Fusion FlashROM

Table Of Contents

Introduction . . . 1
FlashROM Architecture and Applications . . . 3
FlashROM Security . . . 4
Programming and Accessing FlashROM . . . 5
FlashROM Design Flow . . . 7
FlashROM Generation and Instantiation in the Design . . . 8
Simulation of FlashROM Design . . . 10
Programming File Generation for FlashROM Design . . . 10
Custom Serialization Using FlashROM . . . 13
Conclusion . . . 13
Related Documents . . . 13
List of Changes . . . 14

Introduction

The Microsemi Fusion® family, based on the highly successful ProASIC3 flash field programmable gate array (FPGA) architecture, is designed as a high-performance, programmable, mixed-signal platform. Fusion supports many peripherals, including embedded Flash memory, analog-to-digital converter (ADC), high-drive outputs, RC and crystal oscillators, and real-time-counter (RTC). The functionality of the Flash array blocks is similar to that of a large single-port synchronous RAM.

To allow using the devices in diverse system applications, Fusion devices have a dedicated non-volatile FlashROM memory of 1,024 bits in addition to the Flash array, which provides a unique feature in the FPGA market. The FlashROM can be read, modified, and written using the JTAG (or UJTAG) interface; however, it can be read but not modified from the FPGA core. The FlashROM is physically organized as 8×128 bits and logically organized as 8 pages of 16 bytes. Figure 1 shows the Fusion device architecture.

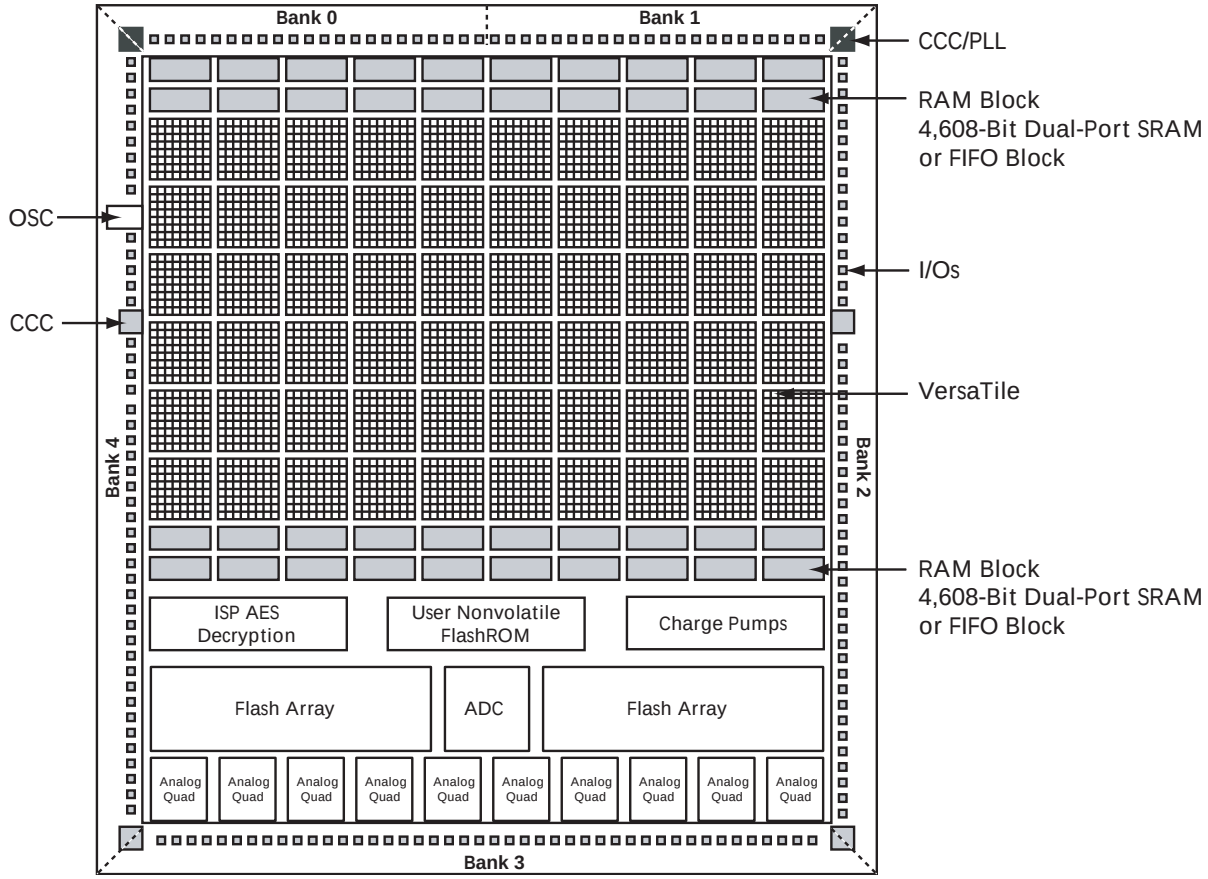

***Figure 1* • Fusion Device Architecture Overview (AFS600) Fusion**

FlashROM Architecture and Applications

Microsemi Fusion devices have 1 kbit of user accessible, non-volatile FlashROM on-chip. The FlashROM is logically organized as 8 pages of 16 bytes (128 bits per page). Figure 2 shows the FlashROM logical level structure. The SmartGen core generator is used to configure FlashROM content. You can configure each page independently. SmartGen enables you to create and modify regions within a page; these regions can be from 1 to 16 bytes long.

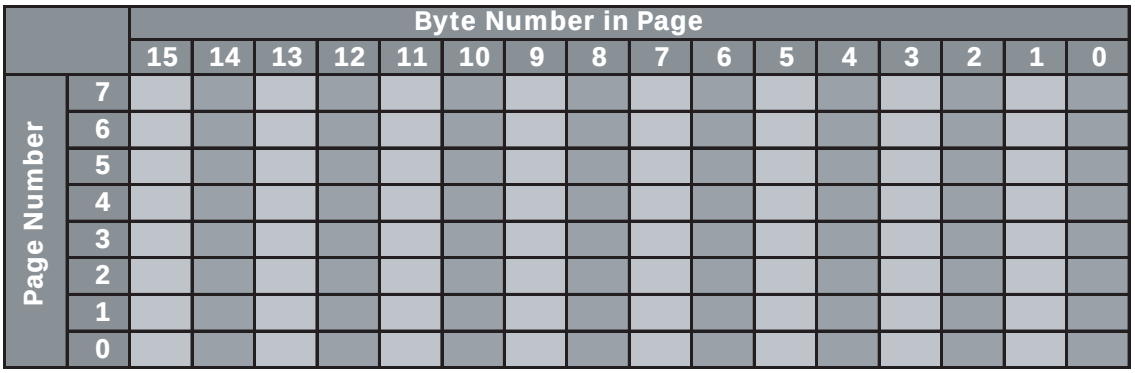

***Figure 2* • FlashROM Configuration**

The FlashROM content may be changed independently of the FPGA core content. It may be easily accessed and programmed via JTAG, depending on the security settings of the device. The SmartGen core generator enables each region to be independently updated (described in the "Programming and Accessing FlashROM" section on page 5). This enables you to change the FlashROM content on a per-part basis while keeping some regions "constant" for all parts. These features allow the FlashROM to be used in diverse system applications. Consider the following possible uses of FlashROM:

- Internet protocol (IP) addressing (wireless or fixed)
- System-calibration settings
- Restoring configuration after unpredictable system power-down
- Device serialization and/or inventory control
- Subscription-based business models (for example, set-top boxes)
- Secure key storage
- Asset management tracking
- Date stamping
- Version management

FlashROM Security

Fusion devices have an on-chip Advanced Encryption Standard (AES) decryption core, combined with an enhanced version of the Microsemi Flash-based lock technology (FlashLock®). Together, they provide unmatched levels of security in a programmable logic device. This security applies to both the FPGA core and FlashROM content. Fusion devices use the 128-bit AES (Rijndael) algorithm to encrypt programming files for secure transmission to the on-chip AES decryption core. The same algorithm is then used to decrypt the programming file. This key size provides approximately 3.4×10^{38} possible 128-bit keys. A computing system that could find a DES key in a second would take approximately 149 trillion years to crack a 128-bit AES key. The 128-bit FlashLock feature in Fusion works via a FlashLock security Pass Key mechanism, where the user locks or unlocks the device with a user-defined key.

If the device is locked with certain security settings, functions such as device read, write, and erase are disabled. This unique feature helps to protect against invasive and noninvasive attacks. Without the correct Pass Key, access to the FPGA is denied. In order to gain access to the FPGA, the device first must be unlocked using the correct Pass Key. During programming of the FlashROM and/or the FPGA core, you can generate the security header programming file, which is used to program the AES key and/or FlashLock Pass Key. The security header programming file can also be generated independently of the FlashROM and FPGA core content. The FlashLock Pass Key is not stored in the FlashROM.

Fusion devices with AES-based security allow for secure remote field updates over public networks such as the Internet, and ensure that valuable intellectual property (IP) remains out of the hands of IP thieves. Figure 3 shows the flow diagram.

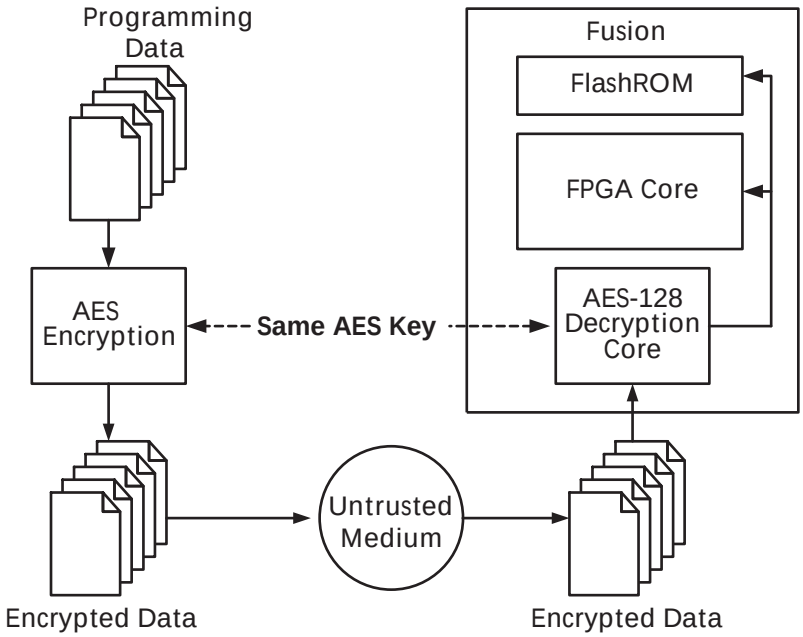

***Figure 3* • Programming FlashROM Using AES**

Programming and Accessing FlashROM

The FlashROM content can only be programmed via JTAG, but it can be read back selectively through the JTAG programming interface, the UJTAG interface, or via direct FPGA core addressing. The pages of the FlashROM can be made secure to prevent read back via JTAG. In that case, read back on these secured pages is only possible by the FPGA core fabric or via UJTAG.

A 7-bit address from the FPGA core defines which of the 8 pages (3 MSBs) is being read, and which of the 16 bytes within the selected page (4 LSBs) are being read. The FlashROM content can be read on a random basis; the access time is 10 ns for a device supporting commercial specifications. The FPGA core will be powered down during writing of the FlashROM content. FPGA power-down during FlashROM programming is managed on-chip, and FPGA core functionality is not available during programming of the FlashROM. Table 1 summarizes various FlashROM accessing scenarios.

Figure 4 shows the accessing of the FlashROM using the UJTAG macro. This is similar to FPGA core access, where the 7-bit address defines which of the 8 pages (three MSBs) is being read and which of the 16 bytes within the selected page (four LSBs) are being read.

Figure 5 on page 6 and Figure 6 on page 6 show the FlashROM access from the JTAG port. The FlashROM content can be read on a random basis. The three-bit address defines which page is being read or updated.

Table 1 • **FlashROM Read/Write Capabilities by Access Mode**

Access Mode	FlashROM Read	FlashROM Write
JTAG	Yes	Yes
UJTAG	Yes	No
FPGA core	Yes	No

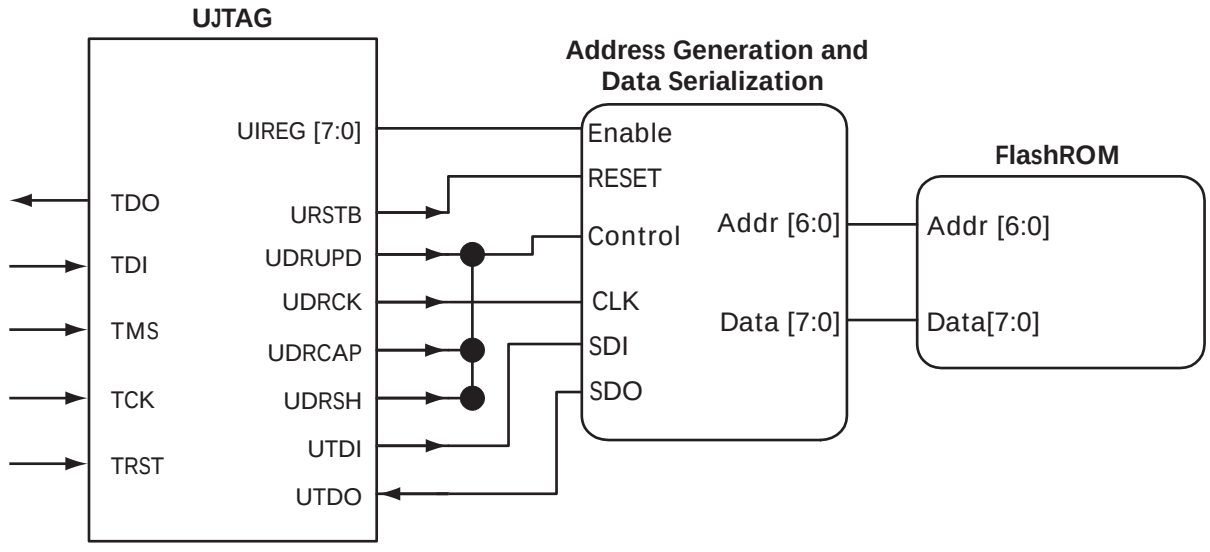

Figure 4 • **Block Diagram of Using UJTAG to Read FlashROM Contents**

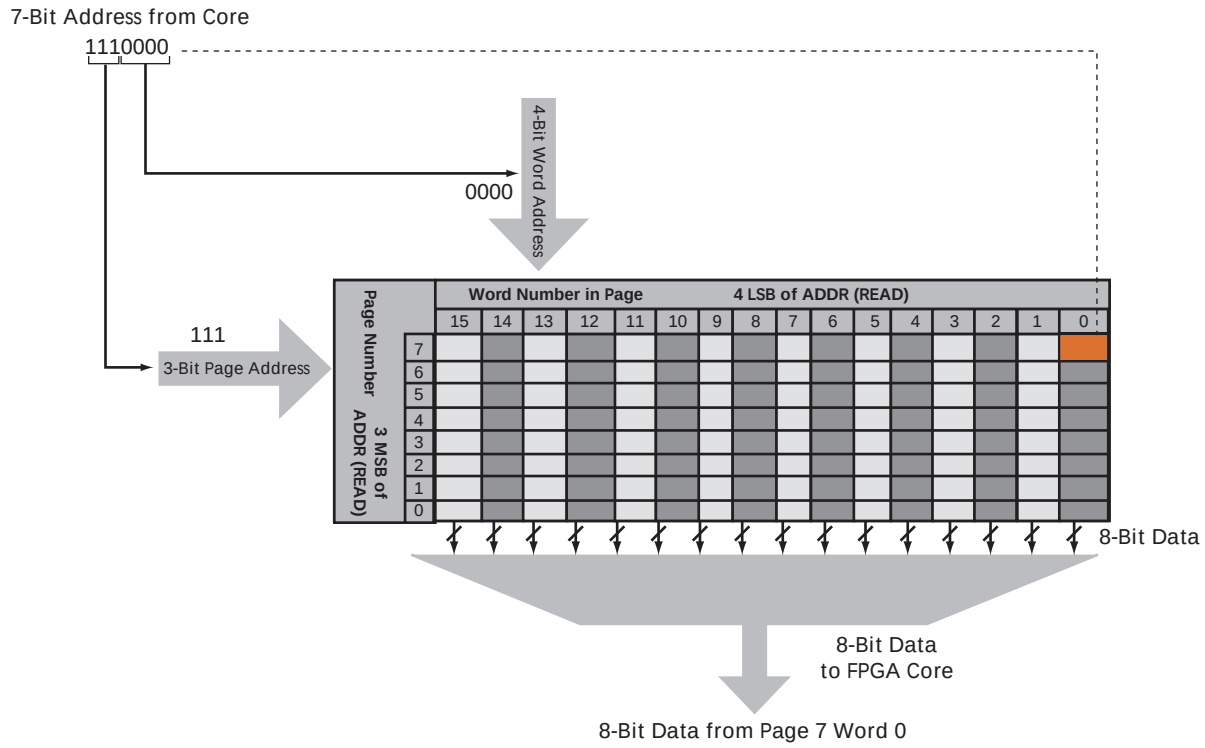

***Figure 5* • Accessing FlashROM Using FPGA Core**

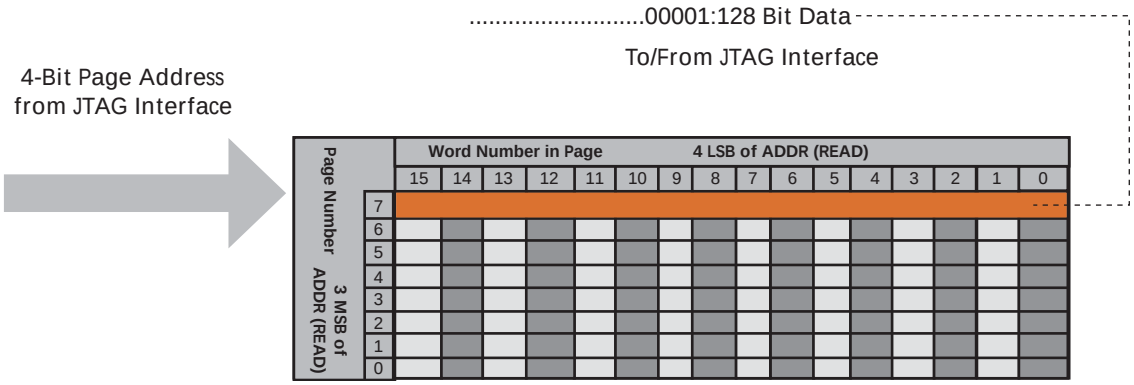

***Figure 6* • Accessing FlashROM Using JTAG Port**

FlashROM Design Flow

The Microsemi Libero® System-on-Chip (SoC) Integrated Design Environment (IDE) software has extensive FlashROM support, including FlashROM generation, instantiation, simulation, and programming. Figure 7 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

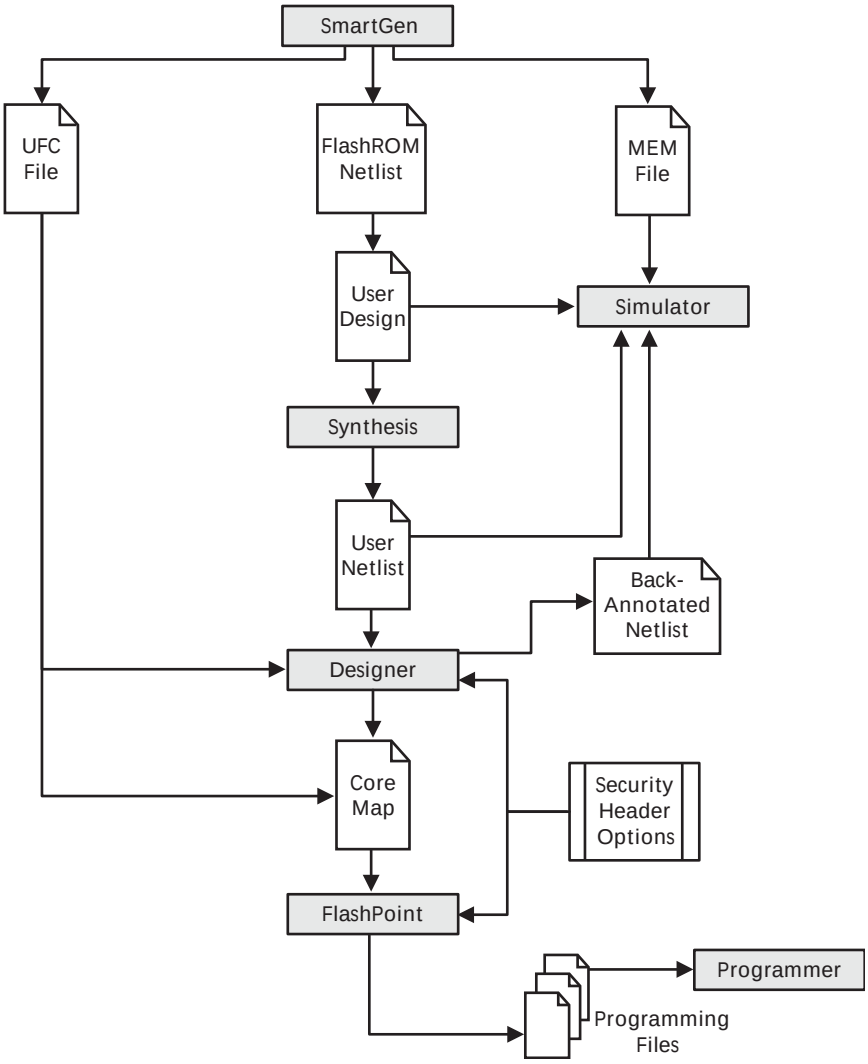

***Figure 7* • FlashROM Design Flow**

FlashROM Generation and Instantiation in the Design

The SmartGen core generator, available in Libero IDE and Designer software, is the only tool that can be used to generate the FlashROM content. SmartGen has several user-friendly features to help generate the FlashROM contents. Instead of selecting each byte and assigning values, you can create a region within a page, modify the region, and assign properties to that region. The FlashROM user interface, shown in Figure 8, includes the configuration grid, existing regions list, and properties field. The Properties field specifies the region-specific information and defines the data used for that region. You can assign values to the following properties:

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

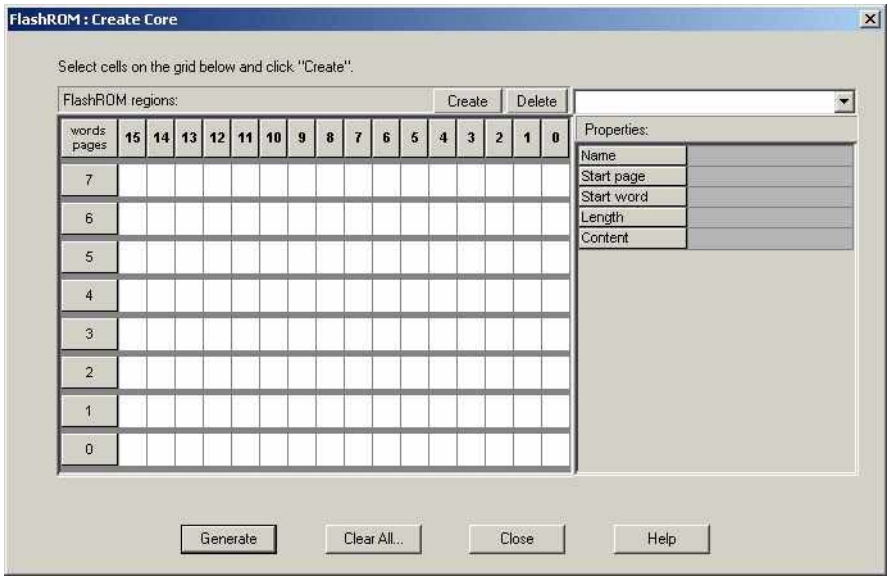

***Figure 8* • SmartGen GUI for the FlashROM**

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

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

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

architecture DEF_ARCH of  FROM_a is

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

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

    signal U_7_PIN2 : std_logic ;
    begin

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

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

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

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

Simulation of FlashROM Design

The MEM file has 128 rows of 8 bits, each representing the contents of the FlashROM that is used for simulation. For example, the first row represents page 0, byte 0; the next row is page 0, byte 1; and so the pattern continues. Note that the three MSBs of the address define the page number, and the four LSBs define the byte number. So, if you send address 0000100 to FlashROM, this corresponds to the page 0 and byte 4 location, which is the fifth row in the MEM file. SmartGen defaults to 0s for any unspecified locations of the FlashROM memory. Besides using the MEM file generated by SmartGen, you can create a binary file with 128 rows of 8 bits each and use this as a MEM file. Microsemi recommends that you use different file names if you plan to generate multiple MEM files. During simulation, Libero IDE passes the MEM file that is used as the generic file in the netlist, along with the design files and testbench. If you want to use different MEM files during simulation, you need to modify the generic file reference in the netlist.

……

UFROM0: UFROM

--generic map(MEMORYFILE => "F:\Appsnotes\FROM\test_designs\testa\smartgen\FROM_a.mem")

--generic map(MEMORYFILE => "F:\Appsnotes\FROM\test_designs\testa\smartgen\FROM_b.mem")

……

The VITAL and Verilog simulation models accept the generics passed by the netlist, read the MEM file, and perform simulation with the data in the file.

Programming File Generation for FlashROM Design

FlashPoint is the programming software used to generate the programming files for Fusion devices. Depending on the applications, you can use the FlashPoint software to generate a STAPL file with different FlashROM contents. In each case optional AES decryption is available. In order to generate a STAPL file that contains the same FPGA core content and different FlashROM contents, the FlashPoint software needs an Array Map file for the core and UFC file(s) for the FlashROM. This final STAPL file represents the combination of the logic of the FPGA core and FlashROM content.

FlashPoint generates the STAPL files that you can use to program the desired FlashROM page and/or FPGA core of the FPGA device contents. FlashPoint supports the encryption of the FlashROM content and/or the FPGA Array configuration data. In the case of using the FlashROM for device serialization, a sequence of unique FlashROM contents will be generated. When generating a programming file with multiple unique FlashROM contents, you can specify in FlashPoint whether to include all FlashROM content in a single STAPL file or generate a different STAPL file for each FlashROM (Figure 9 on page 11). The programming software (FlashPro) handles the single STAPL file that contains the FlashROM content from multiple devices. It enables you to program the FlashROM content into a series of devices sequentially. See the *FlashPro User Guide* for information on serial programming.

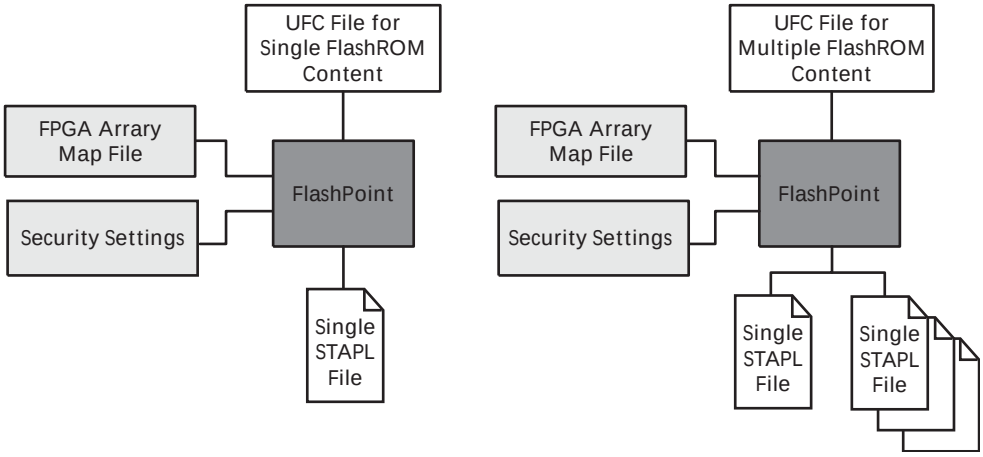

***Figure 9* • Single or Multiple Programming File Generation**

Figure 10 shows the programming file generator, which enables different STAPL file generation methods. When you select **Program FlashROM** and choose the UFC file, the FlashROM Settings window appears, as shown in Figure 11 on page 12. In this window, you can select the FlashROM page that you want to program and the data value for the configured regions. This enables you to use a different page for different programming files.

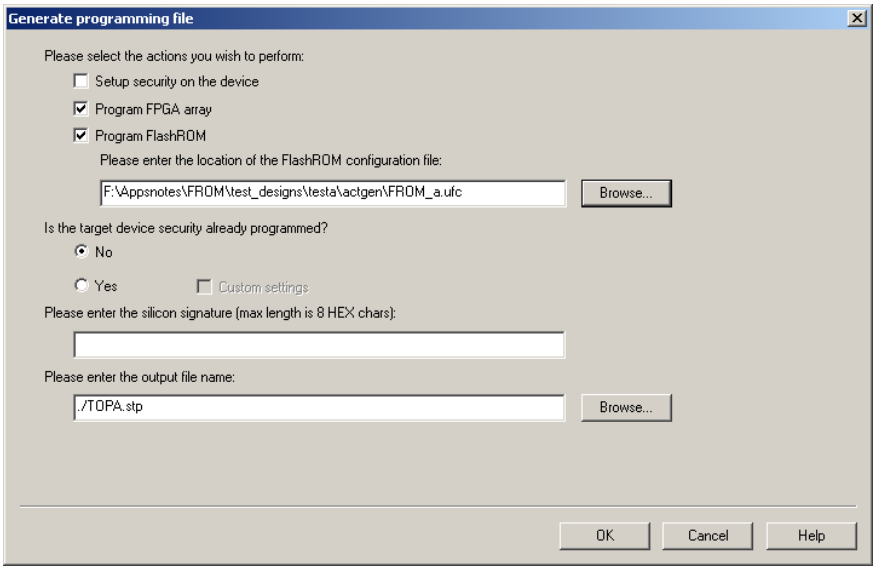

***Figure 10* • Programming File Generator**

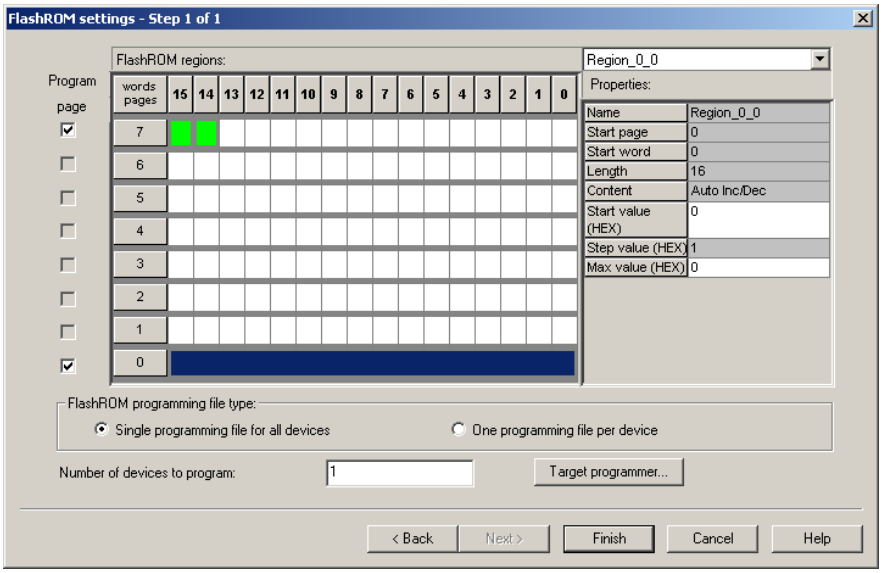

Figure 11 • **Setting FlashROM during Programming File Generation**

The programming hardware and software can load the FlashROM with the appropriate STAPL file. Programming software handles the single STAPL file that contains multiple FlashROM contents for multiple devices, and programs the FROMs in sequential order (for example, for device serialization). This feature is supported in the programming software. After programming with the STAPL file, you can run **DEVICE_INFO** to check the FlashROM content.

DEVICE_INFO displays the FlashROM content, serial number, Design Name, and checksum as shown below:

```
EXPORT IDCODE[32] = 123261CF
EXPORT SILSIG[32] = 00000000
User information :
CHECKSUM: 61A0
Design Name:        TOP
Programming Method: STAPL
Algorithm Version: 1
Programmer: UNKNOWN
=======================================
FlashROM Information :
EXPORT Region_7_0[128] = FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
=======================================
Security Setting :
Encrypted FlashROM Programming Enabled.
Encrypted FPGA Array Programming Enabled.
=======================================
```

The Libero IDE file manager recognizes the UFC and MEM files and displays them in the appropriate view. Libero IDE also recognizes the multiple programming files, if you choose the option to generate multiple files for multiple FlashROM content in Designer. These features enable a user-friendly flow for the FlashROM generation and programming in Libero IDE.

Custom Serialization Using FlashROM

You can use FlashROM for device serialization or inventory control by using the Auto Inc region or Read From File region. FlashPoint will automatically generate the serial number sequence for the Auto Inc region with the **Start Value**, **Max Value**, and **Step Value** provided. If you have a unique serial number generation scheme that you prefer, the Read From File Region allows you to import the file with your serial number scheme programmed into the region. See the *FlashPro User Guide* for custom serialization file format information.

The following steps describe how to perform device serialization or inventory control using FlashROM:

1. Generate FlashROM using SmartGen. From the **Properties** section in the **FlashROM Settings** dialog box, select **Auto Inc** or **Read From File** region. For **Auto Inc** region, specify the desired step value. You will not be able to modify this value in the FlashPoint software.
2. Go through the regular design flow and finish place-and-route.
3. Select **Programming File** in Designer and open **Generate Programming File** (Figure 10 on page 11).
4. Select **Program FlashROM** and browse to the UFC file and select **Next**. The **FlashROM Settings** window appears, as shown in Figure 11 on page 12.
5. Select the FlashROM page you want to program and the data value for the configured regions. The STAPL file generated will contain only the data that targets the selected FlashROM page.
6. Modify properties for the serialization.
   – For Auto Inc region specify the Start and Max values.
   – For Read From File region select the file name of the custom serialization file.
7. Select the FlashROM programming file type you want to generate from the two options below:
   – Single programming file for all devices option: generates one programming file with all FlashROM values.
   – One programming file per device: generates a separate programming file for each FlashROM value.
8. Enter the number of devices you want to program and generate the required programming file.
9. Open the programming software and load the programming file. The Fusion programming software, FlashPro3 and Silicon Sculptor II, supports the device serialization feature. If for some reason the device fails to program a part during serialization, the software allows you to reuse the serial data or skip the serial data. See the *FlashPro User Guide* for details.

Conclusion

The Fusion families are the only FPGAs that offer on-chip FlashROM support. This application note presented information on the FlashROM architecture, possible applications, programming, access through the JTAG and UJTAG interface, and integration into your design. In addition, the Libero IDE tool set enables easy creation and modification of the FlashROM content.

The non-volatile FlashROM block in the FPGA can be customized, enabling multiple applications.

Additionally, the security offered by the Fusion devices keeps both the contents of FlashROM and the FPGA design safe from system over-builders, system cloners, and IP thieves.

Related Documents

FlashPro User Guide

List of Changes

The following table shows important changes made in this document for each revision.

Revision	Changes	Pages
Revision 1 (March 2016)	Non-technical updates.	NA
Revision 0 (November 2005)	Initial release.	NA

**The part number is located on the last page of the document.*

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif., and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

51900113-1/3.16