IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Design and Implementation of 16-Bit RISC Processor On FPGA

Mr. Rajkumar D. Komati[1], Aditya Kolekar[2], Kunal Kasbekar[3], Ms. Avanti Godbole[4]

[1]*Assistant Professor, Dept. of ENTC, MIT College of Engineering, Kothrud, Pune*

[2, 3, 4] *Undergraduate Student Dept. of ENTC, MIT College of Engineering, Kothrud, Pune*

Abstract - *This project includes the designing of 16-Bit RISC processor and modeling of its components using Verilog HDL. The processor is based on Harvard architecture. The instruction set adopted here is extremely simple that gives an insight into the kind of hardware which should be able to execute the set of instructions properly. Along with sequential and combinational building blocks of a processor such as adders and registers more complex blocks such as ALU and memories have been designed and simulated. The modeling of ALU which has been done in this project is fully structural starting from half adders. At the end the semi-custom layout has been developed for ALU. Complex blocks such as memories have been modeled using behavioral approach, whereas simple blocks such as adders have been done through structural approach. The tools which have been used throughout the project work are ModelSim and Intel Quartus prime 18.1. For synthesis purpose the targeted FPGA device technology is ALTERA.*

Key Words: Verilog HDL, ModelSim, Intel Quartus prime.

1. INTRODUCTION

The Reduced Instruction Set Computer**,** or RISC, is one whose instruction set architecture allows it to have fewer cycles per instruction. Such a computer has a small set of simple and general instructions, rather than a large set of complex and specialized instructions. The most common RISC processors are ARM, DEC-Alpha, PA-RISC, SPARC, MIPS and IBM'S PowerPC.

The main idea is to make hardware simpler by using an instruction set composed of a few basic steps for loading, evaluating and storing operations just like a load command will load data, store command will store the data. The term "reduced" in the phrase is intended to describe

the fact that the amount of work any single instruction accomplishes is reduced—at most a single data memory cycle—compared to the "complex instructions" of CISC CPUs that may require dozens of data memory cycles in order to execute a single instruction. The complexity of controller design has been overcome with the help of operands and opcode bits fixed in instruction register.

Pipelining added a new dimension in the speed just with the help of some additional registers. Now what pipeline does is it increases throughput by reducing cycles per instruction (CPI). The instruction can be executed effectively in one clock cycle. The pipelining in any kind of architecture took birth from the inherent parallelism and the idle state of components. For any given level of general performance, a RISC chip will typically have far fewer transistors dedicated to the core logic which originally allowed designers to increase the size of the register set and increase internal parallelism.

2. LITERATURE SURVEY

Low power pipelined 8-bit RISC processor design and implementation on FPGA

Presented in the 2015 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT) by Jikku Jeemon.

The paper available on IEEE Xplore, presents a low power and simple implementation of an 8-bit RISC processor design on a Spartan-6 SP605 Evaluation Platform FPGA using Verilog HDL. The processor is designed using Harvard architecture, having separate instruction and data memory. It has an instruction set of only 29 instructions. The proposed processor is physically verified on Xilinx Spartan-6 SP605 Evaluation Platform.

Design of 16-bit RISC Processor

Published in IJERT Vol. 2 Issue 7, July 2013 by Supraj Gaonkar and Anitha M.

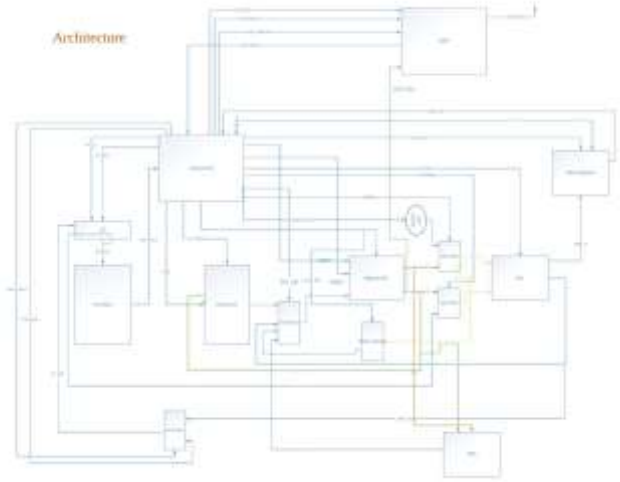
In this paper, the behavioral design and functional characteristics of 16-bit RISC processor is proposed, which utilizes minimum functional units without compromising in performance. The design is based on Harvard architecture. All modules in the design are coded in Verilog. The individual modules are designed and tested at each level of implementation and finally integrated in a top-level module. The tools used are Xilinx ISE 10.1 and ModelSim 10.2.

Design and Analysis of 16-bit RISC Processor

Presented in the 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) by Shraddha M. Bhagat and Sheetal U. Bhandari

This paper is available on IEEE Xplore and the design of a 16-bit RISC processor is explained using Verilog HDL. It comprises of various blocks such as ALU, Controller, register files and data memory unit. It is based on the simple Von Neumann architecture which has a single shared memory for instructions and data. It consists of 15 instructions. The processor is analyzed using Xilinx ISE tool.

3. ARCHITECTURE



The above block schematic shows the overall architecture of the processor. The designed RISC processor has 4-stage pipelining with a 16-bit bus. The blocks present in the architecture are: ALU, Control Unit, MAC Unit, Instruction Memory, Data Memory, Program Counter, Status Register, Register File and various MUX units.

3.1 DATAPATH

Data Path refers to the collection of various units in the CPU like ALU, control unit, various registers, multipliers and buses. The Verilog HDL code for the datapath includes interconnection of the various blocks present in the architecture.

3.2 ALU

The arithmetic logical unit (ALU) we have designed is a very simple one. Its functions include basic arithmetic operations, bit shifting operations and logical operations.

Instruction	OPCODE	Operands
M-type		
LD	000_000	2
ST	000_001	2

INST	Opcode	Operands
R-Type		
MOV	001_000	2
ADD	001_001	2
SUB	001_010	2
XOR	001_011	2
AND	001_100	2

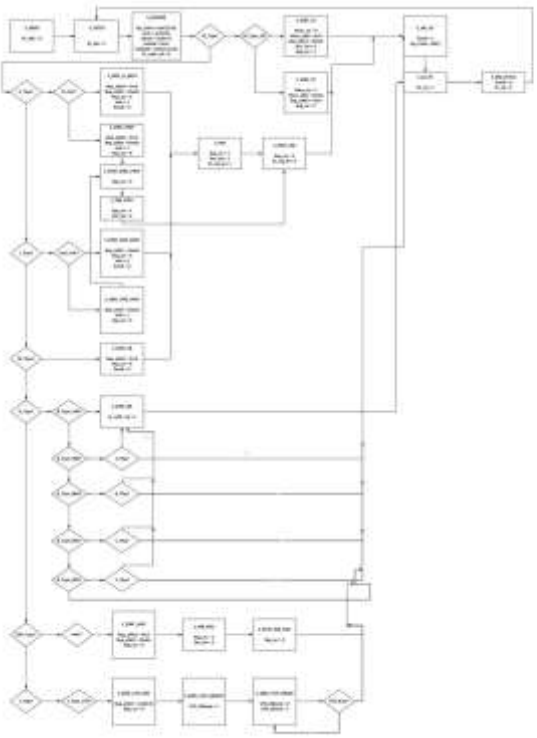
OR	001_101	2
COMP	001_110	2
MUL	001_111	2

INST	Opcode	Operands
I-Type		
MOVi	010_000	2
ADDi	010_001	2
SUBi	010_010	2
XORi	010_011	2
ANDi	010_100	2
ORi	010_101	2
COMPi	010_110	2
SR-Type		
Neg	011_000	1
comp	011_001	1
SRL	011_010	1
SLL	011_011	1
DEC	011_100	1
INC	011_101	1
ASR	011_110	1
CLR	011_111	1

INST	Opcode	Operands
B-Type		
JMP	100_000	1
BRZ	100_001	1
BRN	100_010	1
BRC	100_011	1
BRV	100_100	1
DSP-Type		
MAC	101_000	2
T-Type		
UTX	110_000	1

The processor will be able to perform the following functions, based on the 6-bit Opcode.

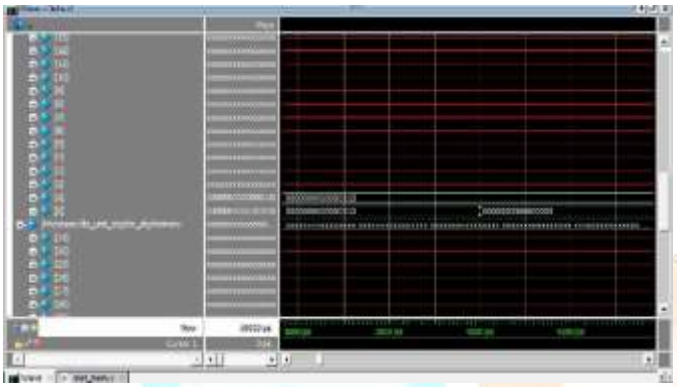
3.3 CONTROL UNIT



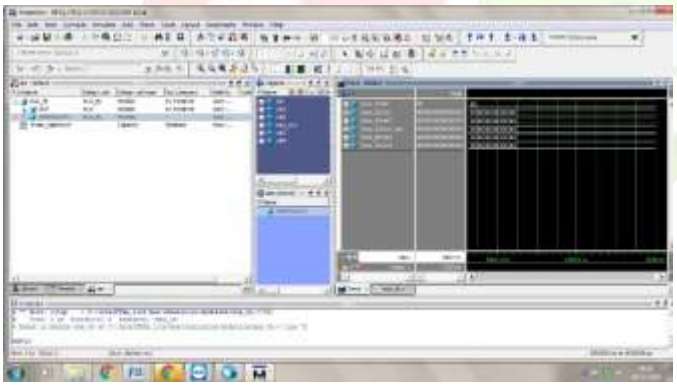
The control unit (CU) is a component of the processor. It tells the computer's memory, arithmetic and logic unit and input and output devices how to respond to the instructions that have been sent to the processor. It directs the operation of the other units by providing timing and control signals. Most computer resources are managed by the CU. Hardwired control units are implemented through use of combinational logic units, featuring a finite number of gates that can generate specific results based on the instructions that were used to invoke those responses. Hardwired control units are generally faster than the micro programmed designs.

The FSM (finite State Machine) for the control unit is shown above.

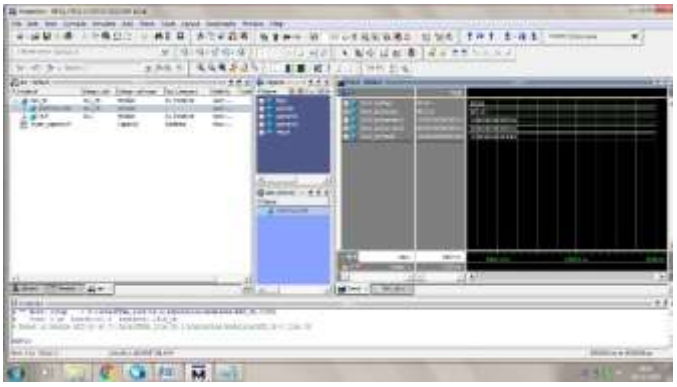
4. RESULT



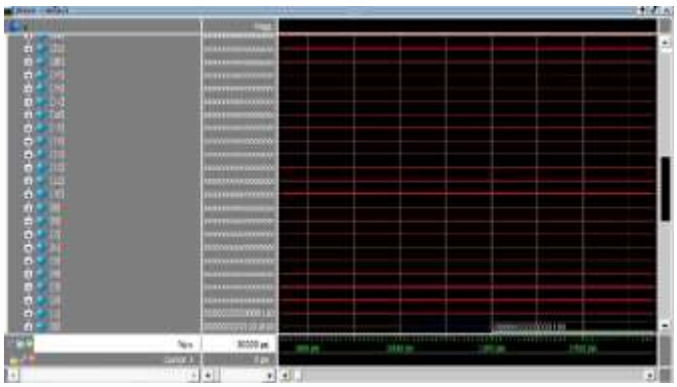
[1] The above screenshots show the simulation of subtraction performed on the processor. It shows the input, output as well as the opcode bits. We used 10.5b version for the simulation purpose.



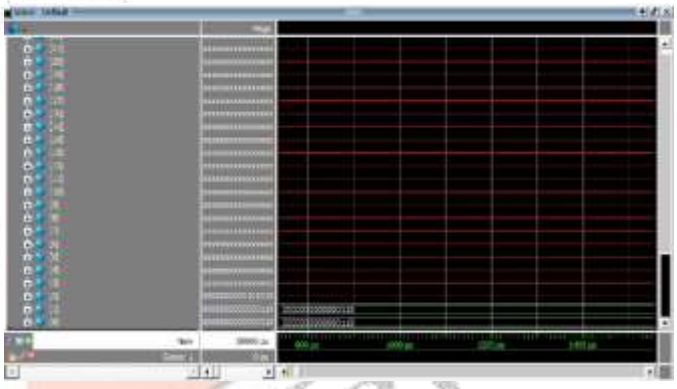
[2] The above screenshot shows the simulation of the 4:1 MUX unit.



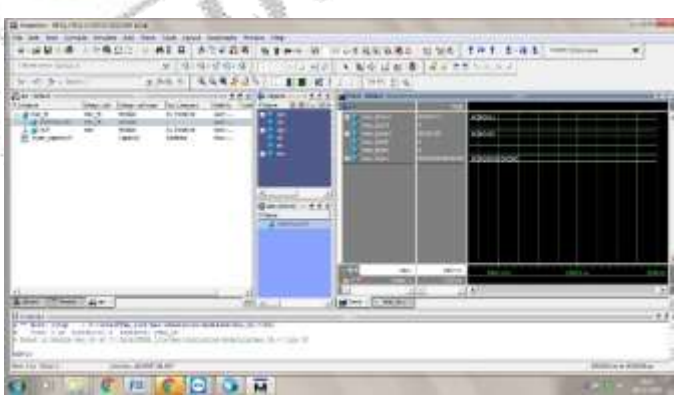
[3] The above screenshot shows the generation of flag bits when multiplication is performed. The opcode for multiplication is 001111 which can be verified from the instruction table in 3.2 ALU.



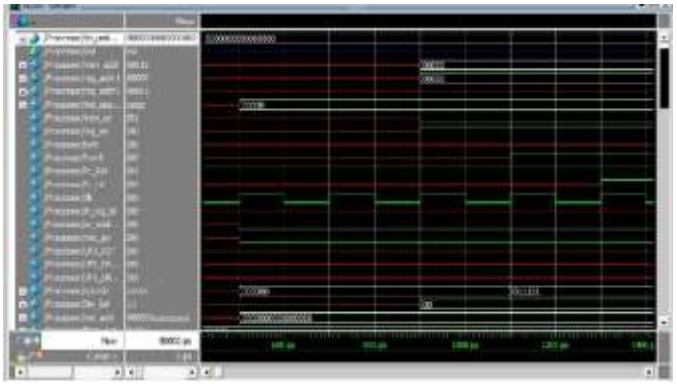
[4] The above screenshot shows the simulation of the register files with given input and output received from the memory.



[5] The above screenshot shows the simulation of data memory with read and write instruction.



[6] The screenshot shows simulation of MAC unit.



[7] The screenshot shows the overall simulation of the datapath code which interconnects all our processor blocks together.

5. CONCLUSION

Thus, we have designed and simulated a 16-bit RISC processor using Verilog HDL and verified its working by simulation. The processor can be extended to a 32 or even a 64-bit processor in the future by simple changes in the code and the datapath can be altered to include various new blocks which is not possible on a traditional processor unit.

6. REFERENCES

[1] Jikku Jeemon, "*Low power pipelined 8-bit RISC processor design and implementation on FPGA"*, ICCICCT 2015.

[2] Supraj Gaonkar and Anitha M, *"Design of 16-bit RISC Processor"*, IJERT Vol. 2 Issue 7, July 2013.

[3] D. J. Smith, "*HDL Chip Design"*, International Edition, Doone Publications, 2000.

[4] J.F. Wakerly, "*Digital Design: Principles and Practices"*, Third Edition, Prentice-Hall, 2000.

[5] A. S. Tanenbaum, "*Structured Computer Organization"*, Fourth Edition, Prentice-Hall, 2000.

[6] Yatin Trivedi and others, "*Verilog HDL",* IC*,* 2000.

[7] Mauriss M Mano, "*Digital Design",* Third Edition, Perason Edition, 2000.

[8] Shraddha M. Bhagat and Sheetal U. Bhandari, *"Design and Analysis of 16-bit RISC Processor",* Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018.