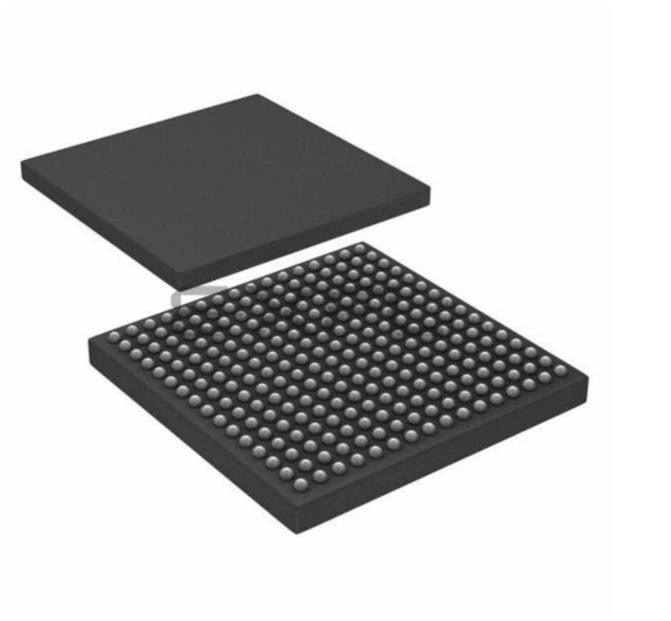
Microchip Technology - M1A3P1000L-1FG256I Datasheet



Welcome to E-XFL.COM

Understanding Embedded - FPGAs (Field Programmable Gate Array)

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

Applications of Embedded - FPGAs

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

Details	
Product Status	Active
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	147456
Number of I/O	177
Number of Gates	1000000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	-40°C ~ 100°C (TJ)
Package / Case	256-LBGA
Supplier Device Package	256-FPBGA (17x17)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/m1a3p1000l-1fg256i

YB and YC are identical to GLB and GLC, respectively, with the exception of a higher selectable final output delay. The SmartGen PLL Wizard will configure these outputs according to user specifications and can enable these signals with or without the enabling of Global Output Clocks.

The above signals can be enabled in the following output groupings in both internal and external feedback configurations of the static PLL:

- One output – GLA only
- Two outputs – GLA + (GLB and/or YB)
- Three outputs – GLA + (GLB and/or YB) + (GLC and/or YC)

PLL Macro Block Diagram

As illustrated, the PLL supports three distinct output frequencies from a given input clock. Two of these (GLB and GLC) can be routed to the B and C global network access, respectively, and/or routed to the device core (YB and YC).

There are five delay elements to support phase control on all five outputs (GLA, GLB, GLC, YB, and YC).

There are delay elements in the feedback loop that can be used to advance the clock relative to the reference clock.

The PLL macro reference clock can be driven in the following ways:

1. By an INBUF* macro to create a composite macro, where the I/O macro drives the global buffer (with programmable delay) using a hardwired connection. In this case, the I/O must be placed in one of the dedicated global I/O locations.
2. Directly from the FPGA core.
3. From an I/O that is routed through the FPGA regular routing fabric. In this case, users must instantiate a special macro, PLLINT, to differentiate from the hardwired I/O connection described earlier.

During power-up, the PLL outputs will toggle around the maximum frequency of the voltage-controlled oscillator (VCO) gear selected. Toggle frequencies can range from 40 MHz to 250 MHz. This will continue as long as the clock input (CLKA) is constant (HIGH or LOW). This can be prevented by LOW assertion of the POWERDOWN signal.

The visual PLL configuration in SmartGen, a component of the Libero SoC and Designer tools, will derive the necessary internal divider ratios based on the input frequency and desired output frequencies selected by the user.

Core Logic Clock Source

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

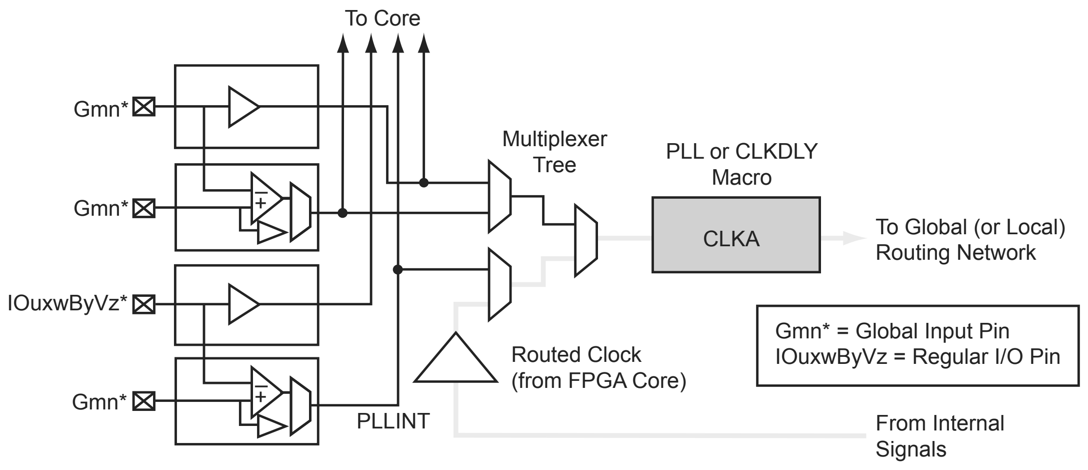


Figure 4-12 **• Illustration of Core Logic Usage**

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

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

External Feedback Configuration

For certain applications, such as those requiring generation of PCB clocks that must be matched with existing board delays, it is useful to implement an external feedback, EXTFB. The Phase Detector of the PLL core will receive CLKA and EXTFB as inputs. EXTFB may be processed by the fixed System Delay element as well as the *M* divider element. The EXTFB option is currently not supported.

After setting all the required parameters, users can generate one or more PLL configurations with HDL or EDIF descriptions by clicking the **Generate** button. SmartGen gives the option of saving session results and messages in a log file:

```
****************
Macro Parameters
****************

Name                            : test_pll
Family                          : ProASIC3E
Output Format                   : VHDL
Type                            : Static PLL
Input Freq(MHz)                 : 10.000
CLKA Source                     : Hardwired I/O
Feedback Delay Value Index      : 1
Feedback Mux Select             : 2
XDLY Mux Select                 : No
Primary Freq(MHz)               : 33.000
Primary PhaseShift              : 0
Primary Delay Value Index       : 1
Primary Mux Select              : 4
Secondary1 Freq(MHz)            : 66.000
Use GLB                         : YES
Use YB                          : YES
GLB Delay Value Index           : 1
YB Delay Value Index            : 1
Secondary1 PhaseShift           : 0
Secondary1 Mux Select           : 4
Secondary2 Freq(MHz)            : 101.000
Use GLC                         : YES
Use YC                          : NO
GLC Delay Value Index           : 1
YC Delay Value Index            : 1
Secondary2 PhaseShift           : 0
Secondary2 Mux Select           : 4

…
…
…

Primary Clock frequency 33.333
Primary Clock Phase Shift 0.000
Primary Clock Output Delay from CLKA 0.180

Secondary1 Clock frequency 66.667
Secondary1 Clock Phase Shift 0.000
Secondary1 Clock Global Output Delay from CLKA 0.180
Secondary1 Clock Core Output Delay from CLKA 0.625

Secondary2 Clock frequency 100.000
Secondary2 Clock Phase Shift 0.000
Secondary2 Clock Global Output Delay from CLKA 0.180
```

Below is an example Verilog HDL description of a legal PLL core configuration generated by SmartGen:

```
module test_pll(POWERDOWN,CLKA,LOCK,GLA);
input POWERDOWN, CLKA;
output  LOCK, GLA;
```

6 – SRAM and FIFO Memories in Microsemi's Low Power Flash Devices

Introduction

As design complexity grows, greater demands are placed upon an FPGA's embedded memory. Fusion, IGLOO, and ProASIC3 devices provide the flexibility of true dual-port and two-port SRAM blocks. The embedded memory, along with built-in, dedicated FIFO control logic, can be used to create cascading RAM blocks and FIFOs without using additional logic gates.

IGLOO, IGLOO PLUS, and ProASIC3L FPGAs contain an additional feature that allows the device to be put in a low power mode called Flash*Freeze. In this mode, the core draws minimal power (on the order of 2 to 127 µW) and still retains values on the embedded SRAM/FIFO and registers. Flash*Freeze technology allows the user to switch to Active mode on demand, thus simplifying power management and the use of SRAM/FIFOs.

Device Architecture

The low power flash devices feature up to 504 kbits of RAM in 4,608-bit blocks (Figure 6-1 on page 148 and Figure 6-2 on page 149). The total embedded SRAM for each device can be found in the datasheets. These memory blocks are arranged along the top and bottom of the device to allow better access from the core and I/O (in some devices, they are only available on the north side of the device). Every RAM block has a flexible, hardwired, embedded FIFO controller, enabling the user to implement efficient FIFOs without sacrificing user gates.

In the IGLOO and ProASIC3 families of devices, the following memories are supported:

- 30 k gate devices and smaller do not support SRAM and FIFO.
- 60 k and 125 k gate devices support memories on the north side of the device only.
- 250 k devices and larger support memories on the north and south sides of the device.

In Fusion devices, the following memories are supported:

- AFS090 and AFS250 support memories on the north side of the device only.
- AFS600 and AFS1500 support memories on the north and south sides of the device.

***Table 7-6* • Maximum I/O Frequency for Single-Ended and Differential I/Os in All Banks in IGLOO and ProASIC Devices (maximum drive strength and high slew selected)**

	Maximum Performance		
Specification	**ProASIC3**	**IGLOO V2 or V5 Devices, 1.5 V DC Core Supply Voltage**	**IGLOO V2, 1.2 V DC Core Supply Voltage**
LVTTL/LVCMOS 3.3 V	200 MHz	180 MHz	TBD
LVCMOS 2.5 V	250 MHz	230 MHz	TBD
LVCMOS 1.8 V	200 MHz	180 MHz	TBD
LVCMOS 1.5 V	130 MHz	120 MHz	TBD
PCI	200 MHz	180 MHz	TBD
PCI-X	200 MHz	180 MHz	TBD
LVDS	350 MHz	300 MHz	TBD
LVPECL	350 MHz	300 MHz	TBD

- In Active and Static modes:
  - Input buffers with pull-up, driven Low
  - Input buffers with pull-down, driven High
  - Bidirectional buffers with pull-up, driven Low
  - Bidirectional buffers with pull-down, driven High
  - Output buffers with pull-up, driven Low
  - Output buffers with pull-down, driven High
  - Tristate buffers with pull-up, driven Low
  - Tristate buffers with pull-down, driven High
- In Flash*Freeze mode:
  - Input buffers with pull-up, driven Low
  - Input buffers with pull-down, driven High
  - Bidirectional buffers with pull-up, driven Low
  - Bidirectional buffers with pull-down, driven High

Electrostatic Discharge Protection

Low power flash devices are tested per JEDEC Standard JESD22-A114-B.

These devices contain clamp diodes at every I/O, global, and power pad. Clamp diodes protect all device pads against damage from ESD as well as from excessive voltage transients.

All IGLOO and ProASIC3 devices are tested to the Human Body Model (HBM) and the Charged Device Model (CDM).

Each I/O has two clamp diodes. One diode has its positive (P) side connected to the pad and its negative (N) side connected to VCCI. The second diode has its P side connected to GND and its N side connected to the pad. During operation, these diodes are normally biased in the off state, except when transient voltage is significantly above VCCI or below GND levels.

In 30K gate devices, the first diode is always off. In other devices, the clamp diode is always on and cannot be switched off.

By selecting the appropriate I/O configuration, the diode is turned on or off. Refer to Table 7-12 on page 193 for more information about the I/O standards and the clamp diode.

The second diode is always connected to the pad, regardless of the I/O configuration selected.

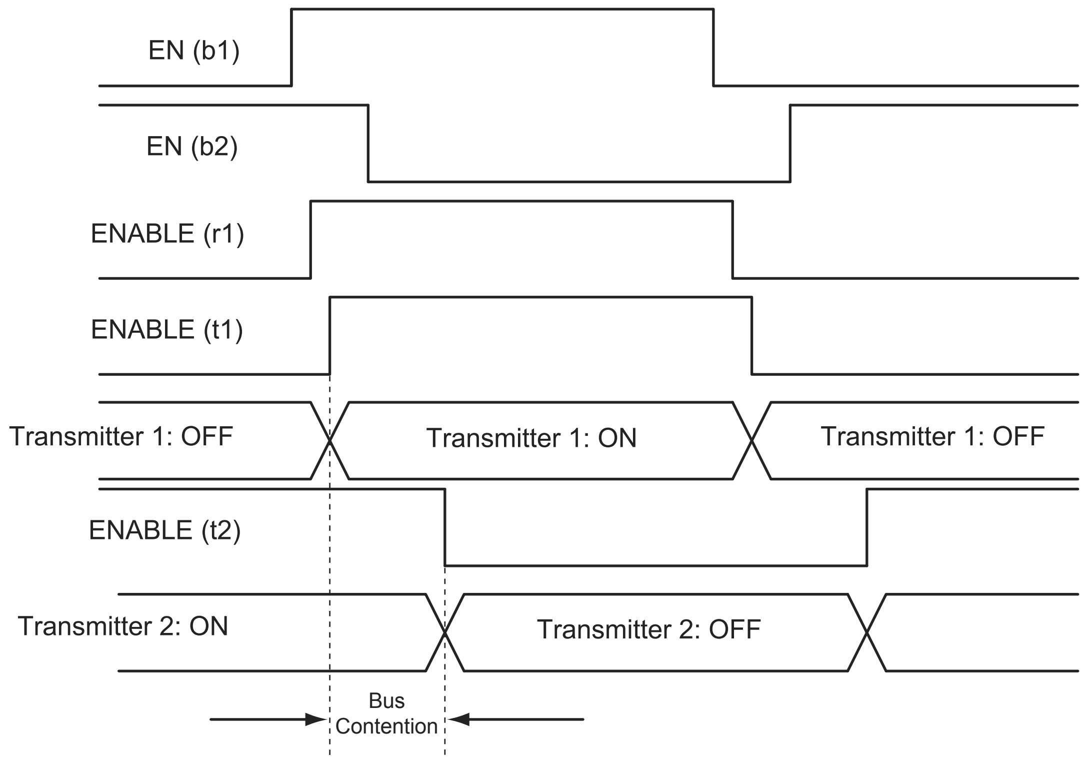


Figure 8-18 **• Timing Diagram (bypasses skew circuit)**

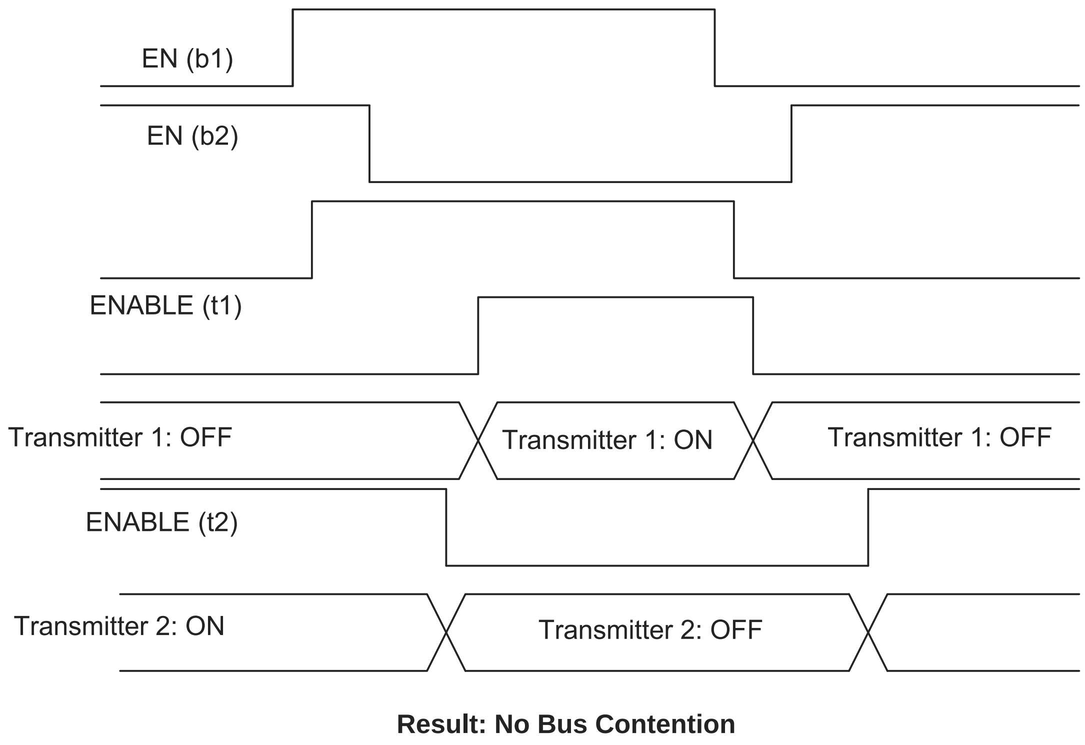


Figure 8-19 **• Timing Diagram (with skew circuit selected)**

Simultaneously Switching Outputs (SSOs) and Printed Circuit Board Layout

Each I/O voltage bank has a separate ground and power plane for input and output circuits (VMV/GNDQ for input buffers and VCCI/GND for output buffers). This isolation is necessary to minimize simultaneous switching noise from the input and output (SSI and SSO). The switching noise (ground bounce and power bounce) is generated by the output buffers and transferred into input buffer circuits, and vice versa.

Since voltage bounce originates on the package inductance, the VMV and VCCI supplies have separate package pin assignments. For the same reason, GND and GNDQ also have separate pin assignments.

The VMV and VCCI pins must be shorted to each other on the board. Also, the GND and GNDQ pins must be shorted to each other on the board. This will prevent unwanted current draw from the power supply.

SSOs can cause signal integrity problems on adjacent signals that are not part of the SSO bus. Both inductive and capacitive coupling parasitics of bond wires inside packages and of traces on PCBs will transfer noise from SSO busses onto signals adjacent to those busses. Additionally, SSOs can produce ground bounce noise and VCCI dip noise. These two noise types are caused by rapidly changing currents through GND and VCCI package pin inductances during switching activities (EQ 8-2 and EQ 8-3).

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

EQ 8-2

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

EQ 8-3

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

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

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

Key issues that need to be considered are as follows:

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

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

3. Double-click I/O to open the Create Core window, which is shown in Figure 9-3).

Figure 9-3 **• I/O Create Core Window**

As seen in Figure 9-3, there are five tabs to configure the I/O macro: Input Buffers, Output Buffers, Bidirectional Buffers, Tristate Buffers, and DDR.

Input Buffers

There are two variations: Regular and Special.

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

The **Special** variation has Width, Technology, Voltage Level, and Resistor Pull-Up/-Down options (see Figure 9-3). All the I/O standards and supply voltages (V_{CCI}) supported for the device family are available for selection.

Instantiating in HDL code

All the supported I/O macros can be instantiated in the top-level HDL code (refer to the *IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide* for a detailed list of all I/O macros). The following is an example:

```
library ieee;
use ieee.std_logic_1164.all;
library proasic3e;

entity TOP is
  port(IN2, IN1 : in std_logic; OUT1 : out std_logic);
end TOP;

architecture DEF_ARCH of TOP is

  component INBUF_LVCMOS5U
    port(PAD : in std_logic := 'U'; Y : out std_logic);
  end component;

  component INBUF_LVCMOS5
    port(PAD : in std_logic := 'U'; Y : out std_logic);
  end component;

  component OUTBUF_SSTL3_II
    port(D : in std_logic := 'U'; PAD : out std_logic);
  end component;

  Other component …..

signal x, y, z…….other signals : std_logic;

begin

  I1 : INBUF_LVCMOS5U
    port map(PAD => IN1, Y =>x);
  I2 : INBUF_LVCMOS5
    port map(PAD => IN2, Y => y);
  I3 : OUTBUF_SSTL3_II
    port map(D => z, PAD => OUT1);

  other port mapping…

end DEF_ARCH;
```

Synthesizing the Design

Libero SoC integrates with the Synplify® synthesis tool. Other synthesis tools can also be used with Libero SoC. Refer to the *Libero SoC User's Guide* or Libero online help for details on how to set up the Libero tool profile with synthesis tools from other vendors.

During synthesis, the following rules apply:

- Generic macros:
  - Users can instantiate generic INBUF, OUTBUF, TRIBUF, and BIBUF macros.
  - Synthesis will automatically infer generic I/O macros.
  - The default I/O technology for these macros is LVTTL.
  - Users will need to use the I/O Attribute Editor in Designer to change the default I/O standard if needed (see Figure 9-6 on page 259).
- Technology-specific I/O macros:
  - Technology-specific I/O macros, such as INBUF_LVCMO25 and OUTBUF_GTL25, can be instantiated in the design. Synthesis will infer these I/O macros in the netlist.

DDR Input Register

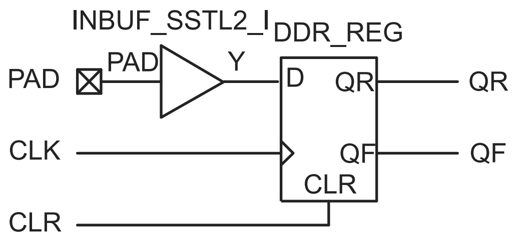


Figure 10-5 **• DDR Input Register (SSTL2 Class I)**

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;
```

General Flash Programming Information

Programming Basics

When choosing a programming solution, there are a number of options available. This section provides a brief overview of those options. The next sections provide more detail on those options as they apply to Microsemi FPGAs.

Reprogrammable or One-Time-Programmable (OTP)

Depending on the technology chosen, devices may be reprogrammable or one-time-programmable. As the name implies, a reprogrammable device can be programmed many times. Generally, the contents of such a device will be completely overwritten when it is reprogrammed. All Microsemi flash devices are reprogrammable.

An OTP device is programmable one time only. Once programmed, no more changes can be made to the contents. Microsemi flash devices provide the option of disabling the reprogrammability for security purposes. This combines the convenience of reprogrammability during design verification with the security of an OTP technology for highly sensitive designs.

Device Programmer or In-System Programming

There are two fundamental ways to program an FPGA: using a device programmer or, if the technology permits, using in-system programming. A device programmer is a piece of equipment in a lab or on the production floor that is used for programming FPGA devices. The devices are placed into a socket mounted in a programming adapter module, and the appropriate electrical interface is applied. The programmed device can then be placed on the board. A typical programmer, used during development, programs a single device at a time and is referred to as a single-site engineering programmer.

With ISP, the device is already mounted onto the system printed circuit board when programming occurs. Typically, ISD programming is performed via a JTAG interface on the FPGA. The JTAG pins can be controlled either by an on-board resource, such as a microprocessor, or by an off-board programmer through a header connection. Once mounted, it can be programmed repeatedly and erased. If the application requires it, the system can be designed to reprogram itself using a microprocessor, without the use of any external programmer.

If multiple devices need to be programmed with the same program, various multi-site programming hardware is available in order to program many devices in parallel. Microsemi In House Programming is also available for this purpose.

Programming Features for Microsemi Devices

Flash Devices

The flash devices supplied by Microsemi are reprogrammable by either a generic device programmer or ISP. Microsemi supports ISP using JTAG, which is supported by the FlashPro4 and FlashPro3, FlashPro Lite, Silicon Sculptor 3, and Silicon Sculptor II programmers.

Levels of ISP support vary depending on the device chosen:

- All SmartFusion, Fusion, IGLOO, and ProASIC3 devices support ISP.
- IGLOO, IGLOOe, IGLOO nano V5, and IGLOO PLUS devices can be programmed in-system when the device is using a 1.5 V supply voltage to the FPGA core.
- IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only) or 1.5 V. IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V.

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:

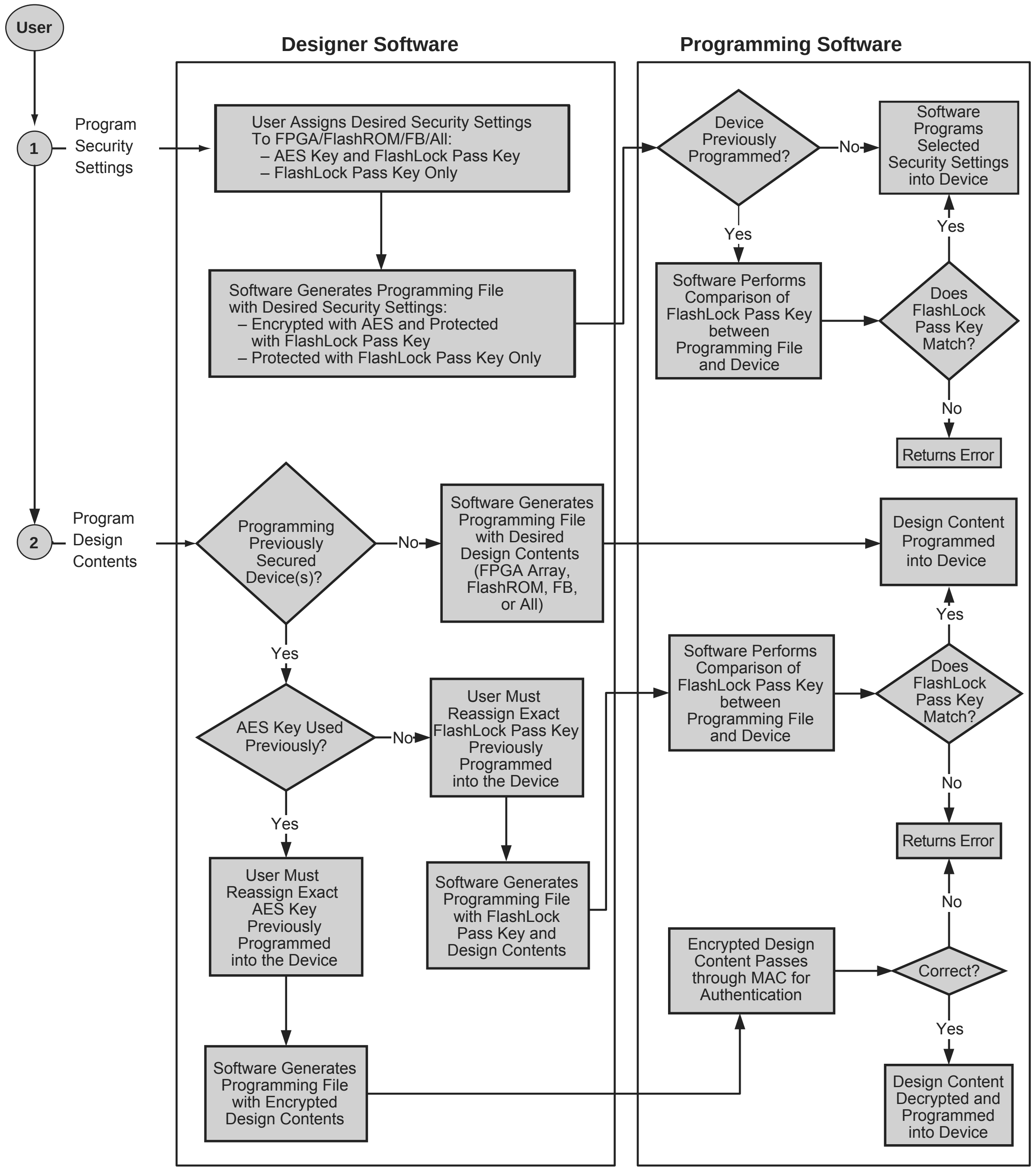
- AES encryption with FlashLock Pass Key protection
- FlashLock protection only (no AES encryption)
- No protection

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]


User
Designer Software
Programming Software
1
Program Security Settings
User Assigns Desired Security Settings To FPGA/FlashROM/FB/All: – AES Key and FlashLock Pass Key – FlashLock Pass Key Only
Software Generates Programming File with Desired Security Settings: – Encrypted with AES and Protected with FlashLock Pass Key – Protected with FlashLock Pass Key Only
Device Previously Programmed?
No
Software Programs Selected Security Settings into Device
Yes
Software Performs Comparison of FlashLock Pass Key between Programming File and Device
Does FlashLock Pass Key Match?
Yes
No
Returns Error
2
Program Design Contents
Programming Previously Secured Device(s)?
No
Software Generates Programming File with Desired Design Contents (FPGA Array, FlashROM, FB, or All)
Design Content Programmed into Device
Yes
AES Key Used Previously?
No
User Must Reassign Exact FlashLock Pass Key Previously Programmed into the Device
Software Generates Programming File with FlashLock Pass Key and Design Contents
Software Performs Comparison of FlashLock Pass Key between Programming File and Device
Does FlashLock Pass Key Match?
Yes
No
Returns Error
Yes
User Must Reassign Exact AES Key Previously Programmed into the Device
Software Generates Programming File with Encrypted Design Contents
Encrypted Design Content Passes through MAC for Authentication
Correct?
No
Yes
Design Content Decrypted and Programmed into Device

Note: *If programming the Security Header only, just perform sub-flow 1.*
If programming design content only, just perform sub-flow 2.

***Figure 12-9* • Security Programming Flows**

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

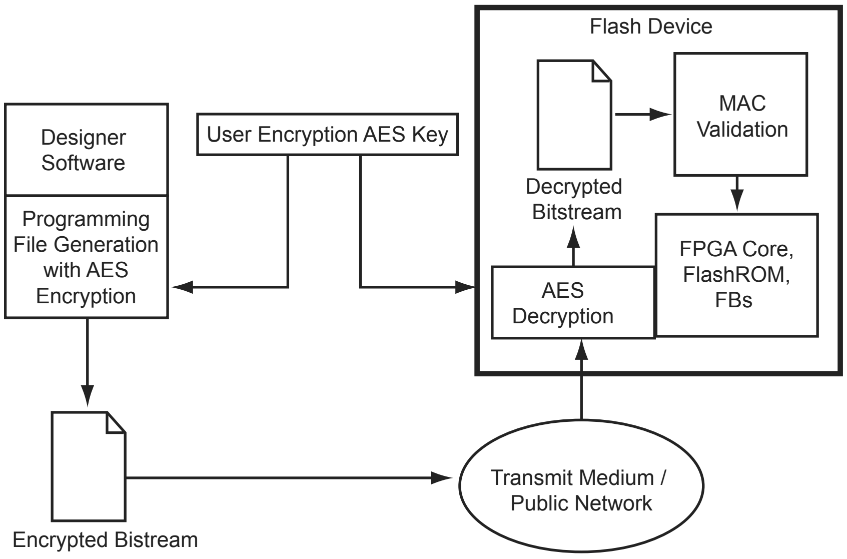


***Figure 13-1* • AES-128 Security Features**

Figure 13-2 shows different applications for ISP programming.

1. In a trusted programming environment, you can program the device using the unencrypted (plaintext) programming file.
2. You can program the AES Key in a trusted programming environment and finish the final programming in an untrusted environment using the AES-encrypted (cipher text) programming file.
3. For the remote ISP updating/reprogramming, the AES Key stored in the device enables the encrypted programming bitstream to be transmitted through the untrusted network connection.

Microsemi low power flash devices also provide the unique Microsemi FlashLock feature, which protects the Pass Key and AES Key. Unless the original FlashLock Pass Key is used to unlock the device, security settings cannot be modified. Microsemi does not support read-back of FPGA core-programmed data; however, the FlashROM contents can selectively be read back (or disabled) via the JTAG port based on the security settings established by the Microsemi Designer software. Refer to the "Security in Low Power Flash Devices" section on page 301 for more information.

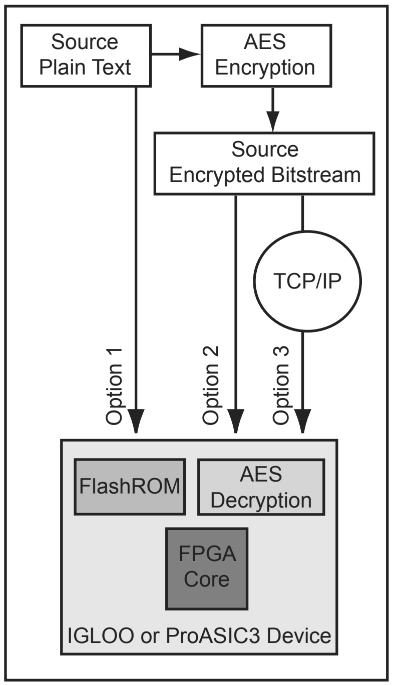


Figure 13-2 • **Different ISP Use Models**

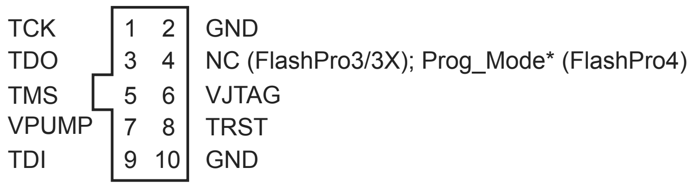
ISP Programming Header Information

The FlashPro4/3/3X programming cable connector can be connected with a 10-pin, 0.1"-pitch programming header. The recommended programming headers are manufactured by AMP (103310-1) and 3M (2510-6002UB). If you have limited board space, you can use a compact programming header manufactured by Samtec (FTSH-105-01-L-D-K). Using this compact programming header, you are required to order an additional header adapter manufactured by Microsemi SoC Products Group (FP3-10PIN-ADAPTER-KIT).

Existing ProASIC<u>PLUS</u> family customers who are using the Samtec Small Programming Header (FTSH-113-01-L-D-K) and are planning to migrate to IGLOO or ProASIC3 devices can also use FP3-10PIN-ADAPTER-KIT.

Table 13-3 **• Programming Header Ordering Codes**

Manufacturer	Part Number	Description
AMP	103310-1	10-pin, 0.1"-pitch cable header (right-angle PCB mount angle)
3M	2510-6002UB	10-pin, 0.1"-pitch cable header (straight PCB mount angle)
Samtec	FTSH-113-01-L-D-K	Small programming header supported by FlashPro and Silicon Sculptor
Samtec	FTSH-105-01-L-D-K	Compact programming header
Samtec	FFSD-05-D-06.00-01-N	10-pin cable with 50 mil pitch sockets; included in FP3-10PIN-ADAPTER-KIT.
Microsemi	FP3-10PIN-ADAPTER-KIT	Transition adapter kit to allow FP3 to be connected to a micro 10-pin header (50 mil pitch). Includes a 6 inch Samtec FFSD-05-D-06.00-01-N cable in the kit. The transition adapter board was previously offered as FP3-26PIN-ADAPTER and includes a 26-pin adapter for design transitions from ProASIC<u>PLUS</u> based boards to ProASIC3 based boards.



*Note: *Prog_Mode on FlashPro4 is an output signal that goes High during device programming and returns to Low when programming is complete. This signal can be used to drive a system to provide a 1.5 V programming signal to IGLOO nano, ProASIC3L, and RT ProASIC3 devices that can run with 1.2 V core voltage but require 1.5 V for programming. IGLOO nano V2 devices can be programmed at 1.2 V core voltage (when using FlashPro4 only), but IGLOO nano V5 devices are programmed with a VCC core voltage of 1.5 V.*

Figure 13-5 **• Programming Header (top view)**

I/O Behavior at Power-Up/-Down

This section discusses the behavior of device I/Os, used and unused, during power-up/-down of V_{CC} and V_{CCI}. As mentioned earlier, VMVx and V_{CCI}Bx are tied together, and therefore, inputs and outputs are powered up/down at the same time.

I/O State during Power-Up/-Down

This section discusses the characteristics of I/O behavior during device power-up and power-down. Before the start of power-up, all I/Os are in tristate mode. The I/Os will remain tristated during power-up until the last voltage supply (VCC or VCCI) is powered to its functional level (power supply functional levels are discussed in the "Power-Up to Functional Time" section on page 378). After the last supply reaches the functional level, the outputs will exit the tristate mode and drive the logic at the input of the output buffer. Similarly, the input buffers will pass the external logic into the FPGA fabric once the last supply reaches the functional level. The behavior of user I/Os is independent of the VCC and VCCI sequence or the state of other voltage supplies of the FPGA (VPUMP and VJTAG). Figure 18-2 shows the output buffer driving HIGH and its behavior during power-up with 10 kΩ external pull-down. In Figure 18-2, VCC is powered first, and VCCI is powered 5 ms after VCC. Figure 18-3 on page 378 shows the state of the I/O when VCCI is powered about 5 ms before VCC. In the circuitry shown in Figure 18-3 on page 378, the output is externally pulled down.

During power-down, device I/Os become tristated once the first power supply (VCC or VCCI) drops below its brownout voltage level. The I/O behavior during power-down is also independent of voltage supply sequencing.

Figure 18-2 **• I/O State when VCC Is Powered before VCCI**