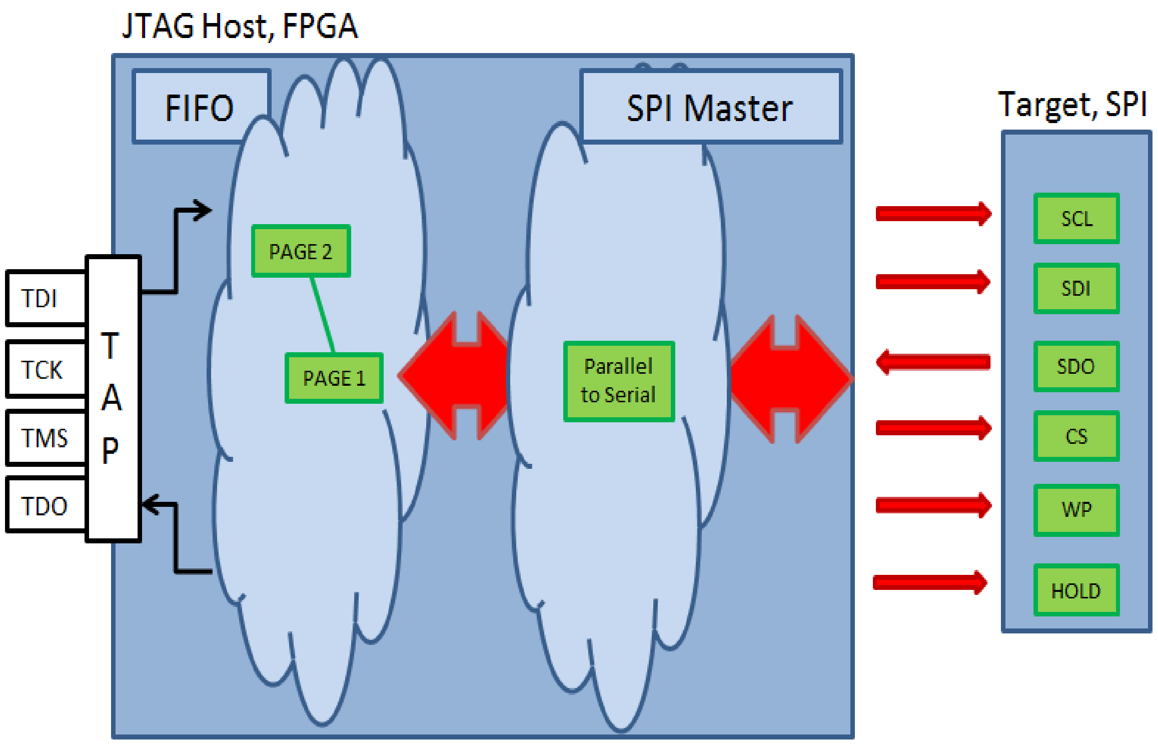
AT-SPEED SPI FLASH / EEPROM PROGRAMMING USING FPGA AND JTAG



BY MICHAEL R. JOHNSON

ASSET™

By Michael R. Johnson – Product Manager

Michael R. Johnson presently serves as Product Manager for Boundary-Scan Test (BST) and FPGA-Controlled Test (FCT) for ASSET InterTech. He also serves as manager of ASSET InterTech's Support services. As Product Manager, Michael is responsible for providing strategic direction for these products while coordinating with ASSET's cross-functional teams to bring the company's goals to fruition.

Prior to joining ASSET InterTech in 2007 as a Sr. Applications Engineer, Michael's background included roles as a Cellular System Engineer with Nortel Networks and a Hardware Design Engineer with Alcatel USA. While at Alcatel USA, Michael's emphasis was high-speed PCB design and layout for digital cross-connect and fiber-optic systems.

Michael earned the Bachelors of Science in Electrical Engineering from Southern University and A&M College located in Baton Rouge, Louisiana and the Masters in Business Administration from Amberton University located in Garland, Texas.

Table of Contents

Table of Figures

Executive Summary

One of the great challenges for designers and manufacturing engineers is programming flash and EEPROM memory as fast as the EEPROM can receive the data and while the device is in-system (that is, after the memories have been soldered down onto a printed circuit board (PCB)). This eBook discusses several methods that utilize an FPGA on the PCB to quickly and efficiently program these memory devices, which typically interface to the SPI (Serial Peripheral Interface) bus. Some of these methods are faster or simpler than the others. These methods are explained and described, starting with the slowest and concluding with the fastest method. In addition, the eBook offers test results showing the actual programming performance results for the three methods described as well as a manufacturing cost savings analysis relative to increasing programming times.

Background

During design, when prototypes of a new board design are being brought up to verify their functionality, engineers often need to program memory devices like EEPROMs and flash memory in-system because removing already soldered down memory to program the devices would unnecessarily lengthen the board bring-up phase and, quite possibly, delay the design's transition into volume manufacturing. Moreover, removing soldered-down chips from a PCB frequently damages them in the process.

Often, the software code that will be deployed in the field in the EEPROMs or flash memory devices is still being modified during board bring-up, causing the memory to be reprogrammed over the course of prototype bring-up with newly revised software. In addition, in some cases, the on-board SPI memories will not be loaded with software until a certain software image is selected by the end user just prior to shipping to the user. In both of these scenarios, the programming of SPI memories in-system increases the designer's and the manufacturing engineer's flexibility and increases the efficiency of the entire design and manufacturing process.

Programming from a Connected FPGA

Practically all FPGAs come with an IEEE boundary-scan 1149.1 (JTAG) Test Access Port (TAP). Because most memory devices do not include a JTAG port or any on-chip boundary-scan resources, they cannot be programmed in-system directly from the PCB's boundary-scan connector, but they can be programmed indirectly with boundary-scan by going through an FPGA connected to the memory devices. SPI memory devices can be accessed by way of the FPGA's boundary-scan TAP and cells because boundary-scan tools connected to the FPGA's JTAG port are able to emulate the SPI bus and its protocols. Some boundary-scan tools provide a complete set of the SPI cycle definitions for read and write operations built into the tool, and employ a model-based approach for device programming. This simplifies and significantly shortens the development of programming routines for the end user in the way that the user can be unaware of how the SPI programming protocol works but still program the device in-system.

Boundary Scan Chain Access Method

The first method is the most straightforward, but also the slowest. The data to be programmed is simply routed through the FPGA's internal boundary scan chain and output to the targeted SPI memory device. (Figure 1)

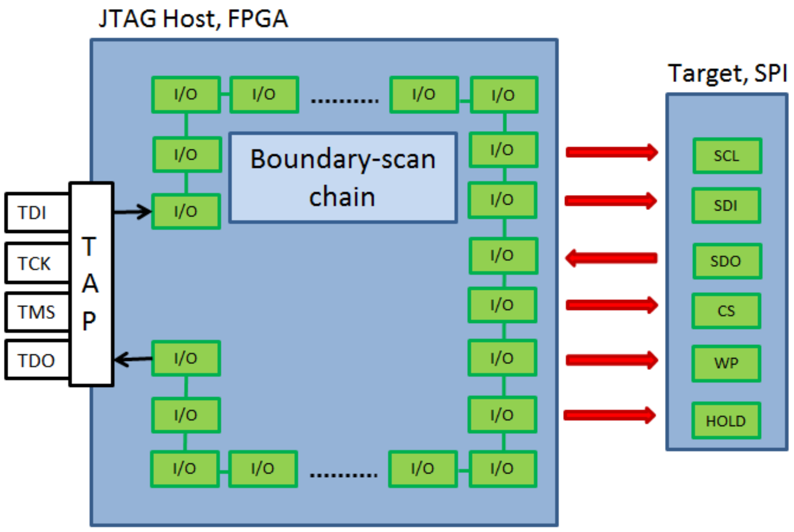


Figure 1: Boundary-scan chain access method

On most high-end FPGAs the boundary-scan chain can be 1,000 cells, and longer. All of these cells must be scanned through before the data reaches the FPGA's SPI pins. Consequently, every change to the FPGA's SPI data pins or clock pin requires a new scan through all 1,000 or so cells on the device, resulting in slower programming speeds for this boundary-scan chain access method. The factors that limit the speed of this method are usually the length of the boundary-scan scan chain in the FPGA and, in some cases, the speed at which the boundary-scan test clock (TCK) can be run on the PCB.

Since this is not the fastest of the boundary-scan-based programming methods, it is typically best suited to applications where only a small amount of code or data is being loaded into the memory devices, such as board revision information or timestamp updates to the current software image.

Boundary-Scan Input/Output - Shortened Boundary-Scan Chain Method

Another method that utilizes the FPGA’s boundary-scan chain requires the user to develop and deploy firmware in the FPGA that would shorten the scan path. (Figure 2). This is known as the Boundary-Scan I/O IP. The boundary-scan tool could then use this shorter scan path through the FPGA and program the SPI memories faster, in a similar fashion as described above in the Boundary-Scan Chain Access Method. Depending on the application and the implementation of this method, the shortened scan chain is typically fewer than 10 or 20 cells. For an FPGA whose boundary-scan chain is at least 1,000 cells long, this method can speed up the programming process significantly. To determine just how much faster this method is than the previous method, divide the total number of boundary-scan cells in the FPGA’s scan chain by the number of cells in the shortened programmed scan chain. For example, if a scan chain whose length is 1,000 cells could be reduced to a length of 5 cells, this method could be 200 times faster. Still, this method will not reach the full speed of SPI EEROMs because of the overhead generated by scanning through the cells on the FPGA’s scan chain as well as any limits imposed by the device’s boundary-scan test clock. In addition, each transition of SPI bus signals will require another scan.

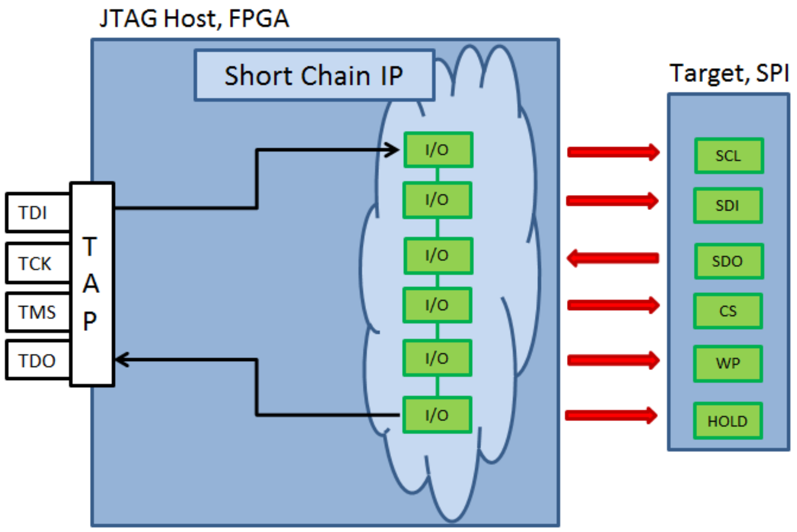


Figure 2: Shortened boundary-scan chain method

If the amount of data to be programmed into the memory device is less than 1megabits, this method can be an adequate solution. It is typically only deployed to program small software images or small amounts of data.

SPI Flash IP - SPI Master IP with FIFO Memory

For this method a dedicated SPI Master IP will be programmed into the FPGA along with a memory area, which will usually be implemented as FIFO memory. (Figure 4) This configuration of a SPI Master with FIFO memory is called a SPI Flash IP. The SPI Master will read the data to be programmed into the connected memory devices from the FPGA's memory area. The size of the FIFO will typically complement the page sizes of the SPI memory devices. After the boundary-scan tool has streamed a full page of data through the FPGA's boundary-scan TAP port into the on-chip FIFO memory space, the SPI Master IP will grab the data and start programming the SPI memory at the access speed of the memory. The boundary-scan tool can continually refill the FIFO with programming data while the SPI Master IP is programming the connected EEPROM or flash memory with the previous page of data. A cleverly designed SPI Master IP and a high-throughput boundary-scan controller are typically able to reach the maximum programming speed of the SPI memory device.

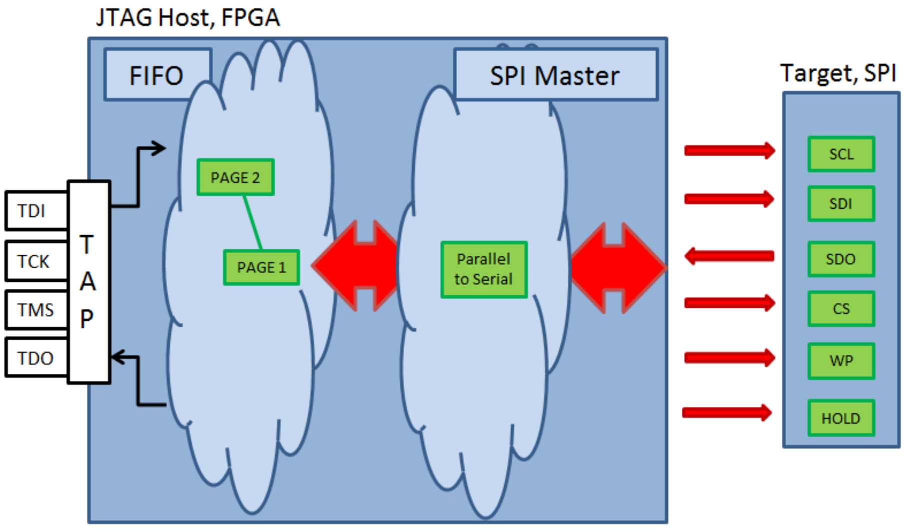


Figure 3: SPI master IP with FIFO memory

Today's boundary-scan controllers are capable of high throughput speeds. As a result, the controller's throughput will not typically be a limiting factor for programming speed with this method. High programming speeds can be achieved with this method, making it suitable for any amount of programming data, any size FPGA and any size memory devices.

Real-World Programming Performance

These methods outlined were tested and their performance measured with a Xilinx Spartan 6 FPGA. A common off-the-shelf SPI EEPROM device was connected to the FPGA. The programming times were affected by the TCK speed of the FPGA and the length of the boundary-scan register chain in the FPGA. With tools and IP that support the recently introduced Dual and Quad programming protocols for SPI devices, programming times could be accelerated even further. Table 1 contains test result for three of the programming methods

Table 1: Programming performance

Access Method	TCK	Boundary-Scan Chain Length	Erase/Program/Verify (1MByte data)
Boundary-Scan Chain Access Method	10 MHz	612 cells	34 minutes 56 seconds
BST_IO IP	10 MHz	4 cells	4 minutes 2 seconds
SPI Flash IP	10 MHz	Embedded Instrument	1 minute 21 seconds

ScanWorks®
Platform for Embedded Instruments

Manufacturing Cost Savings

Assuming a typical cost-per-minute in a manufacturing environment of $1, the cost savings from faster programming times can be considerable. For example, if 100,000 units are manufactured yearly and programming times are reduced by one minute per unit, the saving amounts to $100,000 per year for this particular product. Or, to state it another way, each unit can be produced for $1 less.

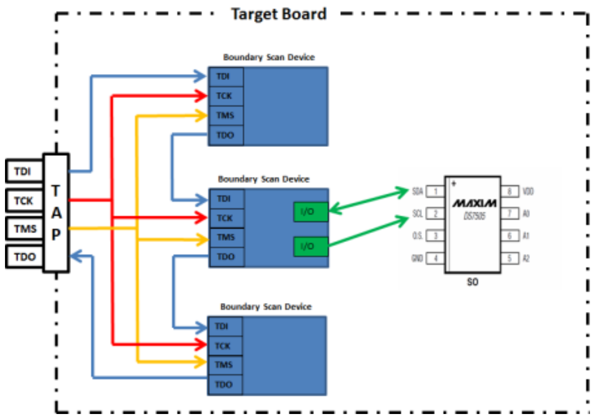
In addition to reduced manufacturing costs, faster programming times can generate other benefits, including products will be delivered from the factory to the market in a more timely fashion, memory devices can be configured and reconfigured while they are soldered to the board, and devices can be readily reconfigured in the field and at repair sites. Capital equipment costs for fixtures and hardware can also be reduced because some tools allow device programming and tests to be applied from the same JTAG controller.

Summary

This paper has described several methods for programming soldered-down SPI memories in-system from a connected FPGA. These methods only require external off-board access to the JTAG/boundary-scan port on the FPGA. This access is typically a necessity because FPGA development tools require it. There are several benefits to programming SPI memory in-system and with the help of a boundary-scan tool. The economic benefits and cost savings can be readily calculated. SPI memory devices can be programmed in a timely fashion: when they need to be programmed and where they are. In addition, the boundary-scan tool that is doing the programming can also detect production-related faults on the SPI bus, as well as other parts of the board where boundary-scan devices are deployed.

Learn More

Learn more about SPI-related topics, such as how to perform functional tests and program SPI and I2C system monitors with JTAG.



Register Today!

ScanWorks®
Platform for Embedded Instruments