

# Coprocessor Design using FPGA

Shivashankar Tonape

Department of Electronics & Telecommunication Engineering  
Nagesh Karajagi Orchid College of Engineering & Tech.,  
Solapur, Maharashtra, India.

Shradha Joshi

Department of Electronics & Telecommunication Engineering  
Nagesh Karajagi Orchid College of Engineering & Tech.,  
Solapur, Maharashtra, India.

---

**Abstract-** *The use of the microprocessor, as a tool, has been widely used for the cost reduction and the technological innovations. The programmability of the microprocessor and its speed are the main performance characteristics of the microprocessor. Therefore, to design a microprocessor to be competitive, its processor (consists an advantages) requires the characteristics, such as relatively inexpensive, flexible, adaptable, fast & reconfigurable. A solution to this is the use of the FPGAs (Field Programmable Gate Array) as a design tool. The importance for the FPGA based embedded applications increases very greatly till today. By using FPGA, the end user can achieve the benefits of the system-on-chip (SoC) by configuring the processor as per his requirement, by implementing the required logic on FPGA fabric core. With a processor core, SoC system requires an FPGA. 'Hard' processor core is a hybrid approach, which is added to the FPGA, which offers a performance trade-offs between FPGA and the traditional ASIC. 'Soft' core uses the programmable logic element which can be existed in the FPGA for implementing the processor logic. The Coprocessor has been specifically designed to be small to reduce the power consumption and extend the battery operations. In this paper, a subset of the MIPS Instruction will be implemented, to use for different applications. A 32-bit instruction of selected instruction set with a single cycle data path and the random logic based instruction decoder has been implemented. The main block of the data path would be the ALU, register file, program counter updating logic & controller. A memory block will be implemented to store the hex codes of a program which will be used to test the implemented coprocessor. For synthesis & simulation, Xilinx ISE 9.2i software is used.*

**Keywords:** ARM, FPGA, SPI, SoC, Microprocessor, Data path.

---

## I. INTRODUCTION

Nowadays, the hardware system required for the embedded microprocessors, need to be develop with a lower power, multi-tasking and fast performance for better communication services. So, most of the embedded system developers and designers are used a microprocessor based methodology. The embedded microprocessors generally consist of thousands of electronic components and use a various machine instructions to perform not only the mathematical operations but also to move the information from one memory location to another memory location [3].

Till now, the ARM processor has been playing a major role in the embedded system, which is widely used in a variety of the electronic products such as mobile phones, robots, personal computers due to its power saving features. The low power has become an important consideration for performance and area [1][9]. RISC uses a minimal set of instructions, emphasizing the instructions used most often and optimizing them for the fastest possible execution. This processor implementation will follows the RISC Instruction set architecture because it supports a predefined set of instructions. RISC processors are less costly for designing, testing and manufacturing. Because the RISC processors have advantages in many applications that benefits from the faster instruction executions[4].

Also due to the exponential increase of the technologies; there are many problems, which are faced by designers for requirement of the fast, flexible and many re-programmable devices. So, there is one option to design using an FPGA because of their various advantages such as flexible, real-time in-circuit re-configurability, programmable and reliable [11].

## II. COPROCESSOR DESIGN PHILOSOPHY

The ARM is one of the most licensed and thus widespread processor cores in the world. The ARM Processor is part of the Advanced RISC Machines (ARM) family of general purpose 32-bit microprocessors, which offer very low power consumption and price for high performance devices. The architecture is based on Reduced Instruction Set Computer (RISC) principles, and the instruction set and related decode mechanism are much simpler in comparison with micro-programmed Complex Instruction Set Computers (CISC). This results in a high instruction throughput and impressive real-time interrupt response from a small and cost-effective chip. The ARM Processor core is generally used in different applications or customer-specific integrated circuits (ASICs or CSICs). It's simple, elegant and fully static design is particularly suitable for cost and power-sensitive applications. The ARM's small die size makes it ideal for integrating into a larger custom chip that could also contain RAM, ROM, logic, DSP and other cells [1]. But due to the cost of ASIC design and the speed of the DSPs processors which is involved in the development of the flexible, faster devices, many designers are now turning to FPGA based designs. Today ARM processor cores are gaining importance for FPGA based embedded applications. By using FPGA, the end user can achieve the benefits of the system-on-chip (SoC) by configuring the processor as per his requirement, by implementing the required logic on FPGA fabric core. With a processor core, SoC system requires an FPGA. Processor cores are classified as either "hard" or "soft" [2].

### III. PROPOSED SYSTEM

A coprocessor is designed using an RISC processor which used the pipelined architecture [4]. In order to design a coprocessor, first examine the sequence of operations during execution of instructions, and then describe the nature of the hardware required to accomplish the instruction execution. In general, any microprocessor or coprocessor works in the following 4-stages:

1. Fetch
2. Decode
3. Execute
4. Memory Read/ Write Back.

Hence, the design must contain a unit to fetch the instructions, a unit to decode the instructions, the arithmetic and logic unit (ALU) to execute the instruction, a register file to hold the operands, and the memory that stores instructions and data [6].

The proposed architecture is a general-purpose RISC processor with pipelining feature. It gets instructions from the dedicated buses to its memory regularly, then executes all its native instructions in stages with pipelining. There are basically three types of instruction formats namely Arithmetic and Logical instructions, Load/Store instructions and Branch instructions[7].

#### 1. ALU Instructions:

Arithmetic operations can either take two registers as operands or take one register and a sign extended immediate value as an operand. And the result is stored in the third register. Logical operations do not usually differentiate between the 32-bit or 64-bit. Some of the ALU instructions are ADD, SUB, MUL, AND, OR, NAND, NOR, XOR, NOT, etc.

#### 2. Load/Store Instructions:

Load/Store instructions usually take a register as an operand and a 16-bit immediate value. The sum of the two will create the effective address. A second register acts as a source in the case of a load operation. In the store operation the second register contains the data to be stored. Some instructions are LW, SW, etc.

#### 3. Branch Instructions:

Branches and Jumps Conditional branches are transfer of control. A branch causes an immediate value to be added to the current program counter. Some common branch instructions are BZ (Branch Zero), BRZ (Branch Register Zero), JMP (Jump Instruction), JMPZ (Jump when Zero), etc.

### IV. ARCHITECTURE

#### 1) Over view of Design:

It is of great concern to build ARM soft processor cores in the context of FPGA based multiprocessor SoC applications. A subset of MIPS instructions will be implemented to cater for different applications. A selected set of 32 bit instructions will be implemented with a single cycle data path and random logic based instruction decoder. A coprocessor design based on a FPGA consists of several unit blocks. This unit blocks can be classified in two different categories. In this architecture design, the main components are: a control unit, an Arithmetic Logic unit (ALU), and the memory. And the sub-components are: a program counter, an instruction register, a data register, multiplexers (MUX), adders, etc [8][9][10].

- Data path
  - Register File
  - ALU
  - Multiplexer
  - Adder
- Controller
- Memory
- SPI Communication Module

#### 2) Data path:

A data path is a central part of the many central processing units along with control units, which largely regulates interaction between the data path and the data itself, usually stored in registers or main memory. Data path is the heart of the coprocessor. It is a collection of the functional units, such as register file, ALU, mux, program counter updating logic and controller that perform the data processing operations. It is the module which helps to form an instruction set to any function.

The following figure shows the overall data path unit.

- Data path
  - Register File
  - ALU
  - Multiplexer
  - Adder

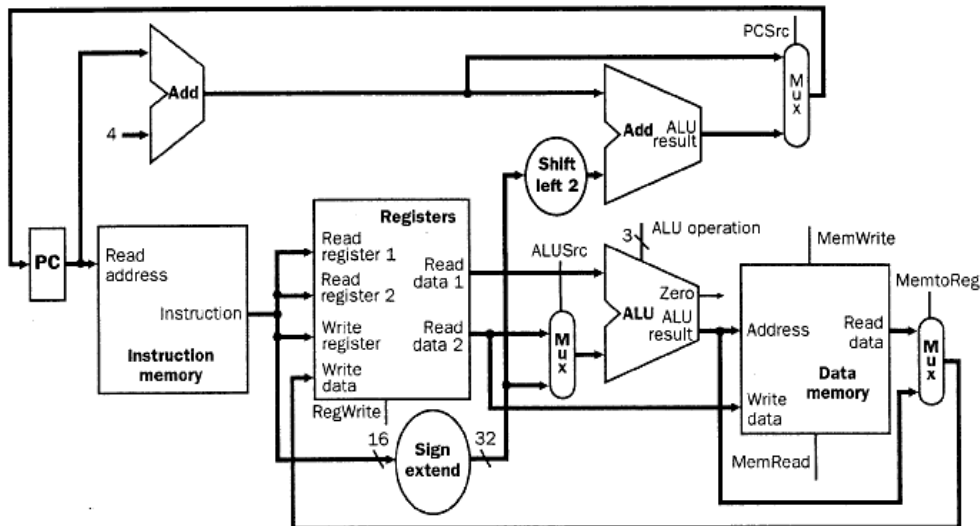


Figure 1: Overall Data path architecture

### 2.1) Register File:

A register file is an array of processor registers in a central processing unit (CPU). Register file is a combination of registers and combinational logic. The register file is the highest level of the memory hierarchy. In a very simple processor, it consists of a single memory location, usually called an accumulator. In a modern processor, it is considered necessary to have at least 32 registers for integer value and often 32 floating-point registers as well. Thus the register file is a small, addressable memory at the top of the memory hierarchy. It is visible to programs, so that the number and type of registers is a part of the instruction set architecture (ISA). A modern processor will have at least 32 integer registers, each capable of storing a word of 32 bits. A processor with floating-point capabilities will generally also provide 32 or more floating-point registers, each capable of holding a double precision floating-point word. These registers are used by programs as temporary storage for values which will be needed for calculations. Because the registers are 'closest' to the processor in terms of access time, able to supply a value within a single clock cycle. Thus the size of the register file is an important factor in the overall speeds of programs.

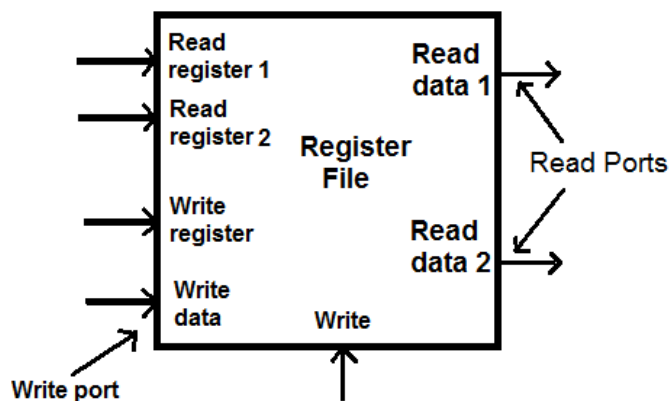


Figure 2: Register File

A register file needs at least 2 read ports: the ALU has two input ports and it may be necessary to supply both of its inputs from the same register.

### 2.2) Arithmetic Logic Unit:

An ALU is a combinational network that implements a function of its input based on either logic or arithmetic operations. ALU's are the heart of all computers as well as most digital hardware systems. An ALU is constructed by using the four hardware building blocks (AND & OR Gates, Inverters and multiplexers). Generally, the MIPS word is 32 bits wide. An n-bit ALU typically has two input words and the one output word, where the high order output bit is actually the carry-out. In addition, there is a carry-in input. Besides data inputs and outputs, an ALU must have control inputs to specify the operations to be performed. One input is a mode selector, which determines the operation is a logic function or arithmetic operations. In addition, there are operation selection input, which determines the particular logic or arithmetic function to be performed.

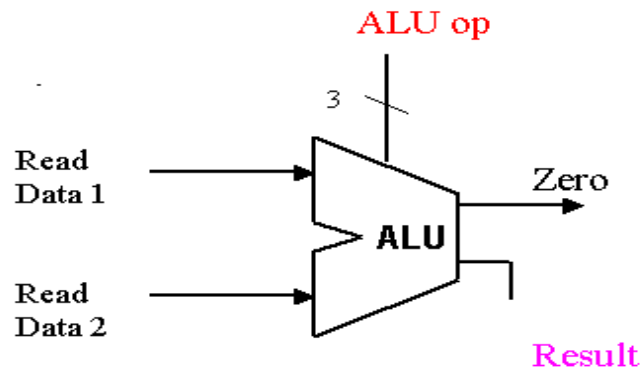


Figure 3: Arithmetic & Logic unit

2.3) Multiplexer:

A multiplexer is a combinational circuit and can be modeled using concurrent statements only or using processes. A multiplexer is a circuit that generates an output that exactly reflects the state of one of a number of data inputs, based on the value of one or more selection control inputs. A multiplexer circuit “multiplexes” the input signals onto a single output. The data input is selected by the values of the select inputs. The select input chooses as the output of the multiplexer either one of the data input.

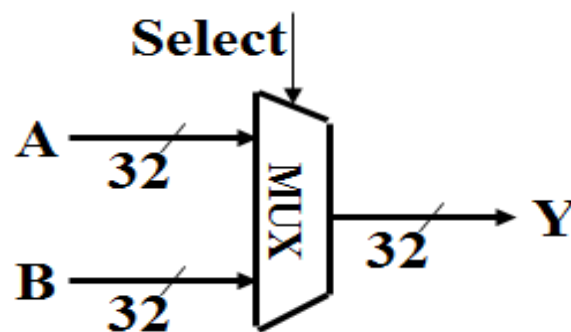


Figure 4: Multiplexer unit

2.4) Adder:

A simple manner to construct an adder is to build a ripple-carry adder. In this adder, 32 copies of a 1 bit full adder are connected in succession to create the 32 bit adder. The carry ripples from the least significant bit to the most significant bit. If gate delays are  $t_g$ , a 1 bit adder delay is  $2t_g$  (assuming a Sum-of-Products expression for Sum and Carry, And ignoring delay for inverters), and a 32 bit ripple carry-adder will take approximately 64 gate delays. This is inadequate for many applications. Hence, designers often resort to faster adders.

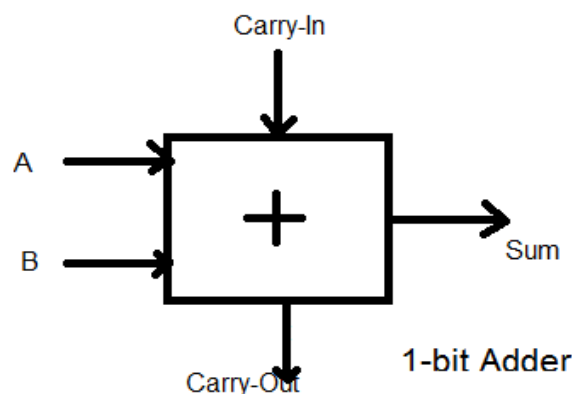


Figure 5: Adder unit

### 3) *Controller of Coprocessor:*

Controller is the hardware that tells the data path what to do, in terms of switching, operation selection, data movement between ALU components, etc. Controller plays an important role in coprocessor design as it controls all the operations performed by the blocks of data path and ROM. The controller in the coprocessor generates the control signals that controls the blocks of the data path for the execution of the instructions given in the ROM. The controller for coprocessor is implemented with random logic method, where the required control signals are generated by pure combinational logic. The instruction set is chosen for implementation can be divided into different types based on the type of control signals need to be issued for various blocks of data path.

### 4) *Memory:*

A memory could be generally used for storing a table of constants to be used as coefficients during processing, or it could be for implementing instruction and data memories for an embedded processor that are designed using FPGA. The memory is typically implemented using a few large blocks of Static RAM (SRAM). Our memory module is 32-bit wide. The instruction memory is implemented as a single port on-chip distributed ROM while the data memory is implemented as a single port on-chip block RAM inside the FPGA. SRAM cells are combined in an array with additional control logic to form a static RAM for a 32X32 SRAM. While in a ROM, a decoder on the address lines selects a specific row of SRAM to be accessed at any time. Once the row is decided, 32-bit data is written in the 32 SRAM or read 32-bit data to next component. The data memory stores ALU results and operands, including instructions, and has two enabling inputs (MemWrite and MemRead) that cannot both be active (have a logical high value) at the same time. The data memory accepts an address and either accepts data (WriteData port if MemWrite is enabled) or outputs data (ReadData port if MemRead is enabled), at the indicated address.

### 5) *Serial Peripheral Interface:*

The SPI is a Full Duplex, Synchronous interface which allows several SPI microcontrollers or SPI-type peripherals to be interconnected. In a serial peripheral interface, separate wires (signals) are required for data and clock. In the SPI format, the clock is not included in the data stream and must be furnished as a separate signal. SPI is based on the Master-Slave protocol where master is the device that drives the clock signal. SPI is a Bi-Directional device means that the data can be transmitted in both directions at the same time in serially (only one bit at a time) manner. The SPI is most often used in the systems for communication purpose between the central processing unit (CPU) and the peripheral devices.

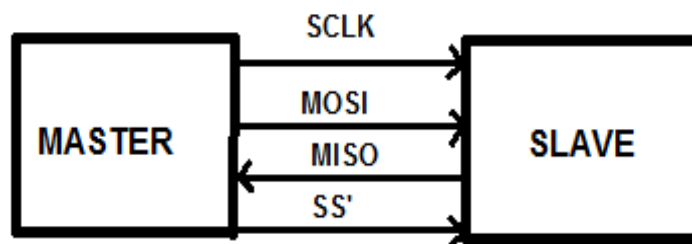


Figure 6: SPI Communication Module

The SPI generally has the four basic signals: MISO, MOSI, SCK, SS'. In the master SPI, the output bits are sent from the MOSI pin and these output bits are received by the MISO pin at Slave SPI. This shifted data are stored in the SPI data register and then send the most significant bit (bit 7) first. When the bit 7 of the master is shifted through the MOSI pin, a bit from the bit 7 of the slave is being shifted into the bit 0 of the master via the MISO pin. This bit will eventually end up in bit 7 of the master after the 8 clock pulses. An SPI transmission is always initiated by the master, and the peripheral device is called the slave.

## V. IMPLEMENTATION & RESULTS

To ensure the performance and the quality of our design, all of the units are designed and tested separately. Xilinx ISE 9.2i is used for simulation of the FPGA based code and results have been verified. Once the functionality of each of the units is verified, then they are combined together as a block and once again tested.

### 1. *Arithmetic & logic unit:*

In this waveform, the various ALU functions are observed. In ALU, according to the ALU OP, many kinds of operations of the arithmetic and logical functions can be performed.

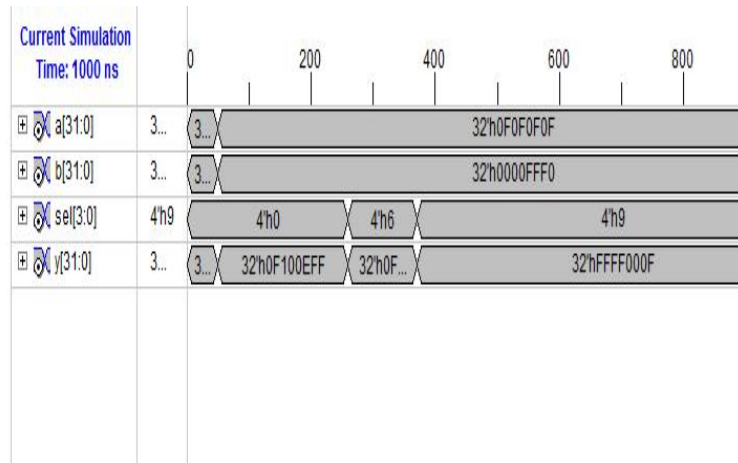


Figure 6: Simulation result of ALU

2. Multiplexer:

The various MUX functions are performed in this waveform. The Mux output is dependent upon the mux's select input which can be either '0' or '1'.

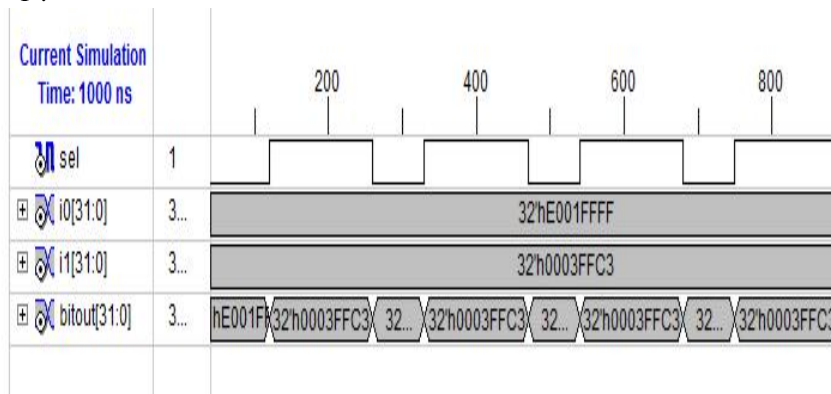


Figure 7: Simulation Result of Multiplexer

3. Data path & Controller:

The data path & controller output waveform is shown below. This output is coming according to the instructions given by the Controller & ROM.

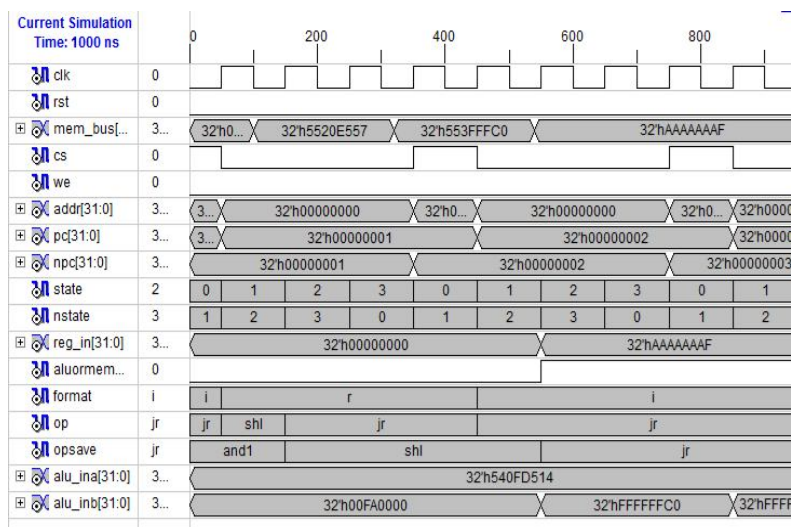


Figure 8: Simulation result of Data path & Controller



4. Processor:

Following waveform shows that the complete processor performance. This processor performance is based upon the fetch, decode & execution of an instructions.

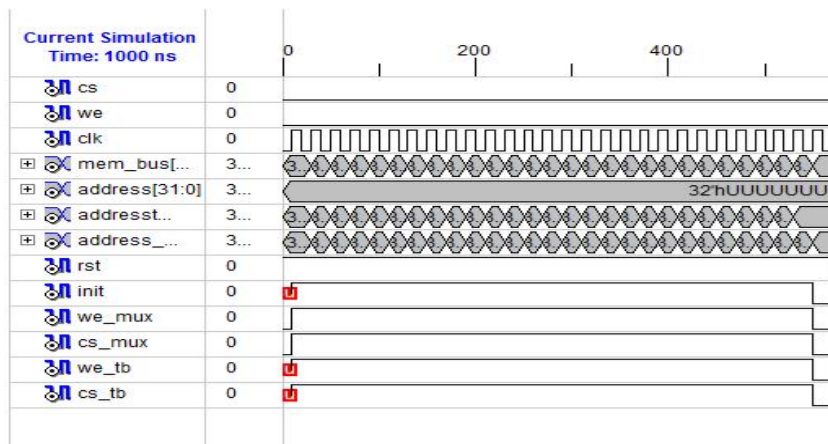


Figure 9: Simulation Result of Processor

5. Serial Peripheral Interface:

The Master-Slave configuration simulation result is shown in below waveform.

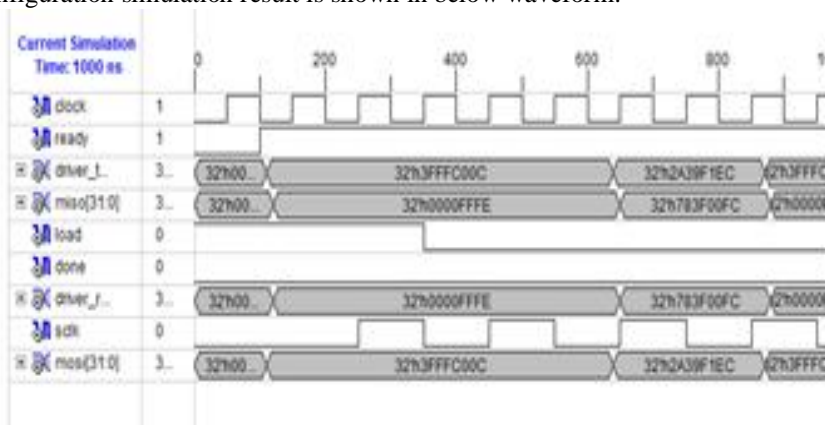


Figure 10: Simulation Result of SPI

**VI. CONCLUSION & FUTURE SCOPE**

A simple implementation of a subset of the 32-bit MIPS has been done in this paper. This subset includes most of the important instructions, including ALU, memory access, and branch instructions. For the sake of simplicity, the proposed architecture is designed for preventing the pipelining structure from the branch instructions. This architecture emphasizes simplicity and excludes instructions that could take longer than the most common instructions. In this paper, the coprocessor is designed by using an FPGA. So, the required instruction subset for this coprocessor is implemented by writing the codes in the VHDL language. The design contains a single cycle pipelining architecture. This can be further implemented with multi-cycle instruction set and also with better hardware design for faster speeds and high performance. The coprocessor can be used for various different applications such as medical applications, industry application, etc.

**ACKNOWLEDGMENT**

This paper work is undertaken and supported by the NK's Orchid College of Engineering & Technology, Solapur. We are very thankful of the our friends, supporters & the Department of Electronics & Telecommunication, NKOCET, Solapur University, Solapur for their constant guidance, support & encouragement in undertaking the work very prosperly.

**REFERENCES**

- [1] Alex Heunhe Han, Young-Si Hwang, Young-Ho An, So-Jin Lee, Ki-Seok Chung "Virtual ARM Platform for Embedded System Developers, IEEE 2008, PP 586-592.
- [2] E. Ayeh, K. Agbedanu, Y. Morita, O. Adamo, P. Guturu "FPGA implementation of an 8 bit Simple Processor" IEEE 978-1, 2008.
- [3] J. O. Hamblen, T. S. Hall "Using System-on-a Programmable-Chip Technology to Design Embedded Systems" IJCA, Vol. 13, No. 3, Sept. 2006.pp 1-11.



- [4] Geun-young Jeong “*Design of 32-bit RISC processor and efficient verification*” Proceedings of the 7th Korea-Russia International Symposium. KORUS2003, PP 222-227.
- [5] Shebli Anvar, Olivier Gachelin, et al. “*FPGA-based System-on-Chip Designs for Real-Time Applications in Particle Physics*” 14th IEEE Real Time Conference, Stockholm, Sweden, June 6-10, 2005 pp 1-5.
- [6] Kane , Gerry, “*MIPS RISC Architecture*”, Upper Saddle River, N. J.: Prentice Hall, 1989.
- [7] Patterson, David A., Hennessey, John L. “*Computer Organization and Design: The Hardware Software Interface*”, 3<sup>rd</sup> ed., San Francisco, Calif.: Morgan Kaufmann, an imprint of Elsevier, 2005.
- [8] Stephen Brown and Zvonko Vranesic, “*Fundamentals of Digital Logic with VHDL Design*”, 2<sup>nd</sup> edition, New York: McGraw-Hill, 2005.
- [9] ARM Ltd, ARM7TDMI Data Sheet (ARM DDI 0029E), Advanced RISC Machines Ltd.
- [10] Maxfield, Clive, “*The Design Warrior's Guide to FPGAs: Devices, Tools and Flows*”, Elsevier. p. 4, ISBN 978-0-7506-7604-5.
- [11] Nass, Rich, *EETimes*, "Xilinx puts ARM core into its FPGAs", April 27, 2010.