Journal of Innovative Engineering and Research (JIER)
Vol.- 1,Issue - 1, April 2018, pp. 28-31 (4 pages)

Design and Implementation of 16 Bit RISC Processor in VHDL Using Xilinx Tools

SHEETESH SAD[1], SANTOSH PAWAR[2]

[1,2]*Department of Electronics & Communication Engineering, Dr. A. P. J. Abdul Kalam University, Indore*

Corresponding Author Email: sheeteshsad@gmail.com

Abstract— The scope of my work includes study of VHDL language and algorithms for computing arithmetic and logical operation with their function suited for hardware implementation. These algorithms are used as functional blocks in RISC processor. The algorithms are coded in VHDL and validated through extensive simulation.ISE Xilinx simulator is used for the simulation of all the models. Experimental results show feasibility of modeling strategy and provide performance measures of 16 Bit RISC Processor design features This VHDL code is then synthesized by Leonardo Spectrum tool to generate the gate level net list that can be implemented on the FPGA. This is general purpose Processor use on Application or Customer Specific Integrated Circuited (ASIC or CSIC)

Index Terms— RISC, CISC, HDL, CPU, RTL, ASIC.

I. INTRODUCTION

A microprocessor is a multipurpose, programmable, clock driven, register based electronic device that reads binary instruction from a storage device called memory, accepts binary data as input and processes data according to those instructions, and provides results as output. A typical programmable machine can be represented with three components: microprocessor memory and I/O (input/output). The microprocessor operates in binary digits, 0 and 1, also known as bits. Each microprocessor recognizes and processes a group of bits called the word, and microprocessor are classified according to their word length. For example, a processor with an 8 bit word is known as an 8 bit microprocessor and a processor with a 16 bit word is known as a 16 bit microprocessor

1.1 RISC Architecture

The concept of a RISC Processor is based upon the idea that a small, basic instruction set in conjunction with a "smart" compiler can deliver superior performance over a Complex Instruction Set Computer (CISC) with a large number of specialized instructions. The simplicity of the operations performed ideally allows every instruction to be completed in one processor cycle. The basic data path of a RISC processor is shown below in Figure 1.

The Instruction Decoder loads the instruction pointed to by the program counter (PC) from processor memory. The Instruction Decoder then generates the appropriate control signals for the Execute unit, which performs the desired function (arithmetic, logic, etc.) on the data. The Write-back unit then updates the memory with any new values.

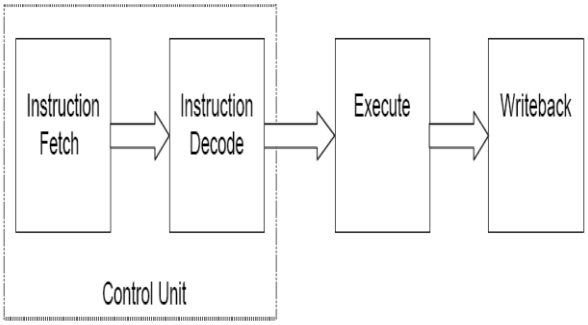

Figure 1: RISC Processor Data-path

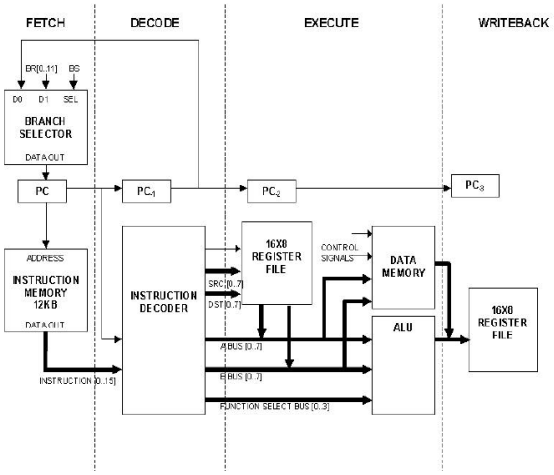

Figure 2: RISC processor data path diagram

A more detailed look at the layout of the RISC processor is shown below in Figure 2. The process starts out at the branch selector, which loads the program counter with either the next sequential address or the address of a program branch depending on the value of the branch select signal. In the case of an interrupt, the branch address input would contain the address of the appropriate interrupt handler. The instruction is then fetched from program memory and sent into the instruction decoder, which loads the operand and functions select buses and generates control signals for the rest of the processor. The execute stage performs an operation or interacts with the data memory. After the execute stage the result is written to the processor registers if applicable.

Journal of Innovative Engineering and Research (JIER)
Vol.- 1,Issue - 1, April 2018, pp. 28-31 (4 pages)

1.2 RISC v/s CISC

What is CISC.
i.Complex Instruction Set Computer.
"High level" Instruction Set.
ii.Executes several "low level operations".
iii..Ex: load, arithmetic operation, memory store.

Features of CISC

- Instructions can operate directly on memory.
- Small number of general purpose registers.
- Instructions take multiple clocks to execute.
- Few lines of code per operation.

What is RISC

- Reduced Instruction Set Computer.
- RISC is a CPU design that recognizes only a limited number of instructions.
  - ♦ Simple instructions.
  - ♦ Instructions are executed quickly.

Features of RISC

- "Reduced" instruction set
- Executes a series of simple instruction instead of a complex instruction
- Instructions are executed within one clock cycle
- Incorporates a large number of general registers for arithmetic operations to avoid storing variables on a stack in memory
- Only the load and store instructions operate directly onto memory
- Pipelining = speed.

II. DESIGN OF 16 BIT RISC PROCESSOR

A microprocessor based system includes three components: microprocessor, I/O (input/output) and memory (read/write memory and read only memory).These components are organized along a common communication path called a bus. The entire group of components is also referred to as a system or a microcomputer system and the component themselves are referred to as sub system. The microprocessor is one component of the microcomputer.
Microprocessor:-The microprocessor can be divided into three segments for the sake of clarity.
Arithmetic/Logic Unit:-this is the area of the microprocessor where various computing function are performed on data. The ALU Unit performs such arithmetic operation as addition and subtraction and such logic operation as AND, OR etc.
Register Array: - This area of the microprocessor consists of various registers. These registers are primarily used to store data temporarily during the execution of a program and are accessible to the user through instruction.
Control Unit: - The control unit provides the necessary timing and control signals to all the operations in the microcomputer. It controls the data flow of data between the microprocessor and memory and peripherals.
2.1 Architecture
Top Level Design of 16 bit processor

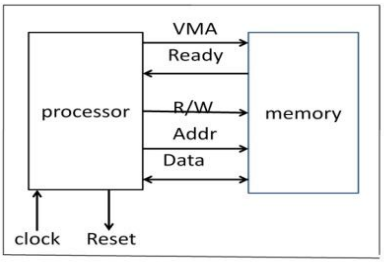

Figure 3: Top Level Design of 16 bit processor

The top-level design consists of the processor block and a memory block communicating through a bidirectional data bus, an address bus, and a few control lines. The processor fetches instructions from the external memory and executes these instructions to run a program. These instructions a restored in the instruction register and decoded by the control unit. The control unit causes the appropriate signal interactions to make the processor unit execute the instruction.
If the instruction is an add of two registers, the control unit would cause the first register value to be written to register OpReg for temporary storage. The second register value would then be placed on the data bus. The ALU would be placed in add mode and the result would be stored in register OutReg. Register OutReg would store the resulting value until it is copied to the final destination.

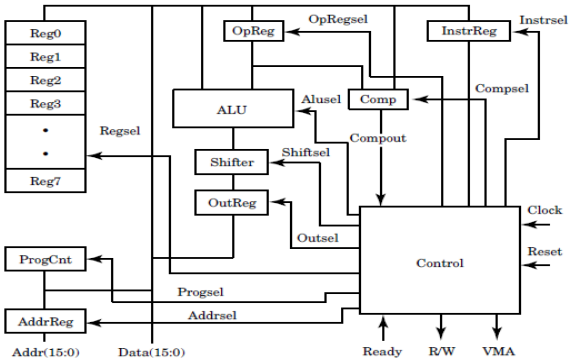

Figure 4: Internal block diagram of processor

III. SIMULATION

Simulating the processor design is different from most other entities because the processor design doesn't need much outside stimulus. the memory device provides the input data for the processor much as a stimulus file would for other entities. the processor reads its program from the memory device. the processor need only have the **clk** signal and **reset** signal stimulated properly, and the processor reads and executes instructions from that point forward.
the only stimulus needed to start the operation of the processor is a uniform signal applied to the **clk** input and a pulse applied to the **reset** input for at least 2 clock cycles. this starts the processor into the reset sequence. after the reset sequence has been started, the processor is initialized and starts executing the processor instructions from the **mem**

entity. The processor is simulated as stimulus only initially to verify that the device seems to be functioning. More complex test benches need to be created that include comparison against a known good result to verify correctness. The simplest method for doing this is to manually verify the results the first time, capture the output results, and then use them for comparison later.

The first step in simulating the processor is to compile all the files that make up the design into a format that the simulator can use. The compiled format is loaded into the simulator, and the simulation is executed. The ModelSim simulator from Model Technology is used for the simulation process.

The first step in compiling all of the files in the design is to create one or more libraries to store the compiled data. The default library to store the compiled data is a library called **work**. The name **work** is the logical name of the library; the physical location of the library can be anywhere.

3.1 Synthesize RTL view

Top module RTL

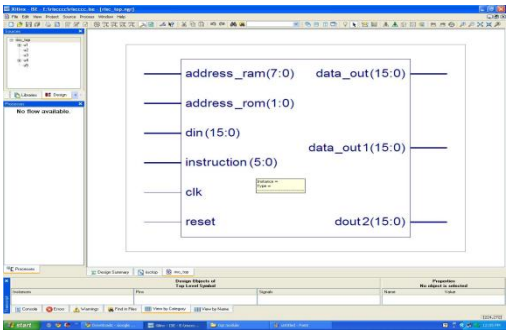

Figure 5: RTL view of 16 Bit RISC processor

ALU RTL

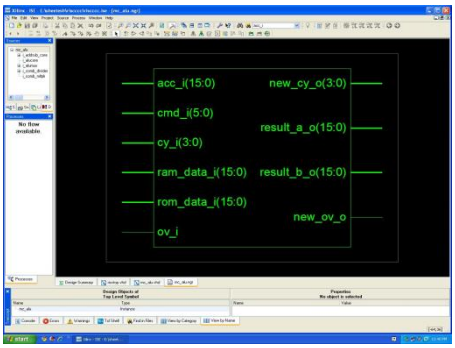

Figure 6: RTL view of ALU

FSM RTL

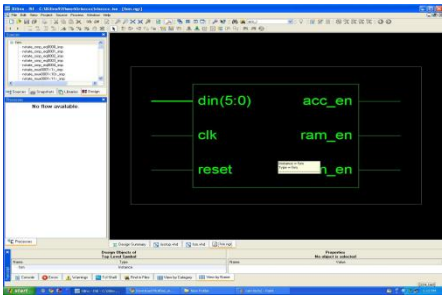

Figure 7: RTL view of FSM (Final State Machine)

RAM RTL

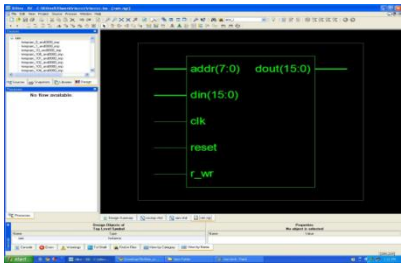

Figure 8: RTL view of RAM

ROM RTL

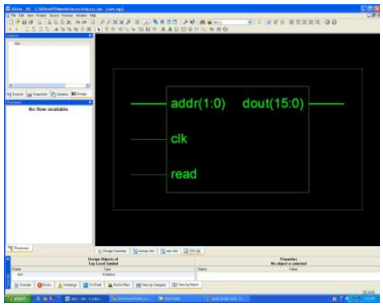

Figure 9: RTL view of ROM

REGISTER RTL

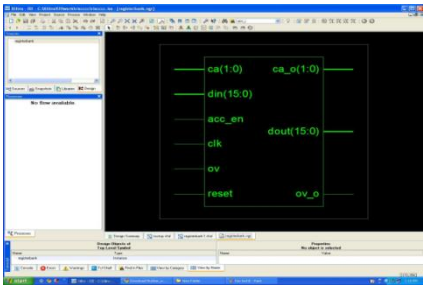

Figure 10: RTL view of Register

3.2 Simulation wave form view

ALU Input Output Simulation wave form

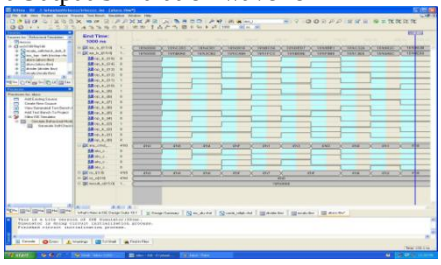

Figure 11: ALU Input Simulation wave form

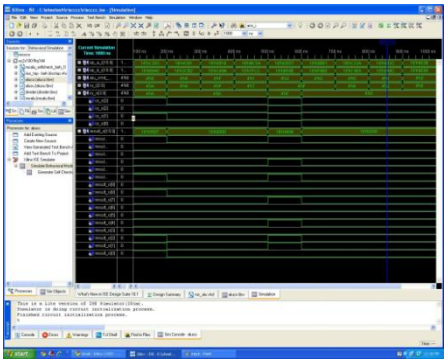

Figure 12: ALU Output Simulation wave form

FSM Input Output Simulation wave form

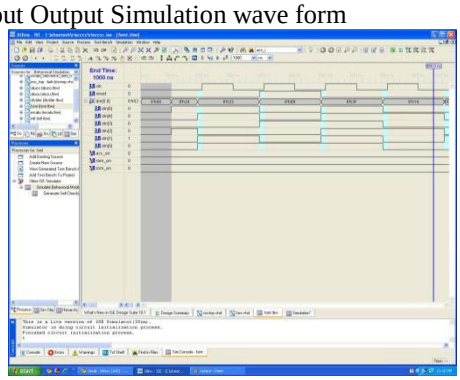

Figure 13: FSM Input Simulation wave form

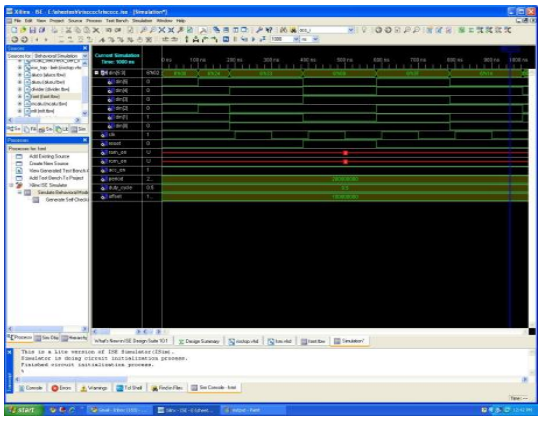

Figure 14: FSM Output Simulation wave form

Register Input Output Simulation wave form

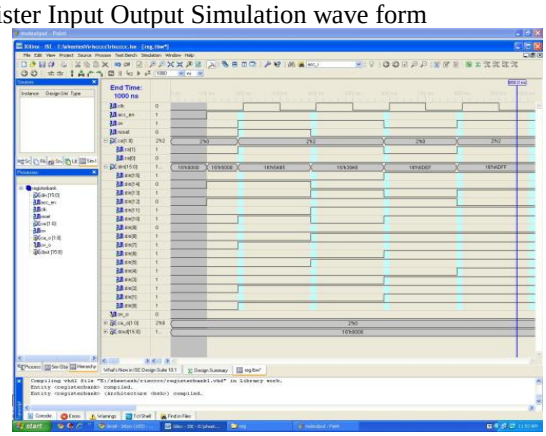

Figure 15: Register Input Simulation wave form

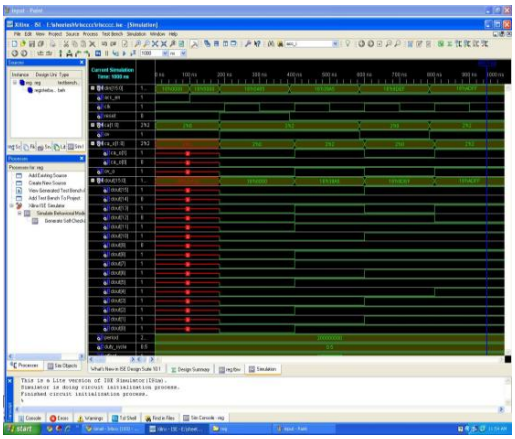

Figure 16: Register Output simulation wave form

VI. CONCLUSION

16 bit RISC processor was designed using top down design methodology. A general purpose RISC processor is design to carry out various arithmetic and logic instructions. The RISC processor is implemented using VHDL language. The baseline architecture of RISC processor is designed including its subsystem blocks. The various blocks are simulated and synthesized using efficient software like Xilinx ISE Simulator. The design is optimized for the area constraint during synthesis. It was then simulated and synthesized successfully by ISE simulator of Xilinx and Implemented on to the SPARTEN 3E FPGA Board

REFERENCES

[1] Douglas L Parry, VHDL Programming By Examples, Tata McGraw-Hill, New Delhi, 2002.

[2] J. Bhasker, VHLD Primer, Pearson Education Asia, 2002

[3] R. Gaonker, Microprocessor Architecture, programming and Application, Penram International Publishing India, 2000.

[4] J. F. Wakerly, Digital design- Principle and practices, Tata McGraw Hill Series.

[5] IEEE RISC Processor MIPI London all members meeting September 21st 2005.

[6] VHDL modelling guide Intel microprocessor data sheet.

[7] R. B. Brown, R. J. Lomax, G. Carichner, and A. J. Drake, "A Microprocessor Design Project in an Introductory VLSI Course".

[8] Y. Zi He , Thesis of Building A RISC Microcontroller in an FPG.

[9] Kang-joo Kim and Koon-shik Cho, "Design & Verification of 16 Bit RISC Processor".

[10] Enoch O. Hwang, Digital Logic and Microprocessor Design with VHDL.

[11] C. H. Roth, Digital system design using VHDL.

[12] D. J. Smith, HDL Chip Design Using VHDL or Verilog, Doone Publication.

[13] H. Woods, Digital Logic Design.

[14] P. S. Mane, " Implementation of RISC Processor on FPGA".

[15] B. L. Di Jasio, Programming 16-Bit PIC Microcontrollers in "C".