IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. IX (Sep-Oct. 2014), PP 77-81
www.iosrjournals.org

Design of Non-pipelined LC3 RISC Microcontroller

Devyani Gera[1], Mehul Garg[2]
[1](Department of Electronics & Communication, Jaypee Institute of Information Technology, India)
[2](Department of Electronics & Communication, Jaypee Institute of Information Technology, India)

Abstract:*This paper highlights the designingof data and control path of a non-pipelined LC3 microcontroller with a 16 bit Reduced instruction set. The designing was done using a Hardware Descriptive Language (HDL) System Verilog on the platform Questasim Simulator. We have demonstrated the use of this RISC microcontroller using basic ALU operations- ADD, AND & NOT, and a memory operation- Load Effective Address (LEA).The 16 bit reduced instruction used in the designing has been chosensuch that it's computing ends in exactly 5 clock cycles.*
Keywords:*HDL, LC3, Load Effective Register, Non-pipelined, Pipelined,RISC*

I. Introduction

The Reduced Instruction Set Computing, or RISC, is a processor design strategy based on small, highly optimized set of instructions instead of more specialized or specific instructions. The RISC strategy is considerably faster than its predecessor – Complex Instruction Set Computing (or CISC) due to the fact that this RISC processors process multiple instructions in one single clock cycle. This increases the overall operating speed of the system.

1.1. Advantages of RISC Processors [1]

1.1.1. RISC processors uses single cycle to process multiple instructions thereby making it faster.
1.1.2. RISC is software oriented, and therefore uses simple instructions to accomplish tasks. This means that the transistor requirement for the RISC processors is lower as compared to its predecessor. This leaves more area on chip for general memory registers, and also reduces production costs.

1.2. Little Computer3

The microcontroller designed in this project is a non-pipelined LC3. LC3 uses 16bit instructions as input. The 16 bit instruction contains information pertaining to sources, operations which are decoded to calculate the appropriate outputs.

1.3. Pipelined vs Non-Pipelined

There two basic approaches through which an LC3 can be designed: pipelined and non-pipelined. Pipelined approach means that the RISC processor simulates multiple process' in one single cycle. The data is processed one after another as in a pipeline. On the other hand, non-pipelined approach waits for the entire address to complete all four stages of the LC3, before the next address is fetched and decoded.

1.4. Advantages of Non-Pipelined[2]

1.4.1. Non-pipelined uses single instructions. This reduces branch delay as well as serial instructions delay.
1.4.2. A non-pipelined processor has a definite instruction set. Its output can be predicted to a certain degree. On the other hand Pipelined processors output varies form program to program.
1.4.3. The output of the Fetch module depends upon the Program counter to be updated in the Writeback. However for pipelined instructions all the modules are executed simultaneously. Therefore, the program might behave incorrectly.

II. Little Computer 3

The LC3 is comprised of 5 basic modules. Their functions and block diagrams are mentioned below.

2.1 Fetch

The fetch module creates the correct Program counter to read values from. When enabled, fetch sends the address of the instruction in the instruction memory (from where the 16 bit instruction is loaded into LC3). This information is sent as a 16 bit address value called as the program counter. The starting address of the program counter is 16'h3000, which is also the value which is set when LC3 is reset. Fetch also sends instrmem_rd as output which instructs the decode block to read instruction from instruction memory.

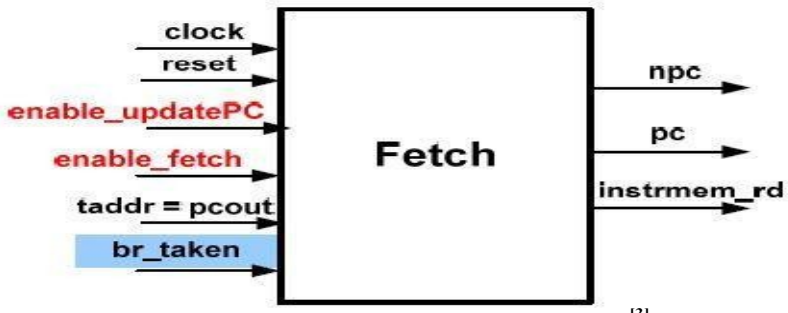


Figure I: Fetch Module Block Diagram [3]

2.2 Decode

The aim of the decode block is to create relevant control signals for a given instruction, i.e. the data sent by Fetch as npc is accepted and the 'decode' is performed on the 16bit value. From the 16bit addresses the decode dissects the address into 3 different outputs, a signal controlling the Execute unit which in turn also controls the typeof operation that is going to be performed, anotherdetermines the right choice between the flowingfor a write to the register file and a signal enabling the selection of the right set of states for the memory based operations.

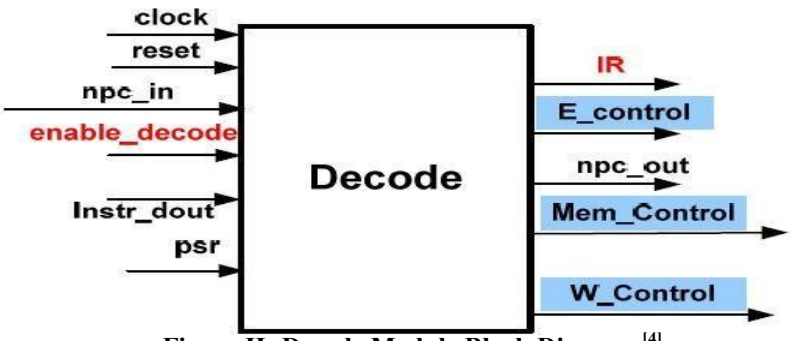


Figure II: Decode Module Block Diagram [4]

2.3 Execute

The execute module forms the soul of the LC3 microcontroller as in this stage the 16bit instruction is computed. This module is closely linked with the Writeback module as it picks up all the data from this module. Also all the memory related operations and control operations need this crucial coupling between the Execute and Writeback modules.

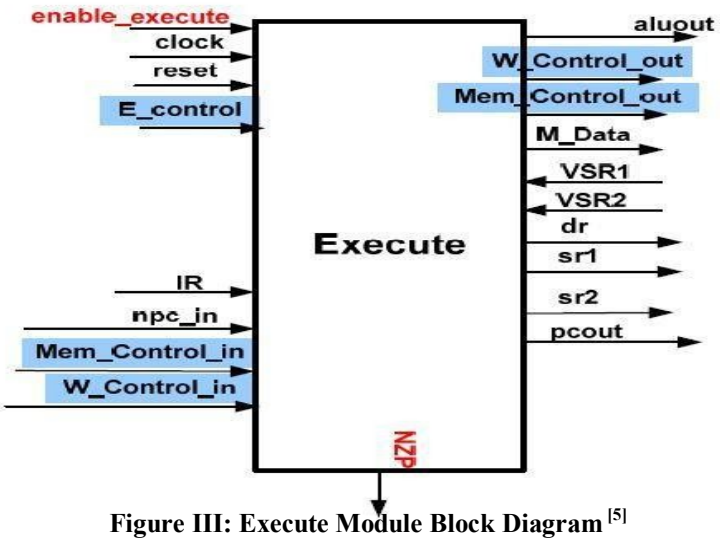
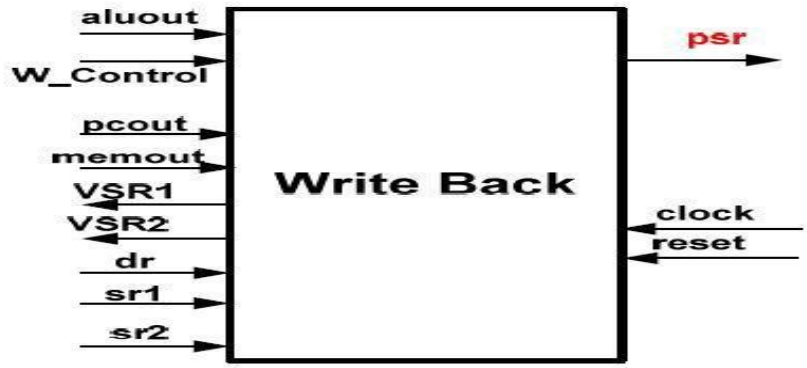


Figure III: Execute Module Block Diagram [5]

2.4 Writeback

This module has the register file that is used for register based operations. Therefore it controls the values that have to be written to the data file. This module is closely linked to the execute module.



2.5 Controller

The controller controls the transition from one stage to another in the computation process. To perform each instruction, this 16 bit instruction undergoes a 4-stage processing system, and when the processing is completed a new instruction is fetched. This can be explained through a simple example of ALU operation; firstly the instruction is fetched, decoded, and then executed. At the end of this cycle, a signal is generated by the controller to fetch the next instruction from the next location in the instruction memory.

III. Operations& Results

Functioning of LC3 was checked by generating all possible inputs randomly within some defined constraints. The set of random values is used to ensure that all possible values of the output are considered. The output of each block as well as the complete controller is shown below. Each figure shows four different operations- AND, OR, NOT& LEA (Load Effective Address)

3.1 Controller Output

Shown below is the output for the RISC controller when run for a finite durations.

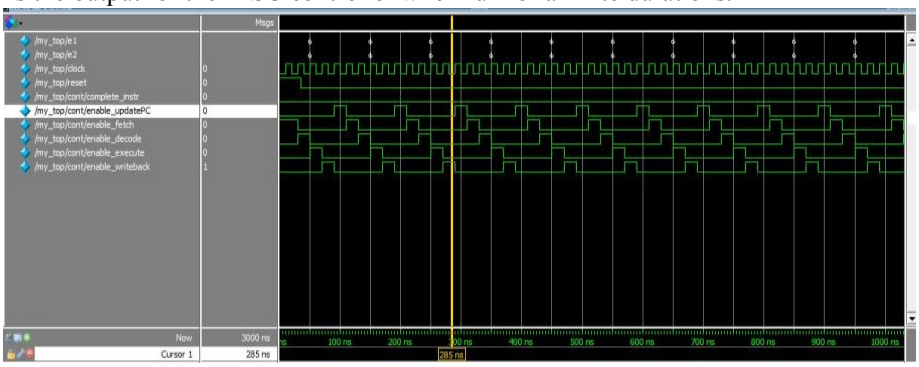


Figure IV: Controller Output

3.2 Fetch

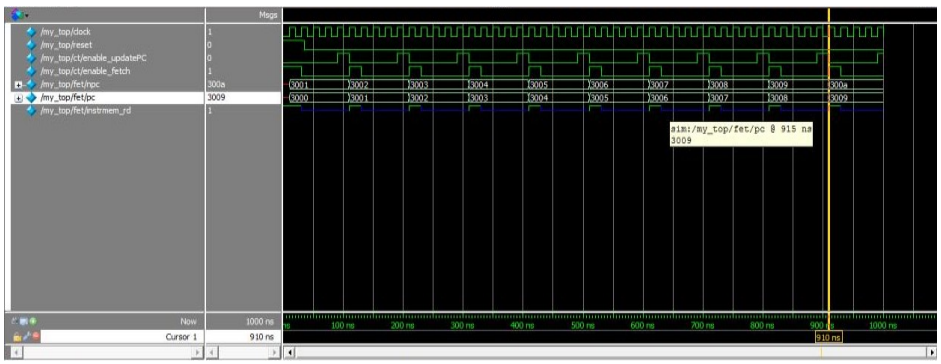


Figure V: Output after Fetch Module

3.3 Decode

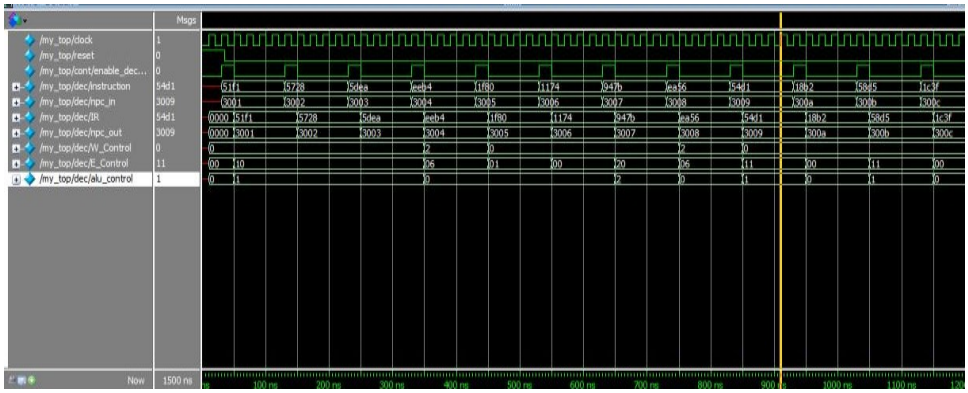


Figure VI: Decode Module Output

3.4 Execute

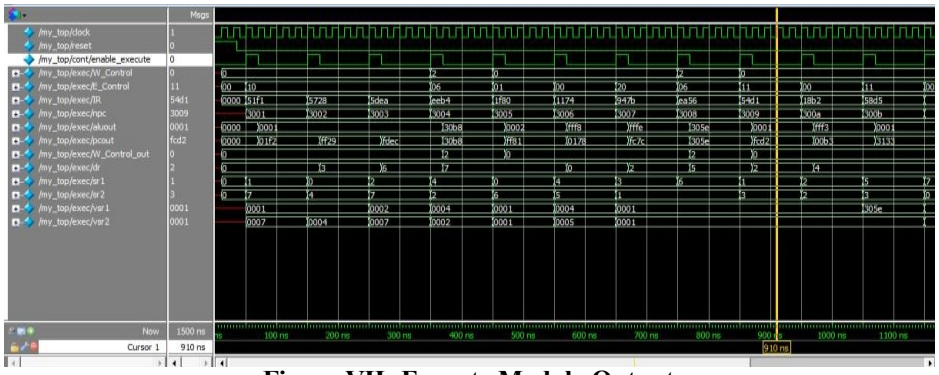


Figure VII: Execute Module Output

3.5 Writeback

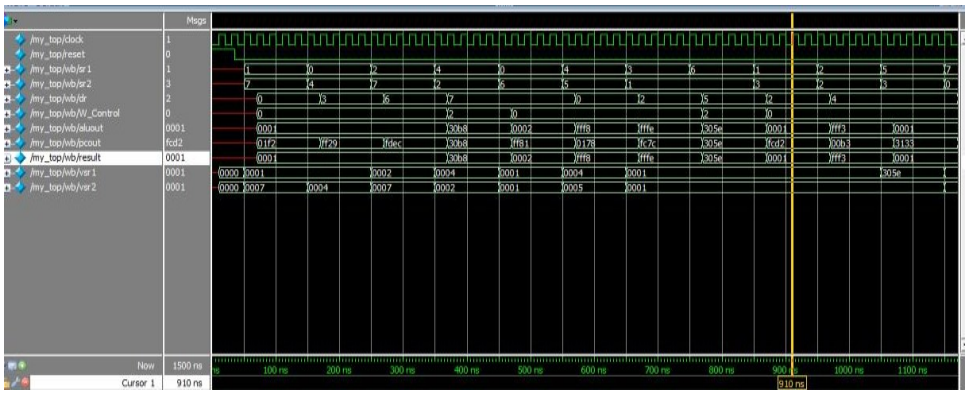


Figure VIII: Writeback module output

3.6 Register File

Register files are used during memory operations to store values. The memory operation demonstrated in this paper is Load Effective Address.

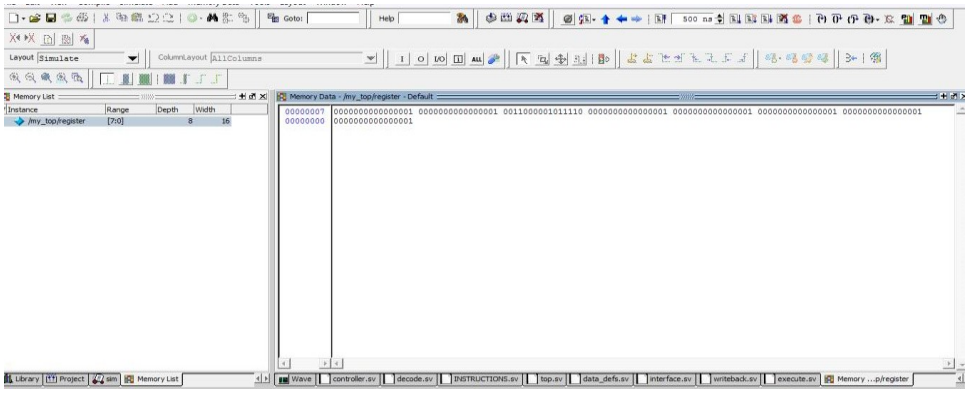


Figure IX: register value for LEA

3.7 Register File with Writeback

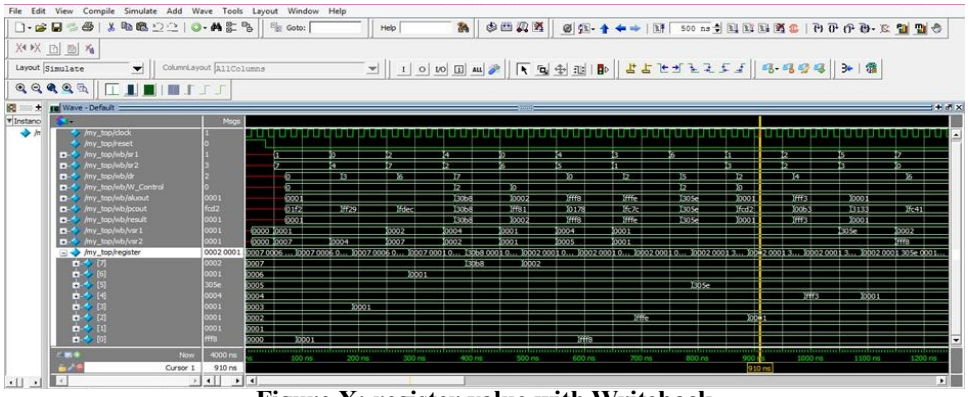


Figure X: register value with Writeback

IV. Conclusion

The here mentioned LC3 controller has been designed using the newer RISC design strategy. This controller has used Non-pipelined methodology for design, and therefore the chance of getting an error due to non-availability of previous values is forestalled. The memory operations performed is Load Effective address.

V. Future Scope Of Work

The LC3 non-pipelined microcontroller can be improved in the following ways:

- Using Pipelining to control the flow of logic. This will result in a much faster operations.
- Shift operations can be designed on this controller
- Memory operations can be implemented on this controller.

References

[1]. Reduced Instruction Set Processing: http://en.wikipedia.org/wiki/Reduced_instruction_set_computing
[2]. Instruction pipelining: http://simple.wikipedia.org/wiki/Instruction_pipelining#Advantages_and_Disadvantages
[3]. NCSU open library : ECE Projects : http://www.ece.ncsu.edu/muse/courses/ece406/labs/proj1/proj1_spr09.pdf
[4]. SystemVerilog for Verification by Chris Spears 2nd Edition.
[5]. Writing TestBenches using SystemVerilog by Janick Bergeron.