Zewail city of science and technology

Digital design and computer architecture (CIE 239)

Stopwatch project

Members:

Norhan mohamed mohsen - 201701436

Rodina mohamed - 201900642

Mariam Tawfik - 201900720

Instructor: Dr.mohamed samir

Table of content:

Abstract:

This project aims to show a detailed steps of how a stopwatch was designed from scratch to meet all its functional requirements including the iterations of the design and how the final design was reached also including the details of each component of the design regarding its input, output and simulation and how all those component modeled and integrated on modelsim to be fully functional on FPGA.

Introduction:

Basically, the digital stopwatch is a device that is meant to measure the time elapsed between the start and the end of the event that the user specifies with a lot of other options including that it can count up or down the time according to what mode the user chooses, pausing and resetting the counters. Also, one from its important options is the segment saver mode that appears to the user while he is not using it. Moreover, there will be a specified error mode that will appear to the user to inform him that he did something illegal to the options of the stopwatch.

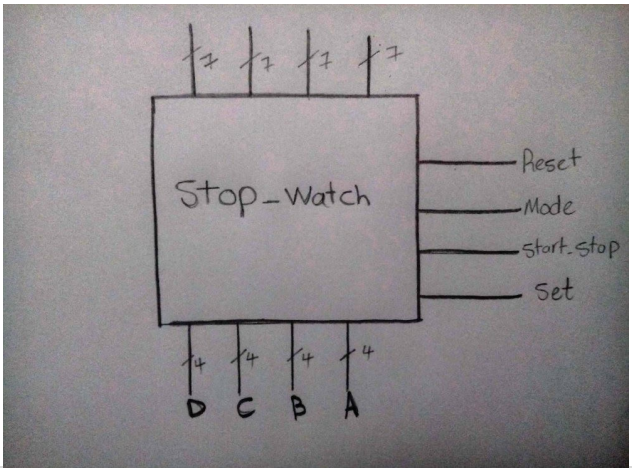
Design iterations:

First, we design the stop-watch as a block and we defined input and outputs for this block

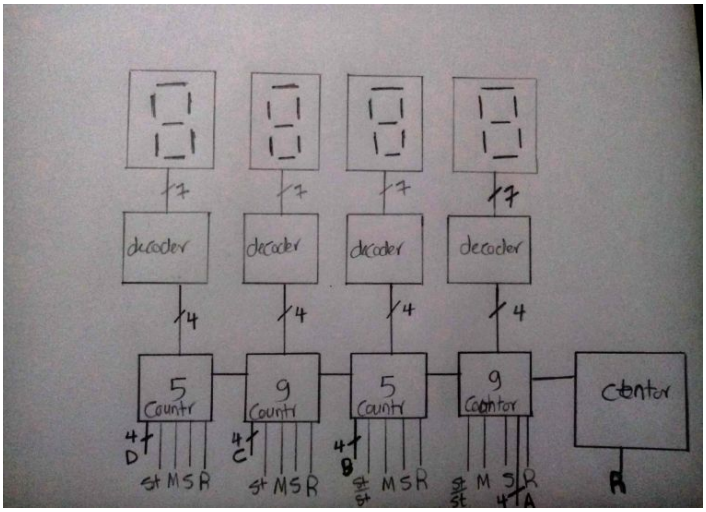
It has four switches and input(A,B,C,D)in case the user presses the set switch to set the time as shown in the figure below. This block has 9 inputs:

Reset-Clock-Set-Mode-Start-A,B,C,D

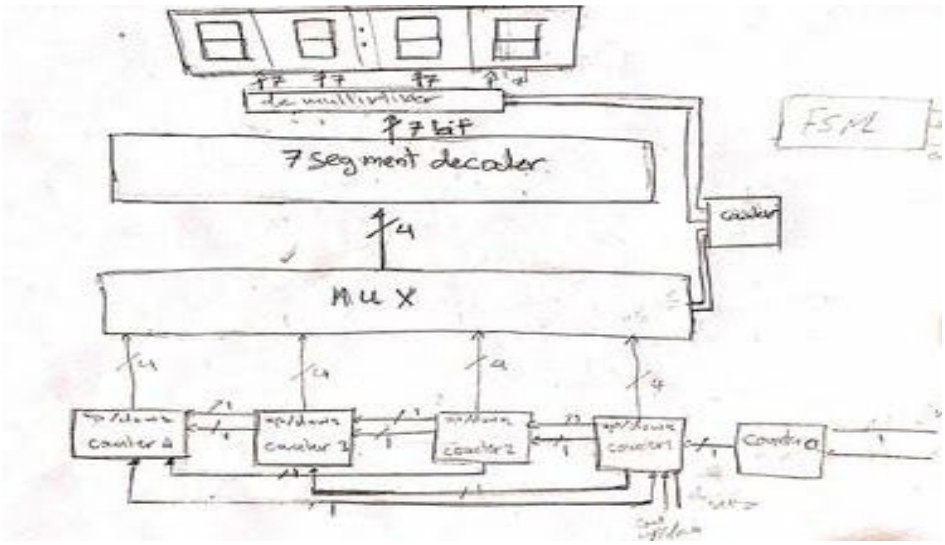
and 4 outputs to the 7 segment display.



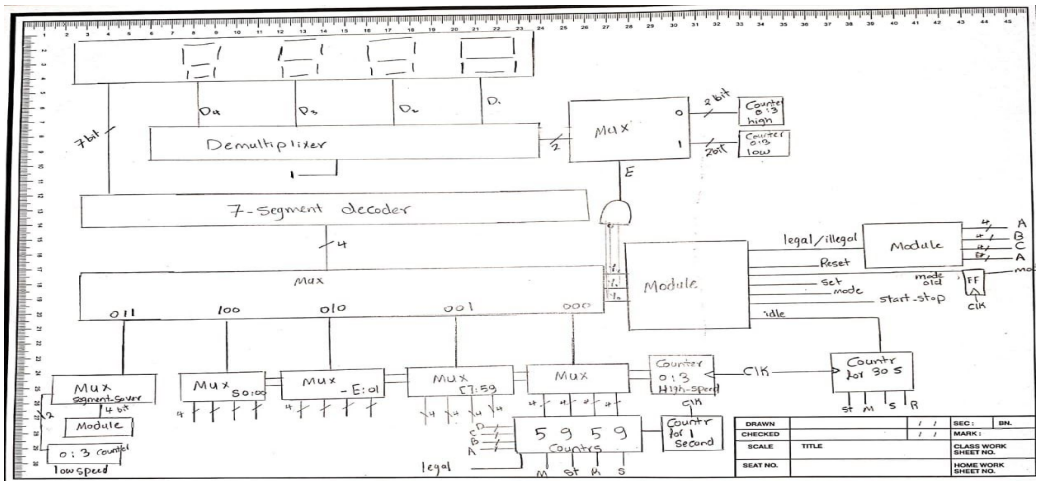
Accordingly, we began to unblock this into some modules without considering cases of errors or segment saver mode for simplicity as shown below, we used 4 segment display for output and four 7-segment decoders. Then it seems that this would need 28 wires just for the output so we replaced the 4 segment display with only one that takes 11 input.



This was an initial design for using 4 digit 7 segment display without deeping in the details of the remaining features of the Stop-Watch.



This was our final design for the stop-Watch and we considered all cases of errors in the specifications is the design and we added one case for error if the user pressed set and reset at the same time but this design does not include two leds which flash each one second between seconds segment display and minutes segment display as they are one component not separated so we found that reducing number of wires are much important than these two leds.



But this design has some kind of error that we observed one day before the deadline of phase 1 that the decoder has only 16 cases for displaying output and we have more 16 because segment saver mode and different errors. This caused a dramatic change in the design to add two decoders (one for counters and other for the segment saver mode and possible errors).

-To consider more than 16 cases in segment display we used a 7-segment decoder and the display is controlled by a multiplexer to select whether counters should be displayed or other cases.

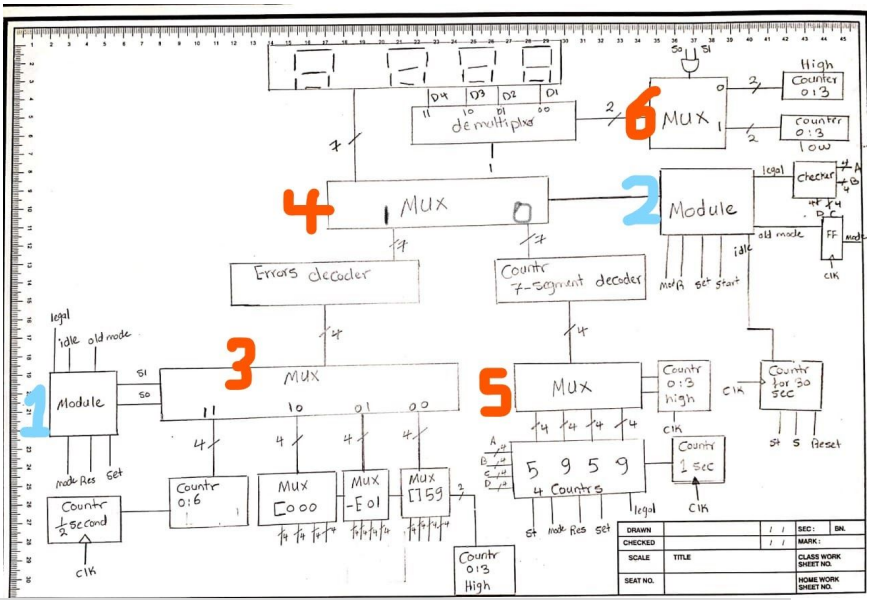
So this was our final design for phase1 which contains:

4 digit segment display -module(checker)

2 decoder-Demultiplexer-7 multiplexer

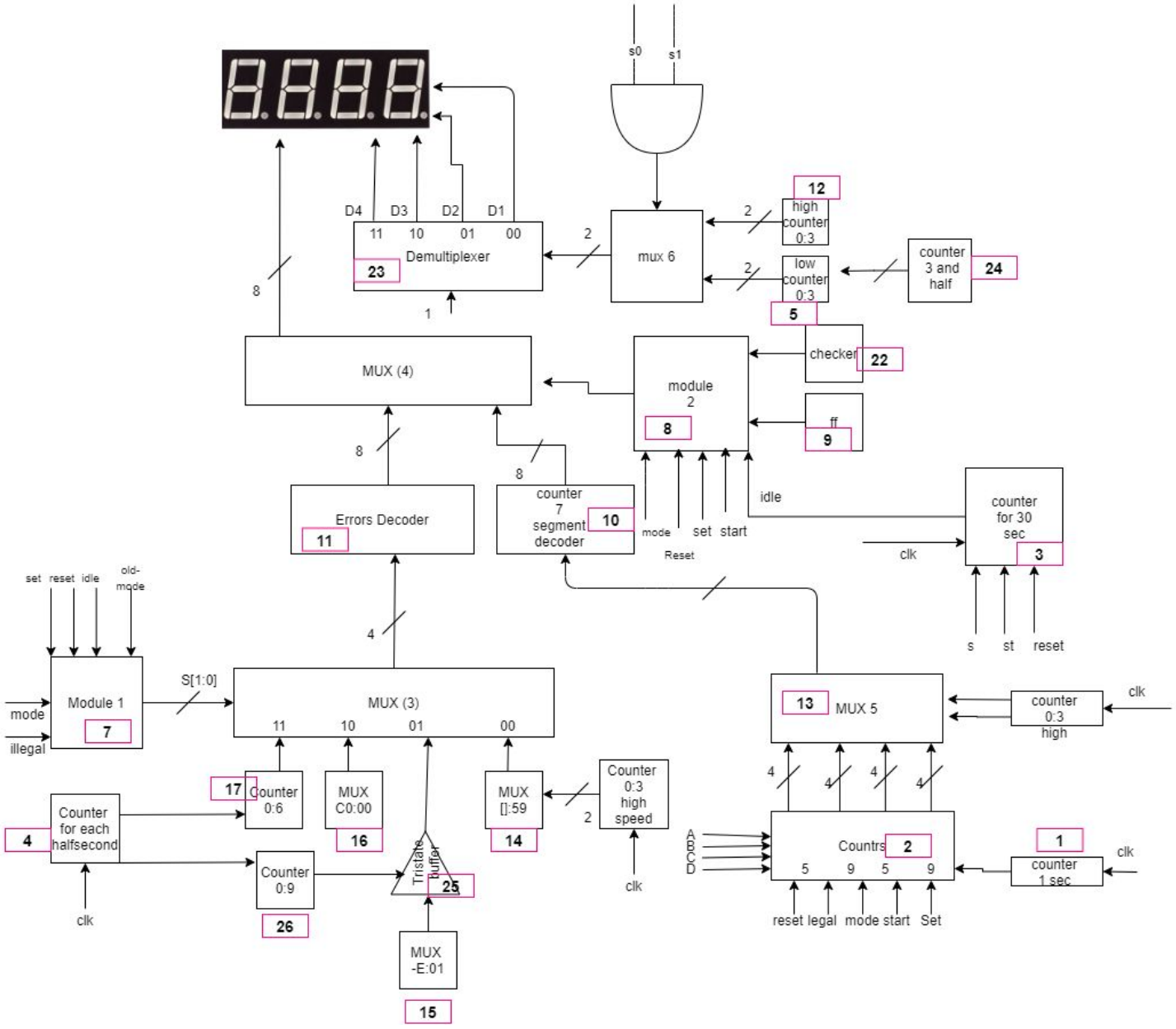
module(checker) -Flip-flop

10 counters-Module(1)-Module(2).

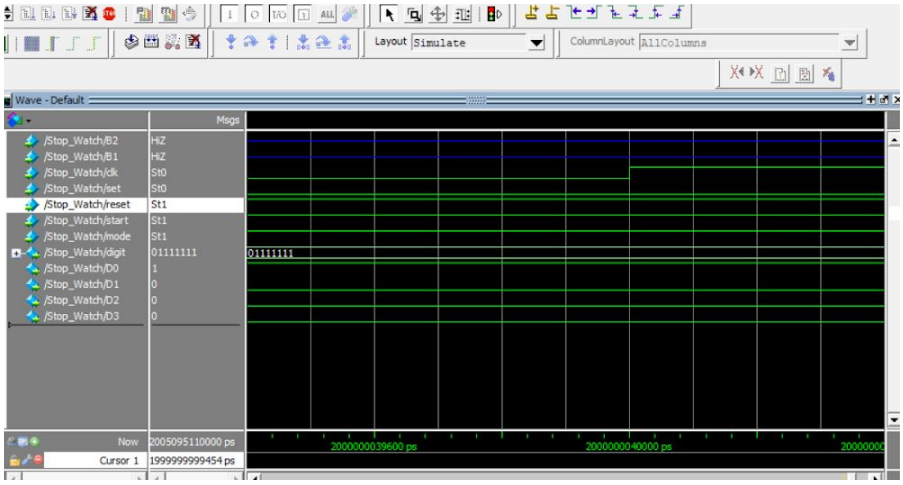


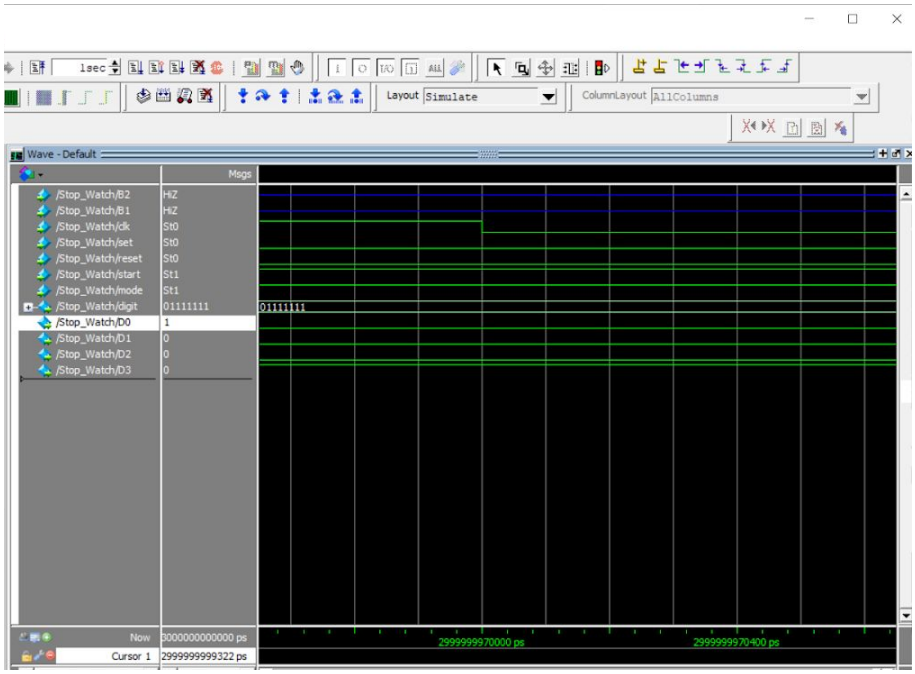
The previous design did not consider the case of flashing Error 3 times so we added counter0to9 to give a signal to tristate buffer when the digit is divisible by 3 to flash the -E:01 3 times. In addition,we found that segment display has 8 input instead of 7 input so we code decoder to 8 bit, Also adding counter 3 and half for segment-saver mode to switch between digits after the illumination of each led.

Final Design

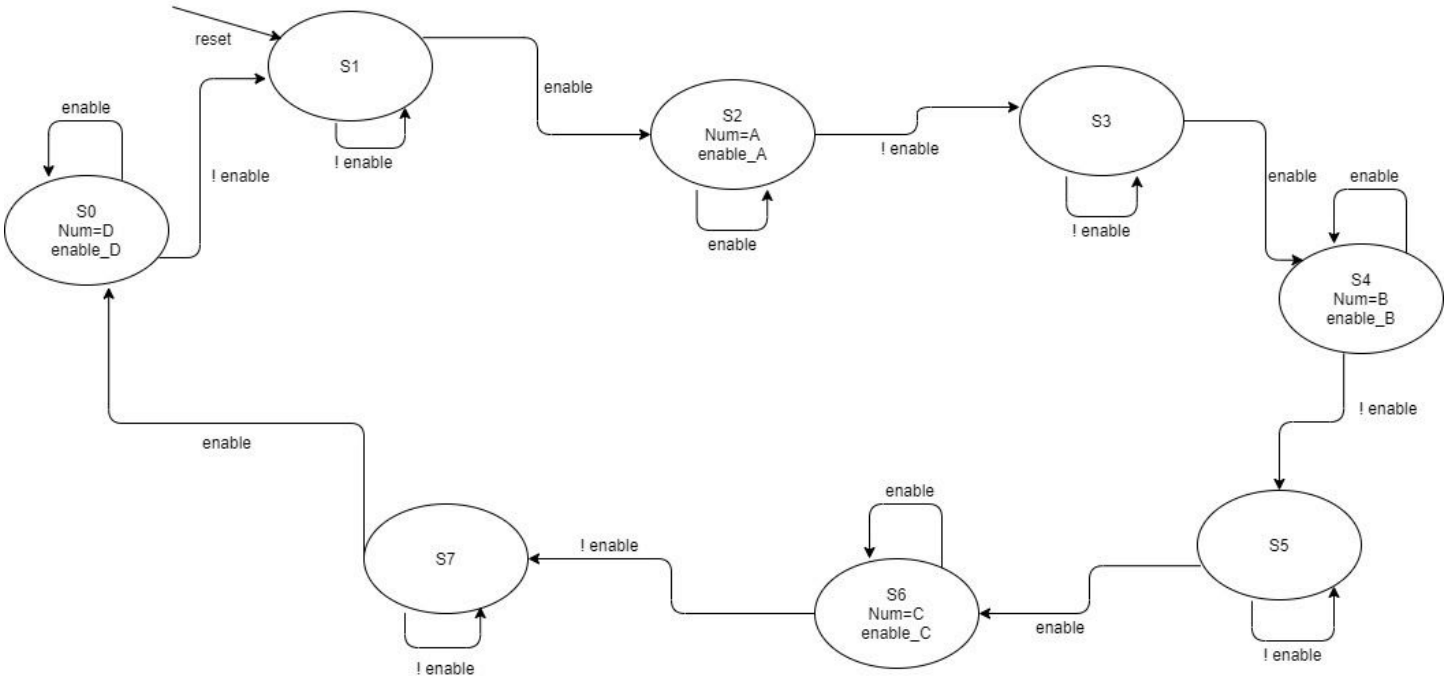


Simulation of the Stop_Watch Module:





Moore FSM:



It is the module responsible for taking the input digits(which represents the second and minutes) from the user when he switches on the set switch to set the number he wants to start from it the countdown.

Its inputs:(clk, reset, enable, Num)

clk:is the clock of the FPGA.

reset:is the switch reset.

enable and Num:They are the outputs of the keypad where they behave as inputs to the FSM**.**

Its outputs:(enable_A,enable_B,enable_C,enable_D,A,B,C,D)

where(A,B,C,D):represents the 4 digits he enters to start from them the countdown.

A&B: represents the seconds.

C&D: represents the minutes.

enable_A,enable_B,enable_C,enable_D:They represent the 4 enables for the 4 registers responsible for storing the outputs (A,B,C,D) to be inputs for the counters after checking that they are legal values.

It consists of 7 states:

The main idea: Is that we didn't receive the next digit from the user until he left his hand which is represented by (enable=0).

S1: is the 1st state which represents the initial state where the user didn't click on the keypad.

Once the user clicked on the digit on the keypad the (enable) of the keypad became 1 and the (Num) the user clicked with the enable became inputs to the FSM.

So it forces the next state to be S2 which sets the input(Num) to the output A and the enable(enable_A) becomes 1 which represents the enable of the register of A.

Then the (output A with the enable_A) becomes inputs to register A to store it until the user finishes entering the 3 digits left where the current state stays S2 as long as the enable equals 1 (This means the user is still entering the 1st digit).

Once the user left his hand to set the 2nd digit,the enable becomes zero so the next state is S3 where we will stay in this state until the user press on the keypad to enter the 2nd digit then the next state will be S4 which will give off the(output B with enable_B) to be stored in register B and so on till the user finishes entering C and D.

Important Notes about the drawing of the FSM for the readability:

1]Only the output that has enable =1 takes the value of the input(Num).

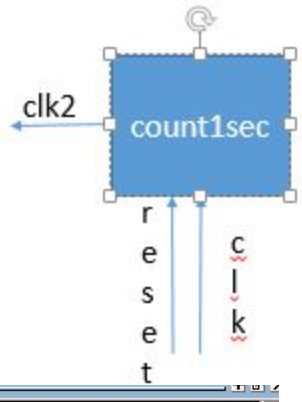
2]All the outputs that aren't mentioned in states preserve their previous values as there enables =0.

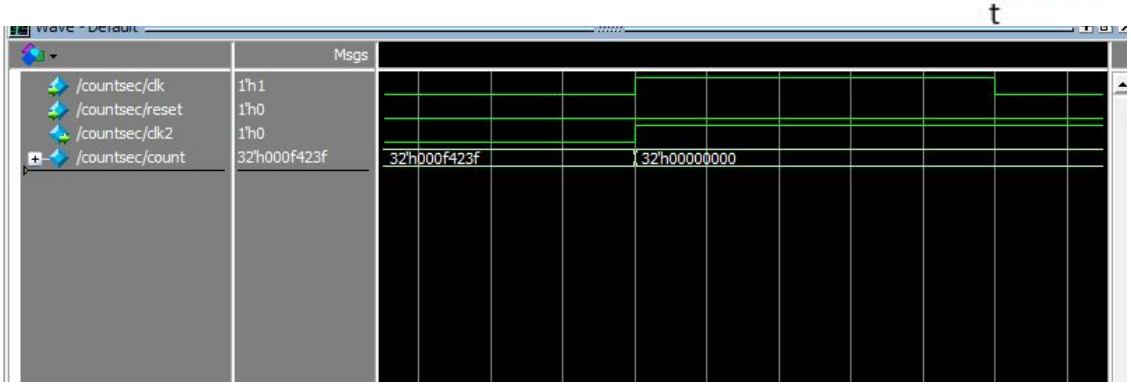
3]All the enables that weren't mentioned in the states =0.

Main Components:

1) Second counter:

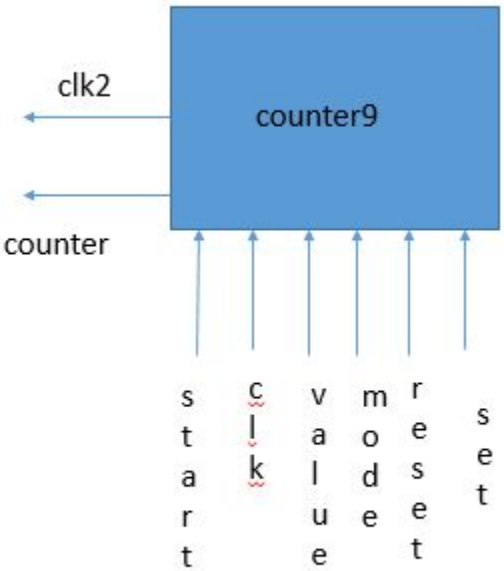
This counter has the input of clk which is on the lap of 1 mega second so that we should count 100000 so that we can count for a sec and it has a reset to reset the count and it has clk2 as output so that it gives 1 every time it counts till one second. And this simulation shows clk2 how it becomes one at the transition of two to three seconds.

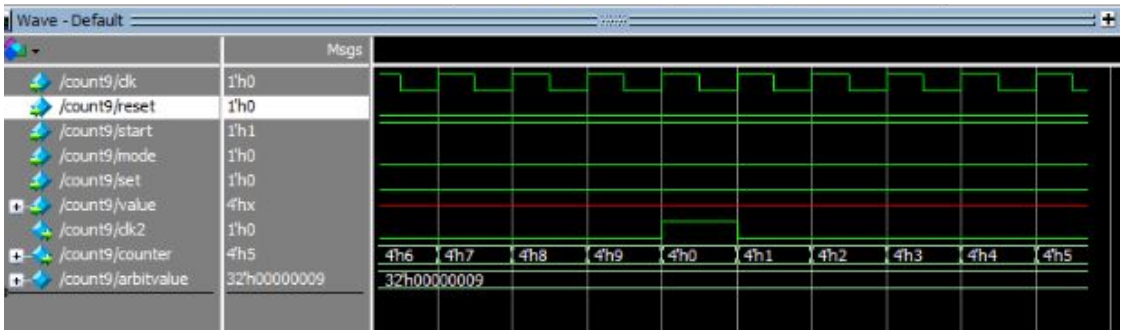


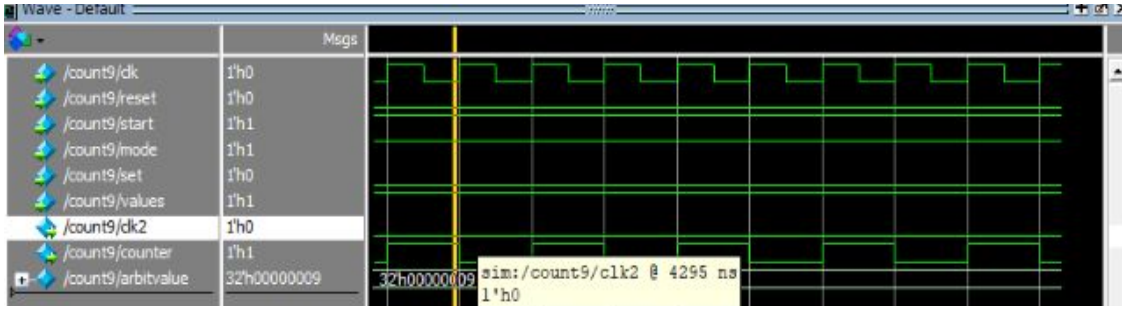


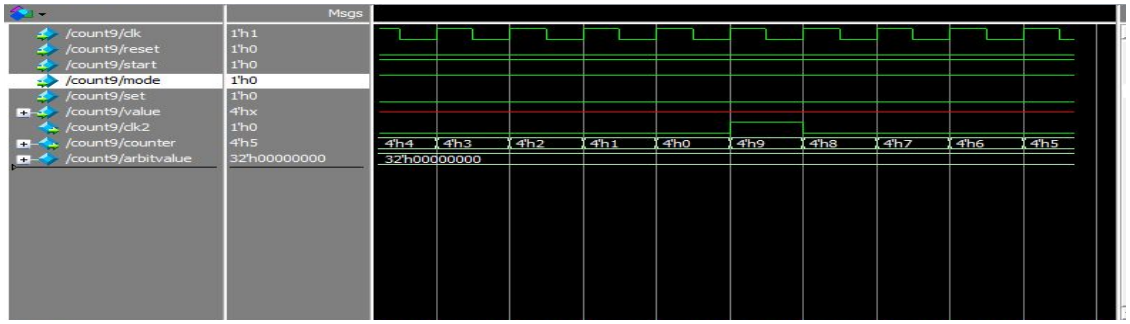
2)Counter 9:

This counter takes its clk from the clk output of seccounter and counts till 9(output) and when it reaches 9 it begins counting again from 0(in case of counting up). In case of counting down, it will count down from 9 to zero till reaching zero and back to 9 again. when its clk is 1 with start as input as if it is not 1 (stop) it will stop counting till it will be 1 again and it will continue counting since then. Reset makes it count from beginning again if it is mode 0(count up). Or from the set value if it is mode 1 (countdown). Set is also input if it is 1 it will initiate the value with the user input. This simulation shows how it count up and down with changing every input (reset,mode,start)



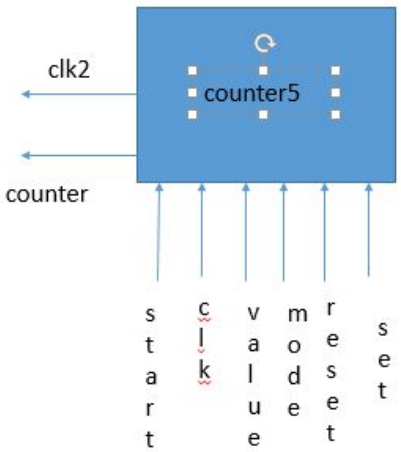






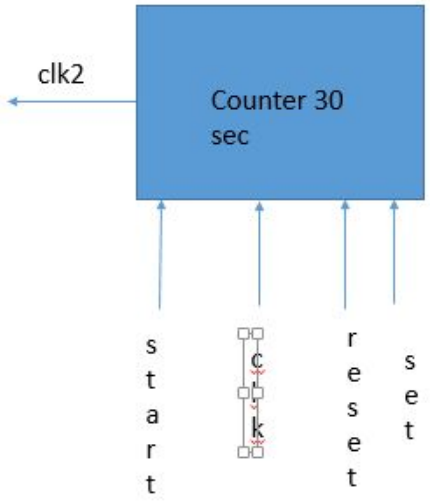
3)**Counter 5:**

This counter takes its clk from the clk output of count 9 and counts till 5(output) and begins from zero again when it's clk is 1(in case of counting up). In case of counting down, it will count down from 5 to zero till reach zero and back to 5 again. with start as input, if it is not 1 (stop) it will stop counting till it will be one again and it will continue counting since then. Reset makes it count from beginning again if it is mode 0(count up).

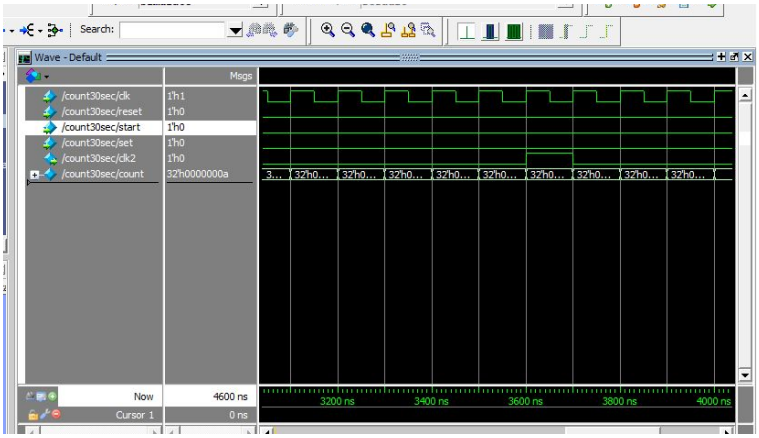


Or from the set value if it is mode 1 (countdown). Set is also input if it is 1 it will initiate the value with the user input.

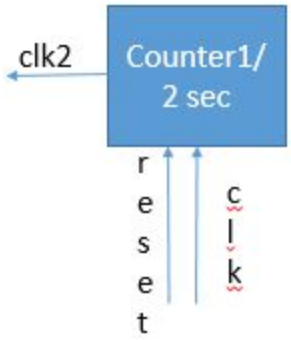
4) Counter 30 sec:



It takes its clk(output of the counter that counts one second) as an input and counts till reaching 30 and begins again counting). With setting and resting as the same function which will initialize it to zero again to achieve the functionality of our design. And start input to just start the count or stop and continue counting when start is 0. This simulation shows that it counts every 30 clk.

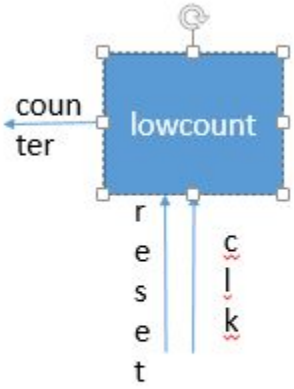


5)Counter half sec:

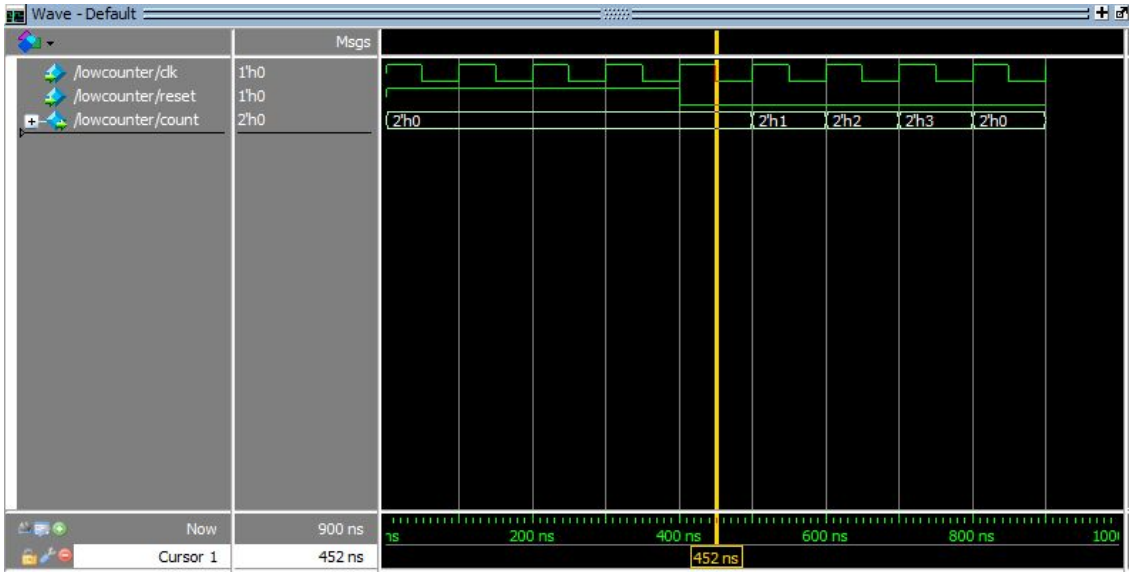


It is the same as the counter that counts one second but i edit the code to be just half the counter.

6)Low counter:



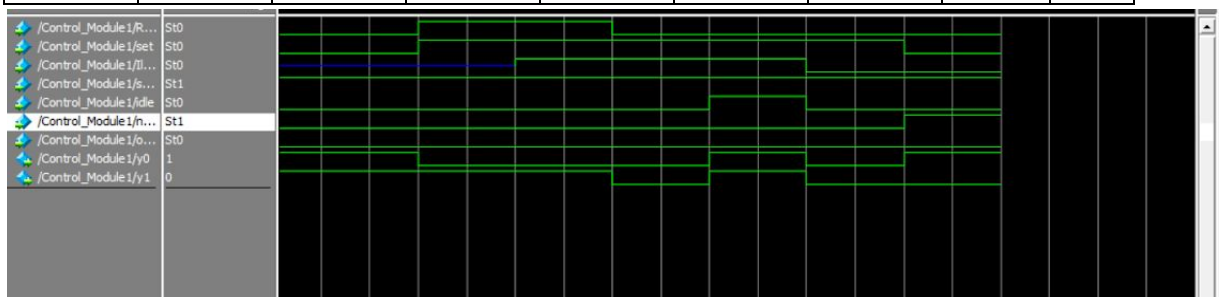
This counter has its input clk from seccounter and it should count for 4 different2-bit binary numbers as an output to be an input for the multiplexer responsible for screen saver mode. And this it's simulation that shows the transition from rest to be forced zero to 1 and how it counts.



7] Control Module(1):

Truth Table:

Reset	Set	Idle	Old Mode	New Mode Count up(1) Countdown (0)	start/stop start(1) stop(0)	legal/illegal legal(0) Illegal(1)	Y1	Y0
x	x	1	x	x	x	x	1	1
0	1	0	x	x	x	1	0	0
1	1	0	x	x	x	x	1	0
0	0	0	1	0	1	X	0	1
0	0	0	0	1	1	X	0	1
0	1	0	0	0	1	0	0	0
0	0	0	1	1	1	x	0	0
1	0	0	X	x	X	X	0	0



Its Inputs:(Reset, set, Illegal, start, idle, new_mode, old_mode)

Its Output: y (It is a 2-bit output which represents the 2 outputs Y0 and Y1)

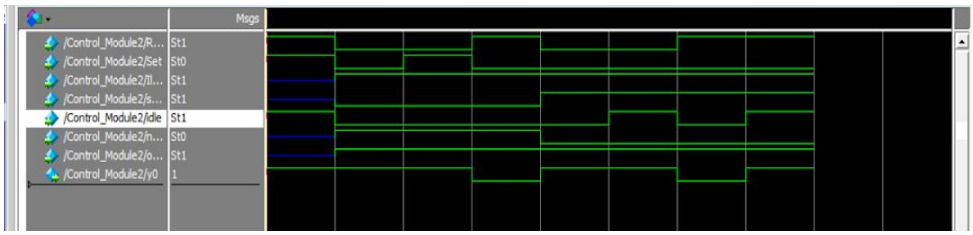
Its Functionality:

This is control module1 that represents a combinational circuit that indicates whether there are errors done by the user or not.it has the inputs(each is one bit): Reset,Set,Illegal(when illegal equals 0 it means legal),start(when start =0 it means stop),idle,new_mode,old_mode(which came from a flip-flop that stores it)and 2 outputs each one represents one bit.

8]Control Module (2):

Truth Table:

Reset	Set	Idle	Old Mode	New Mode Count up(1) Countdown (0)	start/stop start(1) stop(0)	legal/illegal legal(0) Illegal(1)	Y0
x	x	1	x	x	x	x	1
0	1	0	x	x	x	1	1
1	1	0	x	x	x	x	1
0	0	0	1	0	1	X	1
0	0	0	0	1	1	X	1
0	1	0	0	0	1	0	0
0	0	0	1	1	1	x	0
1	0	0	X	x	X	X	0



Its Inputs:(Reset,Set,Illegal,start,idle,new_mode,old_mode)

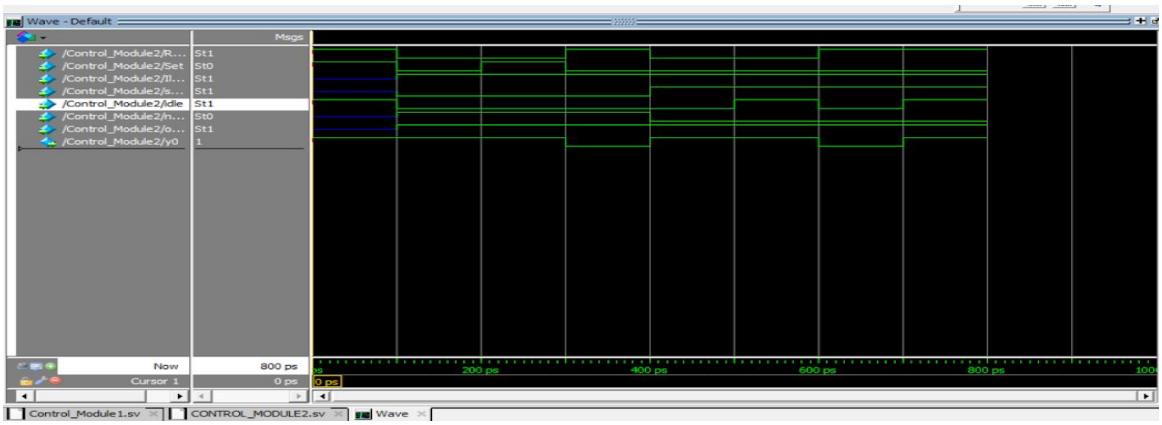
Its Output: (y0) which represents 1 bit

Its Functionality:

This is the control module that represents a combinational circuit to indicate whether it is an error or it is the segment saver.It has the inputs(each is one bit): Reset,Set,Illegal(when illegal equals 0 it means

legal),start(when start =0 it means stop),idle,new_mode,old_mode(which came from a flip-flop that stores it)and 1 output represents one bit.

9)D-Flip-Flop:



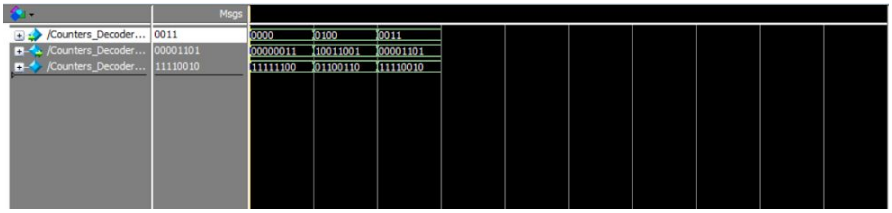
Its Inputs:(clk, Mode)

Its Output: Old_Mode

Its Functionality:

This D flip flop stores the value of the old mode to know when the user changes the mode whether It is the same or not to know whether it is an error or not.

10)Counters Decoder:



Its Inputs: [3:0] data

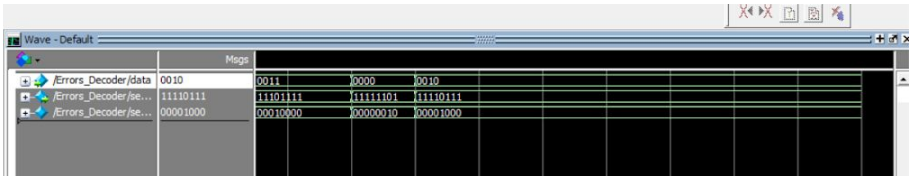
Its Output:[7:0] segmentsout (a,b,c,d,e,f,g,dp)

Its Functionality:

It represents the seven segment decoder used to display the digits on the screen where each digit needs specific segments to be flashed at the same time .

The 7 segment display is a common anode display so it represents the output when we give it 0 so we were forced to reverse the outputs of the decoder.

11)Errors Decoder:



Its Inputs: [3:0] data

Its Output:[7:0] segmentsout (a,b,c,d,e,f,g,dp)

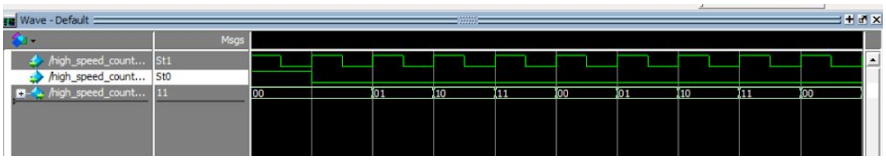
Its Functionality:

It represents the seven segment decoder used to display the errors and saver mode on the display.

Where saver mode is making each segment flashes at a time for each block of 7 segment display.

The 7 segment display is a common anode display so it represents the output when we give it 0 so we were forced to reverse the outputs of the decoder.

12)High Speed Counter:



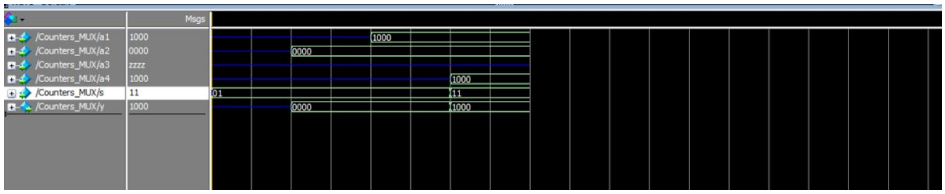
Its Inputs: clk, reset

Its Output: [1:0] out

Its Functionality:

This is the counter connected with the multiplexer of errors where it has high speed to toggle between the bits of each multiplexer fast and it counts from 0 to 4 then after that starts from zero again.

13)Counters Multiplexer:



Its Inputs: [3:0] a1,a2,a3,a4 (which represents the 4 digits entered by the user)

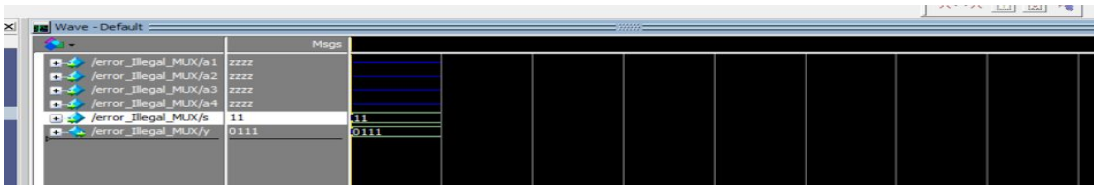
[1:0]s (which represents the selector)

Its Output: [3:0]y (which represents one of the 4 digits)

Its Functionality:

This mux is used to choose a counter as an input via the selector.

14)Illegal error MUX:



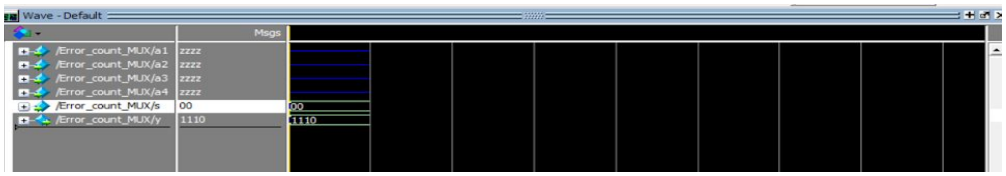
Its Inputs: [1:0] s (which represents the selector)

Its Output: [3:0] y

Its Functionality:

This mux is used to display the error []:59 on the display in case of setting time to an illegal value.

15) count error MUX:



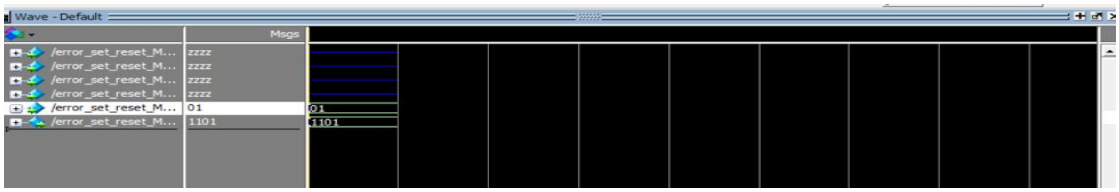
Its Inputs: [1:0]s (which represents the selector)

Its Output: [3:0] y

Its Functionality:

This mux is used to display the error [-E:01] on the display in case the user attempts to change the counting mode while counting is in progress.

16)Set/Reset error MUX:



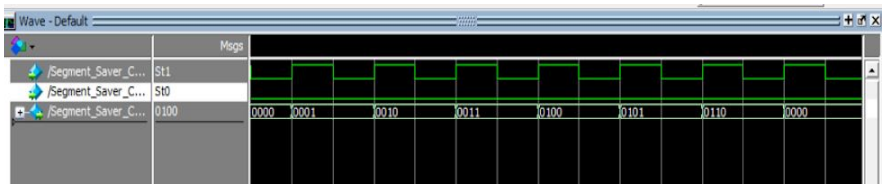
Its Inputs: [1:0]s (which represents the selector)

Its Output: [3:0] y

Its Functionality:

This mux is used to display the error S0:00 on the display in case of setting and resetting at the same time.

17)Segment Saver Counter:



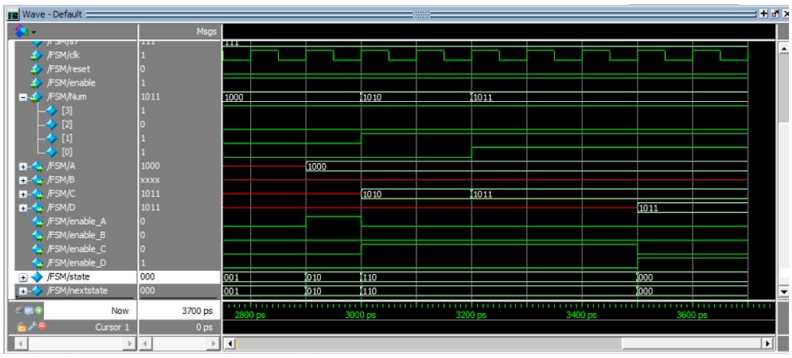
Its Inputs: clk, reset

Its Output: [3:0] out

Its Functionality:

This is the counter responsible for the segment saver where it counts from 0 to 6 where each number represents one segment of the 7 segments of the display.

18)FSM:



Its Inputs: clk,reset,enable

[3:0] Num

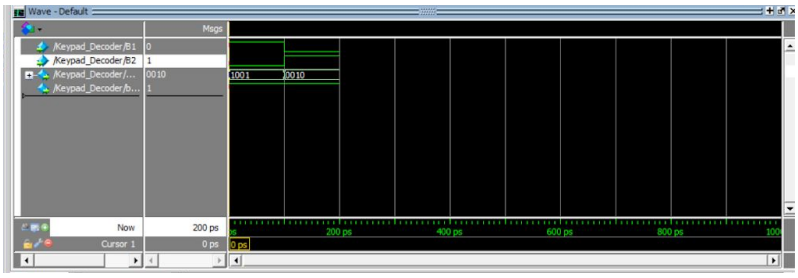
Its Output: [3:0] A,B,C,D;

enable_A,enable_B,enable_C,enable_D (each of them represents 1 bit each of them is the enable for one of the registers to store the 4 outputs)

Its Functionality:

It is the module responsible for taking the input digits(which represents the second and minutes) from the user when he switches on the set switch to set the number he wants to start from it the countdown.

19)Keypad Decoder:



Its Inputs: B1, B2 (each of them represents one button that indicates a special number either 9 or 2)

Its Output: [3:0] Num (either 9 or 2)

buttonPressed(1 bit signal that indicates a signal that a button is pressed or not)

Its Functionality:

We did it today while we were testing the FPGA as the keypad module is out of the scope of the course so we couldn't make it. We thought of using 10 buttons where each one indicates a number but we didn't find 10 buttons in the lab so we used the only 2 buttons in the FPGA to test the set in case of legal (29) in case of illegal (92).

20)Register:

Its Inputs: clk,enable

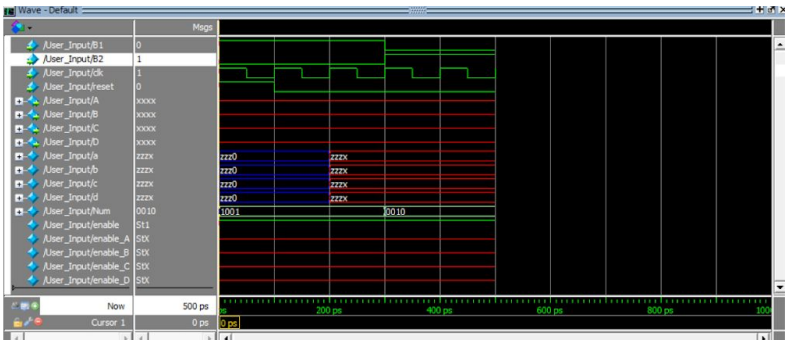
[0:3] in

Its Output: [0:3] Num

Its Functionality:

It stores the digit that the user enters at a time.

21)User Input:



Its Inputs: B1,B2,clk,reset

B1,B2:indicates the button that the user clicked on

Its Output: [3:0] A,B,C,D

Its Functionality:

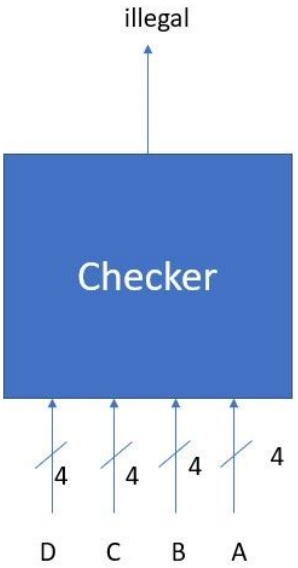
It is a module that relates the keypad decoder with the FSM and the 4 registers that store the 4 outputs A,B,C,D.

22)Module (Checker)

This module has 4 bit 4 input (A,B,C,D) and one bit output

It Checks whether A or exceeds 9 or not ,B or D exceeds 5 or not and it

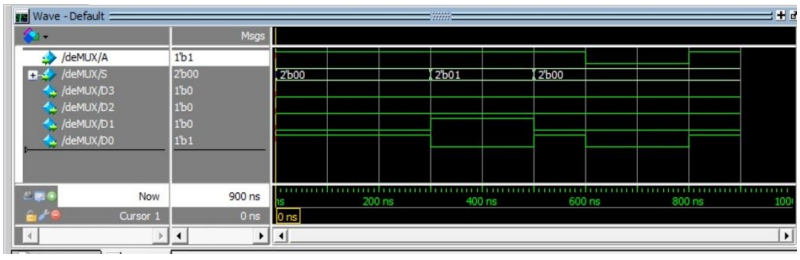
gives value 1 for output “illegal”.

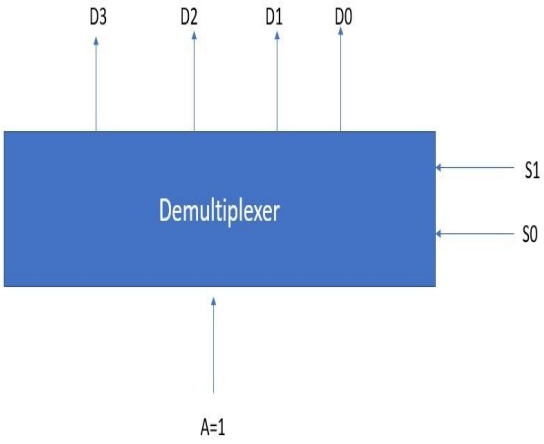


23)Demultiplexer

It has one input initialized to 1 and this input connect to one output based on selection lines

These outputs are connected to 4 digit- 7 segment displays to defne which digit should be illuminated.



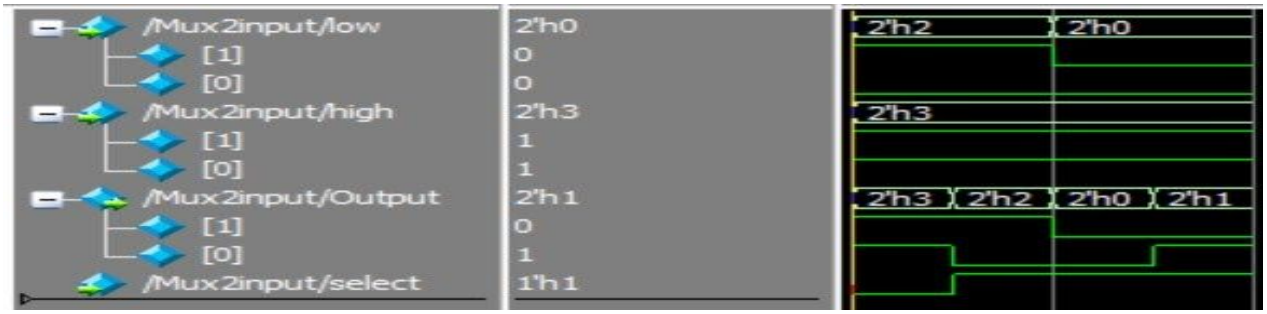


Mux3,Mux4,Mux5,Mux6

Mux3: Has 4 input each is 4 bit ,4 bit output 2 bit selectors and defines which input would connect to the output based on selection lines coming from Module 1.

Mux4: select between errors decoder or counters decoder to be connected to the segment display based on signal comes from Module 2.

Mux5: Select between inputs comes from counters with high speed repetition to display all inputs to the user quickly so he can not notice this transformation between digits of segment display, selectors lines come from

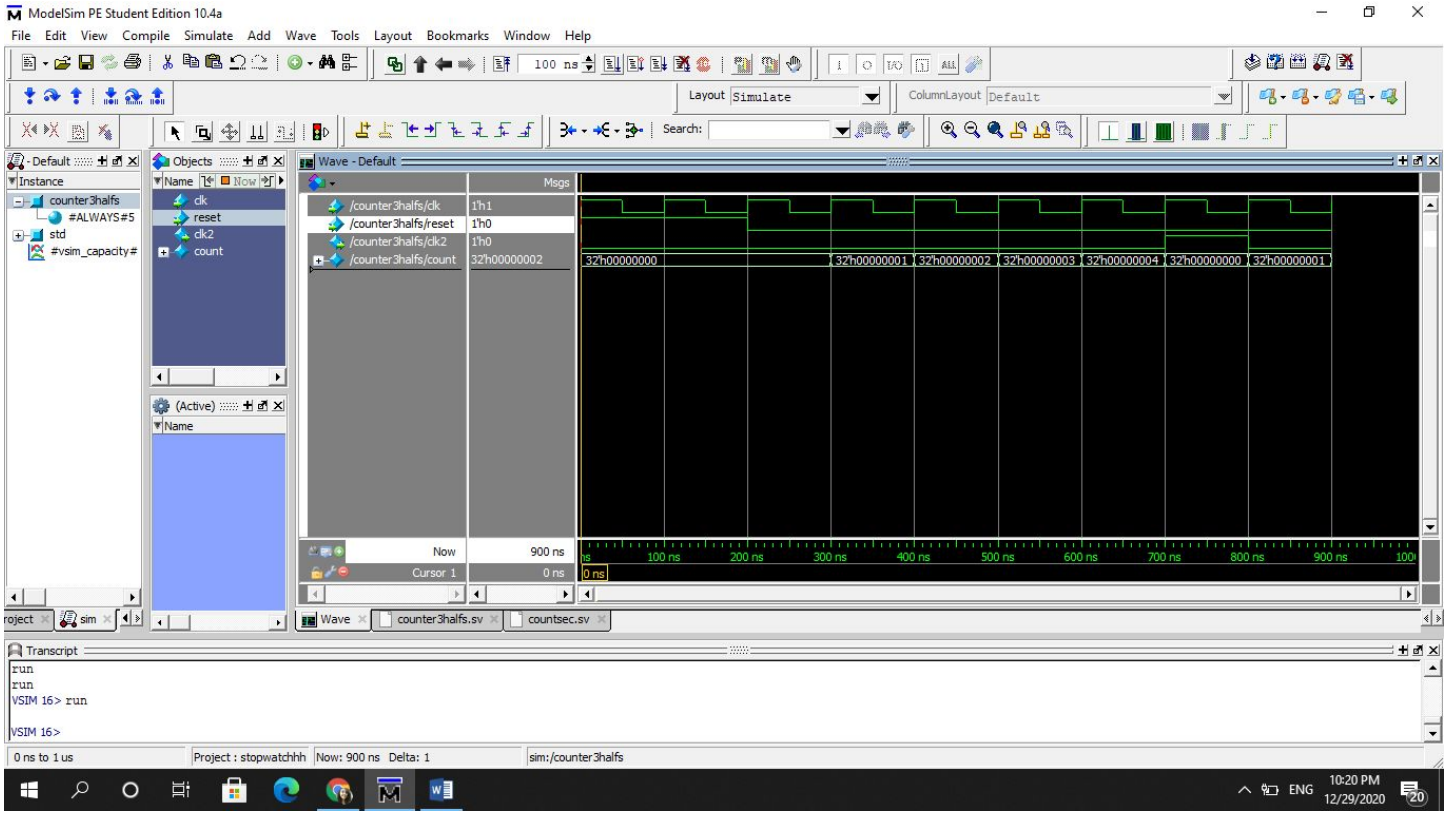


counter from 0 to 3.**Mux6:**select between high speed counter in case of errors, counting up or counting down and low speed in only one case (segment saver mode).

<u>24) count3halfs:</u>

This counter takes its clk1 input from the counter that counts one second and counts till 4 and begins from zero again and it gives 1 for clk2 after reaching 4 and zero otherwise. reset as a second input that then reset the counting.

the simulation is when reset is forced to be 1 then we set it to be 0 and as shown it counts till 4 and returns to 0 again.



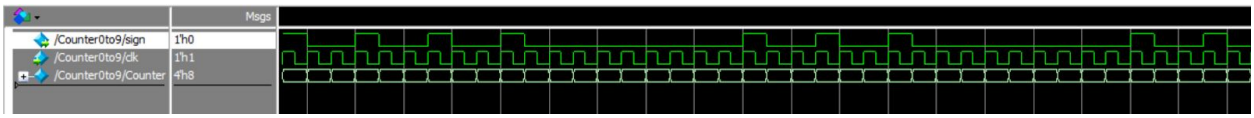
25)**TristateBuffer:**

This module is associated for illegal value Error in order to flash 3 times

It comes from the counter, this counter is counting from 0 to 15 and gives 1 when input divisible by 3.

26)Module :Counter 0 to 9

This counter is counting from 0 to 15 but it gives signal when the number of counting divisible by three



conclusion:

By the end of this project, we gained a lot of experience designing a stop watch with a variety of options and designing a circuit that performs our target in general. We also gained a great experience and knowledge of hardware language – system Verilog- and how we can transform our design to a working code and to simulate it. Furthermore, we learned how to deal with FPGA and how to deal with its errors. Finally, our project gives us the opportunity to deal with great experience to connect between the digital course outcomes and real life and it gives us the experience of working in a team with improving our skills in different areas for every team member.We tried to load our code on the FPGA but unfortunately we spent about 5 hours and there were many technical issues that faced us.One of them,was the keypad. Also,while loading the code on the FPGA it mentioned some pins as inputs but they were internals.So,we didn't manage to load our code on it as we couldn't solve all the technical issues that faced us(there was no time).

effort distribution:

Norhan part:

I read the project document one day after its announcement and i begin first to think about the design with only fsm before watching the lecture of counters then i begin to think with what is the input we have as a whole stopwatch and its output then i begin the design block by block from its basics (counter, 7 segments decoder) this took one week till reaching semi final design and i start coding the counters in parallel. Parallel with it i start to think how we will divide our report.Then i took the feedback from the dr before the deadline with nearly 4 days then we begin to integrate our ideas together thursday. By reaching the final design. I started coding two complete days as it took me too much to handle errors (syntax and logic in modelsim) and parallel with it I started to write the reports of every block I am responsible for (nearly all the counters in the design). After turning phase 1 by 3 days, I started meeting with my team to develop the rest of the design regarding the keypad and fsm responsible

for taking the input to be setted from the user. Then,we met to integrate the code together for nearly 3 days straight then till the end of the deadline we were dealing with errors that we discovered with simulation and I was responsible for the added counters but this time it took me 10 minutes only for each added counter. In parallel with it I was responsible for reporting main parts such as : abstract , introduction, conclusion drawing half of the final design and this took me nearly 6 hours on different days because dealing with the design tool took a time for me. At the end I was responsible for the formatting of the report to integrate all the written parts to be in the same format.

Rodina part:

Time taken to code the modules:

Control Module(1): It took me about 2 hours overall to finish it as I made changes in the equations in many days to reach the best representation.

Control Module (2):It took me about one hour to finish it as I made a lot of changes to it on different days.

D-Flip-Flop:It took me about 15 minutes to finish it.

Counters Decoder:It took me about 1 hour to finish it as I made changes on it .

Errors Decoder:It took me about 45 minutes to finish it as I made changes on it .

High Speed Counter:It took me about 15 minutes to finish it.

Counters Multiplexer:It took me about 30 minutes to finish it.

Illegal error MUX:It took me about 30 minutes to finish it.

count error MUX:It took me about 20 minutes to finish it.

Set/Reset error MUX:It took me about 15 minutes to finish it.

Segment Saver Counter:It took me about 15 minutes to finish it.

FSM:It took me many days to think of it and design it but finally I got the idea from the FSM practice problems which were posted before the last quiz.

Keypad Decoder:It took me about 15 minutes to finish it.

Register:It took us about 15 minutes to finish it.

User Input:It took me about 30 minutes to finish it.

For the 1st time I read the document of the project,I felt it is impossible to do that project but after reading it many times and doing online meetings with my team I began to sense that it could be executed.We made a lot of online meetings to think of the design that we will do a certain functionality, at each time we distributed the tasks between us.After each meeting I spent time in thinking of the code for each of my modules then I discuss what I reached with my team and they gave me feedback.

The 2 control modules took a lot of time thinking about them with my team members till we reached their right way of implementation according to our design then I wrote their codes.

I thought of the error (set / reset) as a possible error from the user where he may switch on the set and reset switches at the same time.Then I suggested that error to my team and they welcomed it.

Drawing the design of the FSM on an online program took about an hour from me.

Mariam part:

Drawing design and solving issues was my main part, i wrote less in the modules code as i was facing problems in modelsim, Also i was responsible for integrating all modules in one module called stop_watch.

Designing stop_watch with all its features was the most challenging part and it took too much time to consider all features with best practice as much as we could. Also integrating modules in one module took time more than expected because it seemed that we needed extra modules to finish it. However, writing codes is the easiest part and takes less time than other responsibilities.

Tristate Buffer :5 minutes

Multiplexer:each takes nearly 5 minutes.

counters 0 to 9 take 15 minutes.

drawing design: nearly one hour.