



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 | 18432 |
| Number of I/O | 71 |
| Number of Gates | 60000 |
| Voltage - Supply | 1.425V ~ 1.575V |
| Mounting Type | Surface Mount |
| Operating Temperature | -40°C ~ 100°C (TJ) |
| Package / Case | 100-TQFP |
| Supplier Device Package | 100-VQFP (14x14) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/a3pn060-1vqg100i |

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, 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, RT3PE3000L | 3 | 6 | 450 | 173 | (3, 2) or (3, 4) | (3, 174) or (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) |

3 – Global Resources in Low Power Flash Devices

Introduction

IGLOO, Fusion, and ProASIC3 FPGA devices offer a powerful, low-delay VersaNet global network scheme and have extensive support for multiple clock domains. In addition to the Clock Conditioning Circuits (CCCs) and phase-locked loops (PLLs), there is a comprehensive global clock distribution network called a VersaNet global network. Each logical element (VersaTile) input and output port has access to these global networks. The VersaNet global networks can be used to distribute low-skew clock signals or high-fanout nets. In addition, these highly segmented VersaNet global networks contain spines (the vertical branches of the global network tree) and ribs that can reach all the VersaTiles inside their region. This allows users the flexibility to create low-skew local clock networks using spines. This document describes VersaNet global networks and discusses how to assign signals to these global networks and spines in a design flow. Details concerning low power flash device PLLs are described in the "Clock Conditioning Circuits in Low Power Flash Devices and Mixed Signal FPGAs" section on page 61. This chapter describes the low power flash devices' global architecture and uses of these global networks in designs.

Global Architecture

Low power flash devices offer powerful and flexible control of circuit timing through the use of global circuitry. Each chip has up to six CCCs, some with PLLs.

- In IGLOOe, ProASIC3EL, and ProASIC3E devices, all CCCs have PLLs—hence, 6 PLLs per device (except the PQ208 package, which has only 2 PLLs).
- In IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3, and ProASIC3L devices, the west CCC contains a PLL core (except in 10 k through 30 k devices).
- 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.

Refer to Table 4-6 on page 84 for details. Each PLL includes delay lines, a phase shifter (0°, 90°, 180°, 270°), and clock multipliers/dividers. Each CCC has all the circuitry needed for the selection and interconnection of inputs to the VersaNet global network. The east and west CCCs each have access to three chip global lines on each side of the chip (six chip global lines total). The CCCs at the four corners each have access to three quadrant global lines in each quadrant of the chip (except in 10 k through 30 k gate devices).

The nano 10 k, 15 k, and 20 k devices support four VersaNet global resources, and 30 k devices support six global resources. The 10 k through 30 k devices have simplified CCCs called CCC-GLs.

The flexible use of the VersaNet global network allows the designer to address several design requirements. User applications that are clock-resource-intensive can easily route external or gated internal clocks using VersaNet global routing networks. Designers can also drastically reduce delay penalties and minimize resource usage by mapping critical, high-fanout nets to the VersaNet global network.

Note: Microsemi recommends that you choose the appropriate global pin and use the appropriate global resource so you can realize these benefits.

The following sections give an overview of the VersaNet global network, the structure of the global network, access point for the global networks, and the clock aggregation feature that enables a design to have very low clock skew using spines.

Figure 3-6 shows all nine global inputs for the location A connected to the top left quadrant global network via CCC.

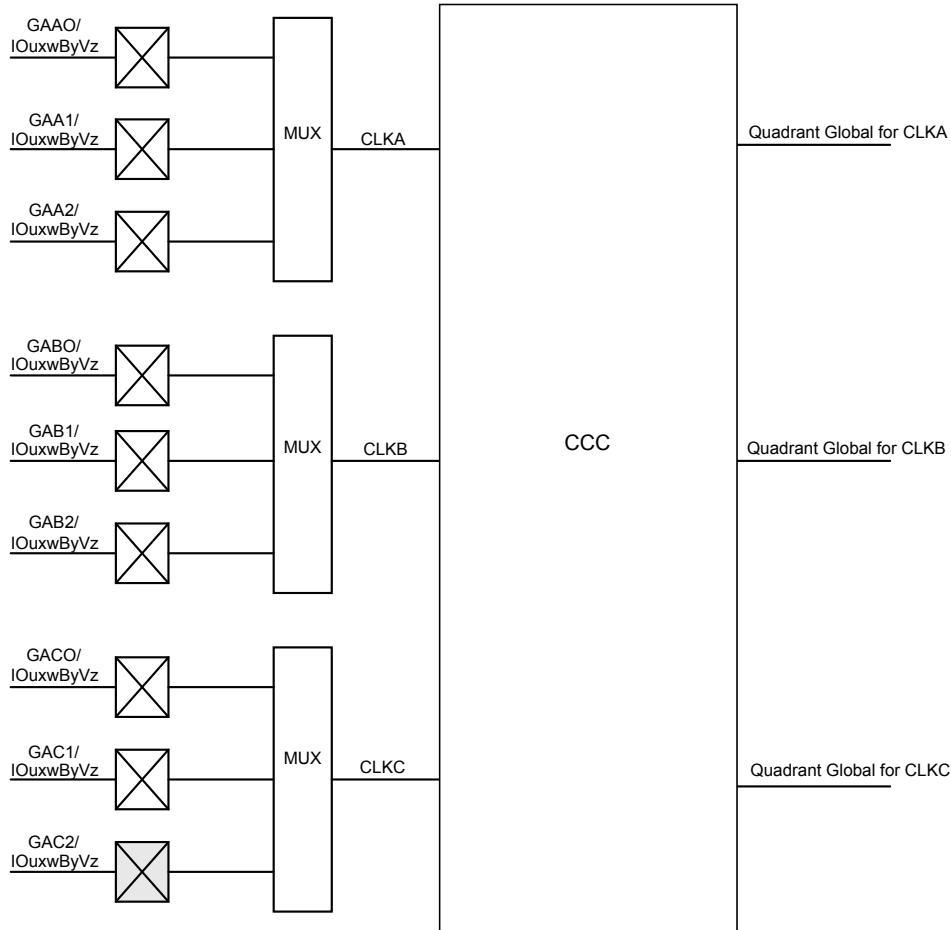


Figure 3-6 • Global Inputs

Since each bank can have a different I/O standard, the user should be careful to choose the correct global I/O for the design. There are 54 global pins available to access 18 global networks. For the single-ended and voltage-referenced I/O standards, you can use any of these three available I/Os to access the global network. For differential I/O standards such as LVDS and LVPECL, the I/O macro needs to be placed on (A0, A1), (B0, B1), (C0, C1), or a similar location. The unassigned global I/Os can be used as regular I/Os. Note that pin names starting with GF and GC are associated with the chip global networks, and GA, GB, GD, and GE are used for quadrant global networks. Table 3-2 on page 38 and Table 3-3 on page 39 show the general chip and quadrant global pin names.

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 ($T_a = +125^\circ\text{C}$, $V_{CCA} = 1.4\text{ V}$) | Typical ($T_a = +25^\circ\text{C}$, $V_{CCA} = 1.5\text{ V}$) | Maximum ($T_a = -55^\circ\text{C}$, $V_{CCA} = 1.6\text{ V}$) |
|------------------------|--|---|---|
| –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.

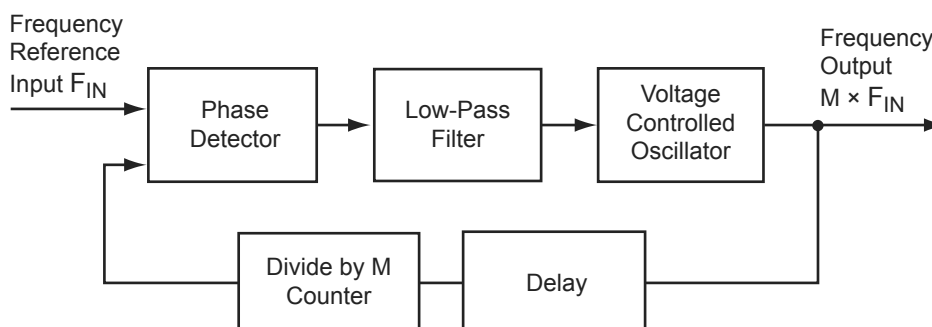


Figure 4-19 • Simplified PLL Core with Feedback Divider and Delay

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 85) is increased, the average phase difference increases. The average phase

Dividers n and m (the input divider and feedback divider, respectively) provide integer frequency division factors from 1 to 128. The output dividers u , v , and w provide integer division factors from 1 to 32. Frequency scaling of the reference clock CLKA is performed according to the following formulas:

$$f_{GLA} = f_{CLKA} \times m / (n \times u) - \text{GLA Primary PLL Output Clock} \quad \text{EQ 4-1}$$

$$f_{GLB} = f_{YB} = f_{CLKA} \times m / (n \times v) - \text{GLB Secondary 1 PLL Output Clock(s)} \quad \text{EQ 4-2}$$

$$f_{GLC} = f_{YC} = f_{CLKA} \times m / (n \times w) - \text{GLC Secondary 2 PLL Output Clock(s)} \quad \text{EQ 4-3}$$

SmartGen provides a user-friendly method of generating the configured PLL netlist, which includes automatically setting the division factors to achieve the closest possible match to the requested frequencies. Since the five output clocks share the n and m dividers, the achievable output frequencies are interdependent and related according to the following formula:

$$f_{GLA} = f_{GLB} \times (v / u) = f_{GLC} \times (w / u) \quad \text{EQ 4-4}$$

Clock Delay Adjustment

There are a total of seven configurable delay elements implemented in the PLL architecture.

Two of the delays are located in the feedback path, entitled System Delay and Feedback Delay. System Delay provides a fixed delay of 2 ns (typical), and Feedback Delay provides selectable delay values from 0.6 ns to 5.56 ns in 160 ps increments (typical). For PLLs, delays in the feedback path will effectively advance the output signal from the PLL core with respect to the reference clock. Thus, the System and Feedback delays generate negative delay on the output clock. Additionally, each of these delays can be independently bypassed if necessary.

The remaining five delays perform traditional time delay and are located at each of the outputs of the PLL. Besides the fixed global driver delay of 0.755 ns for each of the global networks, the global multiplexer outputs (GLA, GLB, and GLC) each feature an additional selectable delay value, as given in Table 4-7.

Table 4-7 • Delay Values in Libero SoC Software per Device Family

| Device | Typical | Starting Values | Increments | Ending Value |
|-----------------------|---------|-----------------|------------|--------------|
| ProASIC3 | 200 ps | 0 to 735 ps | 200 ps | 6.735 ns |
| IGLOO/ProASIC3L 1.5 V | 360 ps | 0 to 1.610 ns | 360 ps | 12.410 ns |
| IGLOO/ProASIC3L 1.2 V | 580 ps | 0 to 2.880 ns | 580 ps | 20.280 ns |

The additional YB and YC signals have access to a selectable delay from 0.6 ns to 5.56 ns in 160 ps increments (typical). This is the same delay value as the CLKDLY macro. It is similar to CLKDLY, which bypasses the PLL core just to take advantage of the phase adjustment option with the delay value.

The following parameters must be taken into consideration to achieve minimum delay at the outputs (GLA, GLB, GLC, YB, and YC) relative to the reference clock: routing delays from the PLL core to CCC outputs, core outputs and global network output delays, and the feedback path delay. The feedback path delay acts as a time advance of the input clock and will offset any delays introduced beyond the PLL core output. The routing delays are determined from back-annotated simulation and are configuration-dependent.

Table 4-18 • Fusion Dynamic CCC Division by Half Configuration

| OADIVHALF / OBDIVHALF / OCDIVHALF | OADIV<4:0> / OBDIV<4:0> / OCDIV<4:0> (in decimal) | Divider Factor | Input Clock Frequency | Output Clock Frequency (MHz) |
|---|--|----------------|--------------------------|--------------------------------------|
| 1 | 2 | 1.5 | 100 MHz RC Oscillator | 66.7 |
| | 4 | 2.5 | | 40.0 |
| | 6 | 3.5 | | 28.6 |
| | 8 | 4.5 | | 22.2 |
| | 10 | 5.5 | | 18.2 |
| | 12 | 6.5 | | 15.4 |
| | 14 | 7.5 | | 13.3 |
| | 16 | 8.5 | | 11.8 |
| | 18 | 9.5 | | 10.5 |
| | 20 | 10.5 | | 9.5 |
| | 22 | 11.5 | | 8.7 |
| | 24 | 12.5 | | 8.0 |
| | 26 | 13.5 | | 7.4 |
| | 28 | 14.5 | | 6.9 |
| 0 | 0–31 | 1–32 | Other Clock Sources | Depends on other divider settings |

Table 4-19 • Configuration Bit <76:75> / VCOSEL<2:1> Selection for All Families

| Voltage | VCOSEL[2:1] | | | | | | | |
|--|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| | 00 | | 01 | | 10 | | 11 | |
| | Min. (MHz) | Max. (MHz) | Min. (MHz) | Max. (MHz) | Min. (MHz) | Max. (MHz) | Min. (MHz) | Max. (MHz) |
| IGLOO and IGLOO PLUS | | | | | | | | |
| 1.2 V ± 5% | 24 | 35 | 30 | 70 | 60 | 140 | 135 | 160 |
| 1.5 V ± 5% | 24 | 43.75 | 30 | 87.5 | 60 | 175 | 135 | 250 |
| ProASIC3L, RT ProASIC3, and Military ProASIC3/L | | | | | | | | |
| 1.2 V ± 5% | 24 | 35 | 30 | 70 | 60 | 140 | 135 | 250 |
| 1.5 V ± 5% | 24 | 43.75 | 30 | 70 | 60 | 175 | 135 | 350 |
| ProASIC3 and Fusion | | | | | | | | |
| 1.5 V ± 5% | 24 | 43.75 | 33.75 | 87.5 | 67.5 | 175 | 135 | 350 |

Table 4-20 • Configuration Bit <74> / VCOSEL<0> Selection for All Families

| VCOSEL[0] | Description |
|-----------|---|
| 0 | Fast PLL lock acquisition time with high tracking jitter. Refer to the corresponding datasheet for specific value and definition. |
| 1 | Slow PLL lock acquisition time with low tracking jitter. Refer to the corresponding datasheet for specific value and definition. |

Dynamic PLL Configuration

To generate a dynamically reconfigurable CCC, the user should select **Dynamic CCC** in the configuration section of the SmartGen GUI (Figure 4-26). This will generate both the CCC core and the configuration shift register / control bit MUX.

Figure 4-26 • SmartGen GUI

Even if dynamic configuration is selected in SmartGen, the user must still specify the static configuration data for the CCC (Figure 4-27). The specified static configuration is used whenever the MODE signal is set to LOW and the CCC is required to function in the static mode. The static configuration data can be used as the default behavior of the CCC where required.

Figure 4-27 • Dynamic CCC Configuration in SmartGen

Figure 4-36 • Second-Stage PLL Showing Input of 256 MHz from First Stage and Final Output of 280 MHz

Figure 4-37 shows the simulation results, where the first PLL's output period is 3.9 ns (~256 MHz), and the stage 2 (final) output period is 3.56 ns (~280 MHz).

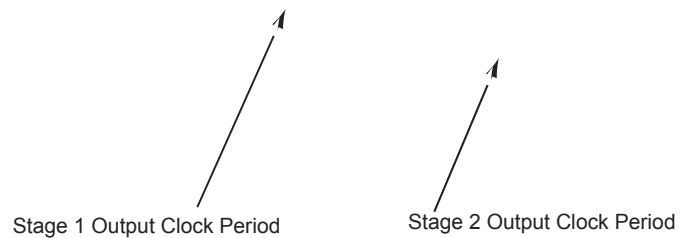


Figure 4-37 • ModelSim Simulation Results

FlashROM Design Flow

The Microsemi Libero System-on-Chip (SoC) software has extensive FlashROM support, including FlashROM generation, instantiation, simulation, and programming. Figure 5-9 shows the user flow diagram. In the design flow, there are three main steps:

1. FlashROM generation and instantiation in the design
2. Simulation of FlashROM design
3. Programming file generation for FlashROM design

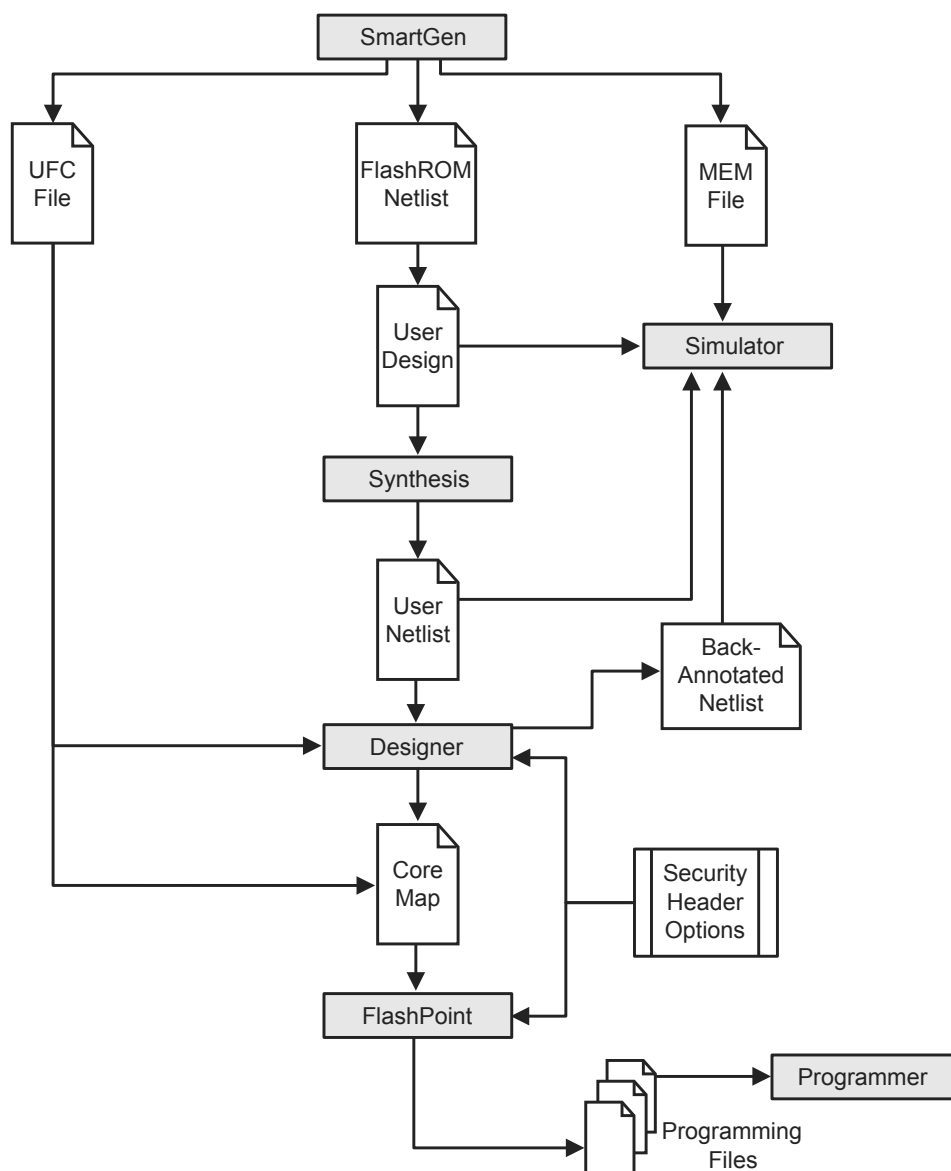


Figure 5-9 • FlashROM Design Flow

FlashROM Generation and Instantiation in the Design

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

1. **Static Fixed Data**—Enables you to fix the data so it cannot be changed during programming time. This option is useful when you have fixed data stored in this region, which is required for the operation of the design in the FPGA. Key storage is one example.
 2. **Static Modifiable Data**—Select this option when the data in a particular region is expected to be static data (such as a version number, which remains the same for a long duration but could conceivably change in the future). This option enables you to avoid changing the value every time you enter new data.
 3. **Read from File**—This provides the full flexibility of FlashROM usage to the customer. If you have a customized algorithm for generating the FlashROM data, you can specify this setting. You can then generate a text file with data for as many devices as you wish to program, and load that into the FlashPoint programming file generation software to get programming files that include all the data. SmartGen will optionally pass the location of the file where the data is stored if the file is specified in SmartGen. Each text file has only one type of data format (binary, decimal, hex, or ASCII text). The length of each data file must be shorter than or equal to the selected region length. If the data is shorter than the selected region length, the most significant bits will be padded with 0s. For multiple text files for multiple regions, the first lines are for the first device. In SmartGen, **Load Sim. Value From File** allows you to load the first device data in the MEM file for simulation.
 4. **Auto Increment/Decrement**—This scenario is useful when you specify the contents of FlashROM for a large number of devices in a series. You can specify the step value for the serial number and a maximum value for inventory control. During programming file generation, the actual number of devices to be programmed is specified and a start value is fed to the software.
-

Figure 5-10 • SmartGen GUI of the FlashROM

Table 6-2 • Allowable Aspect Ratio Settings for WIDTHA[1:0]

| WIDTHA[1:0] | WIDTHB[1:0] | D×W |
|-------------|-------------|-------|
| 00 | 00 | 4k×1 |
| 01 | 01 | 2k×2 |
| 10 | 10 | 1k×4 |
| 11 | 11 | 512×9 |

Note: The aspect ratio settings are constant and cannot be changed on the fly.

BLKA and BLKB

These signals are active-low and will enable the respective ports when asserted. When a BLKx signal is deasserted, that port's outputs hold the previous value.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, BLKB should be tied to ground.

WENA and WENB

These signals switch the RAM between read and write modes for the respective ports. A LOW on these signals indicates a write operation, and a HIGH indicates a read.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, WENB should be tied to ground.

CLKA and CLKB

These are the clock signals for the synchronous read and write operations. These can be driven independently or with the same driver.

Note: For Automotive ProASIC3 devices, 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). 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.

PIPEA and PIPEB

These signals are used to specify pipelined read on the output. A LOW on PIPEA or PIPEB indicates a nonpipelined read, and the data appears on the corresponding output in the same clock cycle. A HIGH indicates a pipelined read, and data appears on the corresponding output in the next clock cycle.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, PIPEB should be tied to ground. 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.

WMODEA and WMODEB

These signals are used to configure the behavior of the output when the RAM is in write mode. A LOW on these signals makes the output retain data from the previous read. A HIGH indicates pass-through behavior, wherein the data being written will appear immediately on the output. This signal is overridden when the RAM is being read.

Note: When using the SRAM in single-port mode for Automotive ProASIC3 devices, WMODEB should be tied to ground.

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.

ADDRA and ADDRb

These are used as read or write addresses, and they are 12 bits wide. When a depth of less than 4 k is specified, the unused high-order bits must be grounded (Table 6-3 on page 139).

- The I/O standard of technology-specific I/O macros cannot be changed in the I/O Attribute Editor (see Figure 8-6).
- The user **MUST** instantiate differential I/O macros (LVDS/LVPECL) in the design. This is the only way to use these standards in the design (IGLOO nano and ProASIC3 nano devices do not support differential inputs).
- To implement the DDR I/O function, the user must instantiate a DDR_REG or DDR_OUT macro. This is the only way to use a DDR macro in the design.

Figure 8-6 • Assigning a Different I/O Standard to the Generic I/O Macro

Performing Place-and-Route on the Design

The netlist created by the synthesis tool should now be imported into Designer and compiled. During Compile, the user can specify the I/O placement and attributes by importing the PDC file. The user can also specify the I/O placement and attributes using ChipPlanner and the I/O Attribute Editor under MVN.

Defining I/O Assignments in the PDC File

A PDC file is a Tcl script file specifying physical constraints. This file can be imported to and exported from Designer.

Table 8-3 shows I/O assignment constraints supported in the PDC file.

Table 8-3 • PDC I/O Constraints

| Command | Action | Example | Comment |
|--------------------------------------|--|--|--|
| I/O Banks Setting Constraints | | | |
| set_iobank | Sets the I/O supply voltage, V_{CCI} , and the input reference voltage, V_{REF} , for the specified I/O bank. | <pre>set_iobank bankname [-vcci vcci_voltage] [-vref vref_voltage] set_iobank Bank7 -vcci 1.50 -vref 0.75</pre> | Must use in case of mixed I/O voltage (V_{CCI}) design |
| set_vref | Assigns a V_{REF} pin to a bank. | <pre>set_vref -bank [bankname] [pinnum] set_vref -bank Bank0 685 704 723 742 761</pre> | Must use if voltage-referenced I/Os are used |
| set_vref_defaults | Sets the default V_{REF} pins for the specified bank. This command is ignored if the bank does not need a V_{REF} pin. | <pre>set_vref_defaults bankname set_vref_defaults bank2</pre> | |

Note: Refer to the Libero SoC User's Guide for detailed rules on PDC naming and syntax conventions.

9 – DDR for Microsemi's Low Power Flash Devices

Introduction

The I/Os in Fusion, IGLOO, and ProASIC3 devices support Double Data Rate (DDR) mode. In this mode, new data is present on every transition (or clock edge) of the clock signal. This mode doubles the data transfer rate compared with Single Data Rate (SDR) mode, where new data is present on one transition (or clock edge) of the clock signal. Low power flash devices have DDR circuitry built into the I/O tiles. I/Os are configured to be DDR receivers or transmitters by instantiating the appropriate special macros (examples shown in Figure 9-4 on page 210 and Figure 9-5 on page 211) and buffers (DDR_OUT or DDR_REG) in the RTL design. This document discusses the options the user can choose to configure the I/Os in this mode and how to instantiate them in the design.

Double Data Rate (DDR) Architecture

Low power flash devices support 350 MHz DDR inputs and outputs. In DDR mode, new data is present on every transition of the clock signal. Clock and data lines have identical bandwidths and signal integrity requirements, making them very efficient for implementing very high-speed systems. High-speed DDR interfaces can be implemented using LVDS (not applicable for IGLOO nano and ProASIC3 nano devices). In IGLOOe, ProASIC3E, AFS600, and AFS1500 devices, DDR interfaces can also be implemented using the HSTL, SSTL, and LVPECL I/O standards. The DDR feature is primarily implemented in the FPGA core periphery and is not tied to a specific I/O technology or limited to any I/O standard.

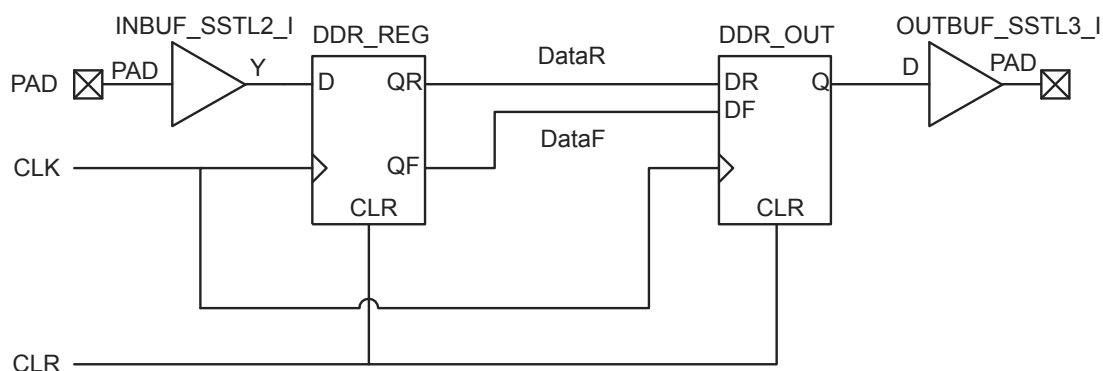


Figure 9-1 • DDR Support in Low Power Flash Devices

10 – Programming Flash Devices

Introduction

This document provides an overview of the various programming options available for the Microsemi flash families. The electronic version of this document includes active links to all programming resources, which are available at <http://www.microsemi.com/soc/products/hardware/default.aspx>. For Microsemi antifuse devices, refer to the *Programming Antifuse Devices* document.

Summary of Programming Support

FlashPro4 and FlashPro3 are high-performance in-system programming (ISP) tools targeted at the latest generation of low power flash devices offered by the SmartFusion,[®] Fusion,[®] IGLOO,[®] and ProASIC[®]3 families, including ARM-enabled devices. FlashPro4 and FlashPro3 offer extremely high performance through the use of USB 2.0, are high-speed compliant for full use of the 480 Mbps bandwidth, and can program ProASIC3 devices in under 30 seconds. Powered exclusively via USB, FlashPro4 and FlashPro3 provide a VPUMP voltage of 3.3 V for programming these devices.

FlashPro4 replaced FlashPro3 in 2010. FlashPro4 supports SmartFusion, Fusion, ProASIC3, and IGLOO devices as well as future generation flash devices. FlashPro4 also adds 1.2 V programming for IGLOO nano V2 devices. FlashPro4 is compatible with FlashPro3; however it adds a programming mode (PROG_MODE) signal to the previously unused pin 4 of the JTAG connector. The PROG_MODE goes high during programming and can be used to turn on a 1.5 V external supply for those devices that require 1.5 V for programming. If both FlashPro3 and FlashPro4 programmers are used for programming the same boards, pin 4 of the JTAG connector must not be connected to anything on the board because FlashPro4 uses pin 4 for PROG_MODE.

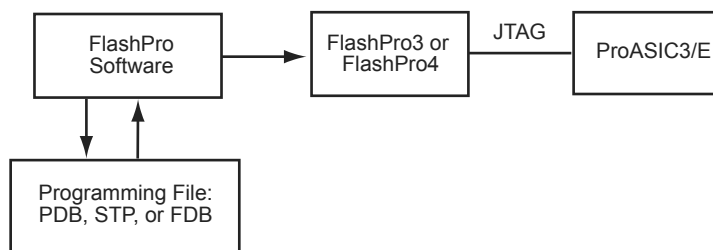


Figure 10-1 • FlashPro Programming Setup

Types of Programming for Flash Devices

The number of devices to be programmed will influence the optimal programming methodology. Those available are listed below:

- In-system programming
 - Using a programmer
 - Using a microprocessor or microcontroller
- Device programmers
 - Single-site programmers
 - Multi-site programmers, batch programmers, or gang programmers
 - Automated production (robotic) programmers
- Volume programming services
 - Microsemi in-house programming
 - Programming centers

In-System Programming

Device Type Supported: Flash

ISP refers to programming the FPGA after it has been mounted on the system printed circuit board. The FPGA may be preprogrammed and later reprogrammed using ISP.

The advantage of using ISP is the ability to update the FPGA design many times without any changes to the board. This eliminates the requirement of using a socket for the FPGA, saving cost and improving reliability. It also reduces programming hardware expenses, as the ISP methodology is die-/package-independent.

There are two methods of in-system programming: external and internal.

- Programmer ISP—Refer to the "In-System Programming (ISP) of Microsemi's Low Power Flash Devices Using FlashPro4/3/3X" section on page 261 for more information.

Using an external programmer and a cable, the device can be programmed through a header on the system board. In Microsemi SoC Products Group documentation, this is referred to as external ISP. Microsemi provides FlashPro4, FlashPro3, FlashPro Lite, or Silicon Sculptor 3 to perform external ISP. Note that Silicon Sculptor II and Silicon Sculptor 3 can only provide ISP for ProASIC and ProASIC^{PLUS}® families, not for SmartFusion, Fusion, IGLOO, or ProASIC3. Silicon Sculptor II and Silicon Sculptor 3 can be used for programming ProASIC and ProASIC^{PLUS} devices by using an adapter module (part number SMPA-ISP-ACTEL-3).

 - Advantages: Allows local control of programming and data files for maximum security. The programming algorithms and hardware are available from Microsemi. The only hardware required on the board is a programming header.
 - Limitations: A negligible board space requirement for the programming header and JTAG signal routing
- Microprocessor ISP—Refer to the "Microprocessor Programming of Microsemi's Low Power Flash Devices" chapter of an appropriate FPGA fabric user's guide for more information.

Using a microprocessor and an external or internal memory, you can store the program in memory and use the microprocessor to perform the programming. In Microsemi documentation, this is referred to as internal ISP. Both the code for the programming algorithm and the FPGA programming file must be stored in memory on the board. Programming voltages must also be generated on the board.

 - Advantages: The programming code is stored in the system memory. An external programmer is not required during programming.
 - Limitations: This is the approach that requires the most design work, since some way of getting and/or storing the data is needed; a system interface to the device must be designed; and the low-level API to the programming firmware must be written and linked into the code provided by Microsemi. While there are benefits to this methodology, serious thought and planning should go into the decision.

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

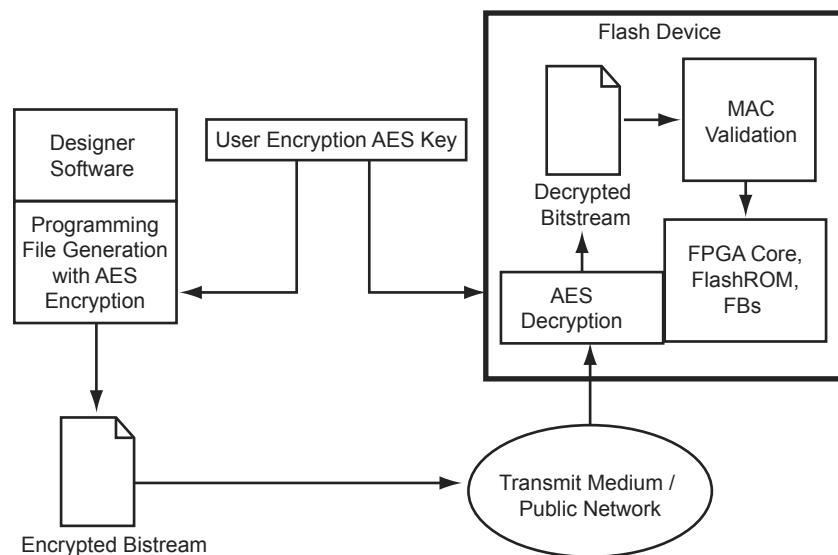


Figure 12-1 • AES-128 Security Features

Circuit Description

All IGLOO devices as well as the ProASIC3L product family are available in two versions: V5 devices, which are powered by a 1.5 V supply and V2 devices, which are powered by a supply anywhere in the range of 1.2 V to 1.5 V in 50 mV increments. Applications that use IGLOO or ProASIC3L devices powered by a 1.2 V core supply must have a mechanism that switches the core voltage from 1.2 V (or other voltage below 1.5 V) to 1.5 V during in-system programming (ISP). There are several possible techniques to meet this requirement. Microsemi recommends utilizing a linear voltage regulator, a resistor voltage divider, and an N-Channel Digital FET to set the appropriate VCC voltage, as shown in Figure 13-1.

Where 1.2 V is mentioned in the following text, the meaning applies to any voltage below the 1.5 V range. Resistor values in the figures have been calculated for 1.2 V, so refer to power regulator datasheets if a different core voltage is required.

The main component of Microsemi's recommended circuit is the LTC3025 linear voltage regulator from LinearTech. The output voltage of the LTC3025 on the OUT pin is set by the ratio of two external resistors, R37 and R38, in a voltage divider. The linear voltage regulator adjusts the voltage on the OUT pin to maintain the ADJ pin voltage at 0.4 V (referenced to ground). By using an R38 value of 40.2 k Ω and an R37 value of 80.6 k Ω , the output voltage on the OUT pin is 1.2 V. To achieve 1.5 V on the OUT pin, R44 can be used in parallel with R38. The OUT pin can now be used as a switchable source for the VCC supply. Refer to the *LTC3025 Linear Voltage Regulator datasheet* for more information.

In Figure 13-1, the N-Channel Digital FET is used to enable and disable R44. This FET is controlled by the JTAG TRST signal driven by the FlashPro3 programmer. During programming of the device, the TRST signal is driven HIGH by the FlashPro3, and turns the N-Channel Digital FET ON. When the FET is ON, R44 becomes enabled as a parallel resistance to R38, which forces the regulator to set OUT to 1.5 V.

When the FlashPro3 is connected and not in programming mode or when it is not connected, the pull-down resistor, R10, will pull the TRST signal LOW. When this signal is LOW, the N-Channel Digital FET is "open" and R44 is not part of the resistance seen by the LTC3025. The new resistance momentarily changes the voltage value on the ADJ pin, which in turn causes the output of the LTC3025 to compensate by setting OUT to 1.2 V. Now the device will run in regular active mode at the regular 1.2 V core voltage.

Figure 13-1 • Circuit Diagram

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 |
| v1.1 (October 2008) | The "Introduction" was revised to include information about the core supply voltage range of operation in V2 devices. | 275 |
| | IGLOO nano device support was added to Table 13-1 • Flash-Based FPGAs Supporting Voltage Switching Circuit. | 276 |
| | The "Circuit Description" section was updated to include IGLOO PLUS core operation from 1.2 V to 1.5 V in 50 mV increments. | 277 |
| v1.0 (August 2008) | The "Microsemi's Flash Families Support Voltage Switching Circuit" section was revised to include new families and make the information more concise. | 276 |



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

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at www.microsemi.com.

© 2012 Microsemi Corporation. All rights reserved. Microsemi and the Microsemi logo are trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.