San Jose State University
Department of Computer Engineering

CMPE 200 Report

Assignment 5 Report

Title Processor Design (1): Design Code Review and Functional Verification

Semester Fall 2022 **Date** 10/10/2022

by

Name HARISH MAREPALLI *(typed)* **SID** 016707314 *(typed)*

ABOUT THE AUTHOR

Harish Marepalli had done bachelor's in Electronics and Communication Engineering at Gokaraju Rangaraju Institute of Engineering and Technology, Hyderabad, Telangana, India. He had 3 years of experience as a Systems Engineer in Tata Consultancy Services and his role was software developer. Currently, he is pursuing his master's in Computer Engineering at San Jose State University. His native place is Hyderabad and mother tongue is Telugu. Additionally, he does speak English and Hindi and would love to connect with the people of any region and build network.

LIST OF CONTENTS

1. INTRODUCTION

This activity is used to gain hands-on processor design experience by reviewing RTL Verilog design code for the initial version of the single-cycle MIPS processor. It is also used to learn the basic technique for functionally verifying a processor.

2. MY GROUP

Name: Student_Team 6
Harish Marepalli – 016707314
Tirumala Saiteja Goruganthu – 016707210

3. GIVEN TASK

a. The task is to thoroughly review MIPS instruction format.
b. To carefully study the provided source code and understand it.
c. To draw datapath with microarchitecture details without including memories.
d. To draw control unit with microarchitecture details.
e. To draw instruction memory and data memory.
f. To draw processor core by showing interconnection between the datapath and its control unit.
g. To draw complete processor by showing interconnections between the processor core and its memories.
h. To run the Verilog code and capture the simulation waveform.
i. Also, to monitor the processor's execution on each instruction by viewing waveforms of the following signals:
clk, rst, pc_current, instr, alu_out, we_dm, wd_dm, rd_dm

4. GIVEN INFORMATION TO ACHIEVE THE TASK

a. MIPS instructions.
b. Verilog design source code files for the initial design of the single-cycle MIPS processor.
c. A memory file 'memfile.dat' containing machine code of the given MIPS program which is to be used as an example for processor functional verification.
d. A Verilog testbench file 'tb_mips_top' for checking the processor's execution results instruction by instruction.
e. A signal naming file 'MIPS Signal Naming_v1', which gives definitions of all the signals that appears in the source code, as well as their relationship with the signals appeared on the lecture slides.

5. STEPS TAKEN TO COMPLETE THE TASK

a. Understood the information that is already provided and noted the important points.
b. Used online resources to understand how to draw block diagrams for the given tasks.
c. Drawn Datapath and other diagrams by using the modules given in the Verilog code.
d. Drawn Control Unit by carefully giving the input and output signals for main decoder and aux decoder.
e. Used Visio to draw instruction and data memory by giving the correct input and output signals.
f. Connected the control unit with the datapath to form a Processor Core system.
g. Drawn Complete processor by including MIPS processor core, instruction memory module and data memory module and also correct signals.
h. Understood the Verilog code and verified the output by running the code and checking simulation waveforms.

6. DATAPATH

a. The datapath system shows the interconnection with the various modules including the external signals coming and going to other submodules of the MIPS processor.
b. The input signals to the datapath are:
    a. alu_ctrl, which is a 3-bit signal.
    b. alu_src, which is a single bit signal.
    c. branch, which is a single bit signal.
    d. clk, which is a single bit signal.
    e. dm2reg which is a single bit signal.
    f. instr, which is a 32-bit signal.
    g. jump, which is a single bit signal.
    h. ra3, which is a 5-bit signal.
    i. rd_dm, which is a 32-bit signal.
    j. reg_dst, which is a single bit signal.
    k. rst, which is a single bit signal.
    l. we_reg, which is a single bit signal.
c. The output signals from the datapath are:
    a. alu_out, which is a 32-bit signal.
    b. pc_current, which is a 32-bit signal.
    c. rd3, which is a 32-bit signal.
    d. wd_dm, which is a 32-bit signal.
d. It consists of a signext (*Figure 7 in appendix*) module, which takes 16-bit input (instr[15:0]) ‘a’ and produces the 32-bit output ‘y’.
e. It contains an adder (*Figure 8 in appendix*), which is used to increment the value of PC by 4. It takes the 32-bit input ‘a’ and ‘b’ and gives the 32-bit output ‘y’.
f. It has an AND gate (*Figure 9 in appendix*) which takes branch input and one more input from the alu module and performs the AND operation.
g. It consists of another adder, which takes 32-bit input from the previous adder and the other input comes from one of the multiplexers.
h. There are 5 multiplexers used in this datapath. The first multiplexer, ‘alu_pb_mux’ takes 32-bit ‘a’ which is given by the register file and the other 32-bit input comes from the signext module. The third input is a select signal.
i. The second multiplexer, ‘rf_wd_mux’ (*Figure 10 in appendix*) takes one 32-bit input ‘a’ from alu module and another 32-bit input and a select input. It gives a 32-bit output ‘y’ to register file.
j. The third multiplexer, ‘pc_src_mux’ takes one 32-bit input ‘a’ from one of the adders and another 32-bit input from another adder and a select signal which comes from the AND gate. It gives its output to the next multiplexer.
k. The fourth multiplexer, ‘pc_imp_mux’ takes one 32-bit from the previous multiplexer and another 32-bit and a jump select signal. It produces a 32-bit output ‘y’, which is given to the ‘dreg’ register.
l. The fifth multiplexer, ‘rf_wa_mux’ takes one 5-bit input ‘a’ and another 5-bit input ‘b’, and a select signal. This produces a 5-bit output, which is given to the register file.
m. It consists of an Arithmetic Logic Unit module (*Figure 11 in appendix*) which is named as alu, which takes a 32-bit input ‘a’ from the register file, the next 32-bit input ‘b’from the first multiplexer and the third 32-bit input ‘op’.
n. The alu produces a 32-bit output which is given to one of the multiplexers and the other output is given to the AND gate.
o. It contains a register file (*Figure 12 in appendix*) which takes 7 inputs. The first input is clock signal ‘clk’. The second input is a 5-bit read address. The third input is the second 5-bit read address. The fourth input is the third 5-bit read address. The fifth is a 5-bit write address, which comes from one of the multiplexers. The sixth input is a

32-bit write data which comes from one of the other multiplexers. The seventh input is a write enable signal.

p. The register file performs its operations and gives one of the 32-bit outputs to alu, the other 32-bit output to one of the multiplexers, and the final 32-bit output is a read data.
q. The datapath also consists of a module ‘pc_reg’ (*Figure 13 in appendix*), which takes 3 inputs. The first input is a clock signal. The second input is a 32-bit input ‘d’ which comes from one of the multiplexers. The third input is a reset signal.
r. This module gives its 32-bit output ‘q’ to the adder, which increments the PC value by 4.
s. *Figure 1* shown in appendix is the system’s datapath drawn using the above modules.

7. CONTROL UNIT

a. The control unit consists of a main decoder and an aux decoder.
b. The inputs to the control unit are:
    i. funct, which is a 6-bit input.
    ii. opcode, which is a 6-bit input.
c. The outputs from the control unit are:
    i. alu_ctrl, which is a 3-bit input.
    ii. alu_src, which is a single bit output.
    iii. branch, which is a single bit output.
    iv. dm2reg, which is a single bit output.
    v. jump, which is a single bit output.
    vi. reg_dst, which is a single bit output.
    vii. we_dm, which is a single bit output.
    viii. we_reg, which is a single bit output.
d. In this, the output alu_op from the maindec is given as input to the auxdec.
e. Another input to auxdec is a funct 5-bit signal.
f. *Figure 2* shown in appendix is a control unit drawn using the above blocks.

8. INSTRUCTION MEMORY

a. Instruction Memory is one of the modules that is used in Complete Processor system.
b. It takes a 6-bit input ‘a’ and produces a 32-bit output ‘y’.
c. This takes the memfile.dat file, which contains machine codes. This file is used in the Verilog code.
d. *Figure 3* shown in appendix is an instruction memory with the mentioned signals.

9. DATA MEMORY

a. Data Memory is another module used in the Complete Processor system.
b. It takes four inputs and produces one output.
c. The first input is a 6-bit input ‘a’.
d. The second input is a clock signal.
e. The third input is a 32-bit ‘d’.
f. The fourth input is a write enable signal.
g. It produces a 32-bit output ‘q’.
h. *Figure 4* shown in appendix is a data memory with the mentioned signals.

10. PROCESSOR CORE

a. Processor core consists of a control unit and a data path connected with each other.
b. The input signals to the processor core are:
    a. ra3, which is a 5-bit input.
    b. instr, which is a 32-bit input.
    c. rd_dm, which is a 32-bit input.
    d. clk, which is a single bit clock input.
    e. rst, which is a single bit reset input.
    f. we_dm, which is a single bit write enable input.
c. The output signals from the processor core are:
    a. alu_out, which is a 32-bit output.
    b. pc_current, which is a 32-bit output.
    c. rd3, which is a 32-bit output.
    d. wd_dm, which is a 32-bit output.
d. Some of the outputs that are coming from the control unit are given as inputs to data path.
e. *Figure 5* shown in appendix is a processor core system with all the mentioned inputs and outputs.

11. COMPLETE PROCESSOR

a. Complete Processor system is a combination of data memory, instruction memory, and a MIPS processor core.
b. The inputs to the MIPS processor core are: The first input is clk signal. The second input is instr 32-bit. The third input is ra3 5-bit. The fourth bit is rd_dm 32-bit. The fifth input is a reset signal.
c. The outputs from MIPS processor core are: The first output is alu_out 32-bit signal. The second output is pc_current 32-bit signal. The third output is rd3 32-bit signal. The fourth output is wd_dm 32-bit signal. The fifth output is we_dm signal.
d. *Figure 6* shown in appendix is a complete processor with the mentioned signals.

12. DISCUSSION SECTION

a. In this assignment we have drawn Datapath which consists of several modules.
b. Some of these signals are single, some are 5-bit and the others are 32-bit.
c. Every module does its respective operation.
d. These modules take respective input from other blocks and gives output to some other block.
e. The Control Unit with microarchitecture details is drawn with all the relevant signals. It consists of main decoder and aux dec which performs decoding operation.
f. There is an instruction memory block which takes a 6-bit input and produces a 32-bit output.
g. There is a data memory block which takes some signals and produces a single bit output.
h. The processor core consists of control unit and data unit connected with each other.
i. The control unit takes some signals which are a mix of different bit inputs and produces outputs some of which are given to the data path.
j. The datapath takes the required inputs are sends the relevant output signals.
k. The output of the processor core consists of all 32-bit signals.
l. Complete processor is a system which consists of an instruction memory, a data memory, and a MIPS processor core module.
m. This complete takes the relevant signals from data memory and instruction memory to produce the required outputs.

n. *Figure 14 and Figure 15* in appendix shows the simulation which consists of processor's execution result showing the signals for each instruction.

13. COLLABORATION SECTION

1. Worked together on the datapath and control unit block diagram designs using Microsoft Vision tool.
2. Imported the given source files into Vivado and observed the output waveforms after the simulation by collaborating with each other.
3. Understood how to draw processor core and complete processor by collaborating with each other.
4. By working together, we have understood the testbench and simulation waveforms.

14. CONCLUSION

In conclusion, this activity has helped me to gain experience in processor design by reviewing the RTL code of the initial version of the single-cycle MIPS processor. The analysis of the code allowed me to draw the block diagrams for the Datapath, Control Unit, Instruction Memory, Data Memory, Processor Core, and Complete Processor. In addition, I learned the basic techniques to functionally verify a processor. This was achieved by verifying processor's execution results through the generated waveforms by the testbench.

APPENDIX

Data Path:

Data path in Portrait

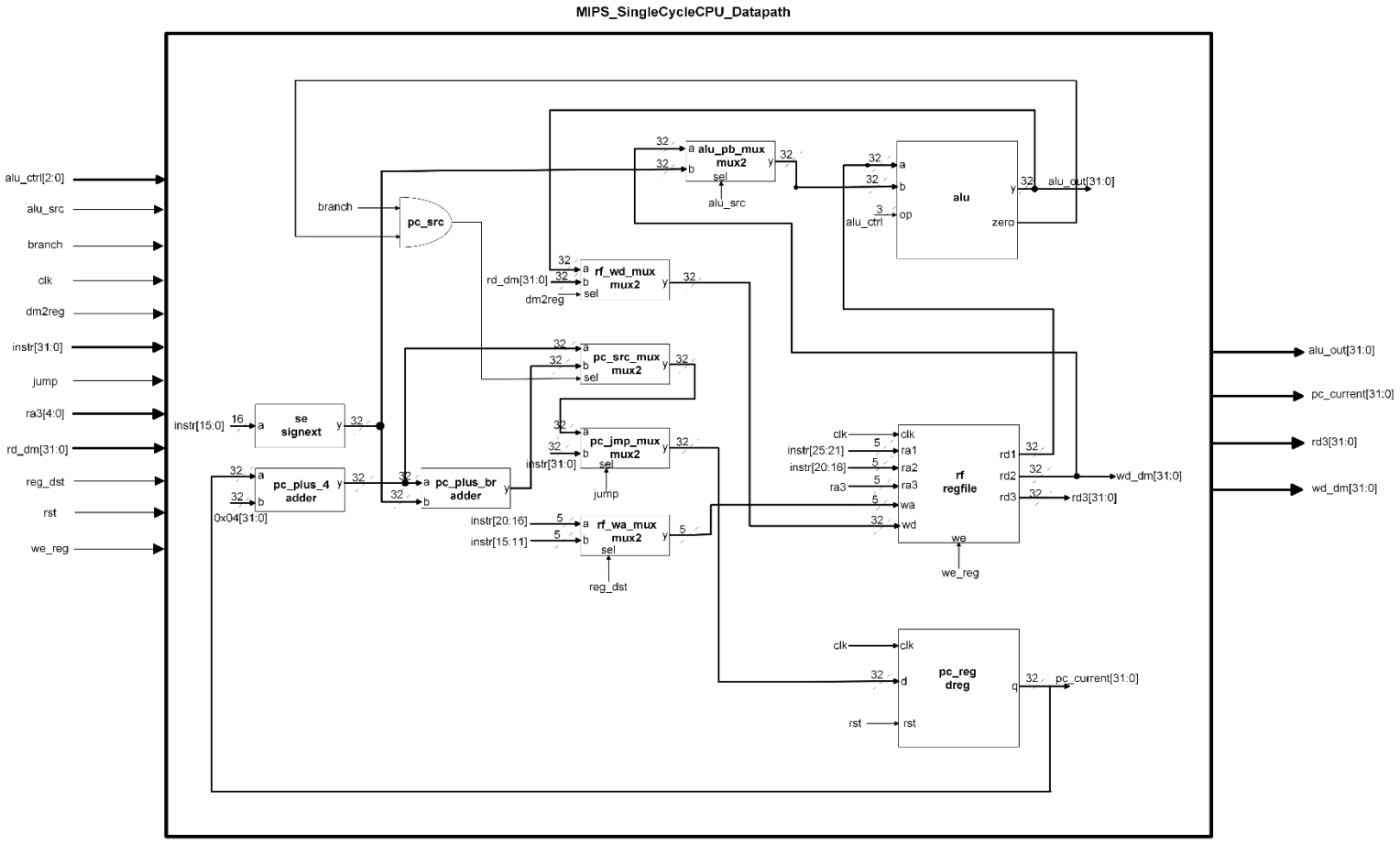

Fig. 1: MIPS Single Cycle CPU Data path

Data path in landscape

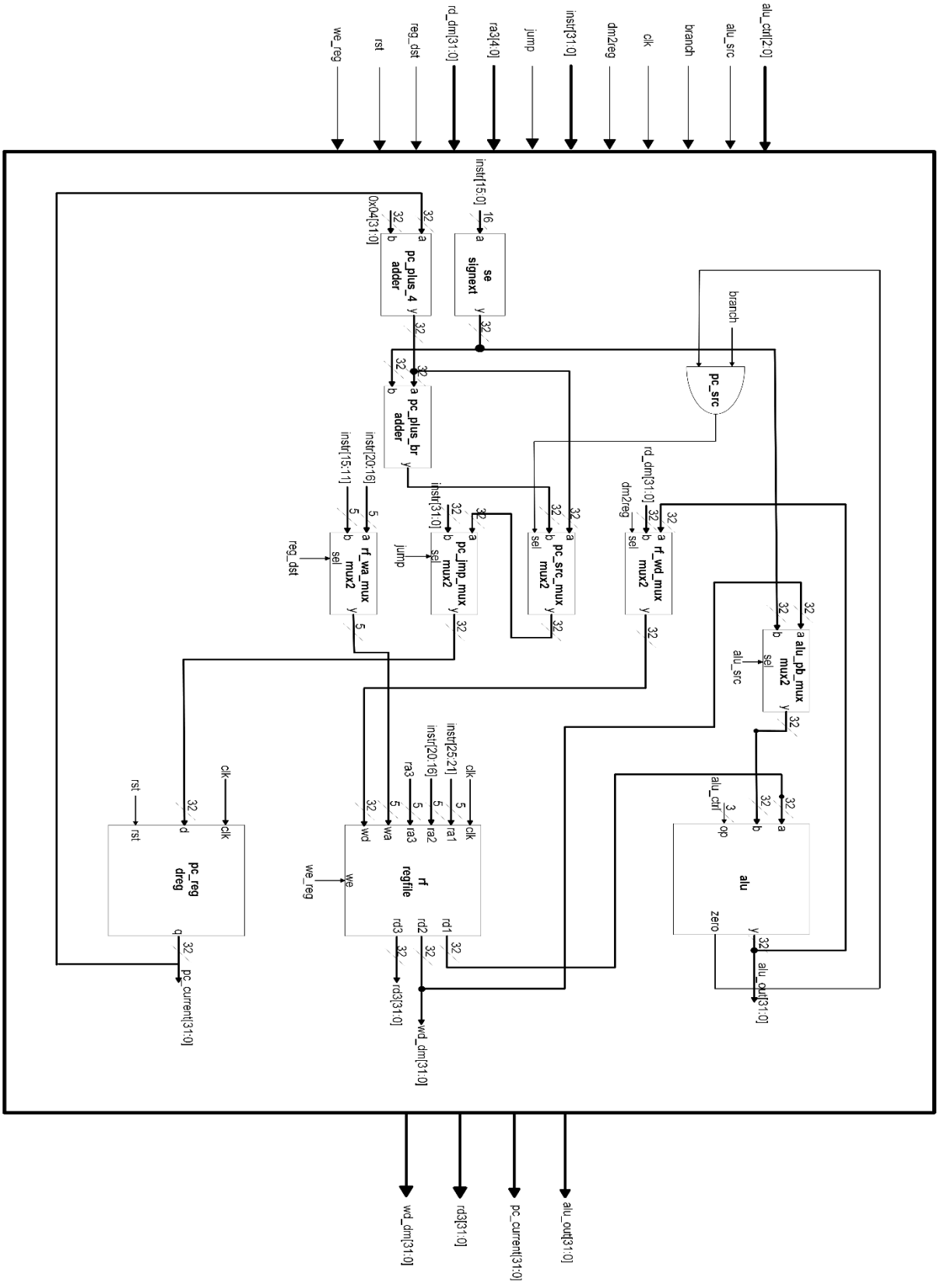

Fig. 1: MIPS Single Cycle CPU Data path

Control Unit

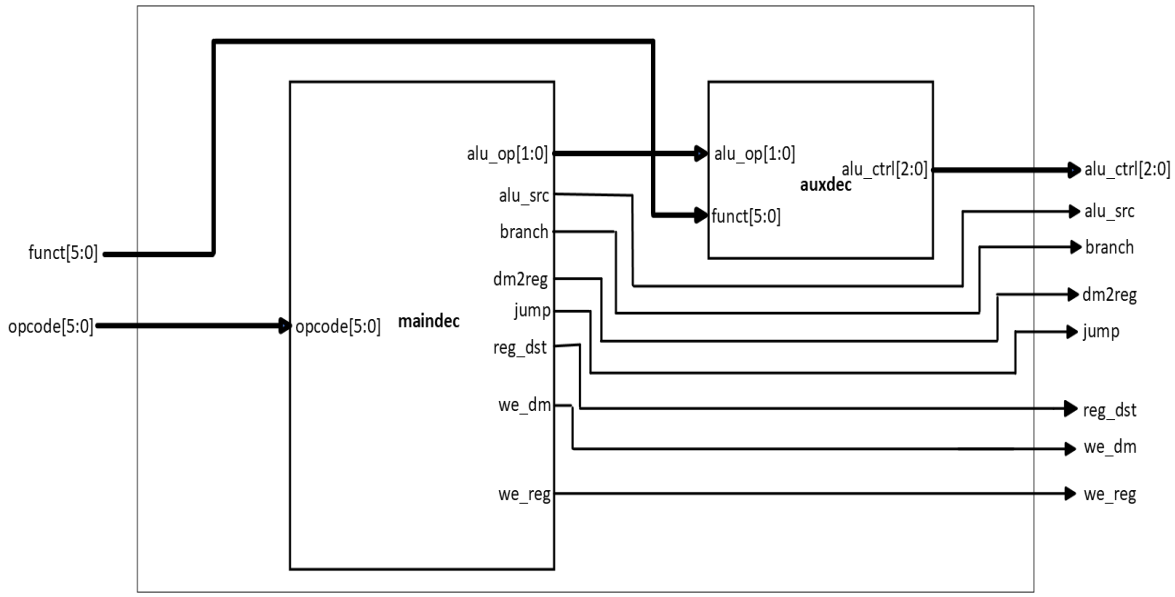

Fig. 2: Control Unit

Instruction Memory

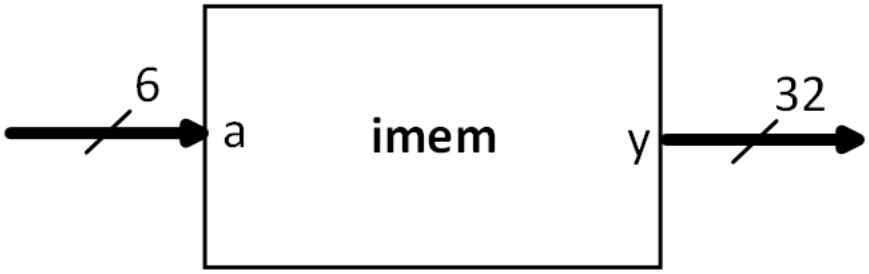

Fig. 3: Instruction Memory

Data Memory

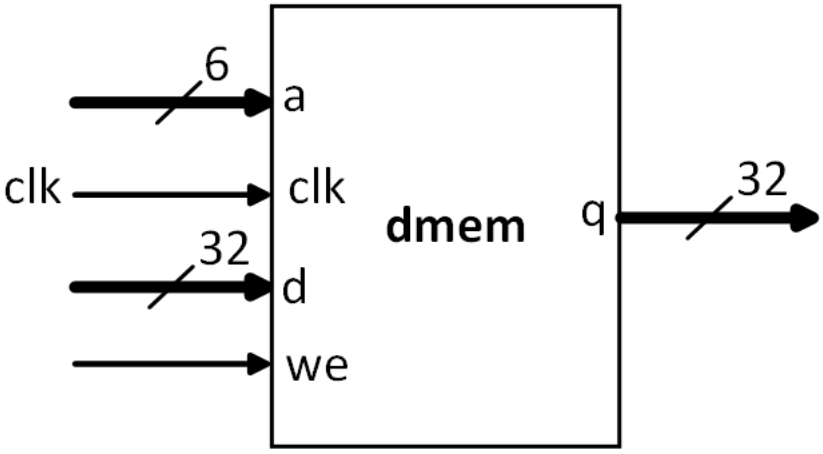

Fig. 4: Data Memory

Processor Core:

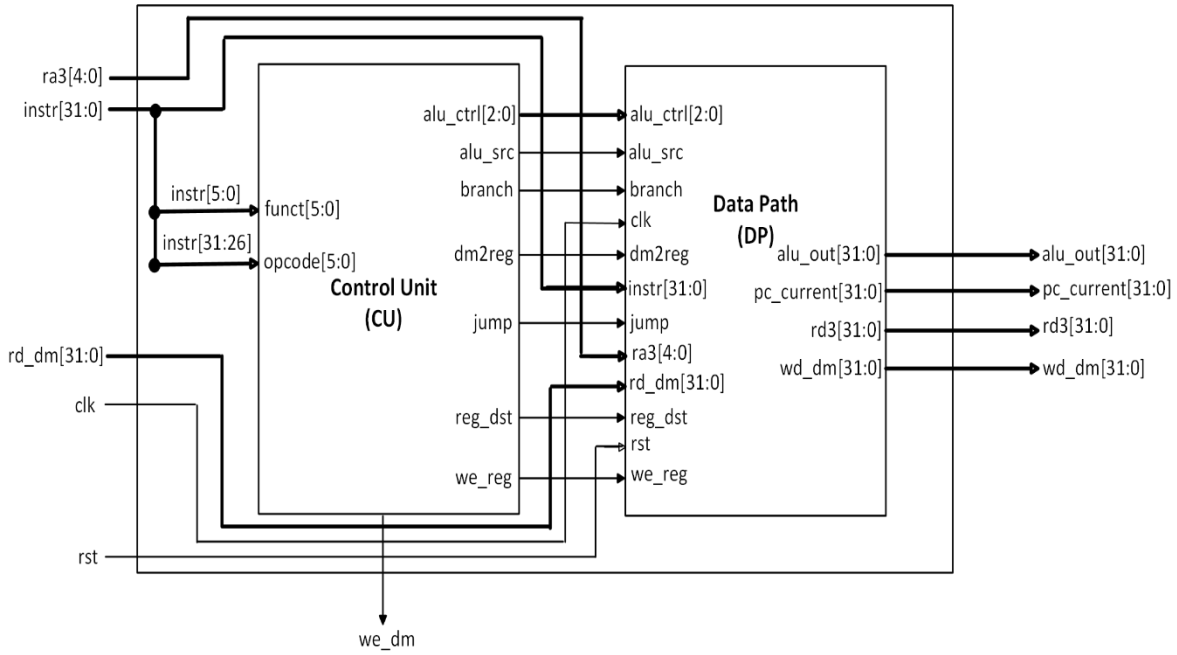

Fig. 5: Processor Core

Complete Processor:

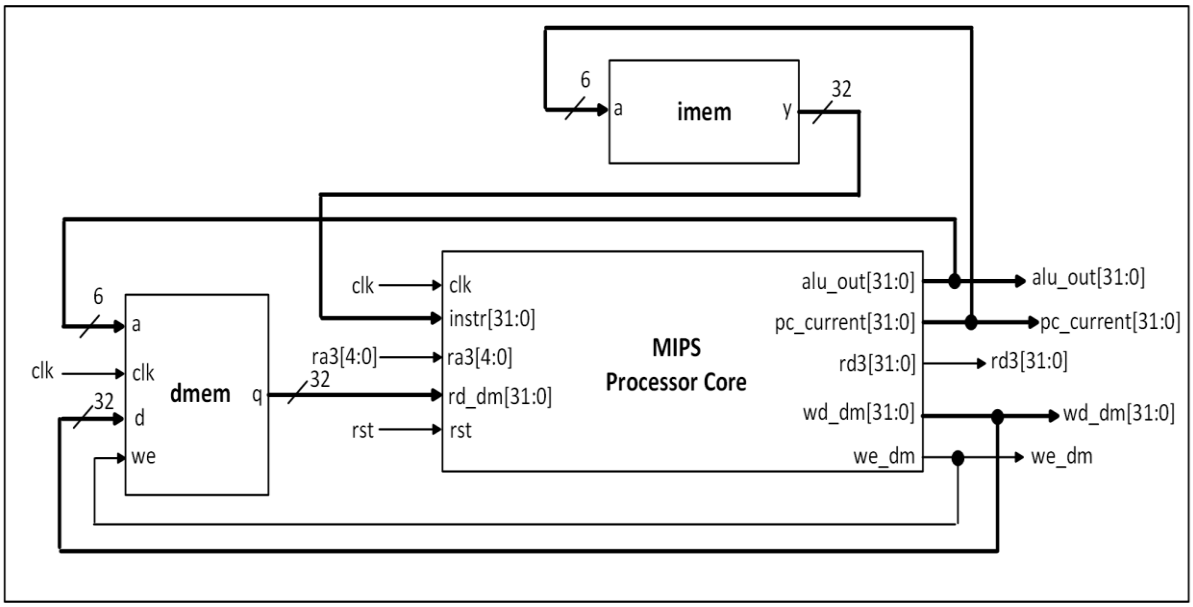

Fig. 6: Complete Processor

Modules:

- **Signext**

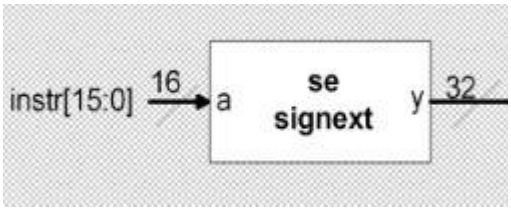

Fig. 7: Signext

- **Adder**

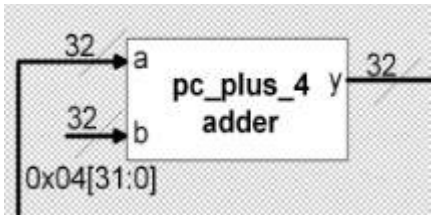

Fig. 8: Adder

- **AND gate**

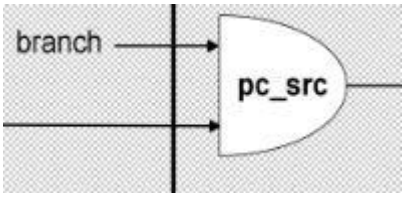

Fig. 9: AND gate

- **Multiplexer**

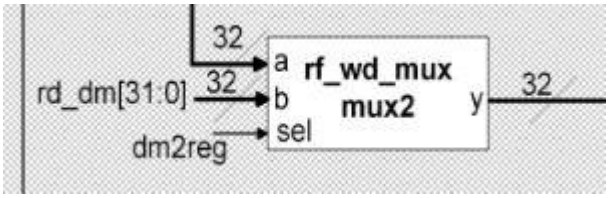

Fig. 10: Multiplexer

- **ALU**

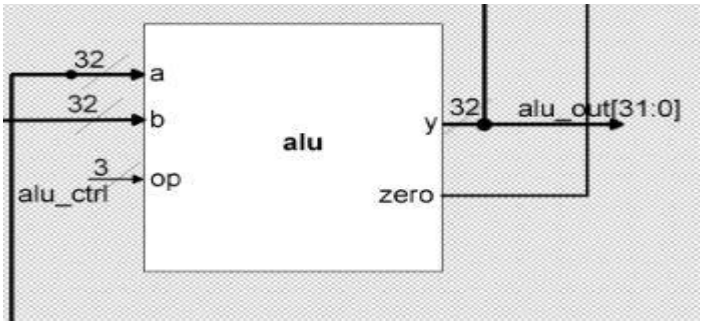

Fig. 11: ALU

- **regfile**

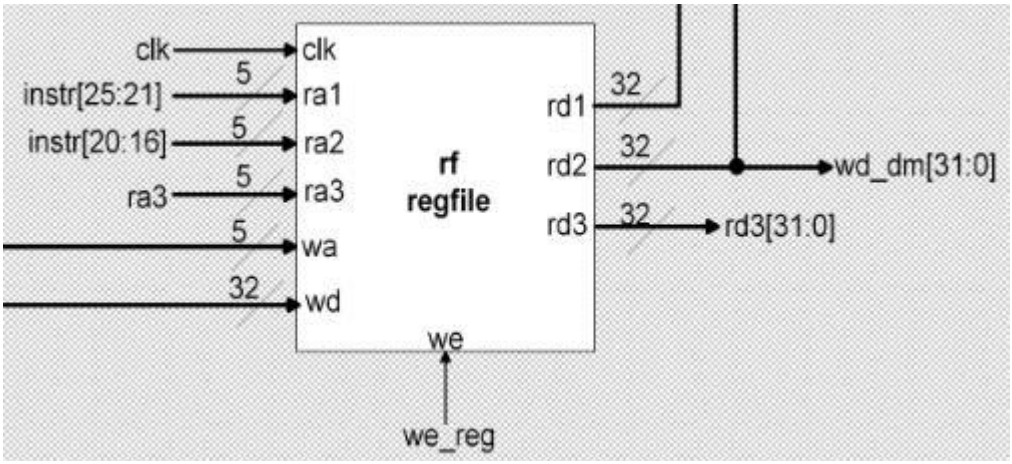

Fig. 12: regfile

- **dreg**

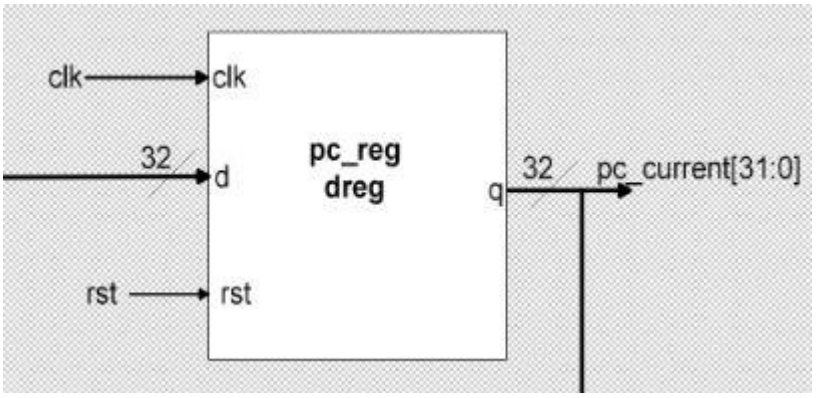

Fig. 13: pc_reg dreg

SIMULATION WAVEFORMS:

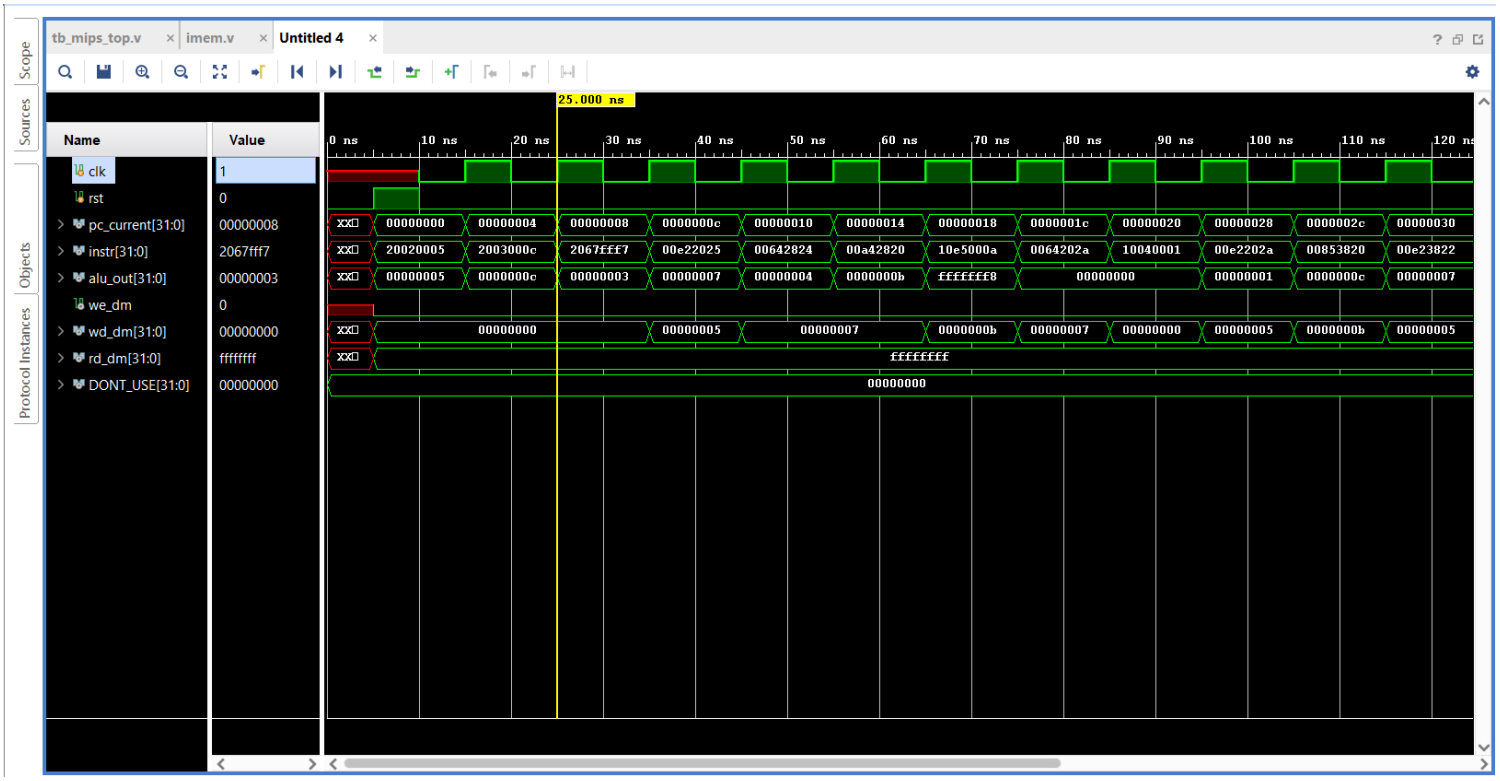

Fig. 14: Waveform after verifying the single cycle MIPS processor functionality

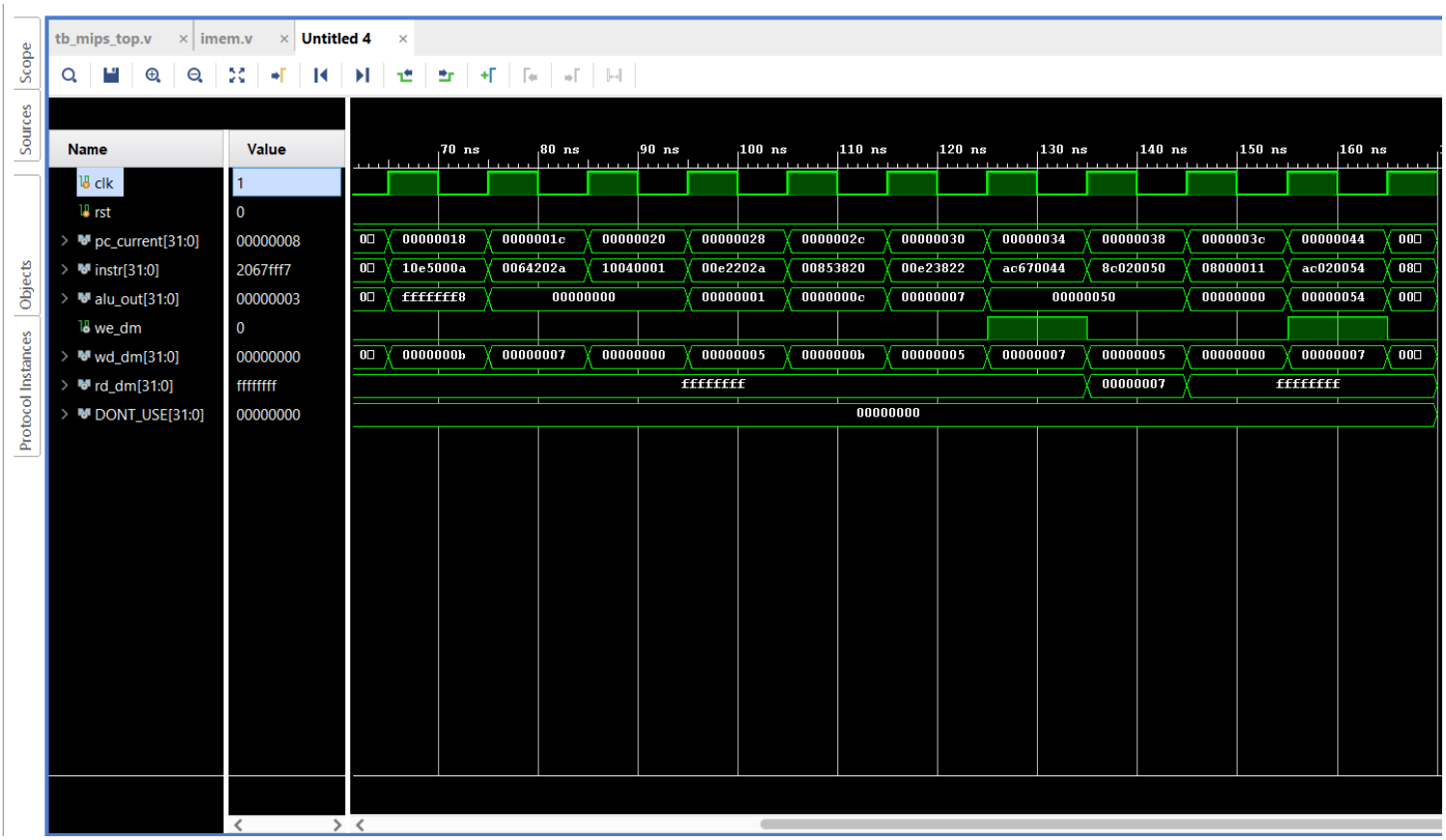

Fig. 15: Waveform after verifying the single cycle MIPS processor functionality