Aryan Rao

CPRE 281 – Section 11

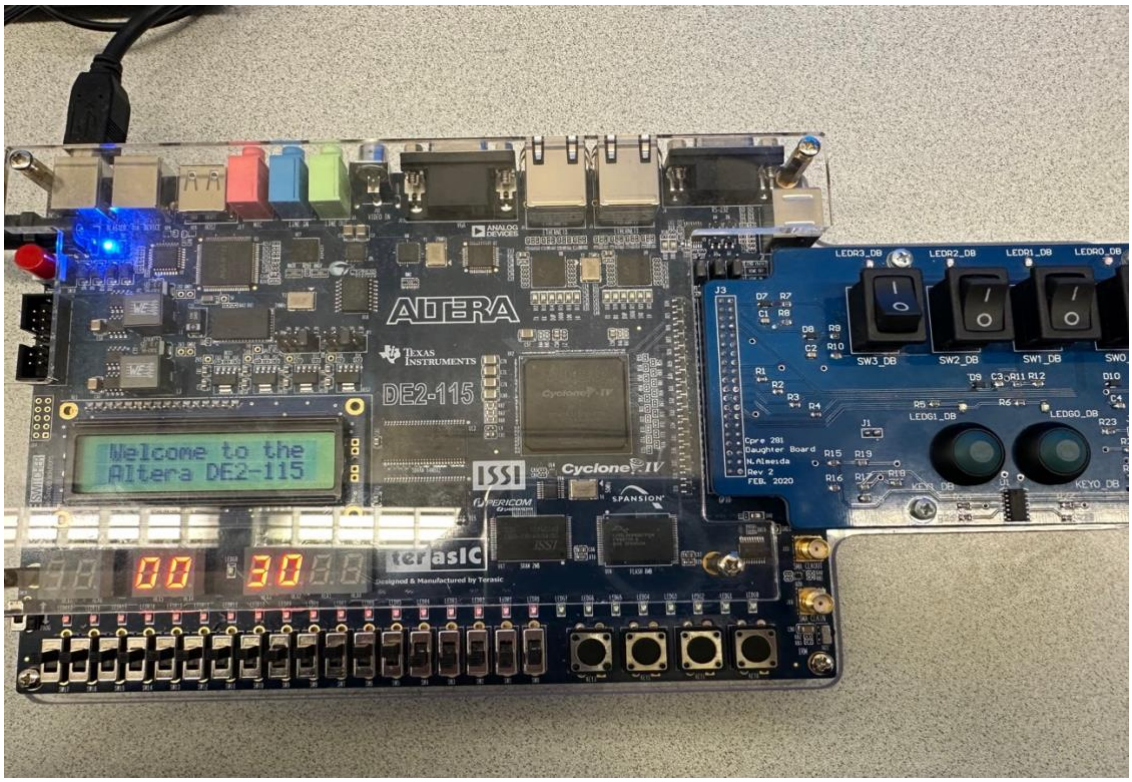
ID: 264954748

FINAL PROJECT REPORT

STOPWATCH

Overview:

For the final project of my CPRE 281 course, I decided to make a stopwatch. The stopwatch includes basic features of starting and stopping. There's also the feature of reset built in. To make use of register files which basically means storing memory, I implemented them so that a total of 4 laps could be easily stored. This project took a lot of time but in the end, I am really satisfied the way it turned out.

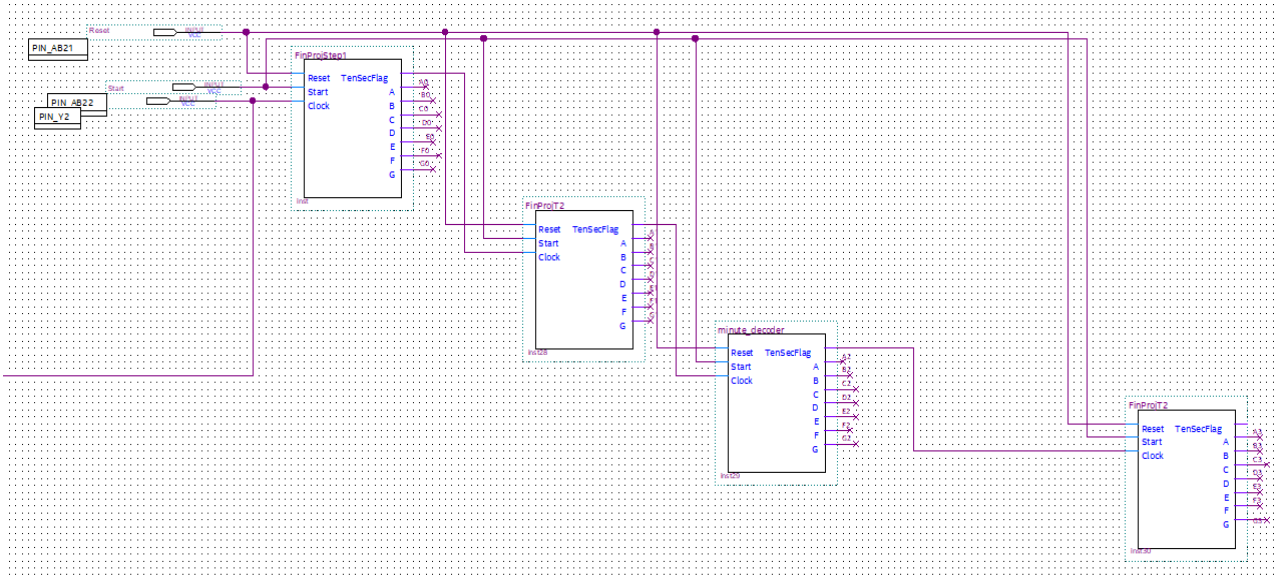


List of elements used:

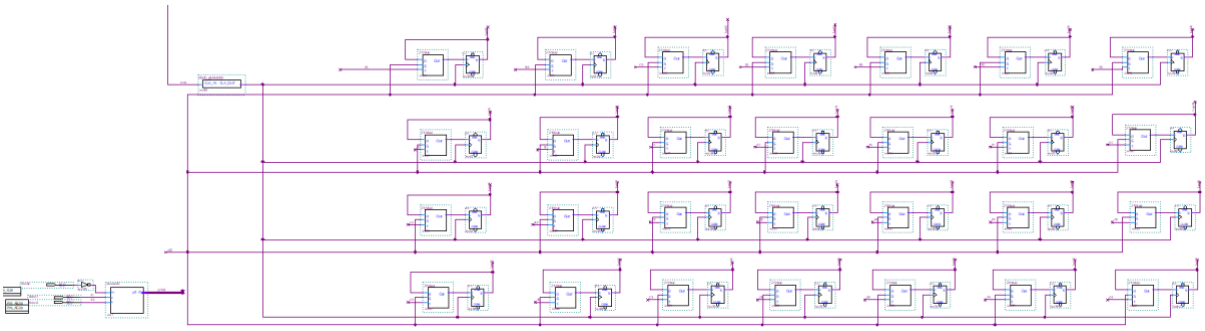
- D – Flip Flop
- 2-1 MUX
- 4-1 MUX
- OR, AND, XOR, NOT, NAND Gates
- Clock generator, clock divider
- 2-4 Decoder
- Seven segment decoders
- 7-bit register
- Register Files
- Finite State Machine

Design Images:

This is the basic design image of the stopwatch. The first and the third counter from top will both count from 0-9. The only difference between them is that the first one has a clock generator inside. The second and the fourth one both count from 0-5.

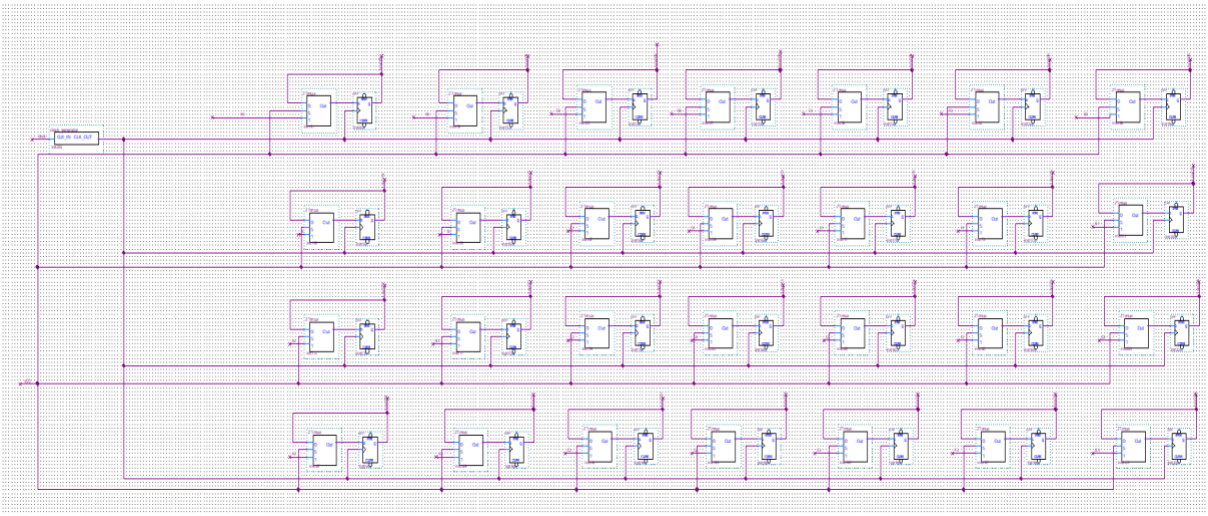


The second part of the design shows a register file with a 2-4 decoder which is a combinational circuit that converts binary information from n input lines to a maximum of m=2^n unique output lines. The register files itself is a means of storing memory and in this case, it stores a time lap.



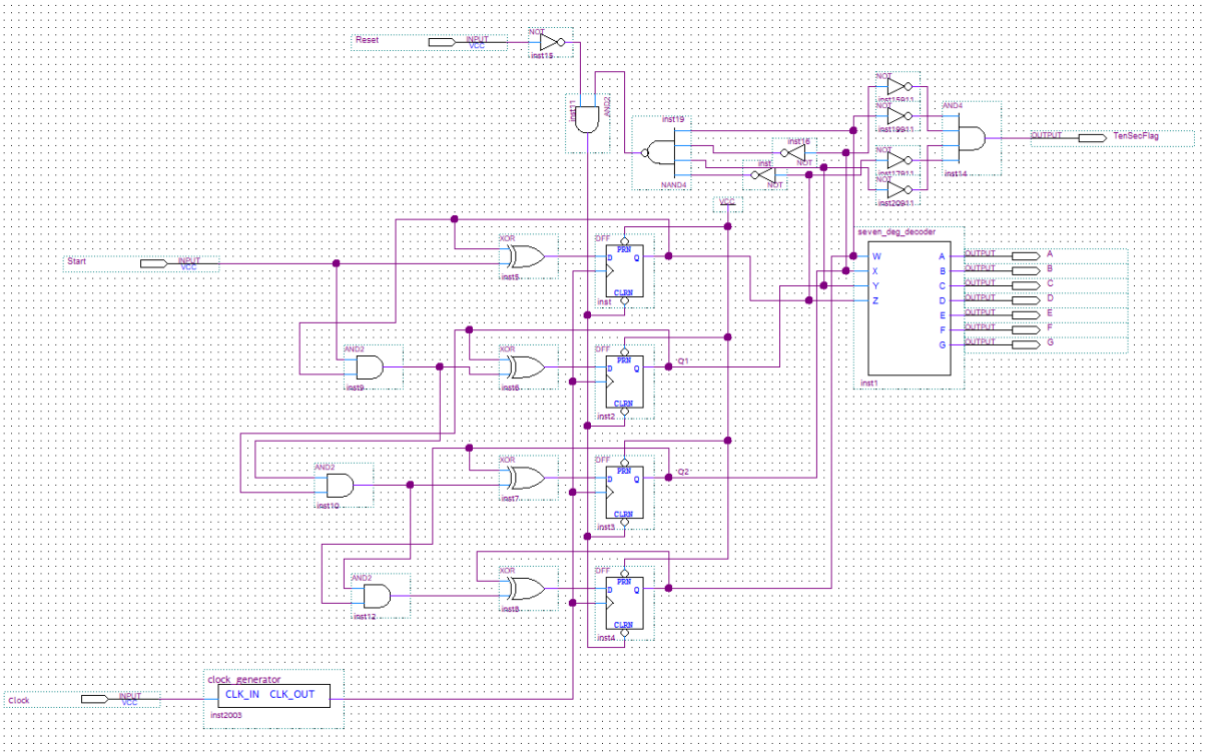
Register Files:

As mentioned above the work of a register file is to store memory. In this case, the file is implemented using a combination of 2-1 multiplexers and D-flip flops. The register files store time laps and there are 4 files similar to the one shown in this image which do the same.

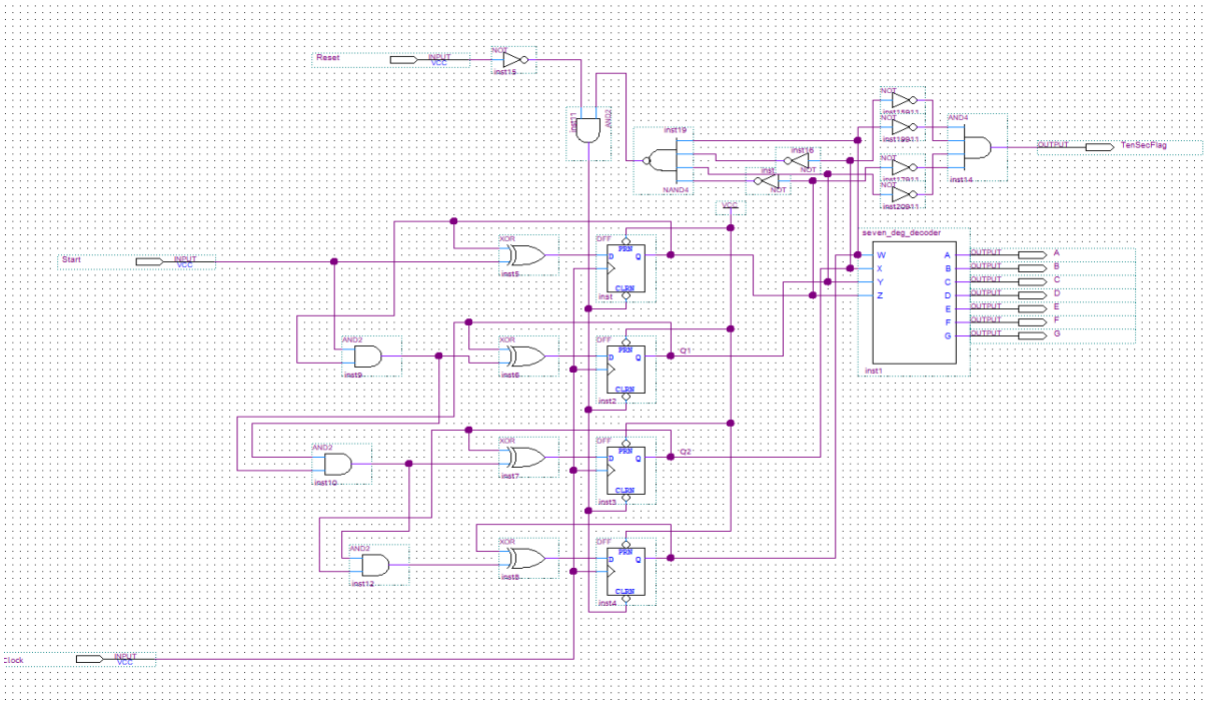


Step 1:

Step 1 basically includes an up-counter that counts from 0-9. It is implemented using a combination of AND, XOR, NOT, NAND GATES, and D- Flip flops. There's a start and reset inside as input lines which gives user the ability to decide.

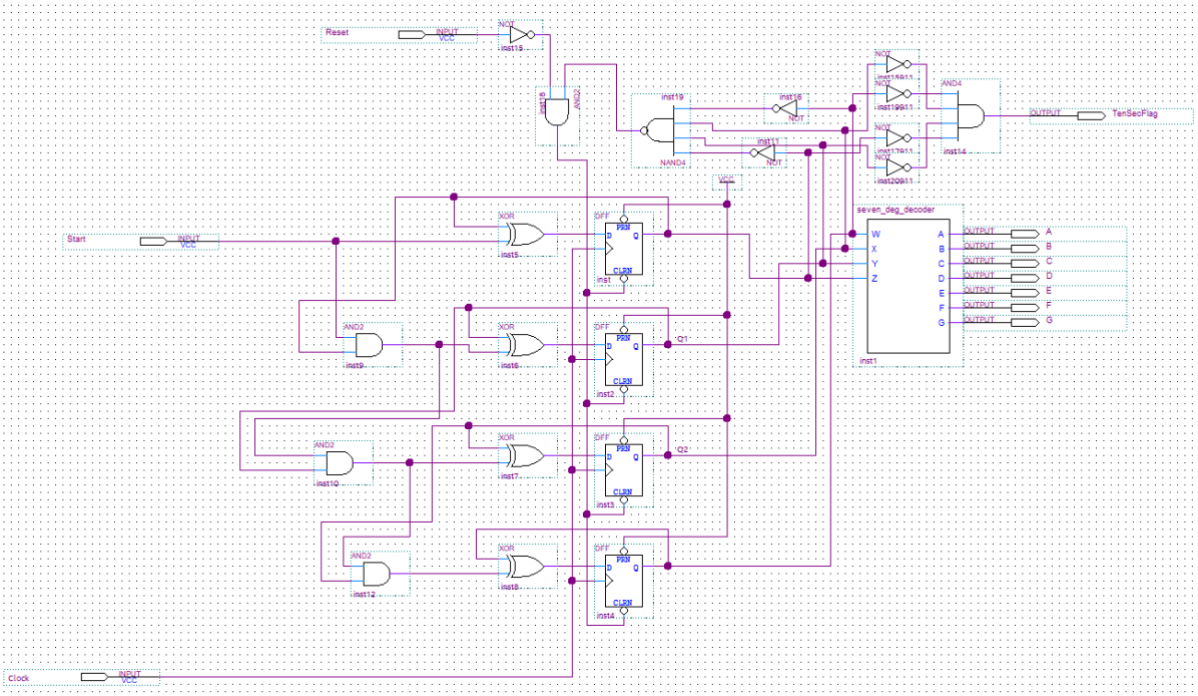


THIRD COUNTER: Similar to the one above except the only difference is that this one does not have a clock generator inside.



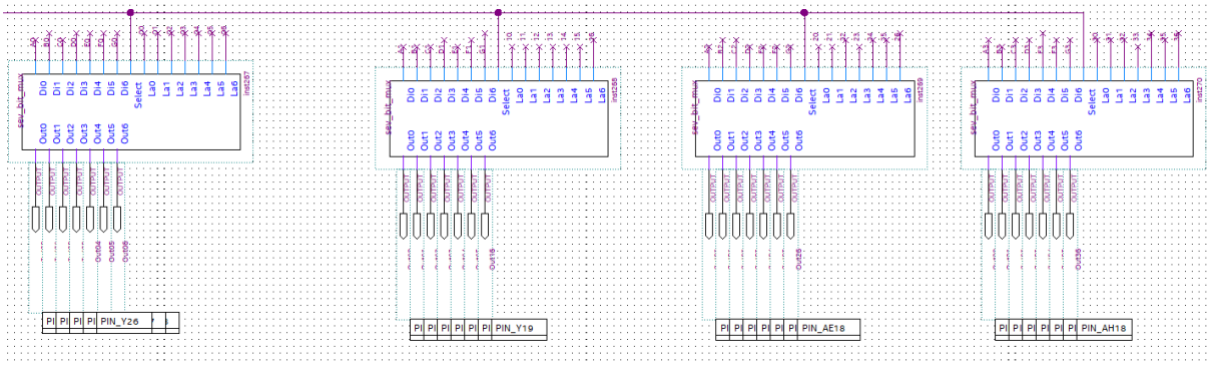
Step 2:

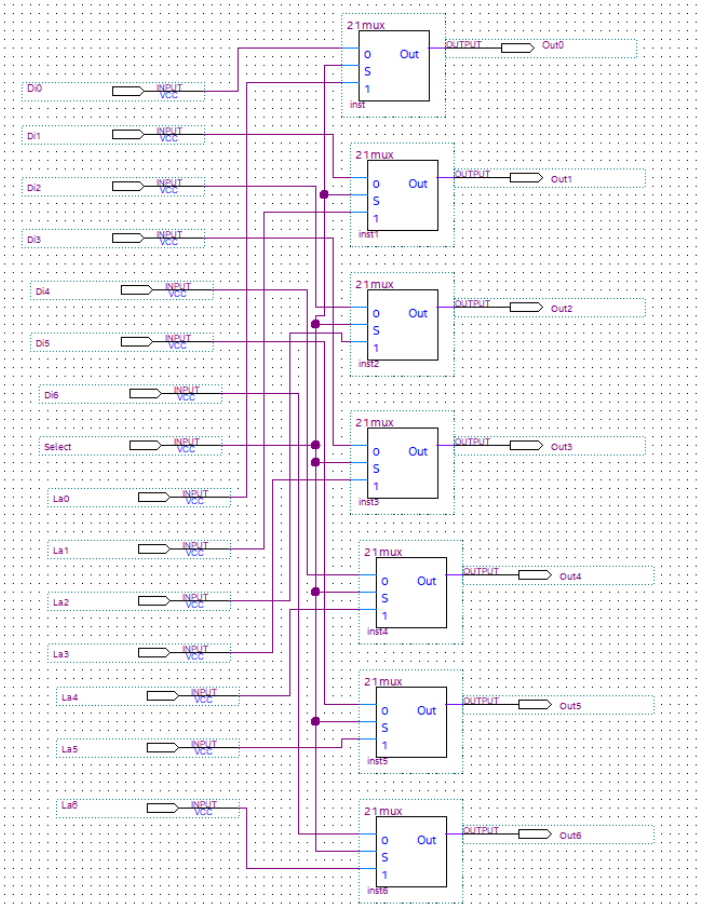
Step 2 also includes an up-counter that on the other hand counts from 0-5. It is implemented using a combination of AND, XOR, NOT, NAND GATES, and D- Flip flops. There's a start and reset inside as input lines which gives user the ability to decide when to do the needful.



7-Bit Multiplexer:

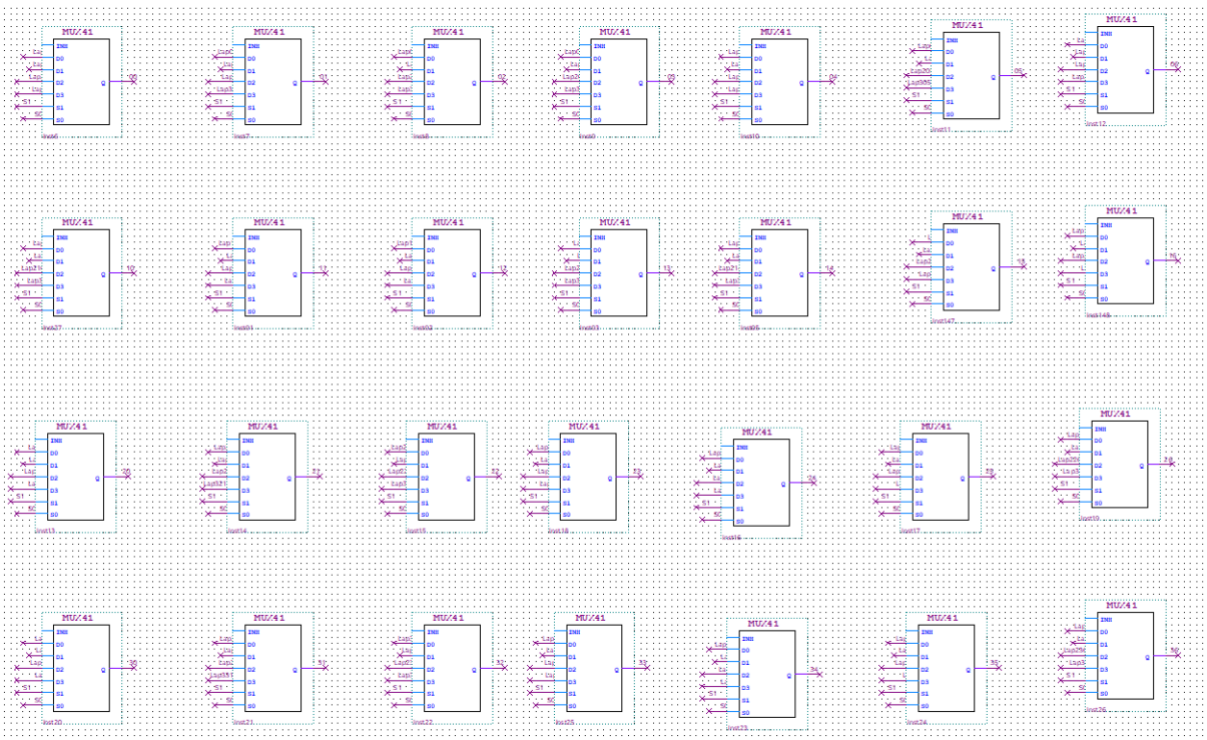
I implemented this 7-bit multiplexer by myself. The functionality of this is to display the time laps store d in the register files.

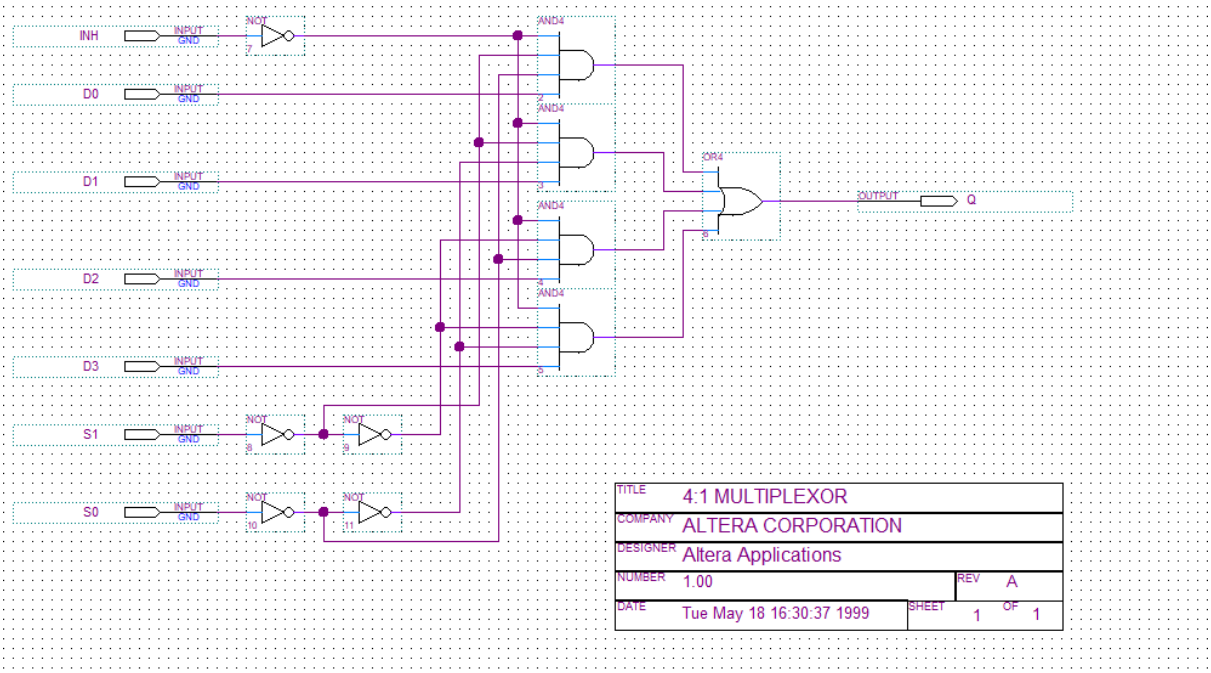




4-1 Multiplexer:

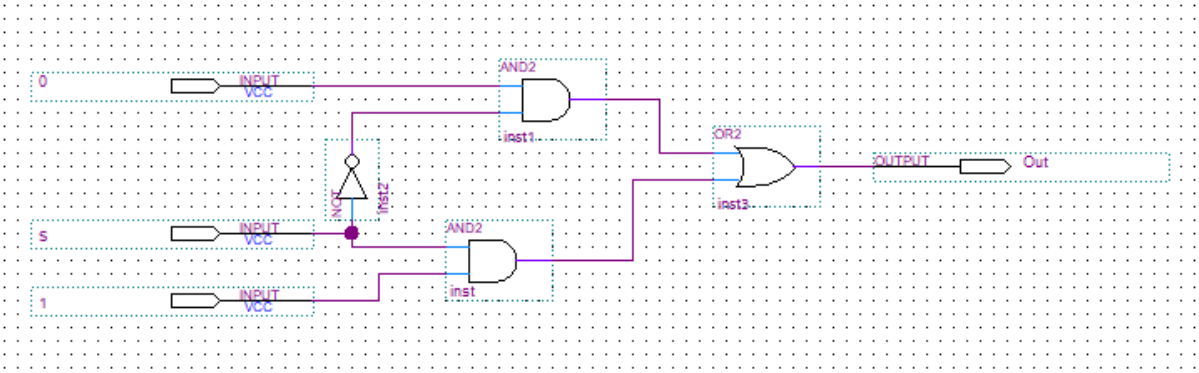
I used the built-in 4-1 multiplexer in Quartus. The purpose of this is to let the user select which time lap they want to display. I will explain how to do so later in this document.


MUX41


INH
D0
D1
D2
D3
S1
S0
INPUT
GND
NOT
AND4
OR4
OUTPUT
Q
TITLE
4:1 MULTIPLEXOR
COMPANY
ALTERA CORPORATION
DESIGNER
Altera Applications
NUMBER
1.00
REV
A
DATE
Tue May 18 16:30:37 1999
SHEET
1
OF
1

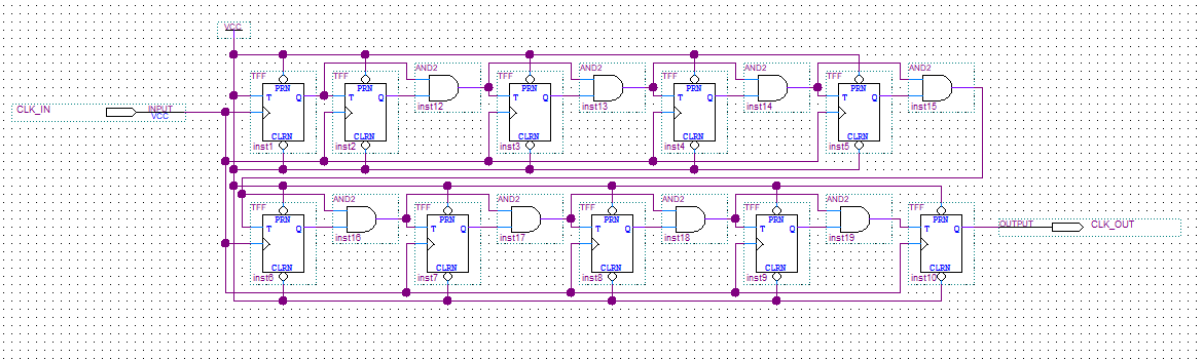
2-1 Multiplexer:

I made my own 2-1 multiplexer and it's been used often in the project, so I thought to include this. 2-1 multiplexer consists of two inputs 0 and 1, one selects input S and one output out. Depending on the select signal, the output is connected to either of the inputs. Since there are two input signals, only two ways are possible to connect the inputs to the outputs, so one selects is needed to do these operations

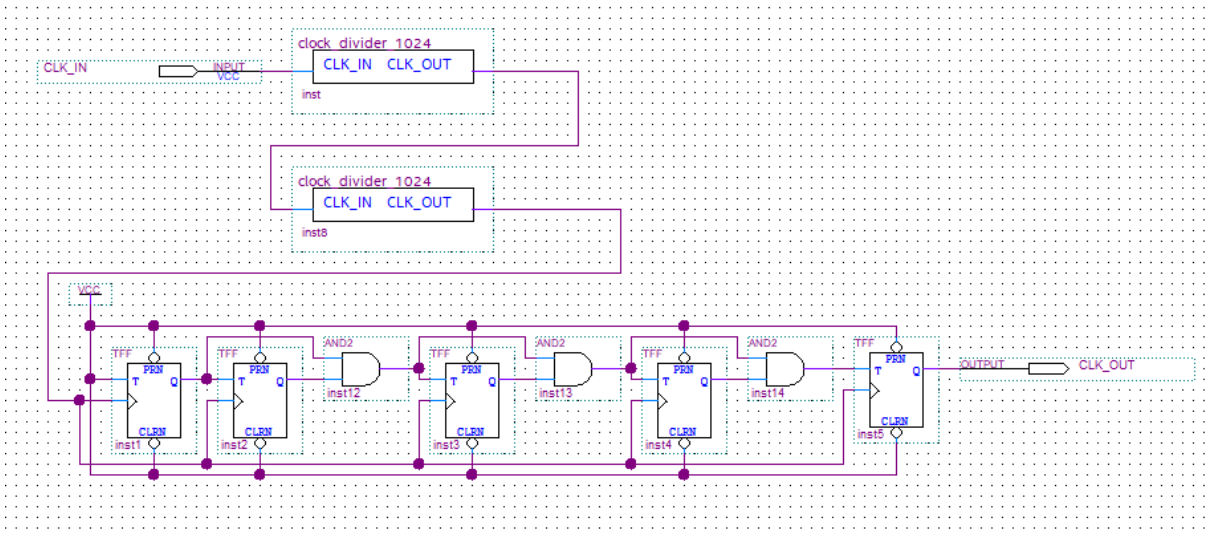


Clock Generator & Divider:

The clock divider consists of 10 T- flip flops. This basically increases the board clock period by ~10 ms. It is implemented using a combination of AND gates and T- flip flops.

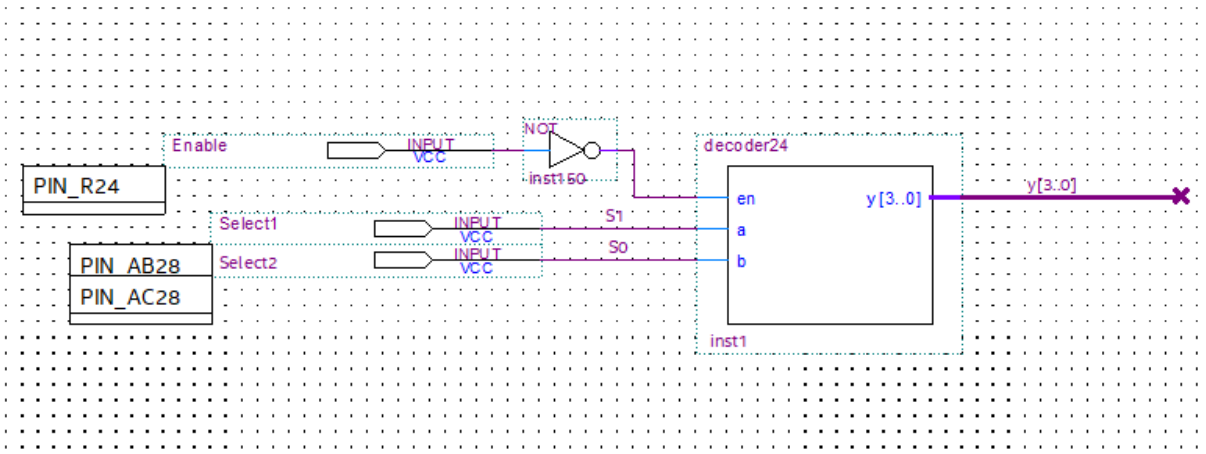


The clock generator is made using a combination of 2 clock divider blocks and another 5 T-flip flops along with AND gates. The clock generator will use the 50MHz clock signal and reduce it to a clock signal with a period of about 670 milliseconds (ms).



2-4 Decoder:

I made the 2-4 Decoder myself using Verilog. As mentioned earlier,
A decoder is a combinational circuit that converts binary information from n input lines to a maximum of m=2^n unique output lines. The decoder has an enable which determines whether it's in action or not. The other two inputs, Select1 and Select2 are used as select lines which basically tells which lap to store. If it's the first lap, then (00) meaning both inputs/switches will be 0/off. And for example, for storing the third one, (10) the first input/switch will be 1/on and second will be 0/off.



```
module decoder24(en,a,b,y);

   input en,a,b;


   output reg [3:0]y;


   always @(en,a,b)
     begin

       if(en==1)
         begin
           if(a==0 & b==0) y=4'b0001;
           else if(a==0 & b==1) y=4'b0010;
           else if(a==1 & b==0) y=4'b0100;
           else if(a==1 & b==1) y=4'b1000;
         end
      else
         begin
         y = 4'b0000;
         end
     end
endmodule
```

Finite State Machine:

STATE P: Stopwatch starts

STATE Q: Reset stopwatch

STATE R: Stopwatch stops

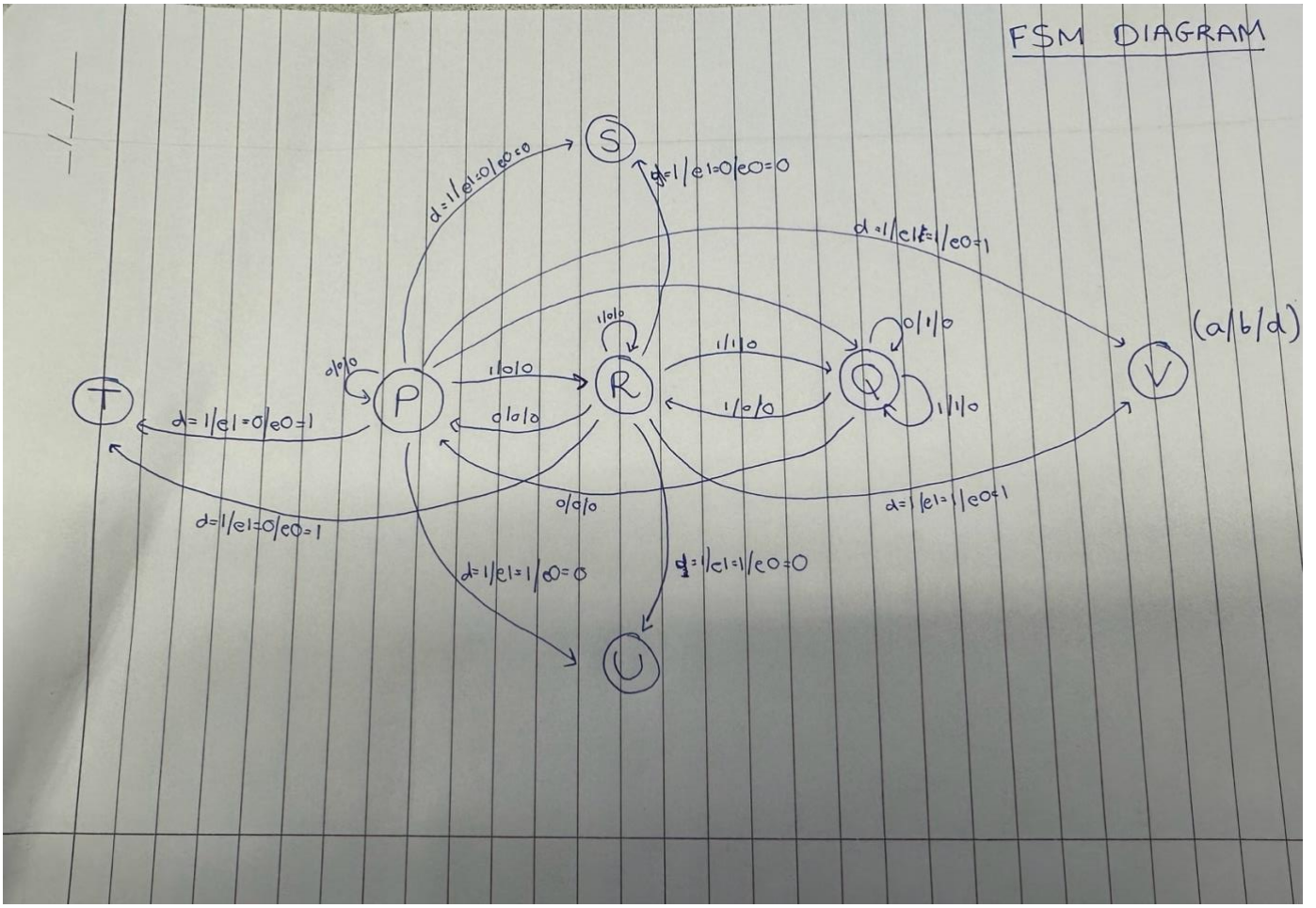
STATE S: Lap time 1

STATE T: Lap time 2

STATE U: Lap time 3

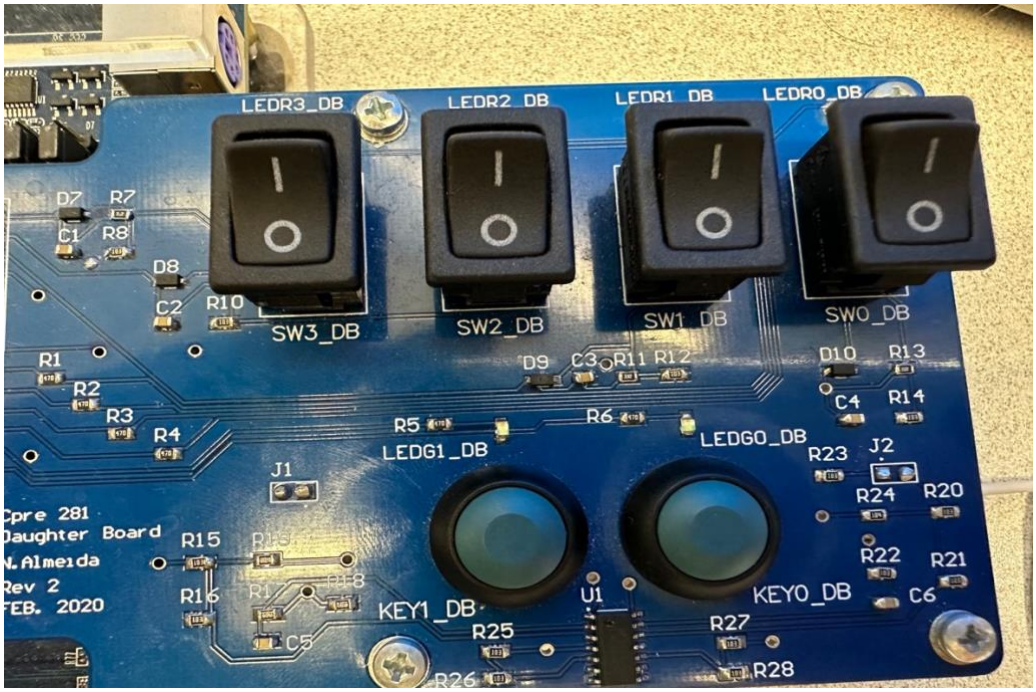
STATE V: Lap time 4

Inputs: a, b, c, d, e1, e0

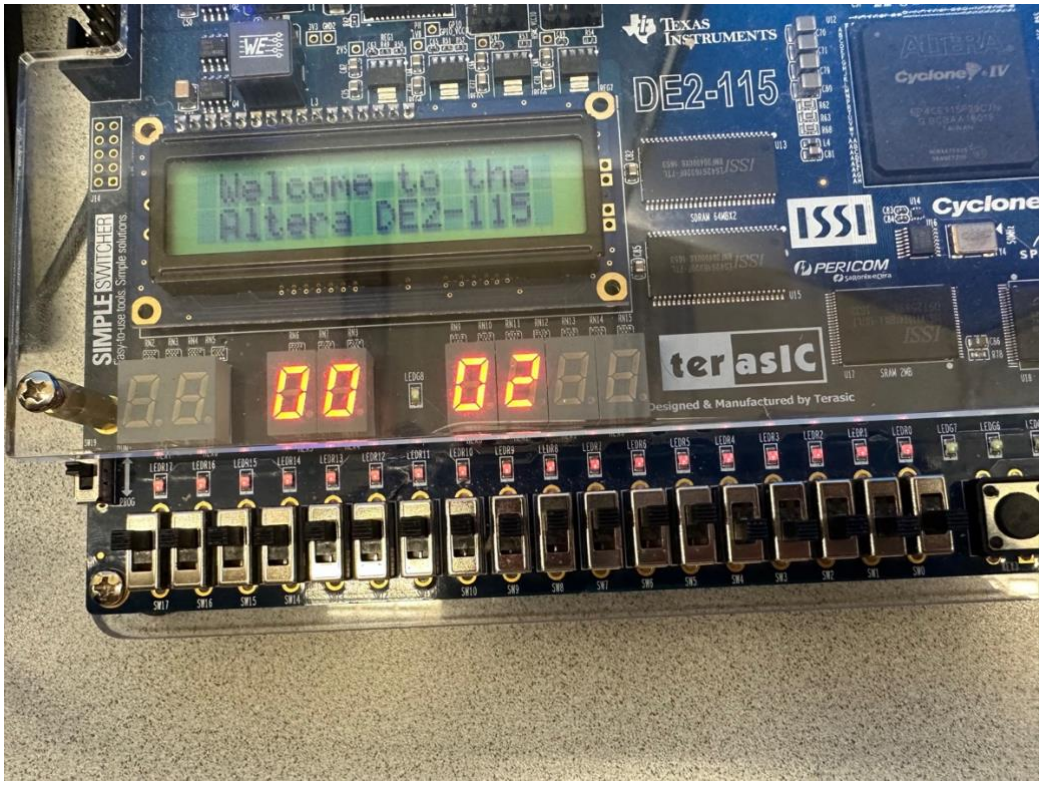

FSM DIAGRAM
S
T
P
R
Q
V
U
(a/b/d)
d=1/e1=0/e0=0
d=1/e1=0/e0=0
d=1/e1=1/e0=1
1/0/0
1/1/0
0/1/0
1/1/0
1/0/0
0/0/0
d=1/e1=0/e0=1
0/0/0
d=1/e1=0/e0=1
d=1/e1=1/e0=1
d=1/e1=1/e0=0
d=1/e1=1/e0=0

Working:

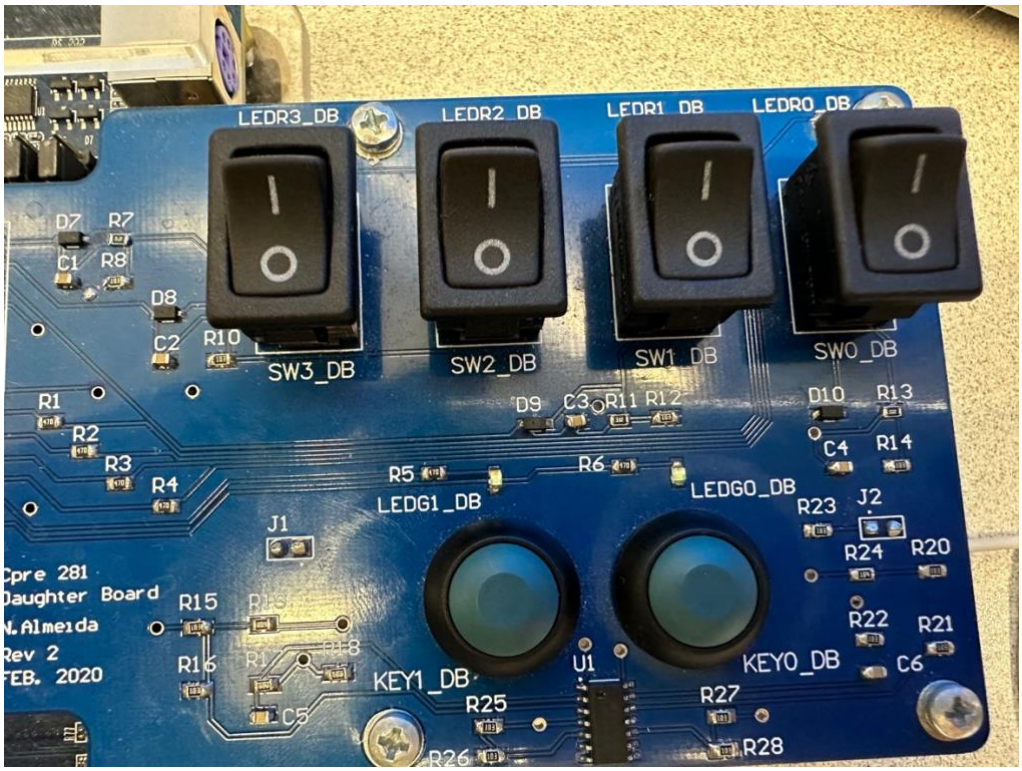
For just starting the stopwatch, one has the to press the first rocker switch from right (SW3_DB) on the FPGA board.

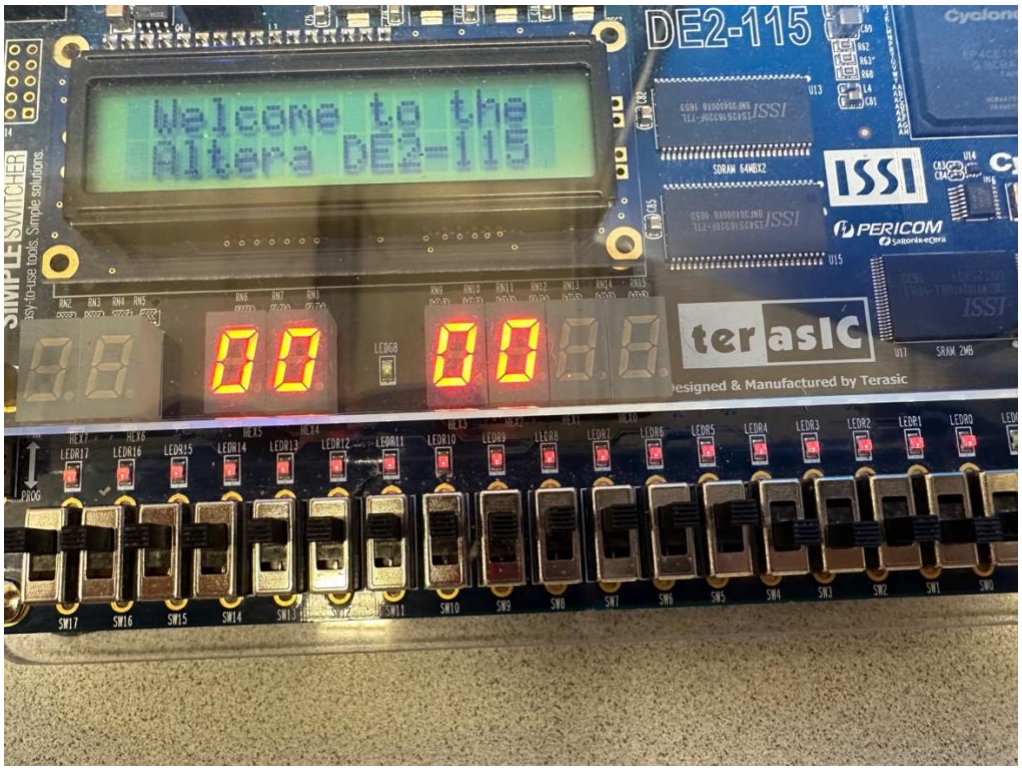


This will start the watch from 00:00 and keep counting.



For resetting the stopwatch, that means to make it count from 00:00 again, one must press the second rocker switch (SW2_DB)



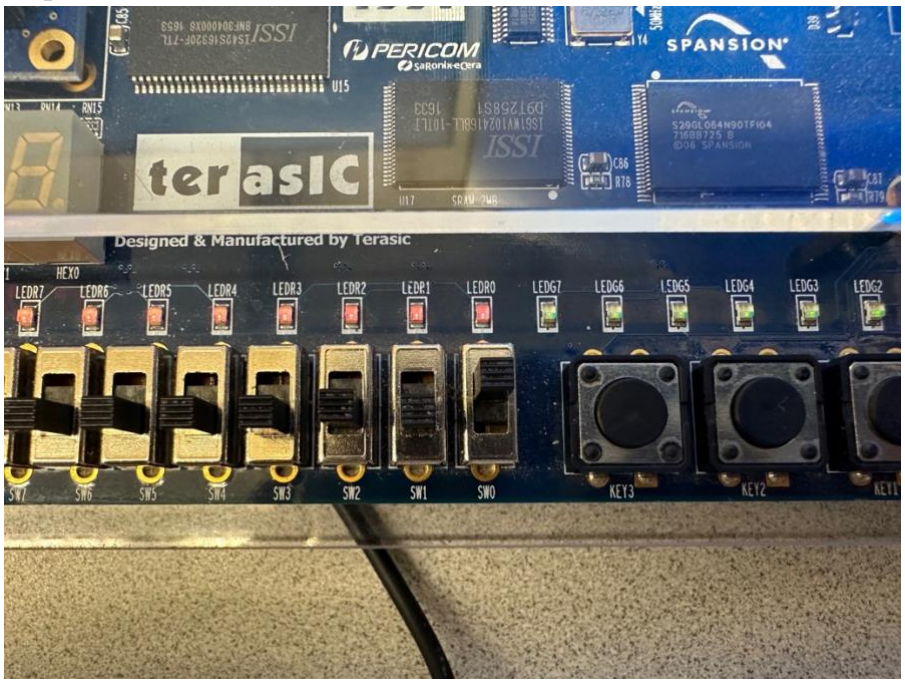


If both the rocker switches (SW3_CB and SW2_DB) are turned on, the clock will just stay on 00:00.

Next is lapping. For lapping the slide switches (SW0 and SW1) and the push button (KEY3) come into play.
Here are images to show the arrangement of slide switches to store the respective number of lap inside:

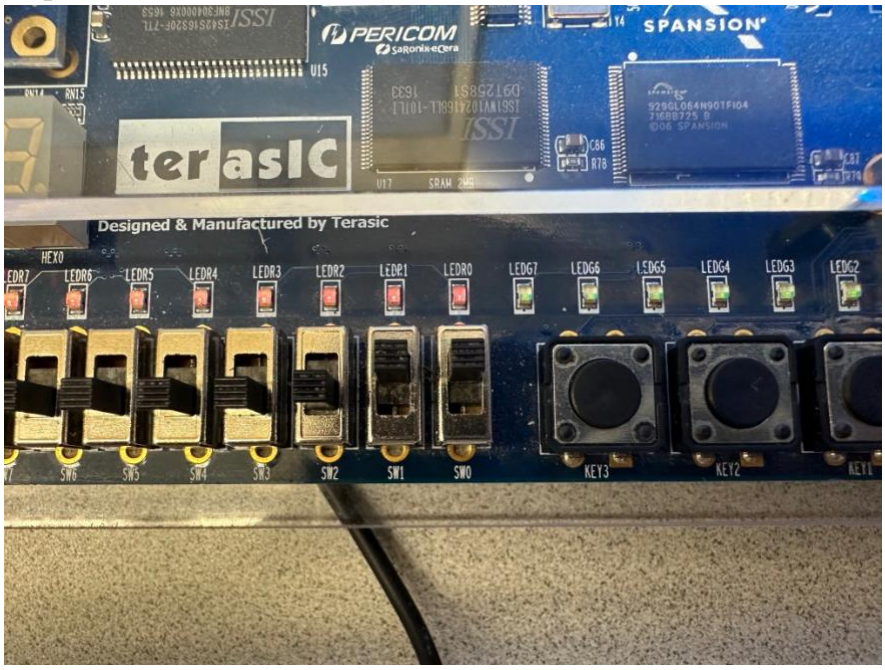
Lap #1: (00)

Lap #2: (01)



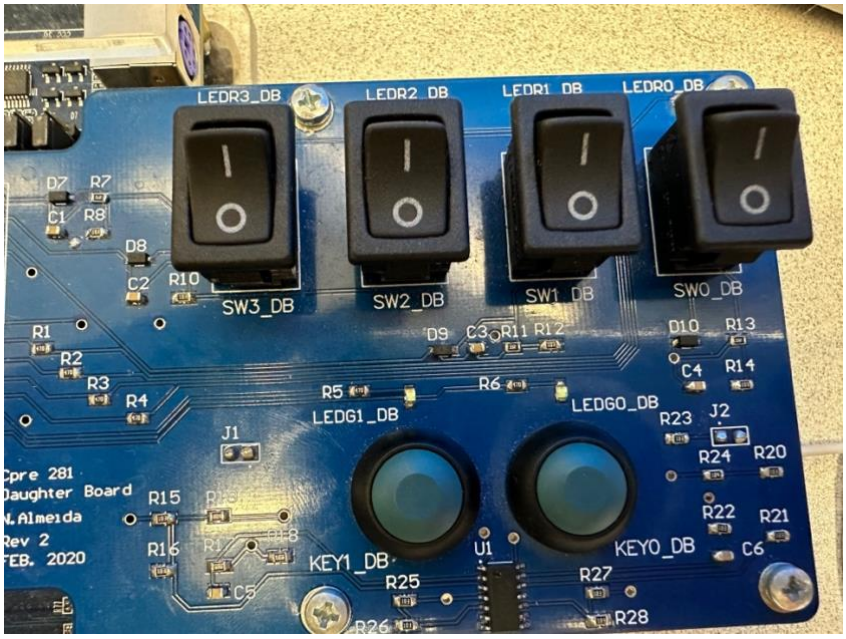
Lap #3: (10)

Lap #4: (11)



Now when you have the lap you have want to store in mind with the switch selection, you have to press the push button (KEY3) to store the respective lap. This will keep storing subsequent lap times in the register files and can even overwrite them when done.

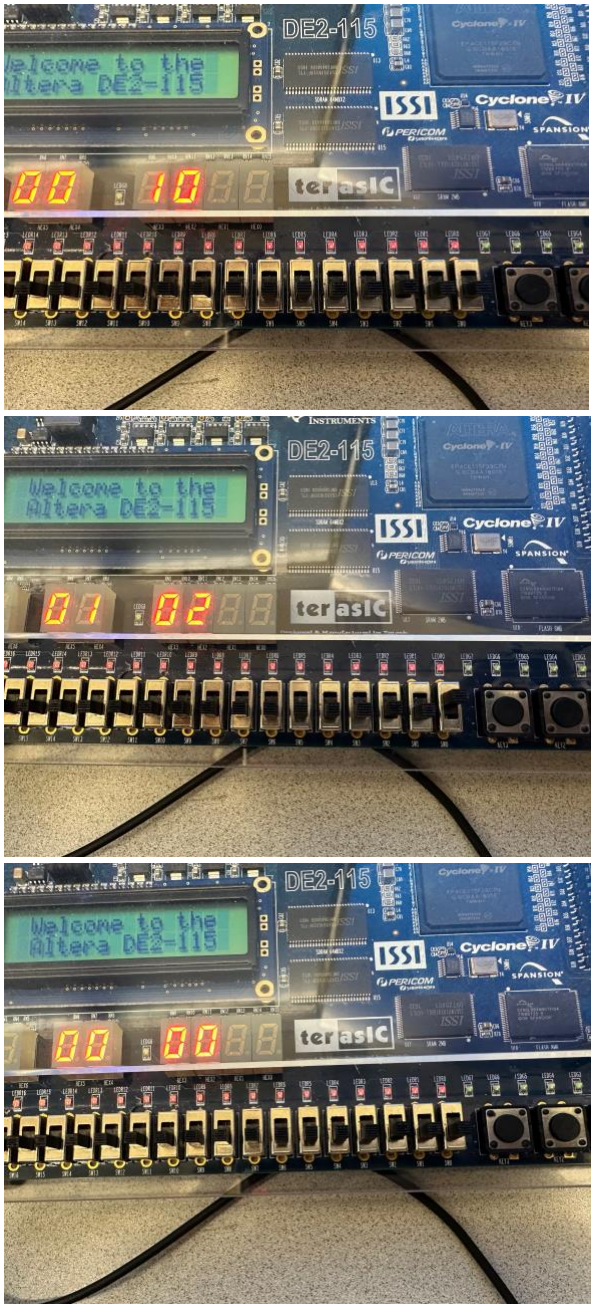
When we have stored our laps, the next step will be displaying them. This is an easy process. We can turn the display mode on easily by turning on the last rocker switch (SW0_DB).



The lap you want to see again depends on the arrangement of slide switches shown above. If you want the see the first lap, turn both slide switches off. If you want to see the second lap, turn the second one on and the first off and so on. It is important to turn the rocker switch (SW0_DB) on for the same.

Sample Run:

I turned the stopwatch on using the rocker switch. (SW3_DB) I put both the slide switches at 0 and 0 and then lapped the first time at 10 seconds using push button (KEY3). Then I put one slide switch (SW0) to 1 and second one was at 0 only and lapped the second time at 1 minute 02 seconds. Then I turned on display mode using SW0_DB and saw the times lapped. I later reset the clock back to 00:00.



This project was really a challenging and a time-consuming one. But I am proud of the effort I put in and the result. There is always room for improvement here and there. I would like to thank Professor Stoytchev for the way he guided all of us and would specially thank my TA's Erik and Steven for their constant support and mentorship.