Welcome to E-XFL.COM

Understanding Embedded - FPGAs (Field Programmable Gate Array)

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

Applications of Embedded - FPGAs

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

Details	
Product Status	Obsolete
Number of LABs/CLBs	-
Number of Logic Elements/Cells	-
Total RAM Bits	147456
Number of I/O	154
Number of Gates	1000000
Voltage - Supply	1.14V ~ 1.575V
Mounting Type	Surface Mount
Operating Temperature	0°C ~ 85°C (TJ)
Package / Case	208-BFQFP
Supplier Device Package	208-PQFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/microsemi/m1a3p1000l-pq208

Routing Architecture

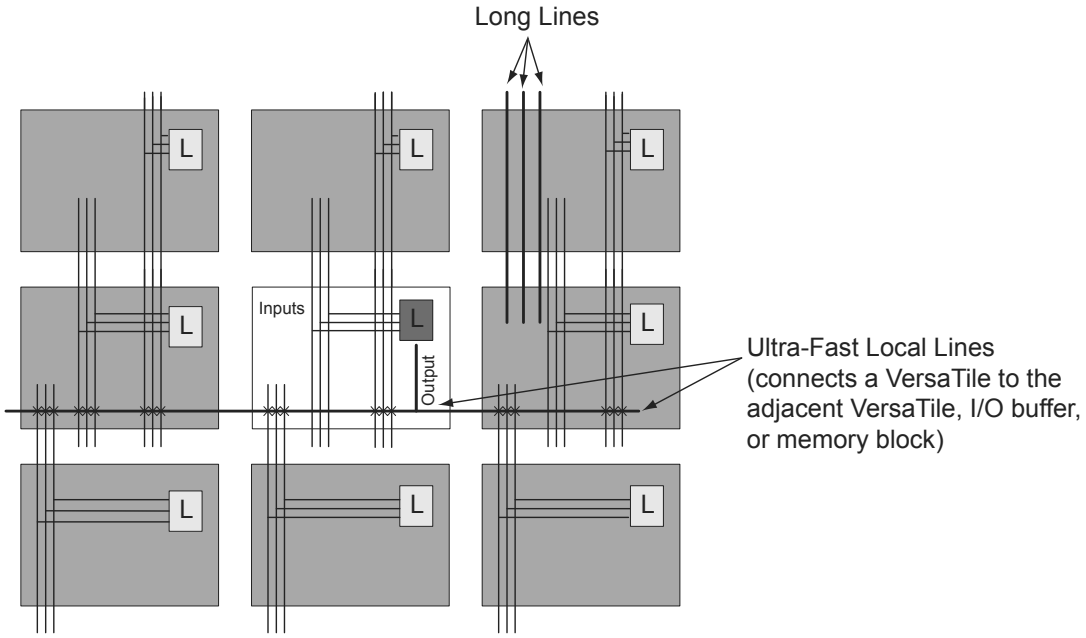
The routing structure of low power flash devices is designed to provide high performance through a flexible four-level hierarchy of routing resources: ultra-fast local resources; efficient long-line resources; high-speed, very-long-line resources; and the high-performance VersaNet networks.

The ultra-fast local resources are dedicated lines that allow the output of each VersaTile to connect directly to every input of the eight surrounding VersaTiles (Figure 1-10). The exception to this is that the SET/CLR input of a VersaTile configured as a D-flip-flop is driven only by the VersaTile global network.

The efficient long-line resources provide routing for longer distances and higher-fanout connections. These resources vary in length (spanning one, two, or four VersaTiles), run both vertically and horizontally, and cover the entire device (Figure 1-11 on page 19). Each VersaTile can drive signals onto the efficient long-line resources, which can access every input of every VersaTile. Routing software automatically inserts active buffers to limit loading effects.

The high-speed, very-long-line resources, which span the entire device with minimal delay, are used to route very long or high-fanout nets: length ±12 VersaTiles in the vertical direction and length ±16 in the horizontal direction from a given core VersaTile (Figure 1-12 on page 19). Very long lines in low power flash devices have been enhanced over those in previous ProASIC families. This provides a significant performance boost for long-reach signals.

The high-performance VersaNet global networks are low-skew, high-fanout nets that are accessible from external pins or internal logic. These nets are typically used to distribute clocks, resets, and other high-fanout nets requiring minimum skew. The VersaNet networks are implemented as clock trees, and signals can be introduced at any junction. These can be employed hierarchically, with signals accessing every input of every VersaTile. For more details on VersaNets, refer to the "Global Resources in Low Power Flash Devices" section on page 47.



Note: Input to the core cell for the D-flip-flop set and reset is only available via the VersaNet global network connection.

Figure 1-10 • **Ultra-Fast Local Lines Connected to the Eight Nearest Neighbors**

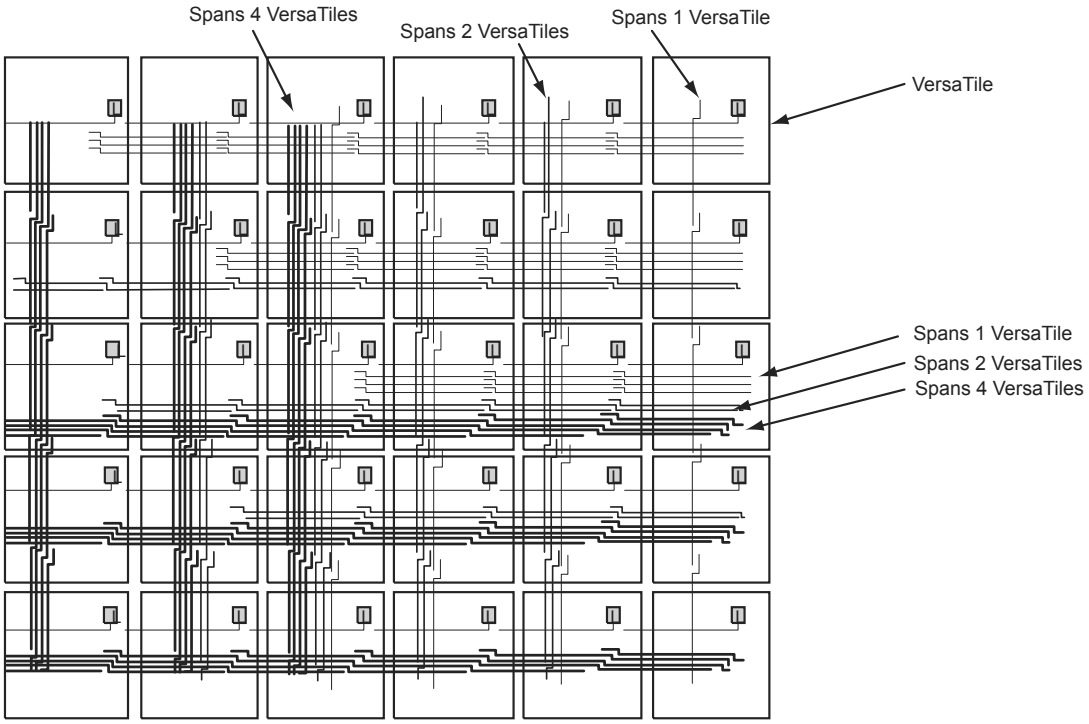


***Figure 1-11* • Efficient Long-Line Resources**

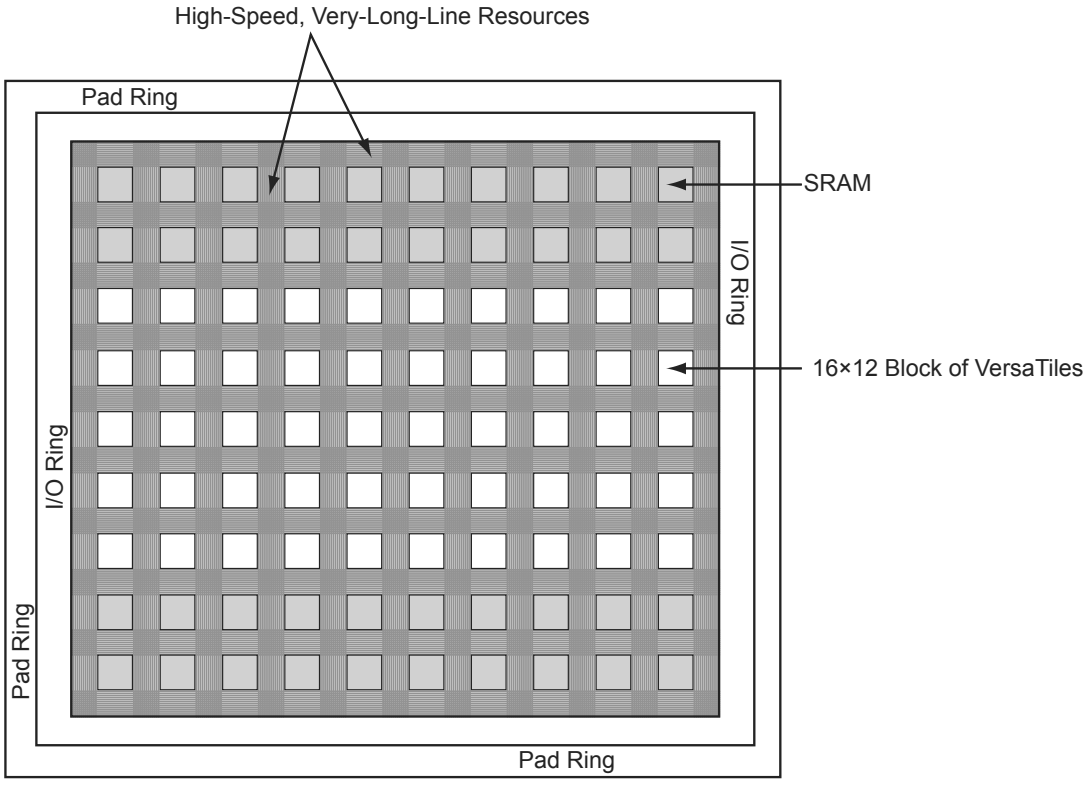


***Figure 1-12* • Very-Long-Line Resources**

Sleep and Shutdown Modes

Sleep Mode

IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 FPGAs support Sleep mode when device functionality is not required. In Sleep mode, V_{CC} (core voltage), V_{JTAG} (JTAG DC voltage), and VPUMP (programming voltage) are grounded, resulting in the FPGA core being turned off to reduce power consumption. While the device is in Sleep mode, the rest of the system can still be operating and driving the input buffers of the device. The driven inputs do not pull up the internal power planes, and the current draw is limited to minimal leakage current.

Table 2-7 shows the power supply status in Sleep mode.

Table 2-7 • **Sleep Mode—Power Supply Requirement for IGLOO, IGLOO nano, IGLOO PLUS, ProASIC3L, and RT ProASIC3 Devices**

Power Supplies	Power Supply State
VCC	Powered off
VCCI = VMV	Powered on
VJTAG	Powered off
VPUMP	Powered off

Refer to the "Power-Up/-Down Behavior" section on page 33 for more information about I/O states during Sleep mode and the timing diagram for entering and exiting Sleep mode.

Shutdown Mode

Shutdown mode is supported for all IGLOO nano and IGLOO PLUS devices as well the following IGLOO/e devices: AGL015, AGL030, AGLE600, AGLE3000, and A3PE3000L. Shutdown mode can be used by turning off all power supplies when the device function is not needed. Cold-sparing and hot-insertion features enable these devices to be powered down without turning off the entire system. When power returns, the live-at-power-up feature enables operation of the device after reaching the voltage activation point.

***Table 3-5* • Globals/Spines/Rows for IGLOO PLUS Devices**

IGLOO PLUS Devices	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
AGLP030	6	0	2	9	18	384*	792	12
AGLP060	6	12	4	9	36	384*	1,584	12
AGLP125	6	12	8	9	72	384*	3,120	12

*Note: *Clock trees that are located at far left and far right will support more VersaTiles.*

***Table 3-6* • Globals/Spines/Rows for Fusion Devices**

Fusion Device	Chip Globals	Quadrant Globals (4×3)	Clock Trees	Globals/ Spines per Tree	Total Spines per Device	VersaTiles in Each Tree	Total VersaTiles	Rows in Each Spine
AFS090	6	12	6	9	54	384	2,304	12
AFS250	6	12	8	9	72	768	6,144	24
AFS600	6	12	12	9	108	1,152	13,824	36
AFS1500	6	12	20	9	180	1,920	38,400	60

List of Changes

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

Date	Changes	Page
July 2010	This chapter is no longer published separately with its own part number and version but is now part of several FPGA fabric user's guides.	N/A
	Notes were added where appropriate to point out that IGLOO nano and ProASIC3 nano devices do not support differential inputs (SAR 21449).	N/A
	The "Global Architecture" section and "VersaNet Global Network Distribution" section were revised for clarity (SARs 20646, 24779).	47, 49
	The "I/O Banks and Global I/Os" section was moved earlier in the document, renamed to "Chip and Quadrant Global I/Os", and revised for clarity. Figure 3-4 • Global Connections Details, Figure 3-6 • Global Inputs, Table 3-2 • Chip Global Pin Name, and Table 3-3 • Quadrant Global Pin Name are new (SARs 20646, 24779).	51
	The "Clock Aggregation Architecture" section was revised (SARs 20646, 24779).	57
	Figure 3-7 • Chip Global Aggregation was revised (SARs 20646, 24779).	59
	The "Global Macro and Placement Selections" section is new (SARs 20646, 24779).	64
v1.4 (December 2008)	The "Global Architecture" section was updated to include 10 k devices, and to include information about VersaNet global support for IGLOO nano devices.	47
	The Table 3-1 • Flash-Based FPGAs was updated to include IGLOO nano and ProASIC3 nano devices.	48
	The "VersaNet Global Network Distribution" section was updated to include 10 k devices and to note an exception in global lines for nano devices.	49
	Figure 3-2 • Simplified VersaNet Global Network (30 k gates and below) is new.	50
	The "Spine Architecture" section was updated to clarify support for 10 k and nano devices.	57
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include IGLOO nano and ProASIC3 nano devices.	57
	The figure in the CLKBUF_LVDS/LVPECL row of Table 3-8 • Clock Macros was updated to change CLKBIBUF to CLKBUF.	62
v1.3 (October 2008)	A third bullet was added to the beginning of the "Global Architecture" section: In Fusion devices, the west CCC also contains a PLL core. In the two larger devices (AFS600 and AFS1500), the west and east CCCs each contain a PLL.	47
	The "Global Resource Support in Flash-Based Devices" section was revised to include new families and make the information more concise.	48
	Table 3-4 • Globals/Spines/Rows for IGLOO and ProASIC3 Devices was updated to include A3PE600/L in the device column.	57
	Table note 1 was revised in Table 3-9 • I/O Standards within CLKBUF to include AFS600 and AFS1500.	63
v1.2 (June 2008)	The following changes were made to the family descriptions in Table 3-1 • Flash-Based FPGAs: • ProASIC3L was updated to include 1.5 V. • The number of PLLs for ProASIC3E was changed from five to six.	48

Available I/O Standards

***Table 4-4* • Available I/O Standards within CLKBUF and CLKBUF_LVDS/LVPECL Macros**

CLKBUF_LVCMOS5
CLKBUF_LVCMOS33 [1]
CLKBUF_LVCMOS25 [2]
CLKBUF_LVCMOS18
CLKBUF_LVCMOS15
CLKBUF_PCI
CLKBUF_PCIX [3]
CLKBUF_GTL25 [2,3]
CLKBUF_GTL33 [2,3]
CLKBUF_GTLP25 [2,3]
CLKBUF_GTLP33 [2,3]
CLKBUF_HSTL_I [2,3]
CLKBUF_HSTL_II [2,3]
CLKBUF_SSTL3_I [2,3]
CLKBUF_SSTL3_II [2,3]
CLKBUF_SSTL2_I [2,3]
CLKBUF_SSTL2_II [2,3]
CLKBUF_LVDS [4,5]
CLKBUF_LVPECL[5]

Notes:

1. *By default, the CLKBUF macro uses 3.3 V LVTTL I/O technology. For more details, refer to the IGLOO, ProASIC3, SmartFusion, and Fusion Macro Library Guide.*
2. *I/O standards only supported in ProASIC3E and IGLOOe families.*
3. *I/O standards only supported in the following Fusion devices: AFS600 and AFS1500.*
4. *B-LVDS and M-LVDS standards are supported by CLKBUF_LVDS.*
5. *Not supported for IGLOO nano and ProASIC3 nano devices.*

Global Synthesis Constraints

The Synplify® synthesis tool, by default, allows six clocks in a design for Fusion, IGLOO, and ProASIC3. When more than six clocks are needed in the design, a user synthesis constraint attribute, syn_global_buffers, can be used to control the maximum number of clocks (up to 18) that can be inferred by the synthesis engine.

High-fanout nets will be inferred with clock buffers and/or internal clock buffers. If the design consists of CCC global buffers, they are included in the count of clocks in the design.

The subsections below discuss the clock input source (global buffers with no programmable delays) and the clock conditioning functional block (global buffers with programmable delays and/or PLL function) in detail.

Phase Adjustment

The four phases available (0, 90, 180, 270) are phases with respect to VCO (PLL output). The VCO is divided to achieve the user's CCC required output frequency (GLA, YB/GLB, YC/GLC). The division happens after the selection of the VCO phase. The effective phase shift is actually the VCO phase shift divided by the output divider. This is why the visual CCC shows both the actual achievable phase and more importantly the actual delay that is equivalent to the phase shift that can be achieved.

Dynamic PLL Configuration

The CCCs can be configured both statically and dynamically.

In addition to the ports available in the Static CCC, the Dynamic CCC has the dynamic shift register signals that enable dynamic reconfiguration of the CCC. With the Dynamic CCC, the ports CLKB and CLKC are also exposed. All three clocks (CLKA, CLKB, and CLKC) can be configured independently.

The CCC block is fully configurable. The following two sources can act as the CCC configuration bits.

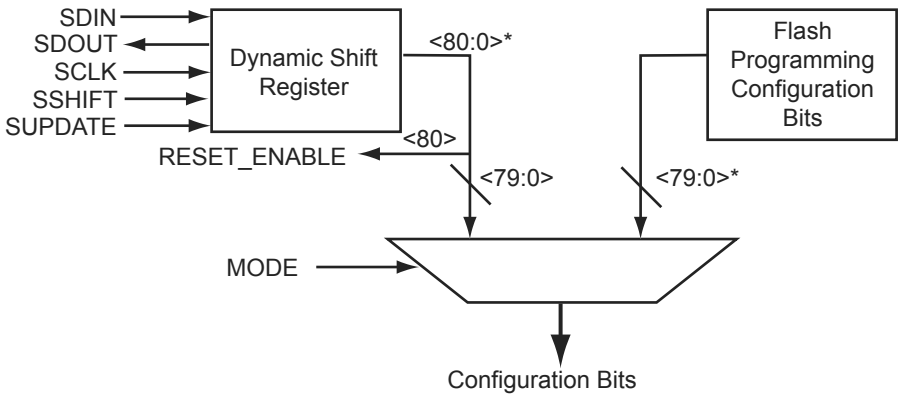
Flash Configuration Bits

The flash configuration bits are the configuration bits associated with programmed flash switches. These bits are used when the CCC is in static configuration mode. Once the device is programmed, these bits cannot be modified. They provide the default operating state of the CCC.

Dynamic Shift Register Outputs

This source does not require core reprogramming and allows core-driven dynamic CCC reconfiguration. When the dynamic register drives the configuration bits, the user-defined core circuit takes full control over SDIN, SDOUT, SCLK, SSHIFT, and SUPDATE. The configuration bits can consequently be dynamically changed through shift and update operations in the serial register interface. Access to the logic core is accomplished via the dynamic bits in the specific tiles assigned to the PLLs.

Figure 4-21 illustrates a simplified block diagram of the MUX architecture in the CCCs.



*Note: *For Fusion, bit <88:81> is also needed.*

***Figure 4-21* • The CCC Configuration MUX Architecture**

The selection between the flash configuration bits and the bits from the configuration register is made using the MODE signal shown in Figure 4-21. If the MODE signal is logic HIGH, the dynamic shift register configuration bits are selected. There are 81 control bits to configure the different functions of the CCC.

The following is an example of a PLL configuration utilizing the clock frequency synthesis and clock delay adjustment features. The steps include generating the PLL core with SmartGen, performing simulation for verification with Model*Sim*, and performing static timing analysis with SmartTime in Designer.

Parameters of the example PLL configuration:

Input Frequency – 20 MHz

Primary Output Requirement – 20 MHz with clock advancement of 3.02 ns

Secondary 1 Output Requirement – 40 MHz with clock delay of 2.515 ns

Figure 4-29 shows the SmartGen settings. Notice that the overall delays are calculated automatically, allowing the user to adjust the delay elements appropriately to obtain the desired delays.

Figure 4-29 • **SmartGen Settings**

After confirming the correct settings, generate a structural netlist of the PLL and verify PLL core settings by checking the log file:

```
Name                          : test_pll_delays
Family                        : ProASIC3E
Output Format                 : VHDL
Type                          : Static PLL
Input Freq(MHz)               : 20.000
CLKA Source                   : Hardwired I/O
Feedback Delay Value Index    : 21
Feedback Mux Select           : 2
XDLY Mux Select               : No
Primary Freq(MHz)             : 20.000
Primary PhaseShift            : 0
Primary Delay Value Index     : 1
Primary Mux Select            : 4
Secondary1 Freq(MHz)          : 40.000
Use GLB                       : YES
Use YB                        : NO
…
…
…
Primary Clock frequency 20.000
Primary Clock Phase Shift 0.000
```

- Use quadrant global region assignments by finding the clock net associated with the CCC macro under the Nets tab and creating a quadrant global region for the net, as shown in Figure 4-33.

***Figure 4-33* • Quadrant Clock Assignment for a Global Net**

External I/O–Driven CCCs

The above-mentioned recommendation for proper layout techniques will ensure the correct assignment. It is possible that, especially with External I/O–Driven CCC macros, placement of the CCC macro in a desired location may not be achieved. For example, assigning an input port of an External I/O–Driven CCC near a particular CCC location does not guarantee global assignments to the desired location. This is because the clock inputs of External I/O–Driven CCCs can be assigned to any I/O location; therefore, it is possible that the CCC connected to the clock input will be routed to a location other than the one closest to the I/O location, depending on resource availability and placement constraints.

Clock Placer

The clock placer is a placement engine for low power flash devices that places global signals on the chip global and quadrant global networks. Based on the clock assignment constraints for the chip global and quadrant global clocks, it will try to satisfy all constraints, as well as creating quadrant clock regions when necessary. If the clock placer fails to create the quadrant clock regions for the global signals, it will report an error and stop Layout.

The user must ensure that the constraints set to promote clock signals to quadrant global networks are valid.

Cascading CCCs

The CCCs in low power flash devices can be cascaded. Cascading CCCs can help achieve more accurate PLL output frequency results than those achievable with a single CCC. In addition, this technique is useful when the user application requires the output clock of the PLL to be a multiple of the reference clock by an integer greater than the maximum feedback divider value of the PLL (divide by 128) to achieve the desired frequency.

For example, the user application may require a 280 MHz output clock using a 2 MHz input reference clock, as shown in Figure 4-34 on page 126.

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

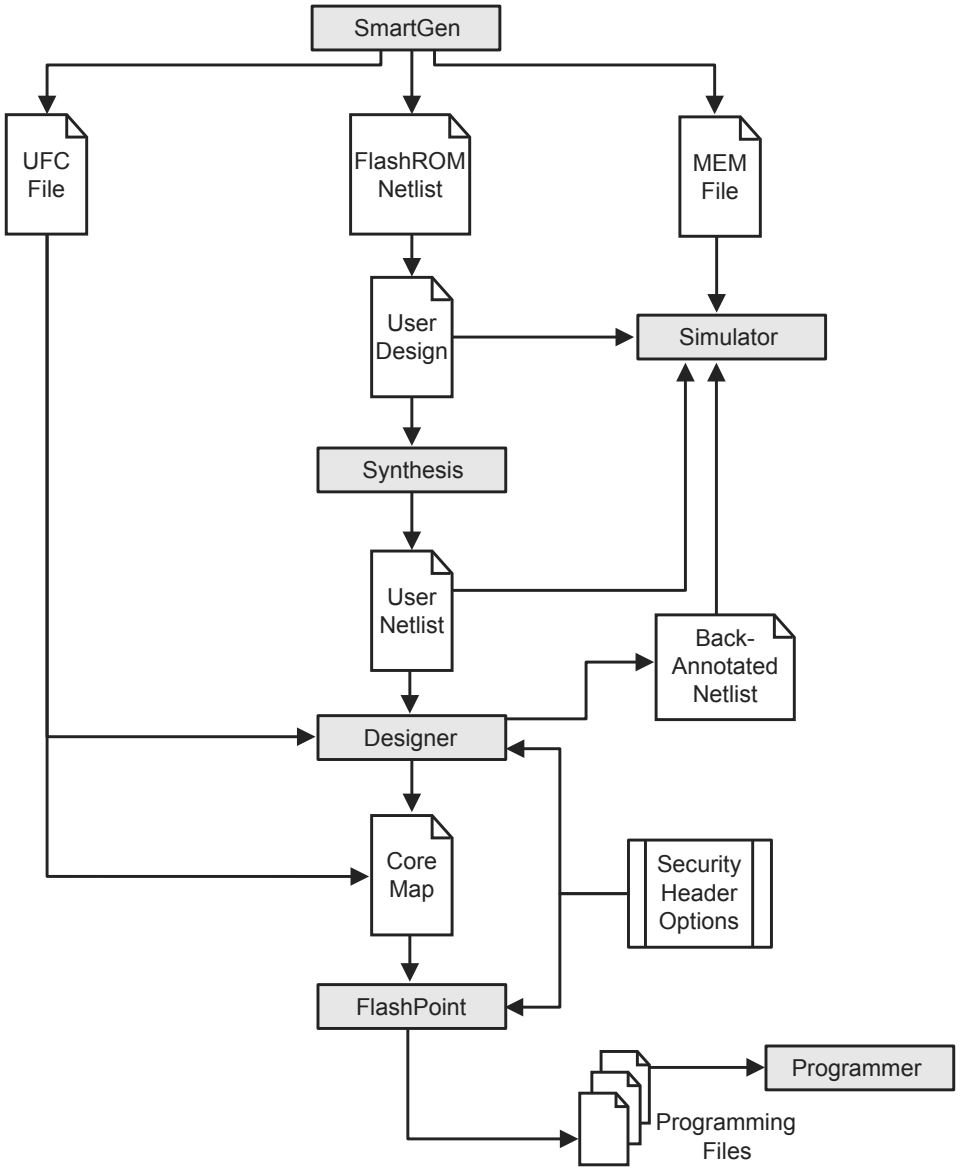


***Figure 5-9* • FlashROM Design Flow**

GTL 2.5 V (Gunning Transceiver Logic 2.5 V)

This is a low power standard (JESD 8-3) for electrical signals used in CMOS circuits that allows for low electromagnetic interference at high transfer speeds. It has a voltage swing between 0.4 V and 1.2 V and typically operates at speeds of between 20 and 40 MHz. VCCI must be connected to 2.5 V. The reference voltage (VREF) is 0.8 V.

GTL 3.3 V (Gunning Transceiver Logic 3.3 V)

This is the same as GTL 2.5 V above, except VCCI must be connected to 3.3 V.

GTL+ (Gunning Transceiver Logic Plus)

This is an enhanced version of GTL that has defined slew rates and higher voltage levels. It requires a differential amplifier input buffer and an open-drain output buffer. Even though the output is open-drain, VCCI must be connected to either 2.5 V or 3.3 V. The reference voltage (VREF) is 1 V.

Differential Standards

These standards require two I/Os per signal (called a "signal pair"). Logic values are determined by the potential difference between the lines, not with respect to ground. This is why differential drivers and receivers have much better noise immunity than single-ended standards. The differential interface standards offer higher performance and lower power consumption than their single-ended counterparts. Two I/O pins are used for each data transfer channel. Both differential standards require resistor termination.

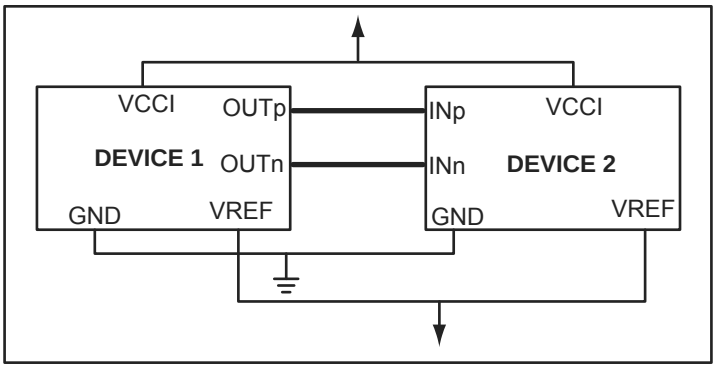


***Figure 8-8* • Differential Topology**

LVPECL (Low-Voltage Positive Emitter Coupled Logic)

LVPECL requires that one data bit be carried through two signal lines; therefore, two pins are needed per input or output. It also requires external resistor termination. The voltage swing between the two signal lines is approximately 850 mV. When the power supply is +3.3 V, it is commonly referred to as Low-Voltage PECL (LVPECL). Refer to the device datasheet for the full implementation of the LVPECL transmitter and receiver.

LVDS (Low-Voltage Differential Signal)

LVDS is a moderate-speed differential signaling system, in which the transmitter generates two different voltages that are compared at the receiver. LVDS uses a differential driver connected to a terminated receiver through a constant-impedance transmission line. It requires that one data bit be carried through two signal lines; therefore, the user will need two pins per input or output. It also requires external resistor termination. The voltage swing between the two signal lines is approximately 350 mV. V_{CCI} is 2.5 V. Low power flash devices contain dedicated circuitry supporting a high-speed LVDS standard that has its own user specification. Refer to the device datasheet for the full implementation of the LVDS transmitter and receiver.

***Table 8-18* • Supported IGLOOe, ProASIC3L, and ProASIC3E I/O Standards and Corresponding VREF and VTT Voltages**

I/O Standard	Input/Output Supply Voltage (VMV_{TYP}/V_{CCI_TYP})	Input Reference Voltage (V_{REF_TYP})	Board Termination Voltage (V_{TT_TYP})
LVTTL/ L VCMOS 3.3 V	3.30 V	–	–
LVCMOS 2.5 V	2.50 V	–	–
LVCMOS 2.5/5.0 V Input	2.50 V	–	–
LVCMOS 1.8 V	1.80 V	–	–
LVCMOS 1.5 V	1.50 V	–	–
PCI 3.3 V	3.30 V	–	–
PCI-X 3.3 V	3.30 V	–	–
GTL+ 3.3 V	3.30 V	1.00 V	1.50 V
GTL+ 2.5 V	2.50 V	1.00 V	1.50 V
GTL 3.3 V	3.30 V	0.80 V	1.20 V
GTL 2.5 V	2.50 V	0.80 V	1.20 V
HSTL Class I	1.50 V	0.75 V	0.75 V
HSTL Class II	1.50 V	0.75 V	0.75 V
SSTL3 Class I	3.30 V	1.50 V	1.50 V
SSTL3 Class II	3.30 V	1.50 V	1.50 V
SSTL2 Class I	2.50 V	1.25 V	1.25 V
SSTL2 Class II	2.50 V	1.25 V	1.25 V
LVDS, DDR LVDS, B-LVDS, M-LVDS	2.50 V	–	–
LVPECL	3.30 V	–	–

The procedure is as follows:

1. Select the bank to which you want VCCI to be assigned from the **Choose Bank** list.
2. Select the I/O standards for that bank. If you select any standard, the tool will automatically show all compatible standards that have a common VCCI voltage requirement.
3. Click **Apply**.
4. Repeat steps 1–3 to assign VCCI voltages to other banks. Refer to Figure 9-11 on page 263 to find out how many I/O banks are needed for VCCI bank assignment.

Manually Assigning VREF Pins

Voltage-referenced inputs require an input reference voltage (VREF). The user must assign VREF pins before running Layout. Before assigning a VREF pin, the user must set a VREF technology for the bank to which the pin belongs.

VREF Rules for the Implementation of Voltage-Referenced I/O Standards

The VREF rules are as follows:

1. Any I/O (except JTAG I/Os) can be used as a V_{REF} pin.
2. One V_{REF} pin can support up to 15 I/Os. It is recommended, but not required, that eight of them be on one side and seven on the other side (in other words, all 15 can still be on one side of VREF).
3. SSTL3 (I) and (II): Up to 40 I/Os per north or south bank in any position
4. LVPECL / GTL+ 3.3 V / GTL 3.3 V: Up to 48 I/Os per north or south bank in any position (not applicable for IGLOO nano and ProASIC3 nano devices)
5. SSTL2 (I) and (II) / GTL+ 2.5 V / GTL 2.5 V: Up to 72 I/Os per north or south bank in any position
6. VREF minibanks partition rule: Each I/O bank is physically partitioned into VREF minibanks. The VREF pins within a VREF minibank are interconnected internally, and consequently, only one VREF voltage can be used within each VREF minibank. If a bank does not require a VREF signal, the VREF pins of that bank are available as user I/Os.
7. The first VREF minibank includes all I/Os starting from one end of the bank to the first power triple and eight more I/Os after the power triple. Therefore, the first VREF minibank may contain (0 + 8), (2 + 8), (4 + 8), (6 + 8), or (8 + 8) I/Os.

   The second VREF minibank is adjacent to the first VREF minibank and contains eight I/Os, a power triple, and eight more I/Os after the triple. An analogous rule applies to all other VREF minibanks but the last.

   The last VREF minibank is adjacent to the previous one but contains eight I/Os, a power triple, and all I/Os left at the end of the bank. This bank may also contain (8 + 0), (8 + 2), (8 + 4), (8 + 6), or (8 + 8) available I/Os.

   Example:

   4 I/Os $\rightarrow$ Triple $\rightarrow$ 8 I/Os, 8 I/Os $\rightarrow$ Triple $\rightarrow$ 8 I/Os, 8 I/Os $\rightarrow$ Triple $\rightarrow$ 2 I/Os

   That is, minibank A = (4 + 8) I/Os, minibank B = (8 + 8) I/Os, minibank C = (8 + 2) I/Os.
8. Only minibanks that contain input or bidirectional I/Os require a VREF. A VREF is not needed for minibanks composed of output or tristated I/Os.

Assigning the VREF Voltage to a Bank

When importing the PDC file, the VREF voltage can be assigned to the I/O bank. The PDC command is as follows:

```
set_iobank –vref [value]
```

Another method for assigning VREF is by using **MVN** > **Edit** > **I/O Bank Settings** (Figure 9-13 on page 266).

Instantiating DDR Registers

Using SmartGen is the simplest way to generate the appropriate RTL files for use in the design. Figure 10-4 shows an example of using SmartGen to generate a DDR SSTL2 Class I input register. SmartGen provides the capability to generate all of the DDR I/O cells as described. The user, through the graphical user interface, can select from among the many supported I/O standards. The output formats supported are Verilog, VHDL, and EDIF.

Figure 10-5 on page 277 through Figure 10-8 on page 280 show the I/O cell configured for DDR using SSTL2 Class I technology. For each I/O standard, the I/O pad is buffered by a special primitive that indicates the I/O standard type.

***Figure 10-4* • Example of Using SmartGen to Generate a DDR SSTL2 Class I Input Register**

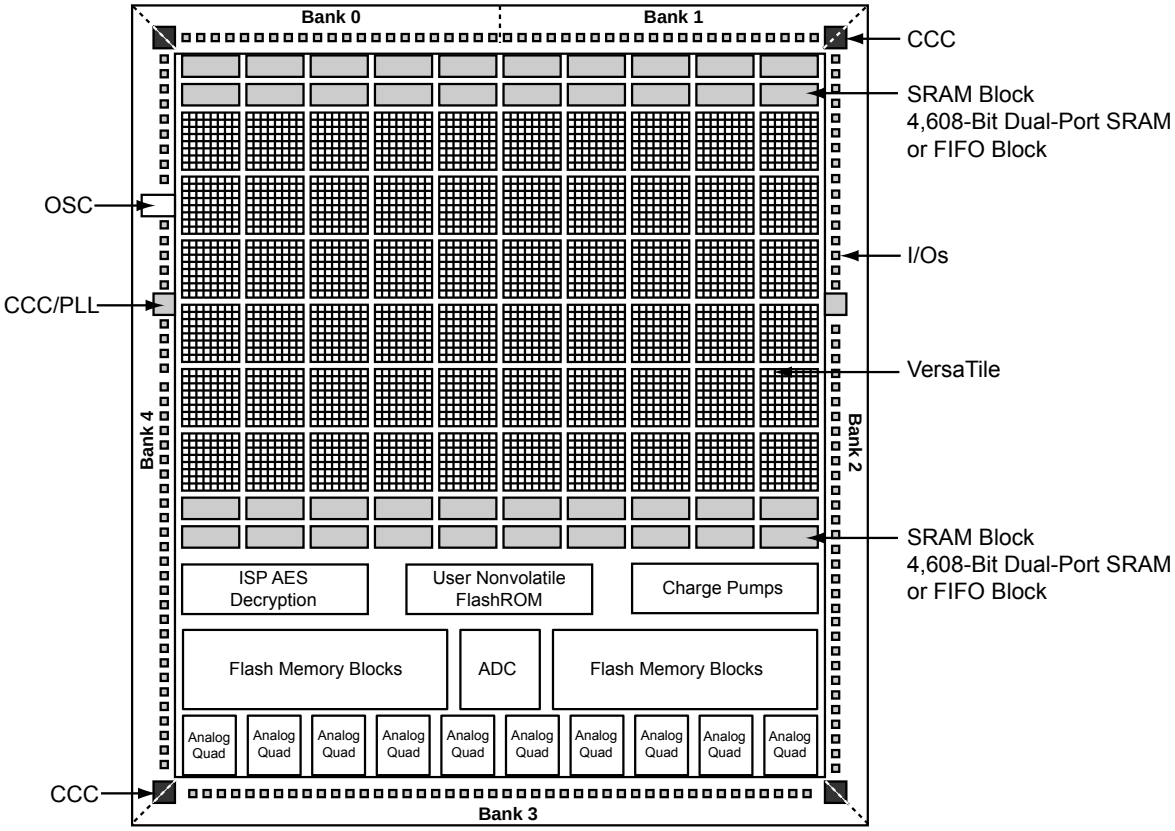


***Figure 12-3* • Block Representation of the AES Decryption Core in a Fusion AFS600 FPGA**

Security Features

IGLOO and ProASIC3 devices have two entities inside: FlashROM and the FPGA core fabric. Fusion devices contain three entities: FlashROM, FBs, and the FPGA core fabric. The parts can be programmed or updated independently with a STAPL programming file. The programming files can be AES-encrypted or plaintext. This allows maximum flexibility in providing security to the entire device. Refer to the "Programming Flash Devices" section on page 287 for information on the FlashROM structure.

Unlike SRAM-based FPGA devices, which require a separate boot PROM to store programming data, low power flash devices are nonvolatile, and the secured configuration data is stored in on-chip flash cells that are part of the FPGA fabric. Once programmed, this data is an inherent part of the FPGA array and does not need to be loaded at system power-up. SRAM-based FPGAs load the configuration bitstream upon power-up; therefore, the configuration is exposed and can be read easily.

The built-in FPGA core, FBs, and FlashROM support programming files encrypted with the 128-bit AES (FIPS-192) block ciphers. The AES key is stored in dedicated, on-chip flash memory and can be programmed before the device is shipped to other parties (allowing secure remote field updates).

Security in ARM-Enabled Low Power Flash Devices

There are slight differences between the regular flash devices and the ARM®-enabled flash devices, which have the M1 and M7 prefix.

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

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]

***Table 12-5* • FlashLock Security Options for Fusion**

Security Option	FlashROM Only	FPGA Core Only	FB Core Only	All
No AES / no FlashLock	–	–	–	–
FlashLock	✓	✓	✓	✓
AES and FlashLock	✓	✓	✓	✓

For this scenario, generate the programming file as follows:

1. Select only the **Security settings** option, as indicated in Figure 12-14 and Figure 12-15 on page 318. Click **Next**.

***Figure 12-14* • Programming IGLOO and ProASIC3 Security Settings Only**

IEEE 1532 (JTAG) Interface

The supported industry-standard IEEE 1532 programming interface builds on the IEEE 1149.1 (JTAG) standard. IEEE 1532 defines the standardized process and methodology for ISP. Both silicon and software issues are addressed in IEEE 1532 to create a simplified ISP environment. Any IEEE 1532 compliant programmer can be used to program low power flash devices. Device serialization is not supported when using the IEEE1532 standard. Refer to the standard for detailed information about IEEE 1532.

Security

Unlike SRAM-based FPGAs that require loading at power-up from an external source such as a microcontroller or boot PROM, Microsemi nonvolatile devices are live at power-up, and there is no bitstream required to load the device when power is applied. The unique flash-based architecture prevents reverse engineering of the programmed code on the device, because the programmed data is stored in nonvolatile memory cells. Each nonvolatile memory cell is made up of small capacitors and any physical deconstruction of the device will disrupt stored electrical charges.

Each low power flash device has a built-in 128-bit Advanced Encryption Standard (AES) decryption core, except for the 30 k gate devices and smaller. Any FPGA core or FlashROM content loaded into the device can optionally be sent as encrypted bitstream and decrypted as it is loaded. This is particularly suitable for applications where device updates must be transmitted over an unsecured network such as the Internet. The embedded AES decryption core can prevent sensitive data from being intercepted (Figure 13-1 on page 331). A single 128-bit AES Key (32 hex characters) is used to encrypt FPGA core programming data and/or FlashROM programming data in the Microsemi tools. The low power flash devices also decrypt with a single 128-bit AES Key. In addition, low power flash devices support a Message Authentication Code (MAC) for authentication of the encrypted bitstream on-chip. This allows the encrypted bitstream to be authenticated and prevents erroneous data from being programmed into the device. The FPGA core, FlashROM, and Flash Memory Blocks (FBs), in Fusion only, can be updated independently using a programming file that is AES-encrypted (cipher text) or uses plain text.

17 – UJTAG Applications in Microsemi's Low Power Flash Devices

Introduction

In Fusion, IGLOO, and ProASIC3 devices, there is bidirectional access from the JTAG port to the core VersaTiles during normal operation of the device (Figure 17-1). User JTAG (UJTAG) is the ability for the design to use the JTAG ports for access to the device for updates, etc. While regular JTAG is used, the UJTAG tiles, located at the southeast area of the die, are directly connected to the JTAG Test Access Port (TAP) Controller in normal operating mode. As a result, all the functional blocks of the device, such as Clock Conditioning Circuits (CCCs) with PLLs, SRAM blocks, embedded FlashROM, flash memory blocks, and I/O tiles, can be reached via the JTAG ports. The UJTAG functionality is available by instantiating the UJTAG macro directly in the source code of a design. Access to the FPGA core VersaTiles from the JTAG ports enables users to implement different applications using the TAP Controller (JTAG port). This document introduces the UJTAG tile functionality and discusses a few application examples. However, the possible applications are not limited to what is presented in this document. UJTAG can serve different purposes in many designs as an elementary or auxiliary part of the design. For detailed usage information, refer to the "Boundary Scan in Low Power Flash Devices" section on page 357.

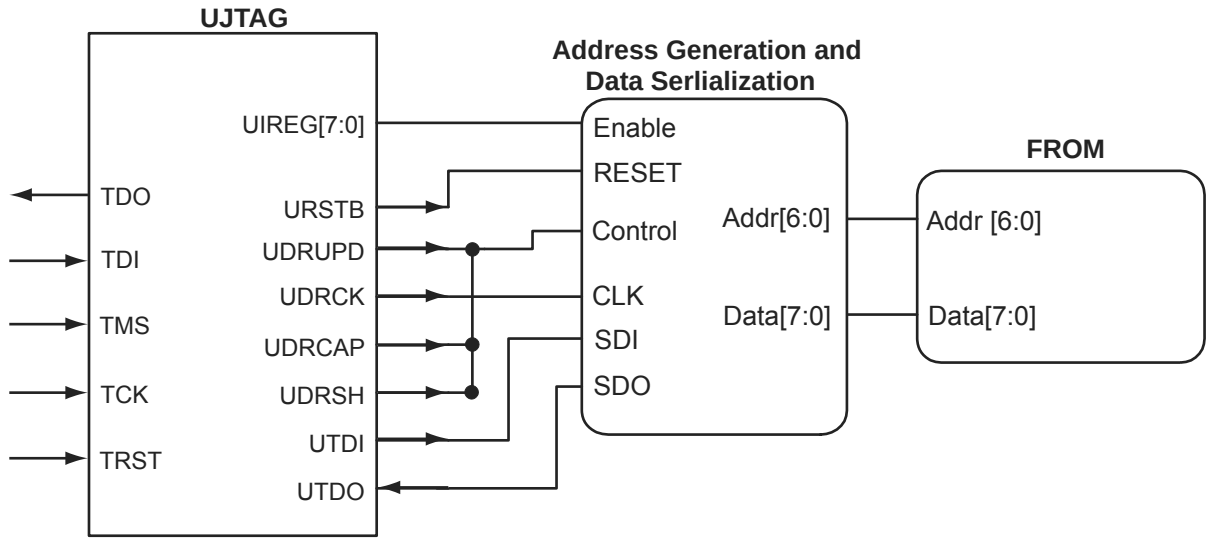


***Figure 17-1* • Block Diagram of Using UJTAG to Read FlashROM Contents**