Welcome to E-XFL.COM

Understanding Embedded - FPGAs (Field Programmable Gate Array)

Embedded - FPGAs, or Field Programmable Gate Arrays, are advanced integrated circuits that offer unparalleled flexibility and performance for digital systems. Unlike traditional fixed-function logic devices, FPGAs can be programmed and reprogrammed to execute a wide array of logical operations, enabling customized functionality tailored to specific applications. This reprogrammability allows developers to iterate designs quickly and implement complex functions without the need for custom hardware.

Applications of Embedded - FPGAs

The versatility of Embedded - FPGAs makes them indispensable in numerous fields. In telecommunications,

Details

Product Status	Active
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	110592
Number of I/O	97
Number of Gates	600000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	144-LBGA
Supplier Device Package	144-FPBGA (13x13)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3p600l-1fg144i

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 50. 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 50). 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 60. 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 (4×3)	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

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

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

Since the architecture of the devices varies as size increases, the following list details I/O types supported for globals:

IGLOO and ProASIC3

- LVDS-based clock sources are available only on 250 k gate devices and above (IGLOO nano and ProASIC3 nano devices do not support differential inputs).
- 60 k and 125 k gate devices support single-ended clock sources only.
- 15 k and 30 k gate devices support these inputs for CCC only and do not contain a PLL.
- nano devices:
  - 10 k, 15 k, and 20 k devices do not contain PLLs in the CCCs, and support only CLKBUF and CLKINT.
  - 60 k, 125 k, and 250 k devices support one PLL in the middle left CCC position. In the absence of the PLL, this CCC can be used by CLKBUF, CLKINT, and CLKDLY macros. The corner CCCs support CLKBUF, CLKINT, and CLKDLY.

Fusion

- AFS600 and AFS1500: All single-ended, differential, and voltage-referenced I/O standards (Pro I/O).
- AFS090 and AFS250: All single-ended and differential I/O standards.

Clock Sources for PLL and CLKDLY Macros

The input reference clock (CLKA for a PLL macro, CLK for a CLKDLY macro) can be accessed from different sources via the associated clock multiplexer tree. Each CCC has the option of choosing the source of the input clock from one of the following:

- Hardwired I/O
- External I/O
- Core Logic
- RC Oscillator (Fusion only)
- Crystal Oscillator (Fusion only)

The SmartGen macro builder tool allows users to easily create the PLL and CLKDLY macros with the desired settings. Microsemi strongly recommends using SmartGen to generate the CCC macros.
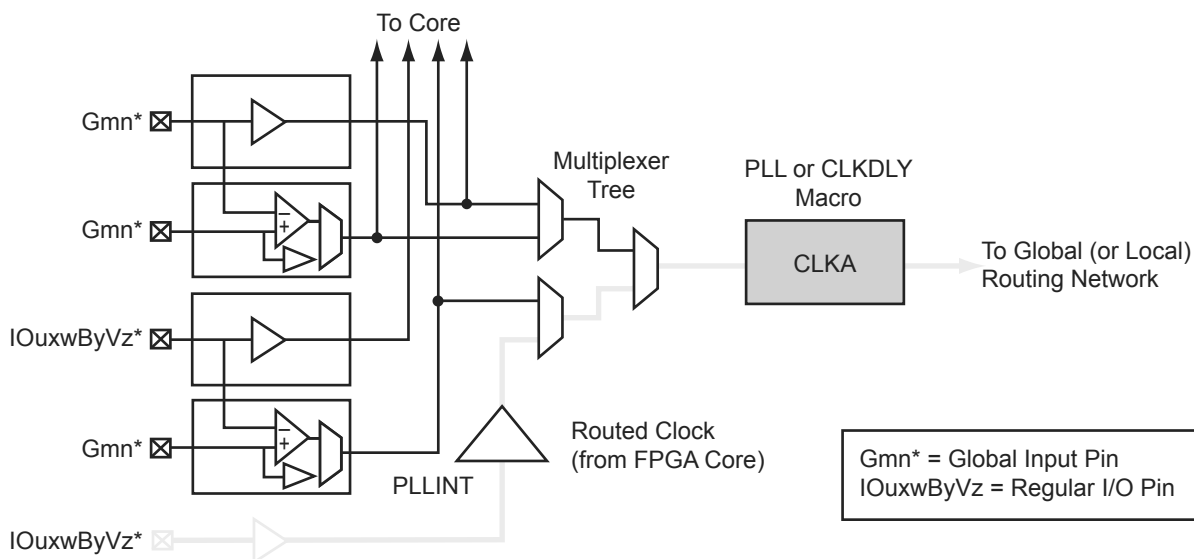
Hardwired I/O Clock Source

Hardwired I/O refers to global input pins that are hardwired to the multiplexer tree, which directly accesses the CCC global buffers. These global input pins have designated pin locations and are indicated with the I/O naming convention *Gmn* (*m* refers to any one of the positions where the PLL core is available, and *n* refers to any one of the three global input MUXes and the pin number of the associated global location, *m*). Choosing this option provides the benefit of directly connecting to the CCC reference clock input, which provides less delay. See Figure 4-9 on page 90 for an example illustration of the connections, shown in red. If a CLKDLY macro is initiated to utilize the programmable delay element of the CCC, the clock input can be placed at one of nine dedicated global input pin locations. In other words, if Hardwired I/O is chosen as the input source, the user can decide to place the input pin in one of the GmA0, GmA1, GmA2, GmB0, GmB1, GmB2, GmC0, GmC1, or GmC2 locations of the low power flash devices. When a PLL macro is used to utilize the PLL core in a CCC location, the clock input of the PLL can only be connected to one of three GmA* global pin locations: GmA0, GmA1, or GmA2.

External I/O Clock Source

External I/O refers to regular I/O pins. The clock source is instantiated with one of the various INBUF options and accesses the CCCs via internal routing. The user has the option of assigning this input to any of the I/Os labeled with the I/O convention *IOuxwByVz*. Refer to the "User I/O Naming Conventions in I/O Structures" chapter of the appropriate device user's guide, and for Fusion, refer to the *Fusion Family of Mixed Signal FPGAs* datasheet for more information. Figure 4-11 gives a brief explanation of external I/O usage. Choosing this option provides the freedom of selecting any user I/O location but introduces additional delay because the signal connects to the routed clock input through internal routing before connecting to the CCC reference clock input.

For the External I/O option, the routed signal would be instantiated with a PLLINT macro before connecting to the CCC reference clock input. This instantiation is conveniently done automatically by SmartGen when this option is selected. Microsemi recommends using the SmartGen tool to generate the CCC macro. The instantiation of the PLLINT macro results in the use of the routed clock input of the I/O to connect to the PLL clock input. If not using SmartGen, manually instantiate a PLLINT macro before the PLL reference clock to indicate that the regular I/O driving the PLL reference clock should be used (see Figure 4-11 for an example illustration of the connections, shown in red).

In the above two options, the clock source must be instantiated with one of the various INBUF macros. The reference clock pins of the CCC functional block core macros must be driven by regular input macros (INBUFs), not clock input macros.


Figure 4-11 • **Illustration of External I/O 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.

IGLOO and ProASIC3 CCC Locations

In all IGLOO and ProASIC3 devices (except 10 k through 30 k gate devices, which do not contain PLLs), six CCCs are located in the same positions as the IGLOOe and ProASIC3E CCCs. Only one of the CCCs has an integrated PLL and is located in the middle of the west (middle left) side of the device. The other five CCCs are simplified CCCs and are located in the four corners and the middle of the east side of the device (Figure 4-14).
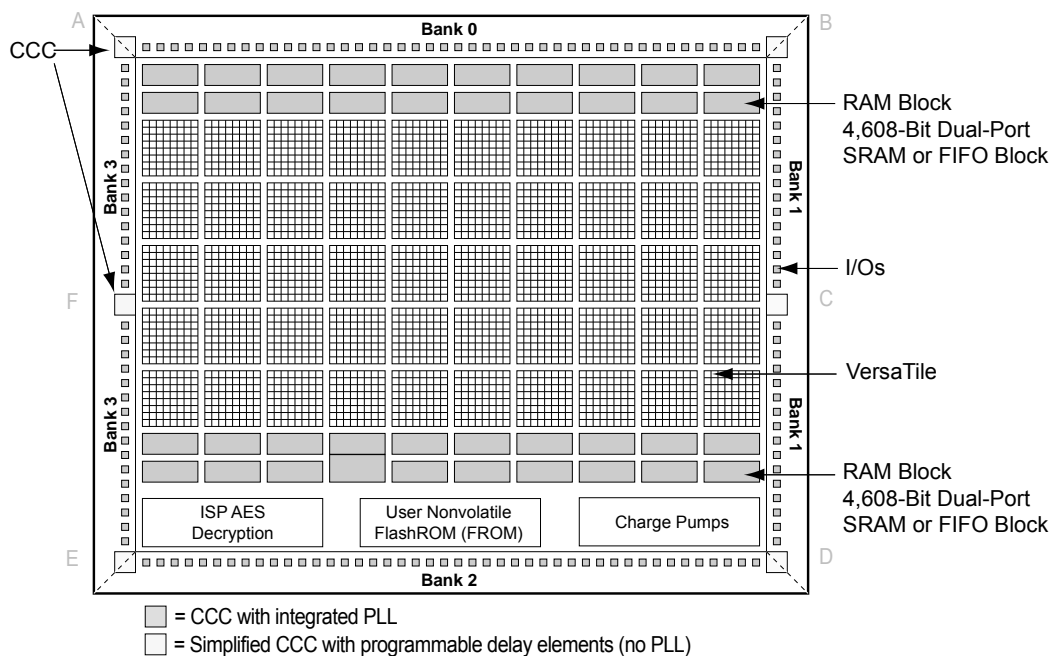
Figure 4-14 • **CCC Locations in IGLOO and ProASIC3 Family Devices (except 10 k through 30 k gate devices)**

Note: The number and architecture of the banks are different for some devices.

10 k through 30 k gate devices do not support PLL features. In these devices, there are two CCC-GLs at the lower corners (one at the lower right, and one at the lower left). These CCC-GLs do not have programmable delays.

PLL Core Specifications

PLL core specifications can be found in the DC and Switching Characteristics chapter of the appropriate family datasheet.

Loop Bandwidth

Common design practice for systems with a low-noise input clock is to have PLLs with small loop bandwidths to reduce the effects of noise sources at the output. Table 4-6 shows the PLL loop bandwidth, providing a measure of the PLL's ability to track the input clock and jitter.

Table 4-6 • **–3 dB Frequency of the PLL**

	Minimum (T_a = +125°C, VCCA = 1.4 V)	Typical (T_a = +25°C, VCCA = 1.5 V)	Maximum (T_a = −55°C, VCCA = 1.6 V)
−3 dB Frequency	15 kHz	25 kHz	45 kHz

PLL Core Operating Principles

This section briefly describes the basic principles of PLL operation. The PLL core is composed of a phase detector (PD), a low-pass filter (LPF), and a four-phase voltage-controlled oscillator (VCO). Figure 4-19 illustrates a basic single-phase PLL core with a divider and delay in the feedback path.
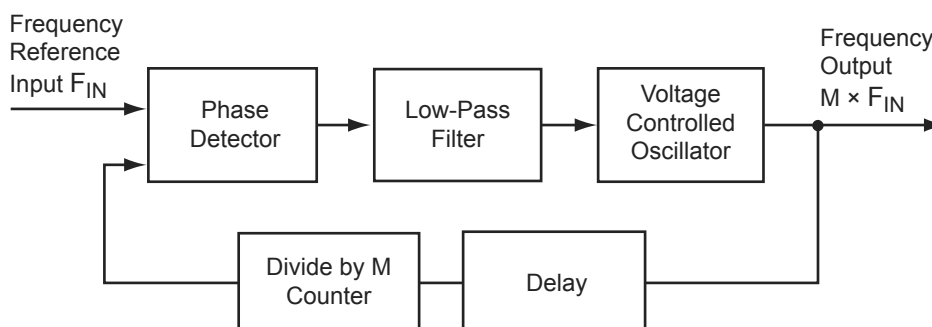
Figure 4-19 • **Simplified PLL Core with Feedback Divider and Delay**

The PLL is an electronic servo loop that phase-aligns the PD feedback signal with the reference input. To achieve this, the PLL dynamically adjusts the VCO output signal according to the average phase difference between the input and feedback signals.

The first element is the PD, which produces a voltage proportional to the phase difference between its inputs. A simple example of a digital phase detector is an Exclusive-OR gate. The second element, the LPF, extracts the average voltage from the phase detector and applies it to the VCO. This applied voltage alters the resonant frequency of the VCO, thus adjusting its output frequency.

Consider Figure 4-19 with the feedback path bypassing the divider and delay elements. If the LPF steadily applies a voltage to the VCO such that the output frequency is identical to the input frequency, this steady-state condition is known as lock. Note that the input and output phases are also identical. The PLL core sets a LOCK output signal HIGH to indicate this condition.

Should the input frequency increase slightly, the PD detects the frequency/phase difference between its reference and feedback input signals. Since the PD output is proportional to the phase difference, the change causes the output from the LPF to increase. This voltage change increases the resonant frequency of the VCO and increases the feedback frequency as a result. The PLL dynamically adjusts in this manner until the PD senses two phase-identical signals and steady-state lock is achieved. The opposite (decreasing PD output signal) occurs when the input frequency decreases.

Now suppose the feedback divider is inserted in the feedback path. As the division factor M (shown in Figure 4-20 on page 101) is increased, the average phase difference increases. The average phase

Table 4-8 • **Configuration Bit Descriptions for the CCC Blocks (continued)**

Config. Bits	Signal	Name	Description
83	RXCSEL[1]	CLKC input selection	Select the CLKC input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 110).[2]
82	RXBSEL[1]	CLKB input selection	Select the CLKB input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 110).[2]
81	RXASEL[1]	CLKA input selection	Select the CLKA input clock source between RC oscillator and crystal oscillator (refer to Table 4-16 on page 110).[2]
80	RESETEN	Reset Enable	Enables (active high) the synchronization of PLL output dividers after dynamic reconfiguration (SUPDATE). The Reset Enable signal is READ-ONLY.
79	DYNCSEL	Clock Input C Dynamic Select	Configures clock input C to be sent to GLC for dynamic control.[2]
78	DYNBSEL	Clock Input B Dynamic Select	Configures clock input B to be sent to GLB for dynamic control.[2]
77	DYNASEL	Clock Input A Dynamic Select	Configures clock input A for dynamic PLL configuration.[2]
<76:74>	VCOSEL[2:0]	VCO Gear Control	Three-bit VCO Gear Control for four frequency ranges (refer to Table 4-19 on page 111 and Table 4-20 on page 111).
73	STATCSEL	MUX Select on Input C	MUX selection for clock input C[2]
72	STATBSEL	MUX Select on Input B	MUX selection for clock input B[2]
71	STATASEL	MUX Select on Input A	MUX selection for clock input A[2]
<70:66>	DLYC[4:0]	YC Output Delay	Sets the output delay value for YC.
<65:61>	DLYB[4:0]	YB Output Delay	Sets the output delay value for YB.
<60:56>	DLYGLC[4:0]	GLC Output Delay	Sets the output delay value for GLC.
<55:51>	DLYGLB[4:0]	GLB Output Delay	Sets the output delay value for GLB.
<50:46>	DLYGLA[4:0]	Primary Output Delay	Primary GLA output delay
45	XDLYSEL	System Delay Select	When selected, inserts System Delay in the feedback path in Figure 4-20 on page 101.
<44:40>	FBDLY[4:0]	Feedback Delay	Sets the feedback delay value for the feedback element in Figure 4-20 on page 101.
<39:38>	FBSEL[1:0]	Primary Feedback Delay Select	Controls the feedback MUX: no delay, include programmable delay element, or use external feedback.
<37:35>	OCMUX[2:0]	Secondary 2 Output Select	Selects from the VCO's four phase outputs for GLC/YC.
<34:32>	OBMUX[2:0]	Secondary 1 Output Select	Selects from the VCO's four phase outputs for GLB/YB.

Notes:

1. *The <88:81> configuration bits are only for the Fusion dynamic CCC.*

2. *This value depends on the input clock source, so Layout must complete before these bits can be set. After completing Layout in Designer, generate the "CCC_Configuration" report by choosing **Tools** > **Report** > **CCC_Configuration**. The report contains the appropriate settings for these bits.*

```
    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.

SmartGen allows you to generate the FlashROM netlist in VHDL, Verilog, or EDIF format. After the FlashROM netlist is generated, the core can be instantiated in the main design like other SmartGen cores. Note that the macro library name for FlashROM is UFROM. The following is a sample FlashROM VHDL netlist that can be instantiated in the main design:

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

entity FROM_a is
  port( ADDR : in std_logic_vector(6 downto 0); DOUT : out std_logic_vector(7 downto 0));
end FROM_a;

architecture DEF_ARCH of  FROM_a is

  component UFROM
    generic (MEMORYFILE:string);
    port(DO0, DO1, DO2, DO3, DO4, DO5, DO6, DO7 : out std_logic;
      ADDR0, ADDR1, ADDR2, ADDR3, ADDR4, ADDR5, ADDR6 : in std_logic := 'U') ;
  end component;

  component GND
    port( Y : out std_logic);
  end component;

signal U_7_PIN2 : std_logic ;

begin

  GND_1_net : GND port map(Y => U_7_PIN2);
  UFROM0 : UFROM
  generic map(MEMORYFILE => "FROM_a.mem")
  port map(DO0 => DOUT(0), DO1 => DOUT(1), DO2 => DOUT(2), DO3 => DOUT(3), DO4 => DOUT(4),
    DO5 => DOUT(5), DO6 => DOUT(6), DO7 => DOUT(7), ADDR0 => ADDR(0), ADDR1 => ADDR(1),
    ADDR2 => ADDR(2), ADDR3 => ADDR(3), ADDR4 => ADDR(4), ADDR5 => ADDR(5),
    ADDR6 => ADDR(6));

end DEF_ARCH;
```

SmartGen generates the following files along with the netlist. These are located in the SmartGen folder for the Libero SoC project.

1. MEM (Memory Initialization) file
2. UFC (User Flash Configuration) file
3. Log file

The MEM file is used for simulation, as explained in the "Simulation of FlashROM Design" section on page 143. The UFC file, generated by SmartGen, has the FlashROM configuration for single or multiple devices and is used during STAPL generation. It contains the region properties and simulation values. Note that any changes in the MEM file will not be reflected in the UFC file. Do not modify the UFC to change FlashROM content. Instead, use the SmartGen GUI to modify the FlashROM content. See the "Programming File Generation for FlashROM Design" section on page 143 for a description of how the UFC file is used during the programming file generation. The log file has information regarding the file type and file location.

I/O Standards

Single-Ended Standards

These I/O standards use a push-pull CMOS output stage with a voltage referenced to system ground to designate logical states. The input buffer configuration, output drive, and I/O supply voltage (VCCI) vary among the I/O standards (Figure 7-5).
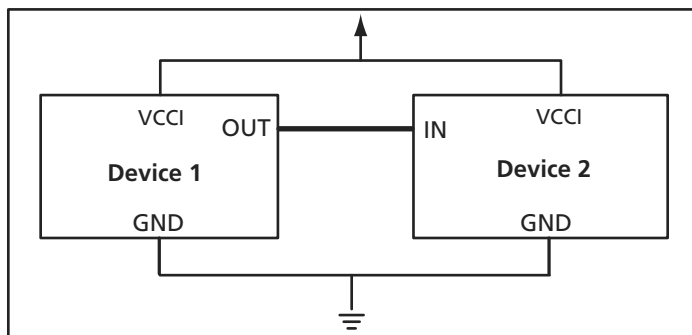
Figure 7-5 • **Single-Ended I/O Standard Topology**

The advantage of these standards is that a common ground can be used for multiple I/Os. This simplifies board layout and reduces system cost. Their low-edge-rate (*dv/dt*) data transmission causes less electromagnetic interference (EMI) on the board. However, they are not suitable for high-frequency (>200 MHz) switching due to noise impact and higher power consumption.

LVTTL (Low-Voltage TTL)

This is a general-purpose standard (EIA/JESD8-B) for 3.3 V applications. It uses an LVTTL input buffer and a push-pull output buffer. The LVTTL output buffer can have up to six different programmable drive strengths. The default drive strength is 12 mA. VCCI is 3.3 V. Refer to "I/O Programmable Features" on page 188 for details.

LVCMOS (Low-Voltage CMOS)

The low power flash devices provide four different kinds of LVCMOS: LVCMOS 3.3 V, LVCMOS 2.5 V, LVCMOS 1.8 V, and LVCMOS 1.5 V. LVCMOS 3.3 V is an extension of the LVCMOS standard (JESD8-B–compliant) used for general-purpose 3.3 V applications.

LVCMOS 2.5 V is an extension of the LVCMOS standard (JESD8-5–compliant) used for general-purpose 2.5 V applications.

There is yet another standard supported by IGLOO and ProASIC3 devices (except A3P030): LVCMOS 2.5/5.0 V. This standard is similar to LVCMOS 2.5 V, with the exception that it can support up to 3.3 V on the input side (2.5 V output drive).

LVCMOS 1.8 V is an extension of the LVCMOS standard (JESD8-7–compliant) used for general-purpose 1.8 V applications. LVCMOS 1.5 V is an extension of the LVCMOS standard (JESD8-11–compliant) used for general-purpose 1.5 V applications.

The VCCI values for these standards are 3.3 V, 2.5 V, 1.8 V, and 1.5 V, respectively. Like LVTTL, the output buffer has up to seven different programmable drive strengths (2, 4, 6, 8, 12, 16, and 24 mA). Refer to "I/O Programmable Features" on page 188 for details.

3.3 V PCI (Peripheral Component Interface)

This standard specifies support for both 33 MHz and 66 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. With the aid of an external resistor, this I/O standard can be 5 V–compliant for low power flash devices. It does not have programmable drive strength.

3.3 V PCI-X (Peripheral Component Interface Extended)

An enhanced version of the PCI specification, 3.3 V PCI-X can support higher average bandwidths; it increases the speed that data can move within a computer from 66 MHz to 133 MHz. It is backward-
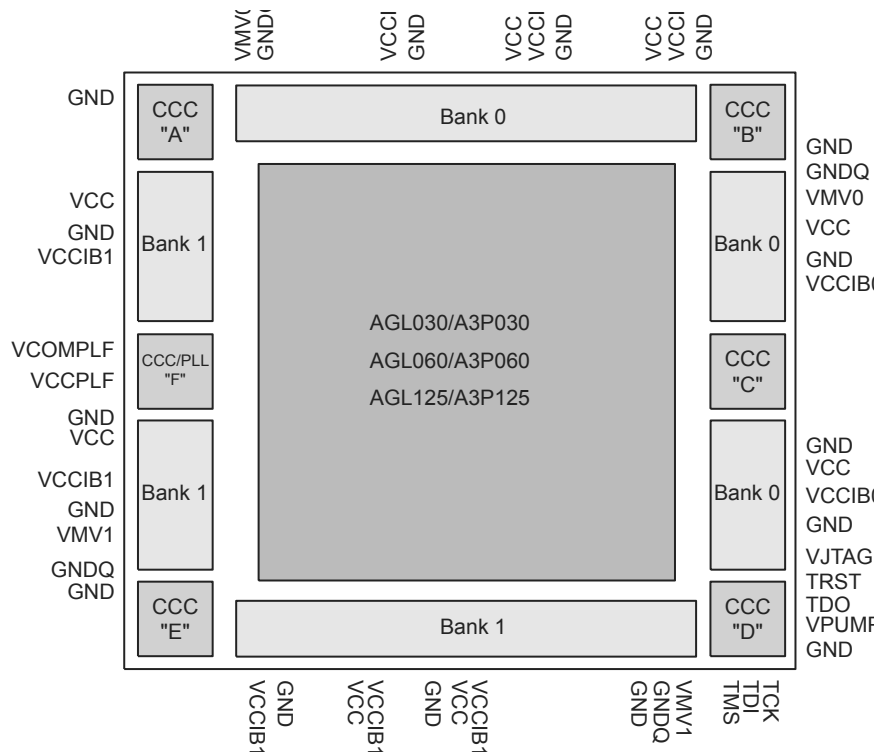
Note: The 30 k gate devices do not support a PLL (V$_{COMPLF}$ and V$_{CCPLF}$ pins).

Figure 7-19 • **Naming Conventions of IGLOO and ProASIC3 Devices with Two I/O Banks – Top View**
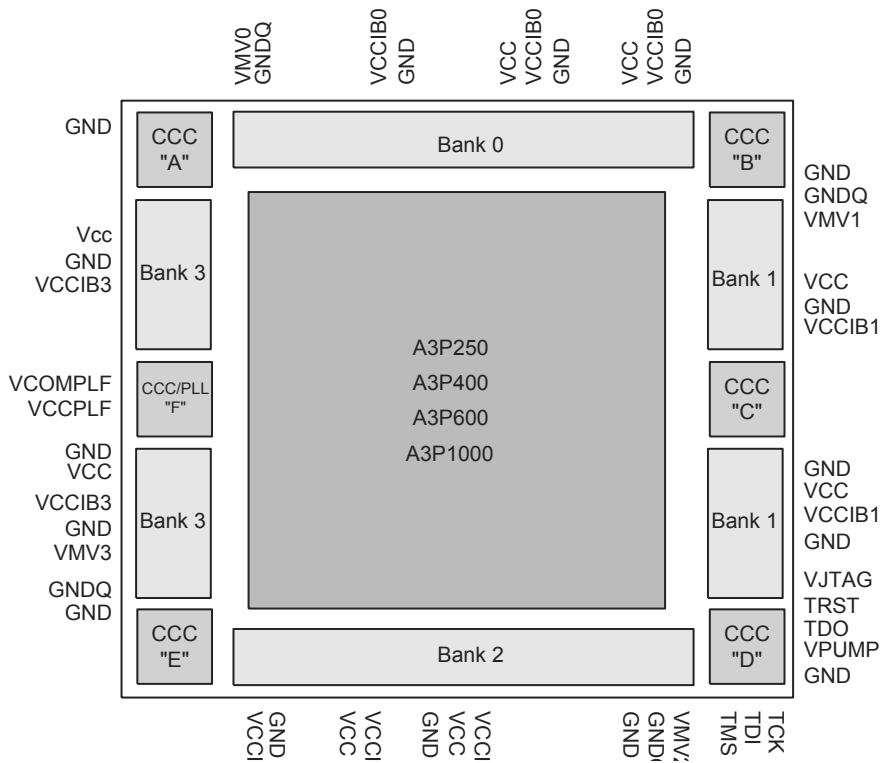
Figure 7-20 • **Naming Conventions of IGLOO and ProASIC3 Devices with Four I/O Banks – Top View**

I/O Banks and I/O Standards Compatibility

I/Os are grouped into I/O voltage banks.

Each I/O voltage bank has dedicated I/O supply and ground voltages (VMV/GNDQ for input buffers and V_{CCI}/GND for output buffers). Because of these dedicated supplies, only I/Os with compatible standards can be assigned to the same I/O voltage bank. Table 8-3 on page 217 shows the required voltage compatibility values for each of these voltages.

There are eight I/O banks (two per side).

Every I/O bank is divided into minibanks. Any user I/O in a VREF minibank (a minibank is the region of scope of a VREF pin) can be configured as a VREF pin (Figure 8-2). Only one V_{REF} pin is needed to control the entire V_{REF} minibank. The location and scope of the V_{REF} minibanks can be determined by the I/O name. For details, see the user I/O naming conventions for "IGLOOe and ProASIC3E" on page 245. Table 8-5 on page 217 shows the I/O standards supported by IGLOOe and ProASIC3E devices, and the corresponding voltage levels.

I/O standards are compatible if they comply with the following:

- Their VCCI and VMV values are identical.
- Both of the standards need a VREF, and their VREF values are identical.
- All inputs and disabled outputs are voltage tolerant up to 3.3 V.

For more information about I/O and global assignments to I/O banks in a device, refer to the specific pin table for the device in the packaging section of the datasheet, and see the user I/O naming conventions for "IGLOOe and ProASIC3E" on page 245.
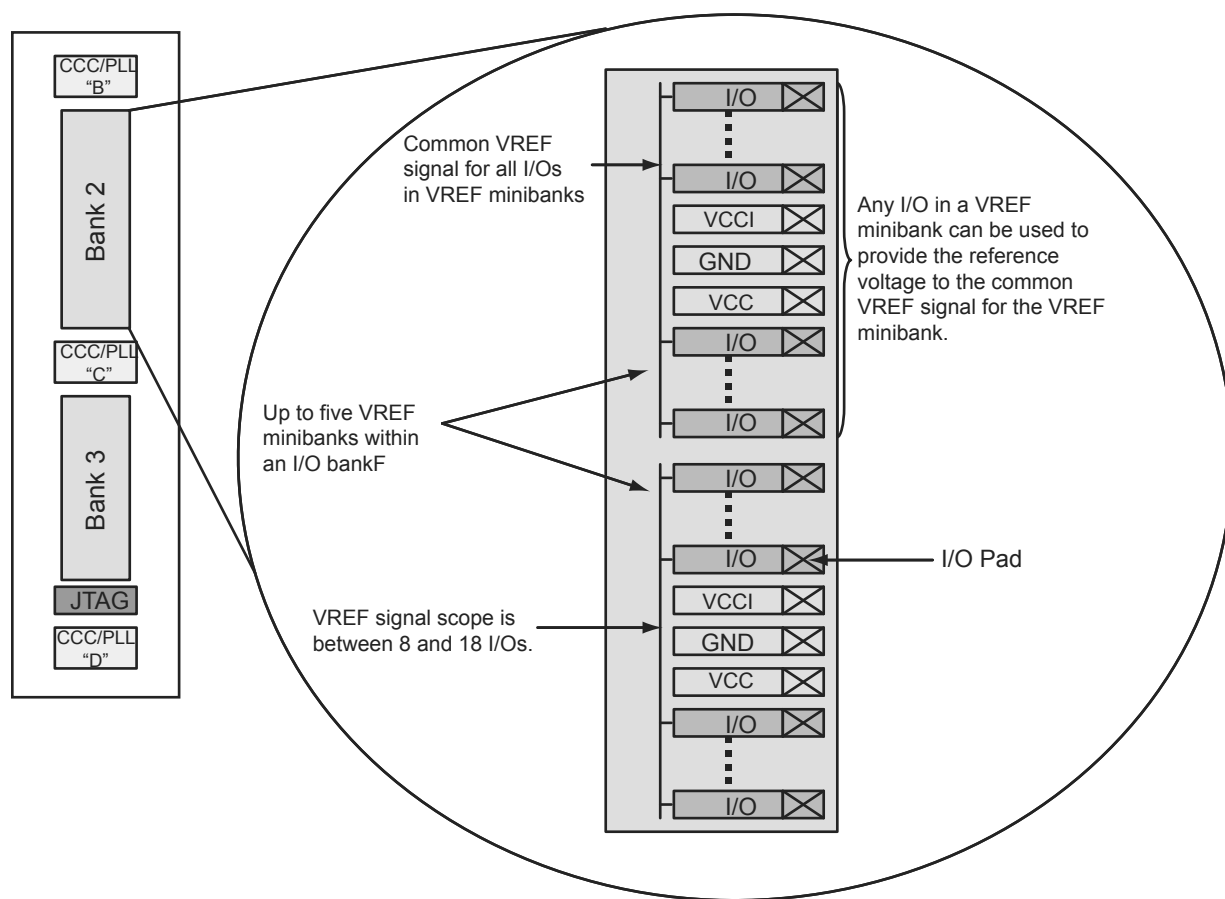
Figure 8-2 • **Typical IGLOOe and ProASIC3E I/O Bank Detail Showing V_{REF} Minibanks**

IGLOOe and ProASIC3E

For devices requiring Level 3 and/or Level 4 compliance, the board drivers connected to the I/Os must have 10 kΩ (or lower) output drive resistance at hot insertion, and 1 kΩ (or lower) output drive resistance at hot removal. This resistance is the transmitter resistance sending a signal toward the I/O, and no additional resistance is needed on the board. If that cannot be assured, three levels of staging can be used to achieve Level 3 and/or Level 4 compliance. Cards with two levels of staging should have the following sequence:

- Grounds
- Powers, I/Os, and other pins

Cold-Sparing Support

Cold-sparing refers to the ability of a device to leave system data undisturbed when the system is powered up, while the component itself is powered down, or when power supplies are floating.

Cold-sparing is supported on ProASIC3E devices only when the user provides resistors from each power supply to ground. The resistor value is calculated based on the decoupling capacitance on a given power supply. The RC constant should be greater than 3 µs.

To remove resistor current during operation, it is suggested that the resistor be disconnected (e.g., with an NMOS switch) from the power supply after the supply has reached its final value. Refer to the "Power-Up/-Down Behavior of Low Power Flash Devices" section on page 373 for details on cold-sparing.

Cold-sparing means that a subsystem with no power applied (usually a circuit board) is electrically connected to the system that is in operation. This means that all input buffers of the subsystem must present very high input impedance with no power applied so as not to disturb the operating portion of the system.

The 30 k gate devices fully support cold-sparing, since the I/O clamp diode is always off (see Table 8-13 on page 231). If the 30 k gate device is used in applications requiring cold-sparing, a discharge path from the power supply to ground should be provided. This can be done with a discharge resistor or a switched resistor. This is necessary because the 30 k gate devices do not have built-in I/O clamp diodes.

For other IGLOOe and ProASIC3E devices, since the I/O clamp diode is always active, cold-sparing can be accomplished either by employing a bus switch to isolate the device I/Os from the rest of the system or by driving each I/O pin to 0 V. If the resistor is chosen, the resistor value must be calculated based on decoupling capacitance on a given power supply on the board (this decoupling capacitance is in parallel with the resistor). The RC time constant should ensure full discharge of supplies before cold-sparing functionality is required. The resistor is necessary to ensure that the power pins are discharged to ground every time there is an interruption of power to the device.

IGLOOe and ProASIC3E devices support cold-sparing for all I/O configurations. Standards, such as PCI, that require I/O clamp diodes can also achieve cold-sparing compliance, since clamp diodes get disconnected internally when the supplies are at 0 V.

When targeting low power applications, I/O cold-sparing may add additional current if a pin is configured with either a pull-up or pull-down resistor and driven in the opposite direction. A small static current is induced on each I/O pin when the pin is driven to a voltage opposite to the weak pull resistor. The current is equal to the voltage drop across the input pin divided by the pull resistor. Refer to the "Detailed I/O DC Characteristics" section of the appropriate family datasheet for the specific pull resistor value for the corresponding I/O standard.

For example, assuming an LVTTL 3.3 V input pin is configured with a weak pull-up resistor, a current will flow through the pull-up resistor if the input pin is driven LOW. For LVTTL 3.3 V, the pull-up resistor is ~45 kΩ, and the resulting current is equal to 3.3 V / 45 kΩ = 73 µA for the I/O pin. This is true also when a weak pull-down is chosen and the input pin is driven High. This current can be avoided by driving the input Low when a weak pull-down resistor is used and driving it High when a weak pull-up resistor is used.

I/O Register Combining

Every I/O has several embedded registers in the I/O tile that are close to the I/O pads. Rather than using the internal register from the core, the user has the option of using these registers for faster clock-to-out timing, and external hold and setup. When combining these registers at the I/O buffer, some architectural rules must be met. Provided these rules are met, the user can enable register combining globally during Compile (as shown in the "Compiling the Design" section on page 261).

This feature is supported by all I/O standards.

Rules for Registered I/O Function

1. The fanout between an I/O pin (D, Y, or E) and a register must be equal to one for combining to be considered on that pin.

2. All registers (Input, Output, and Output Enable) connected to an I/O must share the same clear or preset function:
   – If one of the registers has a CLR pin, all the other registers that are candidates for combining in the I/O must have a CLR pin.
   – If one of the registers has a PRE pin, all the other registers that are candidates for combining in the I/O must have a PRE pin.
   – If one of the registers has neither a CLR nor a PRE pin, all the other registers that are candidates for combining must have neither a CLR nor a PRE pin.
   – If the clear or preset pins are present, they must have the same polarity.
   – If the clear or preset pins are present, they must be driven by the same signal (net).

3. Registers connected to an I/O on the Output and Output Enable pins must have the same clock and enable function:
   – Both the Output and Output Enable registers must have an E pin (clock enable), or none at all.
   – If the E pins are present, they must have the same polarity. The CLK pins must also have the same polarity.

In some cases, the user may want registers to be combined with the input of a bibuf while maintaining the output as-is. This can be achieved by using PDC commands as follows:

```
set_io <signal name> -REGISTER yes ------register will combine
set_preserve <signal name> ----register will not combine
```

Weak Pull-Up and Weak Pull-Down Resistors

When the I/O is pulled up, it is connected to the VCCI of its corresponding I/O bank. When it is pulled down, it is connected to GND. Refer to the datasheet for more information.

For low power applications, configuration of the pull-up or pull-down of the I/O can be used to set the I/O to a known state while the device is in Flash*Freeze mode. Refer to the "Flash*Freeze Technology and Low Power Modes in IGLOO and ProASIC3L Devices" chapter in the *IGLOOe FPGA Fabric User's Guide* or *ProASIC3E FPGA Fabric User's Guide* for more information.

The Flash*Freeze (FF) pin cannot be configured with a weak pull-down or pull-up I/O attribute, as the signal needs to be driven at all times.

Output Slew Rate Control

The slew rate is the amount of time an input signal takes to get from logic LOW to logic HIGH or vice versa.

It is commonly defined as the propagation delay between 10% and 90% of the signal's voltage swing. Slew rate control is available for the output buffers of low power flash devices. The output buffer has a programmable slew rate for both HIGH-to-LOW and LOW-to-HIGH transitions. Slew rate control is available for LVTTL, LVCMOS, and PCI-X I/O standards. The other I/O standards have a preset slew value.

The slew rate can be implemented by using a PDC command (Table 8-6 on page 218), setting it "High" or "Low" in the I/O Attribute Editor in Designer, or instantiating a special I/O macro. The default slew rate value is "High."

Programming Support in Flash Devices

The flash FPGAs listed in Table 11-1 support flash in-system programming and the functions described in this document.

Table 11-1 • **Flash-Based FPGAs**

Series	Family[*]	Description
IGLOO	IGLOO	Ultra-low power 1.2 V to 1.5 V FPGAs with Flash*Freeze technology
	IGLOOe	Higher density IGLOO FPGAs with six PLLs and additional I/O standards
	IGLOO nano	The industry's lowest-power, smallest-size solution, supporting 1.2 V to 1.5 V core voltage with Flash*Freeze technology
	IGLOO PLUS	IGLOO FPGAs with enhanced I/O capabilities
ProASIC3	ProASIC3	Low power, high-performance 1.5 V FPGAs
	ProASIC3E	Higher density ProASIC3 FPGAs with six PLLs and additional I/O standards
	ProASIC3 nano	Lowest-cost solution with enhanced I/O capabilities
	ProASIC3L	ProASIC3 FPGAs supporting 1.2 V to 1.5 V core voltage with Flash*Freeze technology
	RT ProASIC3	Radiation-tolerant RT3PE600L and RT3PE3000L
	Military ProASIC3/EL	Military temperature A3PE600L, A3P1000, and A3PE3000L
	Automotive ProASIC3	ProASIC3 FPGAs qualified for automotive applications
SmartFusion	SmartFusion	Mixed-signal FPGA integrating FPGA fabric, programmable microcontroller subsystem (MSS), including programmable analog and ARM® Cortex™-M3 hard processor and flash memory in a monolithic device
Fusion	Fusion	Mixed signal FPGA integrating ProASIC3 FPGA fabric, programmable analog block, support for ARM® Cortex™-M1 soft processors, and flash memory into a monolithic device
ProASIC	ProASIC	First generation ProASIC devices
	ProASIC^PLUS^	Second generation ProASIC devices

Note: *The device names link to the appropriate datasheet, including product brief, DC and switching characteristics, and packaging information.

IGLOO Terminology

In documentation, the terms IGLOO series and IGLOO devices refer to all of the IGLOO devices as listed in Table 11-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

ProASIC3 Terminology

In documentation, the terms ProASIC3 series and ProASIC3 devices refer to all of the ProASIC3 devices as listed in Table 11-1. Where the information applies to only one product line or limited devices, these exclusions will be explicitly stated.

To further understand the differences between the IGLOO and ProASIC3 devices, refer to the *Industry's Lowest Power FPGAs Portfolio*.

Cortex-M1 Device Security

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

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

AES Encryption of Programming Files

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

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

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

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

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

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

Advanced Encryption Standard

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

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

3. Choose the desired settings for the FlashROM configurations to be programmed (Figure 12-13). Click **Finish** to generate the STAPL programming file for the design.

Figure 12-13 • **FlashROM Configuration Settings for Low Power Flash Devices**

Generation of Security Header Programming File Only— Application 2

As mentioned in the "Application 2: Nontrusted Environment—Unsecured Location" section on page 309, the designer may employ FlashLock Pass Key protection or FlashLock Pass Key with AES encryption on the device before sending it to a nontrusted or unsecured location for device programming. To achieve this, the user needs to generate a programming file containing only the security settings desired (Security Header programming file).

Note: If AES encryption is configured, FlashLock Pass Key protection must also be configured.

The available security options are indicated in Table 12-4 and Table 12-5 on page 317.

Table 12-4 • **FlashLock Security Options for IGLOO and ProASIC3**

Security Option	FlashROM Only	FPGA Core Only	Both FlashROM and FPGA
No AES / no FlashLock	–	–	–
FlashLock only	✓	✓	✓
AES and FlashLock	✓	✓	✓

# Security in ARM-Enabled Low Power Flash Devices

There are slight differences between the regular flash device and the ARM-enabled flash devices, which have the M1 prefix.

The AES key is used by Microsemi and preprogrammed into the device to protect the ARM IP. As a result, the design will be encrypted along with the ARM IP, according to the details below.

Cortex-M1 and Cortex-M3 Device Security

Cortex-M1–enabled and Cortex-M3 devices are shipped with the following security features:

- FPGA array enabled for AES-encrypted programming and verification
- FlashROM enabled for AES-encrypted write and verify
- Embedded Flash Memory enabled for AES encrypted write
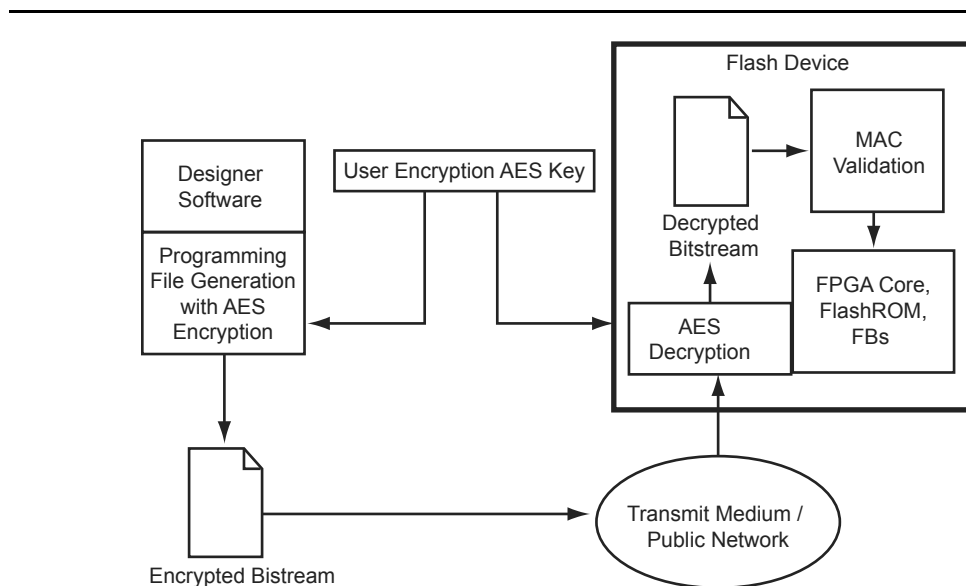
Figure 13-1 • **AES-128 Security Features**

Table 13-4 • **Programming Header Pin Numbers and Description**

Pin	Signal	Source	Description
1	TCK	Programmer	JTAG Clock
2	GND[1]	–	Signal Reference
3	TDO	Target Board	Test Data Output
4	NC	–	No Connect (FlashPro3/3X); Prog_Mode (FlashPro4). See note associated with Figure 13-5 on page 335 regarding Prog_Mode on FlashPro4.
5	TMS	Programmer	Test Mode Select
6	VJTAG	Target Board	JTAG Supply Voltage
7	VPUMP[2]	Programmer/Target Board	Programming Supply Voltage
8	nTRST	Programmer	JTAG Test Reset (Hi-Z with 10 kΩ pull-down, HIGH, LOW, or toggling)
9	TDI	Programmer	Test Data Input
10	GND[1]	–	Signal Reference

Notes:

1. *Both GND pins must be connected.*

2. *FlashPro4/3/3X can provide VPUMP if there is only one device on the target board.*

3. VCC switches from 1.5 V to 1.2 V when TRST is LOW.

Figure 14-4 • **TRST Toggled LOW**

In Figure 14-4, the TRST signal and the VCC core voltage signal are labeled. As TRST is pulled to ground, the core voltage is observed to switch from 1.5 V to 1.2 V. The observed fall time is approximately 2 ms.

DirectC

The above analysis is based on FlashPro3, but there are other solutions to ISP, such as DirectC. DirectC is a microprocessor program that can be run in-system to program Microsemi flash devices. For FlashPro3, TRST is the most convenient control signal to use for the recommended circuit. However, for DirectC, users may use any signal to control the FET. For example, the DirectC code can be edited so that a separate non-JTAG signal can be asserted from the microcontroller that signals the board that it is about to start programming the device. After asserting the N-Channel Digital FET control signal, the programming algorithm must allow sufficient time for the supply to rise to 1.5 V before initiating DirectC programming. As seen in Figure 14-3 on page 345, 50 ms is adequate time. Depending on the size of the PCB and the capacitance on the VCC supply, results may vary from system to system. Microsemi recommends using a conservative value for the wait time to make sure that the VCC core voltage is at the right level.

Conclusion

For applications using IGLOO and ProASIC3L low power FPGAs and taking advantage of the low core voltage power supplies with less than 1.5 V operation, there must be a way for the core voltage to switch from 1.2 V (or other voltage) to 1.5 V, which is required during in-system programming. The circuit explained in this document illustrates one simple, cost-effective way of handling this requirement. A JTAG signal from the FlashPro3 programmer allows the circuit to sense when programming is in progress, enabling it to switch to the correct core voltage.

The following devices and families do not support cold-sparing:
- IGLOO: AGL060, AGL125, AGL250, AGL600, AGL1000
- ProASIC3: A3P060, A3P125, A3P250, A3P400, A3P600, A3P1000
- ProASIC3L: A3P250L, A3P600L, A3P1000L
- Military ProASIC3: A3P1000

Hot-Swapping

Hot-swapping is the operation of hot insertion or hot removal of a card in a powered-up system. The I/Os need to be configured in hot-insertion mode if hot-swapping compliance is required. For more details on the levels of hot-swap compatibility in low power flash devices, refer to the "Hot-Swap Support" section in the I/O Structures chapter of the user's guide for the device you are using.

The following devices and families support hot-swapping:
- IGLOO: AGL015 and AGL030
- All IGLOO nano
- All IGLOO PLUS
- All IGLOOe
- ProASIC3L: A3PE3000L
- ProASIC3: A3P015 and A3P030
- All ProASIC3 nano
- All ProASIC3E
- Military ProASIC3EL: A3PE600L and A3PE3000L
- RT ProASIC3: RT3PE600L and RT3PE3000L

The following devices and families do not support hot-swapping:
- IGLOO: AGL060, AGL125, AGL250, AGL400, AGL600, AGL1000
- ProASIC3: A3P060, A3P125, A3P250, A3P400, A3P600, A3P1000
- ProASIC3L: A3P250L, A3P600L, A3P1000L
- Military ProASIC3: A3P1000

Conclusion

Microsemi's low power flash FPGAs provide an excellent programmable logic solution for a broad range of applications. In addition to high performance, low cost, security, nonvolatility, and single chip, they are live at power-up (meet Level 0 of the LAPU classification) and offer clear and easy-to-use power-up/-down characteristics. Unlike SRAM FPGAs, low power flash devices do not require any specific power-up/-down sequencing and have extremely low power-up inrush current in any power-up sequence. Microsemi low power flash FPGAs also support both cold-sparing and hot-swapping for applications requiring these capabilities.