CSE 664 Introduction to System-on-Chip Design
Final Project – 8-Bit Microcontroller

Revision: 1.0

Date Prepared: June 2023

Prepared by: Austin Benedetti, Daniel Ems, Nicholas Lyons, Mike Peña, Trent Reeves, Mimansa Verma, Derrick Ward, Jiawen Zhen

Prepared for: CSE 664 Introduction to System-on-Chip Design, Spring 2023, section M401, Professor Mohammed Abdallah

Contents

1 Design Documentation 4
1.1 Entity: acc 4
1.2 Entity: ALU_8bit 6
1.3 Entity: controller_fsm 8
1.4 Entity: instruction_memory 10
1.5 Entity: instruction_register 11
1.6 Entity: mcu_8bit 13
1.7 Entity: nbit_two_one_mux 16
1.8 Entity: program_counter 17
1.9 Entity: register_unit 19
2 Simulations 22
2.1 Modelsim - accumulator.v 22
2.2 Modelsim – ALU_8bit.v 22
2.3 Modelsim – controller_fsm.v 23
2.4 Modelsim – instruction_memory.v 24
2.5 Modelsim – instruction_register.v 24
2.6 Modelsim – mcu_8bit_top_level.v 25
2.7 Modelsim – nbit_two_one_mux.v 26
2.8 Modelsim – program_counter.v 26
2.9 Modelsim – 16 slot 8-bit register unit 26
3 References 27
APPENDIX A: 8-Bit Microcontroller Code Base 27

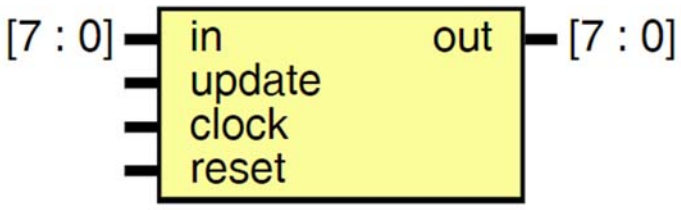
THIS PAGE INTENTIONALLY LEFT BLANK

1 Design Documentation

1.1 Entity: acc

- **File**: accumulator.v

1.1.1 Diagram



1.1.2 Description

A register in which intermediate arithmetic logic unit results are stored. This simple 8-bit accumulator's output will continuously grow by increments of the input until reset.

Note - Overflow is not handled

1.1.3 Ports

Port name	Direction	Type	Description
out	output	[7 : 0]	Value stored in the register, to be returned to ALU.
in	input	[7 : 0]	Value (from ALU) to increment current register value by.
update	input		Signal (active high) to update the register value by "in"'s value.
clock	input		Clock signal and posedge triggered.
reset	input		Signal (active high) to clear register value and set value stored back to 0.

1.1.4 Signals

Name	Type	Description
accumulator	reg [7 : 0]	

1.1.5 Processes

- unnamed: (@(posedge clock or posedge reset))

  Type: always

1.2 Entity: ALU_8bit

- **File**: ALU_8bit.v

1.2.1 Diagram

1.2.2 Description

A combinational digital circuit that performs arithmetic and bitwise operations on integer binary numbers.

1.2.3 Ports

Port name	Direction	Type	Description
alu_out	output	[7:0]	8 bit result (connected to mux that is connected to ACC)
alu_zero_flag	output		Set if result is zero (connected to controller)
alu_carry_out	output		Set if carry bit is needed (connected to controller)
alu_select	input	[3:0]	Select ALU operation (received from controller)
alu_a_in	input	[7:0]	ACC ALU input
alu_b_in	input	[7:0]	REG ALU input

1.2.4 Processes

- unnamed: (@(alu_a_in or alu_b_in or alu_select))

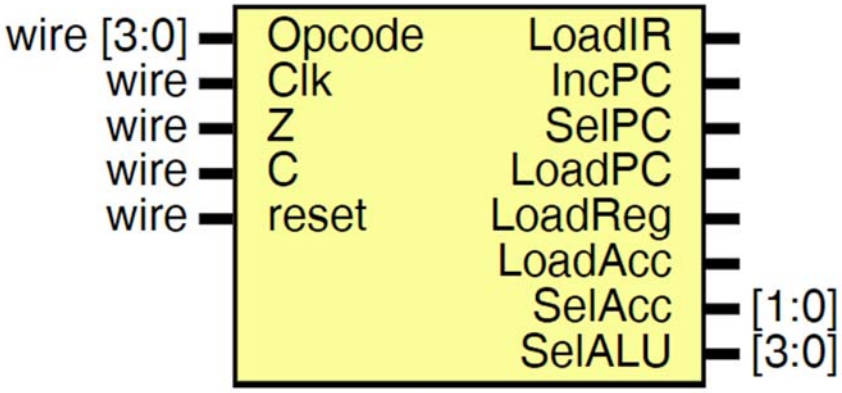
Type: always

Description
The ALU outputs change anytime the inputs change (combinational logic).

1.3 Entity: controller_fsm

- **File**: controller_fsm.v

1.3.1 Diagram



1.3.2 Description

Model that uses a known set of inputs to define several combinations of outputs and states.

1.3.3 Ports

Port name	Direction	Type	Description
LoadIR	output		Load instruction register with next instruction - (Should be checked before update)
IncPC	output		Program counter which is incremented only to next instruction
SelPC	output		Used to increment PC by immediate or val in reg
LoadPC	output		Signal to update PC value - (Should be checked before JUMP ONLY)
LoadReg	output		Signal to update register - (Should be checked before update)

Port name	Direction	Type	Description
LoadAcc	output		Signal to update accumulator - (Should be checked before update)
SelAcc	output	[1:0]	Select signal for ACC muxes - (SelAcc[1] = SelAcc1, SelAcc[0] = SelAcc0)
SelALU	output	[3:0]	Select signal for ALU operation (opcode)
Opcode	input	wire [3:0]	Opcode from instruction register
Clk	input	wire	Clock signal
Z	input	wire	Zero bit
C	input	wire	Carry bit
reset	input	wire	Asynchronous active high reset

1.3.4 Processes

- unnamed: (@(posedge Clk or posedge reset))

  Type: always

  Description
  Case statement for setting control signals

1.4 Entity: instruction_memory

- **File**: instruction_memory.v

1.4.1 Diagram

1.4.2 Description

Instruction memory to execute a program based on a program counter Assumption:

Assumption: Program is precompiled

1.4.3 Ports

Port name	Direction	Type	Description
pc_address	input	[7:0]	Address of the Instruction
out	output	[7:0]	Instruction value at the address location

1.4.4 Processes

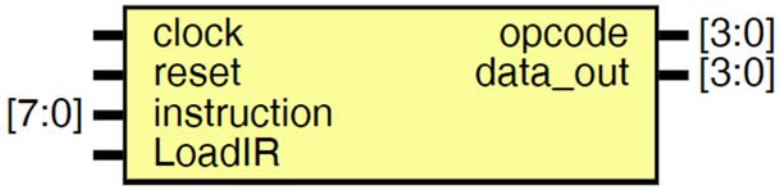
- unnamed: (@(pc_address))

Type: always

1.5 Entity: instruction_register

- **File**: instruction_register.v

1.5.1 Diagram



1.5.2 Description

The part of a CPU's control unit that holds the instruction currently being executed or decoded.

This module transforms instruction into opcode (sent to controller) and data components.

1.5.3 Ports

Port name	Direction	Type	Description
clock	input		Clock signal (Positive edge triggered).
reset	input		Signal to clear and set opcode and data out back to 0.
instruction	input	[7:0]	Instruction received from memory.
opcode	output	[3:0]	Opcode from the instruction.
data_out	output	[3:0]	Data component from the instruction.
LoadIR	input		Signal to load and transform instruction into opcode and data components.

1.5.4 Processes

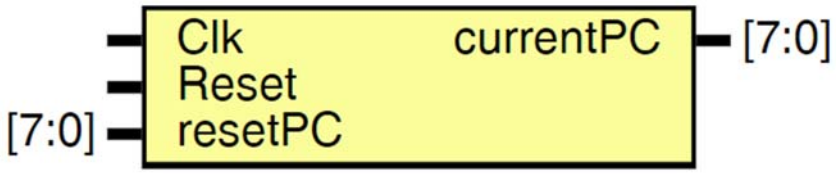
- unnamed: (@(posedge clock or posedge reset))

  Type: always

1.6 Entity: mcu_8bit

- **File**: mcu_8bit_top_level.v

1.6.1 Diagram



1.6.2 Description

8-bit Micro-controller Unit

1.6.3 Ports

Port name	Direction	Type	Description
Clk	input		Clock signal.
Reset	input		Signal to reset controller (all internal components).
currentPC	output	[7:0]	Program Counter: Address of the program instruction we want to execute.
resetPC	input	[7:0]	Signal to reset program counter.

1.6.4 Signals

Name	Type	Description
LoadIR	wire	

Name	Type	Description
IncPC	wire	
SelPC	wire	
LoadPC	wire	
LoadReg	wire	
LoadAcc	wire	
SelAcc	wire [1:0]	
SelALU	wire [3:0]	
alu_zero_flag	wire	
alu_carry_out	wire	
alu_out	wire [7:0]	
reg_out	wire [7:0]	
acc_out	wire [7:0]	
acc_in	wire [7:0]	
a0_to_a1	wire [7:0]	
pc_jump	wire [7:0]	
ir_data	wire [3:0]	

Name	Type	Description
Opcode	wire [3:0]	
instruction	wire [7:0]	

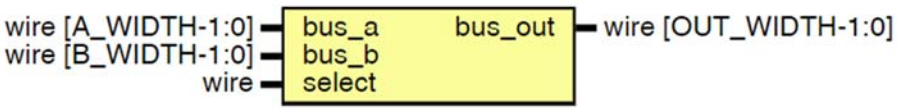
1.6.5 Instantiations

- imm: instruction_memory
- instr_reg: instruction_register
- MUX2_PC: nbit_two_one_mux
- pc_module: program_counter
- controller_fsm: controller_fsm
- register_file: register_unit
- ALU: ALU_8bit
- MUX2_A0: nbit_two_one_mux
- MUX2_A1: nbit_two_one_mux
- accumulator: acc

1.7 Entity: nbit_two_one_mux

- **File**: nbit_two_one_mux.v

1.7.1 Diagram



1.7.2 Description

A data selector device that selects between several analog or digital input signals and forwards the selected input to a single output line.

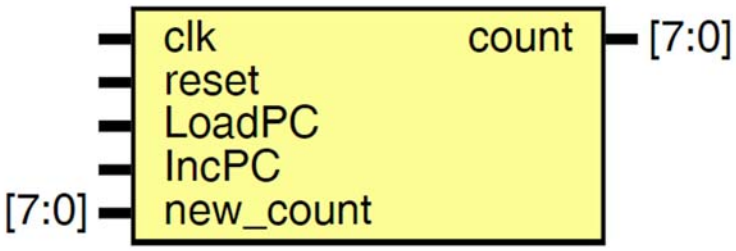
1.7.3 Ports

Port name	Direction	Type	Description
bus_out	output	wire [OUT_WIDTH-1:0]	Output bus
bus_a	input	wire [A_WIDTH-1:0]	Input bus A
bus_b	input	wire [B_WIDTH-1:0]	Input bus B
select	input	wire	Select Signal

1.8 Entity: program_counter

- **File**: program_counter.v

1.8.1 Diagram



1.8.2 Description

Register that contains the address of the instruction to be executed. Counts the instructions opcodes from 0000 to 1111 and it signals the memory address of next instruction to be fetched and executed.

1.8.3 Ports

Port name	Direction	Type	Description
clk	input		cLock signal.
reset	input		Signal to clear and set address back to 8'b00000000.
LoadPC	input		Signal to manually set program address.
IncPC	input		Signal to automatically increment program address by 1 (2'b10)
new_count	input	[7:0]	Address to manually set program address to.
count	output	[7:0]	Current address of the program

1.8.4 Processes

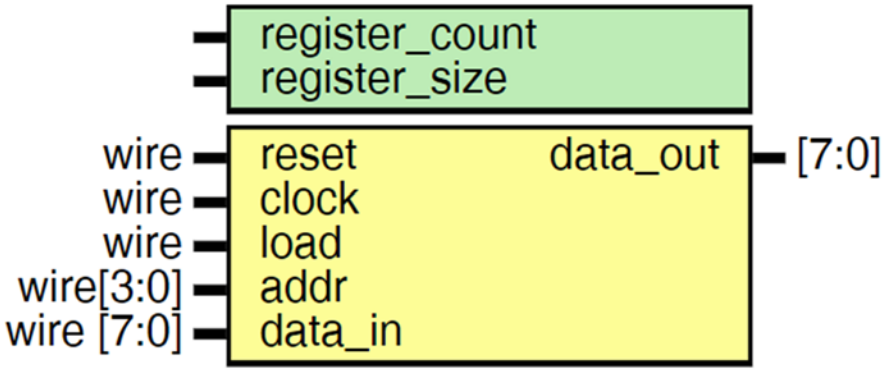
- unnamed: (@ (posedge clk or posedge reset))

  Type: always

1.9 Entity: register_unit

- **File**: registers.v

1.9.1 Diagram



1.9.2 Description

A register module that allows users to store up to 16 numbers that are 8-bits long.

1.9.3 Generics

Generic name	Type	Value	Description
register_count		16	
register_size		8	

1.9.4 Ports

Port name	Direction	Type	Description
reset	input	wire	Signal to clear all 16 register values and set value stored back to 0.

Port name	Direction	Type	Description
clock	input	wire	Clock signal.
load	input	wire	Signal to update register.
addr	input	wire[3:0]	Address of the register slot to update or read from.
data_out	output	[7:0]	Value stored in register slot, addressed by the "addr" 4-bit wires.
data_in	input	wire [7:0]	Value to update register slot to, addressed by the "addr" 4-bit wires. Requires load and clock signal to be high.

1.9.5 Signals

Name	Type	Description
i	integer	

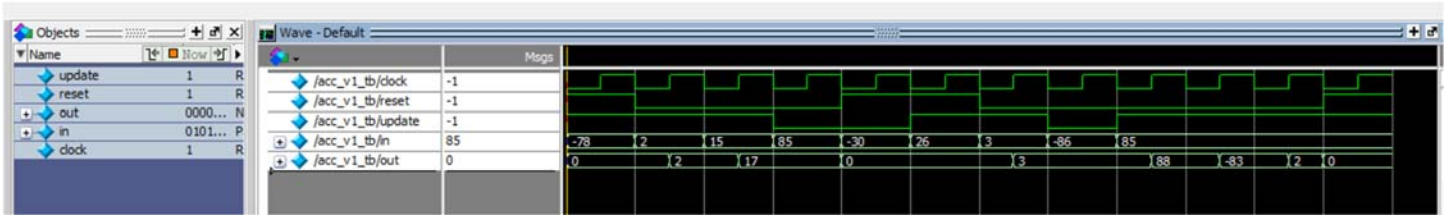
1.9.6 Processes

- unnamed: (@(posedge clock or posedge reset))

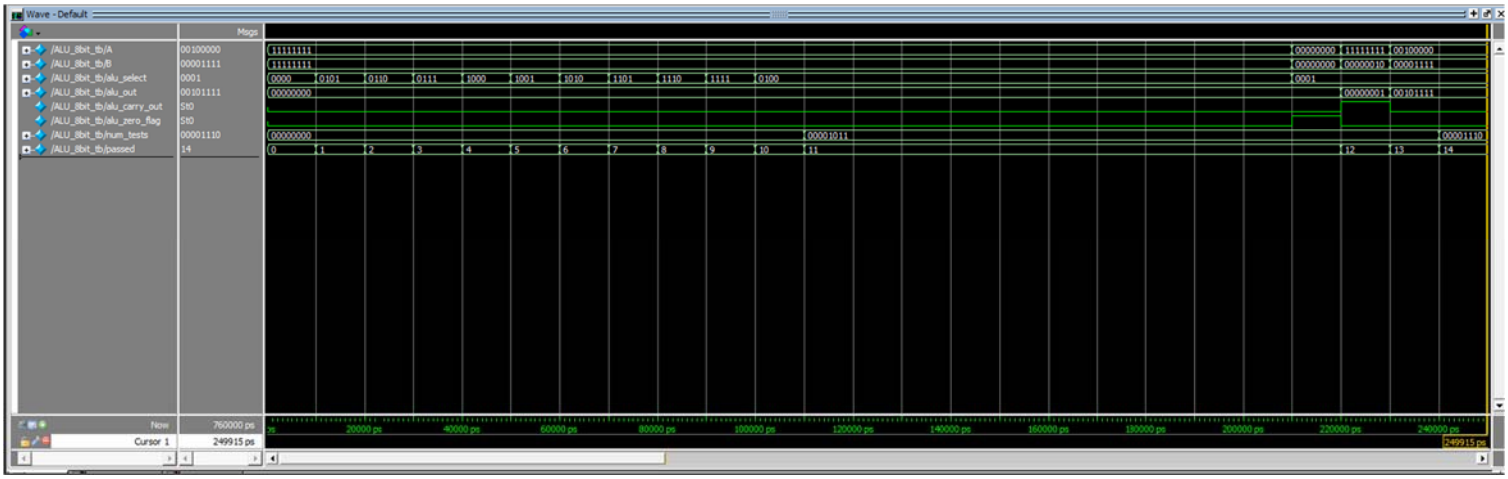
  Type: always

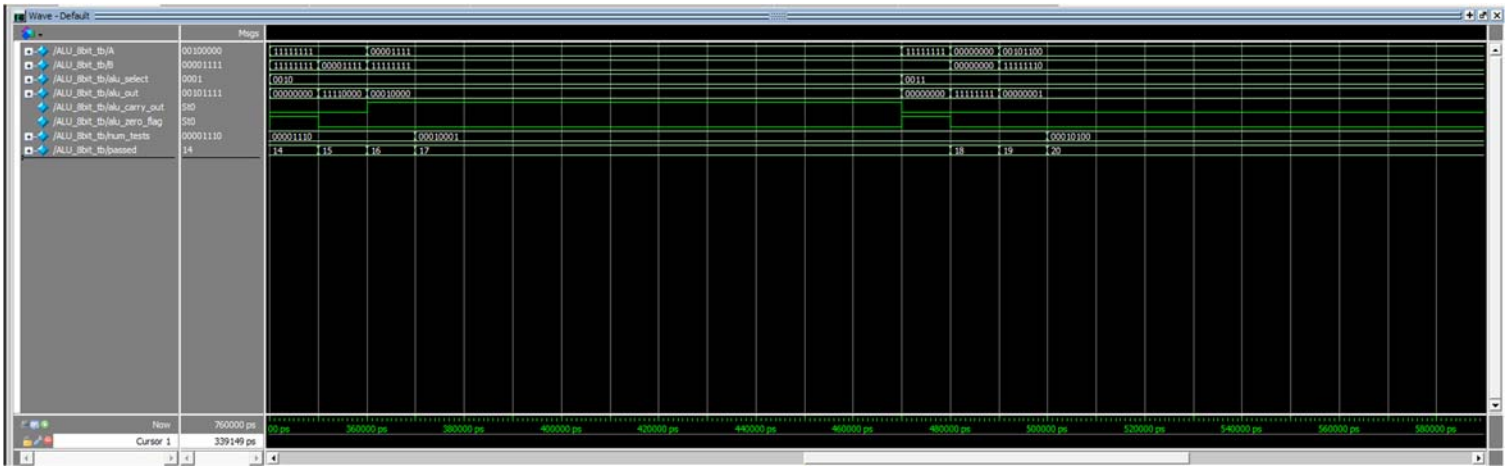
2 Simulations

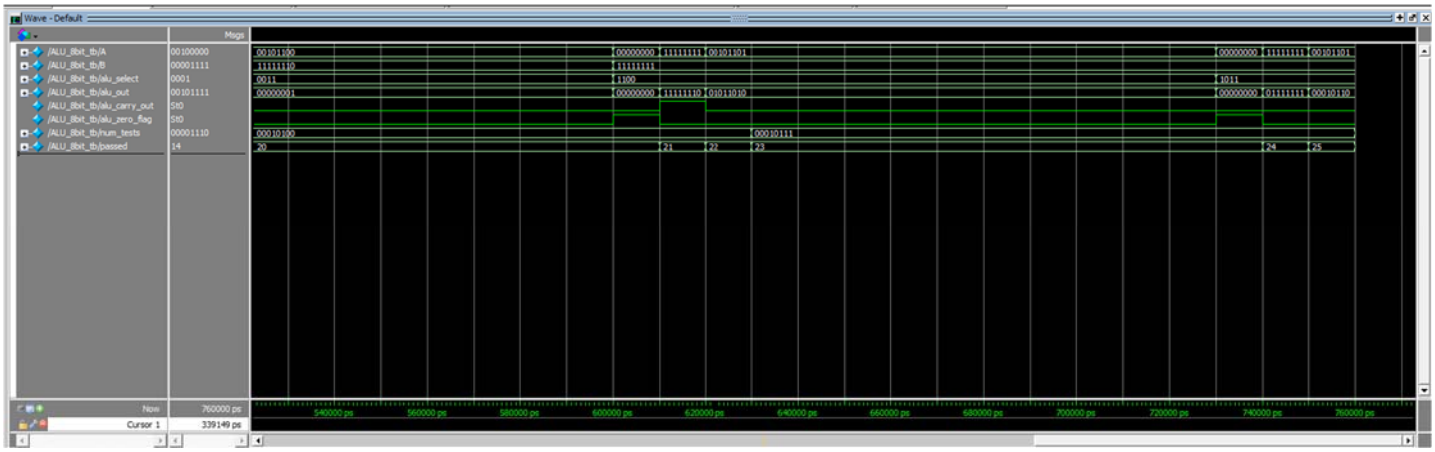
2.1 Modelsim - accumulator.v



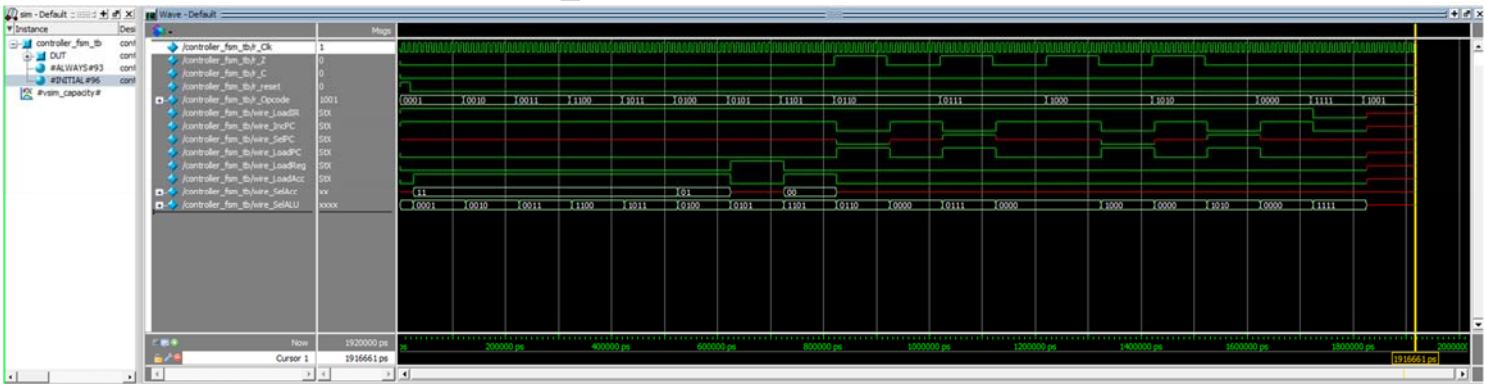
2.2 Modelsim – ALU_8bit.v



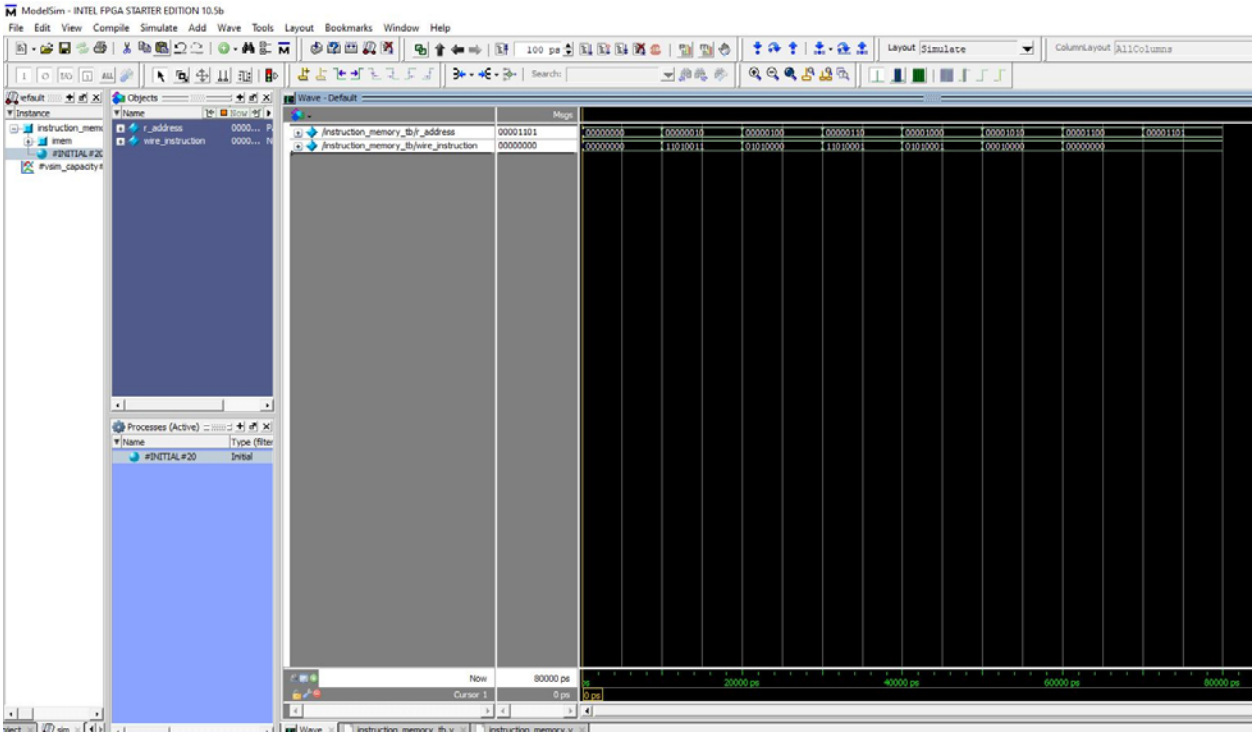




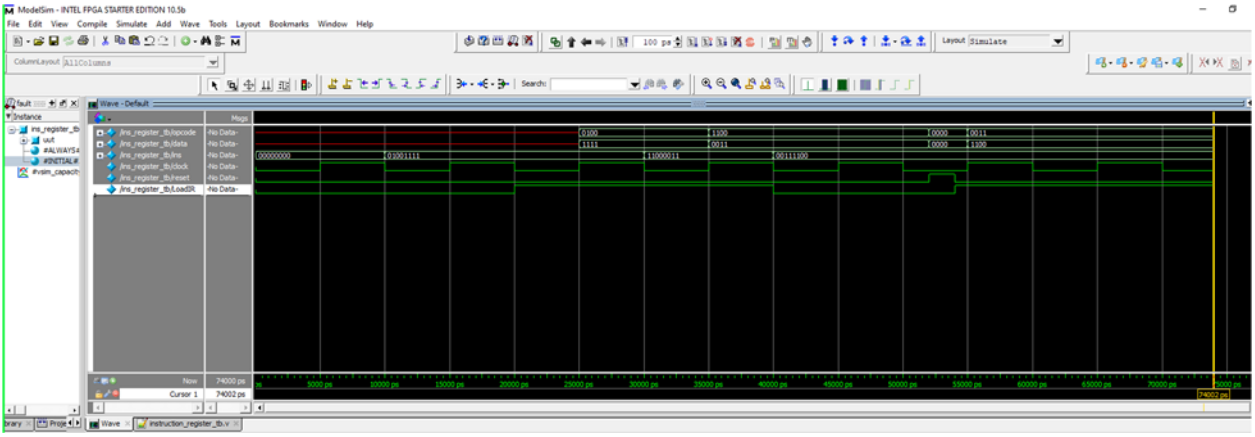
2.3 Modelsim – controller_fsm.v



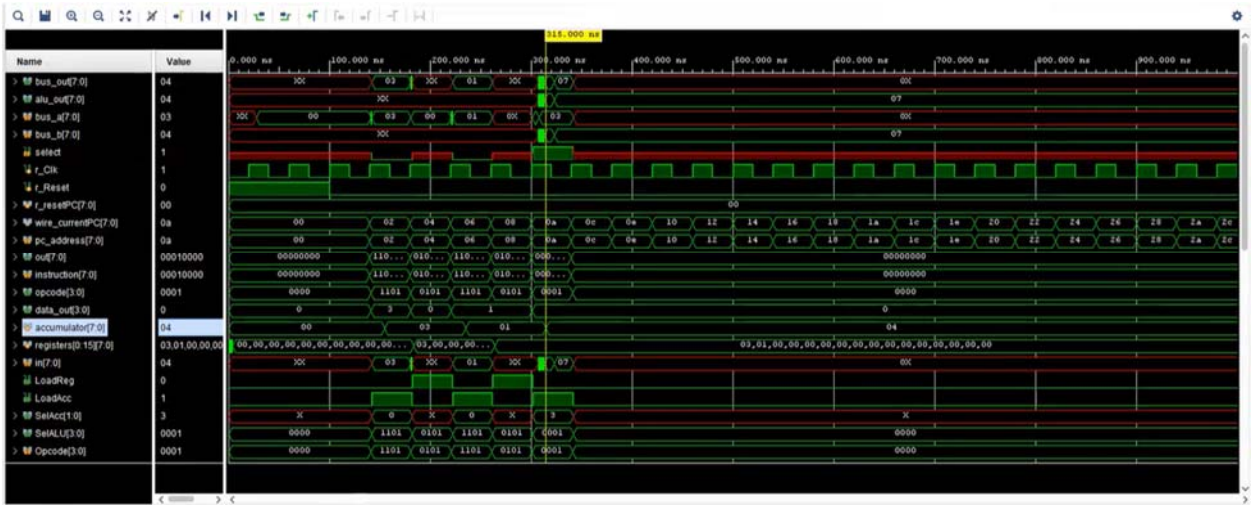
2.4 Modelsim – instruction_memory.v

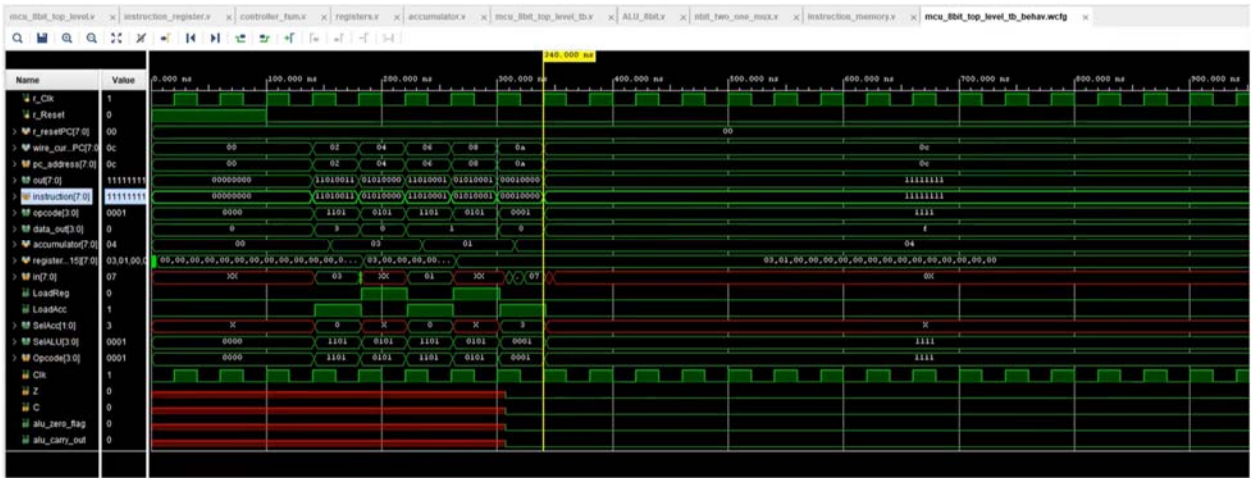


2.5 Modelsim – instruction_register.v

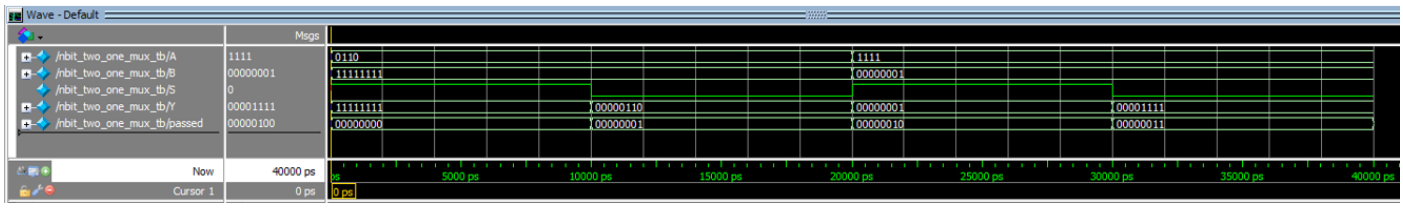


2.6 Modelsim – mcu_8bit_top_level.v

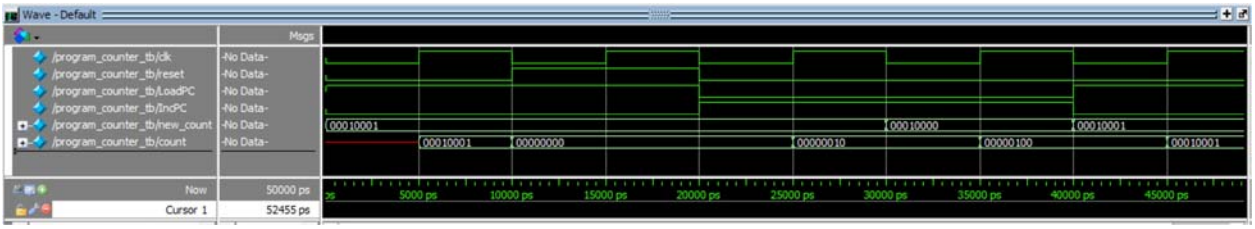




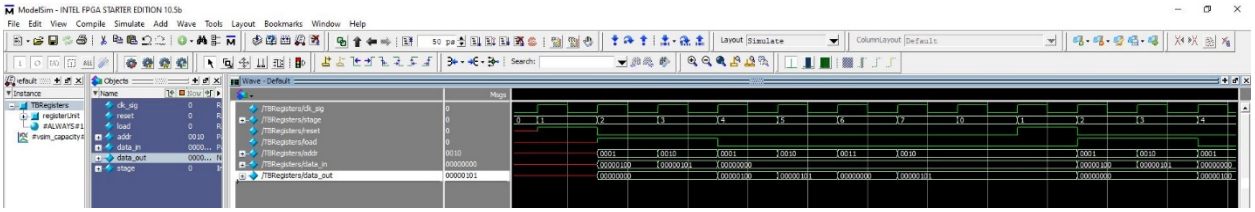
2.7 Modelsim – nbit_two_one_mux.v



2.8 Modelsim – program_counter.v



2.9 Modelsim – 16 slot 8-bit register unit



3 References

APPENDIX A:

8-Bit Microcontroller Code Base

Prepared by:

Austin Benedetti, Daniel Ems, Nicholas Lyons, Mike Peña, Trent Reeves, Mimansa Verma, Derrick Ward, Jiawen Zhen