

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	34
Number of Gates	30000
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/a3pn030-zqng48

17	Power-Up/-Down Behavior of Low Power Flash Devices	307
	Introduction	307
	Flash Devices Support Power-Up Behavior	308
	Power-Up/-Down Sequence and Transient Current	309
	I/O Behavior at Power-Up/-Down	311
	Cold-Sparing	316
	Hot-Swapping	317
	Conclusion	317
	Related Documents	318
	List of Changes	318
A	Summary of Changes	319
	History of Revision to Chapters	319
B	Product Support	321
	Customer Service	321
	Customer Technical Support Center	321
	Technical Support	321
	Website	321
	Contacting the Customer Technical Support Center	321
	ITAR Technical Support	322
	Index	323

Table 3-3 • Quadrant Global Pin Name (continued)

Differential I/O Pairs	GAAO/IOuxwByVz GAA1/IOuxwByVz	The output of the different pair will drive the global.
	GABO/IOuxwByVz GAB1/IOuxwByVz	The output of the different pair will drive the global.
	GACO/IOuxwByVz GAC1/IOuxwByVz	The output of the different pair will drive the global.
	GBAO/IOuxwByVz GBA1/IOuxwByVz	The output of the different pair will drive the global.
	GBBO/IOuxwByVz GBB1/IOuxwByVz	The output of the different pair will drive the global.
	GBCO/IOuxwByVz GBC1/IOuxwByVz	The output of the different pair will drive the global.
	GDAO/IOuxwByVz GDA1/IOuxwByVz	The output of the different pair will drive the global.
	GDBO/IOuxwByVz GDB1/IOuxwByVz	The output of the different pair will drive the global.
	GDCO/IOuxwByVz GDC1/IOuxwByVz	The output of the different pair will drive the global.
	GEAO/IOuxwByVz GEA1/IOuxwByVz	The output of the different pair will drive the global.
	GEB0/IOuxwByVz GEB1/IOuxwByVz	The output of the different pair will drive the global.
	GECO/IOuxwByVz GEC1/IOuxwByVz	The output of the different pair will drive the global.

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

Unused Global I/O Configuration

The unused clock inputs behave similarly to the unused Pro I/Os. The Microsemi Designer software automatically configures the unused global pins as inputs with pull-up resistors if they are not used as regular I/O.

I/O Banks and Global I/O Standards

In low power flash devices, any I/O or internal logic can be used to drive the global network. However, only the global macro placed at the global pins will use the hardwired connection between the I/O and global network. Global signal (signal driving a global macro) assignment to I/O banks is no different from regular I/O assignment to I/O banks with the exception that you are limited to the pin placement location available. Only global signals compatible with both the VCCI and VREF standards can be assigned to the same bank.

Spine Architecture

The low power flash device architecture allows the VersaNet global networks to be segmented. Each of these networks contains spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside its region. The nine spines available in a vertical column reside in global networks with two separate regions of scope: the quadrant global network, which has three spines, and the chip (main) global network, which has six spines. Note that the number of quadrant globals and globals/spines per tree varies depending on the specific device. Refer to Table 3-4 for the clocking resources available for each device. The spines are the vertical branches of the global network tree, shown in Figure 3-3 on page 34. Each spine in a vertical column of a chip (main) global network is further divided into two spine segments of equal lengths: one in the top and one in the bottom half of the die (except in 10 k through 30 k gate devices).

Top and bottom spine segments radiating from the center of a device have the same height. However, just as in the ProASIC^{PLUS} family, signals assigned only to the top and bottom spine cannot access the middle two rows of the die. The spines for quadrant clock networks do not cross the middle of the die and cannot access the middle two rows of the architecture.

Each spine and its associated ribs cover a certain area of the device (the "scope" of the spine; see Figure 3-3 on page 34). Each spine is accessed by the dedicated global network MUX tree architecture, which defines how a particular spine is driven—either by the signal on the global network from a CCC, for example, or by another net defined by the user. Details of the chip (main) global network spine-selection MUX are presented in Figure 3-8 on page 44. The spine drivers for each spine are located in the middle of the die.

Quadrant spines can be driven from user I/Os or an internal signal from the north and south sides of the die. The ability to drive spines in the quadrant global networks can have a significant effect on system performance for high-fanout inputs to a design. Access to the top quadrant spine regions is from the top of the die, and access to the bottom quadrant spine regions is from the bottom of the die. The A3PE3000 device has 28 clock trees and each tree has nine spines; this flexible global network architecture enables users to map up to 252 different internal/external clocks in an A3PE3000 device.

Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices

ProASIC3/ ProASIC3L Devices	IGLOO Devices	Chip Globals	Quadrant Globals (4x3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
A3PN010	AGLN010	4	0	1	0	0	260	260	4
A3PN015	AGLN015	4	0	1	0	0	384	384	6
A3PN020	AGLN020	4	0	1	0	0	520	520	6
A3PN060	AGLN060	6	12	4	9	36	384	1,536	12
A3PN125	AGLN125	6	12	8	9	72	384	3,072	12
A3PN250	AGLN250	6	12	8	9	72	768	6,144	24
A3P015	AGL015	6	0	1	9	9	384	384	12
A3P030	AGL030	6	0	2	9	18	384	768	12
A3P060	AGL060	6	12	4	9	36	384	1,536	12
A3P125	AGL125	6	12	8	9	72	384	3,072	12
A3P250/L	AGL250	6	12	8	9	72	768	6,144	24
A3P400	AGL400	6	12	12	9	108	768	9,216	24
A3P600/L	AGL600	6	12	12	9	108	1,152	13,824	36
A3P1000/L	AGL1000	6	12	16	9	144	1,536	24,576	48
A3PE600/L	AGLE600	6	12	12	9	108	1,120	13,440	35
A3PE1500		6	12	20	9	180	1,888	37,760	59
A3PE3000/L	AGLE3000	6	12	28	9	252	2,656	74,368	83

Spine Access

The physical location of each spine is identified by the letter T (top) or B (bottom) and an accompanying number (Tn or Bn). The number n indicates the horizontal location of the spine; 1 refers to the first spine on the left side of the die. Since there are six chip spines in each spine tree, there are up to six spines available for each combination of T (or B) and n (for example, six T1 spines). Similarly, there are three quadrant spines available for each combination of T (or B) and n (for example, four T1 spines), as shown in Figure 3-7.

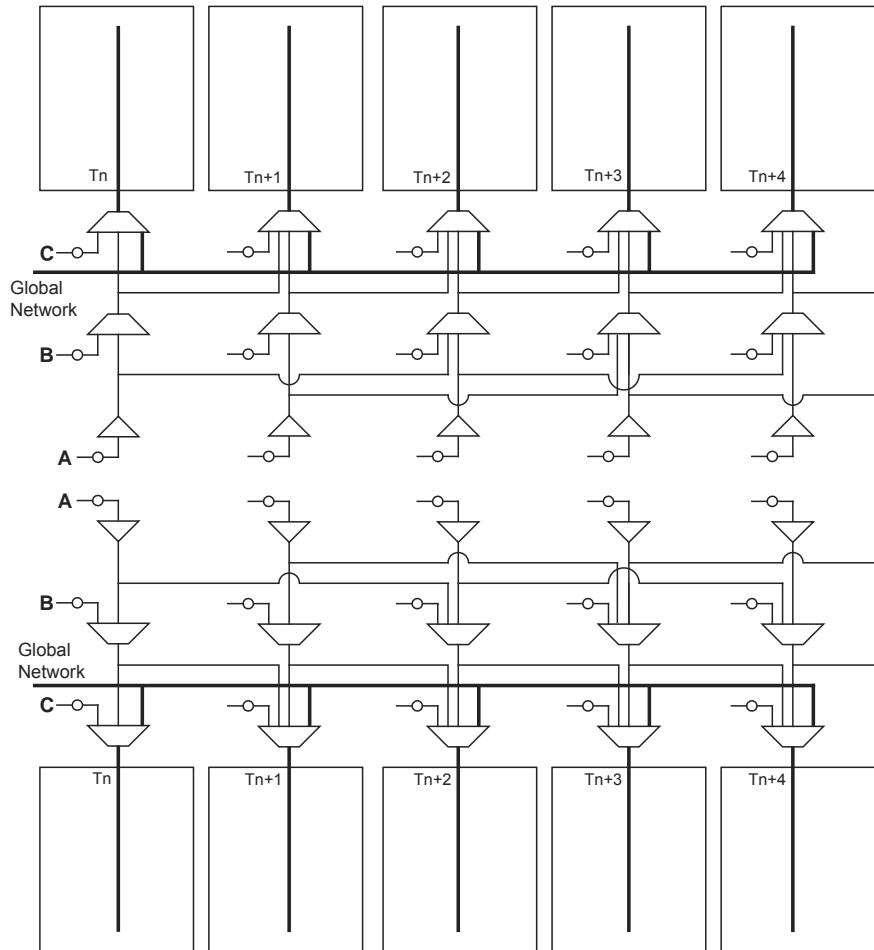


Figure 3-7 • Chip Global Aggregation

A spine is also called a local clock network, and is accessed by the dedicated global MUX architecture. These MUXes define how a particular spine is driven. Refer to Figure 3-8 on page 44 for the global MUX architecture. The MUXes for each chip global spine are located in the middle of the die. Access to the top and bottom chip global spine is available from the middle of the die. There is no control dependency between the top and bottom spines. If a top spine, T1, of a chip global network is assigned to a net, B1 is not wasted and can be used by the global clock network. The signal assigned only to the top or bottom spine cannot access the middle two rows of the architecture. However, if a spine is using the top and bottom at the same time (T1 and B1, for instance), the previous restriction is lifted.

The MUXes for each quadrant global spine are located in the north and south sides of the die. Access to the top and bottom quadrant global spines is available from the north and south sides of the die. Since the MUXes for quadrant spines are located in the north and south sides of the die, you should not try to drive T1 and B1 quadrant spines from the same signal.

Core Logic Clock Source

Core logic refers to internal routed nets. Internal routed signals access the CCC via the FPGA Core Fabric. Similar to the External I/O option, whenever the clock source comes internally from the core itself, the routed signal is instantiated with a PLLINT macro before connecting to the CCC clock input (see Figure 4-12 for an example illustration of the connections, shown in red).

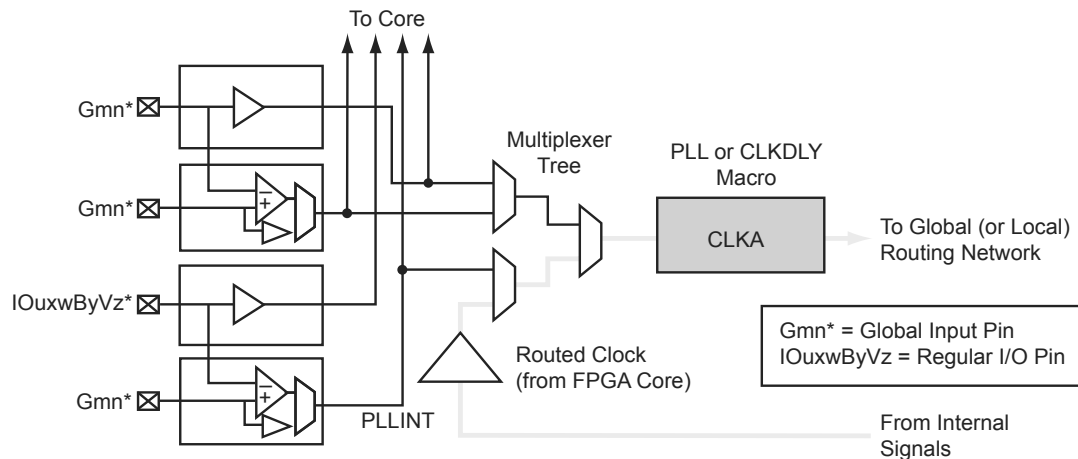


Figure 4-12 • Illustration of Core Logic Usage

For Fusion devices, the input reference clock can also be from the embedded RC oscillator and crystal oscillator. In this case, the CCC configuration is the same as the hardwired I/O clock source, and users are required to instantiate the RC oscillator or crystal oscillator macro and connect its output to the input reference clock of the CCC block.

difference will cause the VCO to increase its frequency until the output signal is phase-identical to the input after undergoing division. In other words, lock in both frequency and phase is achieved when the output frequency is M times the input. Thus, clock division in the feedback path results in multiplication at the output.

A similar argument can be made when the delay element is inserted into the feedback path. To achieve steady-state lock, the VCO output signal will be delayed by the input period less the feedback delay. For periodic signals, this is equivalent to time-advancing the output clock by the feedback delay.

Another key parameter of a PLL system is the acquisition time. Acquisition time is the amount of time it takes for the PLL to achieve lock (i.e., phase-align the feedback signal with the input reference clock). For example, suppose there is no voltage applied to the VCO, allowing it to operate at its free-running frequency. Should an input reference clock suddenly appear, a lock would be established within the maximum acquisition time.

Functional Description

This section provides detailed descriptions of PLL block functionality: clock dividers and multipliers, clock delay adjustment, phase adjustment, and dynamic PLL configuration.

Clock Dividers and Multipliers

The PLL block contains five programmable dividers. Figure 4-20 shows a simplified PLL block.

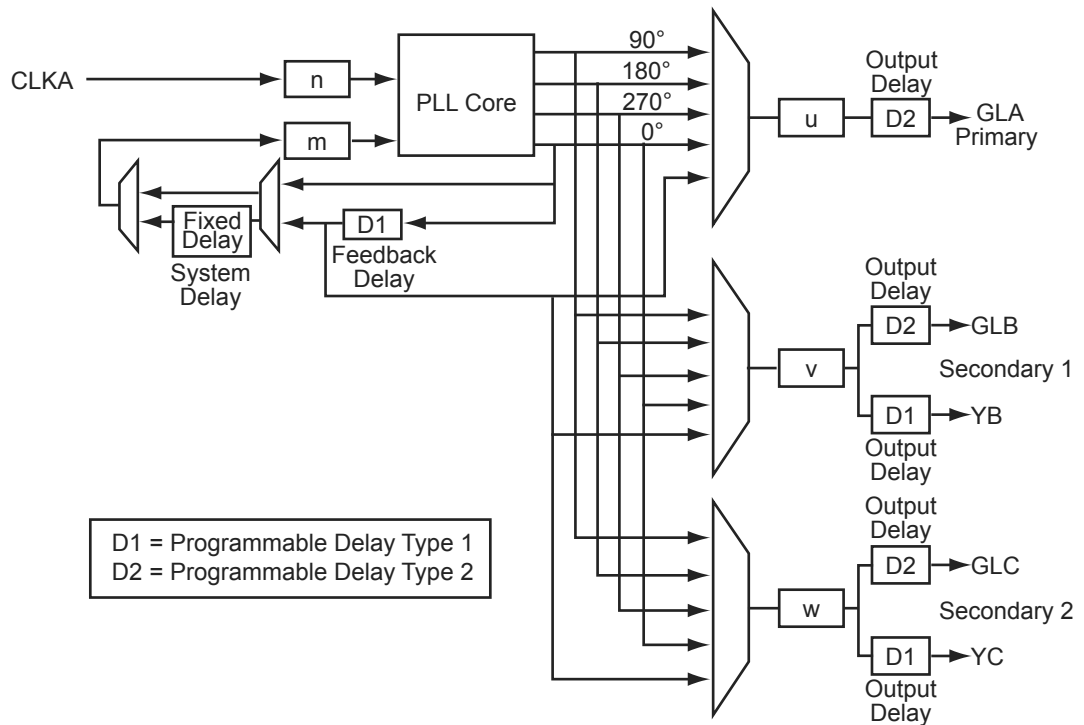


Figure 4-20 • PLL Block Diagram

Table 4-13 • 2-Bit Feedback MUX

FBSEL<1:0> State	MUX Input Selected
0	Ground. Used for power-down mode in power-down logic block.
1	PLL VCO 0° phase shift
2	PLL delayed VCO 0° phase shift
3	N/A

Table 4-14 • Programmable Delay Selection for Feedback Delay and Secondary Core Output Delays

FBDLY<4:0>; DLYYB<4:0>; DLYYC<4:0> State	Delay Value
0	Typical delay = 600 ps
1	Typical delay = 760 ps
2	Typical delay = 920 ps
⋮	⋮
31	Typical delay = 5.56 ns

Table 4-15 • Programmable Delay Selection for Global Clock Output Delays

DLYGLA<4:0>; DLYGLB<4:0>; DLYGLC<4:0> State	Delay Value
0	Typical delay = 225 ps
1	Typical delay = 760 ps
2	Typical delay = 920 ps
⋮	⋮
31	Typical delay = 5.56 ns

Table 4-16 • Fusion Dynamic CCC Clock Source Selection

RXASEL	DYNASEL	Source of CLKA
1	0	RC Oscillator
1	1	Crystal Oscillator
RXBSEL	DYNBSEL	Source of CLKB
1	0	RC Oscillator
1	1	Crystal Oscillator
RXCSEL	DYNCSEL	Source of CLKC
1	0	RC Oscillator
1	1	Crystal Oscillator

Table 4-17 • Fusion Dynamic CCC NGMUX Configuration

GLMUXCFG<1:0>	NGMUX Select Signal	Supported Input Clocks to NGMUX
00	0	GLA
	1	GLC
01	0	GLA
	1	GLINT
10	0	GLC
	1	GLINT

Programming and Accessing FlashROM

The FlashROM content can only be programmed via JTAG, but it can be read back selectively through the JTAG programming interface, the UJTAG interface, or via direct FPGA core addressing. The pages of the FlashROM can be made secure to prevent read-back via JTAG. In that case, read-back on these secured pages is only possible by the FPGA core fabric or via UJTAG.

A 7-bit address from the FPGA core defines which of the eight pages (three MSBs) is being read, and which of the 16 bytes within the selected page (four LSBs) are being read. The FlashROM content can be read on a random basis; the access time is 10 ns for a device supporting commercial specifications. The FPGA core will be powered down during writing of the FlashROM content. FPGA power-down during FlashROM programming is managed on-chip, and FPGA core functionality is not available during programming of the FlashROM. Table 5-2 summarizes various FlashROM access scenarios.

Table 5-2 • FlashROM Read/Write Capabilities by Access Mode

Access Mode	FlashROM Read	FlashROM Write
JTAG	Yes	Yes
UJTAG	Yes	No
FPGA core	Yes	No

Figure 5-6 shows the accessing of the FlashROM using the UJTAG macro. This is similar to FPGA core access, where the 7-bit address defines which of the eight pages (three MSBs) is being read and which of the 16 bytes within the selected page (four LSBs) are being read. Refer to the "UJTAG Applications in Microsemi's Low Power Flash Devices" section on page 297 for details on using the UJTAG macro to read the FlashROM.

Figure 5-7 on page 123 and Figure 5-8 on page 123 show the FlashROM access from the JTAG port. The FlashROM content can be read on a random basis. The three-bit address defines which page is being read or updated.

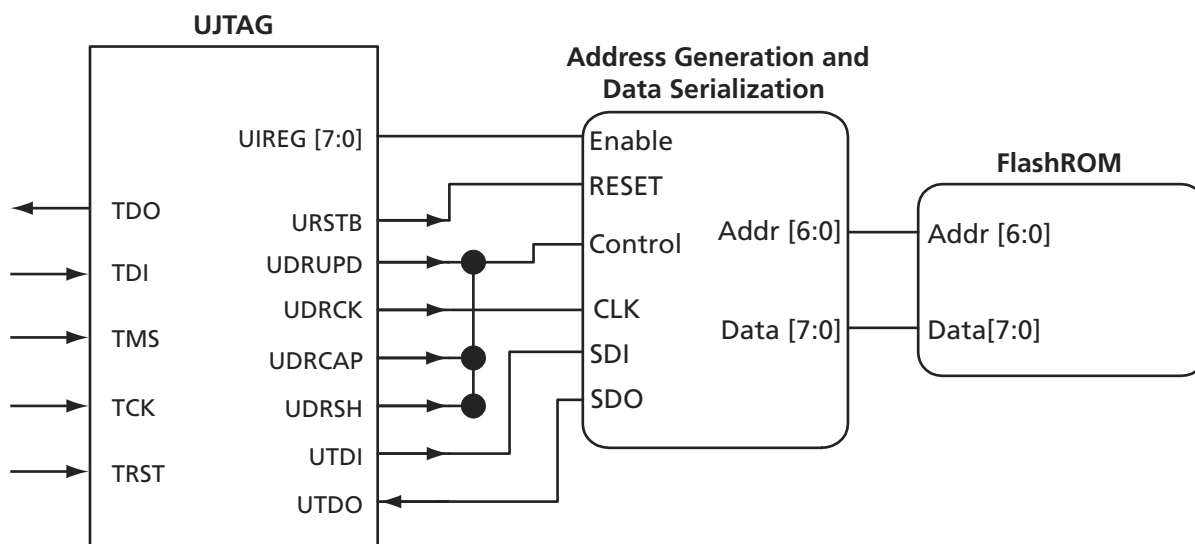


Figure 5-6 • Block Diagram of Using UJTAG to Read FlashROM Contents

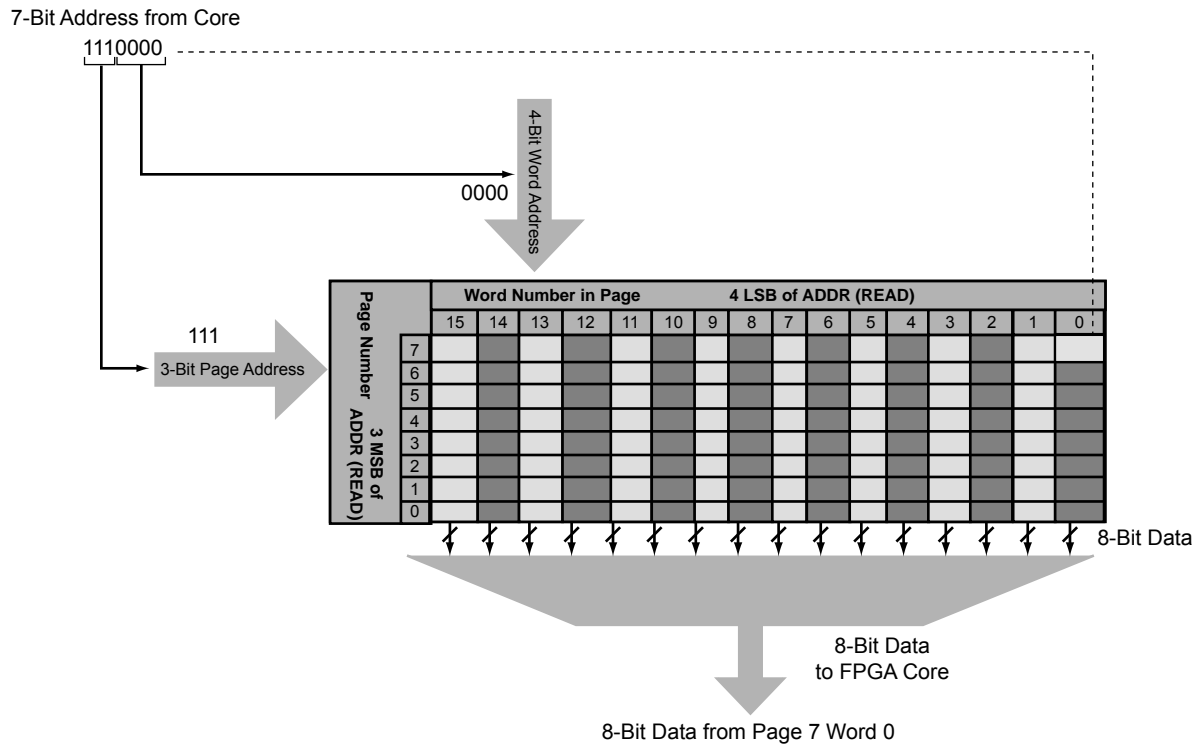


Figure 5-7 • Accessing FlashROM Using FPGA Core

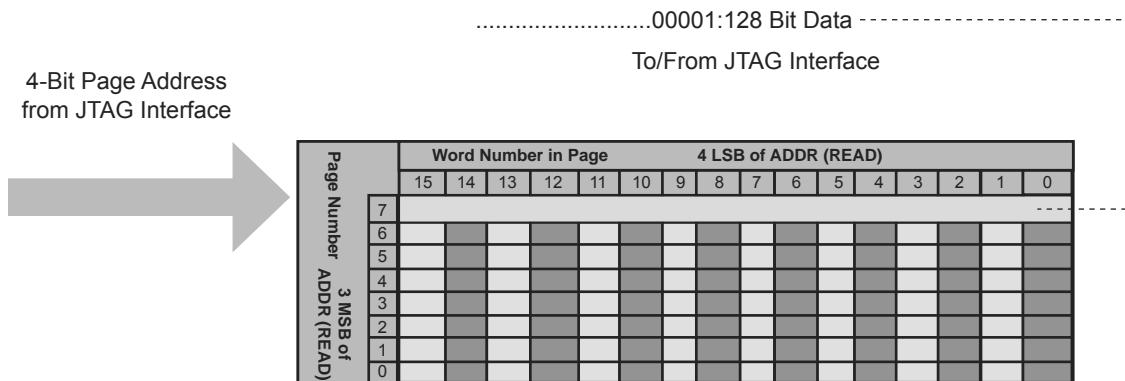


Figure 5-8 • Accessing FlashROM Using JTAG Port

SRAM and FIFO Architecture

To meet the needs of high-performance designs, the memory blocks operate strictly in synchronous mode for both read and write operations. The read and write clocks are completely independent, and each can operate at any desired frequency up to 250 MHz.

- 4k×1, 2k×2, 1k×4, 512×9 (dual-port RAM—2 read / 2 write or 1 read / 1 write)
- 512×9, 256×18 (2-port RAM—1 read / 1 write)
- Sync write, sync pipelined / nonpipelined read

Automotive ProASIC3 devices support single-port SRAM capabilities or dual-port SRAM only under specific conditions. Dual-port mode is supported if the clocks to the two SRAM ports are the same and 180° out of phase (i.e., the port A clock is the inverse of the port B clock). The Libero SoC software macro libraries support a dual-port macro only. For use of this macro as a single-port SRAM, the inputs and clock of one port should be tied off (grounded) to prevent errors during design compile. For use in dual-port mode, the same clock with an inversion between the two clock pins of the macro should be used in the design to prevent errors during compile.

The memory block includes dedicated FIFO control logic to generate internal addresses and external flag logic (FULL, EMPTY, AFULL, AEMPTY).

Simultaneous dual-port read/write and write/write operations at the same address are allowed when certain timing requirements are met.

During RAM operation, addresses are sourced by the user logic, and the FIFO controller is ignored. In FIFO mode, the internal addresses are generated by the FIFO controller and routed to the RAM array by internal MUXes.

The low power flash device architecture enables the read and write sizes of RAMs to be organized independently, allowing for bus conversion. For example, the write size can be set to 256×18 and the read size to 512×9.

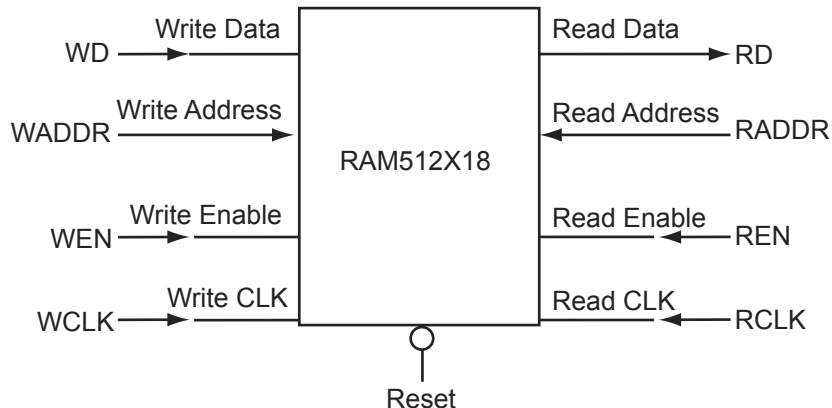
Both the write width and read width for the RAM blocks can be specified independently with the WW (write width) and RW (read width) pins. The different D×W configurations are 256×18, 512×9, 1k×4, 2k×2, and 4k×1. When widths of one, two, or four are selected, the ninth bit is unused. For example, when writing nine-bit values and reading four-bit values, only the first four bits and the second four bits of each nine-bit value are addressable for read operations. The ninth bit is not accessible.

Conversely, when writing four-bit values and reading nine-bit values, the ninth bit of a read operation will be undefined. The RAM blocks employ little-endian byte order for read and write operations.

Memory Blocks and Macros

Memory blocks can be configured with many different aspect ratios, but are generically supported in the macro libraries as one of two memory elements: RAM4K9 or RAM512X18. The RAM4K9 is configured as a true dual-port memory block, and the RAM512X18 is configured as a two-port memory block. Dual-port memory allows the RAM to both read from and write to either port independently. Two-port memory allows the RAM to read from one port and write to the other using a common clock or independent read and write clocks. If needed, the RAM4K9 blocks can be configured as two-port memory blocks. The memory block can be configured as a FIFO by combining the basic memory block with dedicated FIFO controller logic. The FIFO macro is named FIFO4KX18 (Figure 6-3 on page 136).

Clocks for the RAM blocks can be driven by the VersaNet (global resources) or by regular nets. When using local clock segments, the clock segment region that encompasses the RAM blocks can drive the RAMs. In the dual-port configuration (RAM4K9), each memory block port can be driven by either rising-edge or falling-edge clocks. Each port can be driven by clocks with different edges. Though only a rising-edge clock can drive the physical block itself, the Microsemi Designer software will automatically bubble-push the inversion to properly implement the falling-edge trigger for the RAM block.



Note: For timing diagrams of the RAM signals, refer to the appropriate family datasheet.

Figure 6-5 • 512X18 Two-Port RAM Block Diagram

Signal Descriptions for RAM512X18

RAM512X18 has slightly different behavior from RAM4K9, as it has dedicated read and write ports.

WW and RW

These signals enable the RAM to be configured in one of the two allowable aspect ratios (Table 6-5).

Table 6-5 • Aspect Ratio Settings for WW[1:0]

WW[1:0]	RW[1:0]	D×W
01	01	512×9
10	10	256×18
00, 11	00, 11	Reserved

WD and RD

These are the input and output data signals, and they are 18 bits wide. When a 512×9 aspect ratio is used for write, WD[17:9] are unused and must be grounded. If this aspect ratio is used for read, RD[17:9] are undefined.

WADDR and RADDR

These are read and write addresses, and they are nine bits wide. When the 256×18 aspect ratio is used for write or read, WADDR[8] and RADDR[8] are unused and must be grounded.

WCLK and RCLK

These signals are the write and read clocks, respectively. They can be clocked on the rising or falling edge of WCLK and RCLK.

WEN and REN

These signals are the write and read enables, respectively. They are both active-low by default. These signals can be configured as active-high.

RESET

This active-low signal resets the control logic, forces the output hold state registers to zero, disables reads and writes from the SRAM block, and clears the data hold registers when asserted. It does not reset the contents of the memory array.

While the RESET signal is active, read and write operations are disabled. As with any asynchronous reset signal, care must be taken not to assert it too close to the edges of active read and write clocks.

PIPE

This signal is used to specify pipelined read on the output. A LOW on PIPE indicates a nonpipelined read, and the data appears on the output in the same clock cycle. A HIGH indicates a pipelined read, and data appears on the output in the next clock cycle.

Table 6-8 and Table 6-9 show the maximum potential width and depth configuration for each device. Note that 15 k and 30 k gate devices do not support RAM or FIFO.

Table 6-8 • Memory Availability per IGLOO and ProASIC3 Device

Device		RAM Blocks	Maximum Potential Width ¹		Maximum Potential Depth ²	
IGLOO IGLOO nano IGLOO PLUS	ProASIC3 ProASIC3 nano ProASIC3L		Depth	Width	Depth	Width
AGL060 AGLN060 AGLP060	A3P060 A3PN060	4	256	72 (4×18)	16,384 (4,096×4)	1
AGL125 AGLN125 AGLP125	A3P125 A3PN125	8	256	144 (8×18)	32,768 (4,096×8)	1
AGL250 AGLN250	A3P250/L A3PN250	8	256	144 (8×18)	32,768 (4,096×8)	1
AGL400	A3P400	12	256	216 (12×18)	49,152 (4,096×12)	1
AGL600	A3P600/L	24	256	432 (24×18)	98,304 (4,096×24)	1
AGL1000	A3P1000/L	32	256	576 (32×18)	131,072 (4,096×32)	1
AGLE600	A3PE600	24	256	432 (24×18)	98,304 (4,096×24)	1
	A3PE1500	60	256	1,080 (60×18)	245,760 (4,096×60)	1
AGLE3000	A3PE3000/L	112	256	2,016 (112×18)	458,752 (4,096×112)	1

Notes:

1. Maximum potential width uses the two-port configuration.
2. Maximum potential depth uses the dual-port configuration.

Table 6-9 • Memory Availability per Fusion Device

Device	RAM Blocks	Maximum Potential Width ¹		Maximum Potential Depth ²	
		Depth	Width	Depth	Width
AFS090	6	256	108 (6×18)	24,576 (4,096×6)	1
AFS250	8	256	144 (8×18)	32,768 (4,096×8)	1
AFS600	24	256	432 (24×18)	98,304 (4,096×24)	1
AFS1500	60	256	1,080 (60×18)	245,760 (4,096×60)	1

Notes:

1. Maximum potential width uses the two-port configuration.
2. Maximum potential depth uses the dual-port configuration.

Example of RAM Initialization

This section of the document presents a sample design in which a 4×4 RAM block is being initialized through the JTAG port. A test feature has been implemented in the design to read back the contents of the RAM after initialization to verify the procedure.

The interface block of this example performs two major functions: initialization of the RAM block and running a test procedure to read back the contents. The clock output of the interface is either the write clock (for initialization) or the read clock (for reading back the contents). The Verilog code for the interface block is included in the "Sample Verilog Code" section on page 151.

For simulation purposes, users can declare the input ports of the UJTAG macro for easier assignment in the testbench. However, the UJTAG input ports should not be declared on the top level during synthesis. If the input ports of the UJTAG are declared during synthesis, the synthesis tool will instantiate input buffers on these ports. The input buffers on the ports will cause Compile to fail in Designer.

Figure 6-10 shows the simulation results for the initialization step of the example design.

The CLK_OUT signal, which is the clock output of the interface block, is the inverted DR_UPDATE output of the UJTAG macro. It is clear that it gives sufficient time (while the TAP Controller is in the Data Register Update state) for the write address and data to become stable before loading them into the RAM block.

Figure 6-11 presents the test procedure of the example. The data read back from the memory block matches the written data, thus verifying the design functionality.

Figure 6-10 • Simulation of Initialization Step

Figure 6-11 • Simulation of the Test Procedure of the Example

```
//
addr_counter counter_1 (.Clock(data_update), .Q(wr_addr), .Aset(rst_n),
    .Enable(enable));
addr_counter counter_2 (.Clock(test_clk), .Q(rd_addr), .Aset(rst_n),
    .Enable( test_active));

endmodule
```

Interface Block / UJTAG Wrapper

This example is a sample wrapper, which connects the interface block to the UJTAG and the memory blocks.

```
// WRAPPER
module top_init (TDI, TRSTB, TMS, TCK, TDO, test, test_clk, test_out);

input TDI, TRSTB, TMS, TCK;
output TDO;
input test, test_clk;
output [3:0] test_out;

wire [7:0] IR;
wire reset, DR_shift, DR_cap, init_clk, DR_update, data_in, data_out;
wire clk_out, wen, ren;
wire [3:0] word_in, word_out;
wire [1:0] write_addr, read_addr;

UJTAG UJTAG_U1 (.UIREG0(IR[0]), .UIREG1(IR[1]), .UIREG2(IR[2]), .UIREG3(IR[3]),
    .UIREG4(IR[4]), .UIREG5(IR[5]), .UIREG6(IR[6]), .UIREG7(IR[7]), .URSTB(reset),
    .UDRSH(DR_shift), .UDRCAP(DR_cap), .UDRCK(init_clk), .UDRUPD(DR_update),
    .UT-DI(data_in), .TDI(TDI), .TMS(TMS), .TCK(TCK), .TRSTB(TRSTB), .TDO(TDO),
    .UT-DO(data_out));
mem_block RAM_block (.DO(word_out), .RCLOCK(clk_out), .WCLOCK(clk_out), .DI(word_in),
    .WRB(wen), .RDB(ren), .WAD-DR(write_addr), .RADDR(read_addr));
interface init_block (.IR(IR), .rst_n(reset), .data_shift(DR_shift), .clk_in(init_clk),
    .data_update(DR_update), .din_ser(data_in), .dout_ser(data_out), .test(test),
    .test_out(test_out), .test_clk(test_clk), .clk_out(clk_out), .wr_en(wen),
    .rd_en(ren), .write_word(word_in), .read_word(word_out), .rd_addr(read_addr),
    .wr_addr(write_addr));

endmodule
```

Address Counter

```
module addr_counter (Clock, Q, Aset, Enable);

input Clock;
output [1:0] Q;
input Aset;
input Enable;

reg [1:0] Qaux;

always @(posedge Clock or negedge Aset)
begin
    if (!Aset) Qaux <= 2'b11;
    else if (Enable) Qaux <= Qaux + 1;
end

assign Q = Qaux;

endmodule
```

Implementing I/Os in Microsemi Software

Microsemi Libero SoC software is integrated with design entry tools such as the SmartGen macro builder, the ViewDraw schematic entry tool, and an HDL editor. It is also integrated with the synthesis and Designer tools. In this section, all necessary steps to implement the I/Os are discussed.

Design Entry

There are three ways to implement I/Os in a design:

1. Use the SmartGen macro builder to configure I/Os by generating specific I/O library macros and then instantiating them in top-level code. This is especially useful when creating I/O bus structures.
2. Use an I/O buffer cell in a schematic design.
3. Manually instantiate specific I/O macros in the top-level code.

If technology-specific macros, such as INBUF_LVCMOS33 and OUTBUF_PCI, are used in the HDL code or schematic, the user will not be able to change the I/O standard later on in Designer. If generic I/O macros are used, such as INBUF, OUTBUF, TRIBUF, CLKBUF, and BIBUF, the user can change the I/O standard using the Designer I/O Attribute Editor tool.

Using SmartGen for I/O Configuration

The SmartGen tool in Libero SoC provides a GUI-based method of configuring the I/O attributes. The user can select certain I/O attributes while configuring the I/O macro in SmartGen. The steps to configure an I/O macro with specific I/O attributes are as follows:

1. Open Libero SoC.
2. On the left-hand side of the Catalog View, select **I/O**, as shown in Figure 8-2.

Figure 8-2 • SmartGen Catalog

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

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

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

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

Conclusion

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

Related Documents

User's Guides

Libero SoC User's Guide

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

IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide

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

SmartGen Core Reference Guide

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

Input Support for DDR

The basic structure to support a DDR input is shown in Figure 9-2. Three input registers are used to capture incoming data, which is presented to the core on each rising edge of the I/O register clock. Each I/O tile supports DDR inputs.

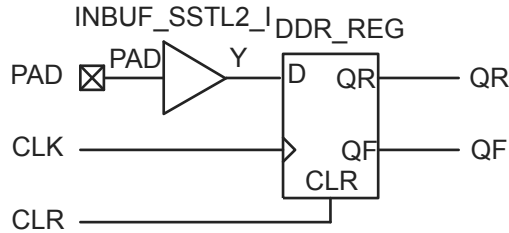


Figure 9-2 • DDR Input Register Support in Low Power Flash Devices

Output Support for DDR

The basic DDR output structure is shown in Figure 9-1 on page 205. New data is presented to the output every half clock cycle.

Note: DDR macros and I/O registers do not require additional routing. The combiner automatically recognizes the DDR macro and pushes its registers to the I/O register area at the edge of the chip. The routing delay from the I/O registers to the I/O buffers is already taken into account in the DDR macro.

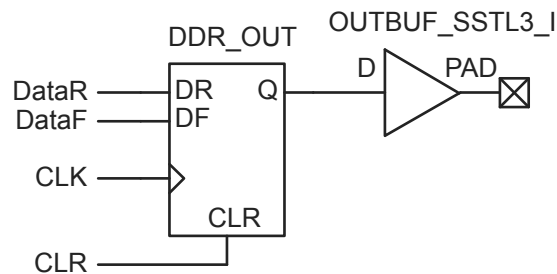


Figure 9-3 • DDR Output Register (SSTL3 Class I)

Programmer Ordering Codes

The products shown in Table 10-4 can be ordered through Microsemi sales and will be shipped directly from Microsemi. Products can also be ordered from Microsemi distributors, but will still be shipped directly from Microsemi. Table 10-4 includes ordering codes for the full kit, as well as codes for replacement items and any related hardware. Some additional products can be purchased from external suppliers for use with the programmers. Ordering codes for adapter modules used with Silicon Sculptor are available at http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx.

Table 10-4 • Programming Ordering Codes

Description	Vendor	Ordering Code	Comment
FlashPro4 ISP programmer	Microsemi	FLASHPRO 4	Uses a 2×5, RA male header connector
FlashPro Lite ISP programmer	Microsemi	FLASHPRO LITE	Supports small programming header or large header through header converter (not included)
Silicon Sculptor 3	Microsemi	SILICON-SCULPTOR 3	USB 2.0 high-speed production programmer
Silicon Sculptor II	Microsemi	SILICON-SCULPTOR II	Requires add-on adapter modules to support devices
Silicon Sculptor ISP module	Microsemi	SMPA-ISP-ACTEL-3-KIT	Ships with both large and small header support
ISP cable for small header	Microsemi	ISP-CABLE-S	Supplied with SMPA-ISP-ACTEL-3-KIT
ISP cable for large header	Microsemi	PA-ISP-CABLE	Supplied with SMPA-ISP-ACTEL-3-KIT

Programmer Device Support

Refer to www.microsemi.com/soc for the current information on programmer and device support.

Certified Programming Solutions

The Microsemi-certified programmers for flash devices are FlashPro4, FlashPro3, FlashPro Lite, FlashPro, Silicon Sculptor II, Silicon Sculptor 3, and any programmer that is built by BP Microsystems. All other programmers are considered noncertified programmers.

- FlashPro4, FlashPro3, FlashPro Lite, FlashPro

The Microsemi family of FlashPro device programmers provides in-system programming in an easy-to-use, compact system that supports all flash families. Whether programming a board containing a single device or multiple devices connected in a chain, the Microsemi line of FlashPro programmers enables fast programming and reprogramming. Programming with the FlashPro series of programmers saves board space and money as it eliminates the need for sockets on the board. There are no built-in algorithms, so there is no delay between product release and programming support. The FlashPro programmer is no longer available.

- Silicon Sculptor 3, Silicon Sculptor II

Silicon Sculptor 3 and Silicon Sculptor II are robust, compact, single-device programmers with standalone software for the PC. They are designed to enable concurrent programming of multiple units from the same PC with speeds equivalent to or faster than previous Microsemi programmers.

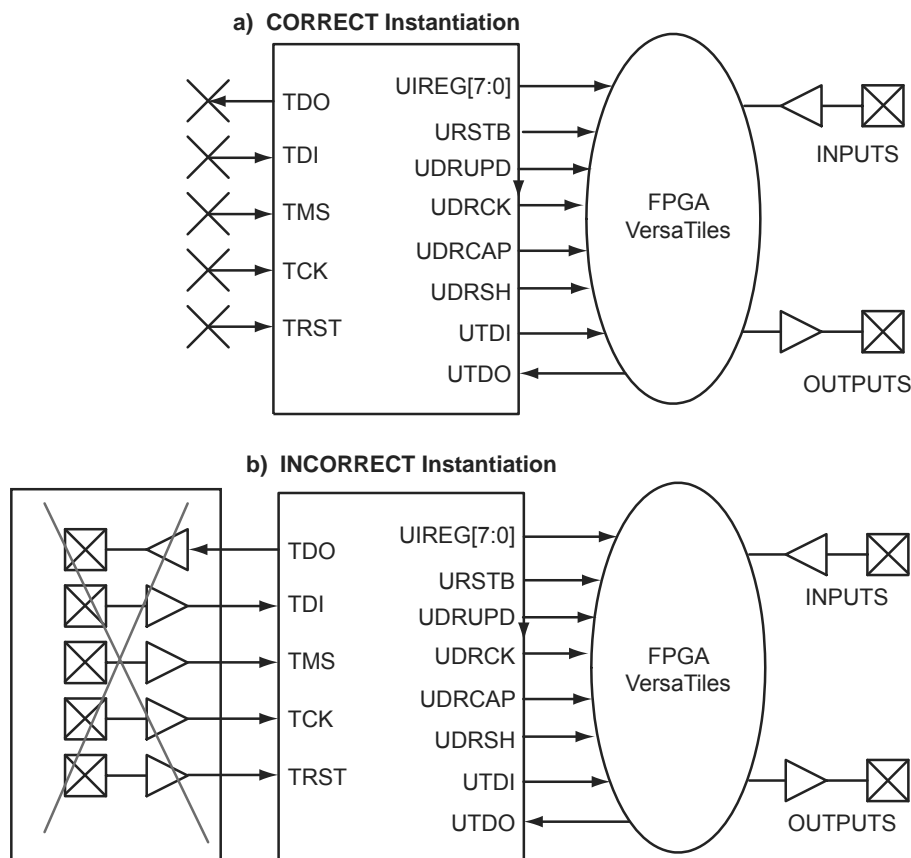
- Noncertified Programmers

Microsemi does not test programming solutions from other vendors, and DOES NOT guarantee programming yield. Also, Microsemi will not perform any failure analysis on devices programmed on non-certified programmers. Please refer to the *Programming and Functional Failure Guidelines* document 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 12-3 • Programming Header Ordering Codes, clarifying that it is the adapter kit used for ProASIC ^{PLUS} based boards, and also for ProASIC3 based boards where a compact programming header is being used (SAR 36779).	269
June 2011	The VPUMP programming mode voltage was corrected in Table 12-2 • Power Supplies. The correct value is 3.15 V to 3.45 V (SAR 30668).	263
	The notes associated with Figure 12-5 • Programming Header (top view) and Figure 12-6 • 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).	269, 271
	Figure 12-6 • 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.	271
	In the "ISP Programming Header Information" section, the kit for adapting ProASIC ^{PLUS} devices was changed from FP3-10PIN-ADAPTER-KIT to FP3-26PIN-ADAPTER-KIT (SAR 20878).	269
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.	261
	SmartFusion was added to Table 12-1 • Flash-Based FPGAs Supporting ISP.	262
	The "Programming Voltage (VPUMP) and VJTAG" section was revised and 1.2 V was added to Table 12-2 • Power Supplies.	263
	The "Nonvolatile Memory (NVM) Programming Voltage" section is new.	263
	Cortex-M3 was added to the "Cortex-M1 and Cortex-M3 Device Security" section.	265
	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.	269
	The description of NC was updated in Figure 12-5 • Programming Header (top view), Table 12-4 • Programming Header Pin Numbers and Description and Figure 12-6 • Board Layout and Programming Header Top View.	269, 270
	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.	271



Note: Do not connect JTAG pins (TDO, TDI, TMS, TCK, or TRST) to I/Os in the design.

Figure 16-3 • Connectivity Method of UJTAG Macro

UJTAG Operation

There are a few basic functions of the UJTAG macro that users must understand before designing with it. The most important fundamental concept of the UJTAG design is its connection with the TAP Controller state machine.

TAP Controller State Machine

The 16 states of the TAP Controller state machine are shown in Figure 16-4 on page 301. The 1s and 0s, shown adjacent to the state transitions, represent the TMS values that must be present at the time of a rising TCK edge for a state transition to occur. In the states that include the letters "IR," the instruction register operates; in the states that contain the letters "DR," the test data register operates. The TAP Controller receives two control inputs, TMS and TCK, and generates control and clock signals for the rest of the test logic.

On power-up (or the assertion of TRST), the TAP Controller enters the Test-Logic-Reset state. To reset the controller from any other state, TMS must be held HIGH for at least five TCK cycles. After reset, the TAP state changes at the rising edge of TCK, based on the value of TMS.