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 Pena, 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

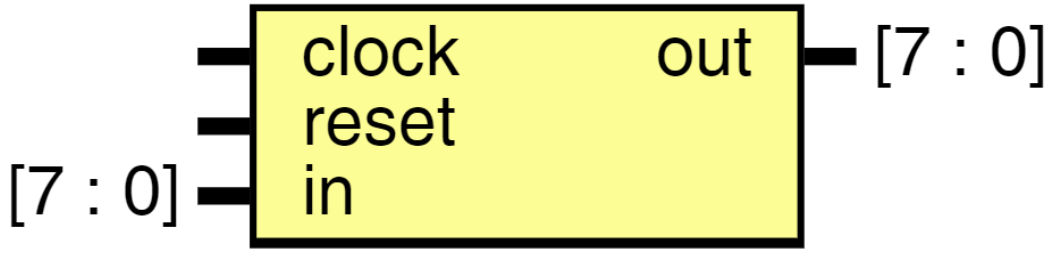
THIS PAGE INTENTIONALLY LEFT BLANK

1 Design Documentation

1.1 Entity: acc_simple

- **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
clock	input		Clock. Output is posedge triggered
reset	input		Clears register value (1 = clear, 0 = hold)
in	input	[7 : 0]	Input from ALU
out	output	[7 : 0]	Output return to ALU

1.1.4 Signals

Name	Type	Description
accumulator	reg [7 : 0]	Temporary storage for accumulator output

1.1.5 Processes

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

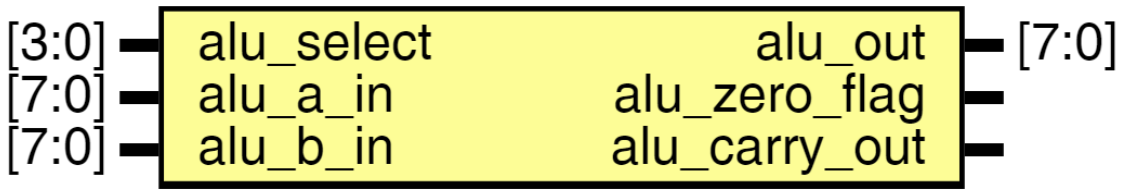
Type: always

Description
Increments output by input at rising edge of clock.
If reset is high then clear output.

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))

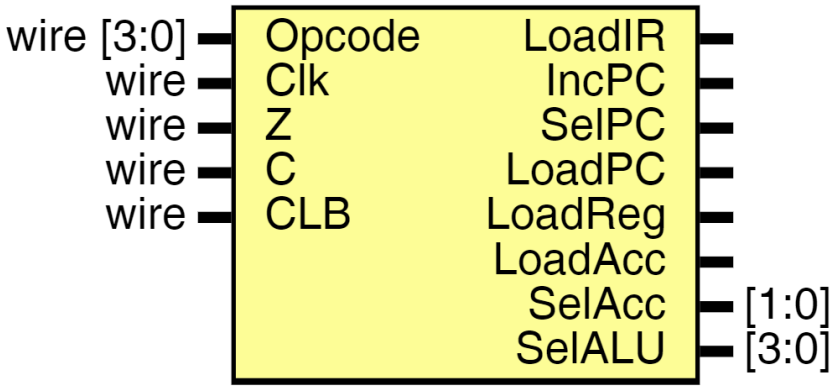
Type: always

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

2 Entity: controller_fsm

- **File**: controller_fsm.v

2.1 Diagram



2.2 Description

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

2.3 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)

Port name	Direction	Type	Description
LoadReg	output		Signal to update register - (Should be checked before update)
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
CLB	input	wire	TODO: WHAT IS THIS?

2.4 Processes

- unnamed: (@(Clk))

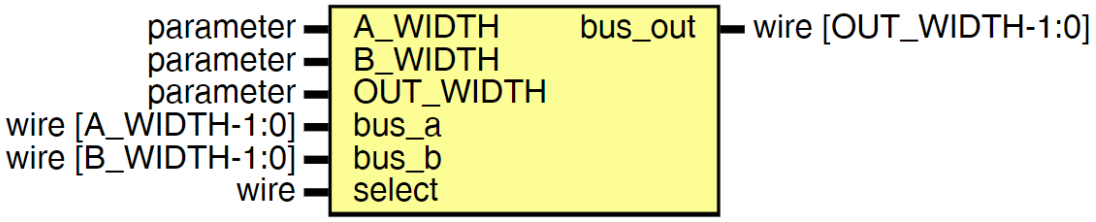
Type: always

Description
Case statement for setting control signals

3 Entity: nbit_two_one_mux

- **File**: nbit_two_one_mux.v

3.1 Diagram



3.2 Description

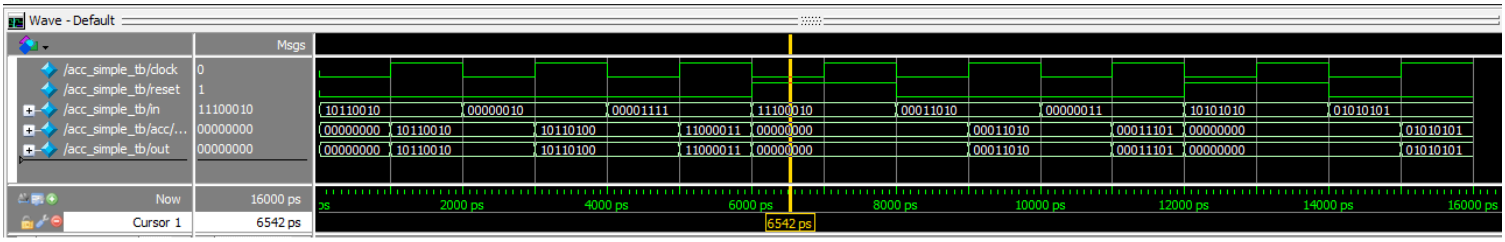
A data selector device that selects between several analog or digital input signals and forwards the selected input to a single output line.

3.3 Ports

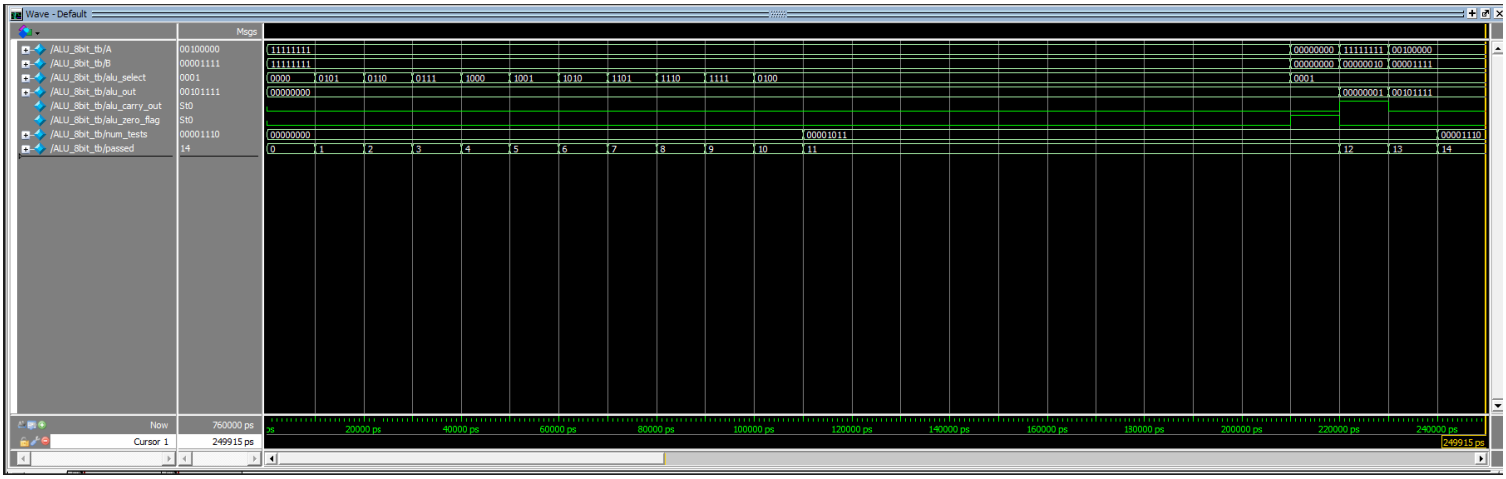
Port name	Direction	Type	Description
A_WIDTH	input	parameter	
B_WIDTH	input	parameter	Width of INPUT BUS A
OUT_WIDTH	input	parameter	Width of INPUT BUS B
bus_out	output	wire [OUT_WIDTH-1:0]	Output bus
bus_a	input	wire [A_WIDTH-1:0]	Input bus A
bus_b	input	wire [B_WIDTH-1:0]	Input bus B
select	input	wire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

4 Simulations

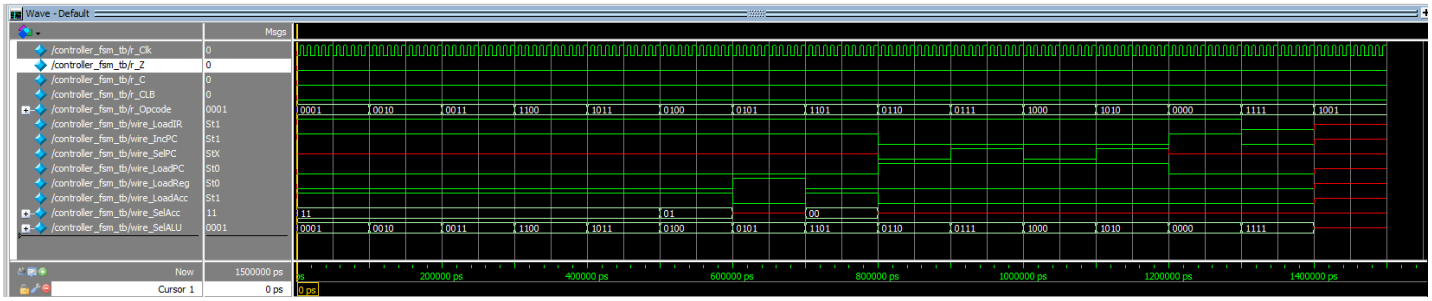
4.1 Modelsim - accumulator.v



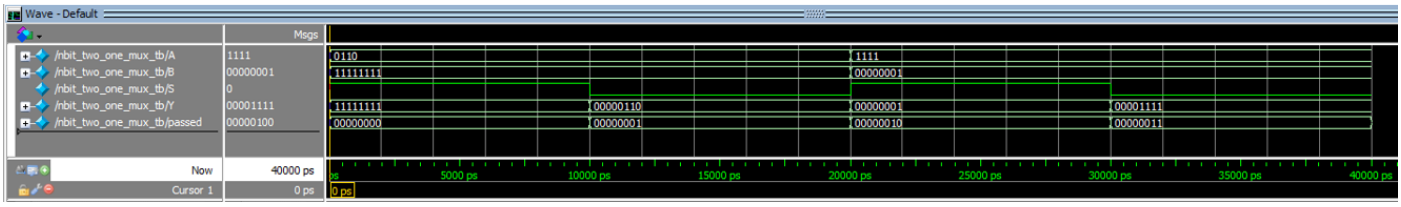
4.2 Modelsim – ALU_8bit.v



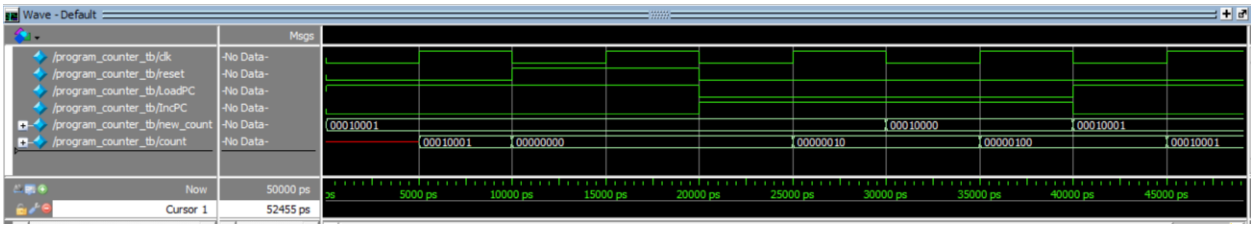
4.3 Modelsim – controller_fsm.v



4.4 Modelsim – nbit_two_one_mux.v



4.5 Modelsim – program_counter.v



5 References

APPENDIX A:

8-Bit Microcontroller Code Base

Prepared by:

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