Hrutvik Patel

COE328

LAB 6 GPU

ID: 500758680

Section 09

Dec 2 2017

TABLE OF CONTENTS

2. Introduction

For this lab our task was to build a GPU (General processing unit). We were assigned 3 problems for our gpu to tackle using our ALUs. I was assigned problem set c for part 2 and problem set a for part 3. The GPU is composed of 4 main parts: the storage elements, control unit, ALU (Arithmetic Logic Unit) and SSEG (seven segment display).

The storage elements consist of 2 latches. Latches are level sensitive so when it is clocked, the circuit is active.

The control unit consists of a FSM and a 4x16 Decoder. It takes in 3 inputs: clock, reset and the data_in (input to the FSM). It outputs an 8 bit student id (output from the FSM), and a 16 bit selector operation (output from the decoder which took the current state output from the FSM as the input). The output of the decoder in the control unit goes as the input of the ALU.

The ALU performs arithmetic functions on 2 8-bit numbers. It consists of 4 inputs: 2 8 bit numbers from latches, clock and a 16 bit selector operations from the decoder. The output is an 8 bit number which is the result of the input 8 bit numbers.

Finally the SSEG outputs the result from the ALU onto a seven segment display.

3. Components: Latch1, Latch2, 4:16 Decoder, FSM

a) Description of the components

Latches: Temporary stores 2 8 bit inputs.

ALU: Performs arithmetic functions on 2 8-bit numbers.

4x16 Decoder: Outputs selector operations for ALU given inputs from the FSM's current state 4 bit number.

FSM: Using Mealy/Moore state machines (we used Moore) we cycle through the states (the digits of our student number) and produce a 4 bit current state output and an 8-bit student number output.

SSEG: Displays the result (an 8 bit number) from the ALU on the FPGA board using the seven-segment display.

b) Truth Tables

Latches:

clock	A	Qn+1 (next Q value)
0	x	Qn (previous Q value)
1	0	0
1	1	1

4x16 Decoder:

	INPUTS				OUTPUTS														
$\overline{E}$	A3	A2	A1	A0	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10	O11	O12	O13	O14	O15
1	X	X	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

FSM (using my student id: 500758680):

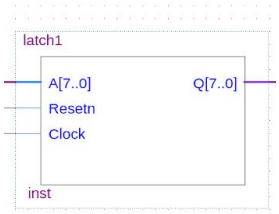
Present state	Next state w=0	Next state w=1	Output
0000	0000	0001	5, 0101
0001	0001	0010	0 , 0000
0010	0010	0011	0 , 0000
0011	0011	0100	7 , 0111
0100	0100	0101	5 , 0101
0101	0101	0110	8 , 1000
0110	0110	0111	6 , 0110
0111	0111	1000	8 , 1000
1000	1000	0000	0 , 0000

:**SSEG:**

Binary Inputs	Decoder Outputs	7 Segment Display Outputs
D C B A	a b c d e f g	
0 0 0 0	1 1 1 1 1 1 0	0
0 0 0 1	0 1 1 0 0 0 0	1
0 0 1 0	1 1 0 1 1 0 1	2
0 0 1 1	1 1 1 1 0 0 1	3
0 1 0 0	0 1 1 0 0 1 1	4
0 1 0 1	1 0 1 1 0 1 1	5
0 1 1 0	1 0 1 1 1 1 1	6
0 1 1 1	1 1 1 0 0 0 0	7
1 0 0 0	1 1 1 1 1 1 1	8
1 0 0 1	1 1 1 1 0 1 1	9

c) Circuit Diagram/Block Diagram of components.

Latches:



4x16 Decoder:

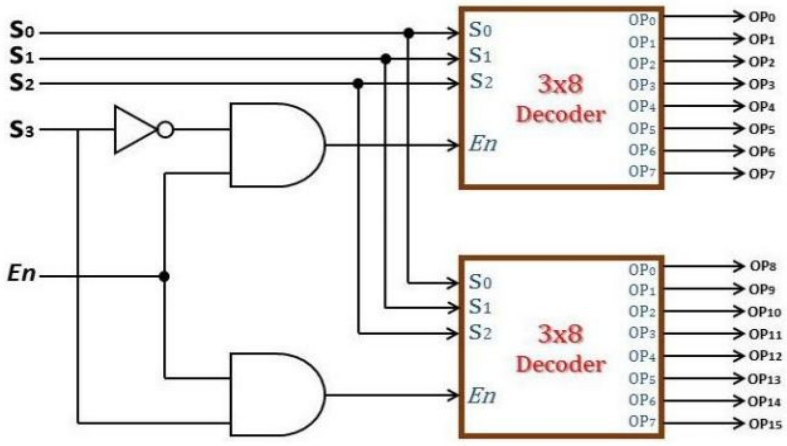
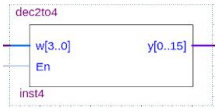
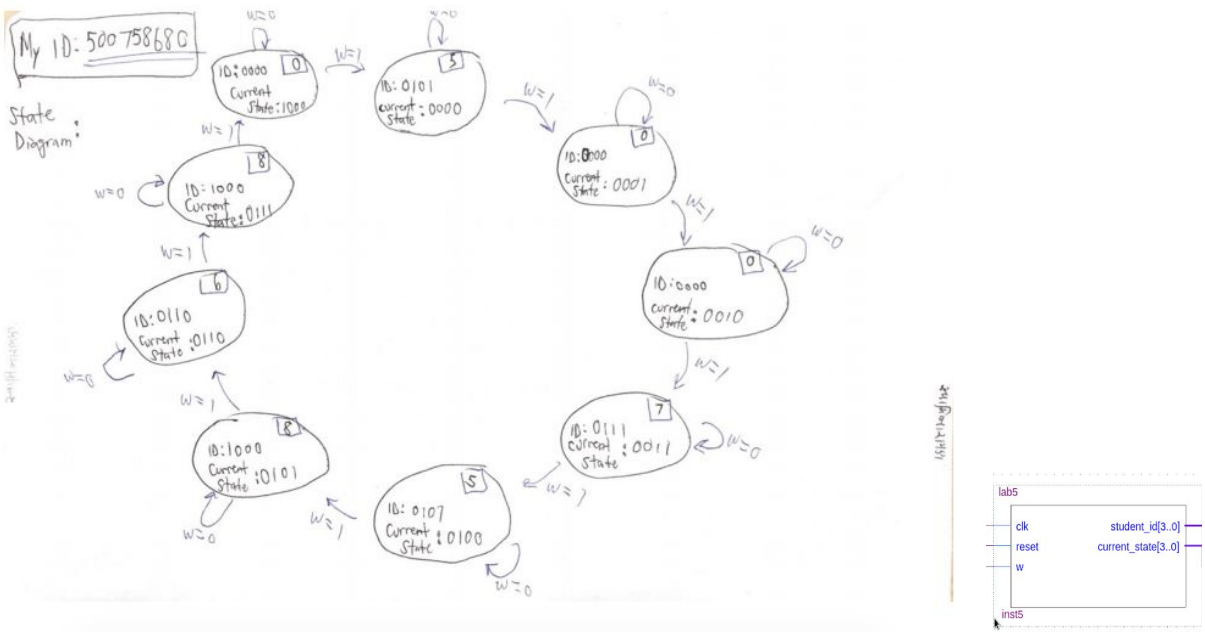


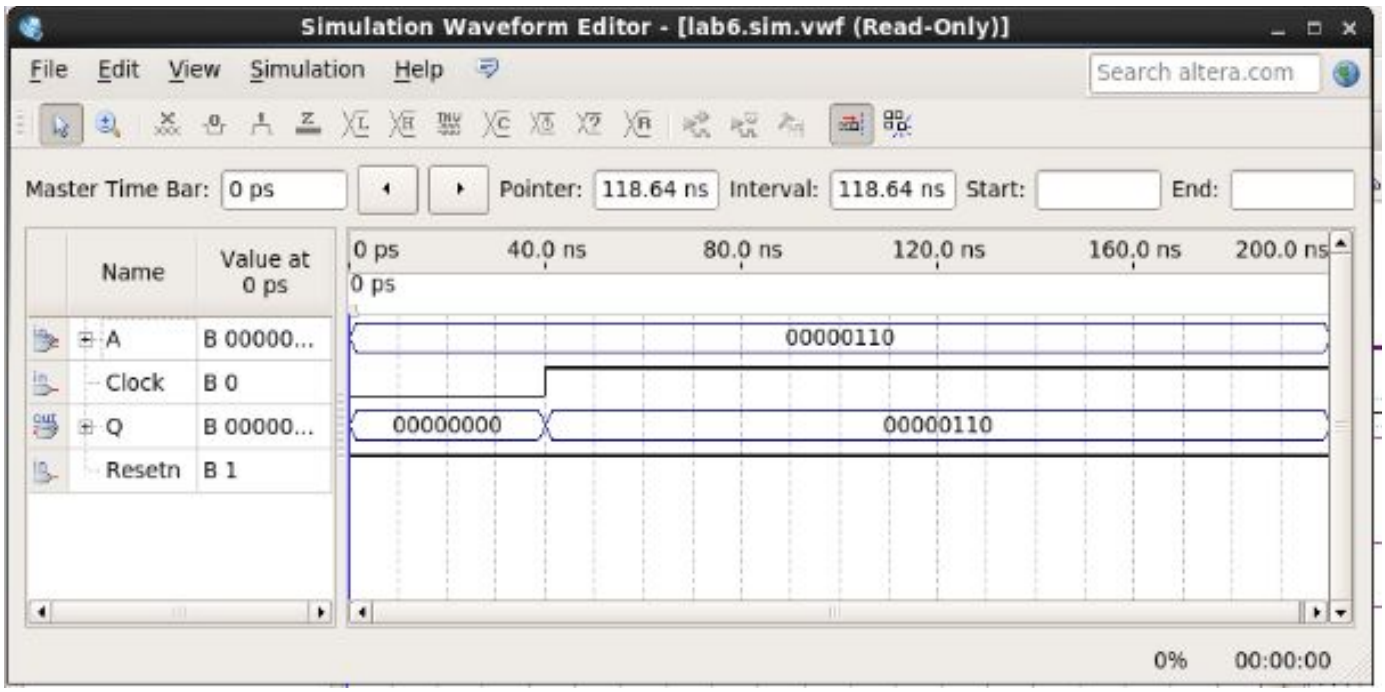
Figure 3. Implementation of 4x16 Decoder using 3x8 decoders



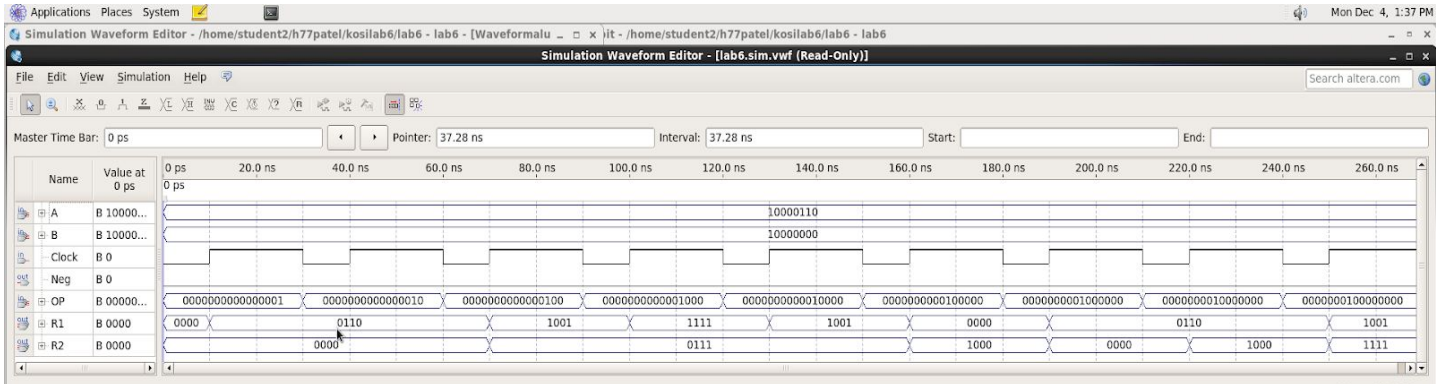
FSM:



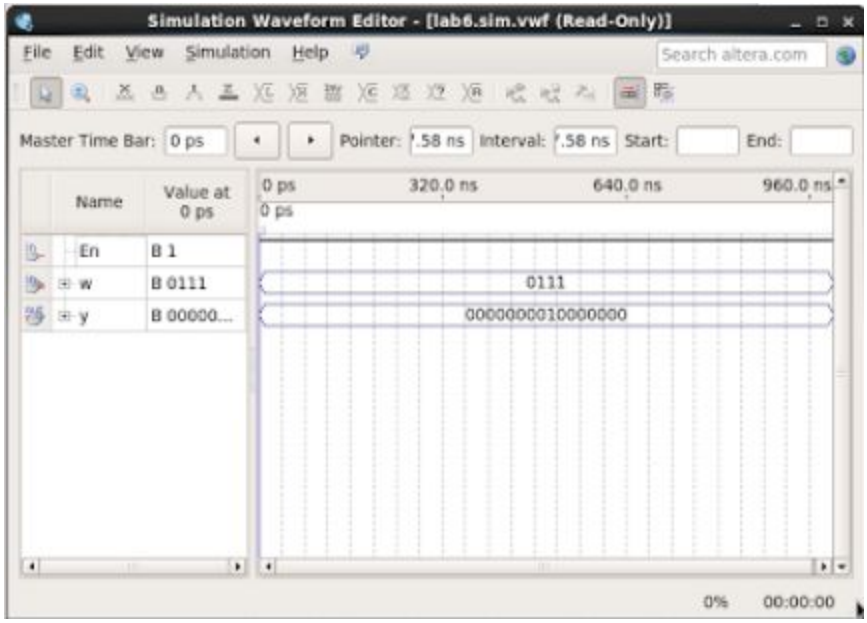
Latches:



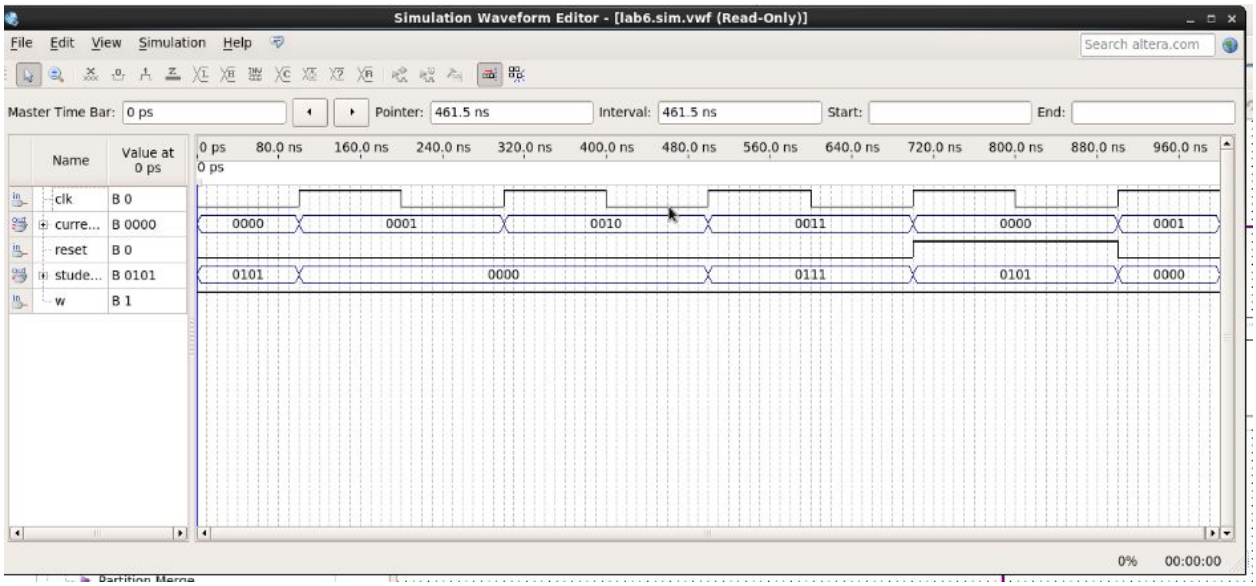
ALU_1: *my student number last digits are: 8680 so A= 10000110 B= 10000000*



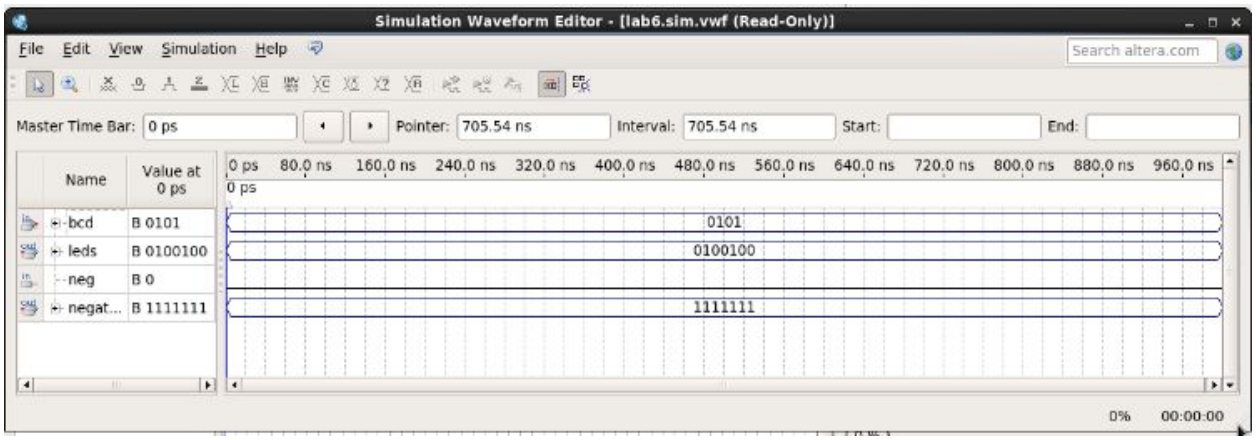
4x16 Decoder:



FSM:(shows digit 5 0 0 7 and *reset back to 5*)



SSEG:

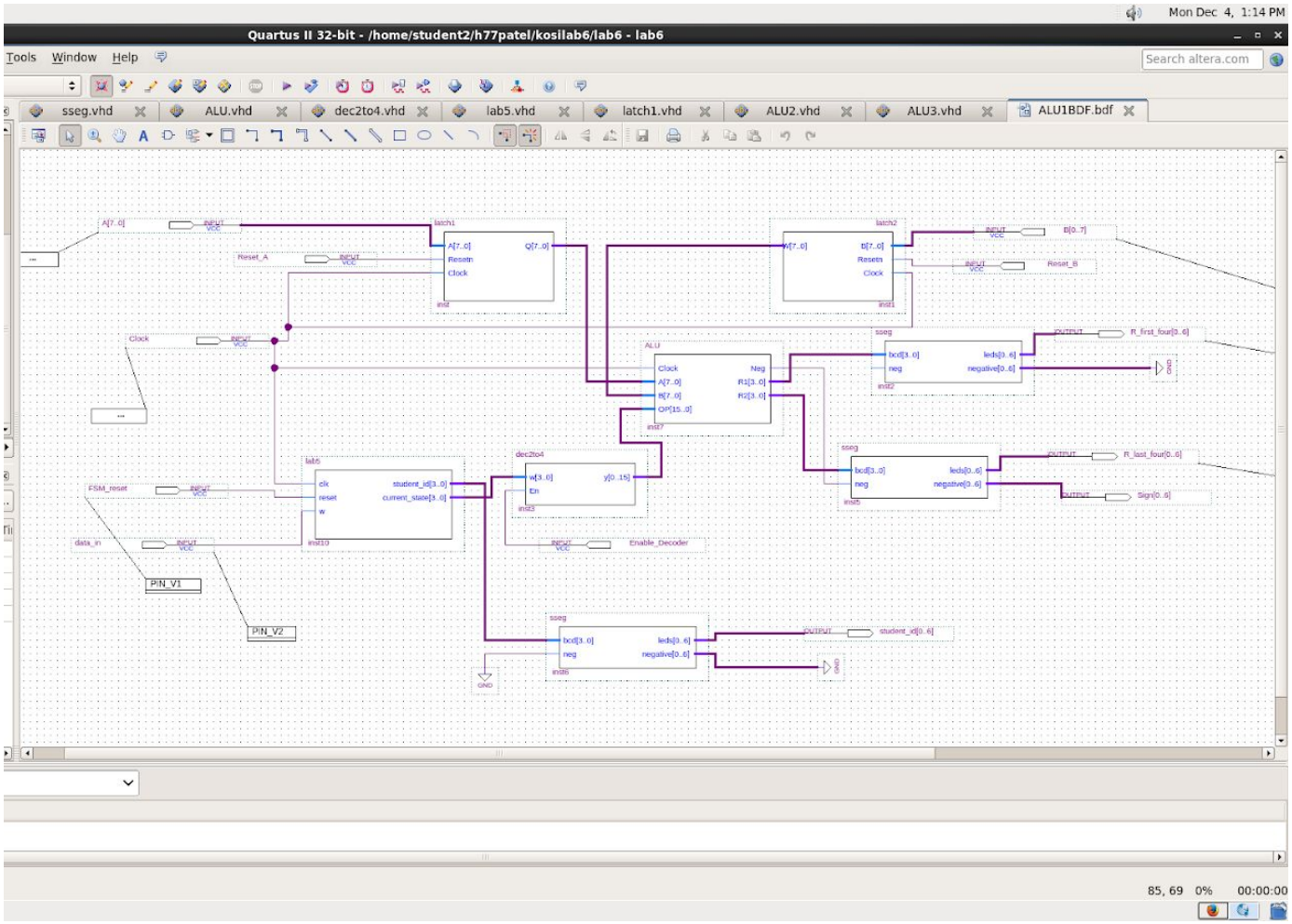


4. ALU_1 for Problem Set 1 of the Lab 6 procedure

a) Description and design of the ALU_1.

ALU 1 is made to perform a set of functions on 2 8-bit numbers given a specific operation. Therefore the design of ALU 1 consists of 2 8-bit number inputs coming from the latches as well as a clock, and a operation coming from the decoder. The outputs 2 4-bit numbers R1 and R2 which are the binary decoded version of the result of the function performed by the ALU.

b) Screen shot of the BDF for ALU_1.



c) Purpose of all inputs and outputs of ALU_1.

The inputs are 2 8-bit number inputs coming from the latches, clock, and a 16- bit operation number. The 2 8-bit numbers' are the numbers on which we are performing the functions. The 16-bit operation number's purpose is to selection which operation should be performed on the 2

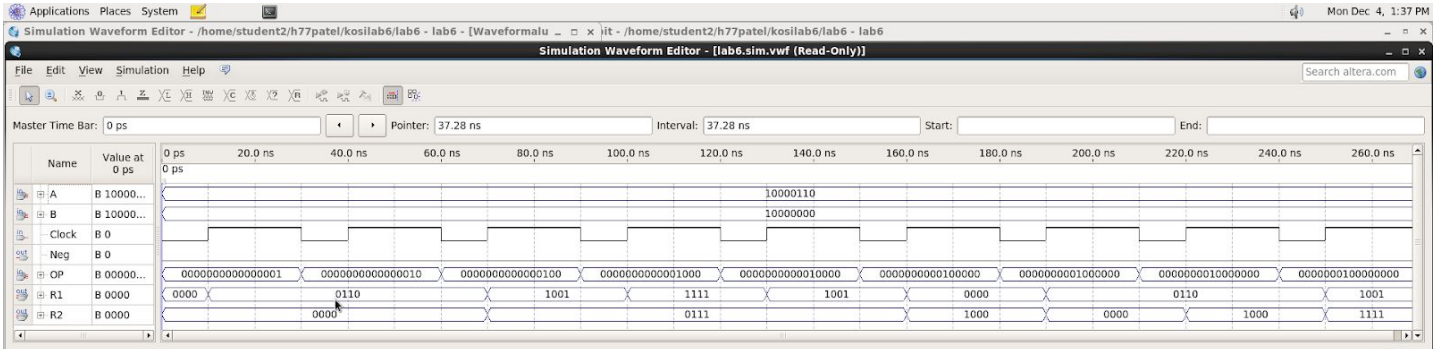
8-bit numbers. It comes from the decoder which took the fsm current state output. The clock's purpose is to allow us to go through each operation.

The outputs are 2 4-bit numbers R1 and R2 which are the binary decoded version of the result of the function performed by the ALU. Their purpose is to to be displayed on the seven segment displays. The result is actually an 8-bit number but since we are displaying on seven segment display (can only display hexadecimal) we are required to split the 8-bit number into 2 4-bit numbers.

d) Table of Microcode's generated by decoder for ALU_1.

Function #	Microcode	Boolean Operation / Function
1	0000000000000001	sum(A, B)
2	0000000000000010	diff(A, B)
3	0000000000000100	$\overline{A}$
4	0000000000001000	$\overline{A \cdot B}$
5	0000000000010000	$\overline{A + B}$
6	0000000000100000	$A \cdot B$
7	0000000001000000	$A \oplus B$
8	0000000010000000	$A + B$
9	0000000100000000	$\overline{A \oplus B}$

e) Complete Waveform File.

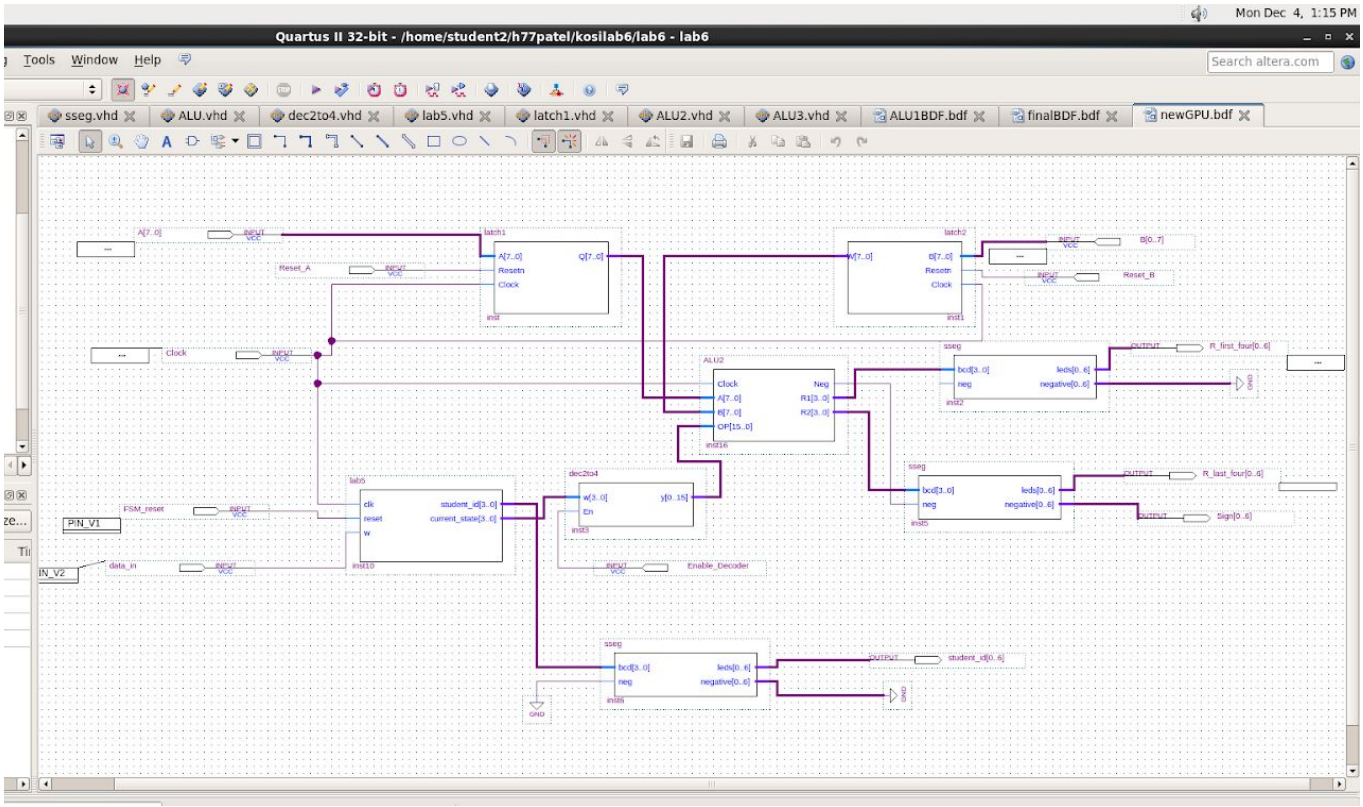


5. ALU_2 for Problem Set 2 of the Lab6 procedure

a) Description and design of the ALU_2.

The description of ALU 2 is similar to ALU 1, the only difference is in the functions. ALU 2 is made to perform a set of functions on 2 8-bit numbers given a specific operation. I was given set C. Therefore the design of ALU 2 consists of 2 8-bit number inputs coming from the latches as well as a clock, and a operation coming from the decoder. The outputs 2 4-bit numbers R1 and R2 which are the binary decoded version of the result of the function performed by the ALU.

b) Screen shot of the BDF for ALU_2.



c) Purpose of all inputs and outputs of ALU_2.

The purpose of all inputs and outputs for ALU 2 is the same as for ALU 1. The inputs are 2 8-bit number inputs coming from the latches, clock, and a 16- bit operation number. The 2 8-bit numbers' are the numbers on which we are performing the functions. The 16-bit operation number's purpose is to selection which operation should be performed on the 2 8-bit numbers. It comes from the decoder which took the fsm current state output. The clock's purpose is to allow us to go through each operation.

The outputs are 2 4-bit numbers R1 and R2 which are the binary decoded version of the result of the function performed by the ALU. Their purpose is to to be displayed on the seven segment displays. The result is actually an 8-bit number but since we are displaying on seven segment display (can only display hexadecimal) we are required to split the 8-bit number into 2 4-bit numbers.

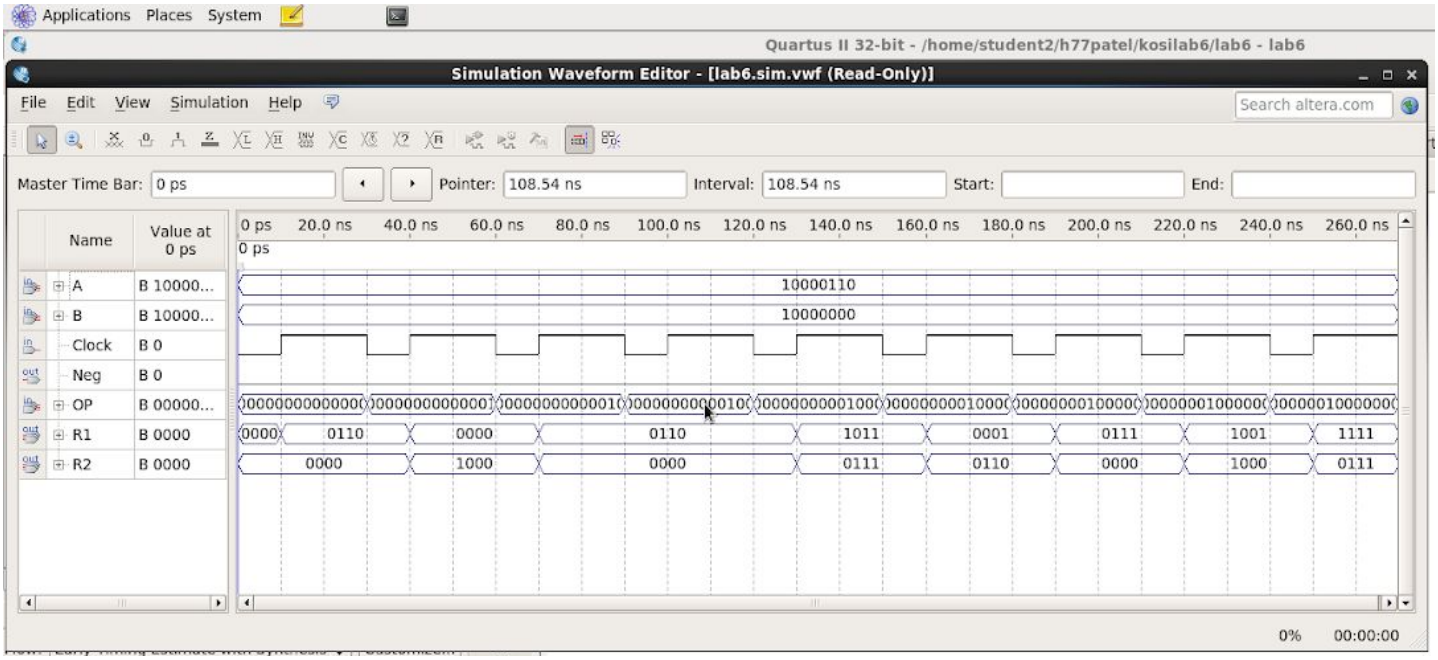
d) Table of Microcode's generated by decoder for ALU_2.

For inputs A=00010110 B=00111001

Function number	microcode	Boolean operation/function	Expected Output
1	0000000000000001	diff(A,B)	1101 1101 (DD)
2	0000000000000010	Produce the 2's complement of B	0011 1001 (39)
3	0000000000000100	Swap the lower 4 bits of A with lower 4 bits of B	1001 0110 (96)
4	0000000000001000	Produce null on the output	0000 0000 (00)
5	0000000000010000	Decrement B by 5	0011 0100 (34)
6	0000000000100000	Invert the bit-significance order of A	1001 1100 (9C)
7	0000000001000000	Shift B to left by three bits, input bit = 1 (SHL)	1100 1111 (CF)
8	0000000010000000	Increment A by 3	0001 1001 (19)
9	0000000100000000	Invert all bits of B	1100 0110 (C6)

e) Complete Waveform File.

Waveform to prove correctness of functions last 4 digits of my student number 500758680
A=10000110 B=10000000

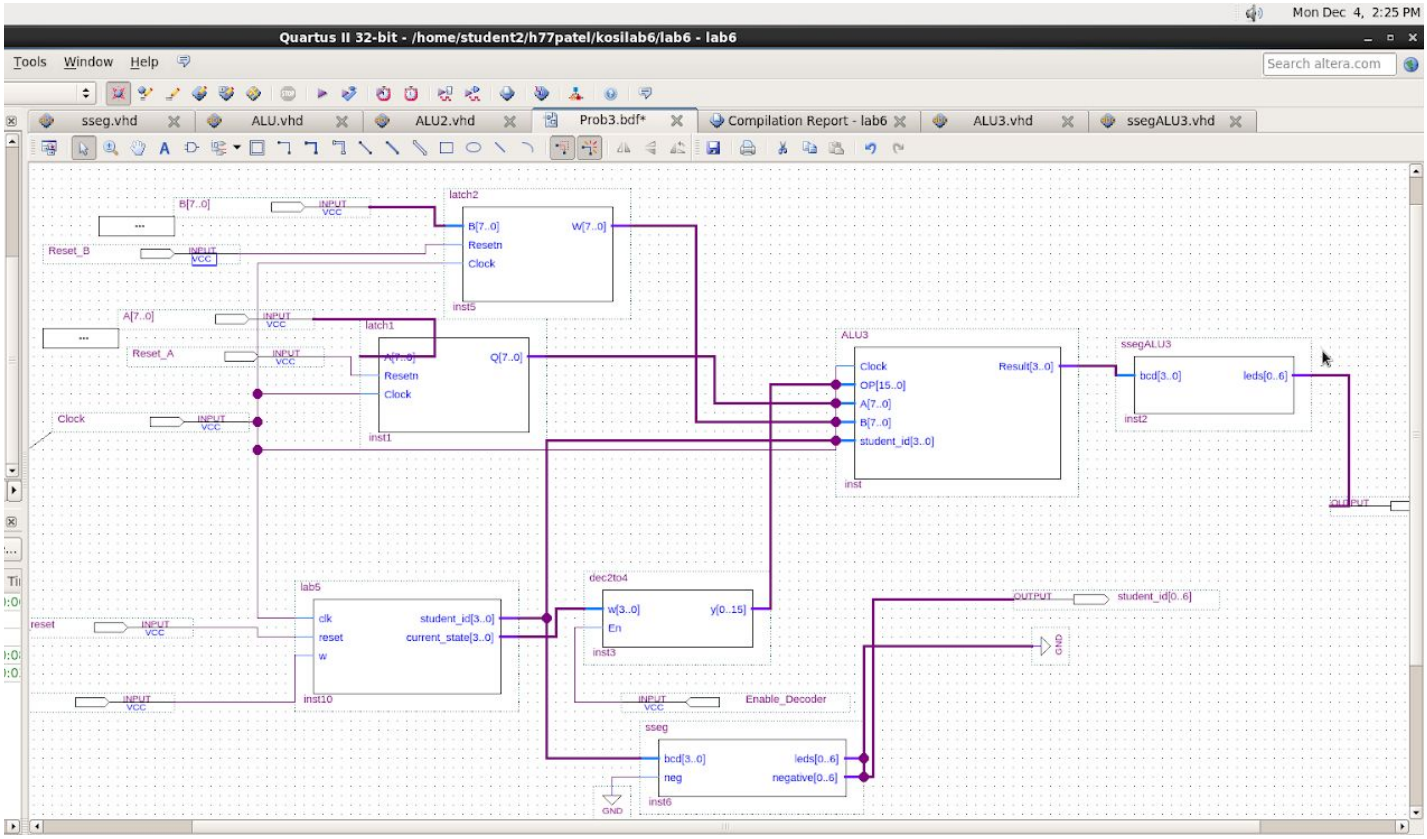

Applications Places System
Quartus II 32-bit - /home/student2/h77patel/kosilab6/lab6 - lab6
Simulation Waveform Editor - [lab6.sim.vwf (Read-Only)]
File Edit View Simulation Help
Search altera.com
Master Time Bar: 0 ps
Pointer: 108.54 ns
Interval: 108.54 ns
Start:
End:
Name
Value at 0 ps
A B 10000...
B B 10000...
Clock B 0
Neg B 0
OP B 00000...
R1 B 0000
R2 B 0000
0 ps 20.0 ns 40.0 ns 60.0 ns 80.0 ns 100.0 ns 120.0 ns 140.0 ns 160.0 ns 180.0 ns 200.0 ns 220.0 ns 240.0 ns 260.0 ns
10000110
10000000
0000 0110 0000 0110 1011 0001 0111 1001 1111
0000 1000 0000 0111 0110 0000 1000 0111
0%
00:00:00

6. ALU_3 for Problem Set 3 of the Lab6 procedure

a) Description and design of the ALU_3.

I got problem a for this part. ALU 3 is made to determine if the digits of my student ID are odd or even. Therefore the design of ALU 3 consists of 2 8-bit number inputs coming from the latches as well as a clock, a 16-bit operation coming from the decoder, and a 4- bit number for the student id digit. The output is a 4-bit number, Result which goes to the sseg and displays y or n.

b) Screen shot of the BDF for ALU_3.



c) Purpose of all inputs and outputs of ALU_3.

The inputs are the a 4-bit number representing the student id, 2 8-bit number inputs coming from the latches, clock, and a 16- bit operation number. The 4-bit number is the number which will be tested to see if it's odd or even. The 16-bit operation number's purpose is to select which digit of the student id is going to be used. It comes from the decoder which took the fsm current state output. The clock's purpose is to allow us to cycle through each digit.

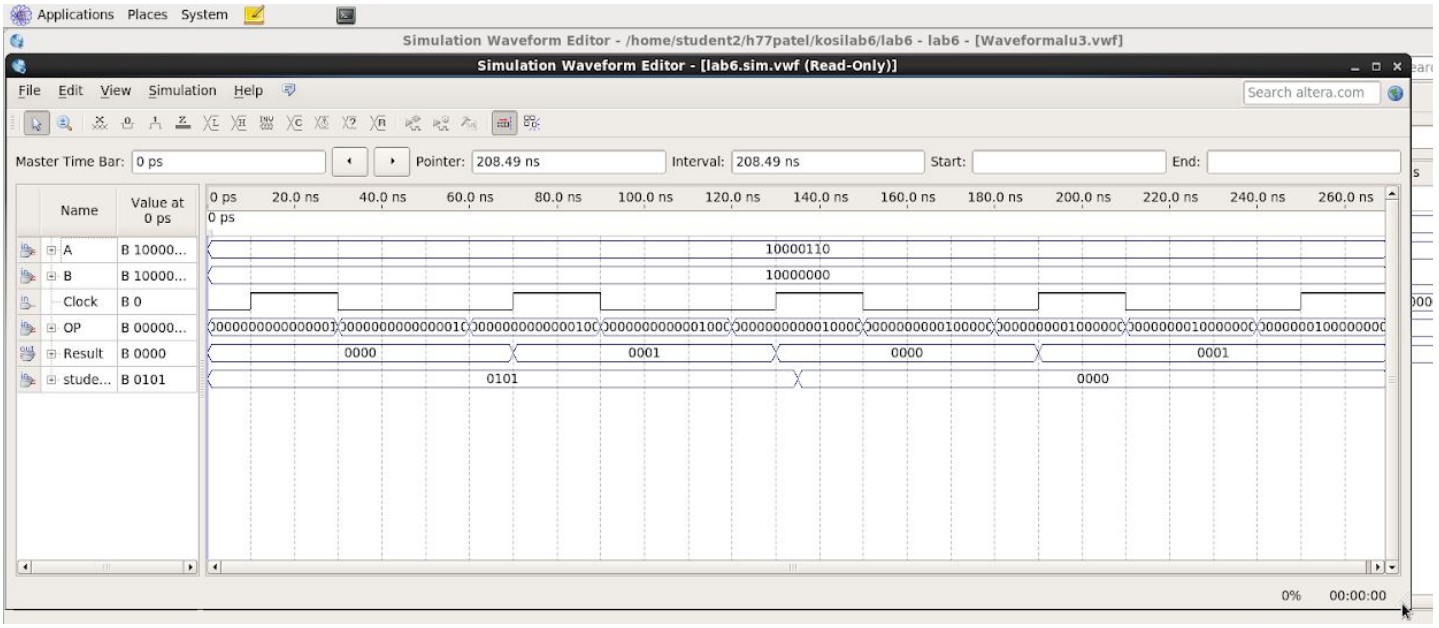
The output is a 4-bit result number. It's purpose is to determine if a 'y' or 'n' will be displayed. If the student id digit was odd, a 'y' prints out on the seven segment display, otherwise a 'n' prints out.

d) Table of Microcode's generated by decoder for ALU_3.

Function number	microcode	Boolean operation/function	Expected Output
1	0000000000000001	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	y
2	0000000000000010	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	n
3	0000000000000100	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	n
4	0000000000001000	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	y
5	0000000000010000	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	y
6	0000000000100000	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	n
7	0000000001000000	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	n
8	0000000010000000	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	n
9	0000000100000000	display 'y' if the FSM output (student_id) is odd and 'n' otherwise	n

e) Complete Waveform File.

Waveform to prove correctness of functions with my student id : 500758680


Applications Places System
Simulation Waveform Editor - /home/student2/h77patel/kosilab6/lab6 - lab6 - [Waveformalu3.vwf]
Simulation Waveform Editor - [lab6.sim.vwf (Read-Only)]
File Edit View Simulation Help
Search altera.com
Master Time Bar: 0 ps
Pointer: 208.49 ns
Interval: 208.49 ns
Start:
End:
Name
Value at 0 ps
A
B 10000...
B
B 10000...
Clock
B 0
OP
B 00000...
Result
B 0000
stude...
B 0101
0 ps
20.0 ns
40.0 ns
60.0 ns
80.0 ns
100.0 ns
120.0 ns
140.0 ns
160.0 ns
180.0 ns
200.0 ns
220.0 ns
240.0 ns
260.0 ns
10000110
10000000
0000
0001
0000
0001
0101
0000
0%
00:00:00

7. Conclusion

In conclusion, the lab portion of this course all built up to this final project. With the help from previous labs and the theory taught in class we were able to create a fully functioning GPU by implementing our knowledge on storage elements, finite state machines, decoders, boolean arithmetic and seven segment displays. In this course we used Quartus to program in vhdl, create block schematics and assign the necessary pins to an fpga board to display our outputs on a seven segment display. By modifying ALU 1 in part 1 we were able to perform a various set of different functions for part 2. By modifying the initial design we were able to perform another function with our GPU. Overall, this project solidified our understanding of basic VHDL coding and tested our ability to implement various designs.