Digital Stop Watch

M1 M0 S1 S0



Team members:

* Ahmed Bassam Abdelaal 202101074
* Mohammed Sayed 2021004384
* Khaled Ashraf Mousa 202100751

Table of Contents

1. [Design Goals](#Design_Goals)
2. [Brainstorming and chosen Approaches](#Brainstoming)

[2.1 Stop watch with Synchronous counter](#two_one)

[2.2 Stop watch with asynchronous counter](#two_two)

[2.3 Stop watch with registers and adders](#two_three)

[2.4 Rating design approaches](#two_four)

1. [Chosen Design Logic and Implementation](#three)

[3.1 )4 Counters](#three_one)(

[3.1.1 BCD Up & Down Counter](#three_one_one)

[3.1.2 0to6 Up & Down Counter](#three_one_two)

[3.2 Clock Divider](#three_two)

[3.3 Operation Control Unit](#three_three)

3.3.1 BCD Adder

3.3.2 BCD Subtractor

3.3.3 BCD Comparator

3.3.4 Checker

[3.4 Count Control Unit](#three_four)

[3.5 Output Display Unit](#three_five)

1. [Used Elements](#B4)

[4.1 JK Flip Flop with async RESET and PRESET](#B4_1)

[4.2 D Flip Flop](#B4_2)

[4.3 MUX 8](#B4_3)

[4.4 MUX 2](#B4_4)

[4.5 Full BCD Adder 1 bit](#B4_5)

[4.6 Full BCD Adder 16 bit](#B4_6)

[4.7 Comparator 1 bit](#B4_7)

[4.8 Comparator 16 bit](#B4_8)

1. Faced Problems while designing & their solutions

[1- Design Goals](#Table)

We have been asked to design a stopwatch and count down in a single circuit.

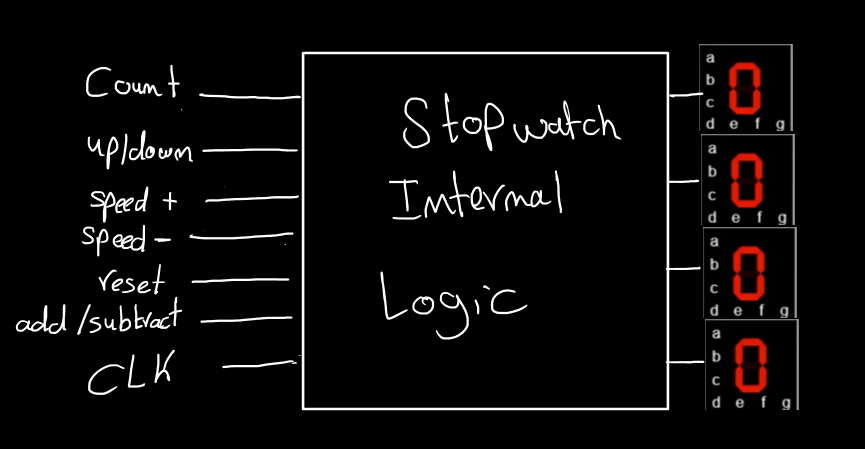
The features are:

* Count up to a specific second then stop
* Count down to a specific second then stop
* You can stop counting whenever you want
* While counting up you can add 2 minutes
* While counting down you can subtract 2 minutes
* Timing rate can be 0.5x or 2x
* The output should be visible on a seven segment display

Based on these features that we want to do, we thought that for this project we need some blocks to do these features:

* Block for counting
* Block for specifying counting mode up/down
* Block for addition and subtraction
* Block for changing the speed of Counting
* Block for stopping the counting
* Block for displaying the output

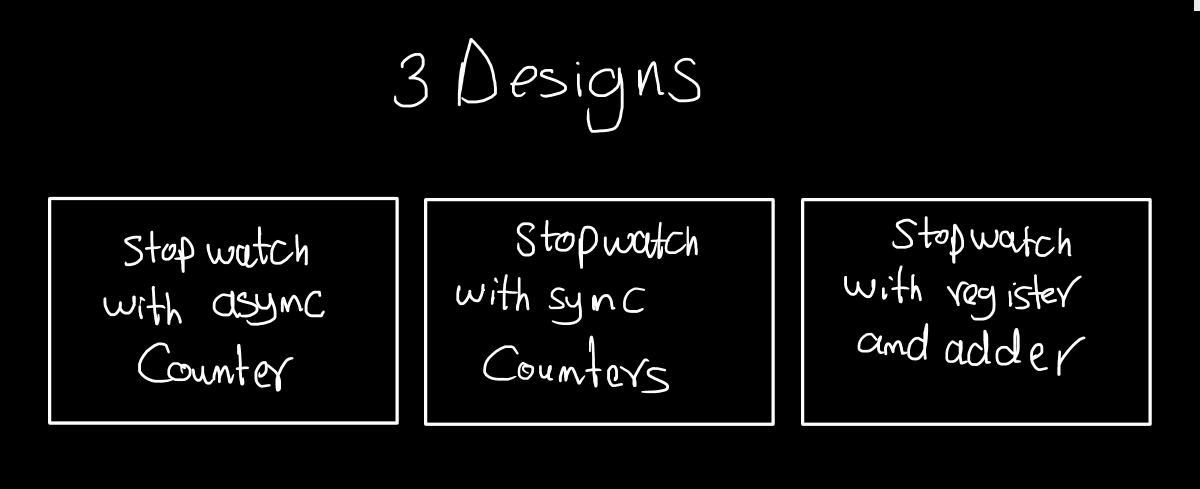
So as a start we want to make this:



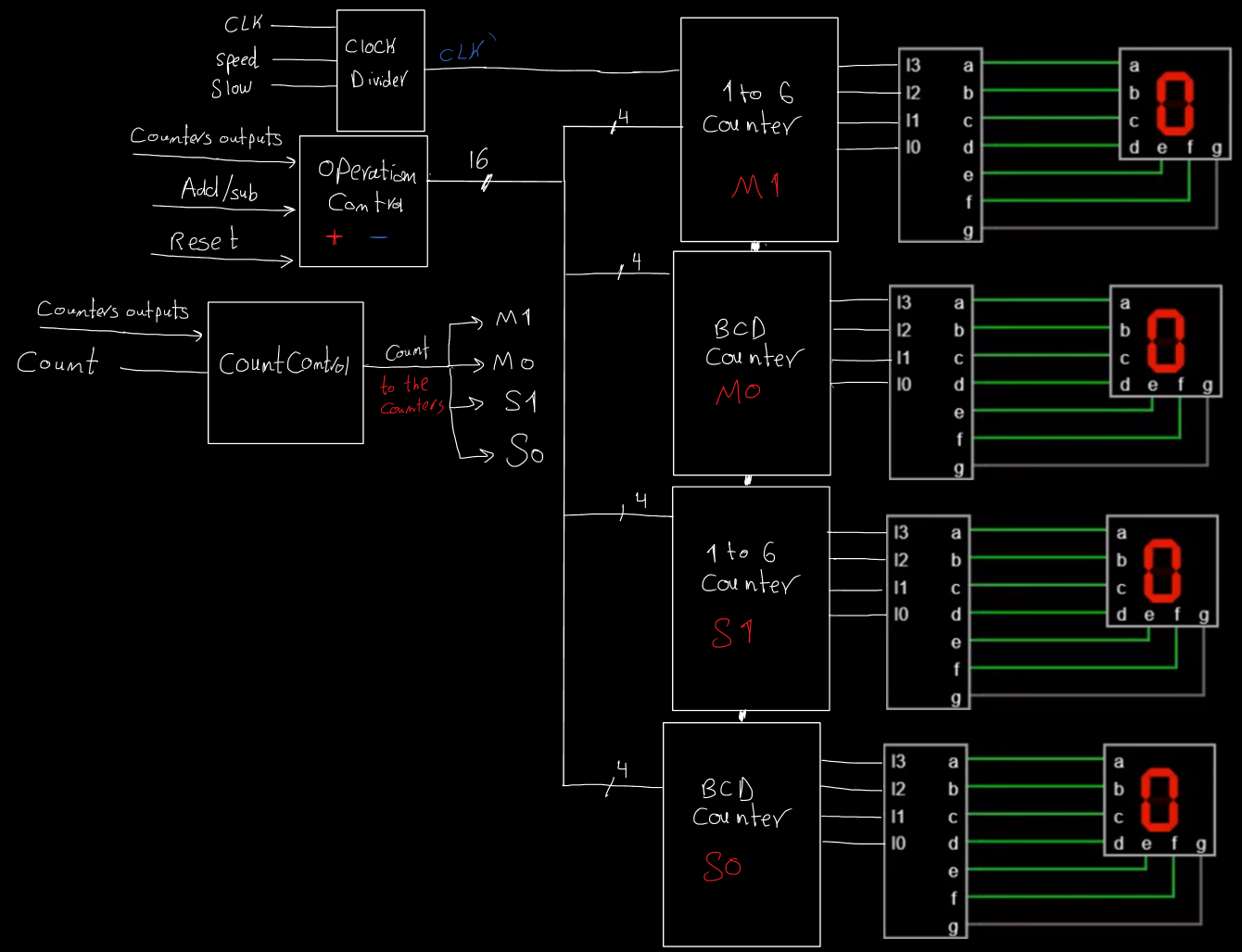
In the next section we gathered the different methods thought of to implement such a thing

[2- Brainstorming &](#Table)

[chosen Approaches](#Table)



[2.1 Stopwatch with synchronous Counter](#Table)

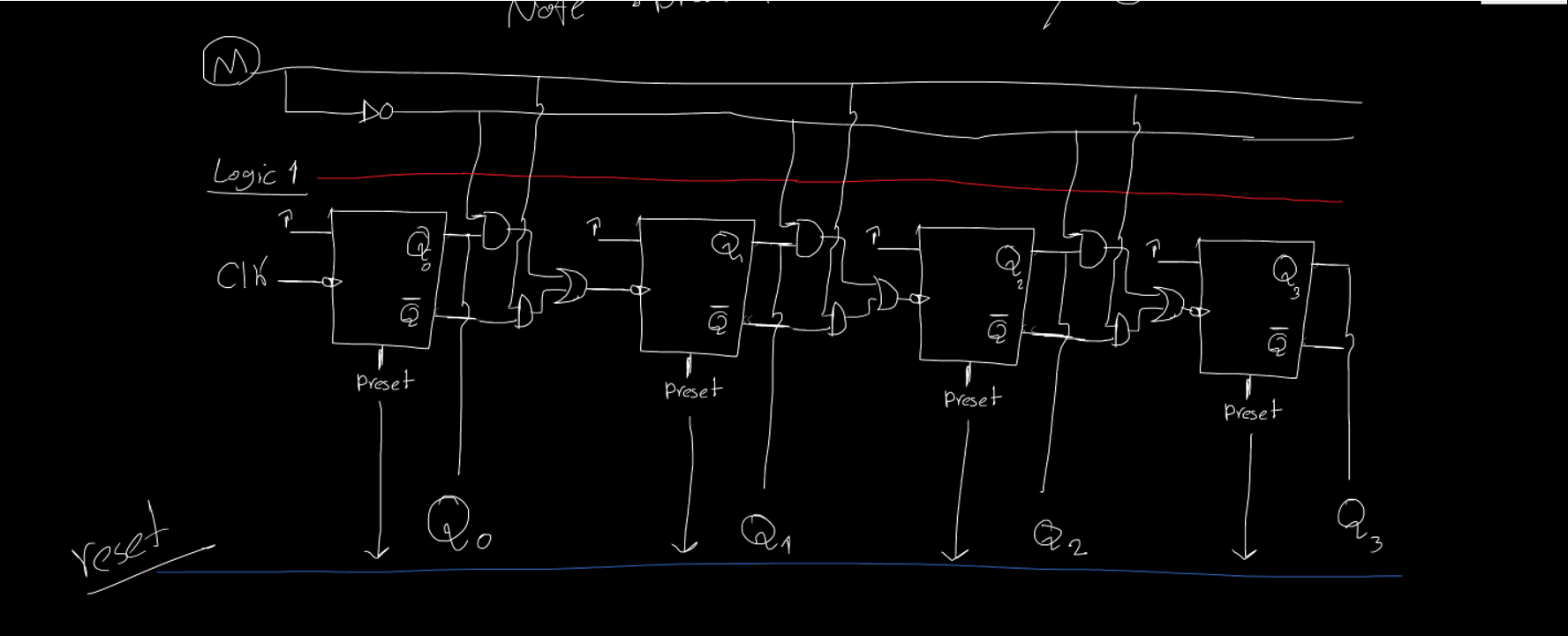


You can see that:

* We used both BCD synchronous counters and 0to5 synchronous counters
* Also the output of the counters is a feedback to the operation control and count control blocks.
* Count control is the block that stops the counter on max or min or when user asks to stop.
* Operation Control is used to add or subtract 2 minutes but if the addition or subtraction can’t exceed the maximum or minimum values.

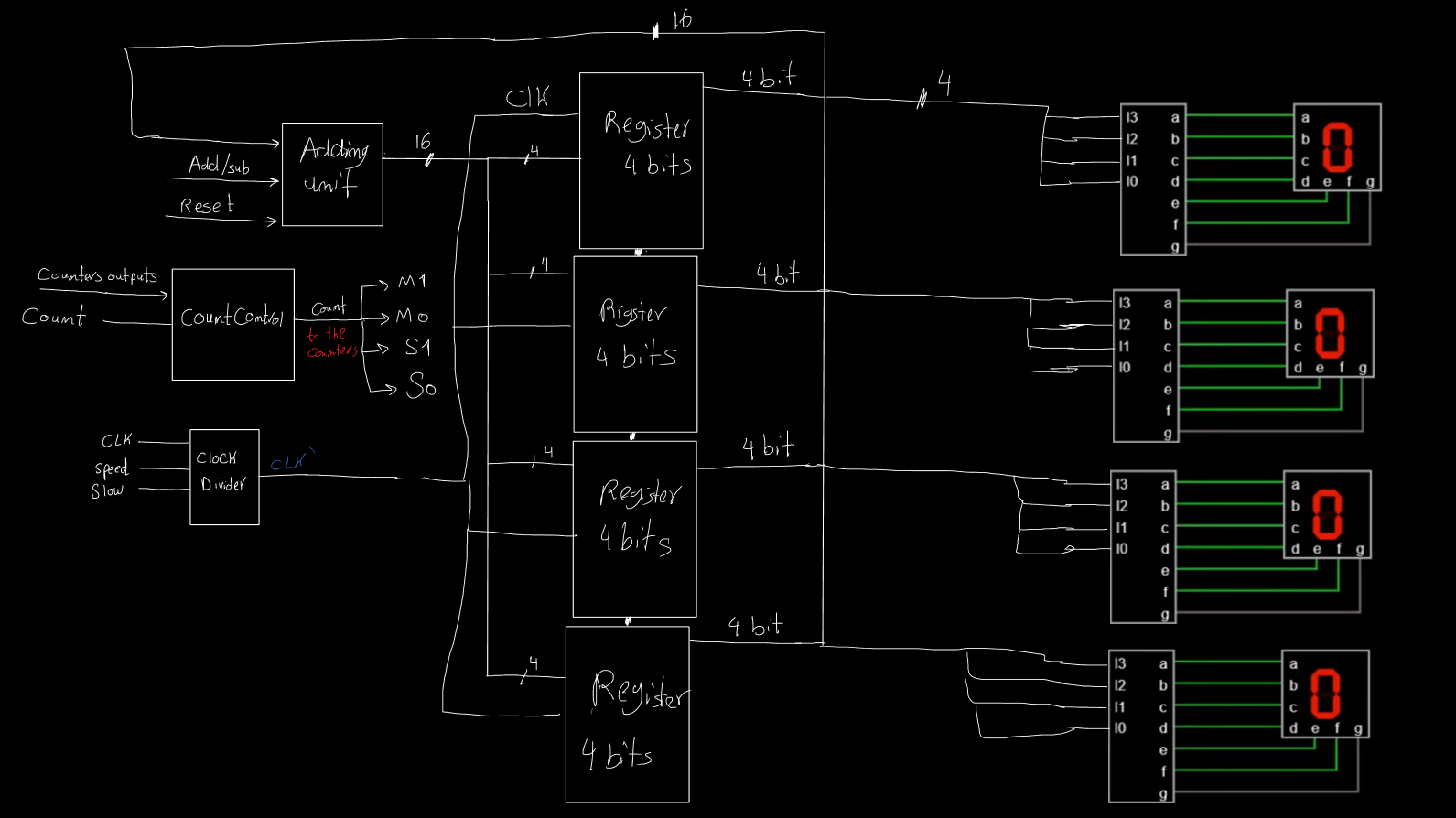
[2.2 Stopwatch with asynchronous Counter](#Table)

This design is the same as the previous in terms of blocks in the design but the counter we thought of was asynchronous.

We tried to draw it and we have done its simulation on Flastad website

Then all counters will be connected to each other asynchronously.

[2.3 Stopwatch with Registers and Adders](#Table)



In this design we thought of storing the value in a register and then display it and also take the output as a feedback to an adding unit that has some adders and subtractors to count up or down based on the inputs used.

* The clock divider will do the same job as the registers require clock
* The count control here will be little bit tricky as I have to stop the clock to stop the register from loading

Now the next sub section will discuss the design chosen and reasons of that choice.

[2.4 Rating Design Approaches](#Table)

let’s first see what are the disadvantages of each design

asynchronous counter design:

The main problem is that the asynchronous counter can have unexpected behavior and also it will be complex to control each value because there will be a different logic for each bit.

Register and adder design:

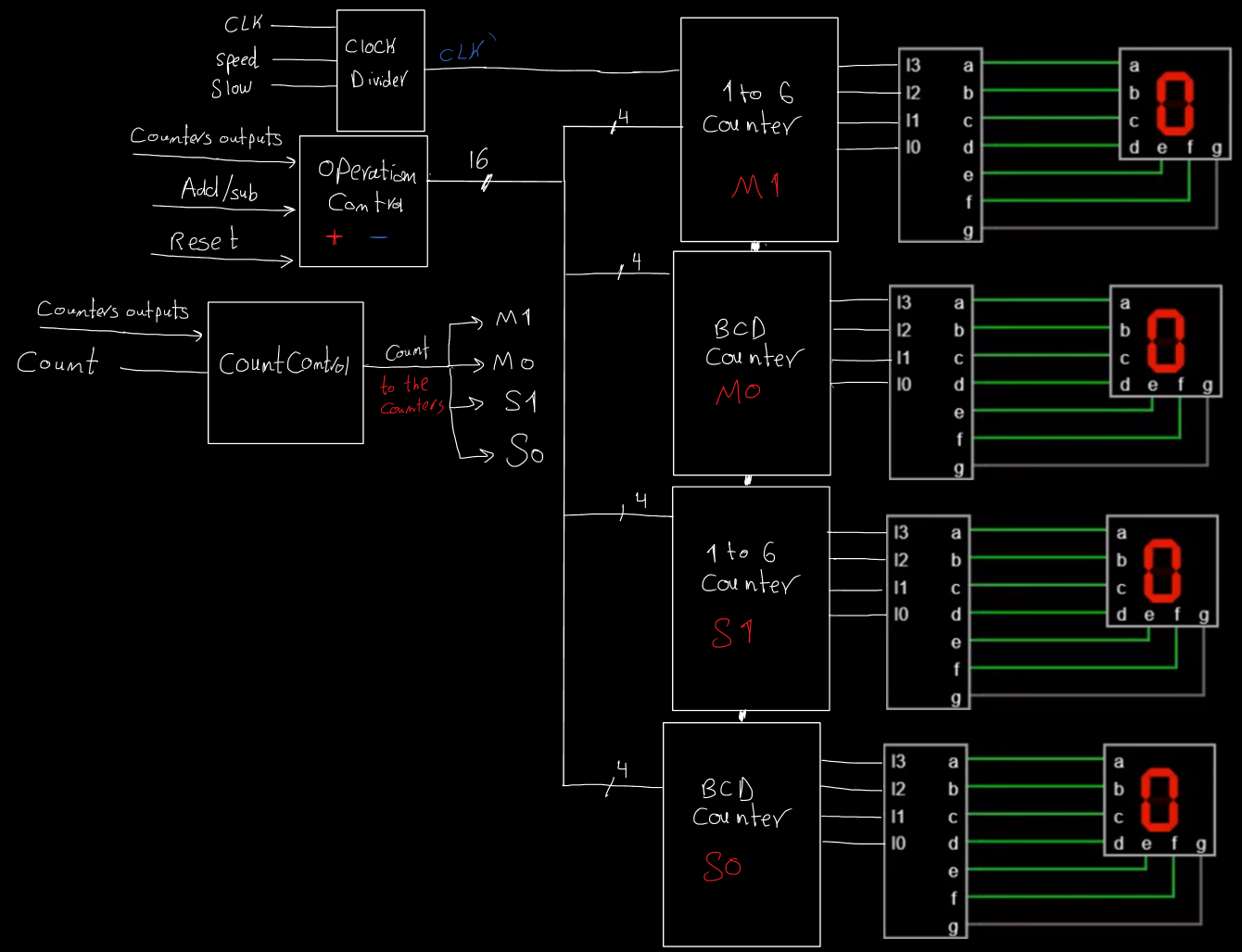
The main problem of this design is the adding unit it will be complicated and why making it add 1 every clock cycle when I have a counter that does the same thing by simple gates between the JK flip flops

Also if we used this logic there will be higher delay in adding ones to the register because the signal goes through more components

asynchronous counter design:

we can conclude that using this method is the best in our opinion because the logic of each bit will be the same unlike the asynchronous counter. Also the asynchronous counter will act as a register to load the new values that the user enters through adding two minutes.

[3- Chosen Design](#Table)

[Logic & Implementation](#Table)

as previously seen in the block design of the synchronous counter we need to make the logic and code of the following components:

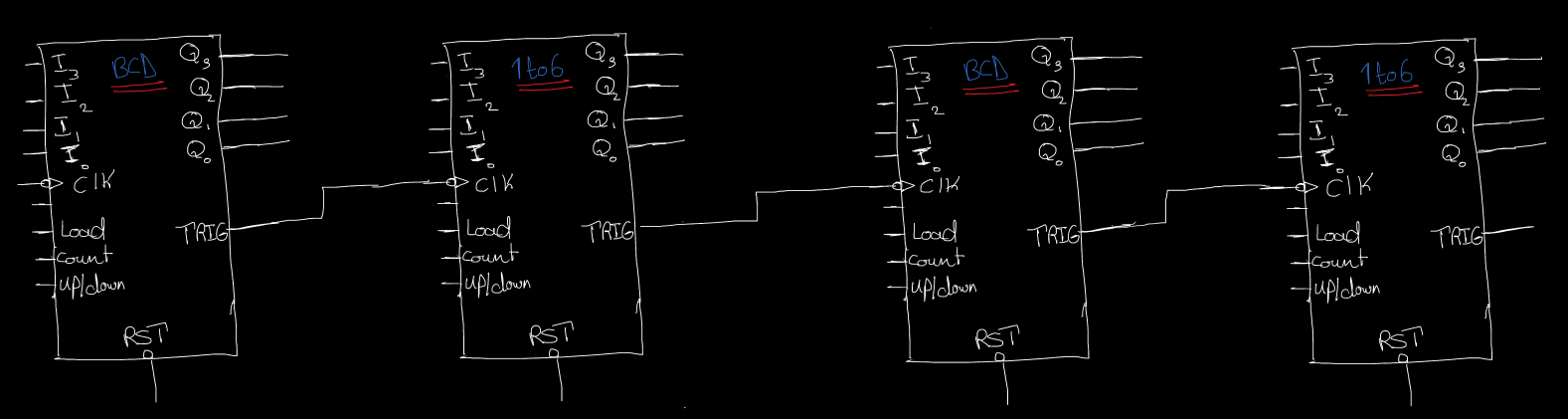
1. Timer (4 counters)
   1. BCD Up & Down Counter
   2. 0To6 Counter
2. Clock Divider
3. Operation Control Unit
   1. BCD adder
   2. BCD subtractor
   3. Comparator
   4. Checker
4. Count Control Unit
5. Output Display Unit

[3.1 Counters](#Table)

In this Logic we need 4 counters (two BCD for seconds and minutes first digit) (two 0 to 5 counters for second digit) As the maximum in natural stopwatches is 59:59

We are aware that the maximum count for this project is different but this will be implemented in the Count Control Unit.

As Seen the first counter has the clock given and when it reaches its maximum value it motivates the next counter to count.



We need in each counter the following inputs:

- I3, I2, I1, I0 is the inputs through which we can change the value when the user adds or subtracts.

- CLK for the general clock of the flip flops inside the counter

- Load input to stop counting and displays the inputs I3, I2, I1, I0

- Count input to start and stop counting

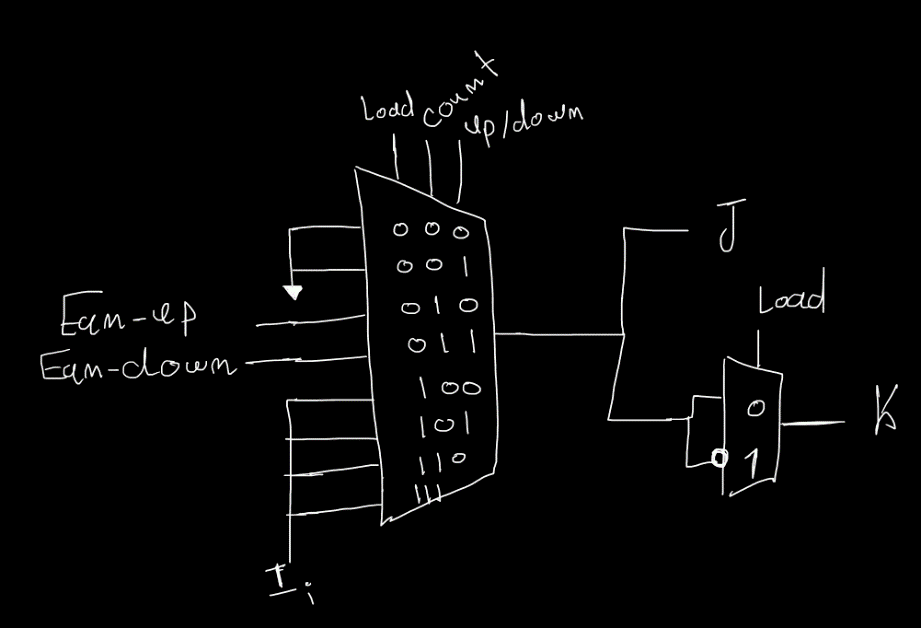
- Up/Down input to specify the mode of counting up or down

In the next section and the following one we will discuss how we build BCD counter with aforementioned inputs, features.

[3.1.1 BCD Up & Down Counter](#Table)

Here is the function table for the counter

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Load | Count | Up/Down | Function | J | K |
| 1 | x | x | Load Inputs | Input | Input’ |
| 0 | 0 | X | Pause | 0 | 0 |
| 0 | 1 | s1 | Count Up | Eqn Up | Eqn Up |
| 0 | 1 | 0 | Count Down | Eqn Down | Eqn Down |

To implement this function table, we will put MUX8 between every JK flip fop to determine the inputs.

As you can see we also used mux 2

And an inverter to make the k input is the I’ while loading only.

The next step is that we want to make the counter reset its value when it reaches to 1001 = 9 when counting up.

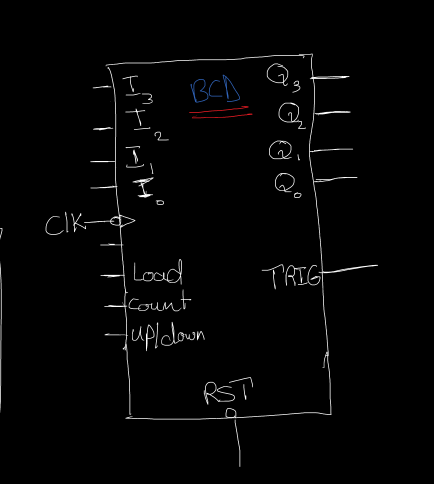
So we can take and gate from [Pending Mohammed Sayed]

We also need to make the counter get back to 9 when counting down and reaching zero because if we didn’t make logic for it, the counting will continue to 1111.

So we can take and gate from [pending Mohammed Sayed]

Finally, we need a trigger to activate the next counter and make it count

We will make this trigger go to the clock of the following counter and this trigger is or gate from the [pending Mohammed Sayed]

BCD up and down Counter Schematic

Components:

. JK Flip Flop with asynchronous Reset and Preset

. MUX 8

. MUX 2

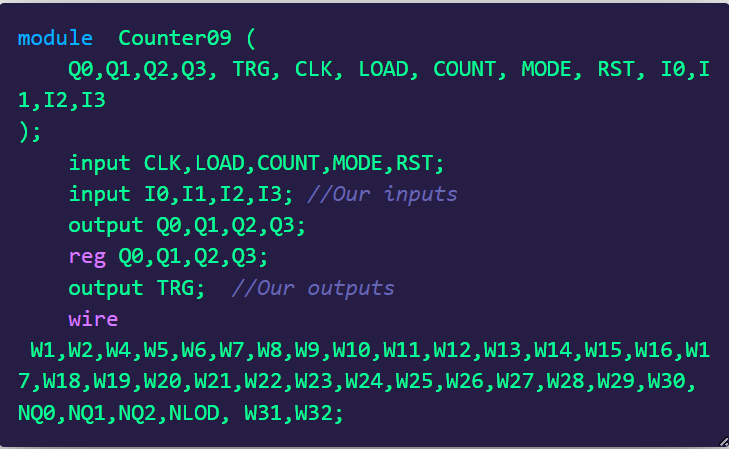
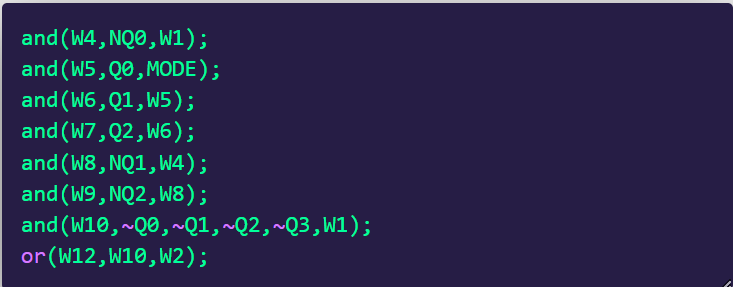
. AND Gate

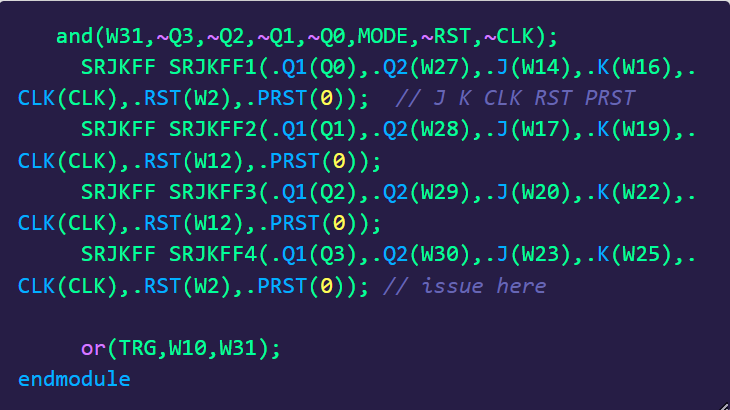
. OR Gate

. Not Gate

Simulation pending

Code Implementation



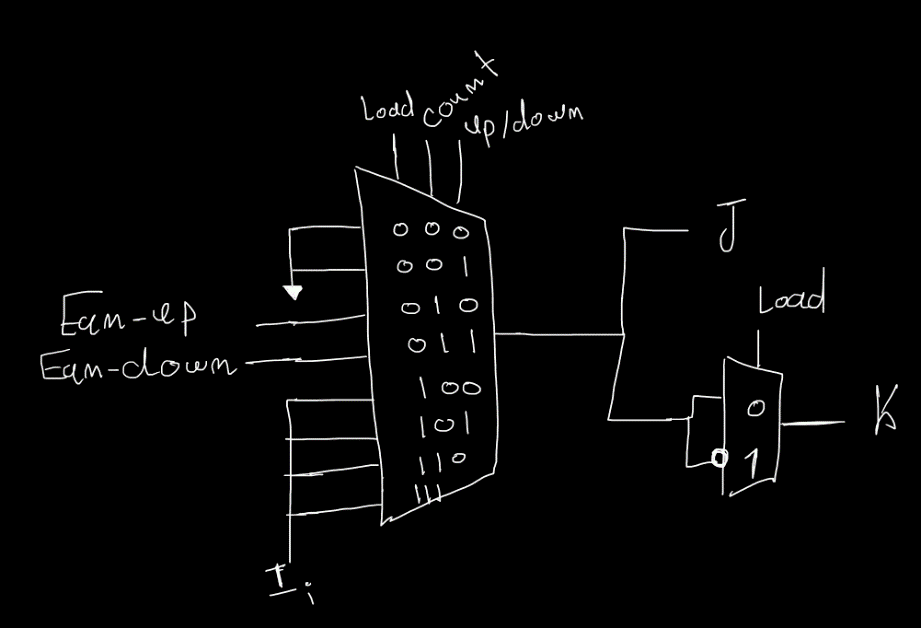
 Simulation pending

[3.1.2 0 to 5 Up & Down Counter](#Table)

This component is similar to the BCD up and down counter but the reset logic and the trigger logic will be different because you want to reset at 5 which is 1010 in counting up mode and also return to the five again when zero is reached.

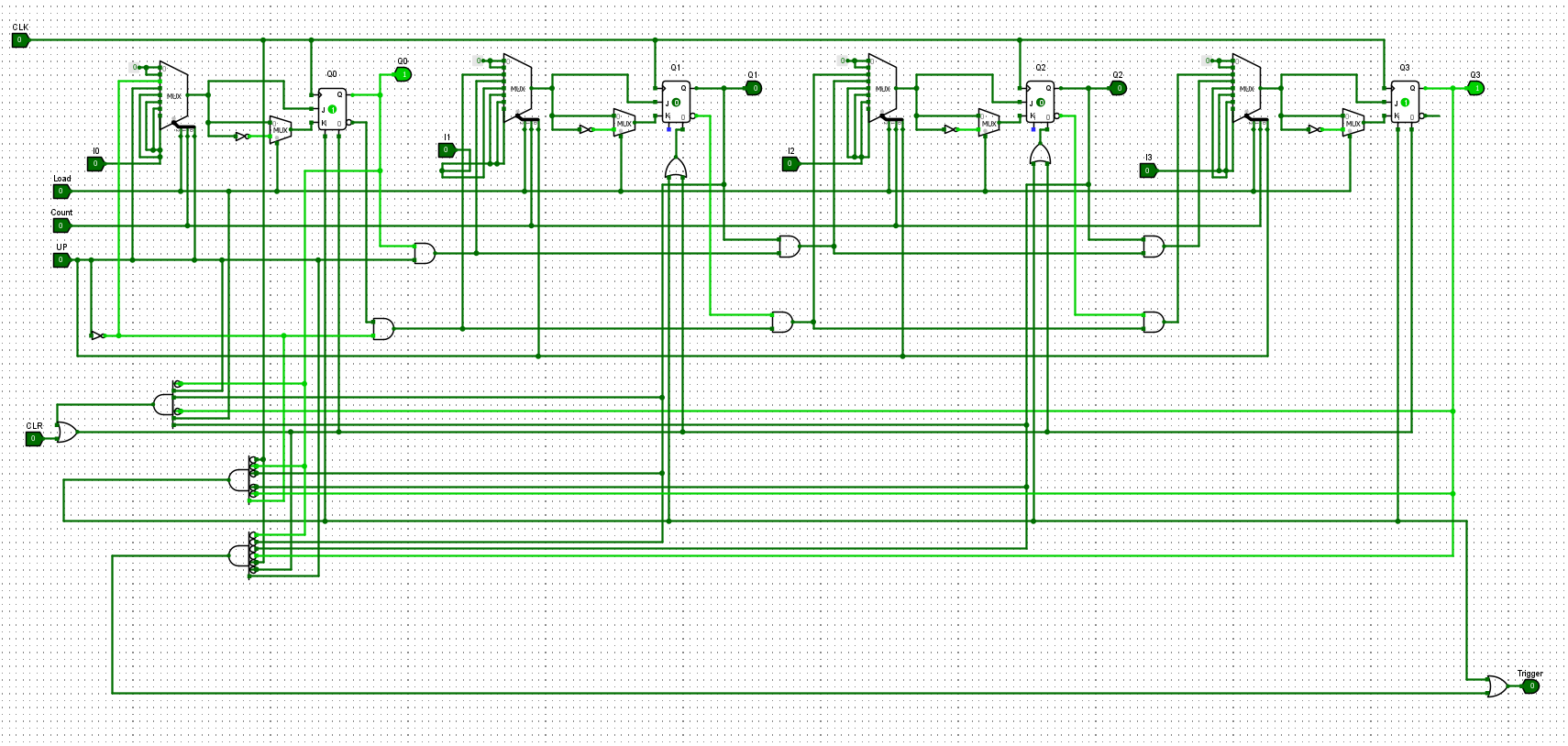
Function table is the same as the BCD up and down counter

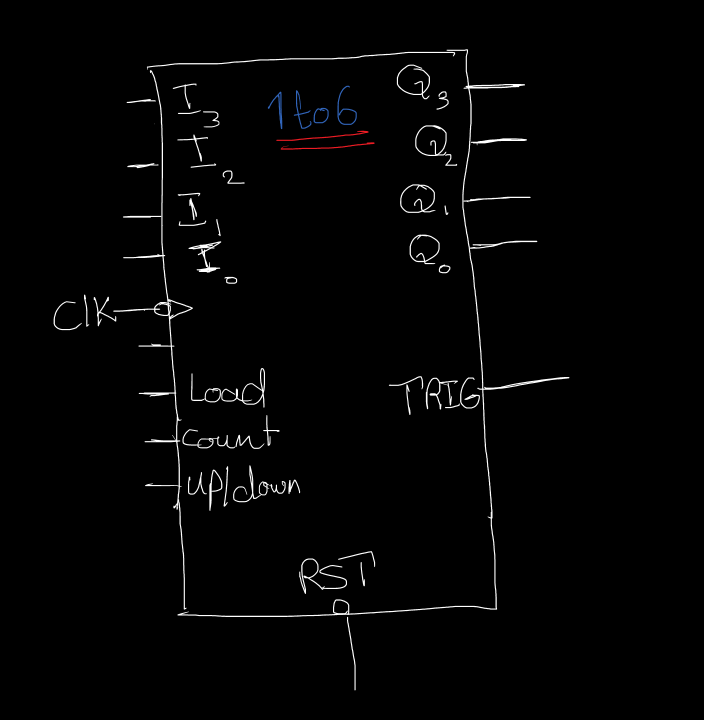
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Load | Count | Up/Down | Function | J | K |
| 1 | x | x | Load Inputs | Input | Input’ |
| 0 | 0 | X | Pause | 0 | 0 |
| 0 | 1 | 1 | Count Up | Eqn Up | Eqn Up |
| 0 | 1 | 0 | Count Down | Eqn Down | Eqn Down |

The same combination of MUX 8 and MUX2 is used to determine the input to the JK flip flops.

For resetting and the Trigger Logic:

Pending Mohammed Sayed

0 to 5 up and down Counter Schematic

Components:

. JK Flip Flop with asynchronous Reset and Preset

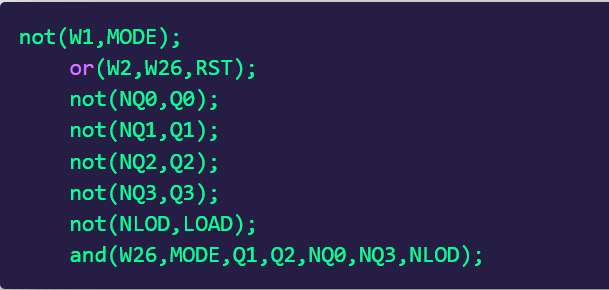
. MUX 8

. MUX 2

. AND Gate

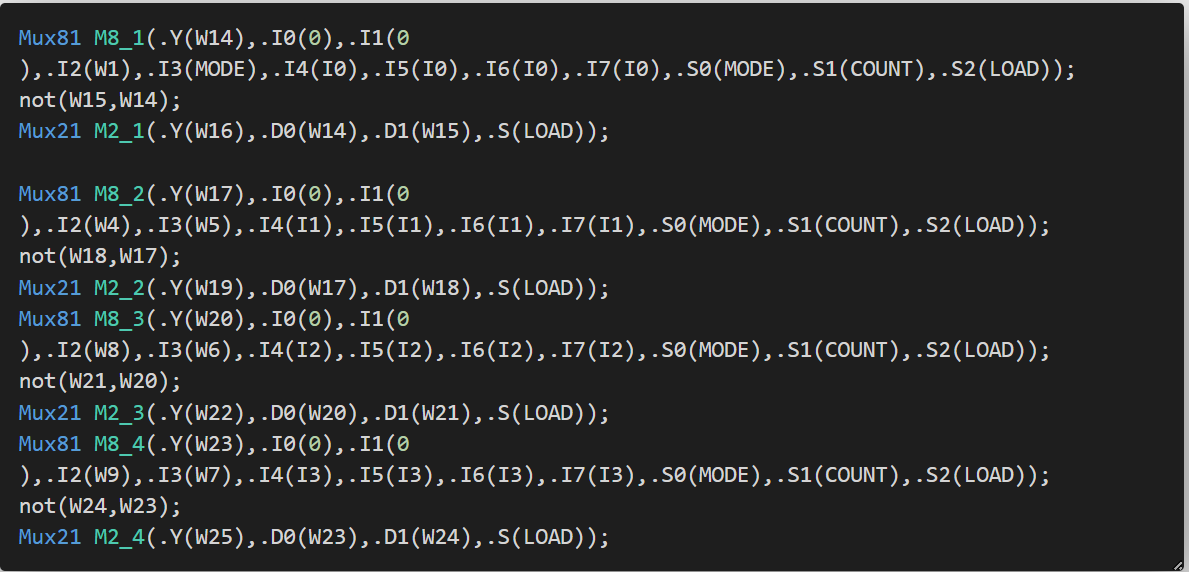
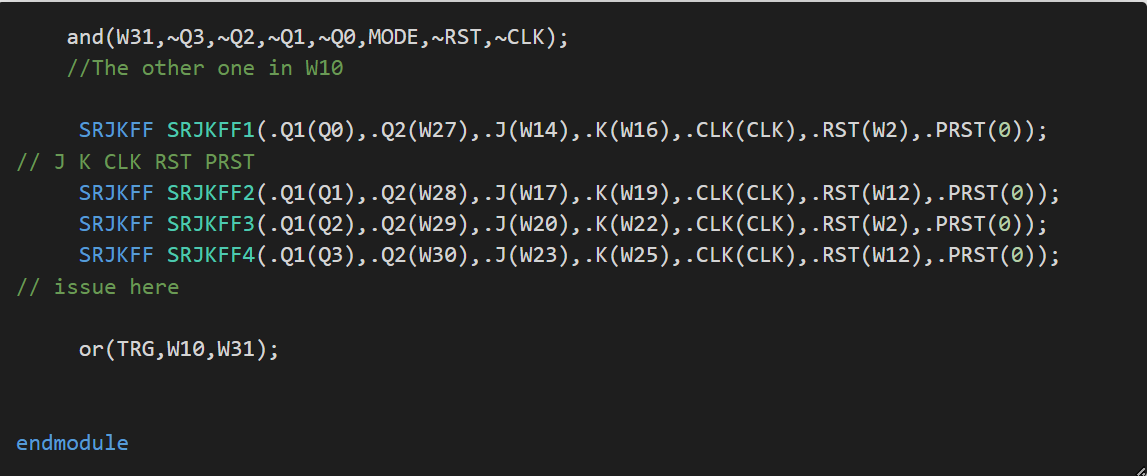
. OR Gate

. Not Gate

Code Implementation



Pending Simulation



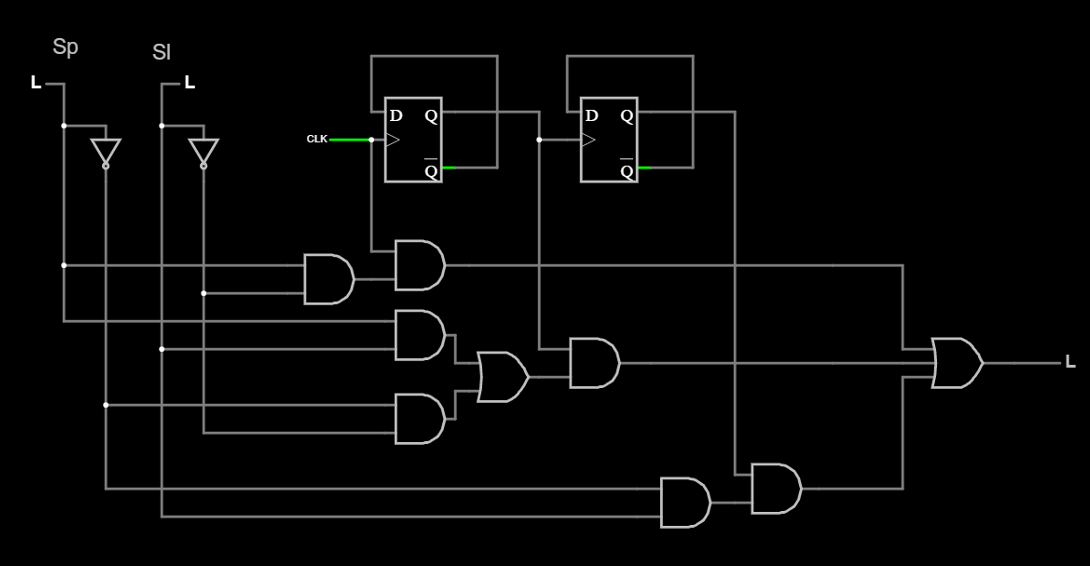
[3.2 Clock Divider](#Table)

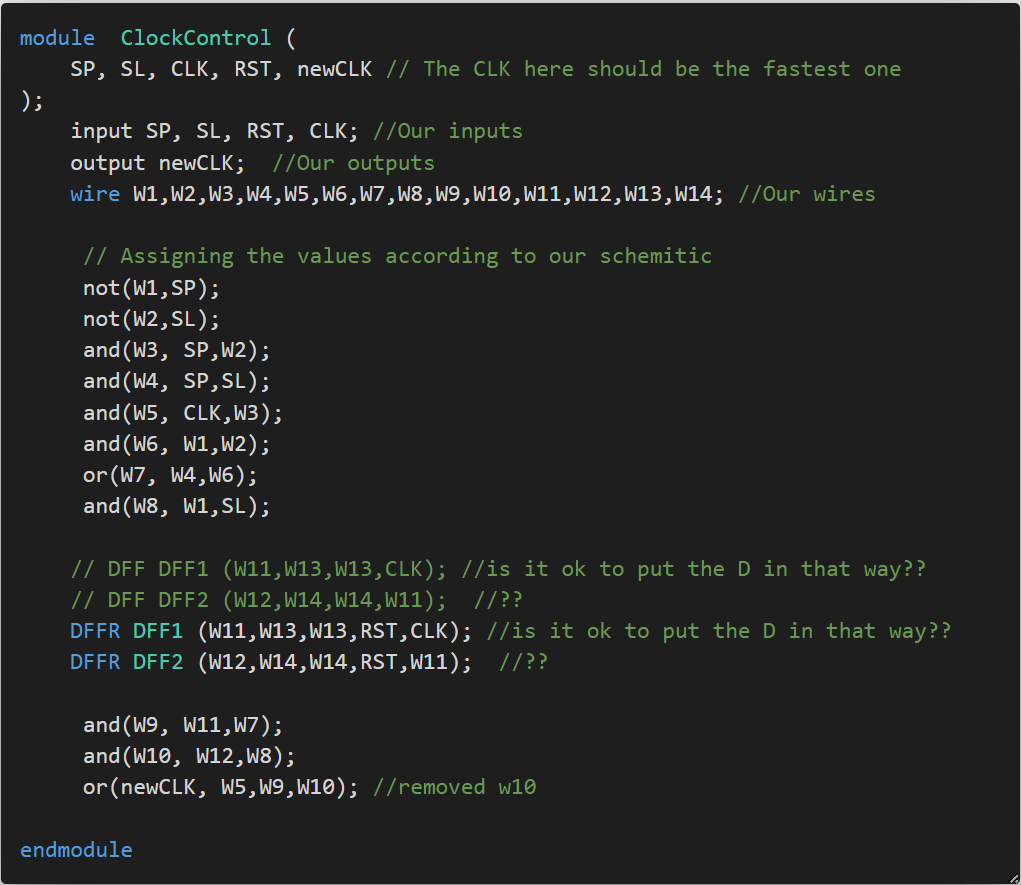
For speeding up and down we wanted to change the clock that enters the counters so its frequency is divided by 2 or 4. The first clock is the 2x clock and when it is divided by 2 gives the normal speed and when it is divided by 4 it will give 0.5x speed.

Using D Flip Flops, we can implement this logic



|  |  |  |
| --- | --- | --- |
| Speed | Slow | CLK(new) |
| 0 | 0 | Q0 |
| 1 | 0 | CLK |
| 0 | 1 | Q1 |
| 1 | 1 | Q0 |





Simulation->>>>>> pending

[3.3 Operation Control Unit](#Table)

3.4 Count Control Unit

This Block is responsible for stopping at maximum value when counting up and stopping at minimum value when counting down.

The maximum Value for Timer is 49:30 which can be translated to:



Eqn Max Q15 Q14 Q13 Q12 Q11 Q10 Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0

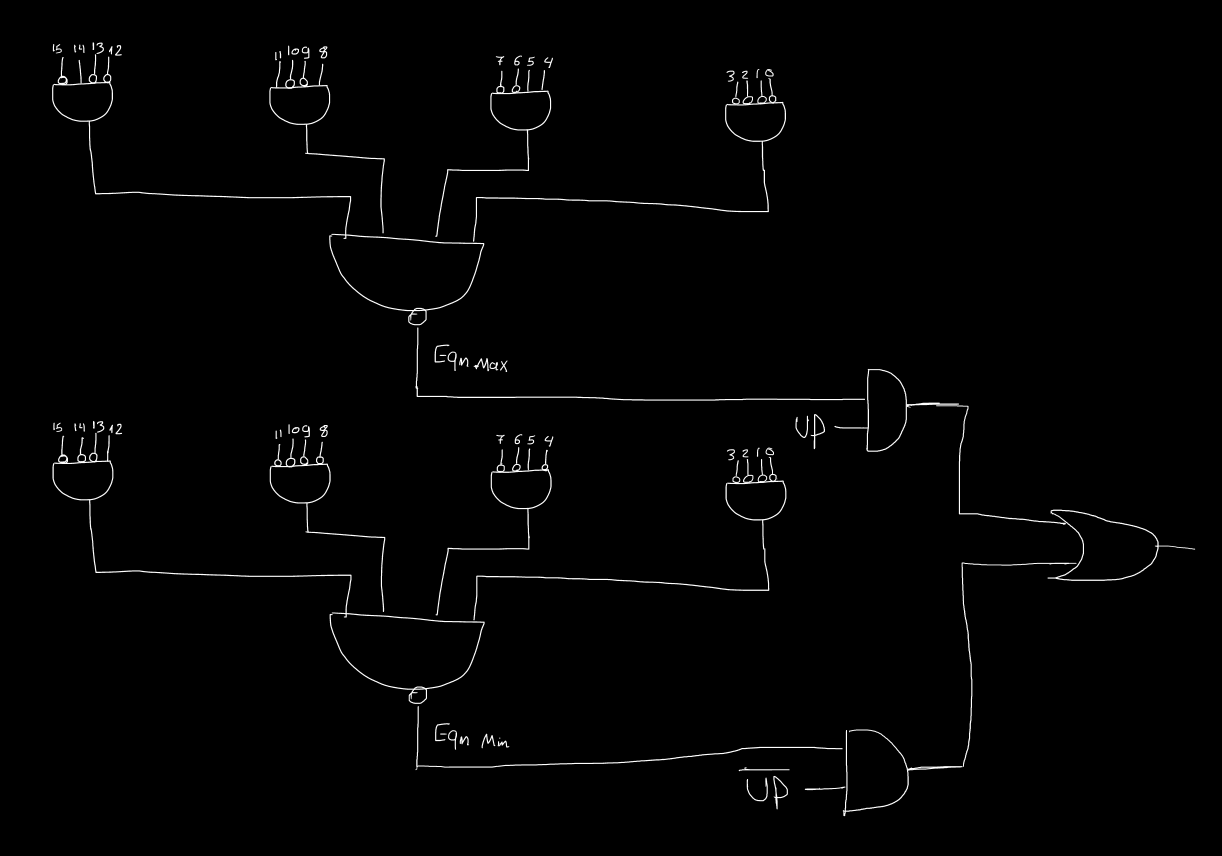
The minimum Value for Timer is 10:20 which can be translated to:



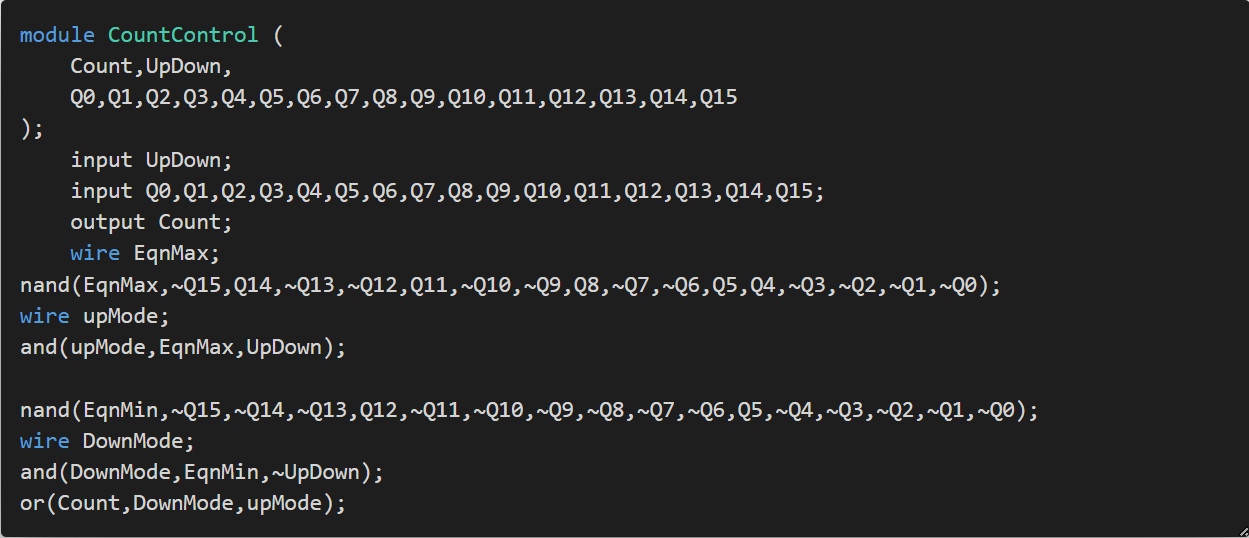
Eqn Min Q15 Q14 Q13 Q12 Q11 Q10 Q9 Q8 Q7 Q6 Q5 Q4 Q3 Q2 Q1 Q0



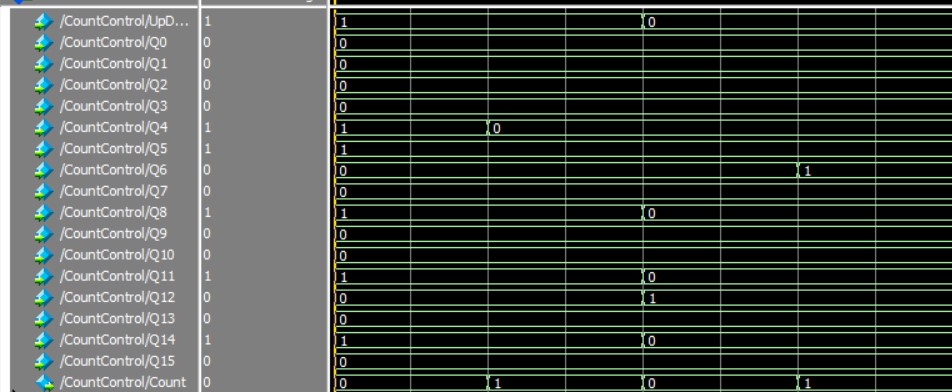
|  |  |  |
| --- | --- | --- |
| Up | Q15 to Q0 | Count |
| 1 | Eqn Max | 0 |
| 0 | Eqn | 0 |

So the equation will be: Count = Up (Eqn Max) + Up (Eqn Min)





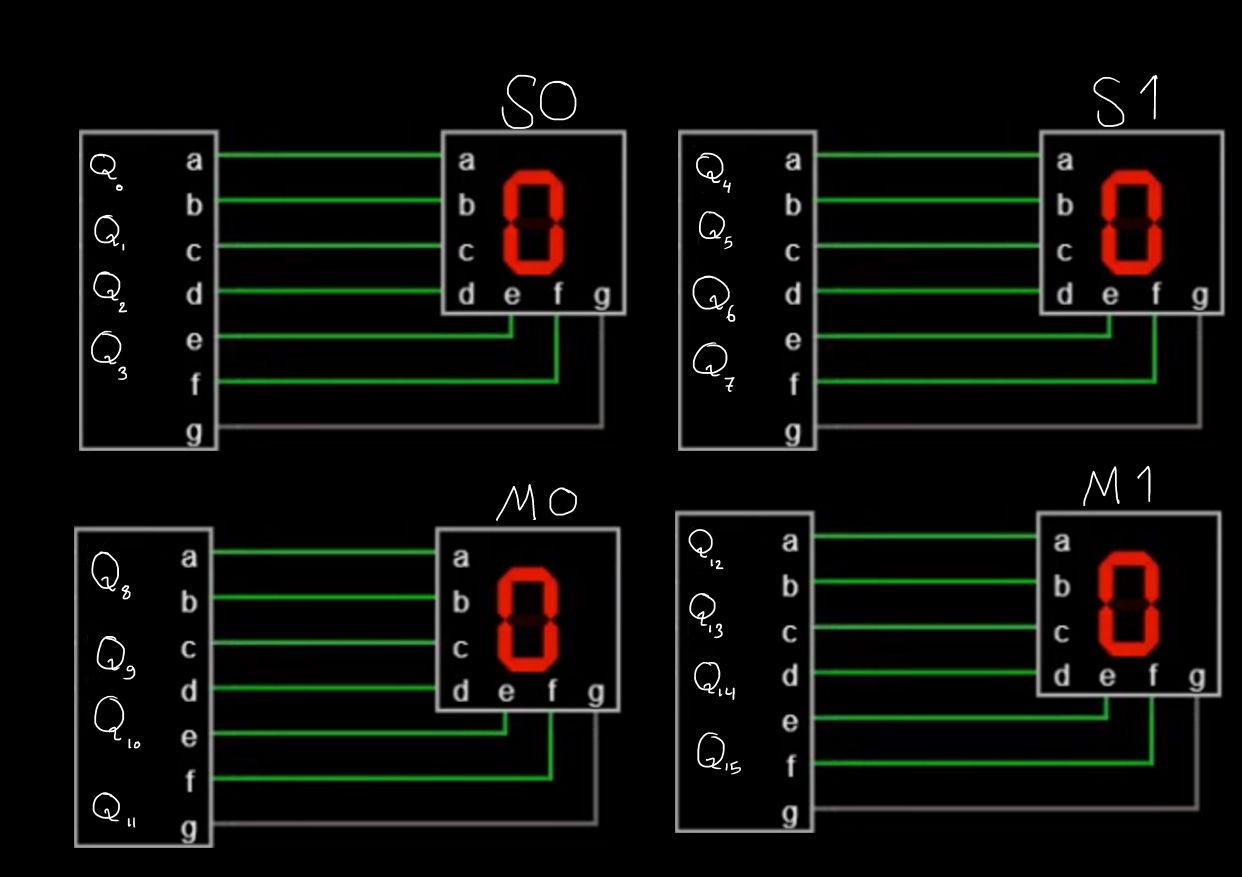
Simulation:



As You can see in the simulation count equals 0 only if the user reached maximum or reached minimum while counting up or down respectively.

3.5 Output Display Unit

In this unit we just need to take the Q15:Q0 resulting from the counter and enter each four bits in a 7 segment decoder and connect the output to 4 seven segments for M1, M0, S1, S0.



As you can see the 7 Segment Decoder is connected with the 4bit output from each counter of the Timer

Schematic of 7 Segment Decoder

[4. Used Elements](#Table)

In this Chapter we will introduce the sub components that we used and include the truth table, Schematic and the Structural code in system Verilog

These are The components that we used:

4.1 JK Flip Flop with async RESET and PRESET

4.2 D Flip Flop

4.3 MUX 8

4.4 MUX 2

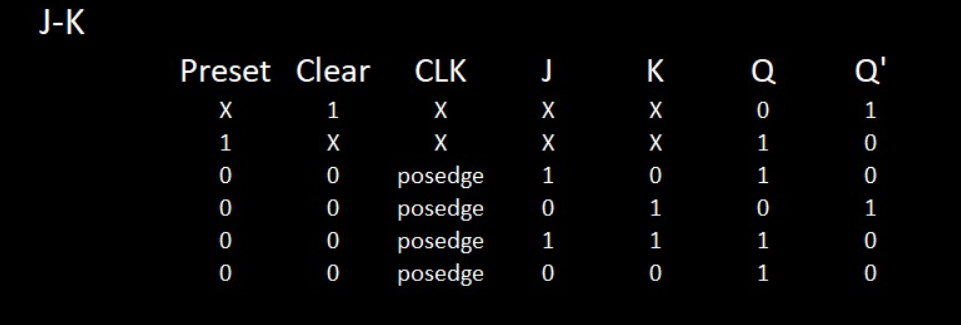
4.5 Full BCD Adder 1 bit

4.6 Full BCD Adder 16 bit

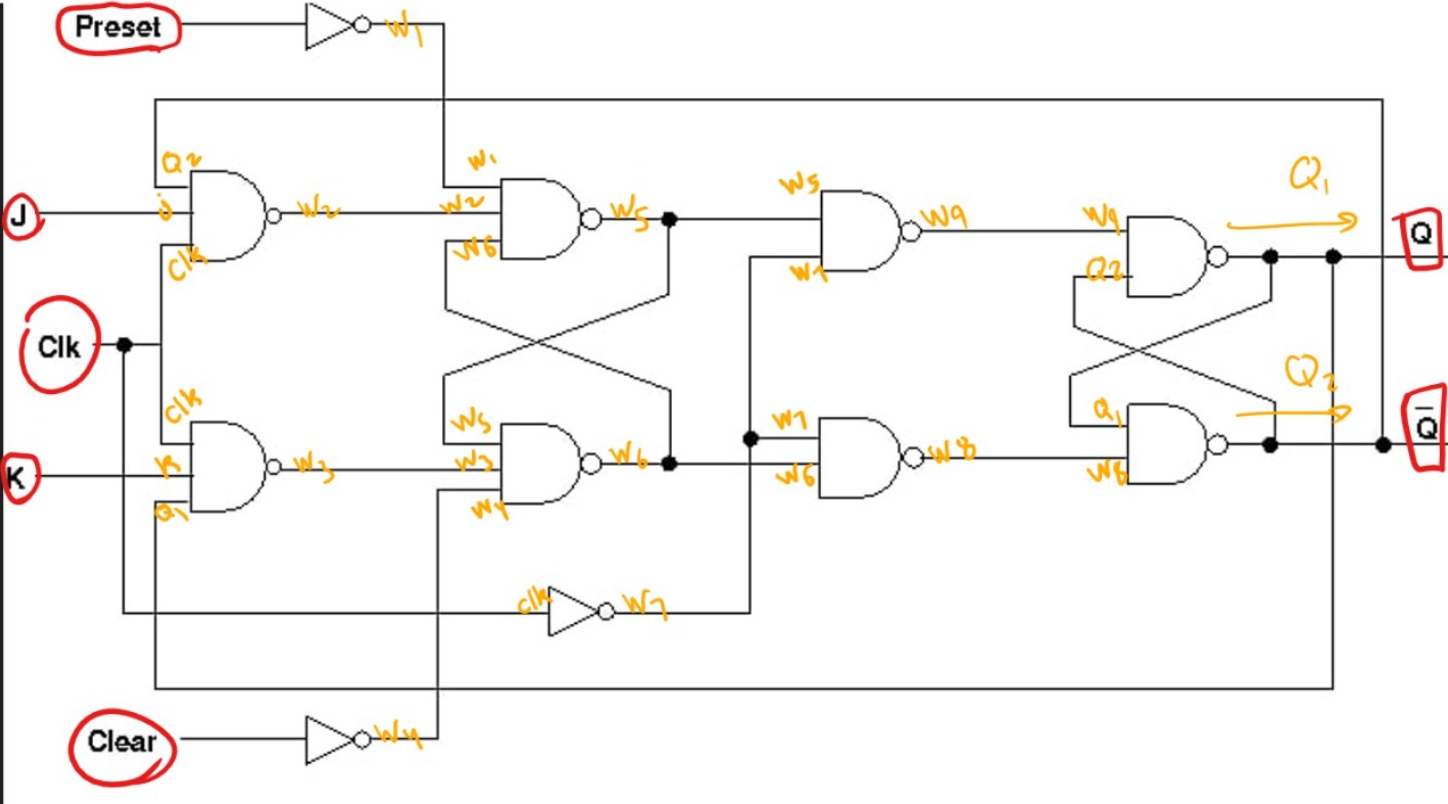
4.7 Comparator 1 bit

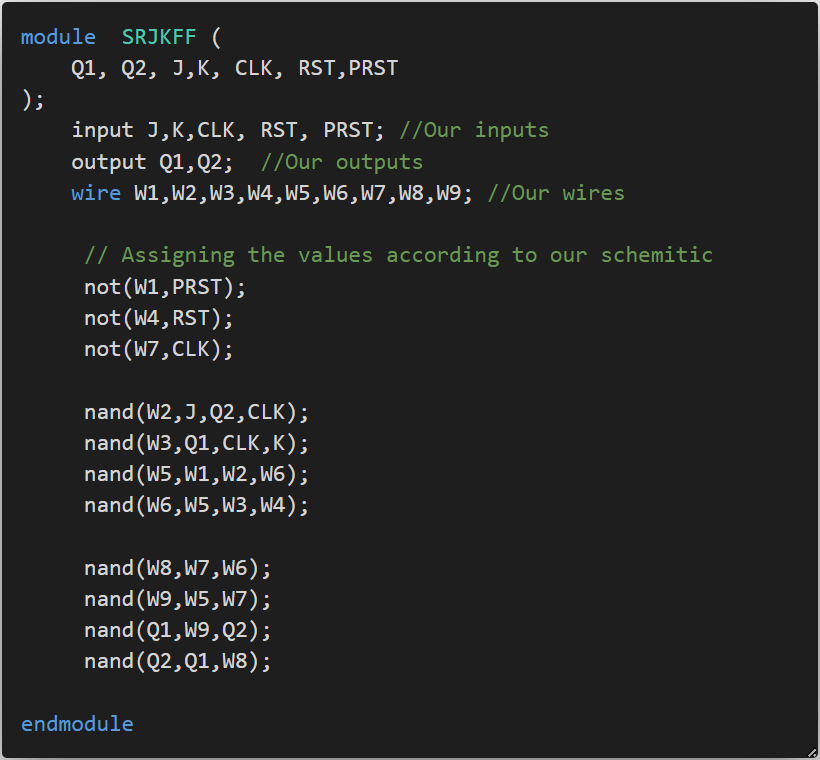
4.8 Comparator 16 bit

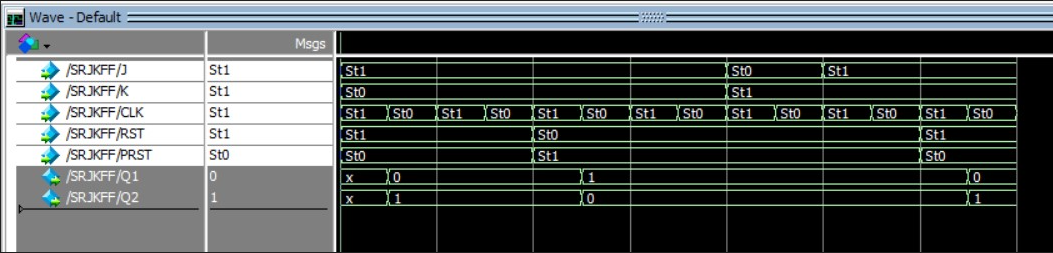
[4.1 JK Flip Flop with asynchronous Reset and Preset](#Table)



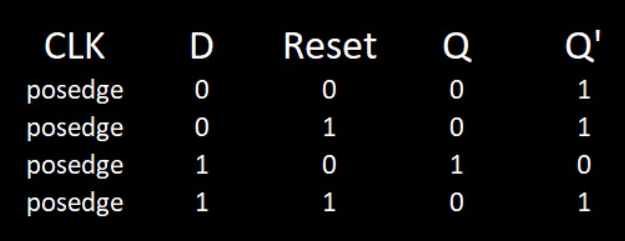
Schematic (Note the wires written are the wires used in coding)

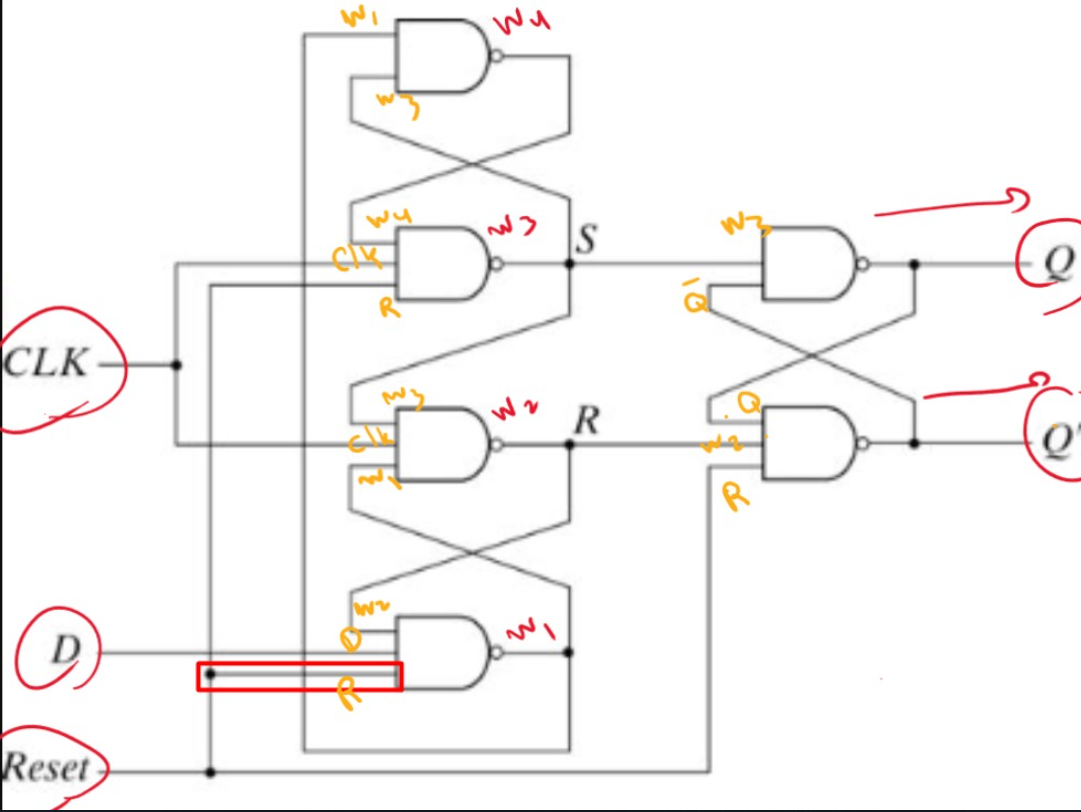


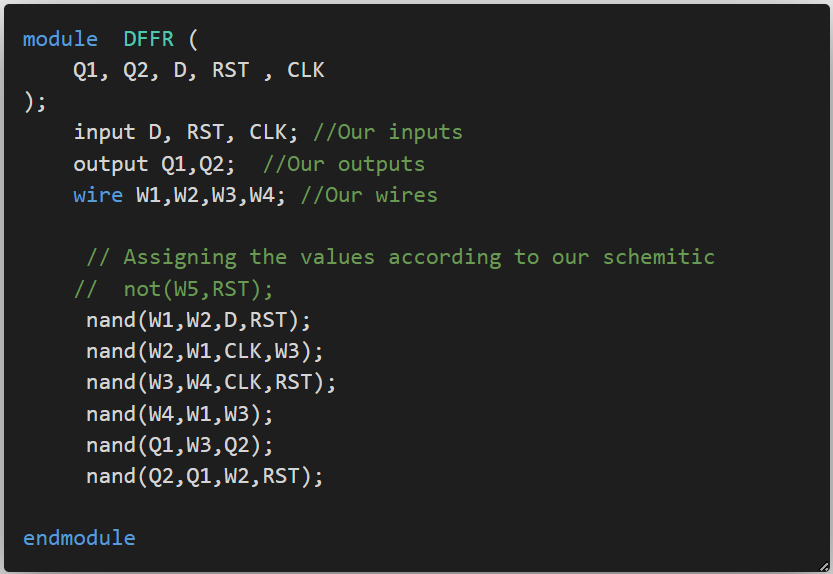
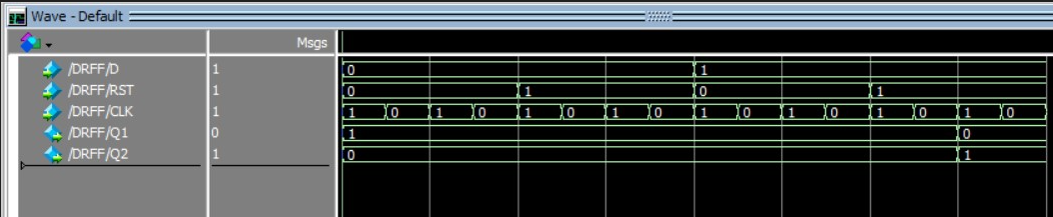


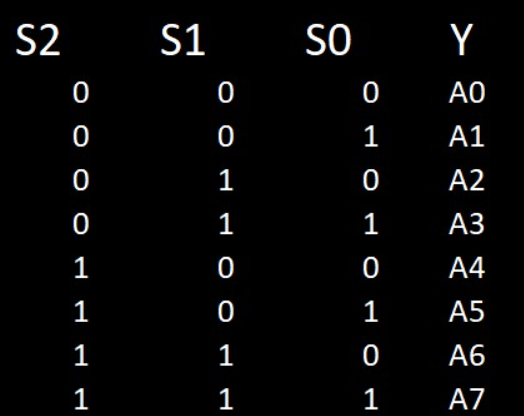


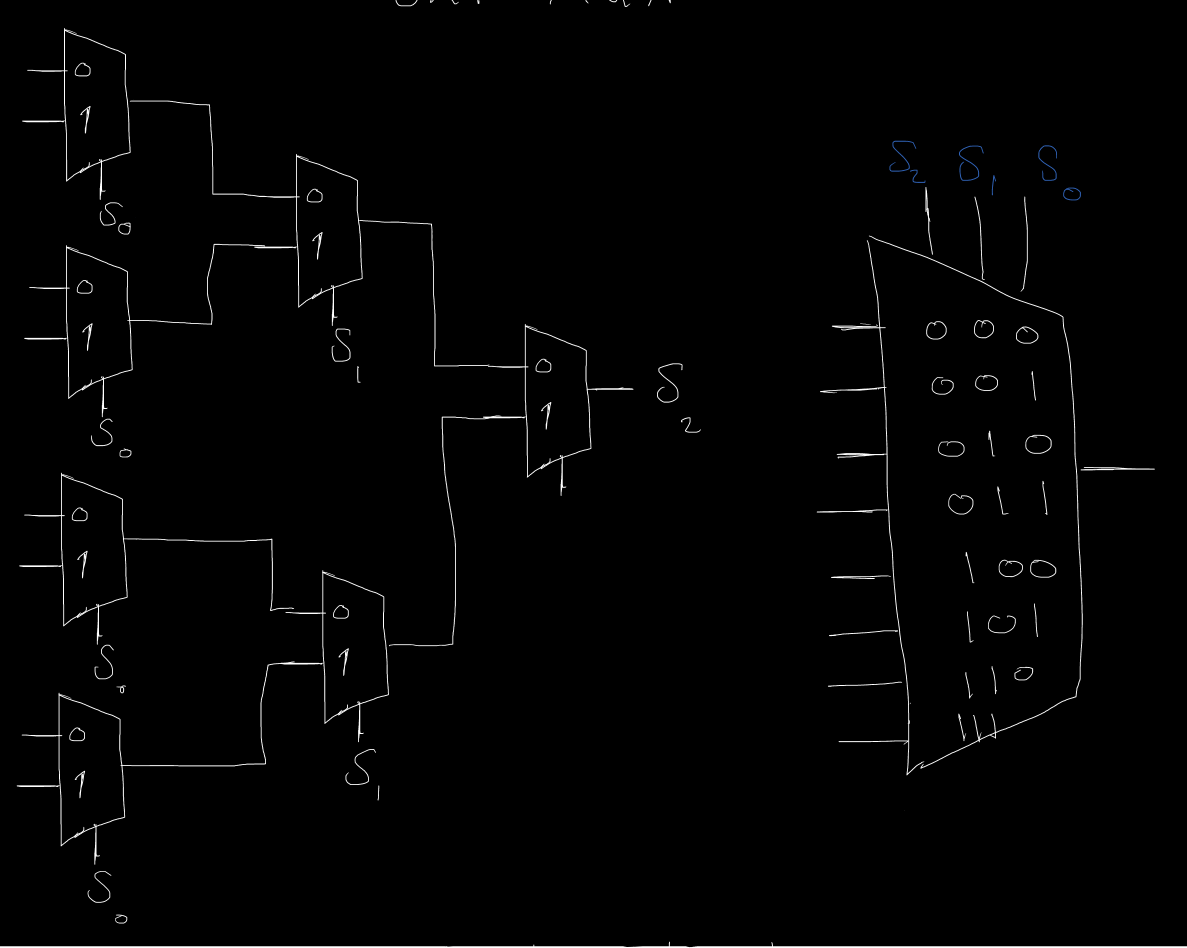
[4.2 D Flip Flop](#Table)

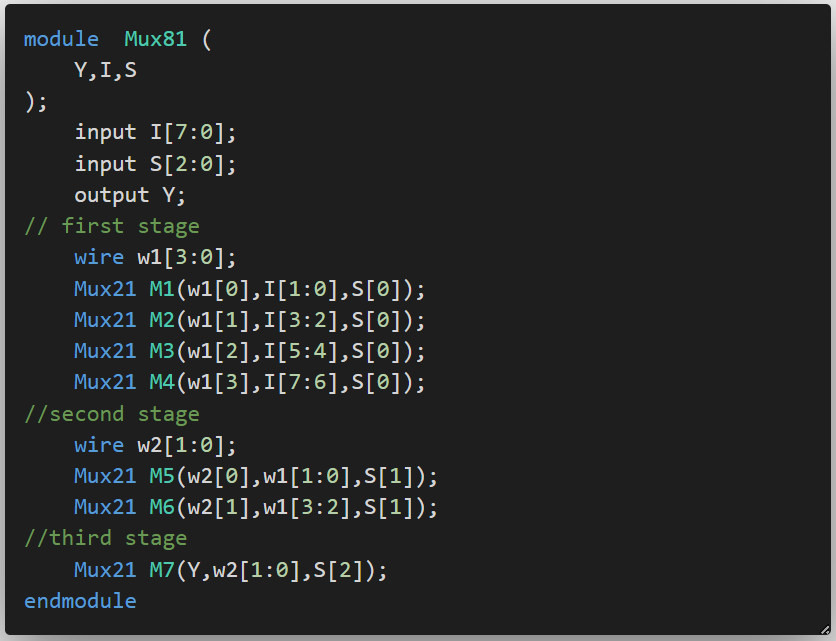


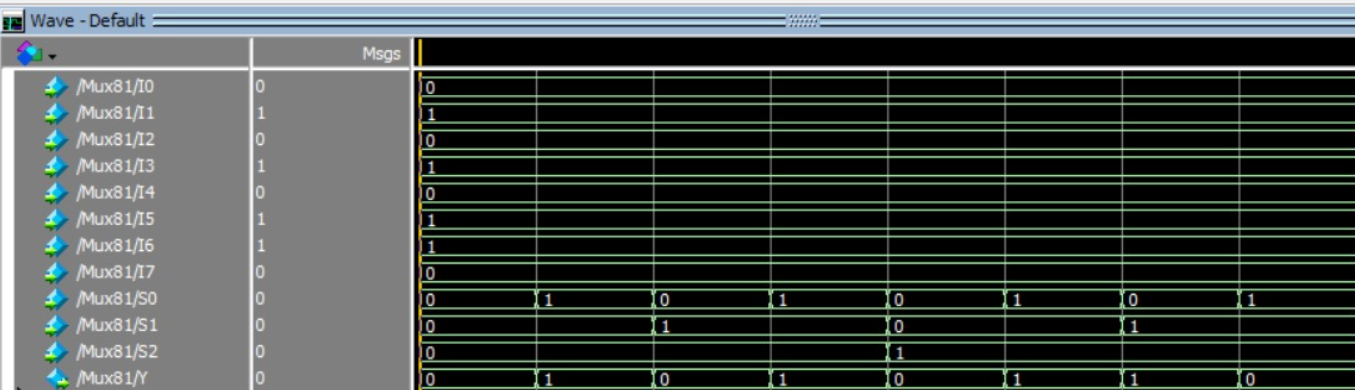


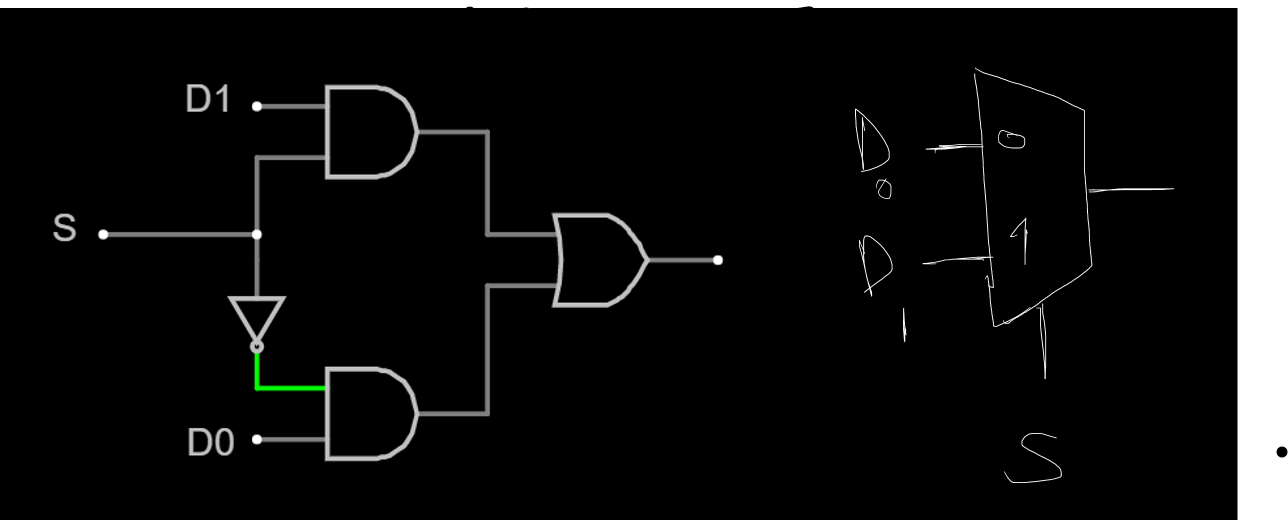
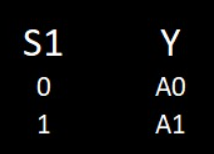
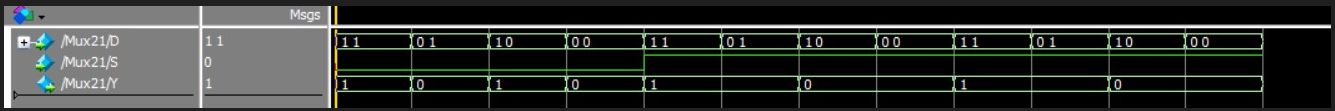
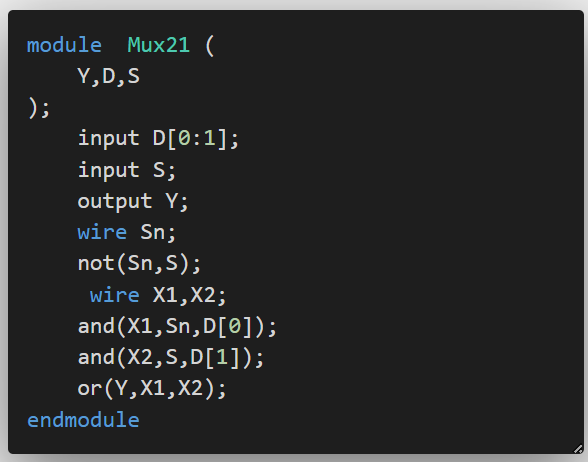


[4.3 MUX](#Table)

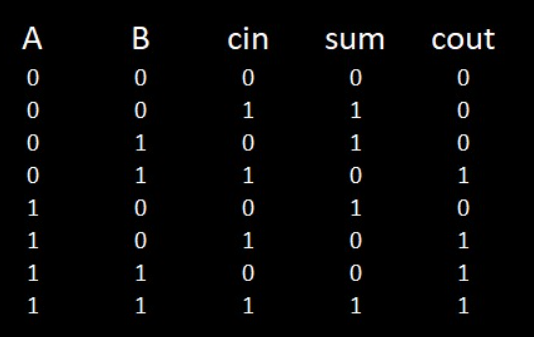


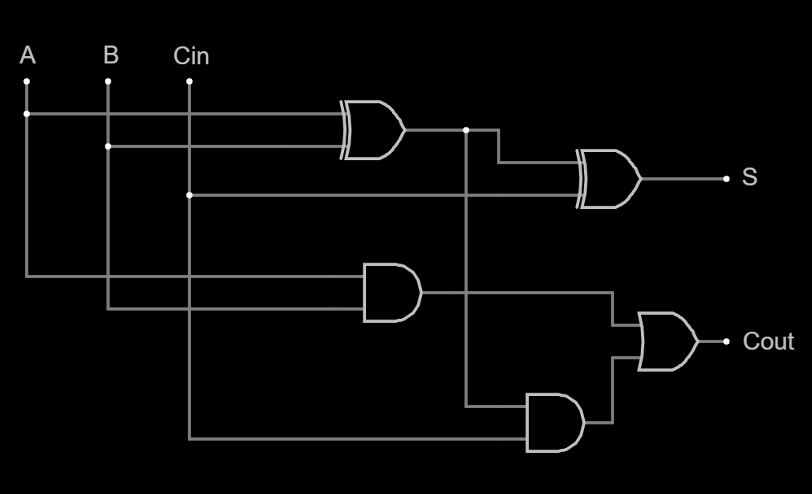


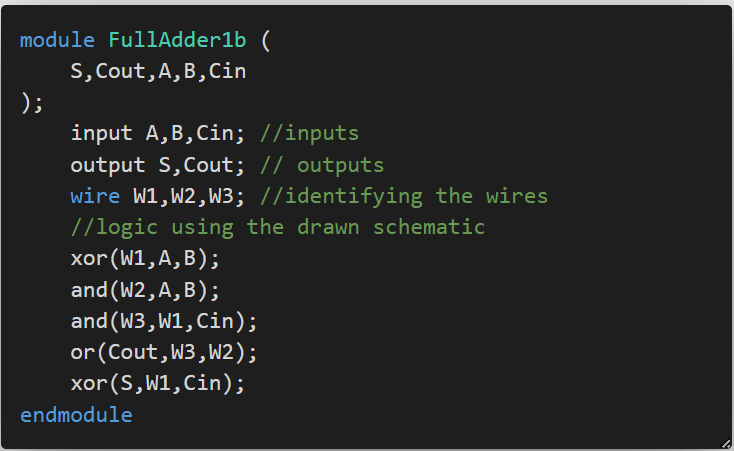
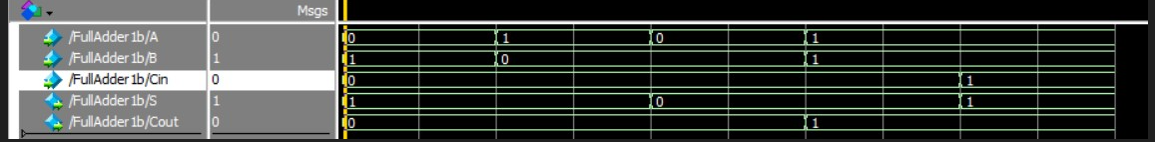
[4.4 Mux 2](#Table)

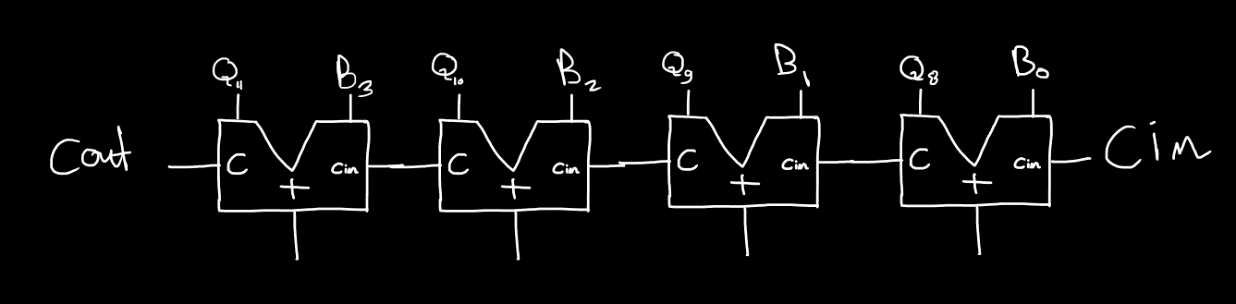
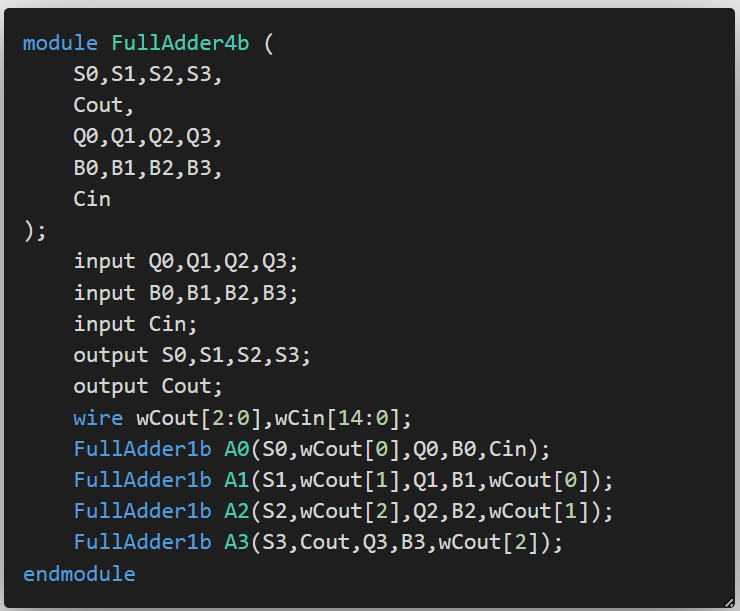




[4.5 Full Adder 1bit](#Table)



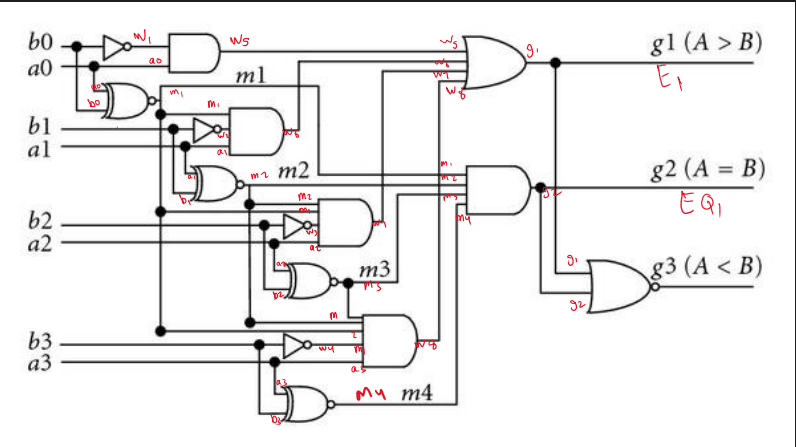


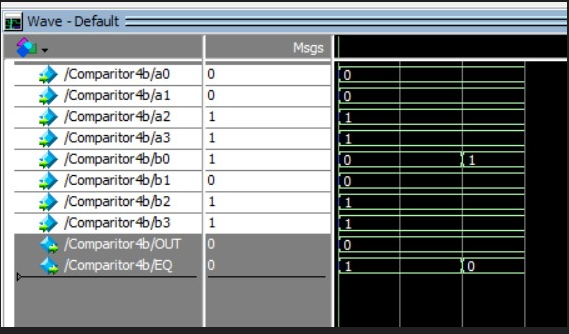
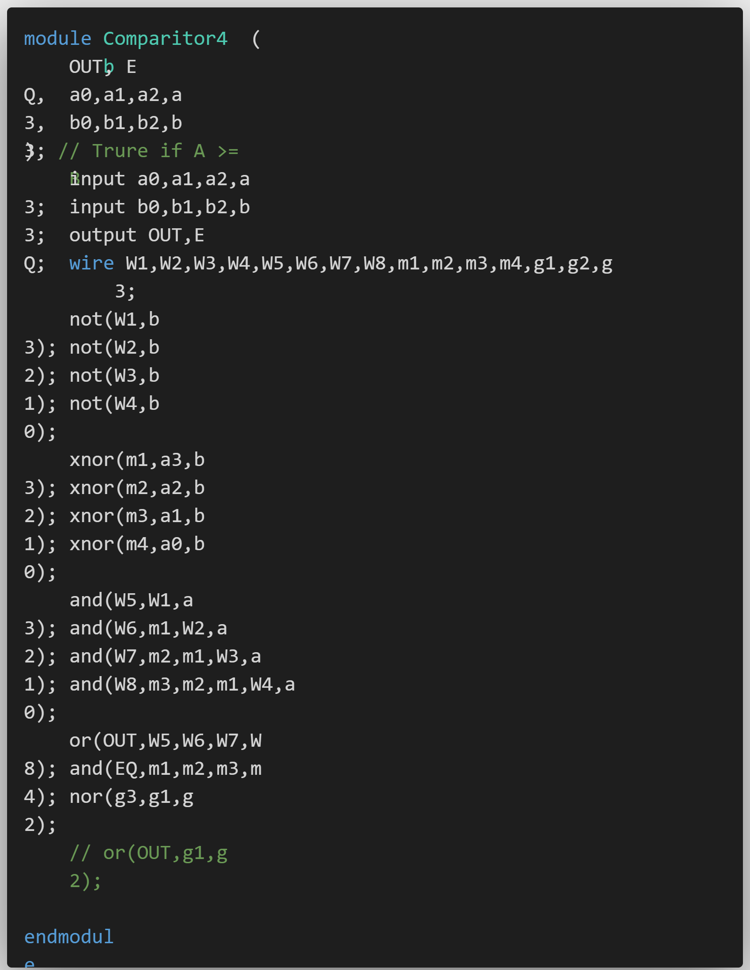
[4.6 Full Adder 4 bit](#Table)



[4.8 BCD Comparator 4bit](#Table)

Pending Truth table





Faced Problems

- We first had a problem in making the counter count up and down and stop and also load the minimum value and add so we did this buy MUX8 but if we managed to make it count up and down we had a problem at loading so we figured out a way to make the counter act like a register using the mux8.

- Then we had a problem in resetting the value of the counter to zero after it reaches the 9 so we did that by simple and gates connected to the asynchronous reset of each JK Flip Flop.

- The logic of adding two minutes was the most difficult logic because we had to compare with the maximum value after we add and we did that using comparator

- Then we discovered a problem in simulation of adding unit, which is that the adder we used exceeded the value we wanted which is 9 and it was adding to 1111 which is 15 so we had to make a BCD adder.

- Making BCD adder was very tricky to understand and the resources were not easy.

- We faced the same problem in subtraction as well as adding so we had to make BCD subtractor.

- We tried to not manipulate the clock in the clock divider but we couldn’t think of anything else.

- In counting down there were a problem in resetting the value of the counter to 9 or 5 again and that was solved by simple logic and gates and inverters that takes input from the Q’s and the output of it is going to the reset and preset of the counter.

- We used at first Flastad website to simulate and then we used Logisim for better space and better control

- We had a problem after designing all counters in Logisim, the counter didn’t function in Model Sim and we fixed it by and gates as mentioned before.

- The connection of each counter using trigger we thought of making an and gate and its output is connected to the J K and the input of the AND gate is the trigger and the output of the MUX 8 but this violated our logic in loading so we put the trigger in the CLK of the following counter.