NOVEMBER 30, 2018

BME 328 LAB 6 REPORT

KUSHAL SHAH

SECTION: 4
ID: XXXXX3903
TA: Ali

Table of Contents:

Introduction:

The objective of this laboratory experiment was to design a Simple General - Purpose processor that functions as an Arithmetic and Logic Unit (ALU) in a VHDL environment and later test it on the FGPA board. The components used to achieve this were two latches, one ALU, a control unit which includes an FSM and a 4-16 decoder, and a 7-segment display to output the result of the program on the FGPA board in the form of hexadecimal. The processor accepts an input of two eight-bits from the latches, which then enters the ALU, in addition to the inputs from the control unit. The desired inputs are then processed, and the required output is generated on the seven-segment display.

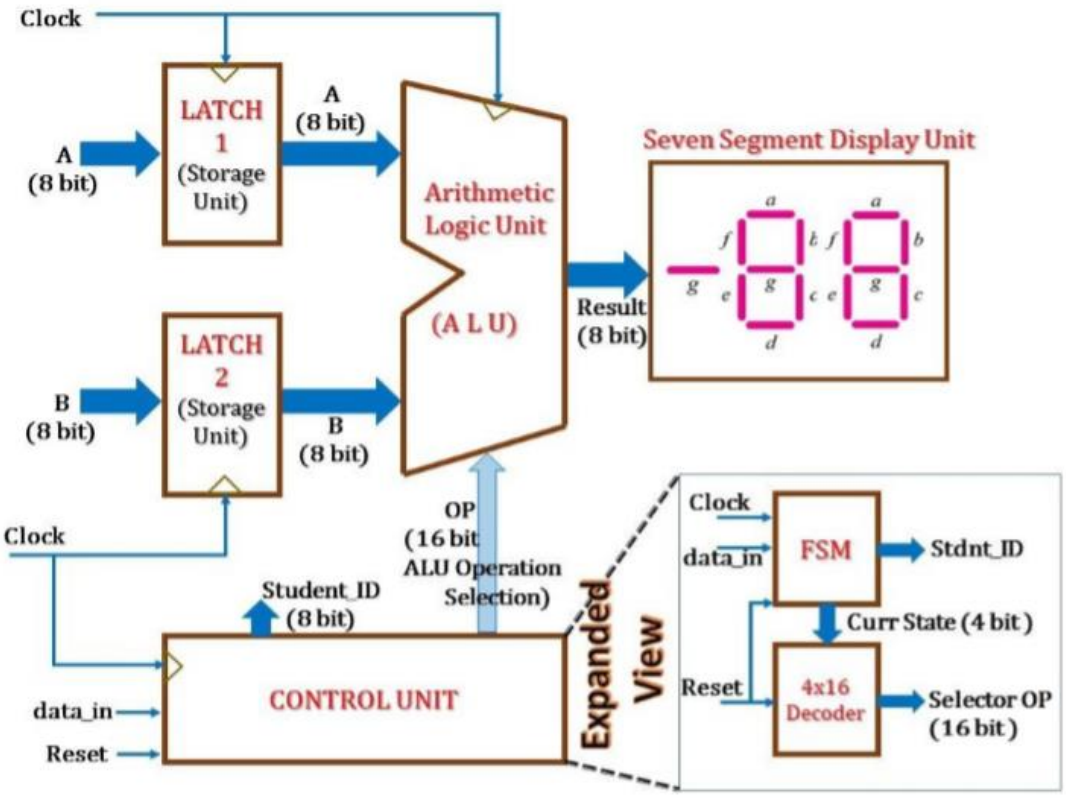


Figure 1. The Block Diagram of the GPU

Components:

Latch: The latch is a storage element which temporarily holds the values of the inputs A(Reg1) and B (Reg2), which are later taken by the latch into processing once it's clocked. The latch is made up of several components including D and clock (inputs), and Q and NOT (Q) which are the outputs. The latch is distinguished from a flip flop through the clock notation. A latch has "clock" written down while a flip flop has a notation of "->". The clock input is either '1' or '0' denoting if it's clocked or not.

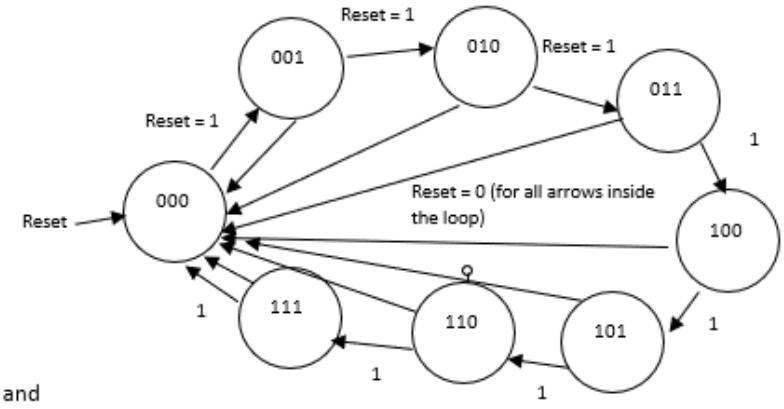


Figure 2: Shows the mechanism of the upcounter.

and

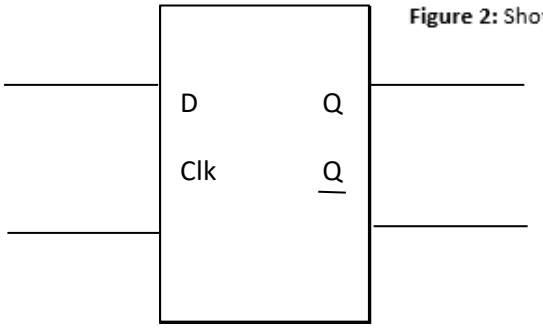
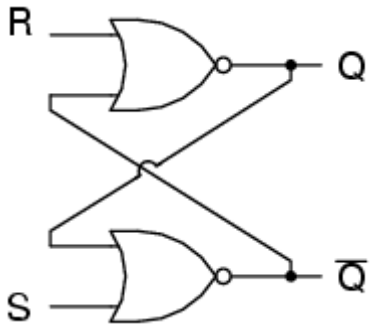


Figure: A sample D-latch



S	R	Q	$\overline{Q}$
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	0	0

Figure: S-R Gated Latch circuit and truth table

Correction: Q and NOT(Q) is X i.e. not possible when S = 1, and R = 1

4-16 Decoder:

A decoder is a component that returns certain outputs determined by the formula 2^m = n where m is the number of inputs and n is the number of outputs (4 input decoder returns 16 outputs). The enable is high when it has a value of 1 and low when it has a value of 0. The decoder also has a reset button which sets everything to its initial state to '0' once it is activated. The user must have inputted 16 functions (16 outputs) however, the seven-segment display seven functions based on certain inputs while the rest are low (0) or don't cares.

W	X	Y	Z	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15
0	0	0	0	1	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	1	0	0	0	1	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	1	0	0	0	0	0	0	1	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	1	1	0	0	0	0	0	0	0	1	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
1	0	0	0	0	0	0	0	0	0	0	0	1	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
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	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
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Figure: 4-16 decoder where W, X, Y, Z are inputs and M0 to M15 are outputs (LSB -> MSB)

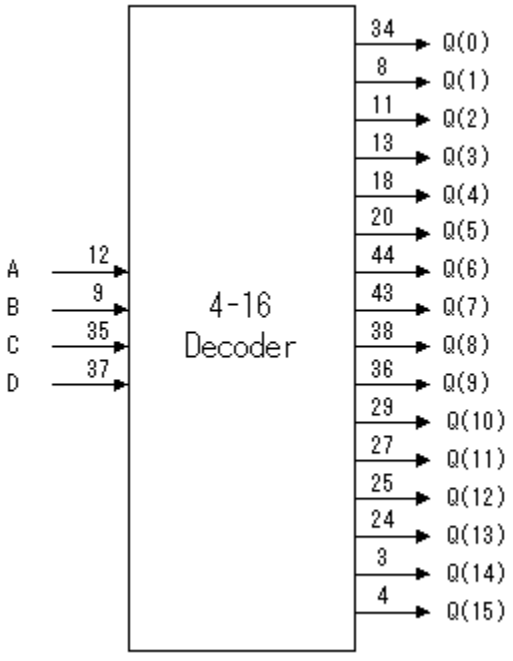


Figure: 4-16 decoder with A, B, C, D as inputs and Q (0) to Q (15) as outputs.

Finite State Machine (FSM):

An FSM is a machine that houses both the combinational and sequential circuit. The sequential circuit stores the present and previous values through its storage elements as mentioned above. The input of the FSM (i.e. the storage elements) is the next state variable and the input is the present state variable. In a Moore FSM (current project), the next state variable is dependent on the primary inputs and the present state variable whereas the primary output is dependent only on the present state variable. On the other hand, in a Mealy FSM the next state variable and the primary output are both dependent on the primary inputs and the present state variable.

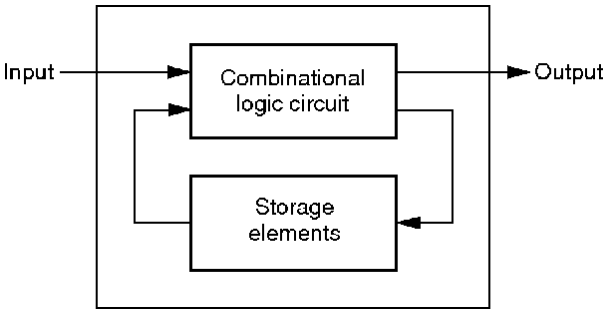


Figure: A sample FSM

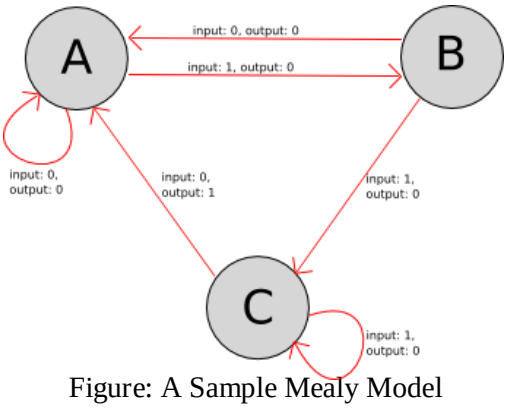


Figure: A Sample Mealy Model

ALU1 for Problem Set 1:

For ALU1, the experiment was designed to take two eight-bit values, Reg1 and Reg2 respectively and process them through the nine functions posted below. The microcode's were initialised and the conditions or the functions were written along with it for all nine functions. The output was then processed to the FGPA board, through the pin planner, where the result was displayed through the hexadecimal 7 segment displays, every time it was clocked. The student id bit was displayed on the 7-segment counter simultaneously. FSM Moore machine was used to jump to next state.

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	$\overline{A \oplus B}$

Table 1. ALU Core Operations for Problem 1

Figure: 9 functions of ALU1

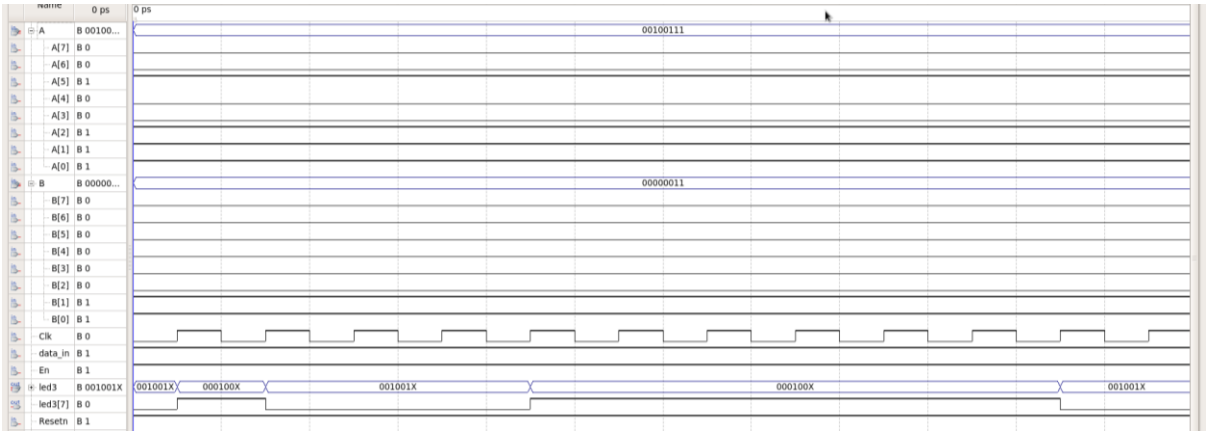


Figure: Waveform for ALU1

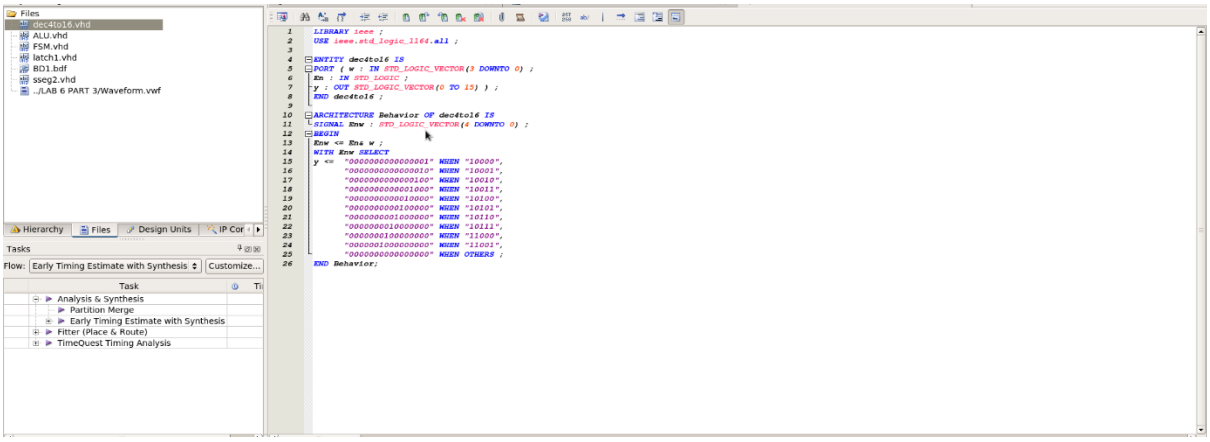


Figure: microcode generated by decoder for ALU1

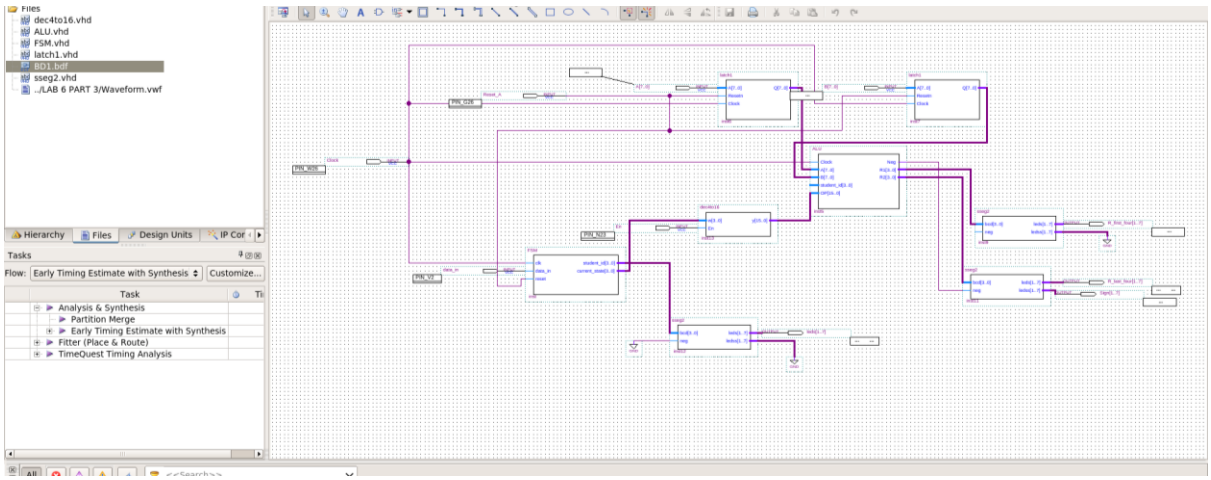


Figure: Block diagram for ALU1

ALU2 for Problem Set 2:

For ALU2, the process was almost like the ALU1. The only difference was the operation in each of the functions. The output of it was again sent to the 7segment displays. I was assigned b) with the following operations:

b)

Function #	Operation / Function
1	Swap the lower and upper 4 bits of **A**
2	Produce the result of ORing **A** and **B**

3	Decrement **B** by 5
4	Invert all bits of **A**
5	Invert the bit-significance order of **A**
6	Find the greater value of **A** and **B** and produce the results (Max(**A**,**B**))
7	Produce the difference between **A** and **B**
8	Produce the result of XNORing **A** and **B**
9	Rotate **B** to left by three bits (ROL)

Figure: My operations for ALU2

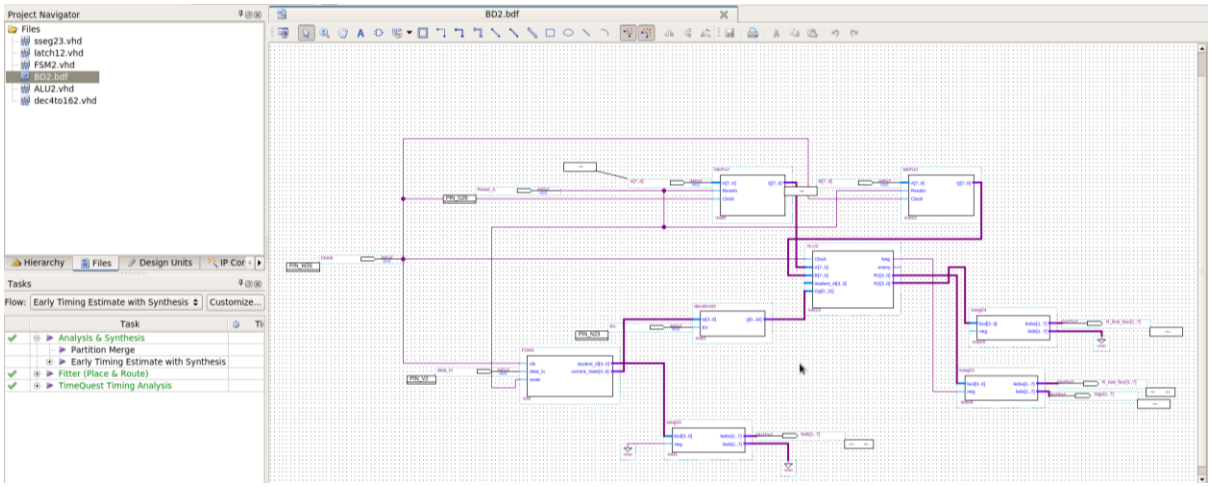


Figure: Block diagram for ALU2

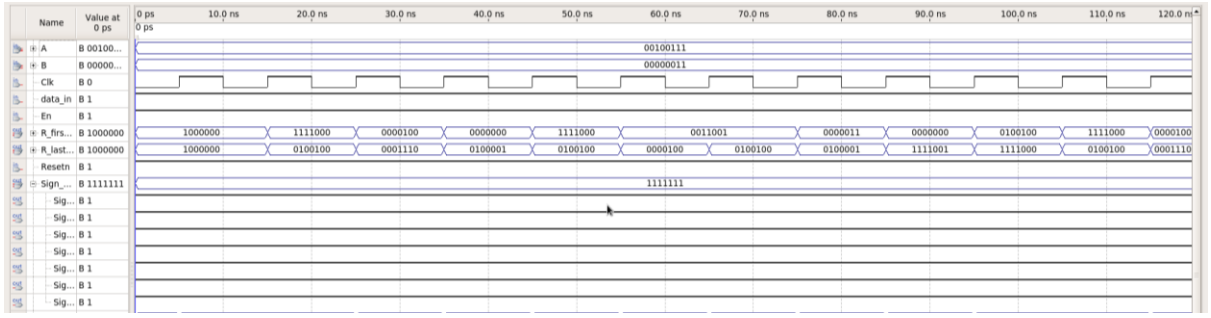


Figure: Waveform for ALU2

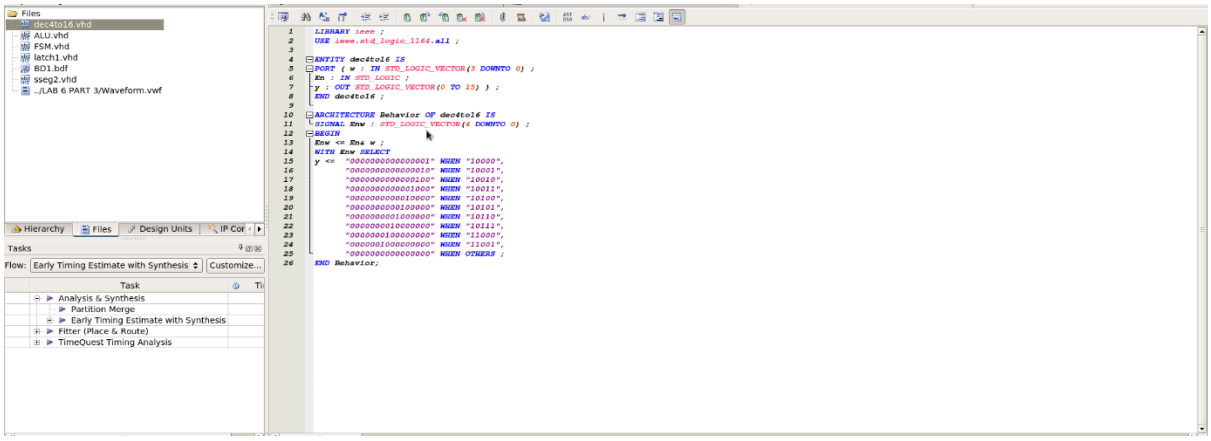


Figure: microcode generated by decoder for ALU2

ALU3 for Problem Set 3:

For ALU problem set three, the ALU is programmed that will take two 8-bit inputs A and B from the latches and 16-bit input from the Control Unit. The ALU is implemented with a function that our TA gave us. Again, it was ‘b’ for me. Here, I had to output or display ‘y’ if the FSM (student ID) was even, else ‘n’, on the hexadecimal 7 segment displays. The objective of this ALU3 was probably the same as the above ALUs. Once again, the purpose of seven

segment display unit is to show the student number along with the functions being operated on the display along with the “y” or “n”. For all parts, FSM unit is used as a Moore machine to go to next state of the function. The VHDL code used in this ALU was quite different from the previous ones. I had to make another code for sseg.vhd, as well as modify the ALU3.vhd code. Of course, this was achieved through if/else statements using the microcode’s.

b) For each microcode instruction, display 'y' if the FSM output (**student_id**) is even and 'n' otherwise

Figure: Operations for ALU3

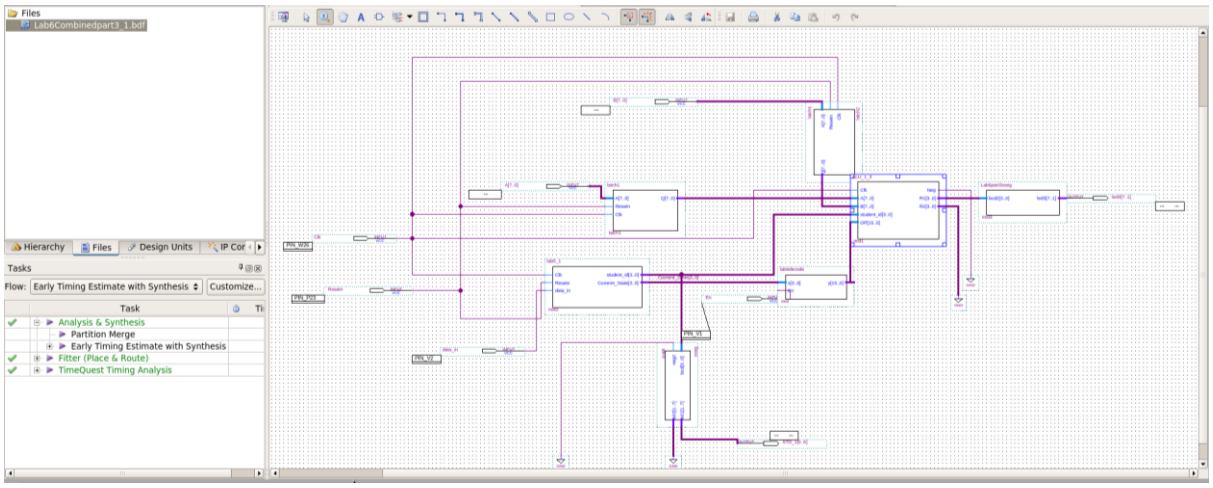


Figure: Block diagram for ALU3

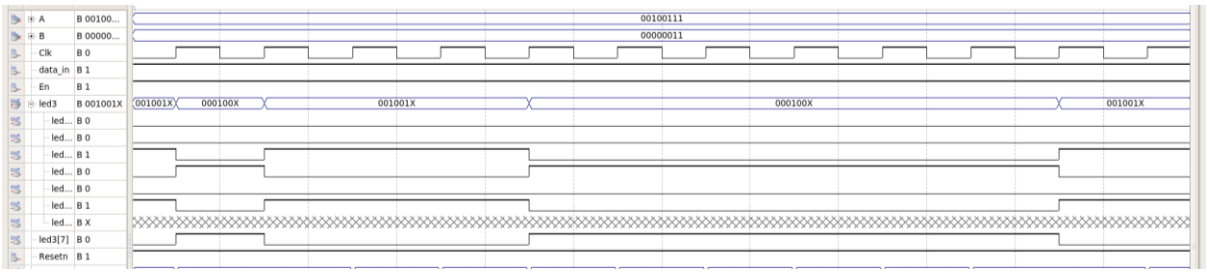


Figure: Waveform for ALU3

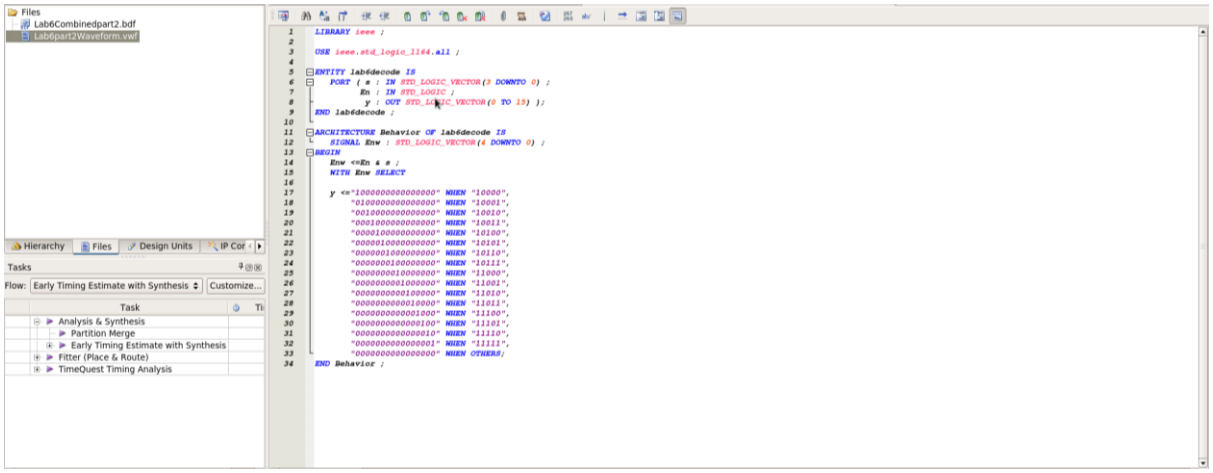


Figure: Decoder microcode for ALU3

Conclusion:

In conclusion, the lab was successfully done for all three problems for ASU using VHDL code. Through this lab, we learned the functionality of latches, FSM, up counter, decoder, and most importantly, the ALU. The latches took 2 8-bit inputs so that any function could be executed of your choice. All these components were used together to generate a Simple General-Purpose Processor.