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	-
Number of I/O	49
Number of Gates	30000
Voltage - Supply	1.425V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-20°C ~ 85°C (TJ)
Package / Case	68-VFQFN Exposed Pad
Supplier Device Package	68-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/a3pn030-z2qng68

2 – Low Power Modes in ProASIC3/E and ProASIC3 nano FPGAs

Introduction

The demand for low power systems and semiconductors, combined with the strong growth observed for value-based FPGAs, is driving growing demand for low power FPGAs. For portable and battery-operated applications, power consumption has always been the greatest challenge. The battery life of a system and on-board devices has a direct impact on the success of the product. As a result, FPGAs used in these applications should meet low power consumption requirements.

ProASIC$^{®}$3/E and ProASIC3 nano FPGAs offer low power consumption capability inherited from their nonvolatile and live-at-power-up (LAPU) flash technology. This application note describes the power consumption and how to use different power saving modes to further reduce power consumption for power-conscious electronics design.

Power Consumption Overview

In evaluating the power consumption of FPGA technologies, it is important to consider it from a system point of view. Generally, the overall power consumption should be based on static, dynamic, inrush, and configuration power. Few FPGAs implement ways to reduce static power consumption utilizing sleep modes.

SRAM-based FPGAs use volatile memory for their configuration, so the device must be reconfigured after each power-up cycle. Moreover, during this initialization state, the logic could be in an indeterminate state, which might cause inrush current and power spikes. More complex power supplies are required to eliminate potential system power-up failures, resulting in higher costs. For portable electronics requiring frequent power-up and -down cycles, this directly affects battery life, requiring more frequent recharging or replacement.

SRAM-Based FPGA Total Power Consumption = $P_{static} + P_{dynamic} + P_{inrush} + P_{config}$

EQ 1

ProASIC3/E Total Power Consumption = $P_{static} + P_{dynamic}$

EQ 2

Unlike SRAM-based FPGAs, Microsemi flash-based FPGAs are nonvolatile and do not require power-up configuration. Additionally, Microsemi nonvolatile flash FPGAs are live at power-up and do not require additional support components. Total power consumption is reduced as the inrush current and configuration power components are eliminated.

Note that the static power component can be reduced in flash FPGAs (such as the ProASIC3/E devices) by entering User Low Static mode or Sleep mode. This leads to an extremely low static power component contribution to the total system power consumption.

The following sections describe the usage of Static (Idle) mode to reduce the power component, User Low Static mode to reduce the static power component, and Sleep mode and Shutdown mode to achieve a range of power consumption when the FPGA or system is idle. Table 2-1 on page 22 summarizes the different low power modes offered by ProASIC3/E devices.

standard for CLKBUF is LVTTL in the current Microsemi Libero® System-on-Chip (SoC) and Designer software.

Table 3-9 • **I/O Standards within CLKBUF**

Name	Description
CLKBUF_LVCMOS5	LVCMOS clock buffer with 5.0 V CMOS voltage level
CLKBUF_LVCMOS33	LVCMOS clock buffer with 3.3 V CMOS voltage level
CLKBUF_LVCMOS25	LVCMOS clock buffer with 2.5 V CMOS voltage level[1]
CLKBUF_LVCMOS18	LVCMOS clock buffer with 1.8 V CMOS voltage level
CLKBUF_LVCMOS15	LVCMOS clock buffer with 1.5 V CMOS voltage level
CLKBUF_LVCMOS12	LVCMOS clock buffer with 1.2 V CMOS voltage level
CLKBUF_PCI	PCI clock buffer
CLKBUF_PCIX	PCIX clock buffer
CLKBUF_GTL25	GTL clock buffer with 2.5 V CMOS voltage level[1]
CLKBUF_GTL33	GTL clock buffer with 3.3 V CMOS voltage level[1]
CLKBUF_GTLP25	GTL+ clock buffer with 2.5 V CMOS voltage level[1]
CLKBUF_GTLP33	GTL+ clock buffer with 3.3 V CMOS voltage level[1]
CLKBUF_ HSTL _I	HSTL Class I clock buffer[1]
CLKBUF_ HSTL _II	HSTL Class II clock buffer[1]
CLKBUF_SSTL2_I	SSTL2 Class I clock buffer[1]
CLKBUF_SSTL2_II	SSTL2 Class II clock buffer[1]
CLKBUF_SSTL3_I	SSTL3 Class I clock buffer[1]
CLKBUF_SSTL3_II	SSTL3 Class II clock buffer[1]

Notes:

1. *Supported in only the IGLOOe, ProASIC3E, AFS600, and AFS1500 devices*

2. *By default, the CLKBUF macro uses the 3.3 V LVTTL I/O technology.*

The current synthesis tool libraries only infer the CLKBUF or CLKINT macros in the netlist. All other global macros must be instantiated manually into your HDL code. The following is an example of CLKBUF_LVCMOS25 global macro instantiations that you can copy and paste into your code:

VHDL

```
component clkbuf_lvcmos25
  port (pad : in std_logic; y : out std_logic);
end component

begin
-- concurrent statements
u2 : clkbuf_lvcmos25 port map (pad =>  ext_clk, y => int_clk);
end
```

Verilog

```
module design (_____);

input _____;
output _____;

clkbuf_lvcmos25 u2 (.y(int_clk), .pad(ext_clk);

endmodule
```

You can control the maximum number of shared instances allowed for the legalization to take place using the Compile Option dialog box shown in Figure 3-17. Refer to Libero SoC / Designer online help for details on the Compile Option dialog box. A large number of shared instances most likely indicates a floorplanning problem that you should address.

Figure 3-17 • **Shared Instances in the Compile Option Dialog Box**

Designer Flow for Global Assignment

To achieve the desired result, pay special attention to global management during synthesis and place-and-route. The current Synplify tool does not insert more than six global buffers in the netlist by default. Thus, the default flow will not assign any signal to the quadrant global network. However, you can use attributes in Synplify and increase the default global macro assignment in the netlist. Designer v6.2 supports automatic quadrant global assignment, which was not available in Designer v6.1. Layout will make the choice to assign the correct signals to global. However, you can also utilize PDC and perform manual global assignment to overwrite any automatic assignment. The following step-by-step suggestions guide you in the layout of your design and help you improve timing in Designer:

1. Run Compile and check the Compile report. The Compile report has global information in the "Device Utilization" section that describes the number of chip and quadrant signals in the design. A "Net Report" section describes chip global nets, quadrant global nets, local clock nets, a list of nets listed by fanout, and net candidates for local clock assignment. Review this information. Note that YB or YC are counted as global only when they are used in isolation; if you use YB only and not GLB, this net is not shown in the global/quadrant nets report. Instead, it appears in the Global Utilization report.

2. If some signals have a very high fanout and are candidates for global promotion, promote those signals to global using the compile options or PDC commands. Figure 3-18 on page 54 shows the Globals Management section of the compile options. Select **Promote regular nets whose fanout is greater than** and enter a reasonable value for fanouts.

CLKDLY Macro Usage

When a CLKDLY macro is used in a CCC location, the programmable delay element is used to allow the clock delays to go to the global network. In addition, the user can bypass the PLL in a CCC location integrated with a PLL, but use the programmable delay that is associated with the global network by instantiating the CLKDLY macro. The same is true when using programmable delay elements in a CCC location with no PLLs (the user needs to instantiate the CLKDLY macro). There is no difference between the programmable delay elements used for the PLL and the CLKDLY macro. The CCC will be configured to use the programmable delay elements in accordance with the macro instantiated by the user.

As an example, if the PLL is not used in a particular CCC location, the designer is free to specify up to three CLKDLY macros in the CCC, each of which can have its own input frequency and delay adjustment options. If the PLL core is used, assuming output to only one global clock network, the other two global clock networks are free to be used by either connecting directly from the global inputs or connecting from one or two CLKDLY macros for programmable delay.

The programmable delay elements are shown in the block diagram of the PLL block shown in Figure 4-6 on page 71. Note that any CCC locations with no PLL present contain only the programmable delay blocks going to the global networks (labeled "Programmable Delay Type 2"). Refer to the "Clock Delay Adjustment" section on page 86 for a description of the programmable delay types used for the PLL. Also refer to Table 4-14 on page 94 for Programmable Delay Type 1 step delay values, and Table 4-15 on page 94 for Programmable Delay Type 2 step delay values. CCC locations with a PLL present can be configured to utilize only the programmable delay blocks (Programmable Delay Type 2) going to the global networks A, B, and C.

Global network A can be configured to use only the programmable delay element (bypassing the PLL) if the PLL is not used in the design. Figure 4-6 on page 71 shows a block diagram of the PLL, where the programmable delay elements are used for the global networks (Programmable Delay Type 2).

This section outlines the following device information: CCC features, PLL core specifications, functional descriptions, software configuration information, detailed usage information, recommended board-level considerations, and other considerations concerning global networks in low power flash devices.

Clock Conditioning Circuits with Integrated PLLs

Each of the CCCs with integrated PLLs includes the following:

- 1 PLL core, which consists of a phase detector, a low-pass filter, and a four-phase voltage-controlled oscillator
- 3 global multiplexer blocks that steer signals from the global pads and the PLL core onto the global networks
- 6 programmable delays and 1 fixed delay for time advance/delay adjustments
- 5 programmable frequency divider blocks to provide frequency synthesis (automatically configured by the SmartGen macro builder tool)

Clock Conditioning Circuits without Integrated PLLs

There are two types of simplified CCCs without integrated PLLs in low power flash devices.

1. The simplified CCC with programmable delays, which is composed of the following:
   - 3 global multiplexer blocks that steer signals from the global pads and the programmable delay elements onto the global networks
   - 3 programmable delay elements to provide time delay adjustments
2. The simplified CCC (referred to as CCC-GL) without programmable delay elements, which is composed of the following:
   - A global multiplexer block that steer signals from the global pads onto the global networks

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**

	VCOSEL[2:1]							
	00		01		10		11	
Voltage	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.

Figure 4-34 • **Cascade PLL Configuration**

Using internal feedback, we know from EQ 4-1 on page 86 that the maximum achievable output frequency from the primary output is

$$f_{GLA} = f_{CLKA} \times m / (n \times u) = 2 \text{ MHz} \times 128 / (1 \times 1) = 256 \text{ MHz}$$

EQ 4-5

Figure 4-35 shows the settings of the initial PLL. When configuring the initial PLL, specify the input to be either Hardwired I/O–Driven or External I/O–Driven. This generates a netlist with the initial PLL routed from an I/O. Do not specify the input to be Core Logic–Driven, as this prohibits the connection from the I/O pin to the input of the PLL.
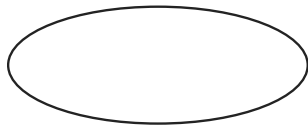

Figure 4-35 • **First-Stage PLL Showing Input of 2 MHz and Output of 256 MHz**

A second PLL can be connected serially to achieve the required frequency. EQ 4-1 on page 86 to EQ 4-3 on page 86 are extended as follows:

$$f_{GLA2} = f_{GLA} \times m_2 / (n_2 \times u_2) = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times u_1 \times n_2 \times u_2) - \text{Primary PLL Output Clock}$$

EQ 4-6

$$f_{GLB2} = f_{YB2} = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times n_2 \times v_1 \times v_2) - \text{Secondary 1 PLL Output Clock(s)}$$

EQ 4-7

$$f_{GLC2} = f_{YC2} = f_{CLKA1} \times m_1 \times m_2 / (n_1 \times n_2 \times w_1 \times w2) - \text{Secondary 2 PLL Output Clock(s)}$$

EQ 4-8

In the example, the final output frequency (f_{output}) from the primary output of the second PLL will be as follows (EQ 4-9):

$$f_{output} = f_{GLA2} = f_{GLA} \times m_2 / (n_2 \times u_2) = 256 \text{ MHz} \times 70 / (64 \times 1) = 280 \text{ MHz}$$

EQ 4-9

Figure 4-36 on page 111 shows the settings of the second PLL. When configuring the second PLL (or any subsequent-stage PLLs), specify the input to be Core Logic–Driven. This generates a netlist with the second PLL routed internally from the core. Do not specify the input to be Hardwired I/O–Driven or External I/O–Driven, as these options prohibit the connection from the output of the first PLL to the input of the second PLL.
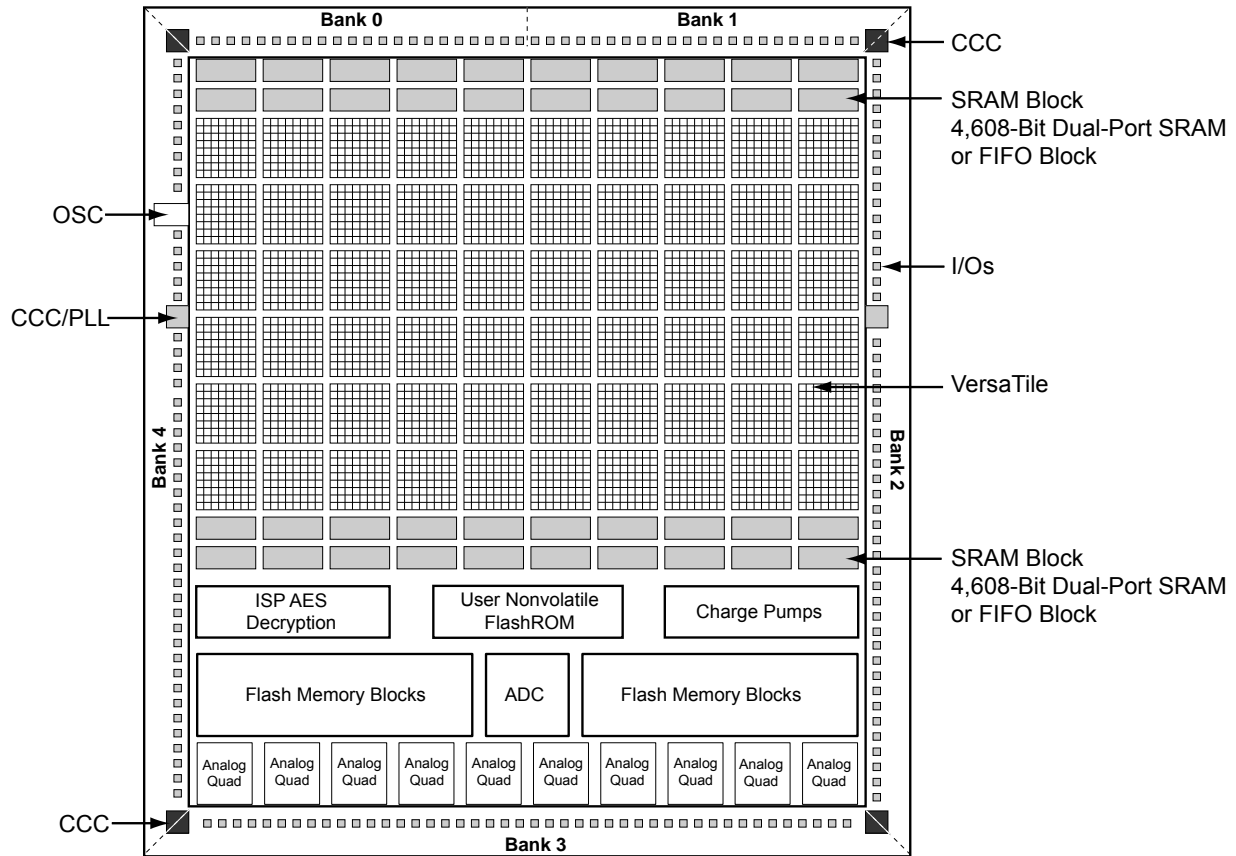
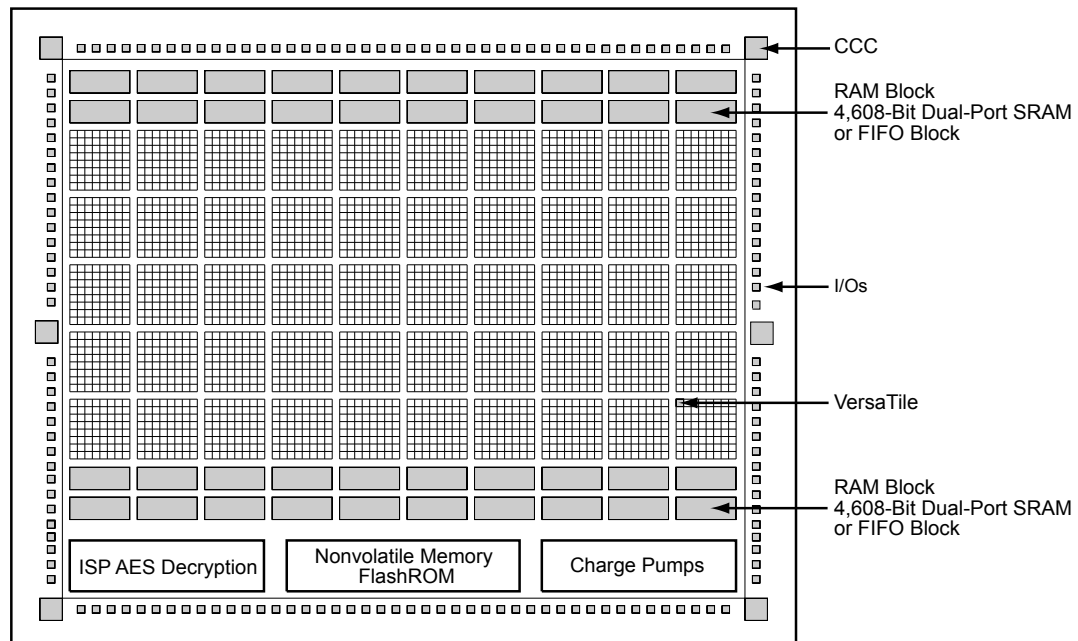*Figure 5-2 •* **Fusion Device Architecture Overview (AFS600)**
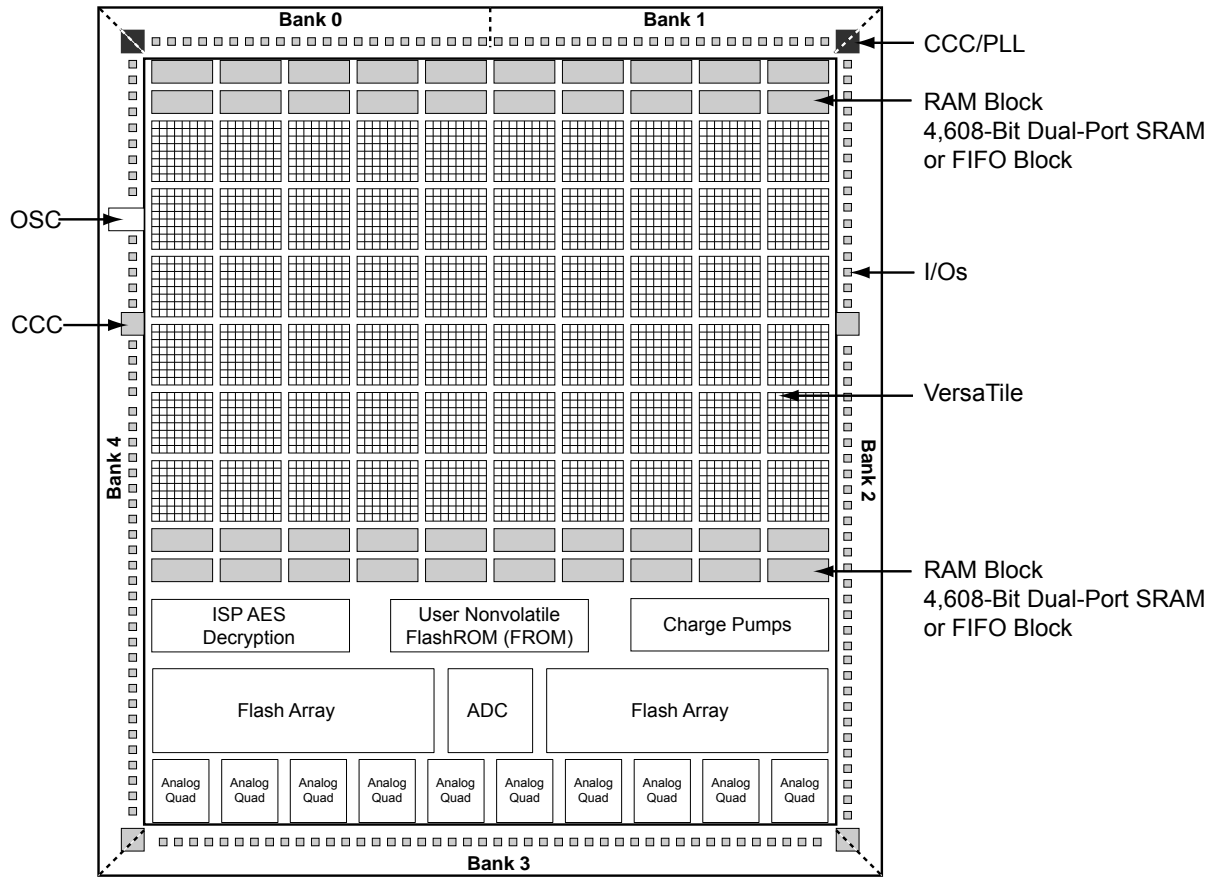

Figure 5-3 • **ProASIC3 and IGLOO Device Architecture**

Figure 6-2 • **Fusion Device Architecture Overview (AFS600)**

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

Table 6-3 • **Address Pins Unused/Used for Various Supported Bus Widths**

D×W	ADDRx	
	Unused	**Used**
4k×1	None	[11:0]
2k×2	[11]	[10:0]
1k×4	[11:10]	[9:0]
512×9	[11:9]	[8:0]

Note: The "x" in ADDRx implies A or B.

DINA and DINB

These are the input data signals, and they are nine bits wide. Not all nine bits are valid in all configurations. When a data width less than nine is specified, unused high-order signals must be grounded (Table 6-4).

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

DOUTA and DOUTB

These are the nine-bit output data signals. Not all nine bits are valid in all configurations. As with DINA and DINB, high-order bits may not be used (Table 6-4). The output data on unused pins is undefined.

Table 6-4 • **Unused/Used Input and Output Data Pins for Various Supported Bus Widths**

D×W	DINx/DOUTx	
	Unused	**Used**
4k×1	[8:1]	[0]
2k×2	[8:2]	[1:0]
1k×4	[8:4]	[3:0]
512×9	None	[8:0]

Note: The "x" in DINx or DOUTx implies A or B.

RAM512X18 Macro

RAM512X18 is the two-port configuration of the same RAM block (Figure 6-5 on page 140). Like the RAM4K9 nomenclature, the RAM512X18 nomenclature refers to both the deepest possible configuration and the widest possible configuration the two-port RAM block can assume. In two-port mode, the RAM block can be configured to either the 512×9 aspect ratio or the 256×18 aspect ratio. RAM512X18 is also fully synchronous and has the following features:

- Dedicated read and write ports
- Active-low read and write enables
- Selectable pipelined or nonpipelined read
- Active-low asynchronous reset
- Designer software will automatically facilitate falling-edge clocks by bubble-pushing the inversion to previous stages.

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**

Table 7-8 • **Hot-Swap Level 1**

Description	Cold-swap
Power Applied to Device	No
Bus State	–
Card Ground Connection	–
Device Circuitry Connected to Bus Pins	–
Example Application	System and card with Microsemi FPGA chip are powered down, and the card is plugged into the system. Then the power supplies are turned on for the system but not for the FPGA on the card.
Compliance of nano Devices	Compliant

Table 7-9 • **Hot-Swap Level 2**

Description	Hot-swap while reset
Power Applied to Device	Yes
Bus State	Held in reset state
Card Ground Connection	Reset must be maintained for 1 ms before, during, and after insertion/removal.
Device Circuitry Connected to Bus Pins	–
Example Application	In the PCI hot-plug specification, reset control circuitry isolates the card busses until the card supplies are at their nominal operating levels and stable.
Compliance of nano Devices	Compliant

Input Support for DDR

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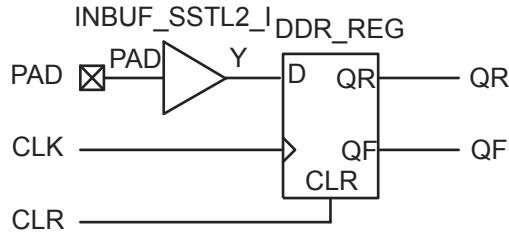

Figure 9-2 • **DDR Input Register Support in Low Power Flash Devices**

Output Support for DDR

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

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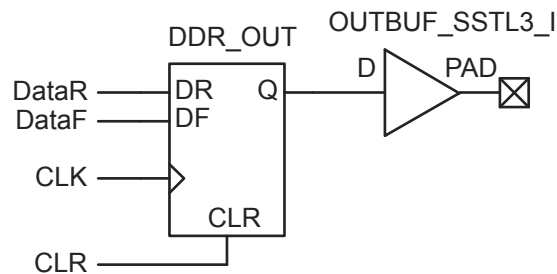

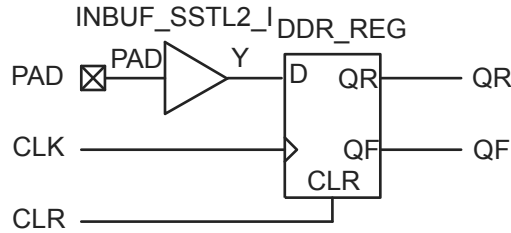*Figure 9-3* • **DDR Output Register (SSTL3 Class I)**

DDR Input Register

Figure 9-5 • **DDR Input Register (SSTL2 Class I)**

The corresponding structural representations, as generated by SmartGen, are shown below:

Verilog

```
module DDR_InBuf_SSTL2_I(PAD,CLR,CLK,QR,QF);

input   PAD, CLR, CLK;
output  QR, QF;

wire Y;

  INBUF_SSTL2_I INBUF_SSTL2_I_0_inst(.PAD(PAD),.Y(Y));
  DDR_REG DDR_REG_0_inst(.D(Y),.CLK(CLK),.CLR(CLR),.QR(QR),.QF(QF));

endmodule
```

VHDL

```
library ieee;
use ieee.std_logic_1164.all;
--The correct library will be inserted automatically by SmartGen
library proasic3; use proasic3.all;
--library fusion; use fusion.all;
--library igloo; use igloo.all;

entity DDR_InBuf_SSTL2_I is
  port(PAD, CLR, CLK : in std_logic;  QR, QF : out std_logic) ;
end DDR_InBuf_SSTL2_I;

architecture DEF_ARCH of  DDR_InBuf_SSTL2_I is

  component INBUF_SSTL2_I
    port(PAD : in std_logic := 'U'; Y : out std_logic) ;
  end component;

  component DDR_REG
    port(D, CLK, CLR : in std_logic := 'U'; QR, QF : out std_logic) ;
  end component;

signal Y : std_logic ;

begin

  INBUF_SSTL2_I_0_inst : INBUF_SSTL2_I
  port map(PAD => PAD, Y => Y);
  DDR_REG_0_inst : DDR_REG
  port map(D => Y, CLK => CLK, CLR => CLR, QR => QR, QF => QF);

end DEF_ARCH;
```

11 – Security in Low Power Flash Devices

Security in Programmable Logic

The need for security on FPGA programmable logic devices (PLDs) has never been greater than today. If the contents of the FPGA can be read by an external source, the intellectual property (IP) of the system is vulnerable to unauthorized copying. Fusion, IGLOO, and ProASIC3 devices contain state-of-the-art circuitry to make the flash-based devices secure during and after programming. Low power flash devices have a built-in 128-bit Advanced Encryption Standard (AES) decryption core (except for 30 k gate devices and smaller). The decryption core facilitates secure in-system programming (ISP) of the FPGA core array fabric, the FlashROM, and the Flash Memory Blocks (FBs) in Fusion devices. The FlashROM, Flash Blocks, and FPGA core fabric can be programmed independently of each other, allowing the FlashROM or Flash Blocks to be updated without the need for change to the FPGA core fabric.

Microsemi has incorporated the AES decryption core into the low power flash devices and has also included the Microsemi flash-based lock technology, FlashLock.® Together, they provide leading-edge security in a programmable logic device. Configuration data loaded into a device can be decrypted prior to being written to the FPGA core using the AES 128-bit block cipher standard. The AES encryption key is stored in on-chip, nonvolatile flash memory.

This document outlines the security features offered in low power flash devices, some applications and uses, as well as the different software settings for each application.

Figure 11-1 • **Overview on Security**

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]

Figure 11-10 • **All Silicon Features Selected for IGLOO and ProASIC3 Devices**

Figure 11-11 • **All Silicon Features Selected for Fusion**

Microprocessor Programming Support in Flash Devices

The flash-based FPGAs listed in Table 14-1 support programming with a microprocessor and the functions described in this document.

Table 14-1 • **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
	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 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
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

*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 14-1. Where the information applies to only one device 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 14-1. Where the information applies to only one device 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*.

Microsemi

17 – Power-Up/-Down Behavior of Low Power Flash Devices

Introduction

Microsemi's low power flash devices are flash-based FPGAs manufactured on a 0.13 µm process node. These devices offer a single-chip, reprogrammable solution and support Level 0 live at power-up (LAPU) due to their nonvolatile architecture.

Microsemi's low power flash FPGA families are optimized for logic area, I/O features, and performance. IGLOO® devices are optimized for power, making them the industry's lowest power programmable solution. IGLOO PLUS FPGAs offer enhanced I/O features beyond those of the IGLOO ultra-low power solution for I/O-intensive low power applications. IGLOO nano devices are the industry's lowest-power cost-effective solution. ProASIC3®L FPGAs balance low power with high performance. The ProASIC3 family is Microsemi's high-performance flash FPGA solution. ProASIC3 nano devices offer the lowest-cost solution with enhanced I/O capabilities.

Microsemi's low power flash devices exhibit very low transient current on each power supply during power-up. The peak value of the transient current depends on the device size, temperature, voltage levels, and power-up sequence.

The following devices can have inputs driven in while the device is not powered:

- IGLOO (AGL015 and AGL030)
- IGLOO nano (all devices)
- IGLOO PLUS (AGLP030, AGLP060, AGLP125)
- IGLOOe (AGLE600, AGLE3000)
- ProASIC3L (A3PE3000L)
- ProASIC3 (A3P015, A3P030)
- ProASIC3 nano (all devices)
- ProASIC3E (A3PE600, A3PE1500, A3PE3000)
- Military ProASIC3EL (A3PE600L, A3PE3000L, but not A3P1000)
- RT ProASIC3 (RT3PE600L, RT3PE3000L)

The driven I/Os do not pull up power planes, and the current draw is limited to very small leakage current, making them suitable for applications that require cold-sparing. These devices are hot-swappable, meaning they can be inserted in a live power system.[1]

---

1. For more details on the levels of hot-swap compatibility in Microsemi's low power flash devices, refer to the "Hot-Swap Support" section in the I/O Structures chapter of the FPGA fabric user's guide for the device you are using.