International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in

Vol.3., Issue.3, 2015

RESEARCH ARTICLE

ISSN: 2321-7758

SPI FLASH MEMORY CONFIGURATION IN ZYNQ FPGA USING MICROPROCESSOR

G.S.MONISHA[1],M.JAGADEESWARI[2]

[1,2]ECE-PG,SREC,Coimbatore,Tamilnadu,India

Article Received:06/05/2015 | Article Revised on:10/05/2015 | Article Accepted on:14/05/2015

ABSTRACT

FPGA becoming more popular, many designers want to reduce their component count and increase flexibility. To accomplish both of these goals, a microprocessor already available in the system can be used to configure an FPGA. Here is a simple and efficient FPGA configuration method that utilizes a microprocessor to configure an FPGA device from a Serial Peripheral Interface (SPI) flash memory. This method reduces hardware components, board space, and costs. Application firmware are included to illustrate the methodology.

keyword: Embedded development Kit, Software Development Kit, Field Programmable gate array, Master Out Slave In, Master In Slave Out, Quad Serial Peripheral Interface, Serial Peripheral Interface, Serial Clock, Slave Select.

I.INTRODUCTION

An FPGA is a type of integrated circuit (IC) that can be programmed for different algorithms after fabrication. Modern FPGA devices consist of up to two million logic cells that can be configured to implement a variety of software algorithms.

An FPGA provides significant cost advantages in comparison to an IC development effort and offers the same level of performance in most cases [4]. Today's systems demand greater functionality in less space and at reduced cost. In addition, each generation of Xilinx FPGAs delivers higher performance and increased capabilities. Although Xilinx FPGAs support direct configuration from third-party flash, an embedded processor-based configuration solution can allow for advanced FPGA configuration applications[1]. Xilinx 7 series FPGAs are configured by loading application-specific configuration data a bitstream into internal memory. 7 series FPGAs can load themselves from an external nonvolatile memory device or they can be configured by an external smart source, such as a microprocessor, DSP processor, microcontroller, PC, or board tester[2]. The FPGAs can be configured directly from the Flash Memory using Microprocessor. This method is to store the user firmware as well as the configuration bitfile on a flash memory device attached to a microprocessor. The microprocessor reads the bitfile through an SPI interface and in turn sends out the bitstream to the FPGA via a slave Serial or Slave SelectMAP interface. This eliminates the need for an extra PROM for FPGA configuration.

II.RELATED WORK

This survey will provide critical reviews, concepts, advantages and disadvantages among survey results. This contribution adds more thoughtful ideas in configuring an FPGA. Matt Nelson et.al in [1]

implemented the configuration of FPGA from an embedded processor. In this the microprocessor whose primary purpose is to perform other tasks can also be used to coordinate the loading of configuration data into a Xilinx FPGA. The processor provides greater flexibility, for example, in choosing which of multiple configuration files to program into the FPGA Microprocessor configuration of 7 series FPGAs is accomplished in either Slave Serial or Slave SelectMAP mode. Mattics Phi et.al in [2] configured the FPGA by loading application-specific configuration data a bitstream into internal memory. 7 series FPGAs can load themselves from an external nonvolatile memory device or they can be configured by an external smart source, such as a microprocessor, DSP processor, microcontroller, PC, or board tester. He described two datapaths for FPGA configuration. Arthur Yang et.al in [3] implemented the configuration of FPGA using serial peripheral interface(SPI). Here he described the advantages of selecting a serial peripheral interface (SPI) flash as the configuration memory storage for the Xilinx 7 series FPGAs. ISE Design Suite has been used for in-system programming of the SPI flash via the FPGA. This allows for configuration flexibility during the debugging stages of development. In this paper Quad Serial Peripheral Interface(QSPI) is used. The SPI flash memories use a 4-wire synchronous serial data bus. The SPI flash configuration requires only four pins, which allows 1- or 2-bit data width for delivery of the configuration bitstream. QSPI flash devices provides six pins to enable 4-bit data width, thereby decreasing configuration time appropriately.

III.PROPOSED SPI FLASH MEMORY CONFIGURATION IN ZYNQ FPGA

Today's systems demand greater functionality in less space and at reduced cost. In addition, each generation of Xilinx FPGAs delivers higher performance and increased capabilities. For reducing the hardware component, board space and cost, a simple and efficient FPGA configuration method that utilizes a microprocessor to configure an FPGA device from a Serial Peripheral Interface (SPI) flash memory is used. In this method the user application firmware as well as the configuration bitfile are stored on a flash memory device attached to a microprocessor. The microprocessor reads the bitfile through an SPI interface and in turn sends out the bitstream to the FPGA via a slave Serial or Slave SelectMAP interface. This eliminates the need for an extra PROM for FPGA configuration.

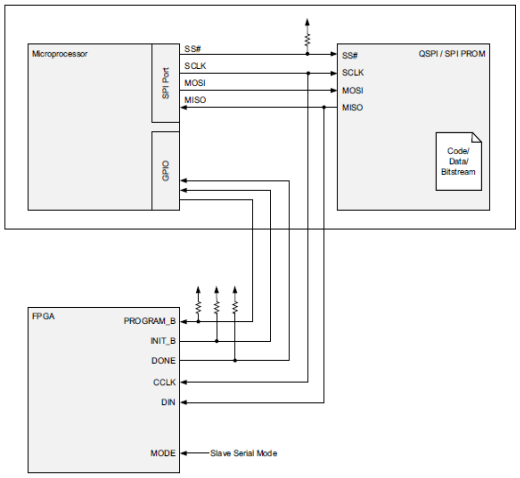

Figure 1: Block Diagram of SPI Flash Memory configuration

The flash memory attached to the microprocessor stores the user application firmware and FPGA bitfiles. However the microprocessor does not configure the FPGA through the FPGA configuration port directly. Instead, the FPGA slave serial DIN and CCLK pins connect to the SPI bus between the flash memory and the microprocessor. Configuration by this method is possible because the slave serial interface and configuration sequence are compatible with SPI protocol coincidentally.

IV. SERIAL PERIPHERAL INTERFACE

Xilinx FPGAs require that a configuration bitstream is delivered at power-up. The SPI flash memories use a 4-wire synchronous serial data bus. The SPI flash configuration requires only four pins, which allows 1- or 2-bit data width for delivery of the configuration bitstream. The QSPI
flash devices provides six pins to enable 4-bit data width, thereby decreasing configuration time appropriately. FPGA configuration via the SPI interface is a very low pin
count configuration solution.

Figure 2: General Block of SPI

The SPI bus is a synchronous serial communication interface specification used for communication between the Flash Memory and the Zynq FPGA.

SPI communicate in full duplex mode using a master slave architecture with a single master. The master device originates the frames for reading and writing. Multiple slave device are supported through selection with individual slave select (SS).

The SPI bus specifies four logic signals:

- Serial Clock(**SCLK**) provides the output from the master
- Master Out Slave In(**MOSI**) provides output from master
- Master In Slave Out(**MISO**) provides output from slave
- Slave Select(**SS**) when active low it provides output from master

The SPI bus can operate with a single master device and with one or more slave devices. If a single slave device is used the SS pin can be fixed to logic low. If there are multiple slave devices an independent SS signal is required from the master for each slave devices.

IV.1 DATA TRANSMISSION

To begin communication, the bus master configures the clock, using a frequency supported by the slave devices, typically up to a few MHz. the master then selects the slave device with a logic level 0 on the select line. During each SPI clock cycle, a full duplex transmission occurs. The master sends a bit on the MOSI line and the slave reads it, while a slave sends a bit on the MISO line and the master reads it.

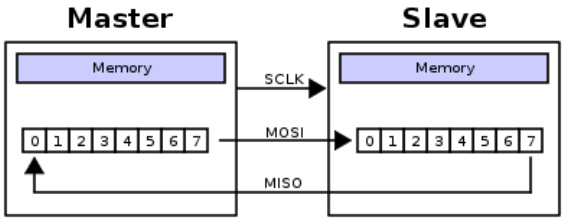

Figure 3: Block Diagram for Data Transmission

Transmission normally involves two shift registers of some given word size such as 8bit, one in the master and one in the slave. They are connected in a virtual ring topology. Data is usually shifted out with the most significant bit first, while shifting a new least significant bit into the same register. After the registers have been shifted out, the master and slave have exchanged register values.

If more data needs to be exchanged, the shift registers are reloaded and the process repeats. Transmission may continue for any number of clock cycles. When complete, the master stops toggling the clock signal, and typically deselects the slave.

Using ISE Design Suite 14.6 the master slave has been designed and simulated

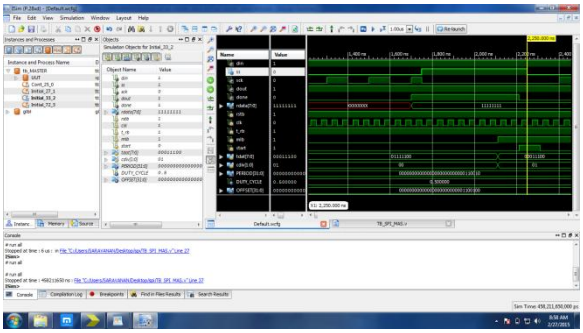

Figure 4: Simulation Result for Master

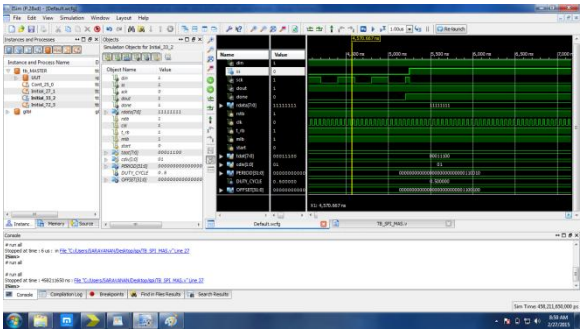

Figure 5: Simulation Result for Slave

V. DESIGN OF MICROBLAZE USING EMBEDDED DEVELOPMENT KIT(EDK)

The MicroBlaze soft core processor is highly configurable, which allows to select a specific set of features required for the application firmware.

MicroBlaze instructions are 32 bits and are defined as either Type A or Type B. Type A instructions have up to two source register operands and one destination register operand. Type B instructions have one source register and a 16-bit immediate operand (which can be extended to 32

bits by preceding the Type B instruction). Type B instructions have a single destination register operand.

MicroBlaze is implemented with a Harvard memory architecture; instruction and data accesses are done in separate address spaces. Each address space has a 32-bit range (that is, handles up to 4-GB of instructions and data memory respectively). The instruction and data memory ranges can be made to overlap by mapping them both to the same physical memory. The MicroBlaze instruction and data caches can be configured to use 4 or 8 word cache lines. When using a longer cache line, more bytes are prefetched, which generally improves

performance for software with sequential access patterns. However, for software with a more random access pattern the performance can instead decrease for a given cache size. This is caused by a reduced cache hit.

Many aspects of the MicroBlaze can be user configured: cache size, pipeline depth (3-stage or 5-stage), embedded peripherals, memory management unit, and bus-interfaces can be customized. The area-optimized version of MicroBlaze, which uses a 3-stage pipeline, sacrifices clock frequency for reduced logic area. The performance-optimized version expands the execution pipeline to 5 stages, allowing top speeds of 210 MHz.

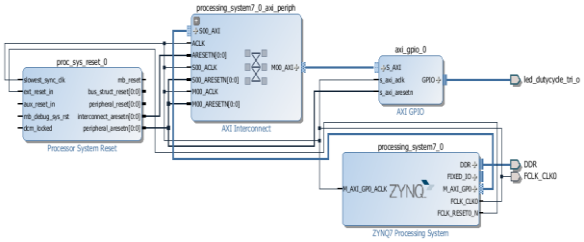

Figure: 6 Vivado Block Design Diagram

By using XPS (Xilinx Platform Studio) the microprocessor's hardware specification and configuration is achieved. With the help of XPS the designer's platform specification is converted into a synthesizeable RTL description and a set of coding (Verilog or VHDL), is written to automate the implementation of the application firmware(from RTL to the bitstream-file.)

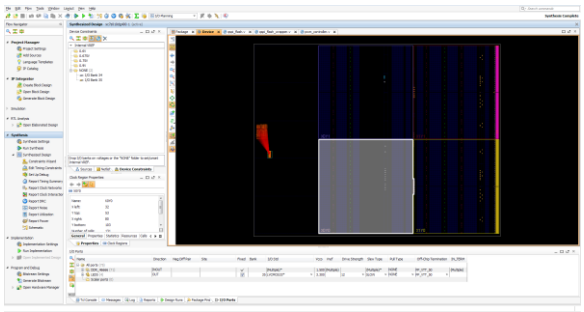

Figure 7: Synthesis report

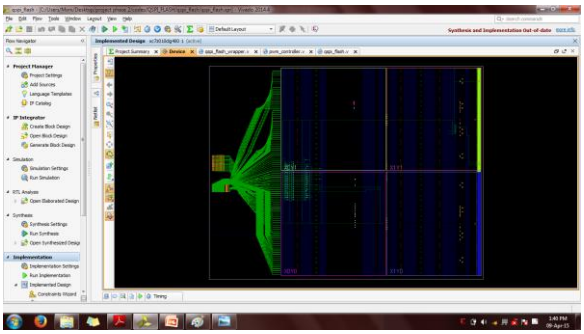

Figure: 8 Implementation and routing report

In Vivado after synthesis and implementation the timing report, synthesis, implementation and routing reports are taken.

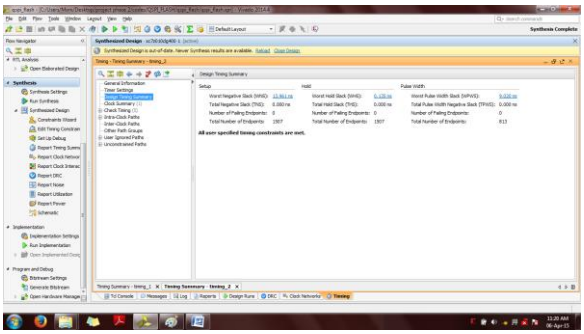

Figure: 9 Design Timing Report

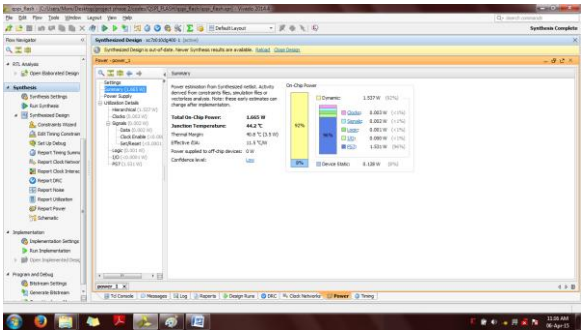

Figure: 10 Power Report

Using the SDK(Software Development Kit) the generated bitfile of the application is dumped into the Zynq FPGA and made to read the application. The example application used here is to change the intensity of the LED with four bits which will have sixteen level of changes.

VI.CONCLUSION

The SPI flash is a low-pin count and simple solution for configuring zynq FPGAs. Support of indirect programming enhances ease of use by allowing in-system programming updates of the SPI flash by reusing connections already required for the configuration solution. A unique FPGA configuration that reduces hardware and application firmware requirements of a typical system that contains a microprocessor, flash memory and FPGA has been

designed. It leverages the compatibility between the FPGA serial configuration mode and SPI memory. Upto flash memory capacity the number of applications can be read.

ACKNOWLEDGMENT

I authors would like to think Sri Ramakrishna Engineering College for providing us full support

REFERENCES

[1]. Using a Microprocessor to Configure 7 Series FPGAs via Slave Serial or Slave SelectMAP Mode(XAPP583)

[2]. 7 Series FPGAs Configuration User Guide (UG470)

[3]. Using SPI Flash with 7 Series FPGAs (XAPP586)

[4]. Introduction to FPGA Design with Vivado HLS(UG998)

[5]. MicroBlaze Processor Reference Guide (UG081)

[6]. Vivado® Design Suite User Guide: High-Level Synthesis (UG902)

[7]. AXI Reference Guide (UG761)

[8]. 7 Series FPGAs Configuration User Guide(UG470)

[9]. Vivado Design Suite Video Tutorials (www.xilinx.com/training/vivado/index.htm)

[10]. 7 Series FPGAs Overview(DS180)