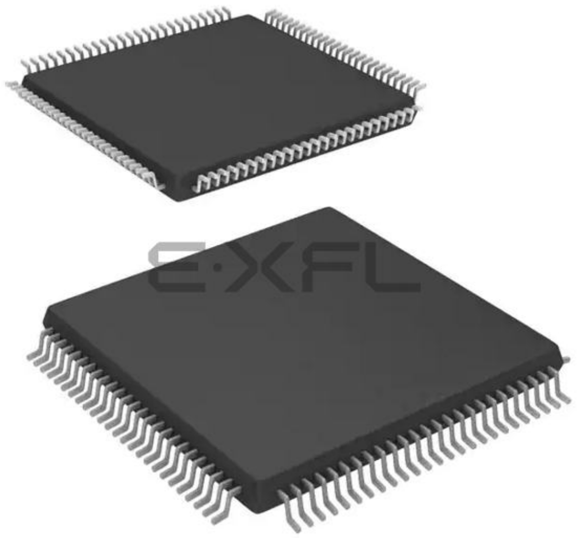
# Microchip Technology - A3PN250-ZVQ100 Datasheet



**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 | 36864 |
| Number of I/O | 68 |
| Number of Gates | 250000 |
| Voltage - Supply | 1.425V ~ 1.575V |
| Mounting Type | Surface Mount |
| Operating Temperature | -20°C ~ 85°C (TJ) |
| Package / Case | 100-TQFP |
| Supplier Device Package | 100-VQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/a3pn250-zvq100 |

Email: info@E-XFL.COM
Address: Room A, 16/F, Full Win Commercial Centre, 573 Nathan Road, Mongkok, Hong Kong

## Array Coordinates

During many place-and-route operations in the Microsemi Designer software tool, it is possible to set constraints that require array coordinates. Table 1-2 provides array coordinates of core cells and memory blocks for IGLOO and ProASIC3 devices. Table 1-3 provides the information for IGLOO PLUS devices. Table 1-4 on page 17 provides the information for IGLOO nano and ProASIC3 nano devices. The array coordinates are measured from the lower left (0, 0). They can be used in region constraints for specific logic groups/blocks, designated by a wildcard, and can contain core cells, memories, and I/Os.

I/O and cell coordinates are used for placement constraints. Two coordinate systems are needed because there is not a one-to-one correspondence between I/O cells and core cells. In addition, the I/O coordinate system changes depending on the die/package combination. It is not listed in Table 1-2. The Designer ChipPlanner tool provides the array coordinates of all I/O locations. I/O and cell coordinates are used for placement constraints. However, I/O placement is easier by package pin assignment.

Figure 1-9 on page 17 illustrates the array coordinates of a 600 k gate device. For more information on how to use array coordinates for region/placement constraints, see the *Designer User's Guide* or online help (available in the software) for software tools.

***Table 1-2*** **• IGLOO and ProASIC3 Array Coordinates**

| **Device** | | **VersaTiles** | | | | **Memory Rows** | | **Entire Die** | |
|---|---|---|---|---|---|---|---|---|---|
| | | **Min.** | | **Max.** | | **Bottom** | **Top** | **Min.** | **Max.** |
| **IGLOO** | **ProASIC3/ ProASIC3L** | **x** | **y** | **x** | **y** | **(x, y)** | **(x, y)** | **(x, y)** | **(x, y)** |
| AGL015 | A3P015 | 3 | 2 | 34 | 13 | None | None | (0, 0) | (37, 15) |
| AGL030 | A3P030 | 3 | 3 | 66 | 13 | None | None | (0, 0) | (69, 15) |
| AGL060 | A3P060 | 3 | 2 | 66 | 25 | None | (3, 26) | (0, 0) | (69, 29) |
| AGL125 | A3P125 | 3 | 2 | 130 | 25 | None | (3, 26) | (0, 0) | (133, 29) |
| AGL250 | A3P250/L | 3 | 2 | 130 | 49 | None | (3, 50) | (0, 0) | (133, 53) |
| AGL400 | A3P400 | 3 | 2 | 194 | 49 | None | (3, 50) | (0, 0) | (197, 53) |
| AGL600 | A3P600/L | 3 | 4 | 194 | 75 | (3, 2) | (3, 76) | (0, 0) | (197, 79) |
| AGL1000 | A3P1000/L | 3 | 4 | 258 | 99 | (3, 2) | (3, 100) | (0, 0) | (261, 103) |
| AGLE600 | A3PE600/L,<br>RT3PE600L | 3 | 4 | 194 | 75 | (3, 2) | (3, 76) | (0, 0) | (197, 79) |
| | A3PE1500 | 3 | 4 | 322 | 123 | (3, 2) | (3, 124) | (0, 0) | (325, 127) |
| AGLE3000 | A3PE3000/L,<br>RT3PE3000L | 3 | 6 | 450 | 173 | (3, 2)<br>or<br>(3, 4) | (3, 174)<br>or<br>(3, 176) | (0, 0) | (453, 179) |

***Table 1-3*** **• IGLOO PLUS Array Coordinates**

| **Device** | **VersaTiles** | | | | **Memory Rows** | | **Entire Die** | |
|---|---|---|---|---|---|---|---|---|
| | **Min.** | | **Max.** | | **Bottom** | **Top** | **Min.** | **Max.** |
| **IGLOO PLUS** | **x** | **y** | **x** | **y** | **(x, y)** | **(x, y)** | **(x, y)** | **(x, y)** |
| AGLP030 | 2 | 3 | 67 | 13 | None | None | (0, 0) | (69, 15) |
| AGLP060 | 2 | 2 | 67 | 25 | None | (3, 26) | (0, 0) | (69, 29) |
| AGLP125 | 2 | 2 | 131 | 25 | None | (3, 26) | (0, 0) | (133, 29) |

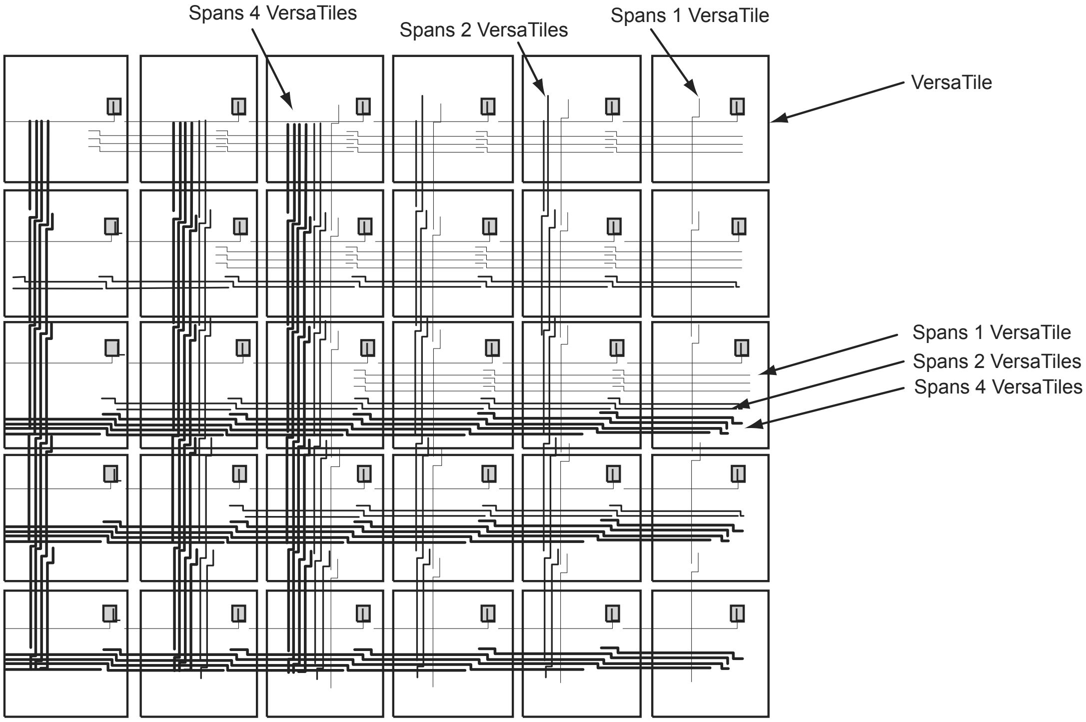


*Figure 1-11* • **Efficient Long-Line Resources**

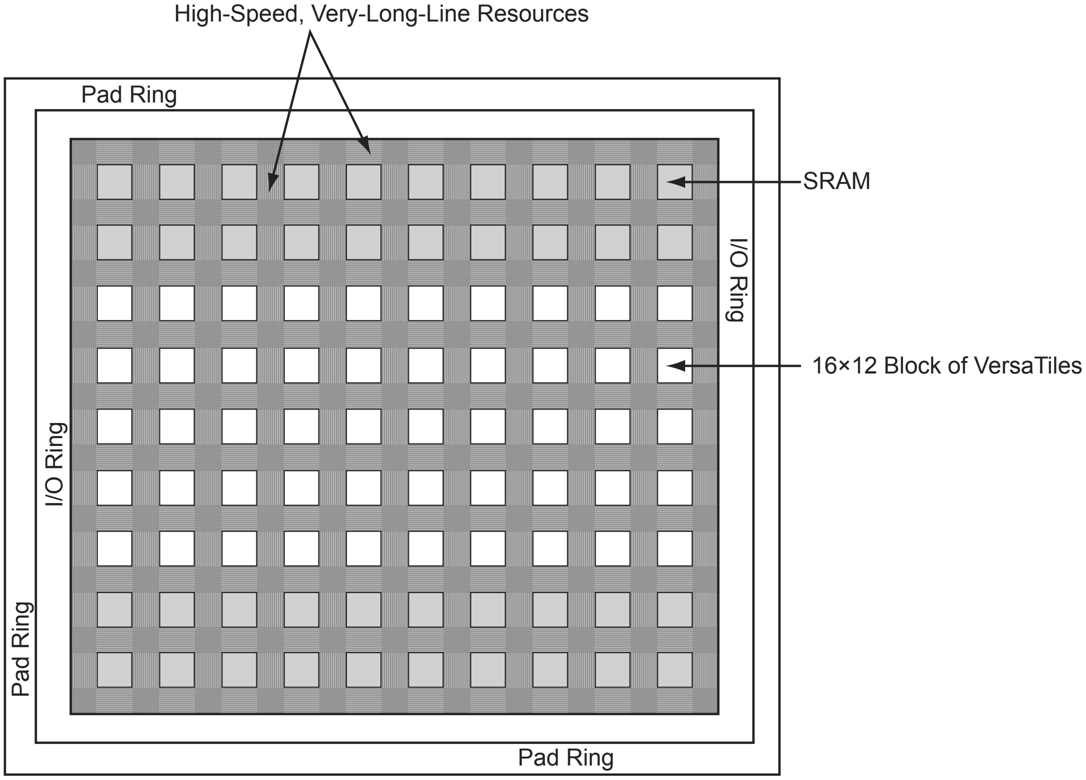


*Figure 1-12* • **Very-Long-Line Resources**

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

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

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

# List of Changes

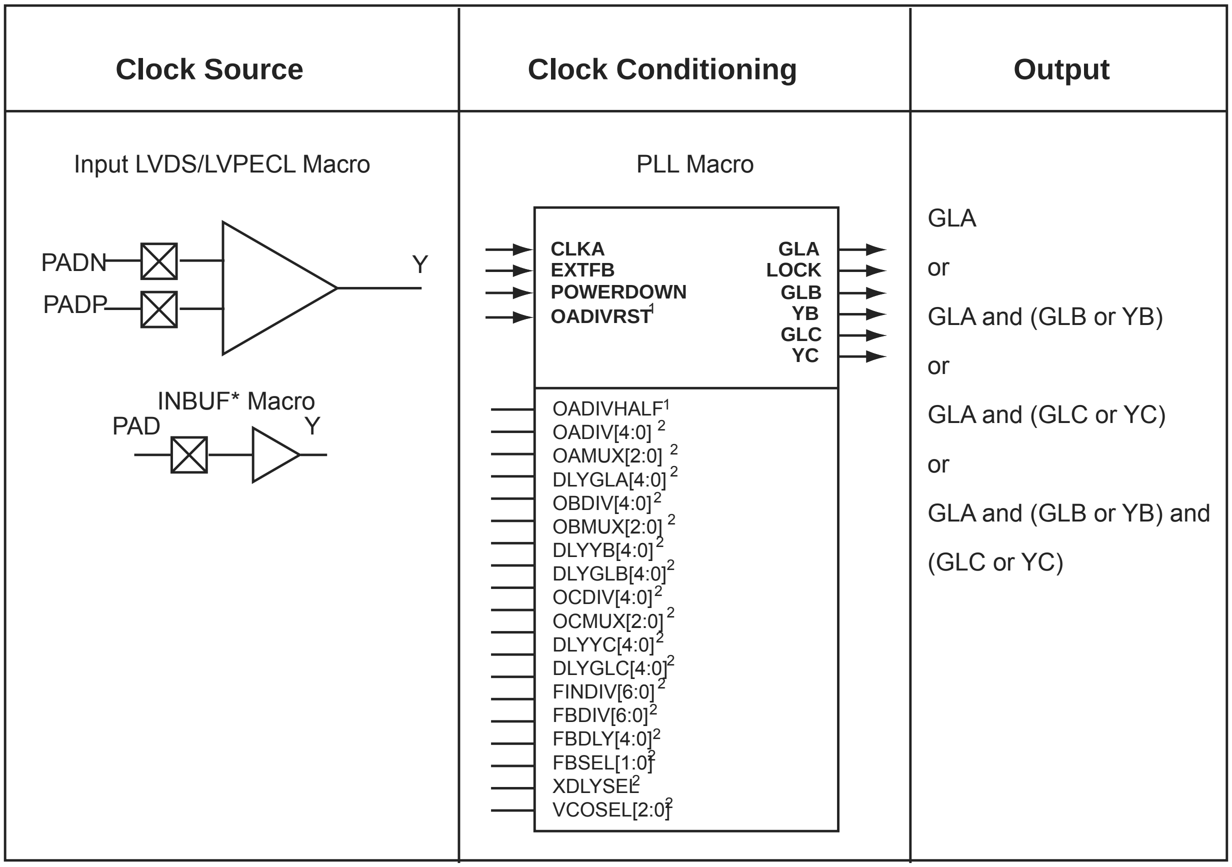
The following table lists critical changes that were made in each revision of the chapter.

| 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 |
| | The "Global Architecture" section and "VersaNet Global Network Distribution" section were revised for clarity (SARs 20646, 24779). | 31, 33 |
| | The "I/O Banks and Global I/Os" section was moved earlier in the document, renamed to "Chip and Quadrant Global I/Os", and revised for clarity. Figure 3-4 • Global Connections Details, Figure 3-6 • Global Inputs, Table 3-2 • Chip Global Pin Name, and Table 3-3 • Quadrant Global Pin Name are new (SARs 20646, 24779). | 35 |
| | The "Clock Aggregation Architecture" section was revised (SARs 20646, 24779). | 41 |
| | Figure 3-7 • Chip Global Aggregation was revised (SARs 20646, 24779). | 43 |
| | The "Global Macro and Placement Selections" section is new (SARs 20646, 24779). | 48 |
| v1.4<br>(December 2008) | The "Global Architecture" section was updated to include 10 k devices, and to include information about VersaNet global support for IGLOO nano devices. | 31 |
| | The Table 3-1 • Flash-Based FPGAs was updated to include IGLOO nano and ProASIC3 nano devices. | 32 |
| | The "VersaNet Global Network Distribution" section was updated to include 10 k devices and to note an exception in global lines for nano devices. | 33 |
| | Figure 3-2 • Simplified VersaNet Global Network (30 k gates and below) is new. | 34 |
| | The "Spine Architecture" section was updated to clarify support for 10 k and nano devices. | 41 |
| | Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include IGLOO nano and ProASIC3 nano devices. | 41 |
| | The figure in the CLKBUF_LVDS/LVPECL row of Table 3-8 • Clock Macros was updated to change CLKBIBUF to CLKBUF. | 46 |
| v1.3<br>(October 2008) | A third bullet was added to the beginning of the "Global Architecture" section: In Fusion devices, the west CCC also contains a PLL core. In the two larger devices (AFS600 and AFS1500), the west and east CCCs each contain a PLL. | 31 |
| | The "Global Resource Support in Flash-Based Devices" section was revised to include new families and make the information more concise. | 32 |
| | Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include A3PE600/L in the device column. | 41 |
| | Table note 1 was revised in Table 3-9 • I/O Standards within CLKBUF to include AFS600 and AFS1500. | 47 |
| v1.2<br>(June 2008) | The following changes were made to the family descriptions in Table 3-1 • Flash-Based FPGAs:<br>• ProASIC3L was updated to include 1.5 V.<br>• The number of PLLs for ProASIC3E was changed from five to six. | 32 |

## Global Buffers with PLL Function

Clocks requiring frequency synthesis or clock adjustments can utilize the PLL core before connecting to the global / quadrant global networks. A maximum of 18 CCC global buffers can be instantiated in a device—three per CCC and up to six CCCs per device. Each PLL core can generate up to three global/quadrant clocks, while a clock delay element provides one.

The PLL functionality of the clock conditioning block is supported by the PLL macro.



*Notes:*

1. *For Fusion only.*
2. *Refer to the* IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide *for more information.*
3. *For INBUF* driving a PLL macro or CLKDLY macro, the I/O will be hard-routed to the CCC; i.e., will be placed by software to a dedicated Global I/O.*
4. *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

***Figure 4-4* • CCC Options: Global Buffers with PLL**

The PLL macro provides five derived clocks (three independent) from a single reference clock. The PLL macro also provides power-down input and lock output signals. The additional inputs shown on the macro are configuration settings, which are configured through the use of SmartGen. For manual setting of these bits refer to the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide* for details.

Figure 4-6 on page 71 illustrates the various clock output options and delay elements.

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.

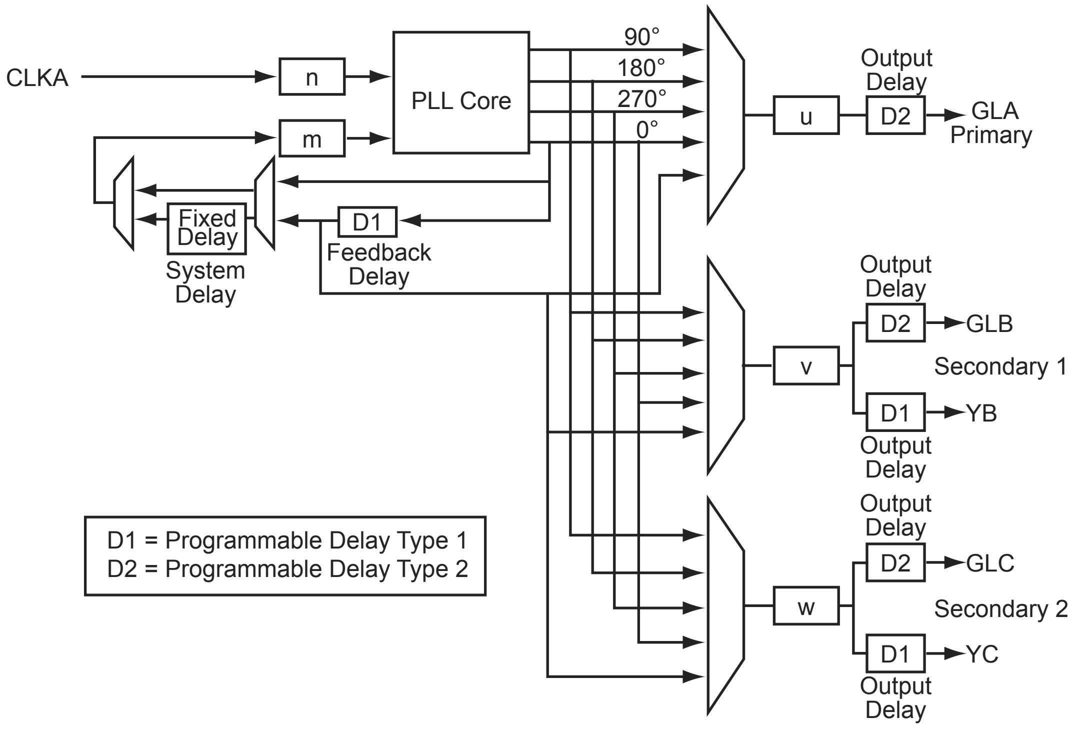


***Figure 4-20*** **• PLL Block Diagram**

# Conclusion

The Fusion, IGLOO, and ProASIC3 families are the only FPGAs that offer on-chip FlashROM support. This document presents information on the FlashROM architecture, possible applications, programming, access through the JTAG and UJTAG interface, and integration into your design. In addition, the Libero tool set enables easy creation and modification of the FlashROM content.

The nonvolatile FlashROM block in the FPGA can be customized, enabling multiple applications.

Additionally, the security offered by the low power flash devices keeps both the contents of FlashROM and the FPGA design safe from system over-builders, system cloners, and IP thieves.

# Related Documents

## User's Guides

*FlashPro User's Guide*

http://www.microsemi.com/documents/FlashPro_UG.pdf

# 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 |
| v1.4<br>(December 2008) | IGLOO nano and ProASIC3 nano devices were added to Table 5-1 • Flash-Based FPGAs. | 118 |
| v1.3<br>(October 2008) | The "FlashROM Support in Flash-Based Devices" section was revised to include new families and make the information more concise. | 118 |
| | Figure 5-2 • Fusion Device Architecture Overview (AFS600) was replaced. Figure 5-5 • Programming FlashROM Using AES was revised to change "Fusion" to "Flash Device." | 119, 121 |
| | The *FlashPoint User's Guide* was removed from the "User's Guides" section, as its content is now part of the *FlashPro User's Guide*. | 130 |
| v1.2<br>(June 2008) | The following changes were made to the family descriptions in Table 5-1 • Flash-Based FPGAs:<br>• ProASIC3L was updated to include 1.5 V.<br>• The number of PLLs for ProASIC3E was changed from five to six. | 118 |
| v1.1<br>(March 2008) | The chapter was updated to include the IGLOO PLUS family and information regarding 15 k gate devices. The "IGLOO Terminology" section and "ProASIC3 Terminology" section are new. | N/A |

## 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**

### Solution 2

This solution requires one board resistor and one Zener 3.3 V diode, as demonstrated in Figure 7-6.

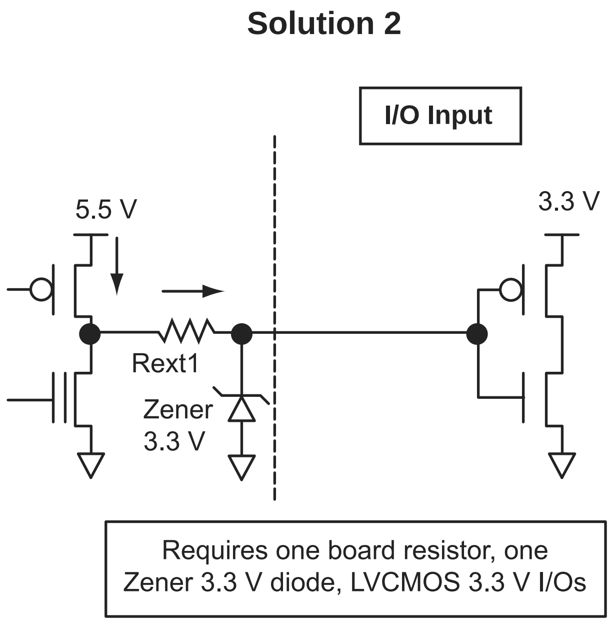


***Figure 7-6*** **• Solution 2**

### Solution 3

This solution requires a bus switch on the board, as demonstrated in Figure 7-7.

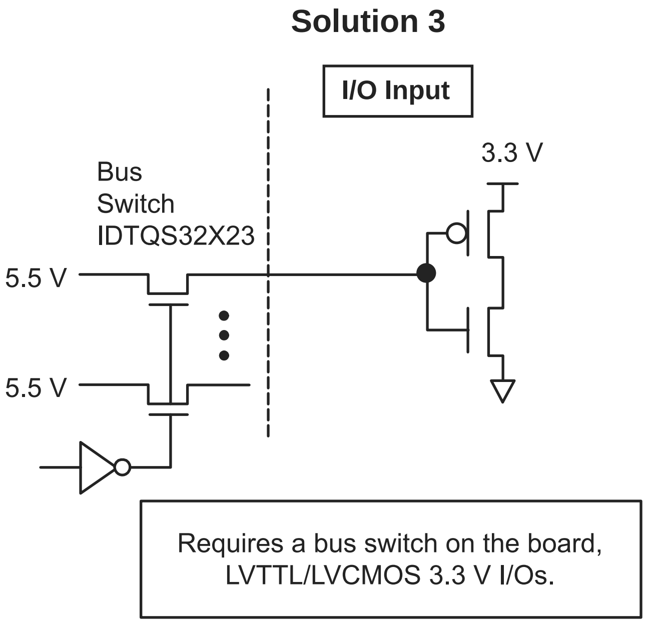


***Figure 7-7*** **• Solution 3**

Refer to Table 7-10 on page 169 for more information about the slew rate and drive strength specification for LVTTL/LVCMOS 3.3 V, LVCMOS 2.5 V, LVCMOS 1.8 V, LVCMOS 1.5 V, and LVCMOS 1.2 V output buffers.

***Table 7-14*** **• nano Output Drive and Slew**

| I/O Standards | 2 mA | 4 mA | 6 mA | 8 mA | Slew | |
|---|---|---|---|---|---|---|
| LVTTL / LVCMOS 3.3 V | ✓ | ✓ | ✓ | ✓ | High | Low |
| LVCMOS 2.5 V | ✓ | ✓ | ✓ | ✓ | High | Low |
| LVCMOS 1.8 V | ✓ | ✓ | – | – | High | Low |
| LVCMOS 1.5 V | ✓ | – | – | – | High | Low |
| LVCMOS 1.2 V | ✓ | – | – | – | High | Low |

# Simultaneously Switching Outputs (SSOs) and Printed Circuit Board Layout

Each I/O voltage bank has a separate ground and power plane for input and output circuits. 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.

SSOs can cause signal integrity problems on adjacent signals that are not part of the SSO bus. Both inductive and capacitive coupling parasitics of bond wires inside packages and of traces on PCBs will transfer noise from SSO busses onto signals adjacent to those busses. Additionally, SSOs can produce ground bounce noise and VCCI dip noise. These two noise types are caused by rapidly changing currents through GND and VCCI package pin inductances during switching activities (EQ 1 and EQ 2).

$$\text{Ground bounce noise voltage} = L(\text{GND}) \times di/dt$$

*EQ 1*

$$\text{VCCI dip noise voltage} = L(\text{VCCI}) \times di/dt$$

*EQ 2*

Any group of four or more input pins switching on the same clock edge is considered an SSO bus. The shielding should be done both on the board and inside the package unless otherwise described.

In-package shielding can be achieved in several ways; the required shielding will vary depending on whether pins next to the SSO bus are LVTTL/LVCMOS inputs or LVTTL/LVCMOS outputs. Board traces in the vicinity of the SSO bus have to be adequately shielded from mutual coupling and inductive noise that can be generated by the SSO bus. Also, noise generated by the SSO bus needs to be reduced inside the package.

PCBs perform an important function in feeding stable supply voltages to the IC and, at the same time, maintaining signal integrity between devices.

Key issues that need to be considered are as follows:

- Power and ground plane design and decoupling network design
- Transmission line reflections and terminations

For extensive data per package on the SSO and PCB issues, refer to the "ProASIC3/E SSO and Pin Placement and Guidelines" chapter of the *ProASIC3 Device Family User’s Guide*.

### Output Buffers

There are two variations: Regular and Special.

If the **Regular** variation is selected, only the Width (1 to 128) needs to be entered. The default value for Width is 1.

The **Special** variation has Width, Technology, Output Drive, and Slew Rate options.

### Bidirectional Buffers

There are two variations: Regular and Special.

The **Regular** variation has Enable Polarity (Active High, Active Low) in addition to the Width option.

The **Special** variation has Width, Technology, Output Drive, Slew Rate, and Resistor Pull-Up/-Down options.

### Tristate Buffers

Same as Bidirectional Buffers.

### DDR

There are eight variations: DDR with Regular Input Buffers, Special Input Buffers, Regular Output Buffers, Special Output Buffers, Regular Tristate Buffers, Special Tristate Buffers, Regular Bidirectional Buffers, and Special Bidirectional Buffers.

These variations resemble the options of the previous I/O macro. For example, the Special Input Buffers variation has Width, Technology, Voltage Level, and Resistor Pull-Up/-Down options. DDR is not available on IGLOO PLUS devices.

4. Once the desired configuration is selected, click the **Generate** button. The Generate Core window opens (Figure 8-4).
5. Enter a name for the macro. Click **OK**. The core will be generated and saved to the appropriate location within the project files (Figure 8-5 on page 191).

*Figure 8-4* • **Generate Core Window**

6. Instantiate the I/O macro in the top-level code.

   The user must instantiate the DDR_REG or DDR_OUT macro in the design. Use SmartGen to generate both these macros and then instantiate them in your top level. To combine the DDR macros with the I/O, the following rules must be met:

## Compiling the Design

During Compile, a PDC I/O constraint file can be imported along with the netlist file. If only the netlist file is compiled, certain I/O assignments need to be completed before proceeding to Layout. All constraints that can be entered in PDC can also be entered using ChipPlanner, I/O Attribute Editor, and PinEditor.

There are certain rules that must be followed in implementing I/O register combining and the I/O DDR macro (refer to the I/O Registers section of the handbook for the device that you are using and the "DDR" section on page 190 for details). Provided these rules are met, the user can enable or disable I/O register combining by using the PDC command `set_io portname –register yes|no` in the I/O Attribute Editor or selecting a check box in the Compile Options dialog box (see Figure 8-7). The Compile Options dialog box appears when the design is compiled for the first time. It can also be accessed by choosing **Options** > **Compile** during successive runs. I/O register combining is off by default. The PDC command overrides the setting in the Compile Options dialog box.

*Figure 8-7* • **Setting Register Combining During Compile**

## Understanding the Compile Report

The I/O bank report is generated during Compile and displayed in the log window. This report lists the I/O assignments necessary before Layout can proceed.

When Designer is started, the I/O Bank Assigner tool is run automatically if the Layout command is executed. The I/O Bank Assigner takes care of the necessary I/O assignments. However, these assignments can also be made manually with MVN or by importing the PDC file. Refer to the "Assigning Technologies and VREF to I/O Banks" section on page 198 for further description.

The I/O bank report can also be extracted from Designer by choosing **Tools** > **Report** and setting the Report Type to **IOBank**.

This report has the following tables: I/O Function, I/O Technology, I/O Bank Resource Usage, and I/O Voltage Usage. This report is useful if the user wants to do I/O assignments manually.

# I/O Cell Architecture

Low power flash devices support DDR in the I/O cells in four different modes: Input, Output, Tristate, and Bidirectional pins. For each mode, different I/O standards are supported, with most I/O standards having special sub-options. For the ProASIC3 nano and IGLOO nano devices, DDR is supported only in the 60 k, 125 k, and 250 k logic densities. Refer to Table 9-2 for a sample of the available I/O options. Additional I/O options can be found in the relevant family datasheet.

***Table 9-2*** **• DDR I/O Options**

| DDR Register Type | I/O Type | I/O Standard | Sub-Options | Comments |
|---|---|---|---|---|
| Receive Register | Input | Normal | None | 3.3 V TTL (default) |
| | | LVCMOS | Voltage | 1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default) |
| | | | Pull-Up | None (default) |
| | | PCI/PCI-X | None | |
| | | GTL/GTL+ | Voltage | 2.5 V, 3.3 V (3.3 V default) |
| | | HSTL | Class | I / II (I default) |
| | | SSTL2/SSTL3 | Class | I / II (I default) |
| | | LVPECL | None | |
| | | LVDS | None | |
| Transmit Register | Output | Normal | None | 3.3 V TTL (default) |
| | | LVTTL | Output Drive | 2, 4, 6, 8, 12, 16, 24, 36 mA (8 mA default) |
| | | | Slew Rate | Low/high (high default) |
| | | LVCMOS | Voltage | 1.5 V, 1.8 V, 2.5 V, 5 V (1.5 V default) |
| | | PCI/PCI-X | None | |
| | | GTL/GTL+ | Voltage | 1.8 V, 2.5 V, 3.3 V (3.3 V default) |
| | | HSTL | Class | I / II (I default) |
| | | SSTL2/SSTL3 | Class | I / II (I default) |
| | | LVPECL* | None | |
| | | LVDS* | None | |

*Note: *IGLOO nano and ProASIC3 nano devices do not support differential inputs.*

## DDR Tristate Output Register

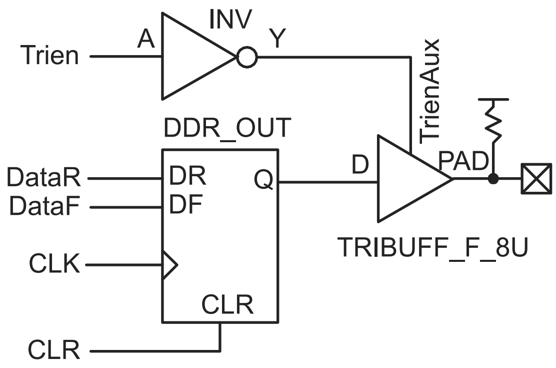


*Figure 9-7* • **DDR Tristate Output Register, LOW Enable, 8 mA, Pull-Up (LVTTL)**

### *Verilog*

```
module DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp(DataR, DataF, CLR, CLK, Trien,
  PAD);

input   DataR, DataF, CLR, CLK, Trien;
output  PAD;

wire TrienAux, Q;

  INV Inv_Tri(.A(Trien),.Y(TrienAux));
  DDR_OUT DDR_OUT_0_inst(.DR(DataR),.DF(DataF),.CLK(CLK),.CLR(CLR),.Q(Q));
  TRIBUFF_F_8U TRIBUFF_F_8U_0_inst(.D(Q),.E(TrienAux),.PAD(PAD));

endmodule
```

### *VHDL*

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

entity DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is
  port(DataR, DataF, CLR, CLK, Trien : in std_logic;  PAD : out std_logic) ;
end DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp;

architecture DEF_ARCH of DDR_TriStateBuf_LVTTL_8mA_HighSlew_LowEnb_PullUp is

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

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

  component TRIBUFF_F_8U
    port(D, E : in std_logic := 'U'; PAD : out std_logic) ;
  end component;

signal TrienAux, Q : std_logic ;

begin

  Inv_Tri : INV
  port map(A => Trien, Y => TrienAux);
```

## *Device Programmers*

### Single Device Programmer

Single device programmers are used to program a device before it is mounted on the system board.

The advantage of using device programmers is that no programming hardware is required on the system board. Therefore, no additional components or board space are required.

Adapter modules are purchased with single device programmers to support the FPGA packages used. The FPGA is placed in the adapter module and the programming software is run from a PC. Microsemi supplies the programming software for all of the Microsemi programmers. The software allows for the selection of the correct die/package and programming files. It will then program and verify the device.

- Single-site programmers

  A single-site programmer programs one device at a time. Microsemi offers Silicon Sculptor 3, built by BP Microsystems, as a single-site programmer. Silicon Sculptor 3 and associated software are available only from Microsemi.

  - Advantages: Lower cost than multi-site programmers. No additional overhead for programming on the system board. Allows local control of programming and data files for maximum security. Allows on-demand programming on-site.
  - Limitations: Only programs one device at a time.
- Multi-site programmers

Often referred to as batch or gang programmers, multi-site programmers can program multiple devices at the same time using the same programming file. This is often used for large volume programming and by programming houses. The sites often have independent processors and memory enabling the sites to operate concurrently, meaning each site may start programming the same file independently. This enables the operator to change one device while the other sites continue programming, which increases throughput. Multiple adapter modules for the same package are required when using a multi-site programmer. Silicon Sculptor I, II, and 3 programmers can be cascaded to program multiple devices in a chain. Multi-site programmers, such as the BP2610 and BP2710, can also be purchased from BP Microsystems. When using BP Microsystems multi-site programmers, users must use programming adapter modules available only from Microsemi. Visit the Microsemi SoC Products Group website to view the part numbers of the desired adapter module:

http://www.microsemi.com/soc/products/hardware/program_debug/ss/modules.aspx.

Also when using BP Microsystems programmers, customers must use Microsemi programming software to ensure the best programming result will occur.

  - Advantages: Provides the capability of programming multiple devices at the same time. No additional overhead for programming on the system board. Allows local control of programming and data files for maximum security.
  - Limitations: More expensive than a single-site programmer
- Automated production (robotic) programmers

Automated production programmers are based on multi-site programmers. They consist of a large input tray holding multiple parts and a robotic arm to select and place parts into appropriate programming sockets automatically. When the programming of the parts is complete, the parts are removed and placed in a finished tray. The automated programmers are often used in volume programming houses to program parts for which the programming time is small. BP Microsystems part number BP4710, BP4610, BP3710 MK2, and BP3610 are available for this purpose. Auto programmers cannot be used to program RTAX-S devices.

Where an auto-programmer is used, the appropriate open-top adapter module from BP Microsystems must be used.

## Application 3: Nontrusted Environment—Field Updates/Upgrades

Programming or reprogramming of devices may occur at remote locations. Reconfiguration of devices in consumer products/equipment through public networks is one example. Typically, the remote system is already programmed with particular design contents. When design update (FPGA array contents update) and/or data upgrade (FlashROM and/or FB contents upgrade) is necessary, an updated programming file with AES encryption can be generated, sent across public networks, and transmitted to the remote system. Reprogramming can then be done using this AES-encrypted programming file, providing easy and secure field upgrades. Low power flash devices support this secure ISP using AES. The detailed flow for this application is shown in Figure 11-8. Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

To prepare devices for this scenario, the user can initially generate a programming file with the available security setting options. This programming file is programmed into the devices before shipment. During the programming file generation step, the user has the option of making the security settings permanent or not. In situations where no changes to the security settings are necessary, the user can select this feature in the software to generate the programming file with permanent security settings. Microsemi recommends that the programming file use encryption with an AES key, especially when ISP is done via public domain.

For example, if the designer wants to use an AES key for the FPGA array and the FlashROM, **Permanent** needs to be chosen for this setting. At first, the user chooses the options to use an AES key for the FPGA array and the FlashROM, and then chooses **Permanently lock the security settings**. A unique AES key is chosen. Once this programming file is generated and programmed to the devices, the AES key is permanently stored in the on-chip memory, where it is secured safely. The devices are sent to distant locations for the intended application. When an update is needed, a new programming file must be generated. The programming file must use the same AES key for encryption; otherwise, the authentication will fail and the file will not be programmed in the device.

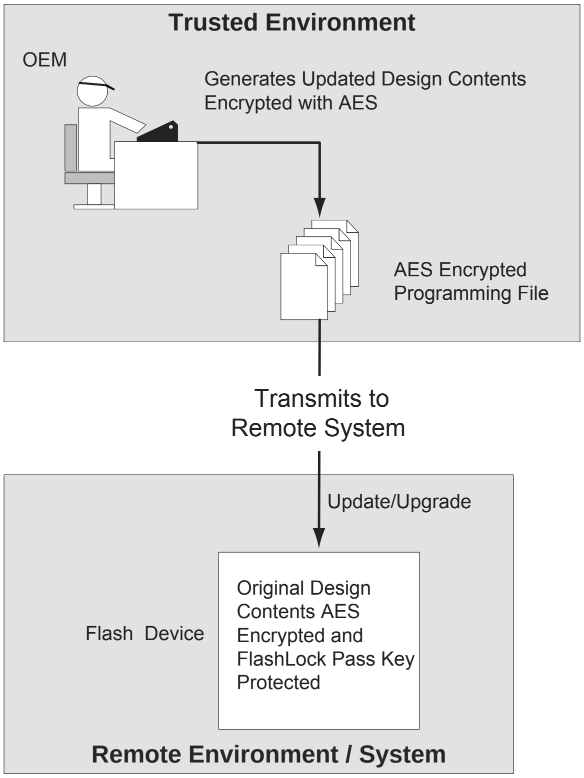


*Figure 11-8* • **Application 3: Nontrusted Environment—Field Updates/Upgrades**

# 13 – Core Voltage Switching Circuit for IGLOO and ProASIC3L In-System Programming

## Introduction

The IGLOO® and ProASIC®3L families offer devices that can be powered by either 1.5 V or, in the case of V2 devices, a core supply voltage anywhere in the range of 1.2 V to 1.5 V, in 50 mV increments.

Since IGLOO and ProASIC3L devices are flash-based, they can be programmed and reprogrammed multiple times in-system using Microsemi FlashPro3. FlashPro3 uses the JTAG standard interface (IEEE 1149.1) and STAPL file (defined in JESD 71 to support programming of programmable devices using IEEE 1149.1) for in-system configuration/programming (IEEE 1532) of a device. Programming can also be executed by other methods, such as an embedded microcontroller that follows the same standards above.

All IGLOO and ProASIC3L devices must be programmed with the VCC core voltage at 1.5 V. Therefore, applications using IGLOO or ProASIC3L devices powered by a 1.2 V supply must switch the core supply to 1.5 V for in-system programming.

The purpose of this document is to describe an easy-to-use and cost-effective solution for switching the core supply voltage from 1.2 V to 1.5 V during in-system programming for IGLOO and ProASIC3L devices.

# 14 – Microprocessor Programming of Microsemi's Low Power Flash Devices

## Introduction

The Fusion, IGLOO, and ProASIC3 families of flash FPGAs support in-system programming (ISP) with the use of a microprocessor. Flash-based FPGAs store their configuration information in the actual cells within the FPGA fabric. SRAM-based devices need an external configuration memory, and hybrid nonvolatile devices store the configuration in a flash memory inside the same package as the SRAM FPGA. Since the programming of a true flash FPGA is simpler, requiring only one stage, it makes sense that programming with a microprocessor in-system should be simpler than with other SRAM FPGAs. This reduces bill-of-materials costs and printed circuit board (PCB) area, and increases system reliability.

Nonvolatile flash technology also gives the low power flash devices the advantage of a secure, low power, live-at-power-up, and single-chip solution. Low power flash devices are reprogrammable and offer time-to-market benefits at an ASIC-level unit cost. These features enable engineers to create high-density systems using existing ASIC or FPGA design flows and tools.

This document is an introduction to microprocessor programming only. To explain the difference between the options available, user's guides for DirectC and STAPL provide more detail on implementing each style.

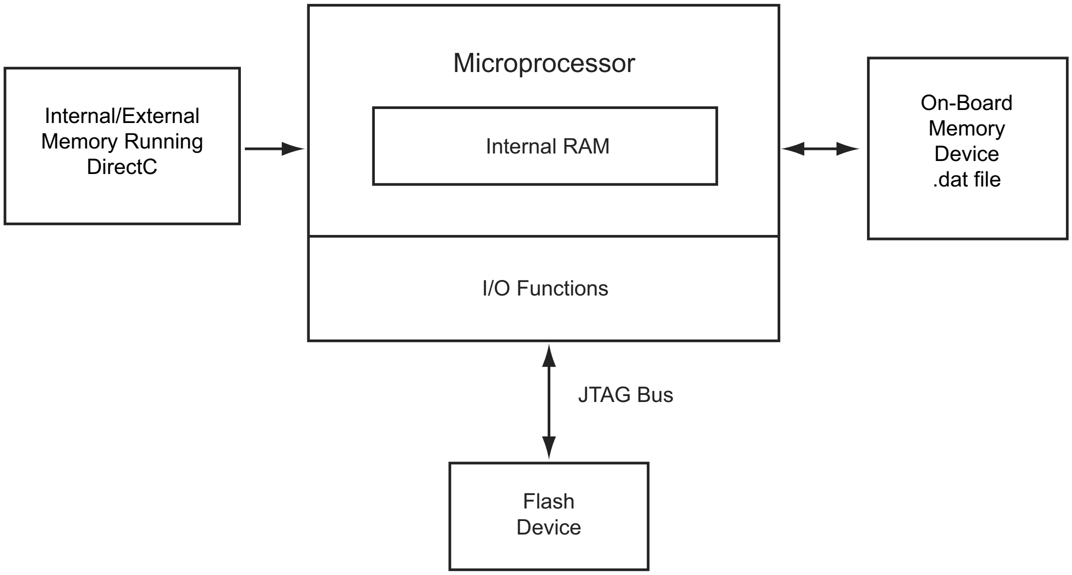


*Figure 14-1* • **ISP Using Microprocessor**

**Microsemi Corporate Headquarters**
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996