REPORT

Name- Bhupesh

Entry No.- 2021CS10101

Name – AStha Meena

Entry No. – 2021CS10122

APPROACH:

MODULES:

1. Reg3: a register for storing 16 bit unsigned.
2. Reg1, Reg2: a register used for storing 8 bit unsigned. Because we want two input matrices
3. MAC : a multiplier accumulator unit.

4.controlFSM: accumlates all modules together. registers are used for storing values from rom and sending these to mac , mac then accumulates the product and stores it at the next register , register writes the stored values in the ram,FSM activates and de-activates registers at different instants to garb values at specified address.

5.The addresses of ram and roms are synchronized through the proceses of control fsm

Multiply-Accumulate block (MAC)

The first component to be designed in this assignment is a MAC unit that has a 8x8 multiplier, a register and a 16-bit adder to keep accumulating the products. Figure 5 shows the high-level overview of a multiplieraccumulate unit with the required input and output ports. For performing matrix multiplication, you will read out two 8 bit input values from the memory, perform the multiplication using multiplier. The multiplier will generate a 16-bit output which will be sent to the adder for accumulation. Once all the input values from row and column have been multiplied and accumulated, the final output will be stored in the RAM. cntrl signal can be used to tell the unit whether it is the first product of the compute or not. If it is the first product, assign sum to the first product. After that the product needs to be accumulated with the previous sum. The other signals are self-explanatory.

Memories

The design requires memories to store the inputs and the output matrices. We will store the input matrices in a read-only memory (ROM) the outputs matrices into a random-access memory (RAM). 2 Figure 4: Overview of the control and data path for one set of inputs from one layer to generate one partial output

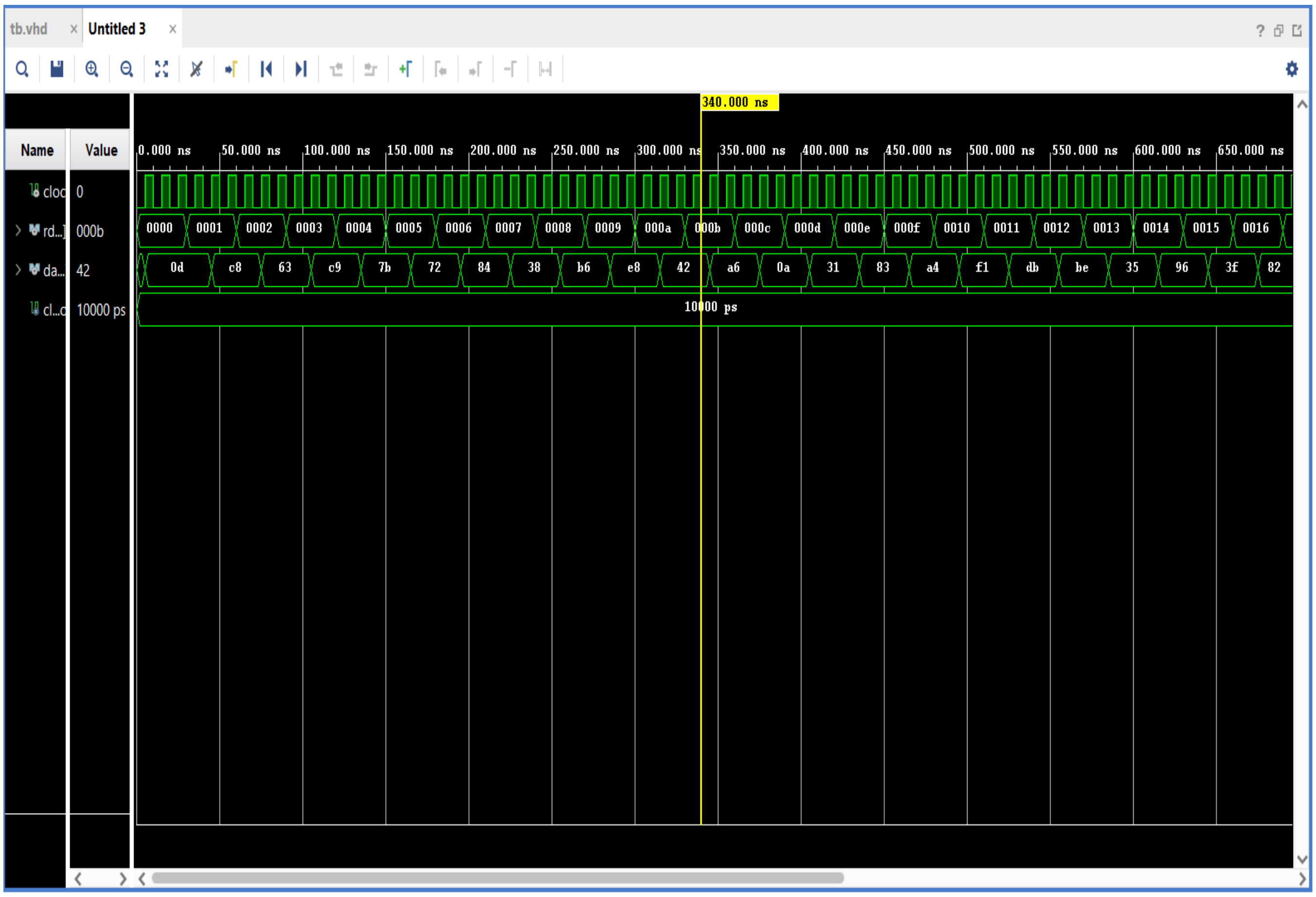
Read-Only Memory or ROM This memory will be used to store the following: • Input matrix: 128x128 image with each value as 8-bit unsigned integer For storing the input we need 16,384 words, 8 bits wide. These will be stored at address 0 (000016) onwards. The address width of the memory should be at least 14-bit.

Random-Access Memory or Read-Write Memory or RAM This memory will be used to store the final output matrix of size 128x128, with each value as 16-bit unsigned integer. 2.3.3 Registers Registers are sequential elements that store temporary data across clock cycles in a multi-cycle design. The signal description is given in Figure 6. You can use these registers as per your design requirements. Control FSM

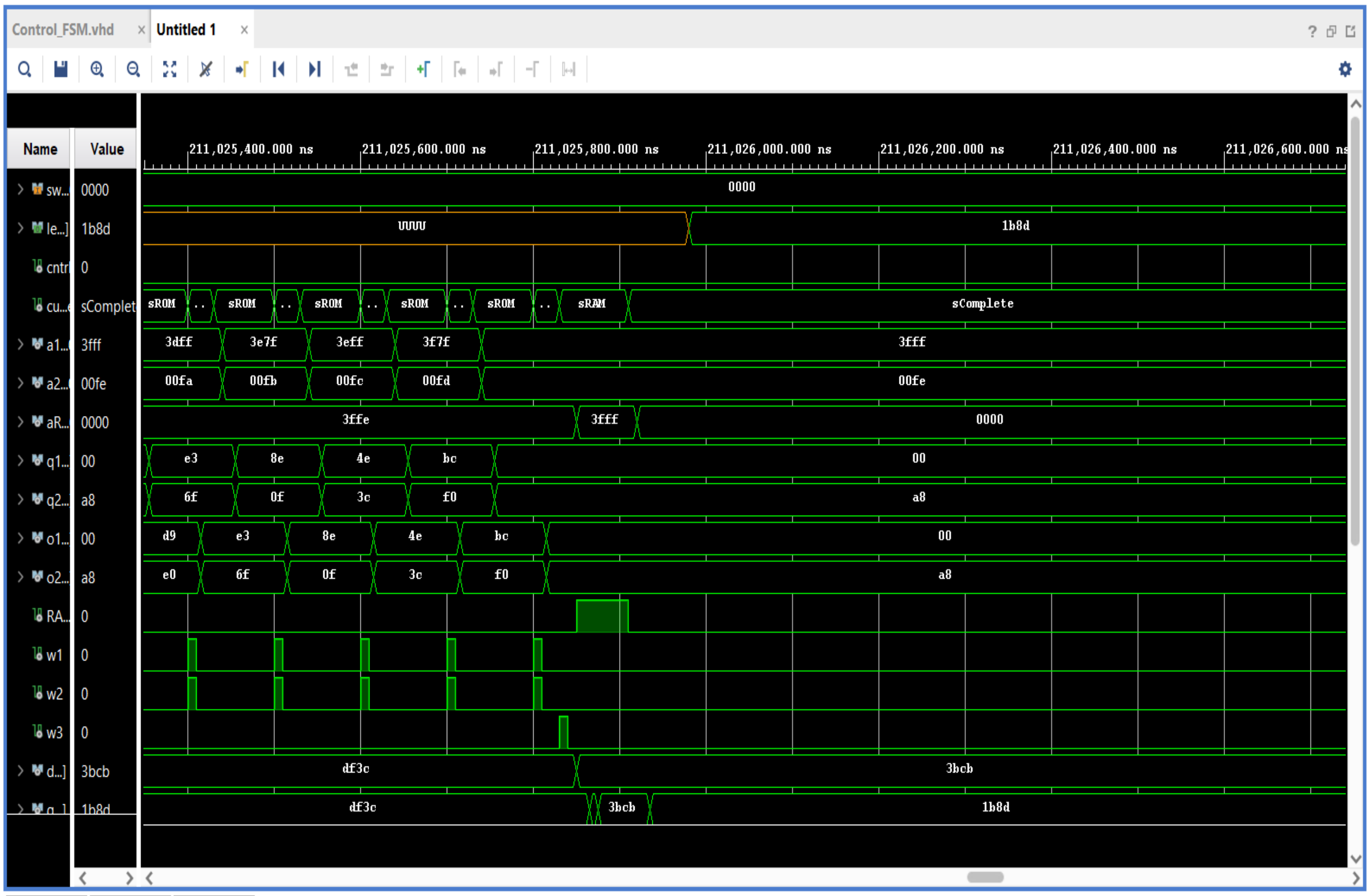
The control FSM is basically an FSM in your design which is supposed to do all the sequencing of the data required starting from generating addresses of the data in the order in which you want to read the data, generating the cycle-wise signals to be sent to the datapath to control the modules in the design. 2.5 Output • You need to verify the output using simulation and show waveforms in your report. • You will be given a special input matrix during the demo. You will be loading that input matrix into the memory. You will be given 2/4 random matrix indices and you will have to display the hexadecimal value of the output matrix at that index on the 7-segment display on the board.

Simulations :

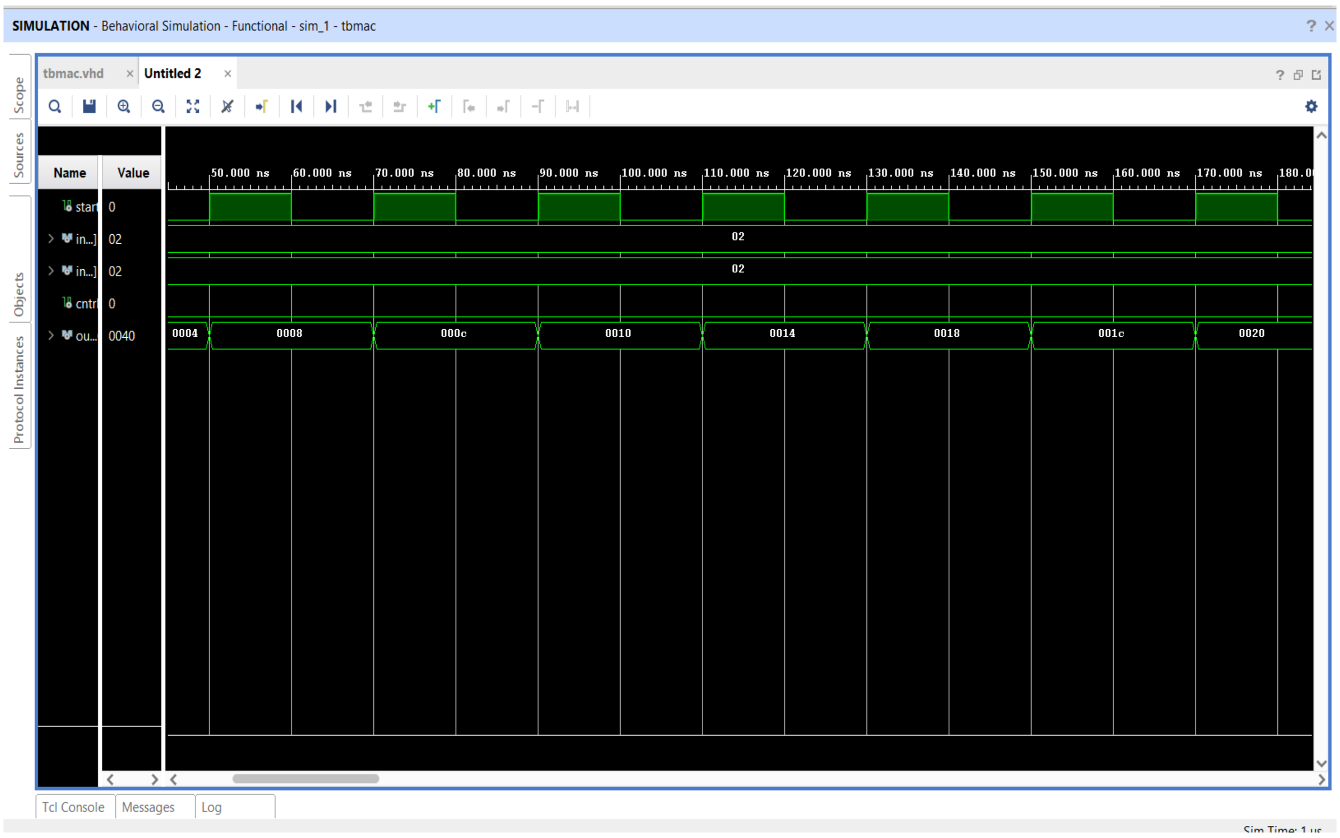
ROM simulation



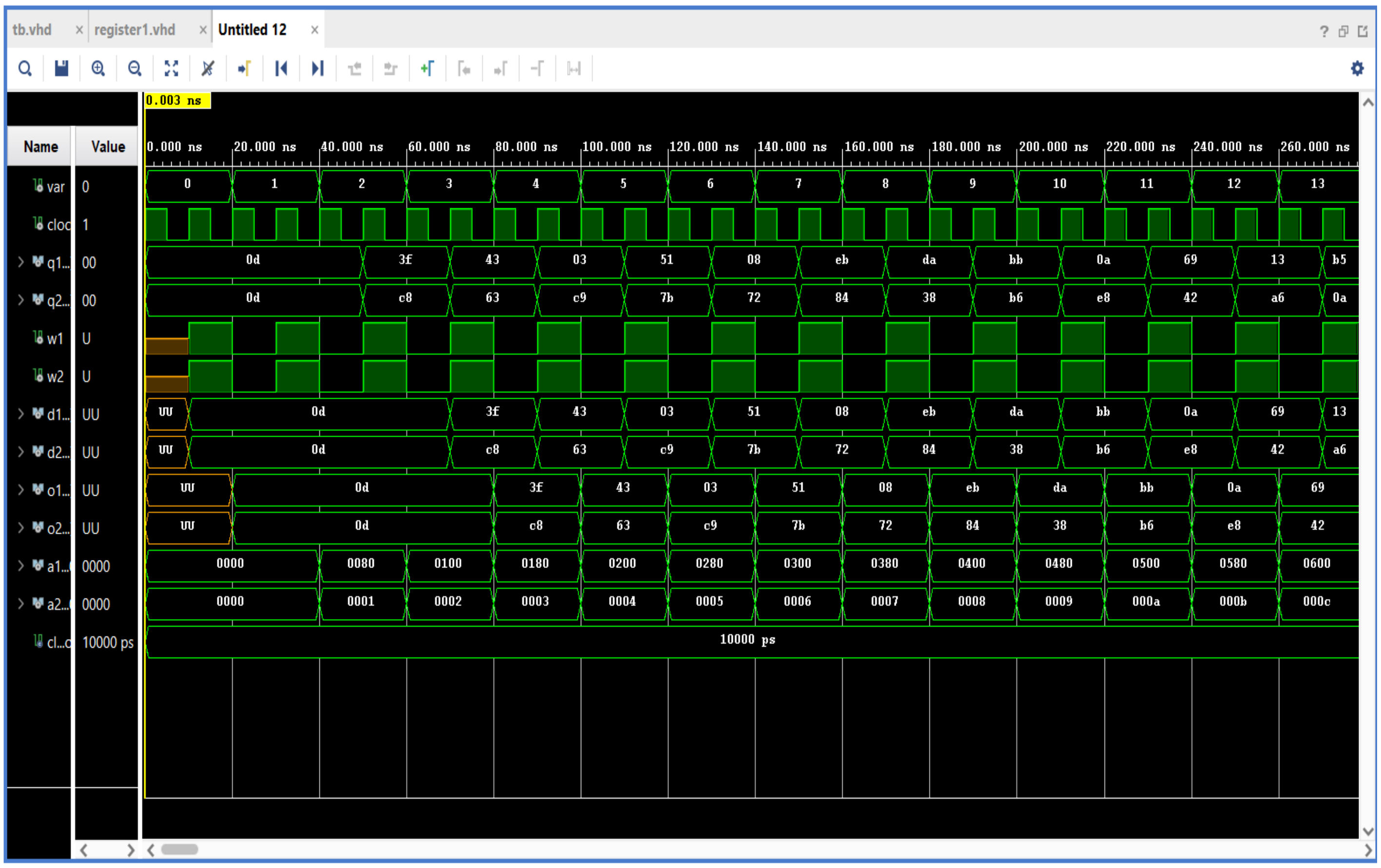
FSM simulation



MAC simulation



Register with RAM simulation


tb.vhd
register1.vhd
Untitled 12
0.003 ns
Name
Value
var 0
cloc 1
q1... 00
q2... 00
w1 U
w2 U
d1... UU
d2... UU
o1... UU
o2... UU
a1... 0000
a2... 0000
cl...d 10000 ps
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
120.000 ns
140.000 ns
160.000 ns
180.000 ns
200.000 ns
220.000 ns
240.000 ns
260.000 ns
0 1 2 3 4 5 6 7 8 9 10 11 12 13
0d 3f 43 03 51 08 eb da bb 0a 69 13 b5
0d c8 63 c9 7b 72 84 38 b6 e8 42 a6 0a
UU 0d 3f 43 03 51 08 eb da bb 0a 69 13
UU 0d c8 63 c9 7b 72 84 38 b6 e8 42 a6
UU 0d 3f 43 03 51 08 eb da bb 0a 69
UU 0d c8 63 c9 7b 72 84 38 b6 e8 42
0000 0080 0100 0180 0200 0280 0300 0380 0400 0480 0500 0580 0600
0000 0001 0002 0003 0004 0005 0006 0007 0008 0009 000a 000b 000c
10000 ps