found on the Microsemi SoC Products Group website:
http://www.microsemi.com/soc/products/software/libero/smartpower.aspx.

Additional Power Conservation Techniques

IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 FPGAs provide many ways to inherently conserve power; however, there are also several design techniques that can be used to reduce power on the board.

- Microsemi recommends that the designer use the minimum number of I/O banks possible and tie any unused power supplies (such as V_{CCPLL}, V_{CCI}, VMV, and V_{PUMP}) to ground.
- Leave unused I/O ports floating. Unused I/Os are configured by the software as follows:
  - Output buffer is disabled (with tristate value of high impedance)
  - Input buffer is disabled (with tristate value of high impedance)
- Use the lowest available voltage I/O standard, the lowest drive strength, and the slowest slew rate to reduce I/O switching contribution to power consumption.
- Advanced and pro I/O banks may consume slightly higher static current than standard and standard plus banks—avoid using advanced and pro banks whenever practical.
  - The small static power benefit obtained by avoiding advanced or pro I/O banks is usually negligible compared to the benefit of using a low power I/O standard.
- Deselect RAM blocks that are not being used.
- Only enable read and write ports on RAM blocks when they are needed.
- Gating clocks LOW offers improved static power of RAM blocks.
- Drive the FF port of RAM blocks with the Flash_Freeze_Enabled signal from the Flash*Freeze management IP.
- Drive inputs to the full voltage level so that all transistors are turned on or off completely.

YB and YC are identical to GLB and GLC, respectively, with the exception of a higher selectable final output delay. The SmartGen PLL Wizard will configure these outputs according to user specifications and can enable these signals with or without the enabling of Global Output Clocks.

The above signals can be enabled in the following output groupings in both internal and external feedback configurations of the static PLL:

- One output – GLA only
- Two outputs – GLA + (GLB and/or YB)
- Three outputs – GLA + (GLB and/or YB) + (GLC and/or YC)

PLL Macro Block Diagram

As illustrated, the PLL supports three distinct output frequencies from a given input clock. Two of these (GLB and GLC) can be routed to the B and C global network access, respectively, and/or routed to the device core (YB and YC).

There are five delay elements to support phase control on all five outputs (GLA, GLB, GLC, YB, and YC).

There are delay elements in the feedback loop that can be used to advance the clock relative to the reference clock.

The PLL macro reference clock can be driven in the following ways:

1. By an INBUF* macro to create a composite macro, where the I/O macro drives the global buffer (with programmable delay) using a hardwired connection. In this case, the I/O must be placed in one of the dedicated global I/O locations.
2. Directly from the FPGA core.
3. From an I/O that is routed through the FPGA regular routing fabric. In this case, users must instantiate a special macro, PLLINT, to differentiate from the hardwired I/O connection described earlier.

During power-up, the PLL outputs will toggle around the maximum frequency of the voltage-controlled oscillator (VCO) gear selected. Toggle frequencies can range from 40 MHz to 250 MHz. This will continue as long as the clock input (CLKA) is constant (HIGH or LOW). This can be prevented by LOW assertion of the POWERDOWN signal.

The visual PLL configuration in SmartGen, a component of the Libero SoC and Designer tools, will derive the necessary internal divider ratios based on the input frequency and desired output frequencies selected by the user.

CCC Locations

CCCs located in the middle of the east and west sides of the device access the three VersaNet global networks on each side (six total networks), while the four CCCs located in the four corners access three quadrant global networks (twelve total networks). See Figure 4-13.

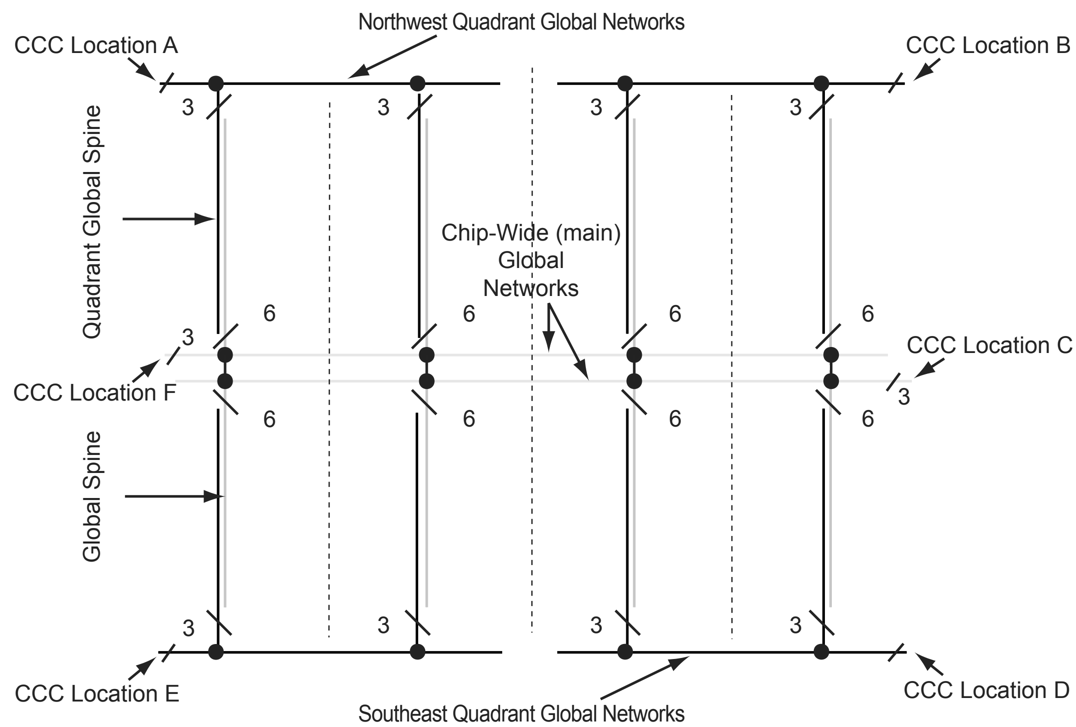

Figure 4-13 **• Global Network Architecture for 60 k Gate Devices and Above**

The following explains the locations of the CCCs in IGLOO and ProASIC3 devices:

In Figure 4-15 on page 98 through Figure 4-16 on page 98, CCCs with integrated PLLs are indicated in red, and simplified CCCs are indicated in yellow. There is a letter associated with each location of the CCC, in clockwise order. The upper left corner CCC is named "A," the upper right is named "B," and so on. These names finish up at the middle left with letter "F."

```
    wire VCC, GND;

    VCC VCC_1_net(.Y(VCC));
    GND GND_1_net(.Y(GND));
    PLL Core(.CLKA(CLKA), .EXTFB(GND), .POWERDOWN(POWERDOWN),
        .GLA(GLA), .LOCK(LOCK), .GLB(), .YB(), .GLC(), .YC(),
        .OADIV0(GND), .OADIV1(GND), .OADIV2(GND), .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(VCC), .FBDIV1(GND), .FBDIV2(VCC),
        .FBDIV3(GND), .FBDIV4(GND), .FBDIV5(GND), .FBDIV6(GND),
        .FBDLY0(GND), .FBDLY1(GND), .FBDLY2(GND), .FBDLY3(GND),
        .FBDLY4(GND), .FBSEL0(VCC), .FBSEL1(GND), .XDLYSEL(GND),
        .VCOSEL0(GND), .VCOSEL1(GND), .VCOSEL2(GND));
    defparam Core.VCOFREQUENCY = 33.000;
endmodule
```

The "PLL Configuration Bits Description" section on page 106 provides descriptions of the PLL configuration bits for completeness. The configuration bits are shown as busses only for purposes of illustration. They will actually be broken up into individual pins in compilation libraries and all simulation models. For example, the FBSEL[1:0] bus will actually appear as pins FBSEL1 and FBSEL0. The setting of these select lines for the static PLL configuration is performed by the software and is completely transparent to the user.

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.

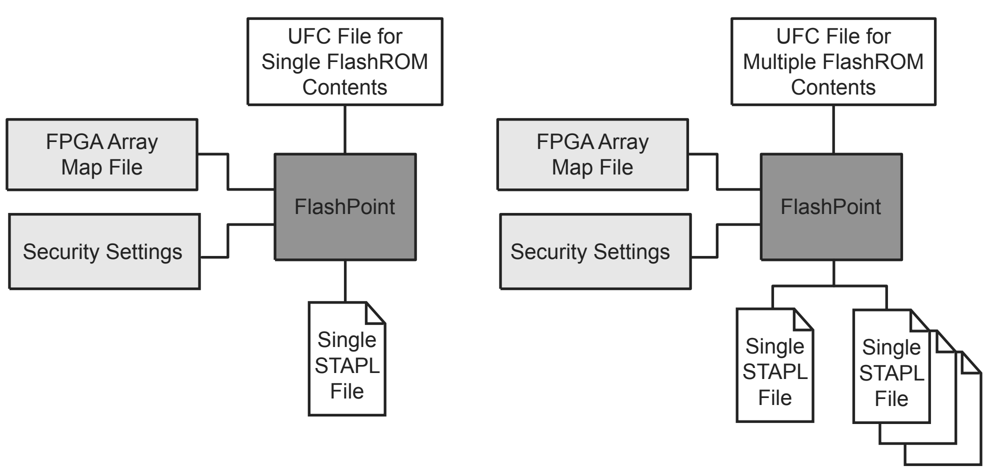

***Figure 5-11* • Single or Multiple Programming File Generation**

***Table 6-10* • RAM and FIFO Memory Block Consumption**

			Depth									
			256		512	1,024	2,048	4,096	8,192	16,384	32,768	65,536
			Two-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port	Dual-Port
Width	1	Number Block	1	1	1	1	1	1	2	4	8	16 × 1
		Configuration	Any	Any	Any	1,024 × 4	2,048 × 2	4,096 × 1	2 × (4,096 × 1) Cascade Deep	4 × (4,096 × 1) Cascade Deep	8 × (4,096 × 1) Cascade Deep	16 × (4,096 × 1) Cascade Deep
	2	Number Block	1	1	1	1	1	2	4	8	16	32
		Configuration	Any	Any	Any	1,024×4	2,048 × 2	2 × (4,096 × 1) Cascaded Wide	4 × (4,096 × 1) Cascaded 2 Deep and 2 Wide	8 × (4,096 × 1) Cascaded 4 Deep and 2 Wide	16 × (4,096 × 1) Cascaded 8 Deep and 2 Wide	32 × (4,096 × 1) Cascaded 16 Deep and 2 Wide
	4	Number Block	1	1	1	1	2	4	8	16	32	64
		Configuration	Any	Any	Any	1,024 × 4	2 × (2,048 × 2) Cascaded Wide	4 × (4,096 × 1) Cascaded Wide	4 × (4,096 × 1) Cascaded 2 Deep and 4 Wide	16 × (4,096 × 1) Cascaded 4 Deep and 4 Wide	32 × (4,096 × 1) Cascaded 8 Deep and 4 Wide	64 × (4,096 × 1) Cascaded 16 Deep and 4 Wide
	8	Number Block	1	1	1	2	4	8	16	32	64	
		Configuration	Any	Any	Any	2 × (1,024 × 4) Cascaded Wide	4 × (2,048 × 2) Cascaded Wide	8 × (4,096 × 1) Cascaded Wide	16 × (4,096 × 1) Cascaded 2 Deep and 8 Wide	32 × (4,096 × 1) Cascaded 4 Deep and 8 Wide	64 × (4,096 × 1) Cascaded 8 Deep and 8 Wide	
	9	Number Block	1	1	1	2	4	8	16	32		
		Configuration	Any	Any	Any	2 × (512 × 9) Cascaded Deep	4 × (512 × 9) Cascaded Deep	8 × (512 × 9) Cascaded Deep	16 × (512 × 9) Cascaded Deep	32 × (512 × 9) Cascaded Deep		
	16	Number Block	1	1	1	4	8	16	32	64		
		Configuration	256 × 18	256 × 18	256 × 18	4 × (1,024 × 4) Cascaded Wide	8 × (2,048 × 2) Cascaded Wide	16 × (4,096 × 1) Cascaded Wide	32 × (4,096 × 1) Cascaded 2 Deep and 16 Wide	32 × (4,096 × 1) Cascaded 4 Deep and 16 Wide		
	18	Number Block	1	2	2	4	8	18	32			
		Configuration	256 × 8	2 × (512 × 9) Cascaded Wide	2 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded 2 Deep and 2 Wide	8 × (512 × 9) Cascaded 4 Deep and 2 Wide	16 × (512 × 9) Cascaded 8 Deep and 2 Wide	16 × (512 × 9) Cascaded 16 Deep and 2 Wide			
	32	Number Block	2	4	4	8	16	32	64			
		Configuration	2 × (256 × 18) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded Wide	8 × (1,024 × 4) Cascaded Wide	16 × (2,048 × 2) Cascaded Wide	32 × (4,096 × 1) Cascaded Wide	64 × (4,096 × 1) Cascaded 2 Deep and 32 Wide			
	36	Number Block	2	4	4	8	16	32				
		Configuration	2 × (256 × 18) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded Wide	4 × (512 × 9) Cascaded 2 Deep and 4 Wide	16 × (512 × 9) Cascaded 4 Deep and 4 Wide	16 × (512 × 9) Cascaded 8 Deep and 4 Wide				
	64	Number Block	4	8	8	16	32	64				
		Configuration	4 × (256 × 18) Cascaded Wide	8 × (512 × 9) Cascaded Wide	8 × (512 × 9) Cascaded Wide	16 × (1,024 × 4) Cascaded Wide	32 × (2,048 × 2) Cascaded Wide	64 × (4,096 × 1) Cascaded Wide				
	72	Number Block	4	8	8	16	32					
		Configuration	4 × (256 × 18) Cascaded Wide	8 × (512 × 9) Cascaded Wide	8 × (512 × 9) Cascaded Wide	16 × (512 × 9) Cascaded Wide	16 × (512 × 9) Cascaded 4 Deep and 8 Wide					

Note: Memory configurations represented by grayed cells are not supported.

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 147 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));
```

Pipeline Register

```
module D_pipeline (Data, Clock, Q);

input [3:0] Data;
input Clock;
output [3:0] Q;

reg [3:0] Q;

always @ (posedge Clock) Q <= Data;

endmodule
```

4x4 RAM Block (created by SmartGen Core Generator)

```
module mem_block(DI,DO,WADDR,RADDR,WRB,RDB,WCLOCK,RCLOCK);

input [3:0] DI;
output [3:0] DO;
input [1:0] WADDR, RADDR;
input WRB, RDB, WCLOCK, RCLOCK;

wire WEBP, WEAP, VCC, GND;

VCC VCC_1_net(.Y(VCC));
GND GND_1_net(.Y(GND));
INV WEBUBBLEB(.A(WRB), .Y(WEBP));
RAM4K9 RAMBLOCK0(.ADDRA11(GND), .ADDRA10(GND), .ADDRA9(GND), .ADDRA8(GND),
  .ADDRA7(GND), .ADDRA6(GND), .ADDRA5(GND), .ADDRA4(GND), .ADDRA3(GND), .ADDRA2(GND),
  .ADDRA1(RADDR[1]), .ADDRA0(RADDR[0]), .ADDRB11(GND), .ADDRB10(GND), .ADDRB9(GND),
  .ADDRB8(GND), .ADDRB7(GND), .ADDRB6(GND), .ADDRB5(GND), .ADDRB4(GND), .ADDRB3(GND),
  .ADDRB2(GND), .ADDRB1(WADDR[1]), .ADDRB0(WADDR[0]), .DINA8(GND), .DINA7(GND),
  .DINA6(GND), .DINA5(GND), .DINA4(GND), .DINA3(GND), .DINA2(GND), .DINA1(GND),
  .DINA0(GND), .DINB8(GND), .DINB7(GND), .DINB6(GND), .DINB5(GND), .DINB4(GND),
  .DINB3(DI[3]), .DINB2(DI[2]), .DINB1(DI[1]), .DINB0(DI[0]), .WIDTHA0(GND),
  .WIDTHA1(VCC), .WIDTHB0(GND), .WIDTHB1(VCC), .PIPEA(GND), .PIPEB(GND),
  .WMODEA(GND), .WMODEB(GND), .BLKA(WEAP), .BLKB(WEBP), .WENA(VCC), .WENB(GND),
  .CLKA(RCLOCK), .CLKB(WCLOCK), .RESET(VCC), .DOUTA8(), .DOUTA7(), .DOUTA6(),
  .DOUTA5(), .DOUTA4(), .DOUTA3(DO[3]), .DOUTA2(DO[2]), .DOUTA1(DO[1]),
  .DOUTA0(DO[0]), .DOUTB8(), .DOUTB7(), .DOUTB6(), .DOUTB5(), .DOUTB4(), .DOUTB3(),
  .DOUTB2(), .DOUTB1(), .DOUTB0());
INV WEBUBBLEA(.A(RDB), .Y(WEAP));

endmodule
```

7 – I/O Structures in IGLOO and ProASIC3 Devices

Introduction

Low power flash devices feature a flexible I/O structure, supporting a range of mixed voltages (1.2 V, 1.5 V, 1.8 V, 2.5 V, and 3.3 V) through bank-selectable voltages. IGLOO,® ProASIC3®L, and ProASIC3 families support Standard, Standard Plus, and Advanced I/Os.

Users designing I/O solutions are faced with a number of implementation decisions and configuration choices that can directly impact the efficiency and effectiveness of their final design. The flexible I/O structure, supporting a wide variety of voltages and I/O standards, enables users to meet the growing challenges of their many diverse applications. Libero SoC software provides an easy way to implement I/Os that will result in robust I/O design.

This document first describes the two different I/O types in terms of the standards and features they support. It then explains the individual features and how to implement them in Libero SoC.

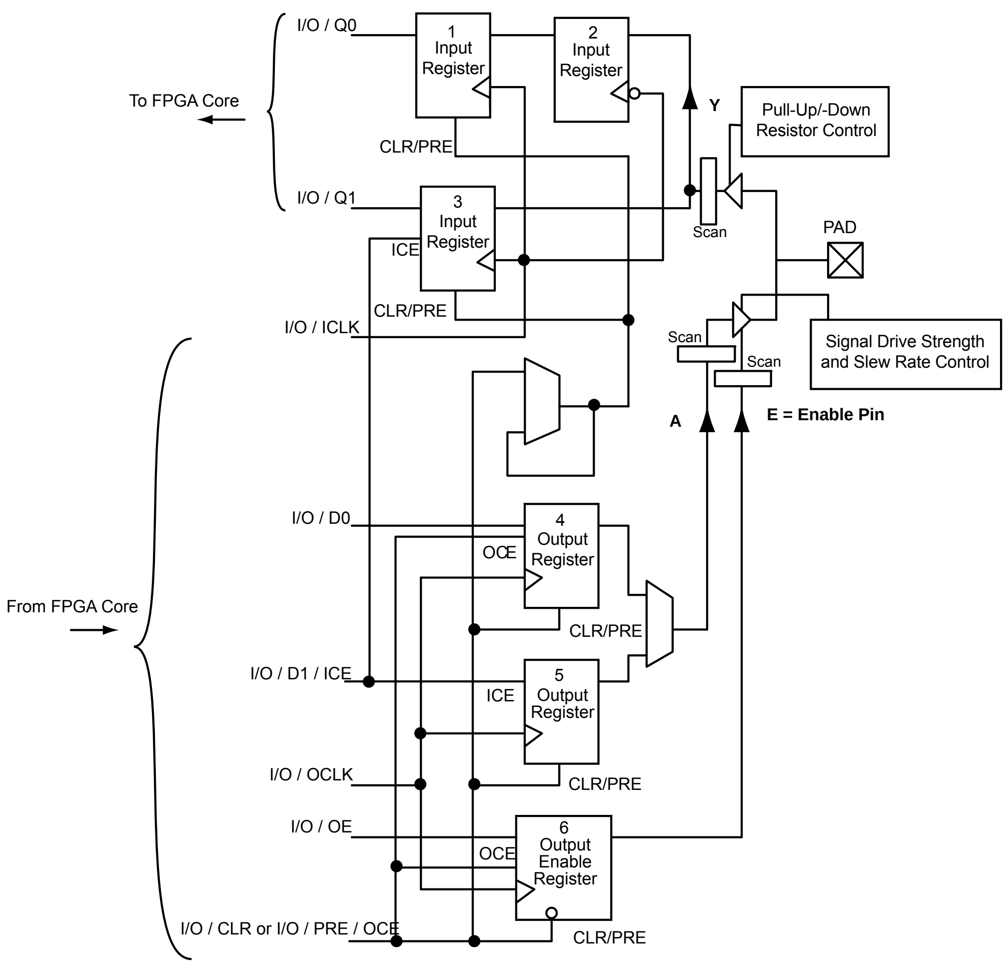

Figure 7-1 • **DDR Configured I/O Block Logical Representation**

Example: For a bus consisting of 20 equidistant loads, the terminations given in EQ 1 provide the required differential voltage, in worst-case industrial operating conditions, at the farthest receiver:

R_S = 60 Ω, R_T = 70 Ω, given Z_O = 50 Ω (2") and Z_{stub} = 50 Ω (~1.5").

EQ 1

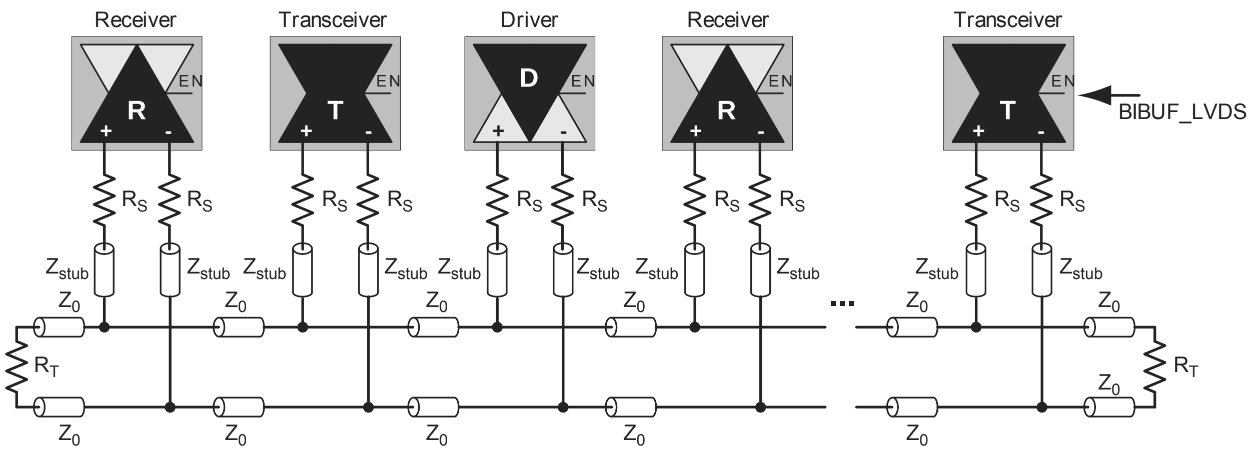

Figure 7-8 **• A B-LVDS/M-LVDS Multipoint Application Using LVDS I/O Buffers**

Design Example

Figure 10-9 shows a simple example of a design using both DDR input and DDR output registers. The user can copy the HDL code in Libero SoC software and go through the design flow. Figure 10-10 and Figure 10-11 on page 283 show the netlist and ChipPlanner views of the ddr_test design. Diagrams may vary slightly for different families.

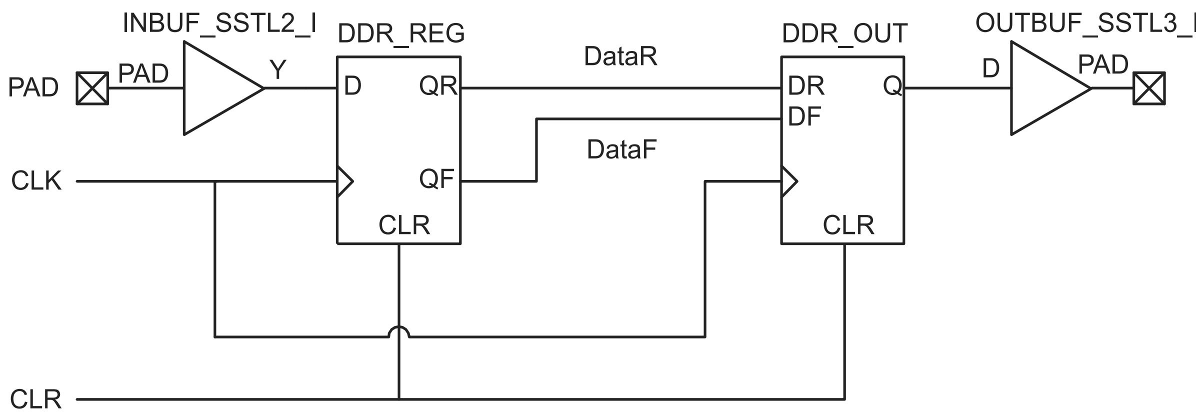

Figure 10-9 • **Design Example**

Figure 10-10 • **DDR Test Design as Seen by NetlistViewer for IGLOO/e Devices**

```
module ddr_test(DIN, CLK, CLR, DOUT);

input  DIN, CLK, CLR;
output DOUT;

  Inbuf_ddr Inbuf_ddr (.PAD(DIN), .CLR(clr), .CLK(clk), .QR(qr), .QF(qf));
  Outbuf_ddr Outbuf_ddr (.DataR(qr),.DataF(qf), .CLR(clr), .CLK(clk),.PAD(DOUT));

  INBUF INBUF_CLR (.PAD(CLR), .Y(clr));
  INBUF INBUF_CLK (.PAD(CLK), .Y(clk));

endmodule
```

Simulation Consideration

Microsemi DDR simulation models use inertial delay modeling by default (versus transport delay modeling). As such, pulses that are shorter than the actual gate delays should be avoided, as they will not be seen by the simulator and may be an issue in post-routed simulations. The user must be aware of the default delay modeling and must set the correct delay model in the simulator as needed.

Conclusion

Fusion, IGLOO, and ProASIC3 devices support a wide range of DDR applications with different I/O standards and include built-in DDR macros. The powerful capabilities provided by SmartGen and its GUI can simplify the process of including DDR macros in designs and minimize design errors. Additional considerations should be taken into account by the designer in design floorplanning and placement of I/O flip-flops to minimize datapath skew and to help improve system timing margins. Other system-related issues to consider include PLL and clock partitioning.

List of Changes

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

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
v1.4 (December 2008)	IGLOO nano and ProASIC3 nano devices were added to Table 10-1 • Flash-Based FPGAs.	272
	The "I/O Cell Architecture" section was updated with information applicable to nano devices.	273
	The output buffer (OUTBUF_SSTL3_I) input was changed to D, instead of Q, in Figure 10-1 • DDR Support in Low Power Flash Devices, Figure 10-3 • DDR Output Register (SSTL3 Class I), Figure 10-6 • DDR Output Register (SSTL3 Class I), Figure 10-7 • DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTL), and the output from the DDR_OUT macro was connected to the input of the TRIBUFF macro in Figure 10-7 • DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTL).	271, 275, 278, 279
v1.3 (October 2008)	The "Double Data Rate (DDR) Architecture" section was updated to include mention of the AFS600 and AFS1500 devices.	271
	The "DDR Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	272
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 10-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	272
v1.1 (March 2008)	The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new.	272

Programming Solutions

Details for the available programmers can be found in the programmer user's guides listed in the "Related Documents" section on page 297.

All the programmers except FlashPro4, FlashPro3, FlashPro Lite, and FlashPro require adapter modules, which are designed to support device packages. All modules are listed on the Microsemi SoC Products Group website at http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx. They are not listed in this document, since this list is updated frequently with new package options and any upgrades required to improve programming yield or support new families.

***Table 11-3* • Programming Solutions**

Programmer	Vendor	ISP	Single Device	Multi-Device	Availability
FlashPro4	Microsemi	Only	Yes	Yes[1]	Available
FlashPro3	Microsemi	Only	Yes	Yes[1]	Available
FlashPro Lite[2]	Microsemi	Only	Yes	Yes[1]	Available
FlashPro	Microsemi	Only	Yes	Yes[1]	Discontinued
Silicon Sculptor 3	Microsemi	Yes[3]	Yes	Cascade option (up to two)	Available
Silicon Sculptor II	Microsemi	Yes[3]	Yes	Cascade option (up to two)	Available
Silicon Sculptor	Microsemi	Yes	Yes	Cascade option (up to four)	Discontinued
Sculptor 6X	Microsemi	No	Yes	Yes	Discontinued
BP MicroProgrammers	BP Microsystems	No	Yes	Yes	Contact BP Microsystems at www.bpmicro.com

Notes:

1. *Multiple devices can be connected in the same JTAG chain for programming.*
2. *If FlashPro Lite is used for programming, the programmer derives all of its power from the target pc board's VDD supply. The FlashPro Lite's VPP and VPN power supplies use the target pc board's VDD as a power source. The target pc board must supply power to both the VDDP and VDD power pins of the ProASIC^{PLUS} device in addition to supplying VDD to the FlashPro Lite. The target pc board needs to provide at least 500 mA of current to the FlashPro Lite VDD connection for programming.*
3. *Silicon Sculptor II and Silicon Sculptor 3 can only provide ISP for ProASIC and ProASIC^{PLUS} families, not for Fusion, IGLOO, or ProASIC3 devices.*

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.

STAPL File with AES Encryption

- Does not contain AES key / FlashLock Key information
- Intended for transmission through web or service to unsecured locations for programming

```
=============================================
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" "$123261CF";
NOTE "DESIGN" "counter32";
NOTE "CHECKSUM" "$EF57";
NOTE "SAVE_DATA" "FRomStream";
NOTE "SECURITY" "ENCRYPT FROM CORE ";
NOTE "ALG_VERSION" "1";
NOTE "MAX_FREQ" "20000000";
NOTE "SILSIG" "$00000000";
```

Conclusion

The new and enhanced security features offered in Fusion, IGLOO, and ProASIC3 devices provide state-of-the-art security to designs programmed into these flash-based devices. Microsemi low power flash devices employ the encryption standard used by NIST and the U.S. government—AES using the 128-bit Rijndael algorithm.

The combination of an on-chip AES decryption engine and FlashLock technology provides the highest level of security against invasive attacks and design theft, implementing the most robust and secure ISP solution. These security features protect IP within the FPGA and protect the system from cloning, wholesale “black box” copying of a design, invasive attacks, and explicit IP or data theft.

Glossary

Term	Explanation
Security Header programming file	Programming file used to program the FlashLock Pass Key and/or AES key into the device to secure the FPGA, FlashROM, and/or FBs.
AES (encryption) key	128-bit key defined by the user when the AES encryption option is set in the Microsemi Designer software when generating the programming file.
FlashLock Pass Key	128-bit key defined by the user when the FlashLock option is set in the Microsemi Designer software when generating the programming file. The FlashLock Key protects the security settings programmed to the device. Once a device is programmed with FlashLock, whatever settings were chosen at that time are secure.
FlashLock	The combined security features that protect the device content from attacks. These features are the following: • Flash technology that does not require an external bitstream to program the device • FlashLock Pass Key that secures device content by locking the security settings and preventing access to the device as defined by the user • AES key that allows secure, encrypted device reprogrammability

References

National Institute of Standards and Technology. “ADVANCED ENCRYPTION STANDARD (AES) Questions and Answers.” 28 January 2002 (10 January 2005).
See http://csrc.nist.gov/archive/aes/index1.html for more information.

List of Changes

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

Date	Changes	Page
August 2012	This chapter will now be published standalone as an application note in addition to being part of the IGLOO/ProASIC3/Fusion FPGA fabric user's guides (SAR 38769).	N/A
	The "ISP Programming Header Information" section was revised to update the description of FP3-10PIN-ADAPTER-KIT in Table 13-3 • Programming Header Ordering Codes, clarifying that it is the adapter kit used for ProASIC$^{\underline{\text{PLUS}}}$ based boards, and also for ProASIC3 based boards where a compact programming header is being used (SAR 36779).	335
June 2011	The VPUMP programming mode voltage was corrected in Table 13-2 • Power Supplies. The correct value is 3.15 V to 3.45 V (SAR 30668).	329
	The notes associated with Figure 13-5 • Programming Header (top view) and Figure 13-6 • Board Layout and Programming Header Top View were revised to make clear the fact that IGLOO nano V2 devices can be programmed at 1.2 V (SAR 30787).	335, 337
	Figure 13-6 • Board Layout and Programming Header Top View was revised to include resistors tying TCK and TRST to GND. Microsemi recommends tying off TCK and TRST to GND if JTAG is not used (SAR 22921). RT ProASIC3 was added to the list of device families.	337
	In the "ISP Programming Header Information" section, the kit for adapting ProASIC$^{\underline{\text{PLUS}}}$ devices was changed from FP3-10PIN-ADAPTER-KIT to FP3-26PIN-ADAPTER-KIT (SAR 20878).	335
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	References to FlashPro4 and FlashPro3X were added to this chapter, giving distinctions between them. References to SmartGen were deleted and replaced with Libero IDE Catalog.	N/A
	The "ISP Architecture" section was revised to indicate that V2 devices can be programmed at 1.2 V VCC with FlashPro4.	327
	SmartFusion was added to Table 13-1 • Flash-Based FPGAs Supporting ISP.	328
	The "Programming Voltage (VPUMP) and VJTAG" section was revised and 1.2 V was added to Table 13-2 • Power Supplies.	329
	The "Nonvolatile Memory (NVM) Programming Voltage" section is new.	329
	Cortex-M3 was added to the "Cortex-M1 and Cortex-M3 Device Security" section.	331
	In the "ISP Programming Header Information" section, the additional header adapter ordering number was changed from FP3-26PIN-ADAPTER to FP3-10PIN-ADAPTER-KIT, which contains 26-pin migration capability.	335
	The description of NC was updated in Figure 13-5 • Programming Header (top view), Table 13-4 • Programming Header Pin Numbers and Description and Figure 13-6 • Board Layout and Programming Header Top View.	335, 336
	The "Symptoms of a Signal Integrity Problem" section was revised to add that customers are expected to troubleshoot board-level signal integrity issues by measuring voltages and taking scope plots. "FlashPro4/3/3X allows TCK to be lowered from 6 MHz down to 1 MHz to allow you to address some signal integrity problems" formerly read, "from 24 MHz down to 1 MHz." "The Scan Chain command expects to see 0x2" was changed to 0x1.	337

16 – Boundary Scan in Low Power Flash Devices

Boundary Scan

Low power flash devices are compatible with IEEE Standard 1149.1, which defines a hardware architecture and the set of mechanisms for boundary scan testing. JTAG operations are used during boundary scan testing.

The basic boundary scan logic circuit is composed of the TAP controller, test data registers, and instruction register (Figure 16-2 on page 360).

Low power flash devices support three types of test data registers: bypass, device identification, and boundary scan. The bypass register is selected when no other register needs to be accessed in a device. This speeds up test data transfer to other devices in a test data path. The 32-bit device identification register is a shift register with four fields (LSB, ID number, part number, and version). The boundary scan register observes and controls the state of each I/O pin. Each I/O cell has three boundary scan register cells, each with serial-in, serial-out, parallel-in, and parallel-out pins.

TAP Controller State Machine

The TAP controller is a 4-bit state machine (16 states) that operates as shown in Figure 16-1.

The 1s and 0s represent the values that must be present on TMS at a rising edge of TCK for the given state transition to occur. IR and DR indicate that the instruction register or the data register is operating in that state.

The TAP controller receives two control inputs (TMS and TCK) and generates control and clock signals for the rest of the test logic architecture. On power-up, the TAP controller enters the Test-Logic-Reset state. To guarantee a reset of the controller from any of the possible states, TMS must remain HIGH for five TCK cycles. The TRST pin can also be used to asynchronously place the TAP controller in the Test-Logic-Reset state.

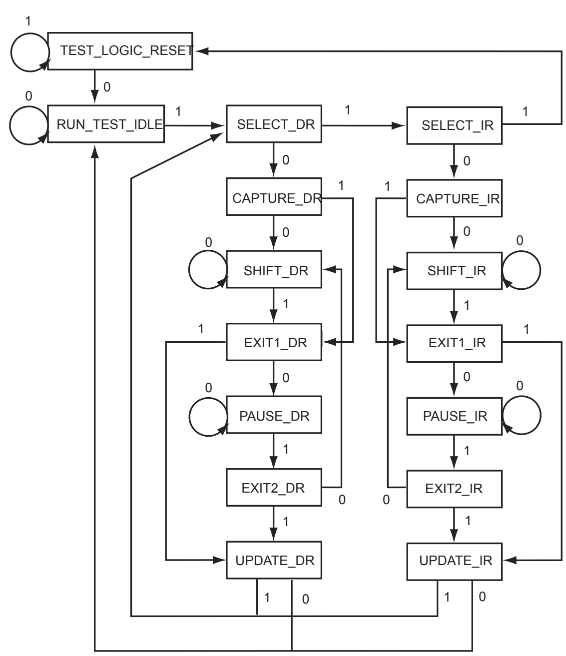

Figure 16-1 • **TAP Controller State Machine**