DESIGN OF PIPELINED RISC MIPS PROCESSOR (16-BIT) USING VLSI TECHNOLOGY

[1]Nyamatulla M Patel, [2]Pramod V Patil, [3]Gazalatabassum M Alan, [4]Priyanka B Done
[1]Assistant Professor, [2]Assistant Professor, [3] Student, [4]Student
[1]Department of Electronics and Communication Engineering
[1]Hirasugar Institute of Technology, Nidasoshi-591236, Belagavi, Karnataka, India

Abstract: The main aim of this paper is to design and implement RISC MIPS processor using VLSI technology. The project involves simulation and synthesis. The processor is designed with Verilog HDL, synthesized using XILINX-13.1. A Reduced Instruction Set compiler (RISC) is a microprocessor that had been designed to perform a small set of instructions, with the aim of increasing the overall speed of the processor. The idea of this project was to create a RISC MIPS processor as a building block in Verilog HDL. Each block is separated by pipeline to speed up the processor. High level of complexity is easier to implement the function in software. The objective of project is to increase the speed and reduce the power consumption. Single cycle execution method applied to complete one instruction through all stages.

Index Terms: RISC, MIPS, Pipelined Processor, Verilog HDL

I. INTRODUCTION

Reduced Instruction Set Computers (RISCs) are now used for all types of computational tasks such as DSP, DIP etc. A RISC is a microprocessor that is designed to perform a smaller number of types of computer instruction so that it can operate at a higher speed. John von Neumann designed a Reduced Instruction Set Computer (RISC) includes separate data memory and program memory to execute a set of instructions. The aim of project is to implement the five pipelined stage processor using RISC and MIPS architecture. Using single cycle, processor executes instructions and increases the overall speed and reduces the power consumption. In this work, analyze MIPS instruction format, instruction execution path through all stages, control unit performance for each instruction. Project designed with RISC philosophy, for load and store separate instructions used. To avoid access of memory repeatedly, separate register bank is designed. The project is build using Verilog HDL. The code is synthesized and simulated using XILINX-13.1.

II. OBJECTIVES

- To increase instruction execution speed and to reduce the power consumption of RISC processor.
- To apply single cycle execution method to complete one instruction through all stages.

III. MOTIVATION

Reduced Instruction Set Computer is a type of microprocessor. RISC processors are also used in supercomputers such as 'k' computer and especially representing a major force in the UNIX workstation market as well as embedded processors. It reduces the transistor count of a MIPS processing unit by scaling down the bus and register width.

IV. LITERATURE SURVEY

Table1 Comparison of different technologies used for RISC implementation.

SL. No	Author	Title	Technology	Limitations
1.	N.Alekya, P.Ganesh Kumar	Design of 32-Bit RISC CPU Based on MIPS	MIPS	VHDL language is used to implement the 32bit processor.
2.	Galani Tina G. Riya Saini and R.D.Daruwala	Design and Implementation of 32 – bit RISC Processor using Xilinx	Spartan 2E	A 32 bit RISC processor build using Xilinx virtex4 Tool for embedded and portable applications required minimum area and minimum delay.
3.	Navneetkaur, Adesh Kumar, Lipika Gupta	VHDL Design and Synthesis of 64 bit RISC Processor System on Chip (SOC)	Spartan 2	Four stage Pipelines used. 8bit and 16 bit instruction set is used to access logical, arithmetic and memory, jump instructions.
4.	R. Uma	Design and Performance analysis of 8-bit RISC Processor	Xlinix Tool	For sign multiplication Booth Multiplier is used

V. SYSTEM MODELING

Figure 1 illustrates the procedure to design a system. With the help of XILINX 13.1, all stages of processor are developed as a separate module using Verilog language. Bit file generation is important criteria to dump into any FPGA.

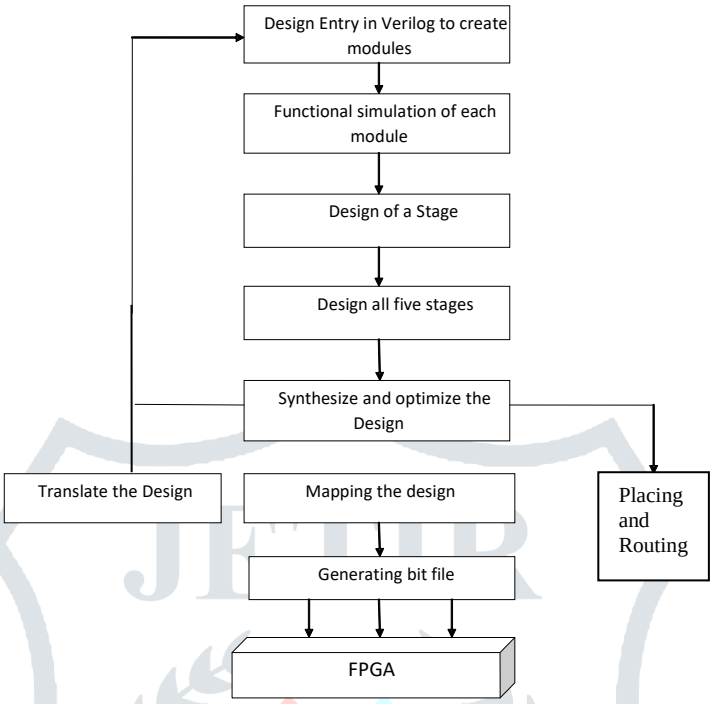

Figure 1: Generation of bit stream for FPGA

Construct five pipeline stages

The project includes instruction fetch stage (IF), decode stage (ID), execution stage (EX), data memory (MEM), write back (WB) stages. For jump instruction separate block is designed. Pipeline consists of the overlapping of set of instructions. Pipeline reduces the execution time of instructions. The processor executes an instruction in single cycle. Each instruction passes through all stages or passes according to instruction. Program counter loaded with the address it acts as a pointer to the program memory.

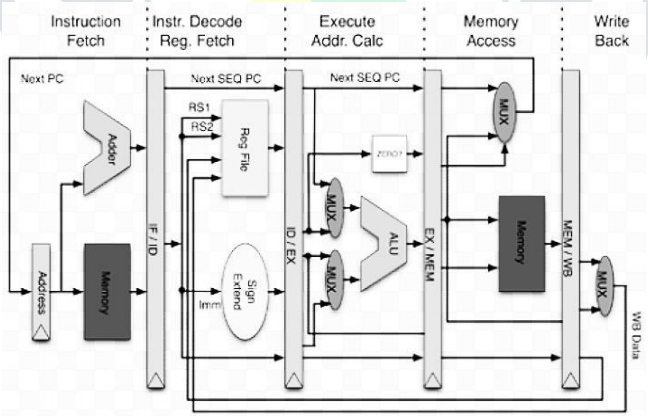

Figure 2: Five Stage Pipeline

Instruction Fetch

The program memory loaded with Instructions. The instruction fetched from memory. Each time program counter has the address of memory location. All fetched instructions passed to decoder stage through latch.

Instruction Decode

The Instruction Decode stage decodes the instruction. The six bit op-code is passed to the control unit to generate the signals according to instruction code. The data passed through decoder latch for execution as per signals generated by control unit. Sign extension unit used to extend the value according to control signals. If instruction is load or store type, immediate [15-0] bits extend to 32 bit for ALU. In decode stage Register Bank performs as cache memory to store the data for fast calculation.

Execute Stage

In Execute stage, the instructions are executed. All arithmetic and logical operations perform by ALU. Type of operations are addition, subtraction, AND, OR etc.

Memory Access
Data memory is storage device accessed as per the load and store instruction. For load instruction, Data is loaded to register bank from memory. For store instruction, current data stores in the data memory.
Write back stage
The result writes back to the register file. All instructions passed via Write back stage. Except nops and store type of instruction.

Advantages:

- Easier to implement.
- Faster clock speed.
- Power consumed per instruction execution is less.
- Simpler hardware.
- Shorter design cycle.
- Fewer transistors count for RISC cores.

Applications:

- Used in video processing, telecommunication and image.
- Used in digital signal processing.
- Used in high performance applications like servers (mobile).
- Used in embedded applications where ultra low power consumption is needed.
- Used in virtualization and memory management.

VI. RESULTS AND DISCUSSIONS

Manual Calculation:

Manual Calculation for 16 bit ALU:
Consider two inputs at port1 and port2 of 16 bits each and Op-code of 3 bits.
Port1= 1111111100000000
Port2= 0000000011111111

Table 2 Manual Calculation of 16 Bit ALU

Opcode	Opcode Description	Result
000	Addition	1111111111111111
001	Subtraction	1111111000000001
010	AND	0000000000000000
011	OR	1111111111111111
100	EX-OR	1111111111111111
101	NOT(port1)	0000000011111111
110	Shift Left (Port1)	1111111000000000
111	Shift Right(Port1)	0111111110000000

Manual Calculation for 16 bit Adder:

Let us consider two inputs A and B of 16 bits each and one input C of one bit.
A=1010101010101010
B=1111000011110000
C=0
Sum[i]= A[i]^B[i]^C[i]
Cout= (A[i]&B[i]) | (B[i]&C[i]) | (C[i]&A[i])
Cp=sum[0]^sum[1]^sum[2]^sum[3]^sum[4]^sum[5]^sum[6]^sum[7]^sum[8]^sum[9]^sum[10]^sum[11]^sum[12]^sum[13]^ sum[14]^sum[15]

Thus output is

Cp=1, Sum=0101100111011001, Cout=0000011100000111

Table 3 Manual Calculation of 16 Bit Adder

A[i]	B[i]	Sum[i]	Cout
0	0	0	0
1	0	1	0
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1
0	1	0	1
1	1	1	1
0	0	1	0
1	0	1	0
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1
0	1	0	1
1	1	1	1

Manual Calculation for 16 Bit Logic Unit:

Consider two inputs ain & bin each of 16 bits and 2 bit selection line sel.
Five 16 bit Outputs and_out, or_out, xor_out, xnor_out and res_out
For sel=00 res_out=and_out
sel=01 res_out=or_out
sel=10 res_out=xor_out
sel=11 res_out=xnor_out

Table 4 Manual Calculation of 16 Bit Logic Unit

ain[i]	bin[i]	and_out	or_out	xor_out	xnor_out
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	0
1	1	1	1	0	1
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	0
1	1	1	1	0	1
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	0
1	1	1	1	0	1
0	0	0	0	0	1
1	0	0	1	1	0
0	1	0	1	1	0
1	1	1	1	0	1

RISC Processor (16-Bit ALU)

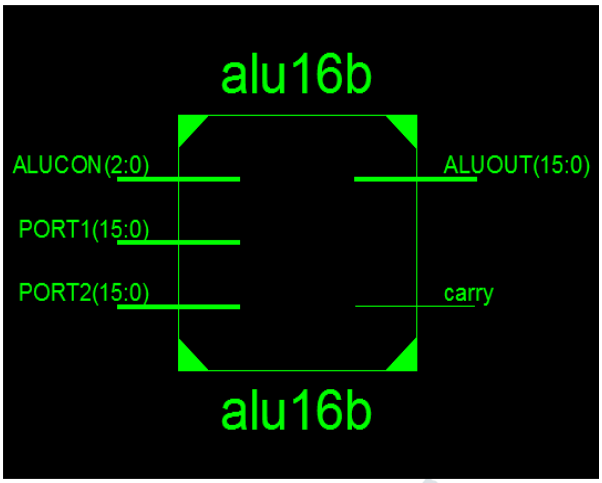

Figure 3: Top Module of 16-Bit ALU

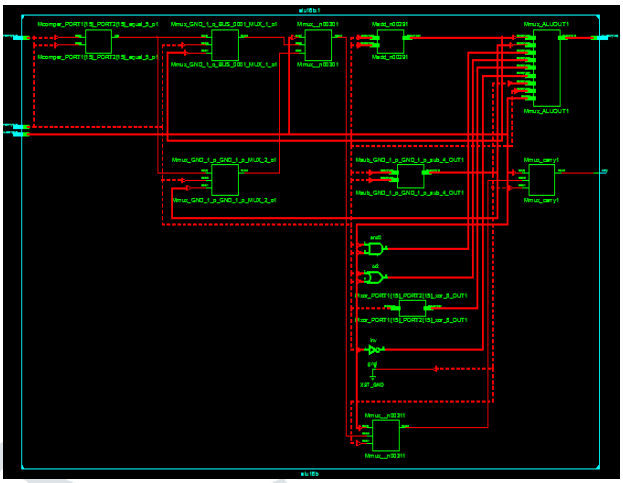

Figure 4: RTL Schematic View of 16-Bit ALU

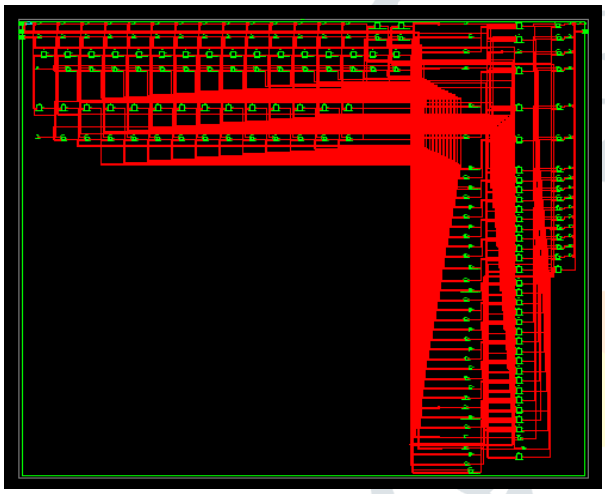

Figure 5: Technological Schematic View of 16-Bit ALU

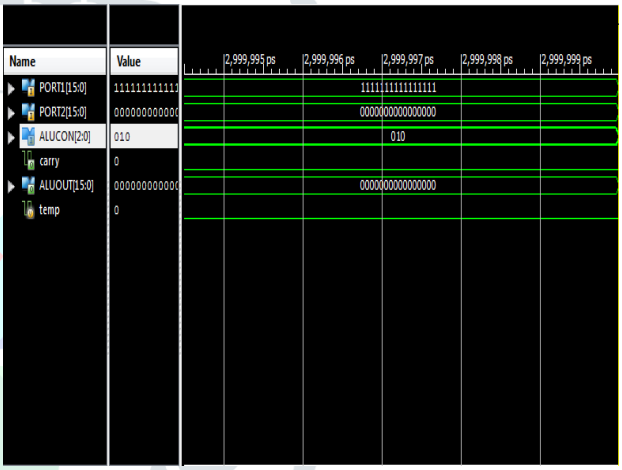

Figure 6: Simulation Results of 16-Bit ALU

Table 5 Timing Report of 16-Bit ALU

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	66	178800	0%
Number of fully used LUT-FF pairs	0	66	0%
Number of bonded IOBs	52	600	8%

16-Bit Adder:

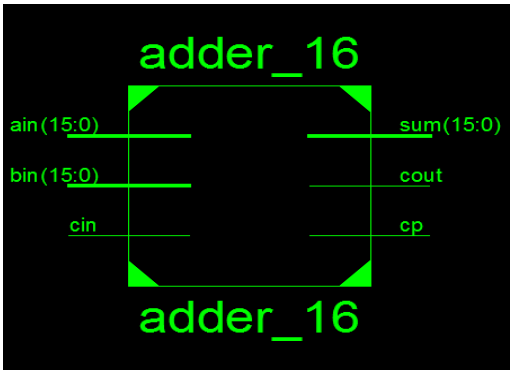

Figure 7: Top Module of 16-Bit Adder

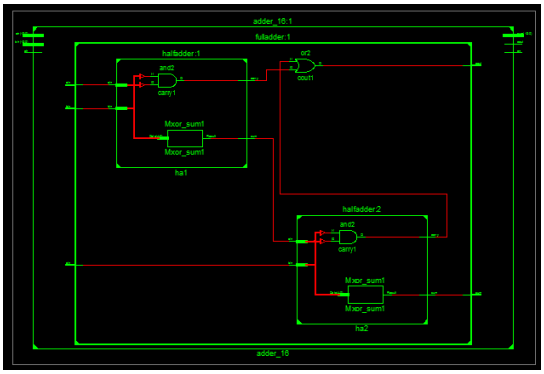

Figure 8: RTL Schematic View of 16-Bit Adder

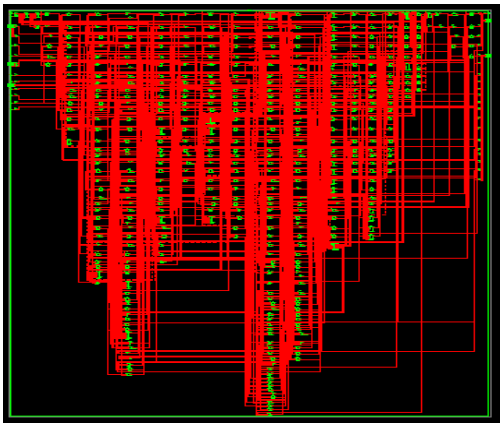

Figure 9: Technological Schematic View 16-Bit Adder

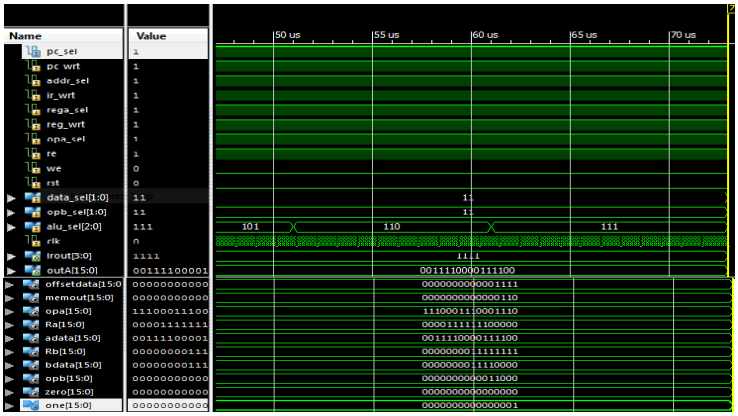

Figure 10: Simulation Results of 16-Bit Adder

Table 6 Timing Report of 16-Bit Adder

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	28	178800	0%
Number of fully used LUT-FF pairs	0	28	0%
Number of bonded IOBs	51	600	8%

16-Bit Logic Unit:

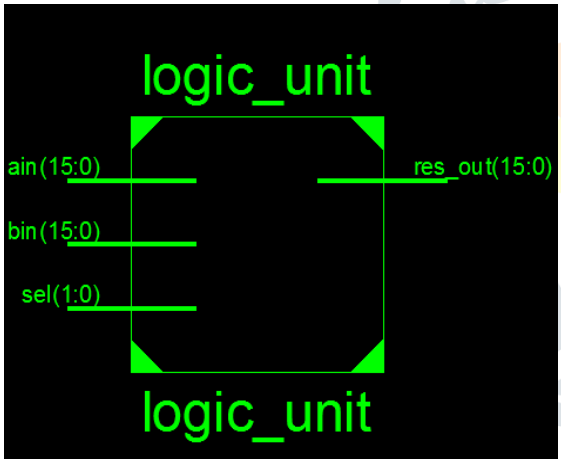

Figure 11: Top Module of 16-Bit Logic Unit

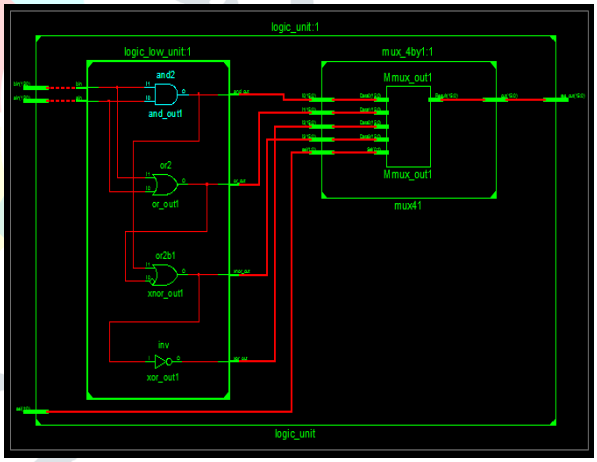

Figure 12: RTL Schematic View of 16-Bit Logic Unit

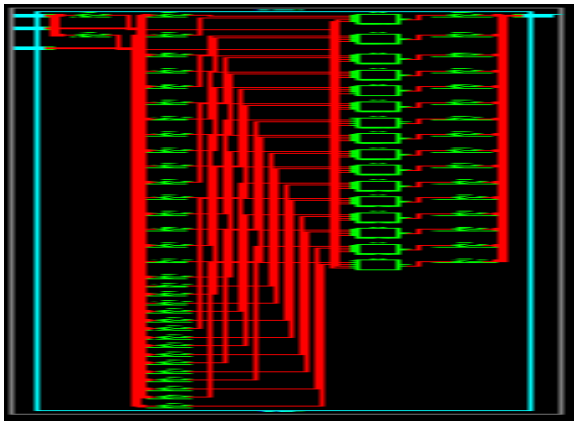

Figure 13: Technological Schematic View of 16-Bit Logic Unit

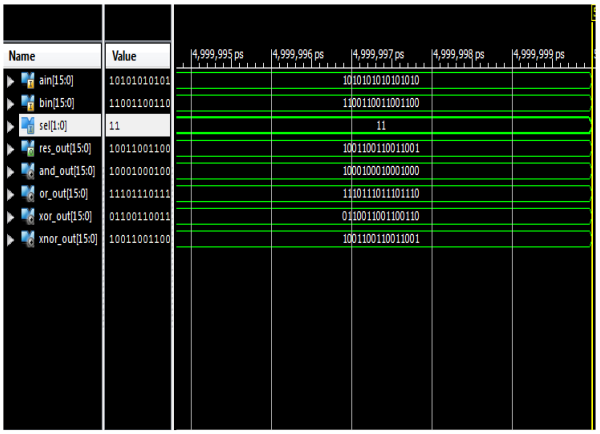

Figure 14: Simulation Results of 16-Bit Logic Unit

Table 7 Timing Report of 16-Bit Logic Unit

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice LUTs	16	178800	0%
Number of fully used LUT-FF pairs	0	16	0%
Number of bonded IOBs	50	600	8%

16-Bit Control Unit:

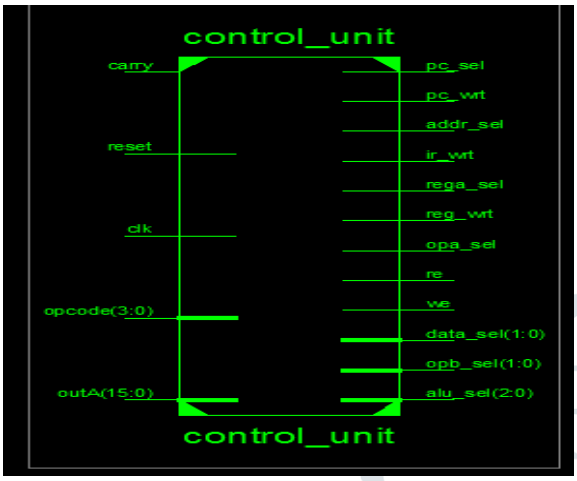

Figure 15: Top Module of 16-Bit Control Unit

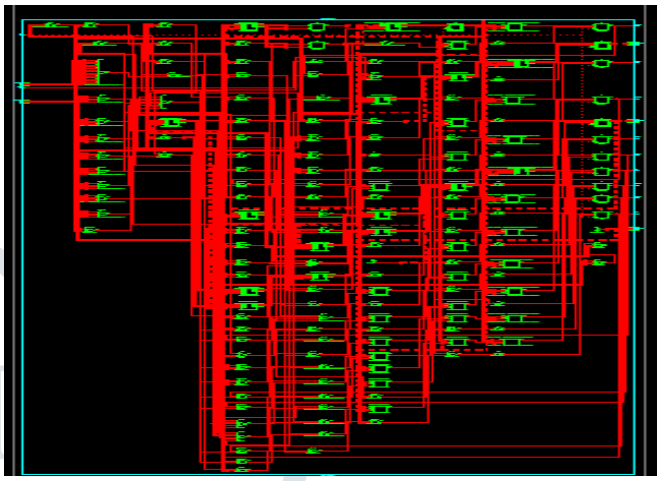

Figure 16: RTL Schematic view of 16-Bit Control Unit

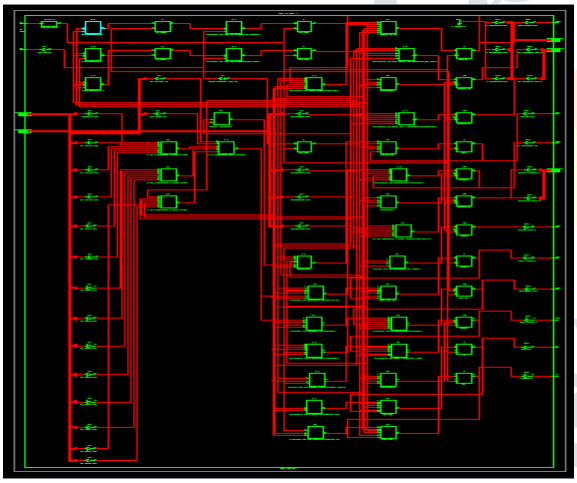

Figure 17: Technological Schematic View of 16-Bit Control Unit

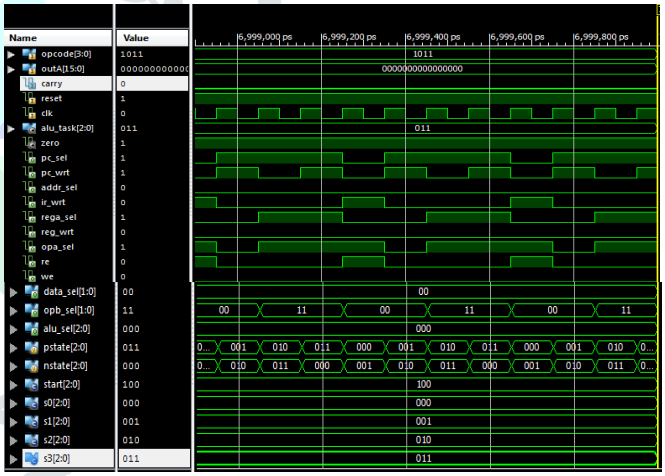

Figure 18: Simulation Results 16-Bit Control Unit

Table 8 Timing Report of 16-Bit Control Unit

Device Utilization Summary				[-]
Slice Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of Slice Registers	17	357,600	1%	
Number used as Flip Flops	15			
Number used as Latches	2			
Number used as Latch-thrus	0			
Number used as AND/OR logics	0			
Number of Slice LUTs	28	178,800	1%	
Number used as logic	28	178,800	1%	
Number using O6 output only	22			
Number using O5 output only	0			
Number using O5 and O6	6			
Number used as ROM	0			
Number used as Memory	0	55,600	0%	
Number used exclusively as route-thrus	0			
Number of occupied Slices	13	44,700	1%	
Number of LUT Flip Flop pairs used	29			
Number with an unused Flip Flop	13	29	44%	
Number with an unused LUT	1	29	3%	
Number of fully used LUT-FF pairs	15	29	51%	
Number of unique control sets	6			

16-Bit Data Path:

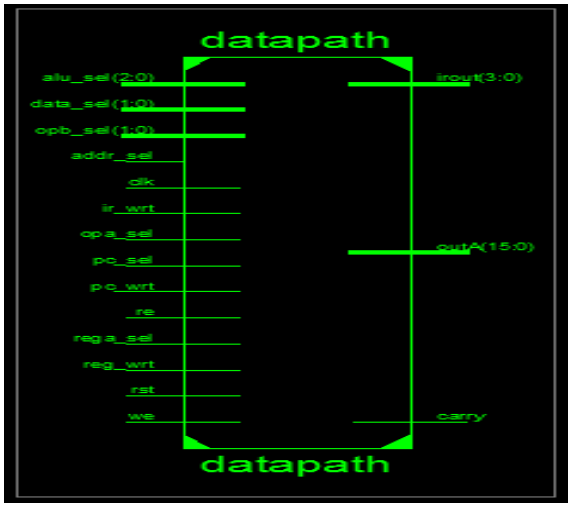

Figure 19: Top Module of 16-Bit Data Path

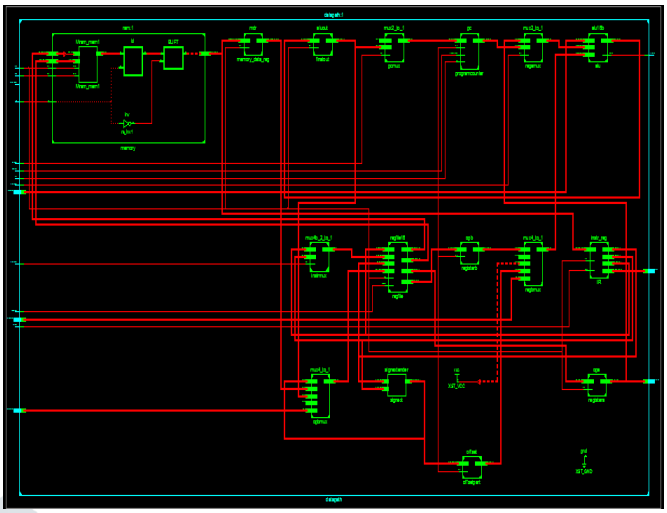

Figure 20: RTL Schematic View of 16-Bit Data Path

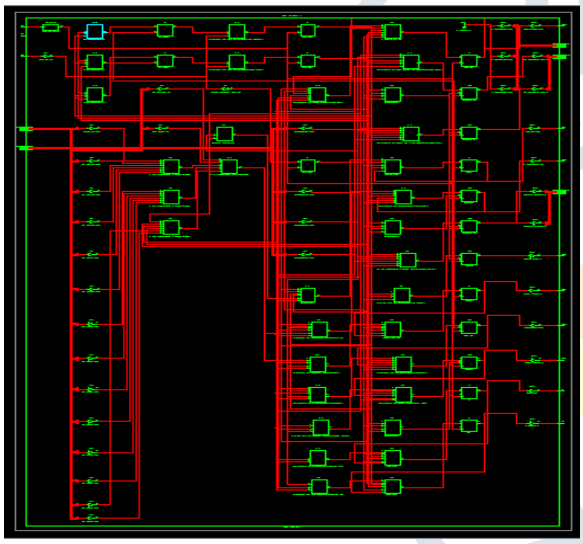

Figure 21: Technological Schematic View of 16-Bit Data Path

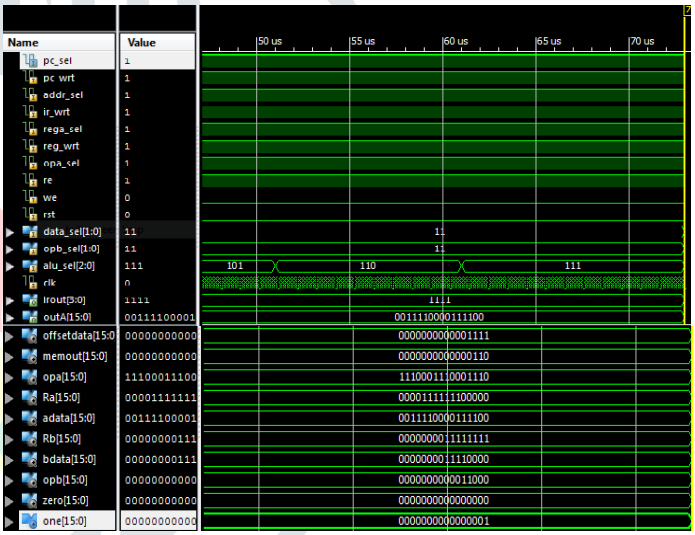

Figure 22: Simulation Results of 16-Bit Data Path

Table 9 Timing Report of 16-Bit Data-path

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	116	357600	0%
Number of Slice LUTs	264	178800	0%
Number of fully used LUT-FF pairs	96	284	33%
Number of bonded IOBs	38	600	6%
Number of BUFG/BUFGCTRLs	2	32	6%

Program execution on FPGA kit:

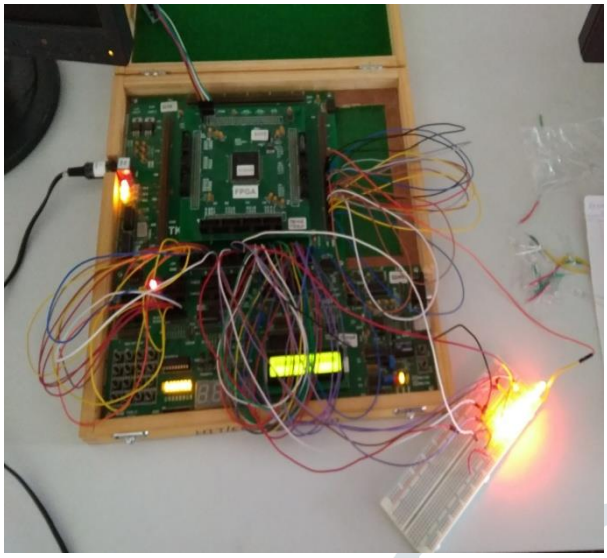

Figure 23: Execution of 16-Bit ALU

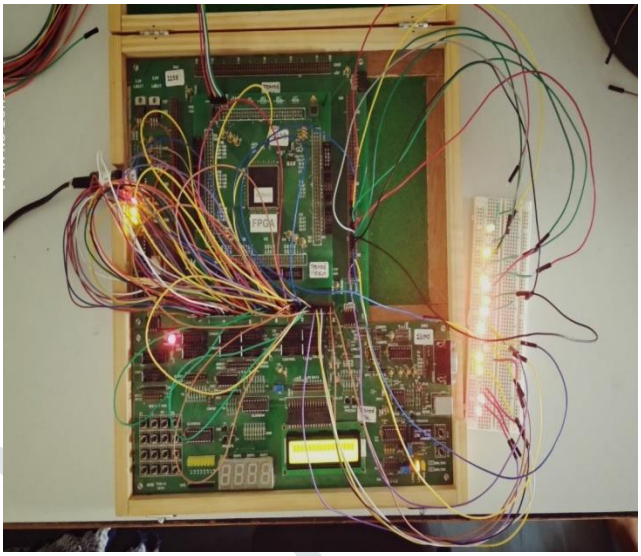

Figure 24: Execution of 16-Bit Adder

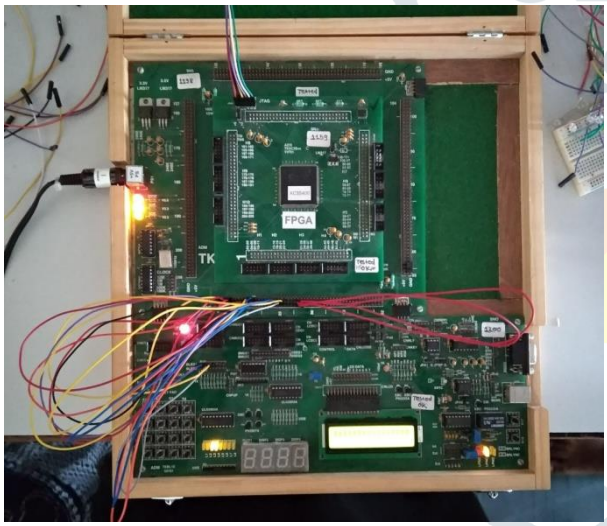

Figure 25: Execution of 16-Bit Logic Unit

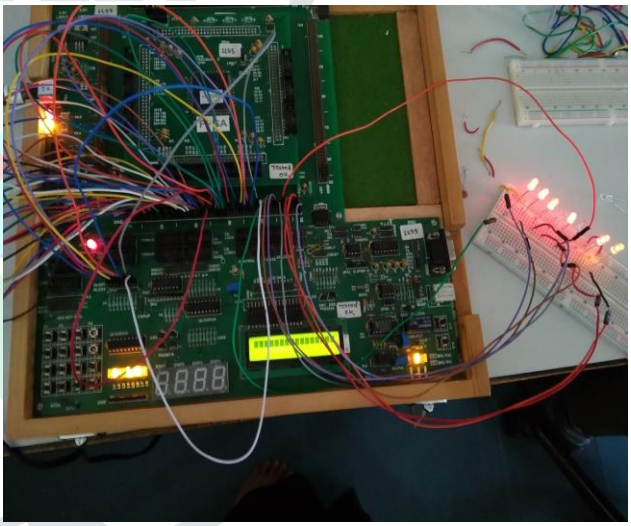

Figure 26: Execution of 16-Bit Control Unit

Comparison of RISC with pipeline and RISC without pipeline:

Table 10 Comparison of time delay and power dissipation for RISC processor and proposed RISC processor

Design Type	No. of bits	Time delay(ns)	Power dissipation (mWatt)
RISC Processor	16 bit	22.323ns	0.87
Implemented RISC Processor	16 bit	16.732ns	0.098

Future Scope and Conclusion:

The RISC MIPS architecture of pipelined 16 bit is executed successfully and got the correct results. The project has reduced the power consumption considerably, and increased the speed of execution. The program has given correct results for number of times. The project code is much efficient in terms of number of bits and in reducing the amount complexity accomplished. This project can be further modified to obtain the results for the numbers with the numbers of bits greater than 32 bit and 64 bit and memory management.

References:

1] Galani Tina G., Riya Saini and R. D. Daruwala, Design and Implementation of 32-bit RISC Processor using Xilinx, IJITEE Volume-5, Issue-1, August 2013.

2] J. Poornima, G. V. Ganesh, M. jyothi, M. Sahithi, A. Jhansi Rani B.RaghuKant," Design and Implementation of Pipelined 32-bit Advanced RISC Processor for Various D. S. P Applications", (IJCSIT)International Journal of Computer Sceince and Information Technologies, Vol. 3(1) 2012, 3208-3213.

3] Marri Mounika, Aleti Shankar, Design and Implementation of 32-bit RISC (MIOS) Processor, International Journal of Engineering Trends and Technology(IJETT) Volume-4 Issue 10-Oct 2013.

4] N-Alekya, P. Ganesh Kumar, Design of 32-bit RISC CPU Based on MIPS, JGRCS Volume 2, No 9, September 2011.

5] Navneet Kaur, Adesh Kumar, Lipika Gupta, VHDL Design and synthesis of 64-bit RISC Processor System on Chip (SoC), IOSR Journal of VLSI and Signal Processing(IOSR_JVSP) Volume-3, Issue 5 (Nov-Dec 2013), PP 31-41 e-ISSN:2319-4200, p-ISSN No:2319-4197.

6] Preetam Bhosale, Hari Krishna Murti, FPGA Implementation of Low Power Pipelined 32-bit RISC Processor, IJITEE Volume-1, Issue-3, August 2012.

7] R. Uma, Design and Performance Analysis of 8-bit RISC Processor using Xilinx Tool, International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622, Vol 2, Issue 2, Mar-Apr 2012, pp. .053-058.