Assignment 2 - Part 2

Aman Verma, Shreyansh Singh

2019CS50419, 2020CS10385

1 Design Decisions

- Image is copied from ROM to RAM since the image is needed to be accessed multiple times. Thus for both layers image is accessed from RAM, and weights are accessed from ROM.
- The output from MAC is transferred to the shifter and then to the comparator before storing it in RAM.
- For calculating the max value at the end, a Max_value module is used, and then its output is passed to the Display module, which calculates the cathode and anode values.

2 Components

** RAM, ROM, Shifter, Comparator, Registers, and MAC were already described in part1

2.1 Max_value Module

To calculate the maximum value from the final output, this component is used. When the control signal is '1', it compares the value with the maximum value found till now and store its location when needed.

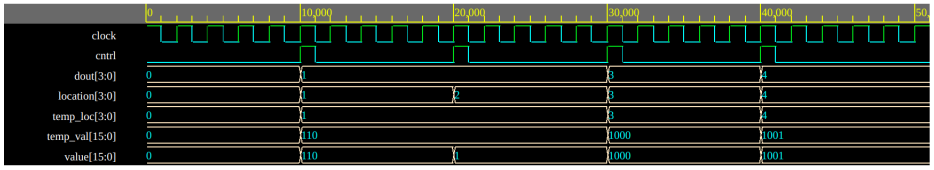


Figure 1: Max_Value Component

2.2 Control_Path Module

This module implements the control part. It provides all the control signals needed in the data part. It maintains the state and counters as internal signals. There are in total 29 states in our FSM.

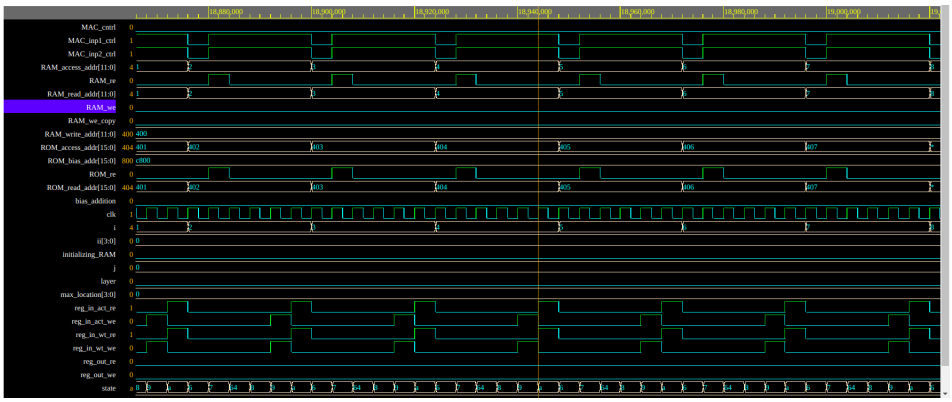


Figure 2: Control Part Component

2.3 Display Module

This module takes the answer to be shown on the FPGA board and returns the anode and cathode values. It uses bcd_to_ssd and clock divider module for its computation.

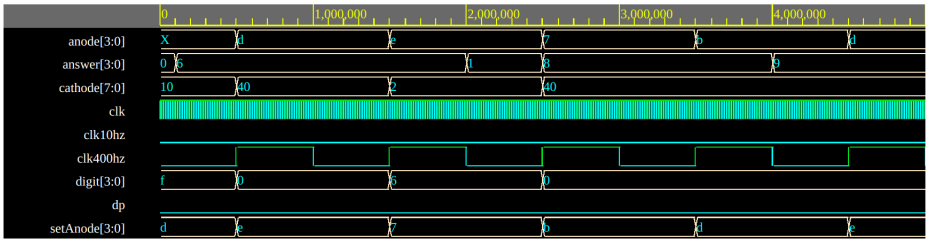


Figure 3: Display Component

2.4 Bcd_to_ssd Module

This module takes in the digit to be displayed and returns the cathode values.

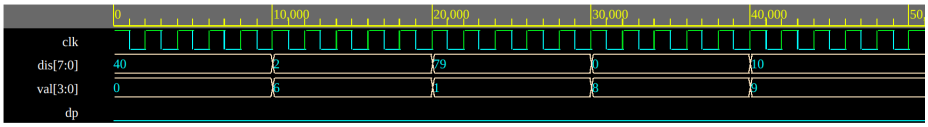


Figure 4: BCD_to_SSD Component

2.5 Clock Divider Module

This module calculates the refresh rate, which is used to drive the seven segment display output in the Display module.

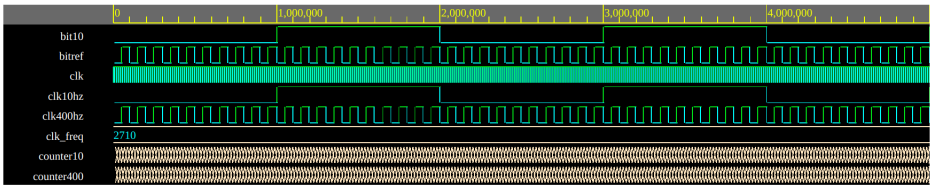


Figure 5: Clock Divider Component

2.6 Main Module

This module implements the data part. It assigns the control signals from the control part to all the modules and also implements multiplexing using control signals.

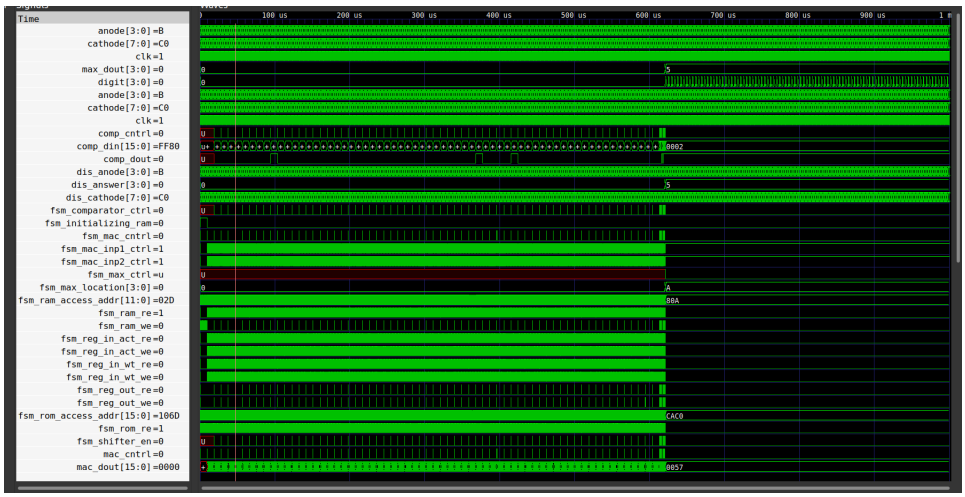


Figure 6: Main Component Overview

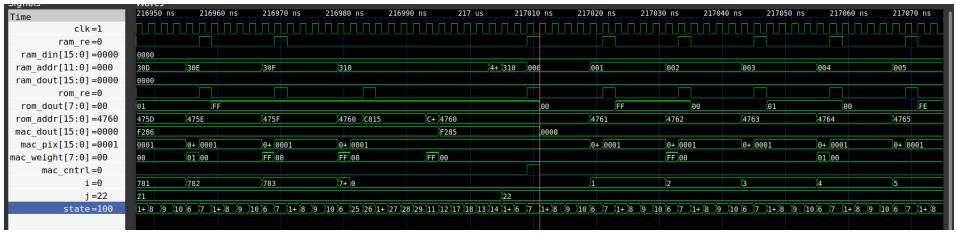


Figure 7: RAM, ROM, MAC signals in Main Component

Figure 8: Signals after final value is found in Main Component

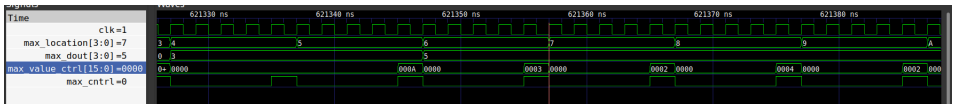


Figure 9: Max_Value Signals in Main Component

3 Block Diagram

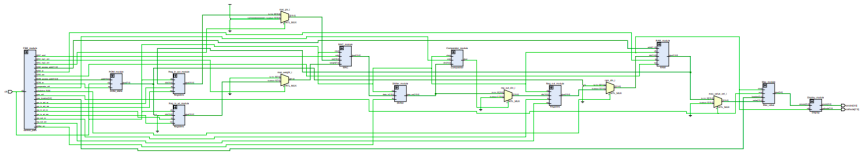


Figure 10: Block Diagram

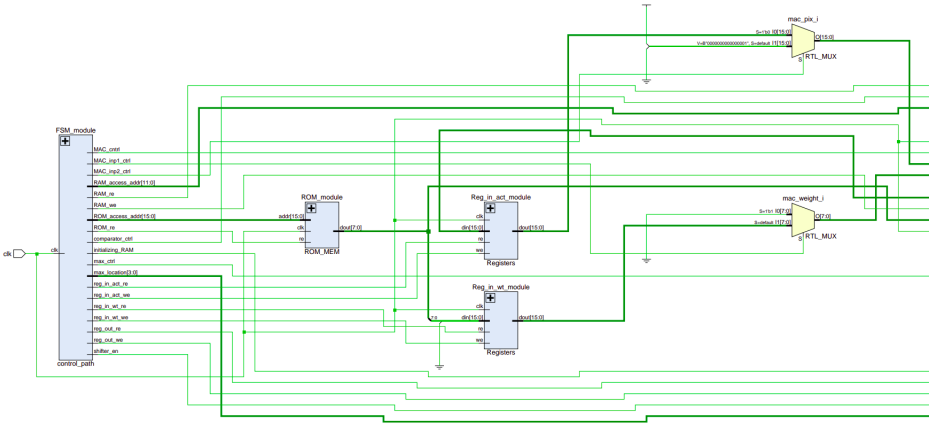


Figure 11: Block Diagram Part1

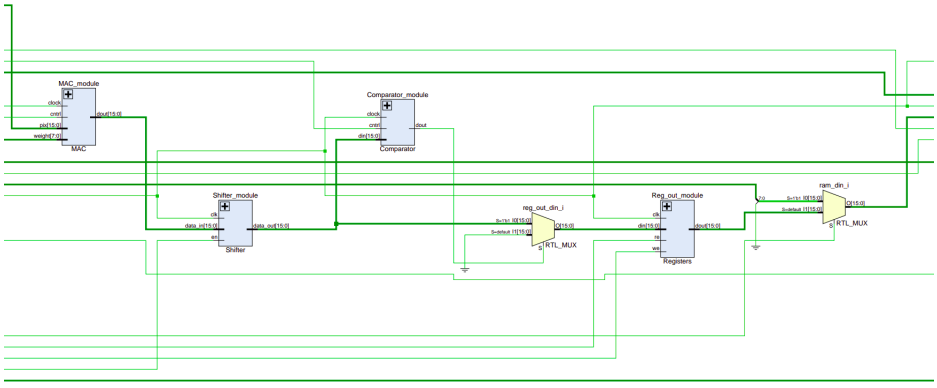


Figure 12: Block Diagram Part2

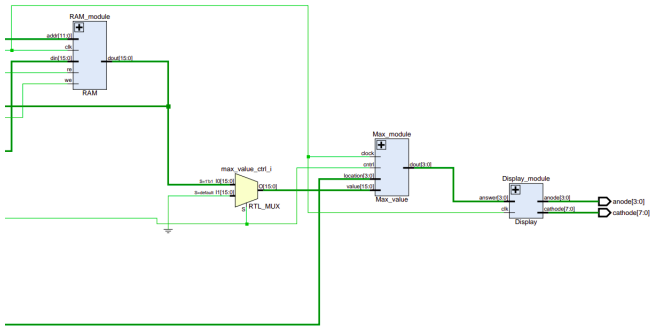
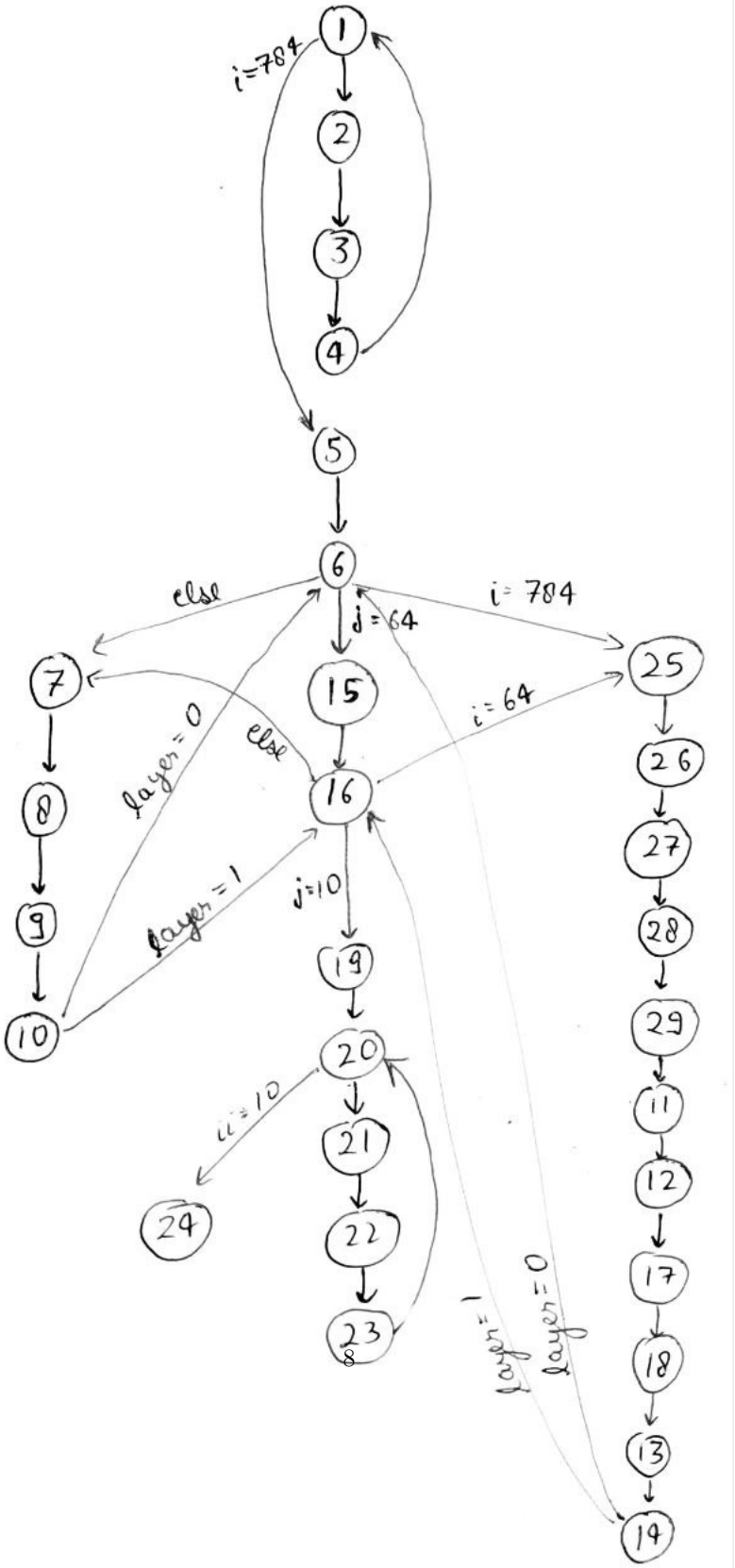


Figure 13: Block Diagram Part3

4 State Diagram



Description of States

(State 1-4 are used to read images from ROM and store them in RAM) 1: Initialise the ROM address to the image which is to be read.
2: Wait cycle for ROM to complete its read.
3: Write enable in RAM.
4: Update ROM address for next read and wait for RAM to complete its write.

(State 5-10 are used to read images from RAM and weight from RAM and pass them to MAC for layer 1)
5: Initialise the address of RAM and ROM for the image and weight to be read.
6: Start reading from RAM and ROM.
7: Wait cycle for RAM and ROM read to be completed.
8: Write ROM and RAM output to register.
9: Read image and weight from registers.
10: Pass the weight and image to MAC from registers and update RAM and ROM address to be read.

(State 15-16 and 7-10 are used to read images from RAM and weight from RAM and pass them to MAC for layer 2)
15: Initialise the address of RAM and ROM for the image and weight to be read.
16: Start reading from RAM and ROM.

(State 25-29 are used to add bias)
25: Initialise ROM address.
26: Wait cycle for bias to be read from ROM.
27: Write bias in the register.
28: Read bias from the register.
29: Pass bias from register to MAC.

(States 11, 12, 17, 18, 13, and 14 are used to write MAC output to RAM after shifting and ReLU)
11: Pass MAC output to the shifter.
12: Pass the shifter output to the comparator.
17: Write comparator output to register.
18: Read from the output register.
13: Write output register value to RAM.
14: Wait cycle for RAM write to be completed and update RAM write address.

(States 19-23 are used to calculate the maximum value from the output layer)
19: Initialise RAM address.
20: Start reading from RAM.
21: Wait cycle for RAM to complete its read.
22: Compare this value.
23: Loop to check the next value.

24: End State.