Configuring SRAM FPGAs Using Actel Fusion™

Introduction

Due to the nature of SRAM technology, SRAM-based FPGAs are volatile and lose their configuration when powered off, so they must be reconfigured at every power-up. Hence, almost every system using SRAM-based FPGAs contains an additional nonvolatile memory, such as flash PROM or EEPROM, to store the configuration data and load it into the SRAM-based FPGA after power-up. In many applications, a complex programmable logic device (CPLD) is used in addition to the external configuration memory to perform the vital functions of the system necessary at power-up. One Actel Fusion device can replace several components: the configuration memory, the power management CPLD, and often the SRAM FPGA itself.

Actel Fusion devices are nonvolatile, live at power-up, and contain embedded flash memory blocks, providing a single-chip, low-cost solution for many applications. These features enable the user to integrate both CPLD and configuration memory into the Fusion device if the functionality of the SRAM-based FPGA itself cannot be incorporated in the Fusion device. In addition, Fusion FPGAs offer many other unique features, such as on-chip voltage regulators, analog-to-digital (A/D) converters, and real-time counters, which enable designers to integrate even more functionalities of their system into a single Fusion FPGA. For example, Fusion devices can incorporate and perform the following power management and system supervisory functions:

- Power-up ramp-rate and sequence control
- Voltage, current, and temperature monitoring
- Flagging supervisory system protection based on implemented flags and power-on and brownout detection

This document provides an overview of how an Actel Fusion device can be used to configure an SRAM-based FPGA, and includes reference designs.

General Implementation Overview

The flash memory in Fusion devices can be used to store the configuration bitstream of SRAM-based FPGAs. The size of the available flash memory varies between 2 Mbits and 8 Mbits, depending on the size of the Fusion device (AFS090 and AFS1500 are the smallest and largest members of the Fusion family, respectively). This memory size is sufficient for configuration of a variety of SRAM-based FPGAs (refer to each SRAM-based FPGA datasheet for the size of the memory required to store the configuration bitstream). User logic, implemented in the FPGA fabric of Fusion devices, is needed to interface between the embedded flash memory and the SRAM-based FPGA configuration pins. Depending on the user logic, the SRAM-based FPGA can be configured in various modes, such as SPI, Master parallel, Slave parallel, Master serial, and Slave serial. Figure 1 shows the simplified block diagram of the general implementation of such applications.

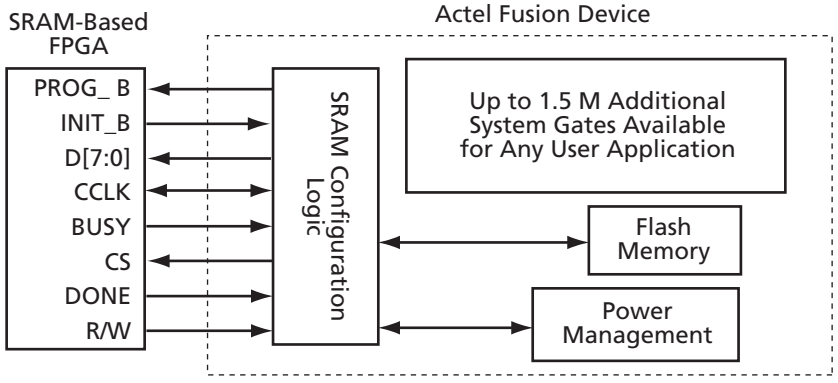

Figure 1 • **Simplified Block Diagram**

In Figure 1 on page 1, the SRAM-based FPGA is being configured in parallel mode (Master or Slave, depending on the direction of the configuration clock). The Actel CoreCFI can be used to create a common parallel flash interface to the embedded flash memory blocks within Fusion devices. In this case, the user interface of CoreCFI can be connected to the SRAM-based FPGA configuration pins, similar to any common parallel flash PROM. However, any other user logic can also be used to interface between the embedded flash memory block and the SRAM–based FPGA. The contents of the flash memory (the SRAM-based FPGA configuration bitstream) should be in Intel-Hex, Motorola-S, Actel-Hex, or Actel-Binary format. Using SmartGen and Actel Libero® Integrated Design Environment (IDE), the configuration bitstream is incorporated into a STAPL file and programmed into the embedded flash memory of Fusion devices.

Reference Design Implementation – SPI

The SPI reference design provided in this application note demonstrates the capability of configuring an SRAM FPGA using an Actel Fusion device via a Serial Peripheral Interface (SPI). Fusion embedded flash memory is used to store the configuration data. The FPGA core tiles are used to emulate a generic SPI flash ROM to program the Xilinx® FPGA via the SPI interface. The Actel Fusion Evaluation Kit (Part number: AFS-EVAL-KIT) and Xilinx Spartan™-3E Starter Kit (Part Number: HW-SPAR3E-SK-US-G) are used for the hardware testing and implementation of the reference design.

The timing diagram in Figure 2 describes the programming sequence of Spartan-3E.

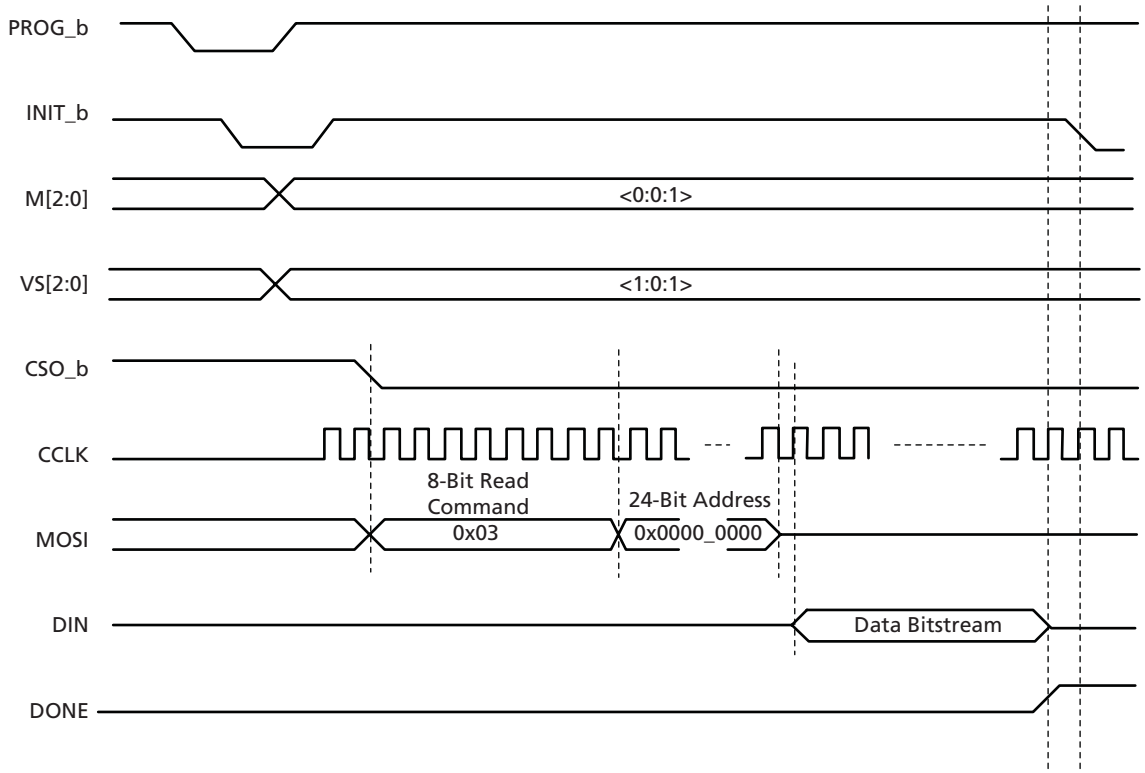

Figure 2 • **Spartan-3E SPI Configuration Timing Diagram**

During the device power-up or when PROG_B is pulsed low, the device goes through an initialization stage to clear the internal memory. At this stage, both INIT_B and DONE drive low. When INIT_B drives high again, the M[2:0] and VS[2:0] signals must be ready for the device to recognize the device programming mode and the corresponding SPI flash command. To program the Xilinx part using the SPI interface, M[2:0] is set to "001" on the Spartan-3E Starter Kit (jumper J12). The reference design on the

Fusion device is emulating a generic SPI flash ROM supporting the READ (0x03) SPI flash command, so VS[2:0] on the Spartan-3E Starter Kit is set to "101".

Once CSO_B starts driving low, Spartan-3E drives 8 bits of the SPI flash command (READ – 0x03) and 24 bits of the starting address (default value 0x0000000) of the configuration data on the MOSI pin. When the last address bit is sent on the positive clock edge of CCLK, the Fusion device will start sending the configuration bits on the following falling clock edge of CCLK. When a successful device configuration is completed, INIT_B drives low or becomes the user I/O, and DONE drives high. For detailed information on the programming procedure and timing diagrams for SRAM FPGAs, refer to their corresponding datasheets and application notes.

Figure 3 shows the simplified block diagram for this reference design. Implementation and functionality of each of the design blocks in Figure 3 is described in the following sections. Refer to "Appendix A – SPI Reference Design" on page 9 for design files and other board-level configuration information.

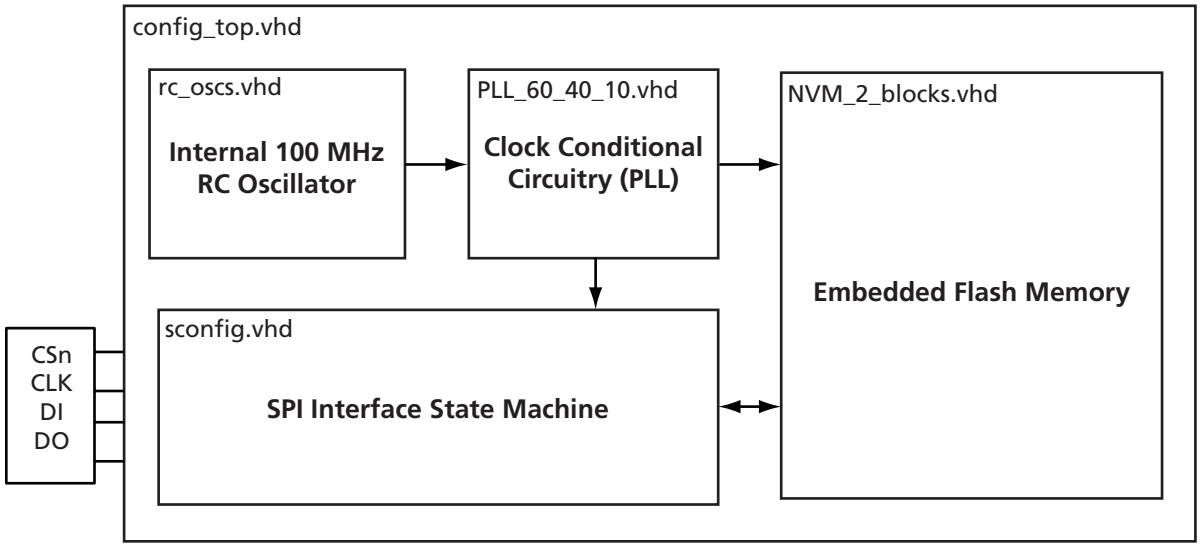

Figure 3 • **Top -Level Design Block Diagram**

RC Oscillator

The integrated RC oscillator generates a 100 MHz clock, which can be used to provide a known clock source to the internal logic or clock conditional circuitry. In this reference design, the RC oscillator is providing a 100 MHz clock source to the clock conditional circuitry. The VHDL file, *rc_oscs.vhd*, is created using SmartGen (Figure 4) in the Libero IDE software. Note that the usage of RC oscillator is not mandatory if other clock sources are available in the user's system application.

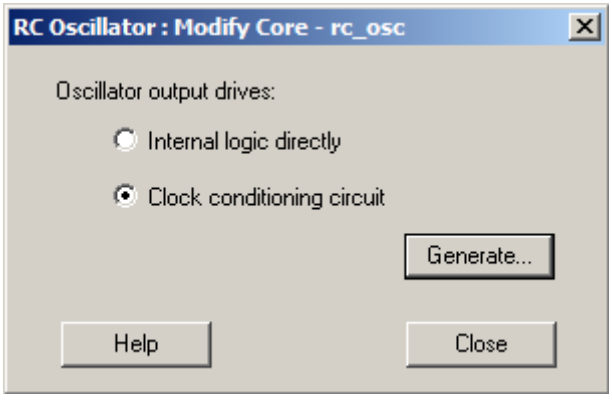

Figure 4 • **SmartGen RC Oscillator**

Clock Conditional Circuitry

The clock conditional circuitry consists of a phase-locked loop (PLL) which is used to generate different clock frequencies used in the design. Only the 60 MHz clock is used in this design, and it is connected to the embedded flash memory blocks. The 60 MHz fast clock is needed for the flash memory read so the BUSY signal from the flash memory blocks will not interrupt the serial data transmission process. This block, *PLL_60_40_10.vhd*, is generated using Smartgen in the Libero IDE software (Figure 5).

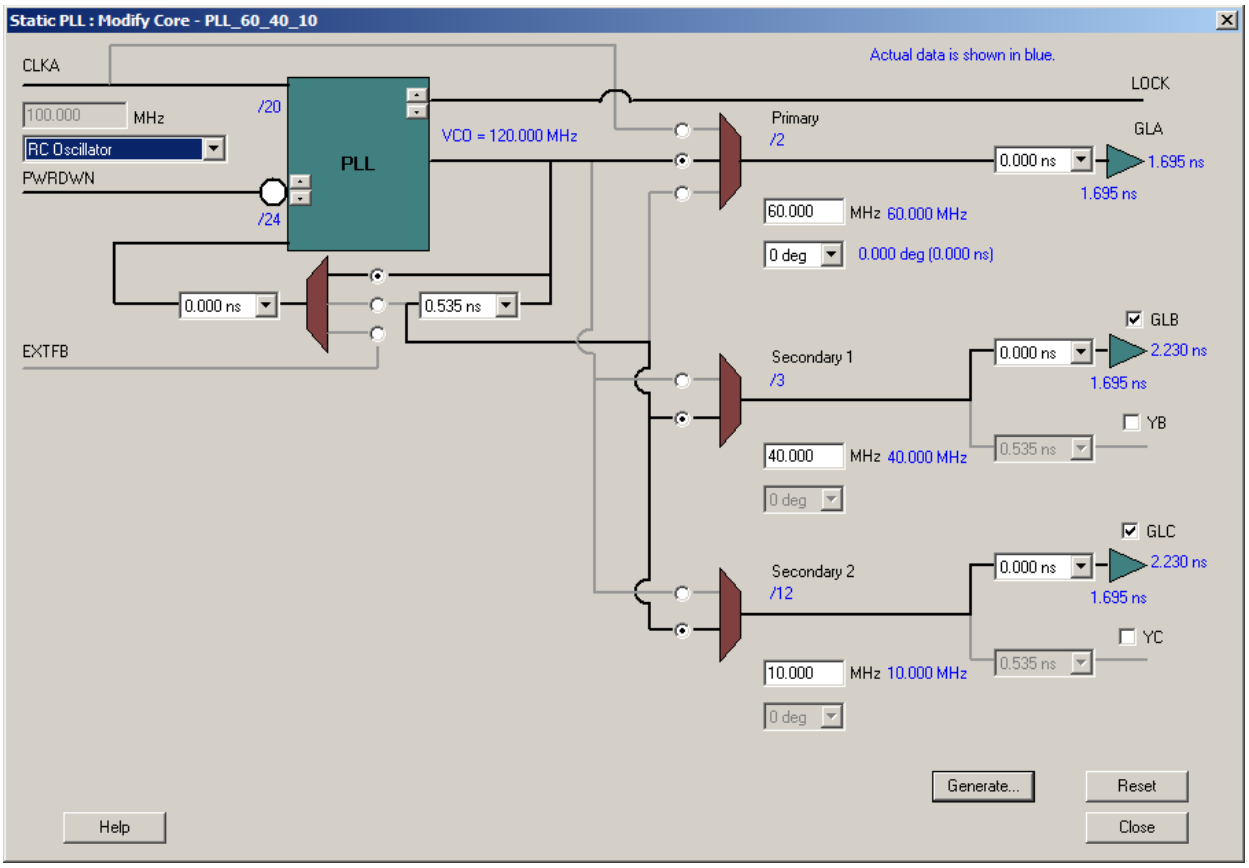

Figure 5 • **SmartGen Clock Conditional Circuitry Configuration**

Embedded Flash Memory

The embedded flash memory block in Fusion devices provides nonvolatile memory (NVM) to store the SRAM-based FPGA configuration data. Each flash memory block can store 2 Mbits of data. Depending on the size of the Fusion device, it can have up to 4 blocks of 2-Mbit flash memory blocks.

In this reference design, the scrolling text demo design configuration data file (*s3esk_startup.bit*), which is provided with the Xilinx starter kit, is stored in the Fusion flash memory block. The *s3esk_startup.bit* file is formatted to program the Xilinx PROM. In order to store this configuration data in Fusion embedded flash memory, the configuration data must be formatted appropriately using the iMPACT tool included in the Xilinx ISE software (Figure 6 on page 5). In iMPACT, create a new project and select the configuration data file (*s3esk_startup.bit*) for the targeted device (xc3s500e). Once this is completed, select the PROM File formatter and configure the files as shown in Figure 6 on page 5. Note that the third-party SPI PROM should be selected; this reference design serially puts out data bytes MSB first while Xilinx PROM puts out data LSB first.

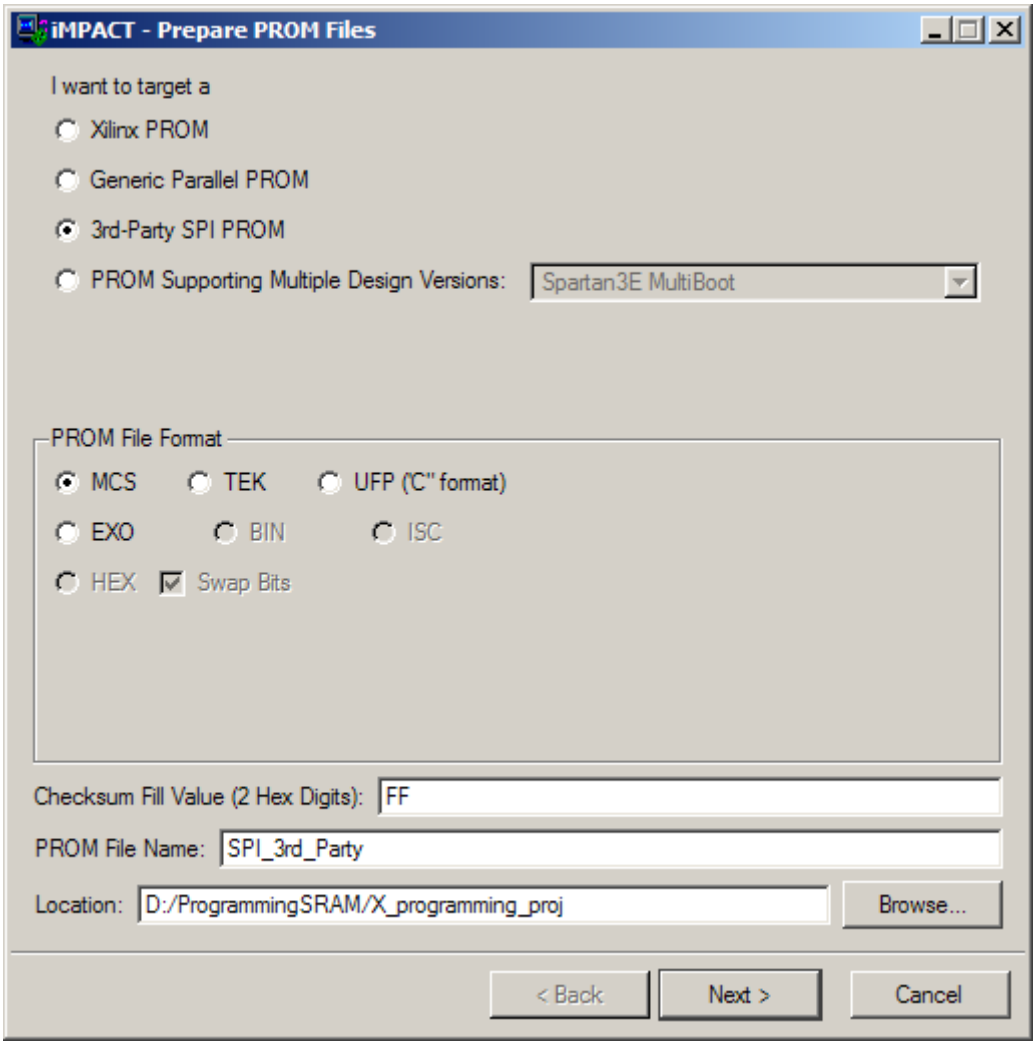

Figure 6 • **iMPACT PROM Formatter**

As the size of the created *SPI_3rd_Party.mcs* file (intel-hex file format) is larger than two Mbits, two flash memory blocks are needed to store all the configuration data. A software program is used to split the hex file into two different hex files.

Download the software tool from http://www.keil.com/support/docs/963.htm.

Step 1: hex2bin test_spi.mcs test_spi.bin (convert hex to bin)

Step 2: bin2hex /L262144 /4 test_spi.bin test_one.hex (part1)

Step 3: bin2hex /L262144 /4 /I262144 test_spi.bin test_two.hex (part2)

Using the SmartGen Flash Memory System Builder (Figure 7 and Figure 8 on page 7), the corresponding hex file can be associated with each block so that the proper files used for simulation are generated. NVM1 (*NVM1.vhd*) is used to store the first part of the configuration file and NVM2 (*NVM2.vhd*) is used to store the second part of the configuration file. *NVM_2_blocks.vhd* is the wrapper for NVM1 and NVM2.

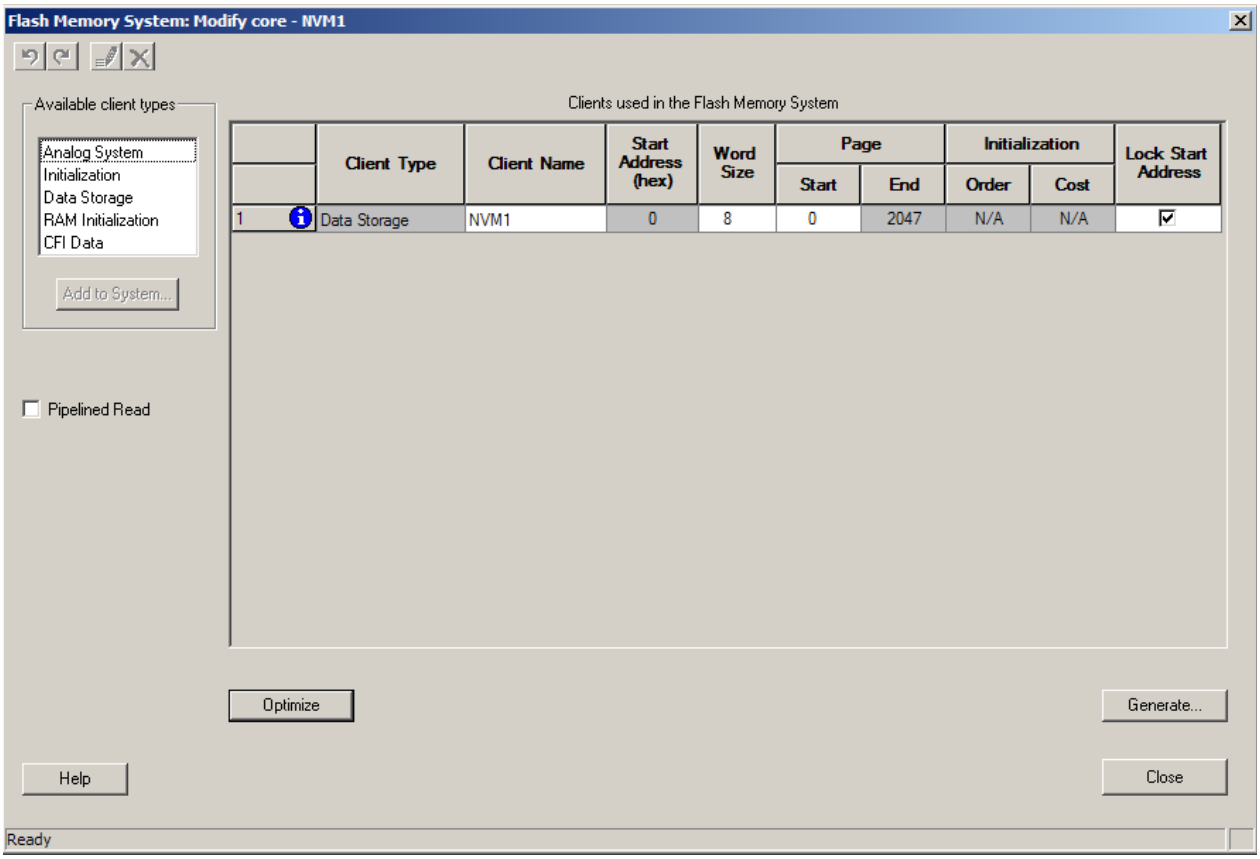

Figure 7 • **SmartGen Core Flash Memory System Builder**

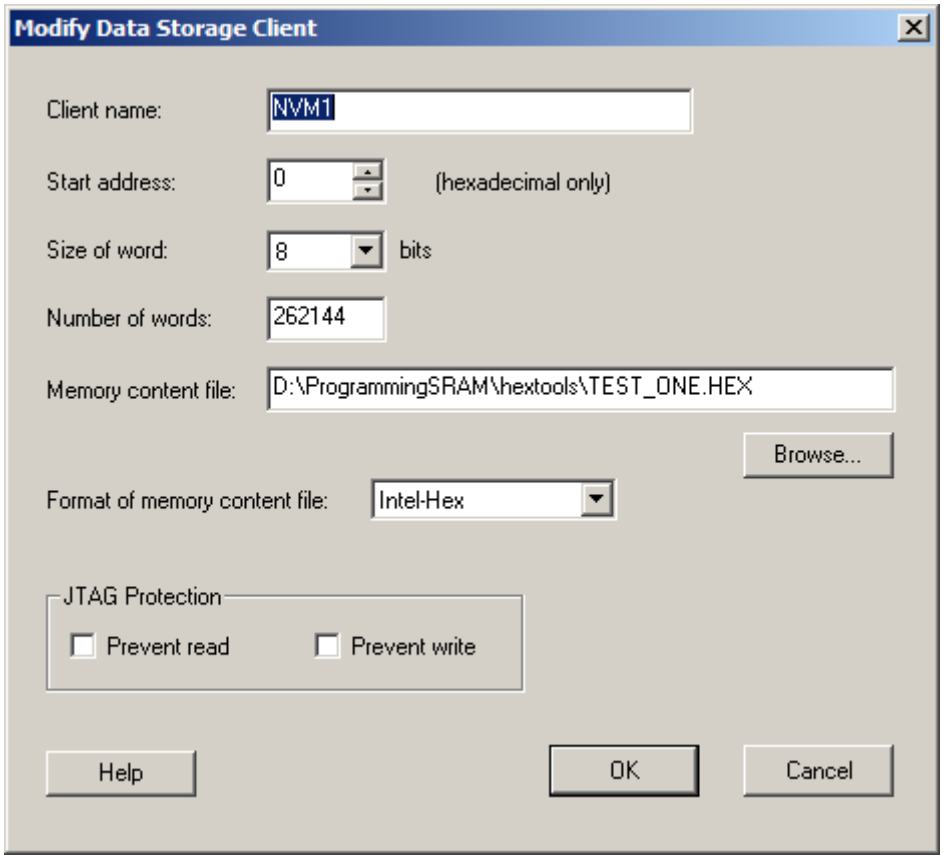

Figure 8 • **Configure the Data Storage Client**

SPI Interface Block

The SPI interface component has a simple state machine to interface the embedded flash memory with the Xilinx device via the standard SPI communication protocol. As shown in the timing diagram, Figure 2 on page 2, the state machine captures the first 8 bits for the command and if it matches the read command (0x03), the following 24 bits of the read address is captured one byte at a time in the next 3 states. The state machine then enters the READ state, in which the data configuration bits are loaded from the embedded flash memory into the shift registers and are shifted out MSB first on each subsequent falling edge of the clock. The data configuration bit continues to shift out of the Fusion device until it is completed, which is indicated by CSn = 1, driven by the connected Xilinx device. Figure 9 shows a section of the post-layout simulation of this reference design.

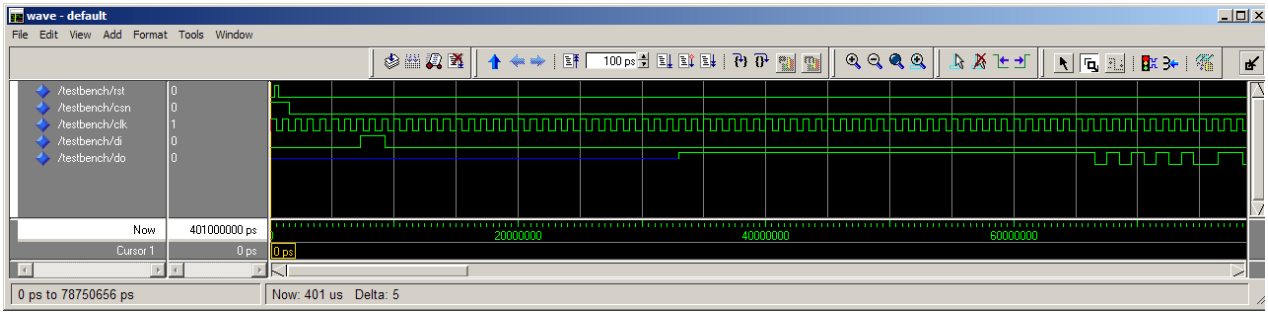

Figure 9 • **Post-Layout Simulation Results**

Reference Design Implementation – Master Serial

While SPI is a popular serial interface, not all SRAM-based FPGAs support this interface. Instead, other serial or parallel interfaces are supported. In this section, another reference design is provided to program the SRAM-based FPGA in Master serial mode. In Master serial mode, the SRAM-based FPGA is providing the clock signal while the Fusion FPGA is acting as a Slave device. Figure 10 shows the simplified block diagram of this reference design. For detailed information regarding the Master serial configuration mode supported by the SRAM-based FPGA, refer to the corresponding datasheets and application notes.

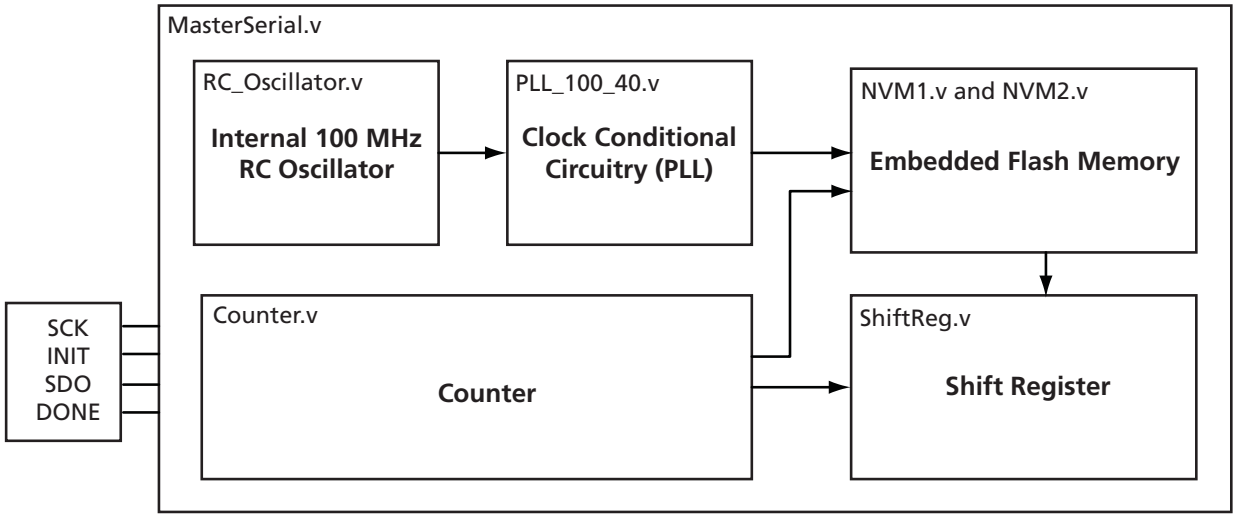

Figure 10 • **Master Serial Top-Level Design Block Diagram**

Similar to the SPI reference design, the Master serial design uses a 100 MHz RC oscillator as the clock source and the PLL is used to generate a fast clock for reading the embedded flash memory. *Counter.v* generates the read address for the embedded flash memory and the load signal for the shift register. The load signal from the counter is generated every 8 bits to load 1 byte of data from the flash memory. The shift register simply shifts the captured data out on SDO port. Figure 11 shows the post-layout simulation of this reference design. Refer to "Appendix B – Master Serial Reference Design" on page 10 for design file information.

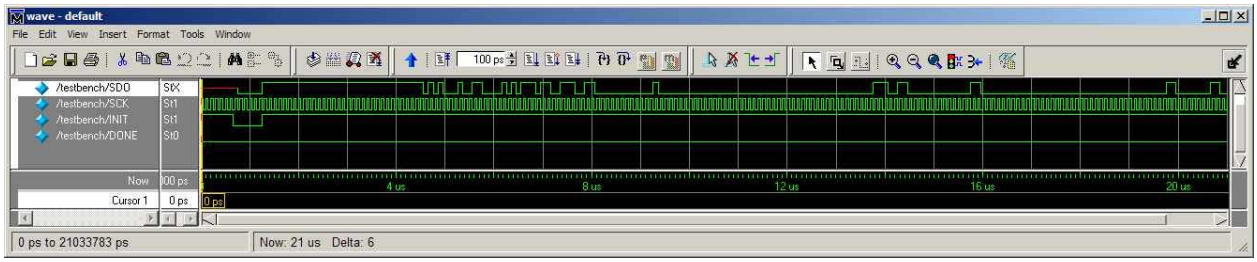

Figure 11 • **Master Serial Reference Design Post-Layout Simulation**

Conclusion

Actel Fusion FPGAs contain an embedded flash memory block, which can be used to store the configuration bitstream of an SRAM-based FPGA. Additionally, the power and clock management of the system can also be integrated into Fusion devices, providing a cost-effective solution that incorporates configuration, power management, clock management, and many other applications into a single chip.

Appendix A – SPI Reference Design

Design Files Summary

The full design can be downloaded at http://www.actel.com/documents/Fusion_MasterSerial_DF.zip. Table 1 gives descriptions of key design source files.

Table 1 • **Design Files Description**

Files	Functionality
config_top.vhd	Top-level wrapper
NVM_2_blocks.vhd	Wrapper for the two flash memory blocks
NVM1.vhd	SmartGen – Flash Memory data storage client
NVM2.vhd	SmartGen – Flash Memory data storage client
rc_osc.vhd	SmartGen – 100 MHz RC oscillator
PLL_60_40_10.vhd	SmartGen – PLL for generating different clock signals
sconfig.vhd	State machines for the bit shifting

Programming File

Table 2 • **SRAM FPGA Configuration Data**

Files	Description
s3esk_startup.bit	Xilinx Starter Kit demo design configuration data
test_one.hex	Configuration data stored on NVM1
test_two.hex	Configuration data stored on NVM2

Pin List

Table 3 • **Pin Assignment and Connection**

Xilinx Board (J12 header)	Fusion Board
SEL	Csn - K14
SDI	DI – M5
SDO	DO – B6
SCK	CLK – G4
GND	GND – G5
GND	GND – GND on J13 header

Using the SPI header (J12) on the Xilinx Spartan-3E Starter Kit, this reference design on Fusion demonstrates the capability of programming the SRAM FPGA via the SPI interface.

Board-Level Configuration for the Xilinx Starter Kit

1. SPI configuration jumper setting: J30, M0–open, M1–closed, M2–closed
2. VS[2:0] = 0x03. Connect IO_L25N/VS1/A18 (V15) to GND.

Appendix B – Master Serial Reference Design

Design Files Summary

The full design can be downloaded at http://www.actel.com/documents/SPI_SRAM_CONFIG_DF.zip. Table 4 gives descriptions of key design source files.

Table 4 • **Design Files Description**

Files	Functionality
MasterSerial.v	Top-level wrapper
counter.v	Counter to generate the read address and load signal for the shift register
NVM1.vhd	SmartGen – Flash Memory data storage client
NVM2.vhd	SmartGen – Flash Memory data storage client
RC_Oscillator.v	SmartGen – 100 MHz RC oscillator
PLL_100_60.v	SmartGen – PLL for generating different clock signals
ShiftReg.v	Shift register loads the data from the flash memory and shifts the data (MSB first) on the output pin, SDO

www.actel.com

Actel Corporation
2061 Stierlin Court
Mountain View, CA
94043-4655 USA
Phone 650.318.4200
Fax 650.318.4600

Actel Europe Ltd.
River Court, Meadows Business Park
Station Approach, Blackwater
Camberley Surrey GU17 9AB
United Kingdom
Phone +44 (0) 1276 609 300
Fax +44 (0) 1276 607 540

Actel Japan
EXOS Ebisu Bldg. 4F
1-24-14 Ebisu Shibuya-ku
Tokyo 150 Japan
Phone +81.03.3445.7671
Fax +81.03.3445.7668
www.jp.actel.com

Actel Hong Kong
Suite 2114, Two Pacific Place
88 Queensway, Admiralty
Hong Kong
Phone +852 2185 6460
Fax +852 2185 6488
www.actel.com.cn

59100168-0/9.07