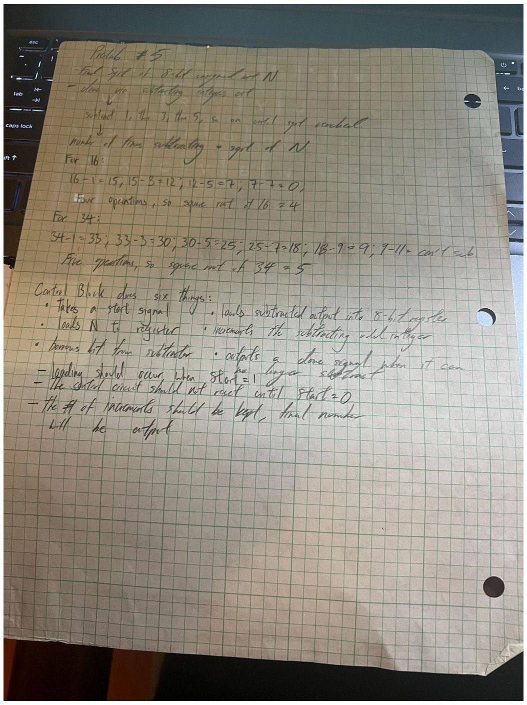
EEC 180 Lab 5
2/22/25
David Zhang and Justin Hsu

Prelab:



Prelab #5

- Find Sqrt of 8-bit unsigned int N
- done via subtracting integers out

↓

subtract 1, then 3, then 5, so on until sqrt reached.

↓

number of times subtracting = sqrt of N

For 16:

16-1=15; 15-3=12; 12-5=7; 7-7=0;

Four operations, so square root of 16 = 4

For 34:

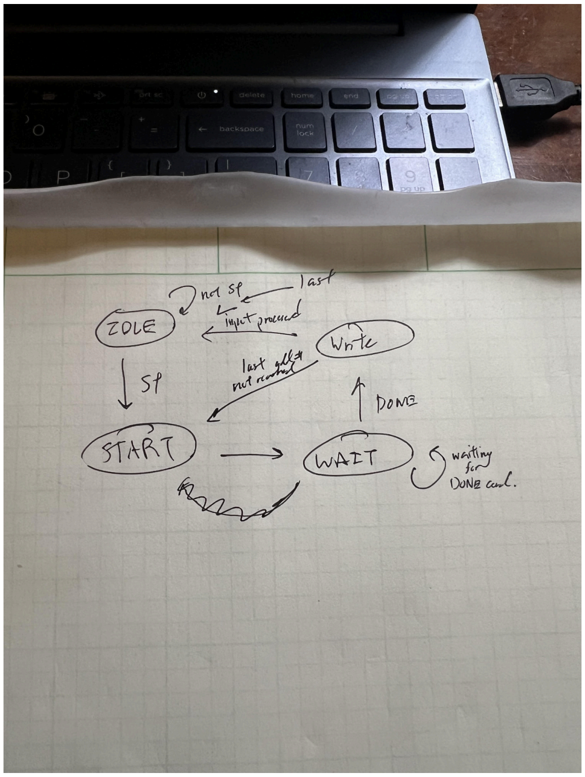
34-1=33; 33-3=30; 30-5=25; 25-7=18; 18-9=9; 9-11= can't sub.

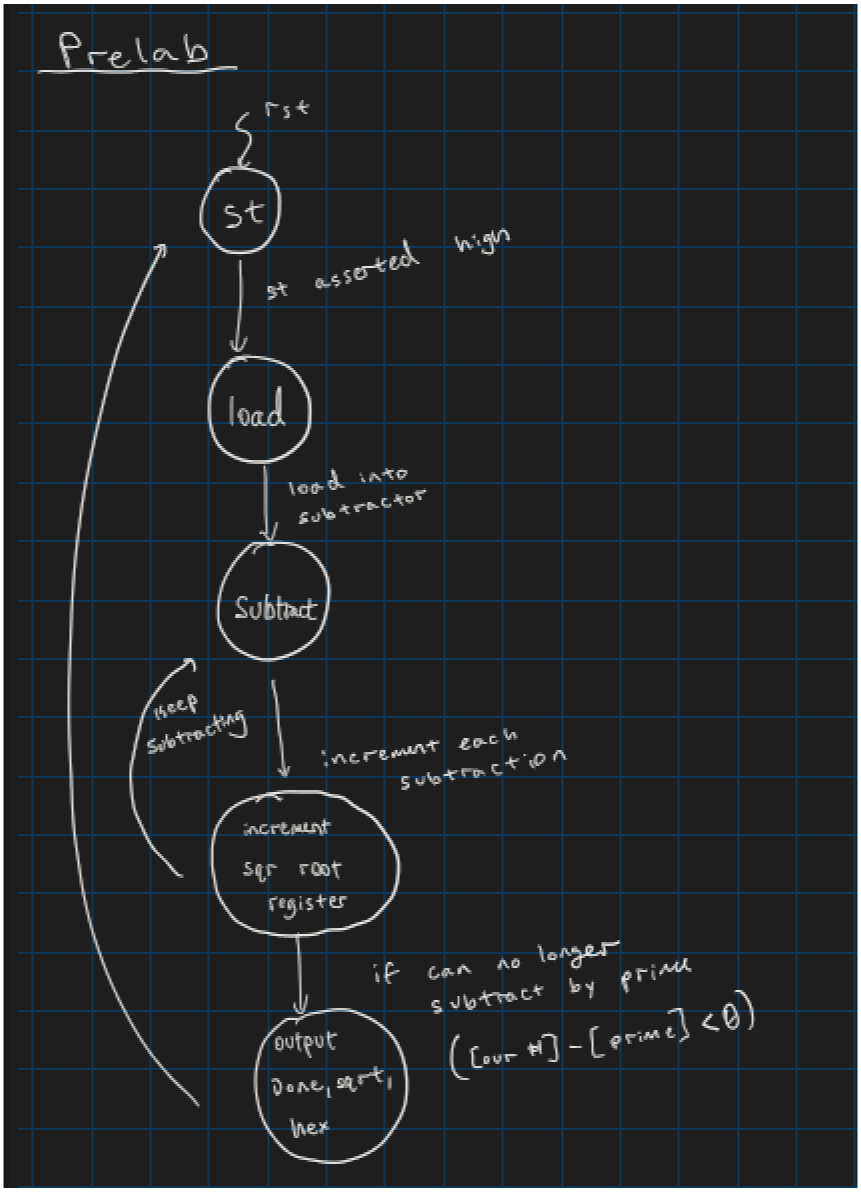
Five operations, so square root of 34 = 5

Control Block does six things:

- Takes a start signal
- loads N to register
- borrows bit from subtractor
- loads subtracted output into 8-bit register
- increments the subtracting odd integer
- outputs a done signal when it can no longer subtract

- loading should occur when Start=1
- The control circuit should not reset until Start=0
- The # of increments should be kept, final number will be output

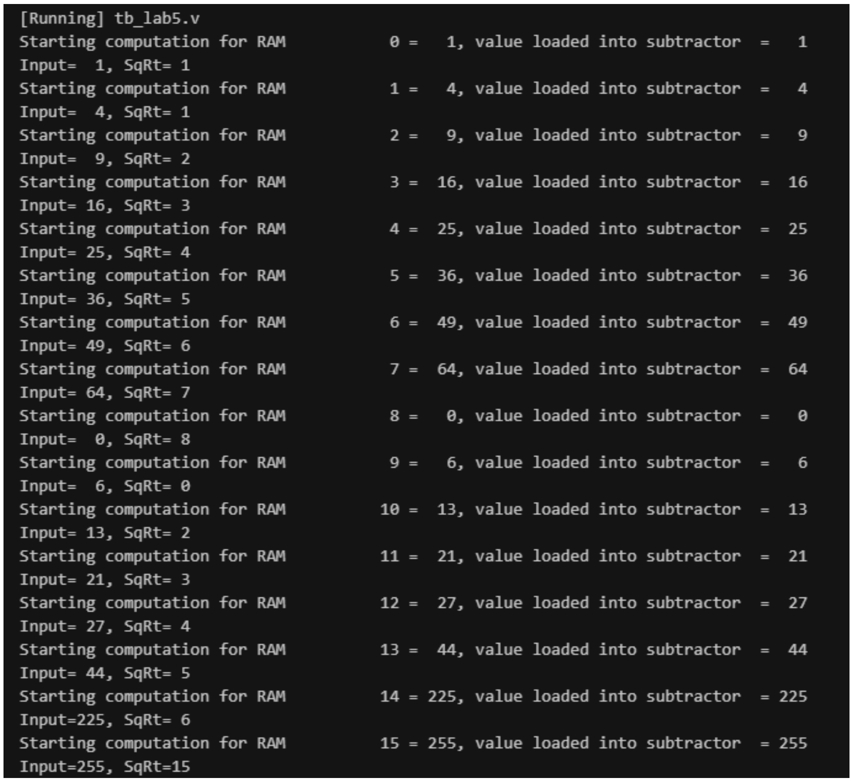




Part 1:

Part 1 works as expected. We are able to use the switches to input data and write to the ram, then we are able to read from the ram and display onto our hex output. This works for both ram modules instantiated and both hex displays.

Part 2:



```
[Running] tb_lab5.v
Starting computation for RAM          0 =   1, value loaded into subtractor  =   1
Input=  1, SqRt= 1
Starting computation for RAM          1 =   4, value loaded into subtractor  =   4
Input=  4, SqRt= 1
Starting computation for RAM          2 =   9, value loaded into subtractor  =   9
Input=  9, SqRt= 2
Starting computation for RAM          3 =  16, value loaded into subtractor  =  16
Input= 16, SqRt= 3
Starting computation for RAM          4 =  25, value loaded into subtractor  =  25
Input= 25, SqRt= 4
Starting computation for RAM          5 =  36, value loaded into subtractor  =  36
Input= 36, SqRt= 5
Starting computation for RAM          6 =  49, value loaded into subtractor  =  49
Input= 49, SqRt= 6
Starting computation for RAM          7 =  64, value loaded into subtractor  =  64
Input= 64, SqRt= 7
Starting computation for RAM          8 =   0, value loaded into subtractor  =   0
Input=  0, SqRt= 8
Starting computation for RAM          9 =   6, value loaded into subtractor  =   6
Input=  6, SqRt= 0
Starting computation for RAM         10 =  13, value loaded into subtractor  =  13
Input= 13, SqRt= 2
Starting computation for RAM         11 =  21, value loaded into subtractor  =  21
Input= 21, SqRt= 3
Starting computation for RAM         12 =  27, value loaded into subtractor  =  27
Input= 27, SqRt= 4
Starting computation for RAM         13 =  44, value loaded into subtractor  =  44
Input= 44, SqRt= 5
Starting computation for RAM         14 = 225, value loaded into subtractor  = 225
Input=225, SqRt= 6
Starting computation for RAM         15 = 255, value loaded into subtractor  = 255
Input=255, SqRt=15
```

Our code works as expected when uploaded onto the FPGA. However, in the simulation all the outputs are offset by one because since this project required a sequential circuit, so when it executes for the first time, there is no value for the first square root, since everything is being executed in parallel rather than consecutively, and as a result it will generate a one just as a placeholder, and after the first cycle it will work.

Work Distribution

Work was distributed evenly for both part I and part II. For both parts, each lab partner initially completed their own design. Whichever design was more functional initially would get debugged. Debugging was split between both members of the group.