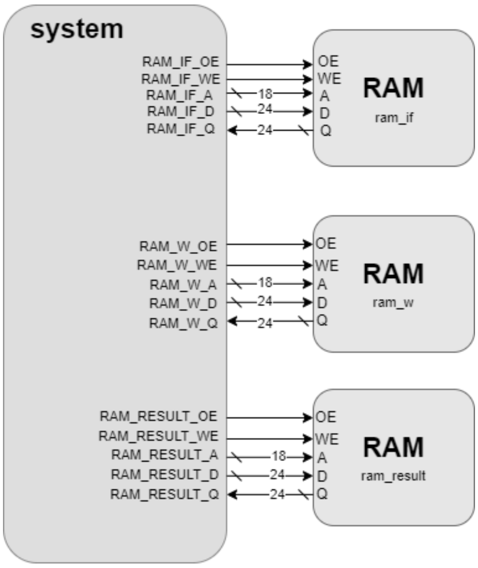
SOM Processing System Report

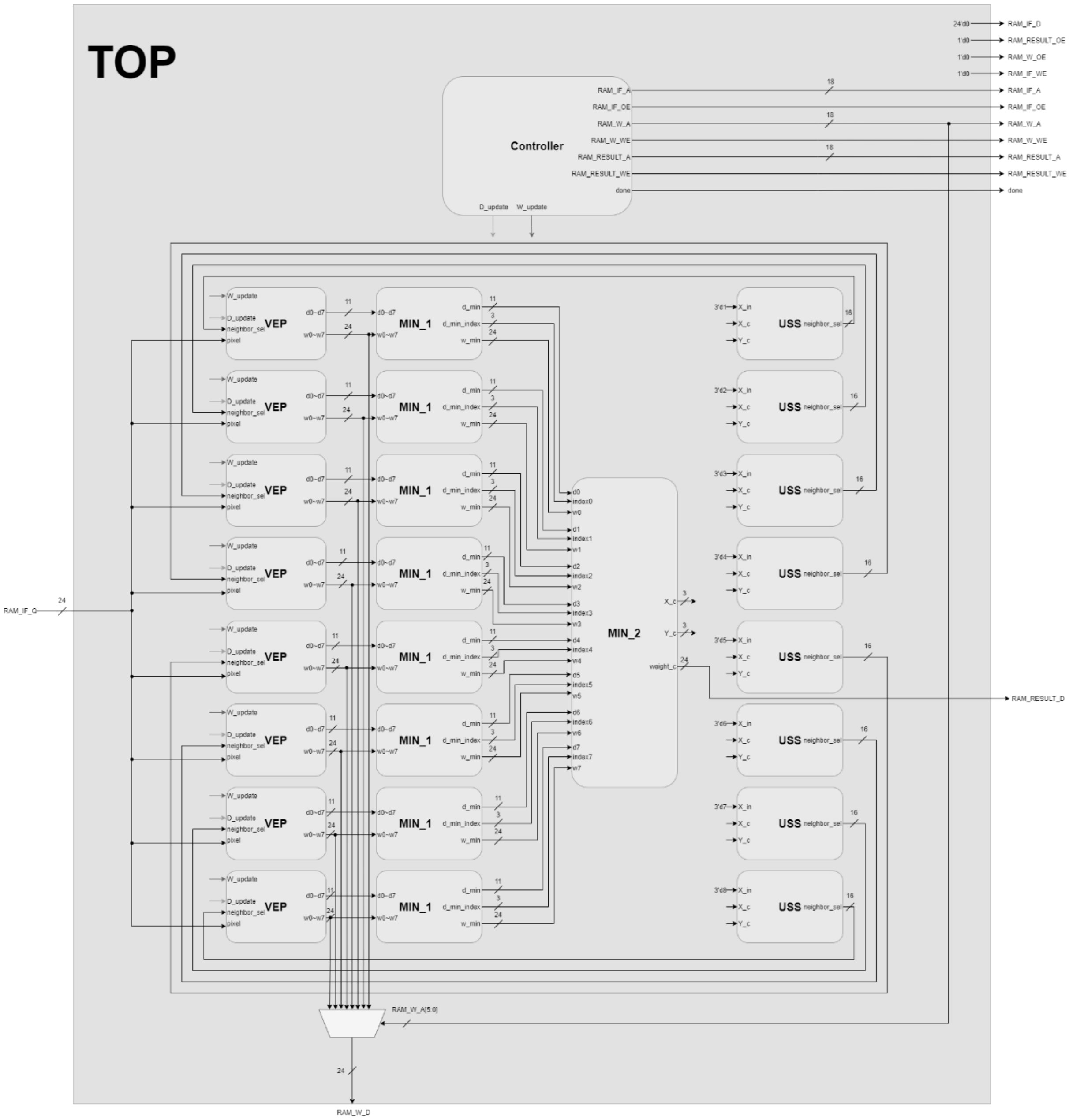
Brian (Po-Jui) Tseng

1. **Introduction**

   This project implements a **Self-Organizing Map (SOM) processing system** that integrates VEP, MIN_1, MIN_2, USS, and a Controller to perform pixel input, Manhattan distance calculation, weight updates, and image compression. **Phase 1** focuses on single-image training, providing a foundation for system operation, while **Phase 2** extends the design to multiple-image training, requiring controller modifications and resulting in longer simulation cycles. Together, these phases demonstrate the scalability of the design and the trade-offs between performance and efficiency in hardware implementation.

2. **The block diagram of system** : System includes VEP, MIN_1, MIN_2, USS, Controller.





3. Port list of each module:

    A. Controller

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
D_update	Output	1	Distance update enable, active high
W_update	Output	1	Weight update enable, active high
RAM_IF_A	Output	18	Input feature RAM address
RAM_IF_OE	Output	1	Input feature RAM output enable
RAM_W_A	Output	18	Weight RAM address
RAM_W_WE	Output	1	Weight RAM write enable
RAM_RESULT_A	Output	18	Result RAM address
RAM_RESULT_WE	Output	1	Result RAM write enable
done	Output	1	Pull to 1 if the system is done

B. **VEP**

Signal	I/O	Bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
W_update	Input	1	Weight update enable, active high
D_update	Input	1	Distance update enable, active high
neighbor_sel	Input	16(2x8)	Neighborhood function of 8 VEP weights (00=>1, 01=>0.25, 10=>0.125, 11=>0)
pixel	Input	24	Input pixel from RAM_if
d0~d7	Output	11	Manhattan distance between 8 weights
w0~w7	Output	24	8 weights

C. **MIN_1**

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
d0~d7	Input	11	Manhattan distance between 8 weights
w0~w7	Input	24	8 weights
d_min	Output	11	Minimum distance between d0~d7
d_min_index	Output	3	Index of minimum distance
W_min	output	24	Weight of minimum distance

D. **MIN_2**

signal	I/O	bit	Description
clk	Input	1	clock
rst	Input	1	Reset signal, active high
d0~d7	Input	11	Minimum distance from MIN_1
w0~w7	Input	24	Weight of minimum distance from MIN_1
index0~index7	Input	3	Index of minimum distance from MIN_1
X_c	Output	3	The X coordinate of center weight
Y_c	Output	3	The Y coordinate of center weight
weight_c	output	24	Center weight

E. USS

signal	I/O	bit	Description
clk	input	1	Clock
rst	Input	1	Reset signal, active high
X_in	Input	3	USS module index
X_c	Input	3	The X coordinate of center weight
Y_c	Input	3	The Y coordinate of center weight
neighbor_sel	Output	16(2x8)	Neighborhood function of 8 VEP weights (00=>1, 01=>0.25, 10=>0.125, 11=>0)

F. Top

Signal	I/O	bit	Description
clk	Input	1	Clock
rst	Input	1	Reset signal, active high
RAM_IF_Q RAM_W_Q RAM_RESULT_Q	Input	24	Data output from RAM
RAM_IF_OE RAM_W_OE RAM_RESULT_OE	Output	1	RAM output enable signal
RAM_IF_WE RAM_W_WE RAM_RESULT_WE	Output	1	RAM write enable signal
RAM_IF_A RAM_W_A RAM_RESULT_A	Output	18	RAM address
RAM_IF_D RAM_W_D RAM_RESULT_D	output	24	Data written into RAM
done	Output	1	Pull to 1 if the system is done

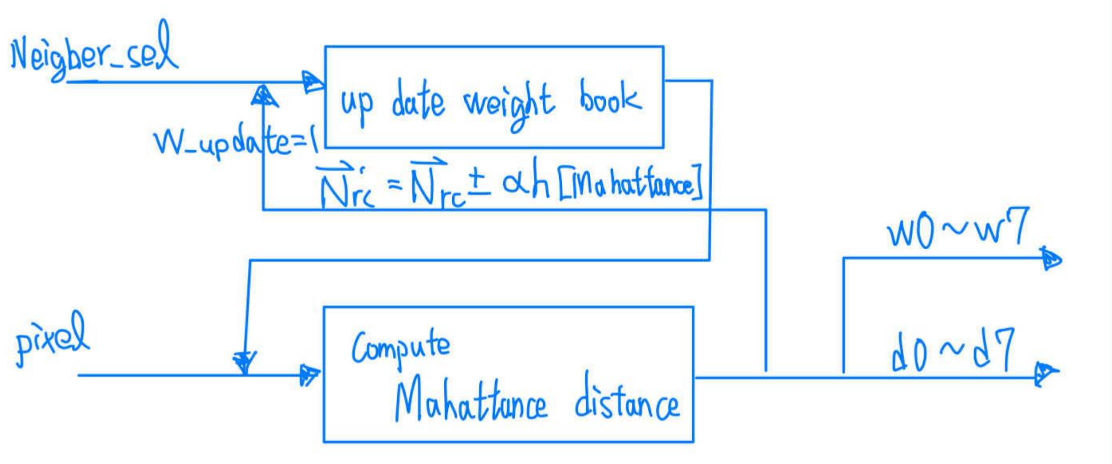
4. System Operation Flow

A. System is initialized

B. Read input pixel from RAM_if
C. find the minimum distance
D. Update the weight memory
E. Repeats the process step(a)~(c)until the last pixel of RAM_if is read
F. writes the trained codebook to the RAM_w
G. read input pixel from RAM_if and inference the picture
H. writes the lossy compression picture to the RAM_result
I. repeats the process step (f)~(g) until the last pixel of RAM_result is writed;
J. flags "done" when system is completed

5.

A. **VEP (Combinational & Sequential)**

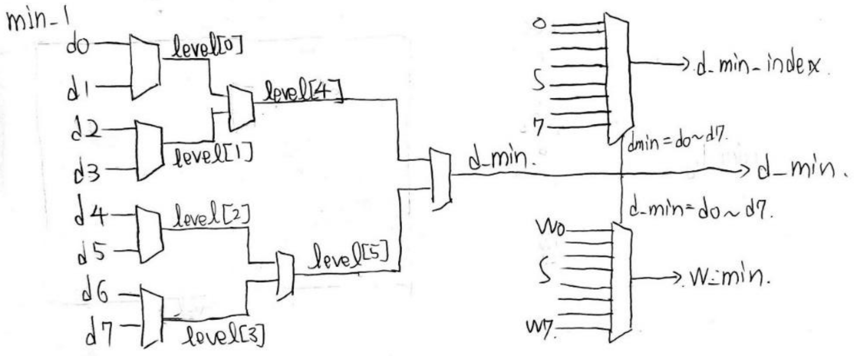


i. The VEP operates in two main phases. In the first phase, the incoming pixel is used to calculate the Manhattan distance, which is then sent to the *min* module. Based on the result, the neighbor_sel signal is generated and used to update the weight book. In the second phase, the VEP simply calculates the Manhattan distance of the incoming pixel, and the operation performed depends on whether D_update or W_update is enabled.

ii. During the first phase, D_update and W_update are alternately enabled (two cycles per period). After reading the data, the VEP calculates the Manhattan distance and stores it in a register. This avoids redundant calculations during weight book updates, helping to reduce area. The stored distance is then passed to *min_1* and *min_2*, which determine the center and generate neighbor_sel. The weight book is updated according to the neighbor_sel values.

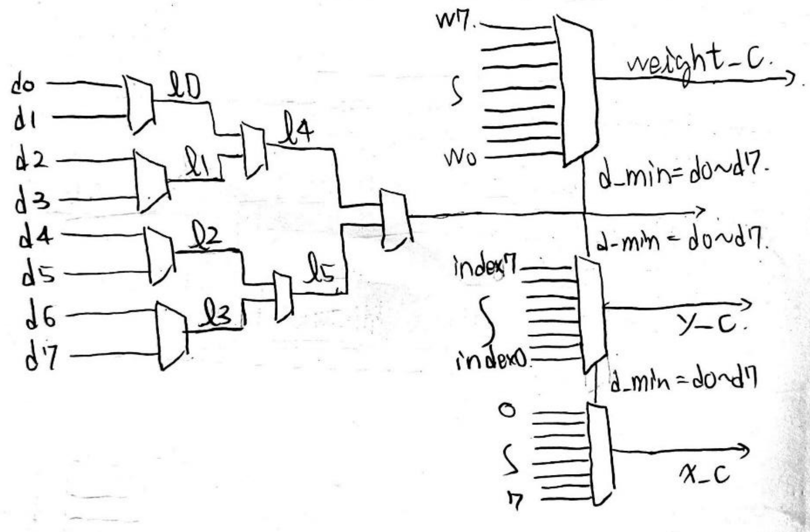
iii. In the second phase, only D_update is enabled. The primary function here is to find the weight with the shortest Manhattan distance and use it to replace the incoming pixel.

B. **MIN_1 (Combinational)**



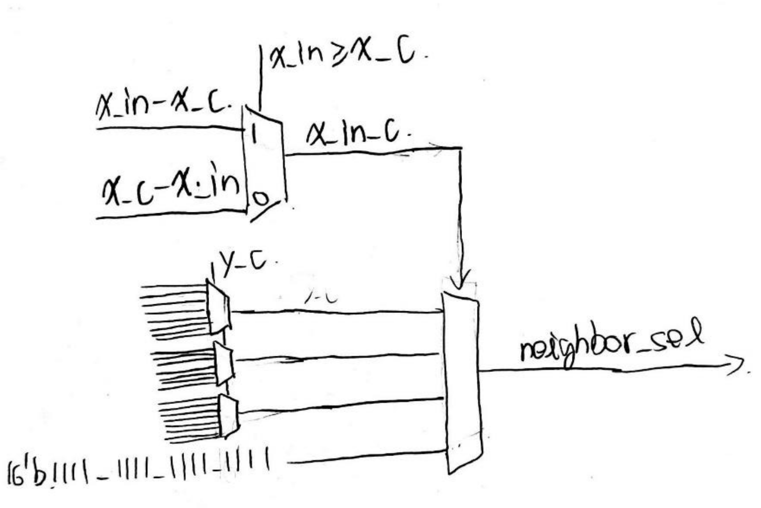
i. The implementation of *MIN_1* is similar to Lab 6. It uses a three-level multiplexer structure to select the smallest Manhattan distance. An important detail is that when two values are equal, the output will be the one with the larger index. After identifying the minimum Manhattan distance, the result is compared with the original input to determine the index of the minimum distance and the corresponding weight value. These values are then passed to *MIN_2* for further processing.

C. **MIN_2 (Combinational)**



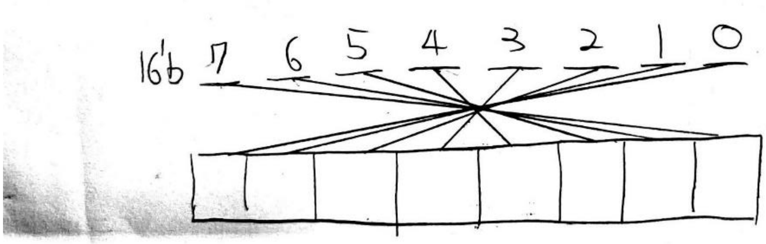
i. The implementation of *MIN_2* is similar to *MIN_1*. It also uses a three-level multiplexer structure, but instead of operating directly on pixel inputs, it selects the minimum Manhattan distance from the results provided by *MIN_1*. As in *MIN_1*, if two distances are equal, the output chosen is the one with the larger index. After determining the smallest Manhattan distance, the result is compared with the original input to identify the corresponding index. This index is output as y_c, while the originating *MIN_1* block is indicated by x_c. The associated weight value is also output, and all of these results are passed to the corresponding *USS* modules for further processing.

D. **USS (Combinational)**



i. The primary function of the *USS* module is to determine the neighbor_sel signal. From prior observation, it was found that neighbor_sel follows a certain pattern that allows simplification, eliminating the need to explicitly handle all 8×8×8 cases. The implemented approach first checks the distance between x_in and x_c: if the distance is 0, it is categorized as the first case; if the distance is 2, it is the second case; if the distance is 3, it is the third case. All other situations indicate that x_in is too far from the center, and in these cases the output is set to 16 bits of all ones. At the second level, the value of y_c is used to further determine the appropriate neighbor_sel output.

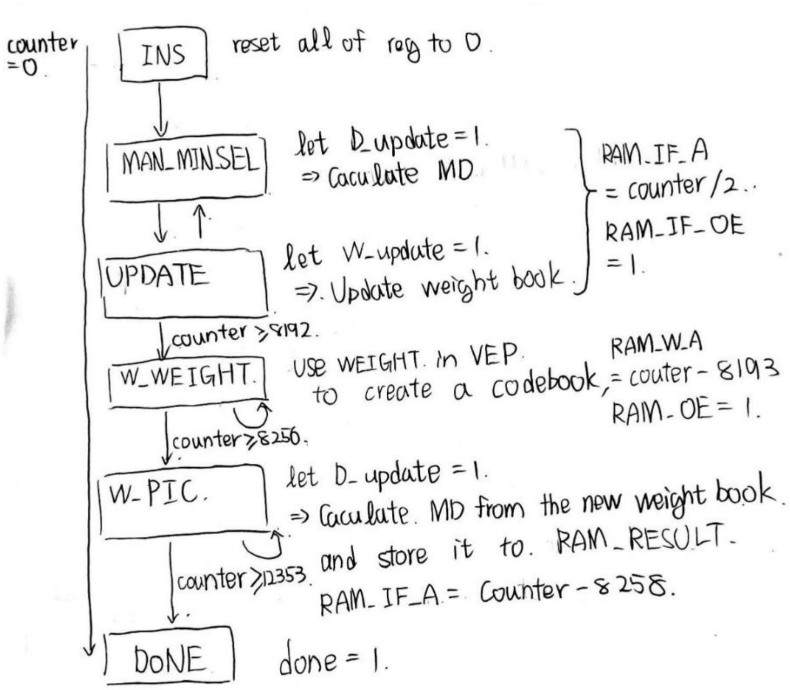
ii. An important detail is that our neighbor_sel output format is slightly different. In the weight book, the rightmost position corresponds to the leftmost two bits (15:14) of the 16-bit sel signal, meaning the order is reversed compared to the weight book. As shown in the figure, I addressed this difference within the VEP logic, ensuring that the weight book can still be updated correctly.



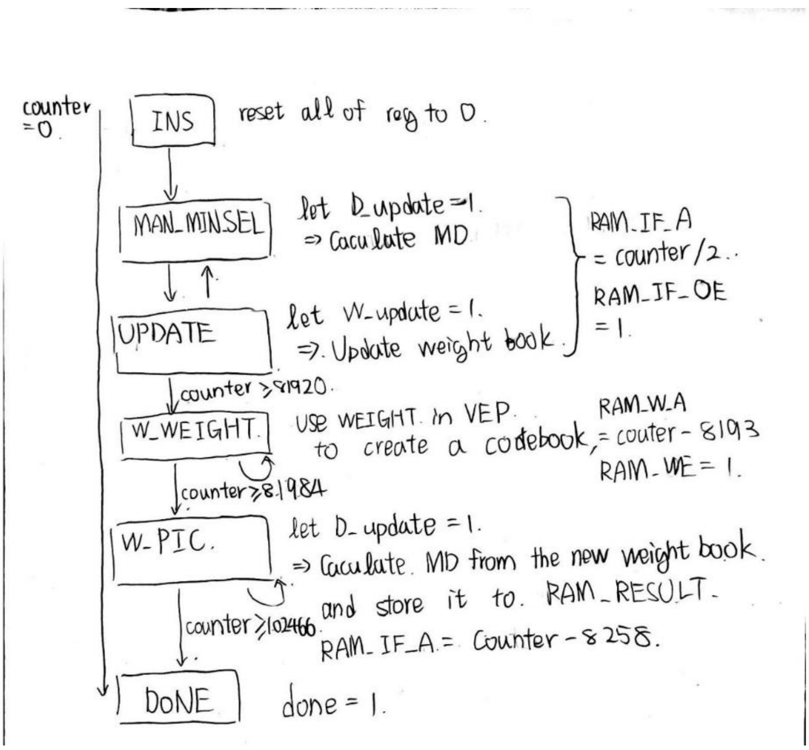
E. **Controller**

i. **State diagram in the controller**

a. Phase 1(Single picture)



b. Phase 2(Single picture)



ii. In the implementation of the controller, the operation is divided into six states: **INS, MAN_MIN_SEL, UPDATE, W_WEIGHT, W_PIC, and DONE**. The transitions between these states are determined using a counter to track the corresponding cycles. A detailed explanation of each state is as follows:

iii. **INS**:
In this state, all registers are reset to 0. After running for one cycle, the system transitions to the next state.

iv. **MAN_MIN_SEL**:
In this state, the VEP calculates the Manhattan distance and selects the center and neighbor_sel. The signal D_update is enabled, and since this state shares a two-cycle period with UPDATE, the address RAM_IF_A is assigned as counter/2. A new pixel can only be read every two cycles, and RAM_IF_OE is also enabled.

v. **UPDATE**:
In this state, the incoming neighbor_sel updates the weight book. D_update is disabled, while W_update is enabled. Similarly, RAM_IF_A = counter/2, and RAM_IF_OE remains enabled.

vi. **W_WEIGHT**:

In this state, the final codebook trained by the neighbor function is written to RAM_W. The address starts from 0, so it is set to counter minus the number of previously executed cycles. RAM_W_WE is set to 1.

vii. **W_PIC**:

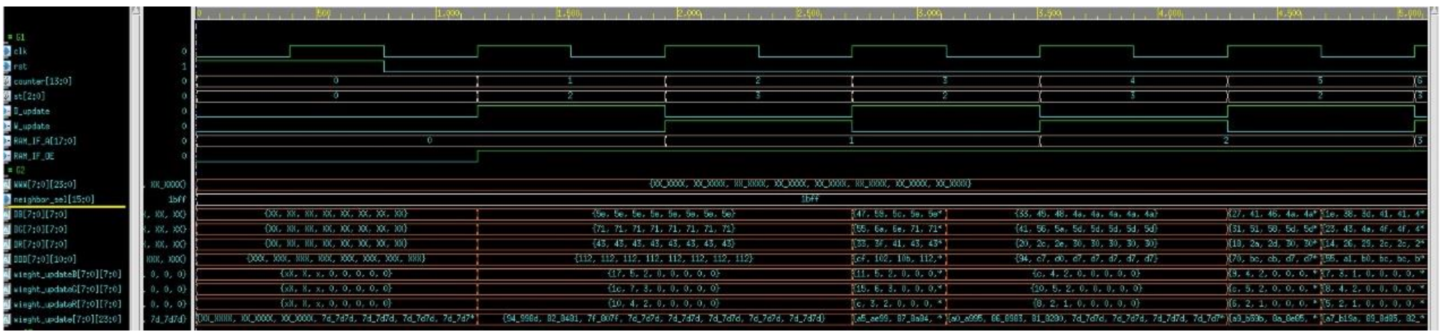
In this state, the input pixels are mapped to the codebook tags and written to the output. D_update is enabled. The address for RAM_RESULT also starts from 0, so it is set to counter minus the number of previously executed cycles. RAM_RESULT_WE is set to 1.

viii. **DONE**:

This state indicates that all computations are finished, and the done signal is set to 1.

6. **Waveform**

A. VEP Training Phase (Weight Book Update)



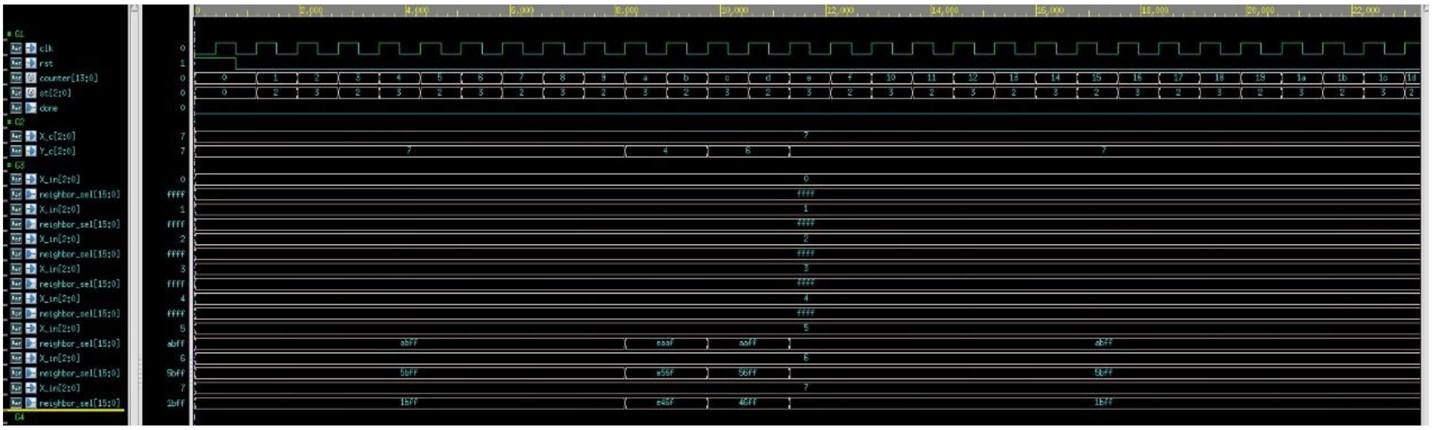
i. The waveform above illustrates the operation of the VEP during the first phase of training the weight book. It can be observed that D_update is first enabled for one clock cycle, allowing the VEP to calculate the Manhattan distance. Afterward, the circuit waits for the combinational logic to return the neighbor_sel value, at which point W_update is enabled to update the weight book based on the corresponding value.

ii. In this process, DB, DG, and DR represent the three Manhattan distance components temporarily stored within the VEP. This storage avoids redundant recalculations since the values are reused during weight book updates. The neighbor_sel function can be expressed as:

$$N(t+1)=N(t)\pm\alpha h[\text{Manhattan distance}]$$

iii. where the sign (±) is determined by comparing the original pixel with the weight value: if the pixel is larger, the weight is increased; otherwise, it is decreased. The weight_update signal indicates the updated weights stored in the VEP. The design chooses to rely on VEP for this stage because, when all initial values are the same, *MIN_2* prioritizes updating the entries with larger indices.

B. USS Neighbor Selection Operation



i. This waveform illustrates the operation of the **USS** module. From the signals, we can observe the mapping of sel bits to different values of **Y** (sel[15:14] corresponds to Y=7, sel[13:12] to Y=6, sel[11:10] to Y=5, sel[9:8] to Y=4, sel[7:6] to Y=3, sel[5:4] to Y=2, sel[3:2] to Y=1, and sel[1:0] to Y=0).

ii. **When X_c = 7, Y_c = 7**:

a. X_in[5] = 1010_1011_1111_1111
b. X_in[6] = 0101_1011_1111_1111
c. X_in[7] = 0001_1011_1111_1111
d. For all other cases where the distance between X_c and X_in exceeds 3, the output sel is 16 bits of all ones, consistent with the expected design.

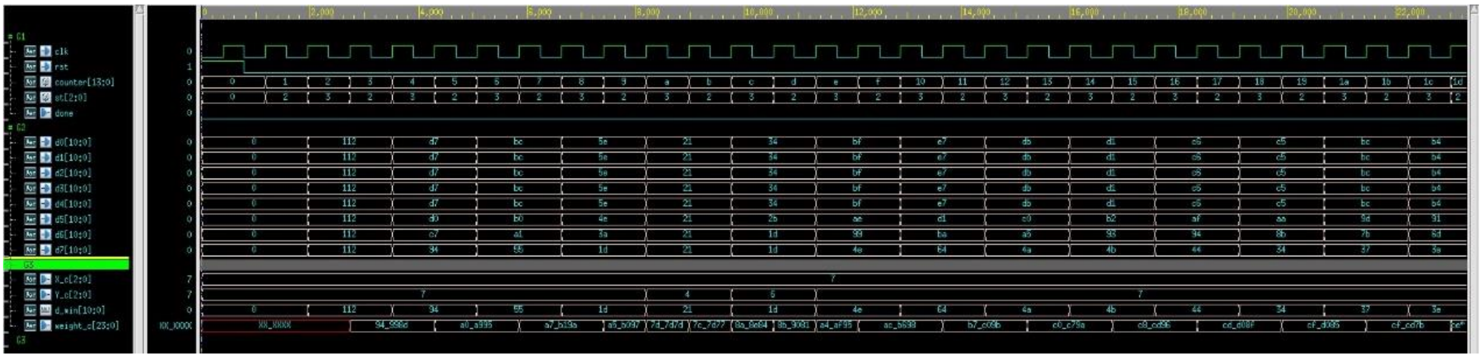
iii. **When X_c = 7, Y_c = 4**:

a. X_in[5] = 1110_1010_1010_1111
b. X_in[6] = 1110_0101_0110_1111
c. X_in[7] = 1110_0100_0110_1111
d. Again, for cases where the difference between X_c and X_in exceeds 3, the output sel produces 16 bits of all ones, which matches the intended design.

iv. **When X_c = 7, Y_c = 6**:

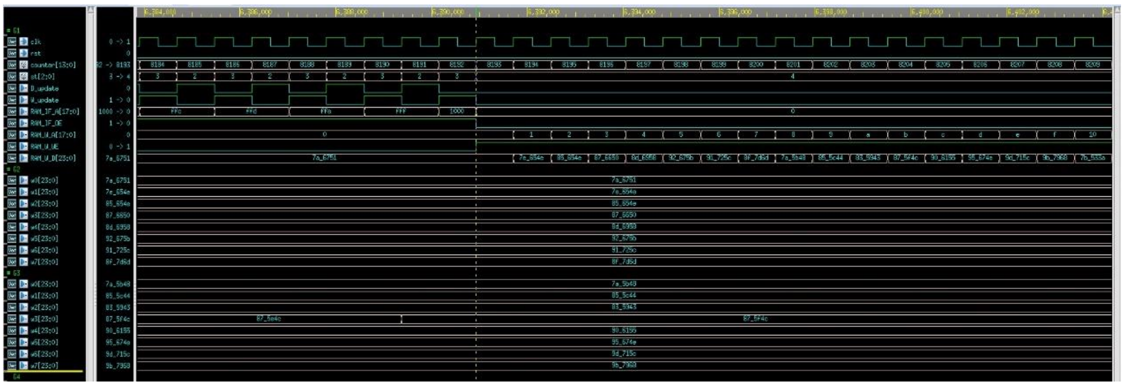
a. X_in[5] = 1010_1010_1111_1111
b. X_in[6] = 0101_0110_1111_1111
c. X_in[7] = 0100_0110_1111_1111
d. As before, all other inputs with X_c differing by more than 3 result in sel being 16 bits of ones, in line with the specified behavior.

C. MIN_2 Manhattan Distance Calculation



i. This waveform shows the operation of **MIN_2** in calculating the Manhattan distance. At the beginning, when all Manhattan distances are identical, the outputs X_c and Y_c are both 7. This is because, in the case of equal distances, the module outputs the value with the larger index. From the subsequent results, it can also be confirmed that **MIN_2** functions correctly by selecting the smallest Manhattan distance.

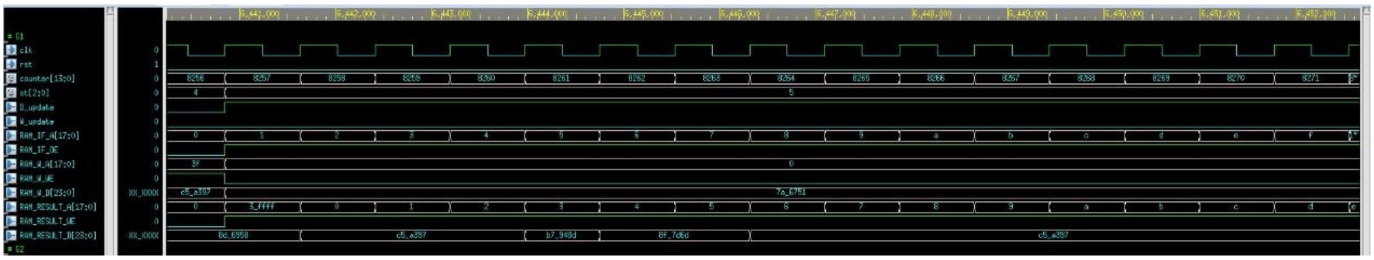
D. Controller State 4 (W_UPDATE) – Writing Codebook to RAM_W



i. The waveform above corresponds to **state 4 (W_UPDATE)**, where the trained weight values are written into RAM_W. At this stage, both D_update and W_update signals from the previous state are disabled, while RAM_W_WE is enabled. The address RAM_W_A ranges from

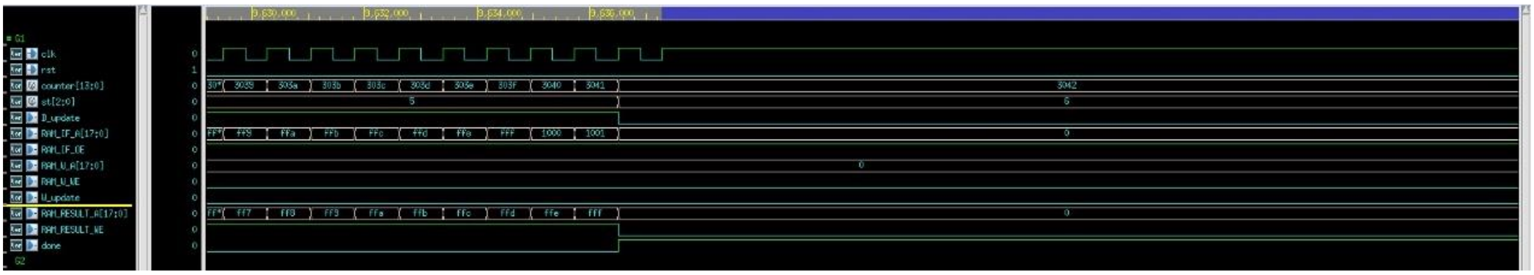
0 to 63 (calculated as counter – 8193). The w signals shown below correspond to weight0 through weight13, and the waveform confirms that these values are stored sequentially and correctly in RAM_W with the proper addresses.

E. Controller State 5 (W_PIC) – Writing Pixels to RAM_RESULT



i. This part illustrates the operation of writing pixels into RAM_RESULT and processing them through the VEP to calculate their Manhattan distance with respect to the codebook. In this state, RAM_W_WE from state 4 has already been disabled, while RAM_RESULT_WE is enabled. The address RAM_RESULT_A is calculated as counter – 8257, and RAM_RESULT_D represents the pixels written into RAM_RESULT.

F. Controller DONE State – Completion Signal



i. From this waveform, it can be seen that once all values have been written into RAM_RESULT, the state transitions to **DONE (state 5)**. At this point, the done signal is asserted (set to 1), while all other controller outputs are deactivated, indicating that the computation has been fully completed.

G. VEP Training Phase (Weight Book Update)

i. The waveform above illustrates the operation of the VEP during the first phase of training the weight book. It can be observed that D_update is first enabled for one clock cycle, allowing the VEP to calculate the Manhattan distance. The circuit then waits for the combinational logic to return the neighbor_sel value, after which W_update is enabled to update the weight book accordingly.

ii. In this process, DB, DG, and DR represent the three Manhattan distance components temporarily stored in the VEP. Storing these values avoids redundant recalculations, as they are reused during the weight book update. The update function for neighbor_sel can be expressed as:

$$N(t+1)=N(t)\pm\alpha h[\text{Manhattan distance}]$$

where the sign (±) is determined by comparing the pixel with the corresponding weight value: if the pixel is larger, the weight increases; otherwise, it decreases. The weight_update signal indicates the updated weights stored in the VEP. The design specifically uses the VEP for this stage because, when all initial values are identical, *MIN_2* prioritizes updating entries with larger indices.

H. USS Neighbor Selection Operation

i. This waveform, similar to Lab7_1, illustrates the operation of the **USS** module. From the signals, we can see the mapping of the sel bits to different values of Y (sel[15:14] corresponds to Y=7, sel[13:12] to Y=6, sel[11:10] to Y=5, sel[9:8] to Y=4, sel[7:6] to Y=3, sel[5:4] to Y=2, sel[3:2] to Y=1, and sel[1:0] to Y=0).

ii. **When X_c = 7, Y_c = 7**:

   a. X_in[5] = 1010_1011_1111_1111
   b. X_in[6] = 0101_1011_1111_1111
   c. X_in[7] = 0001_1011_1111_1111
   d. For all other cases where the difference between X_c and X_in exceeds 3, the output sel is 16 bits of all ones, consistent with the intended design.

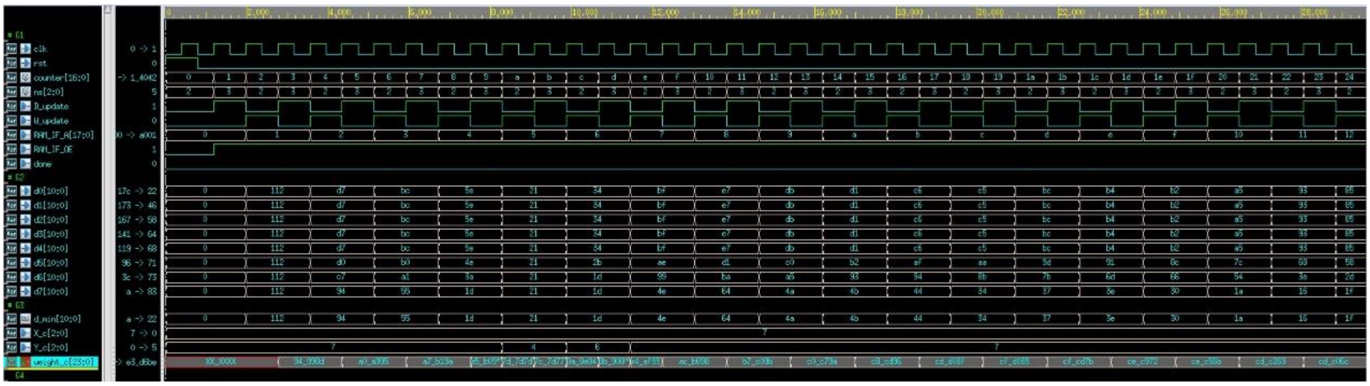
iii. **When X_c = 7, Y_c = 4**:

   a. X_in[5] = 1110_1010_1010_1111
   b. X_in[6] = 1110_0101_0110_1111
   c. X_in[7] = 1110_0100_0110_1111
   d. Similarly, when the distance between X_c and X_in exceeds 3, the sel output is 16 bits of ones, as expected.

iv. **When X_c = 7, Y_c = 6**:

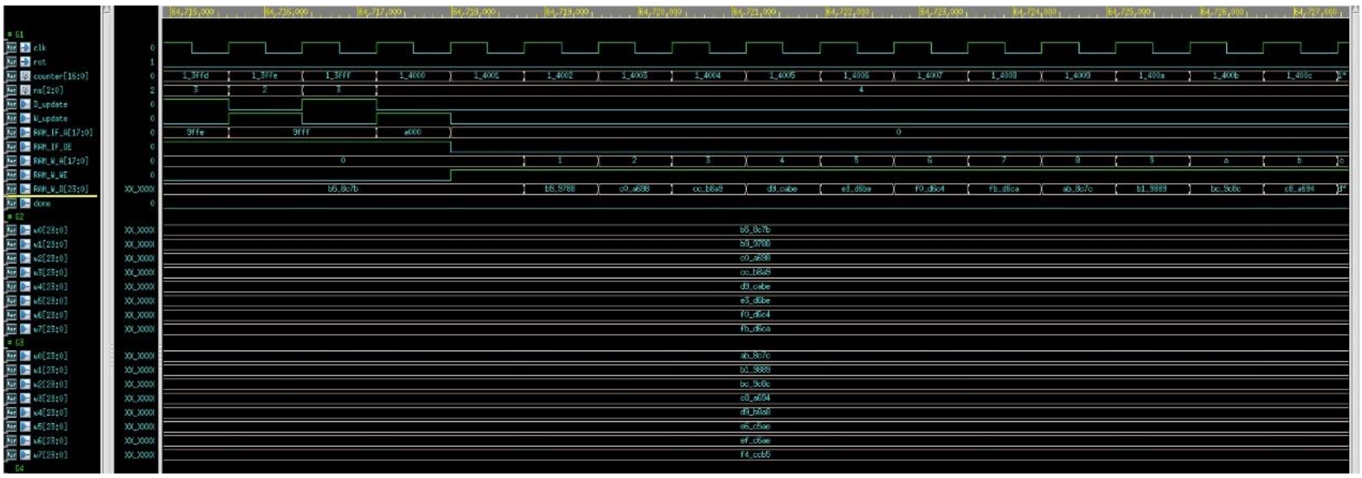
   a. X_in[5] = 1010_1010_1111_1111
   b. X_in[6] = 0101_0110_1111_1111
   c. X_in[7] = 0100_0110_1111_1111
   d. Once again, inputs with X_c differing by more than 3 produce an output sel of 16 bits of ones, consistent with the design specification.

I. MIN_2 Manhattan Distance Calculation



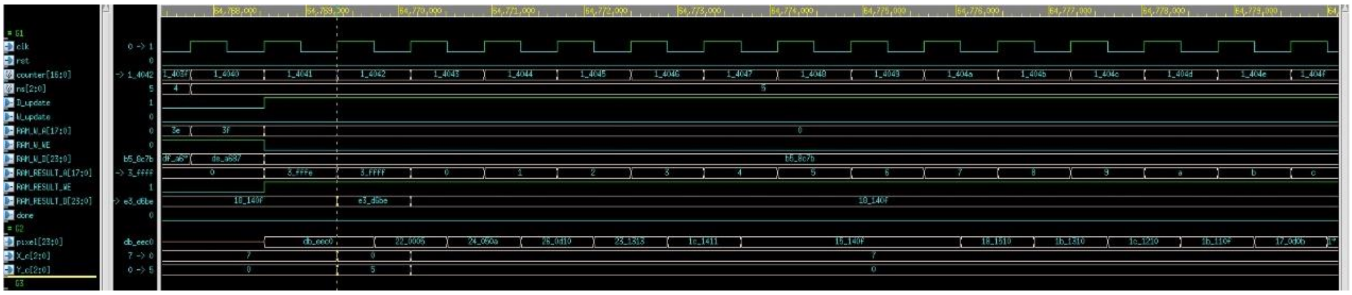
i. This waveform shows the operation of **MIN_2** in calculating the Manhattan distance. At the beginning, when all Manhattan distances are identical, the outputs X_c and Y_c are both 7. This occurs because, in the case of equal distances, the module outputs the index with the larger value. From the remaining results, it can be confirmed that **MIN_2** operates correctly by selecting the smallest Manhattan distance.

J. Controller State 4 (W_UPDATE) – Writing Codebook to RAM_W



i. The waveform above corresponds to **state 4 (W_UPDATE)**, where the trained weight values are written into RAM_W. At this point, both D_update and W_update from the previous state are disabled, while RAM_W_WE is enabled. The address RAM_W_A ranges from 0 to 63 (calculated as counter – 14001 in hexadecimal). The w signals, representing weight0 through weight13, are sequentially and correctly stored in RAM_W, with the addresses verified to be accurate.

K. Controller State 5 (W_PIC) – Writing Pixels to RAM_RESULT



i. This part shows the operation of writing pixels into RAM_RESULT and processing them through the VEP to calculate their Manhattan distance with respect to the codebook. At this stage, RAM_W_WE from state 4 has already been disabled, while RAM_RESULT_WE is enabled. The address RAM_RESULT_A is calculated as counter – 14041 (in hexadecimal), and RAM_RESULT_D represents the pixels being written into RAM_RESULT.

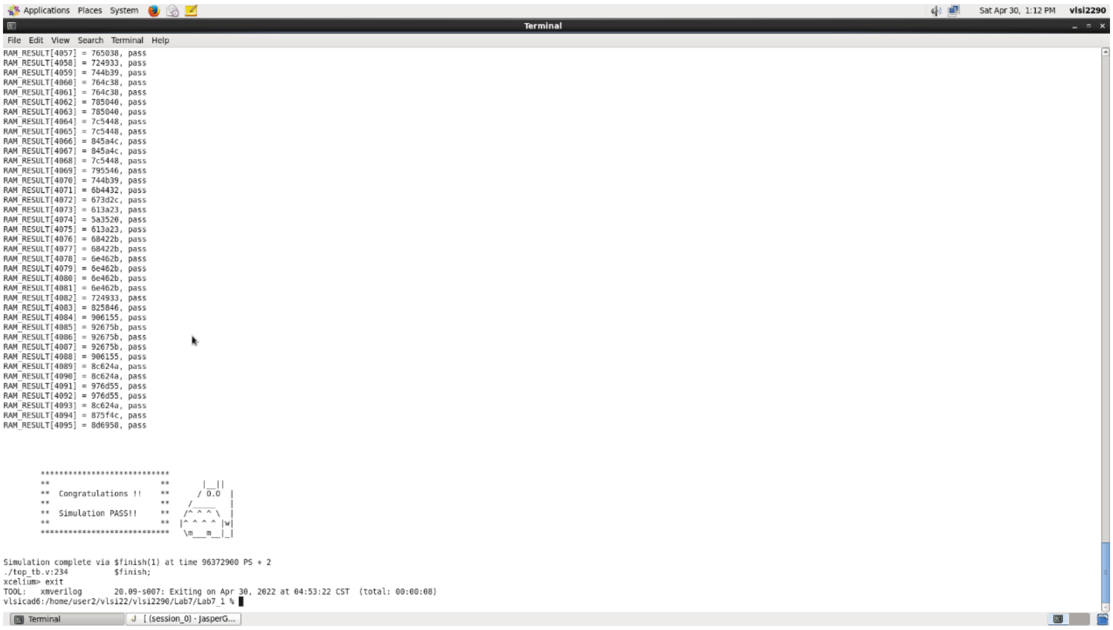
L. Controller DONE State – Completion Signal (Extended Cycles)

i. From this waveform, it can be observed that once all values have been written into RAM_RESULT, the state transitions to **DONE (state 5)**. At this point, the done signal is asserted (set to 1), while all other controller outputs are deactivated, indicating that the computation has been fully completed. It is worth noting that the counter value (in hexadecimal) is approximately ten times larger than in Lab7_1, reflecting the significantly increased cycle count.

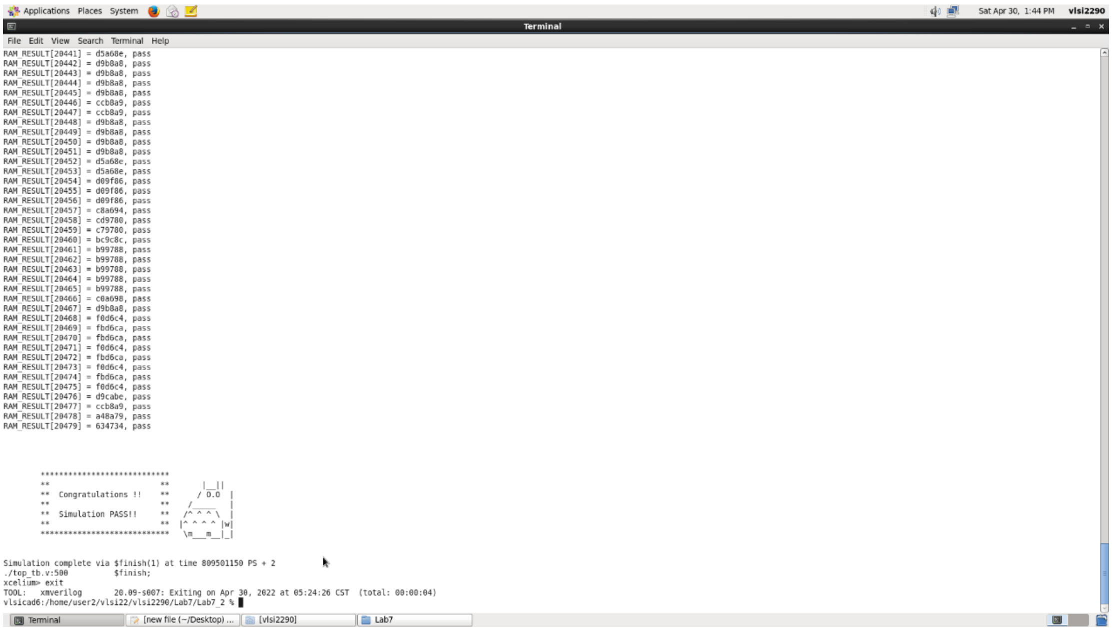
7. Simulation result

A. Presim

i. Single picture

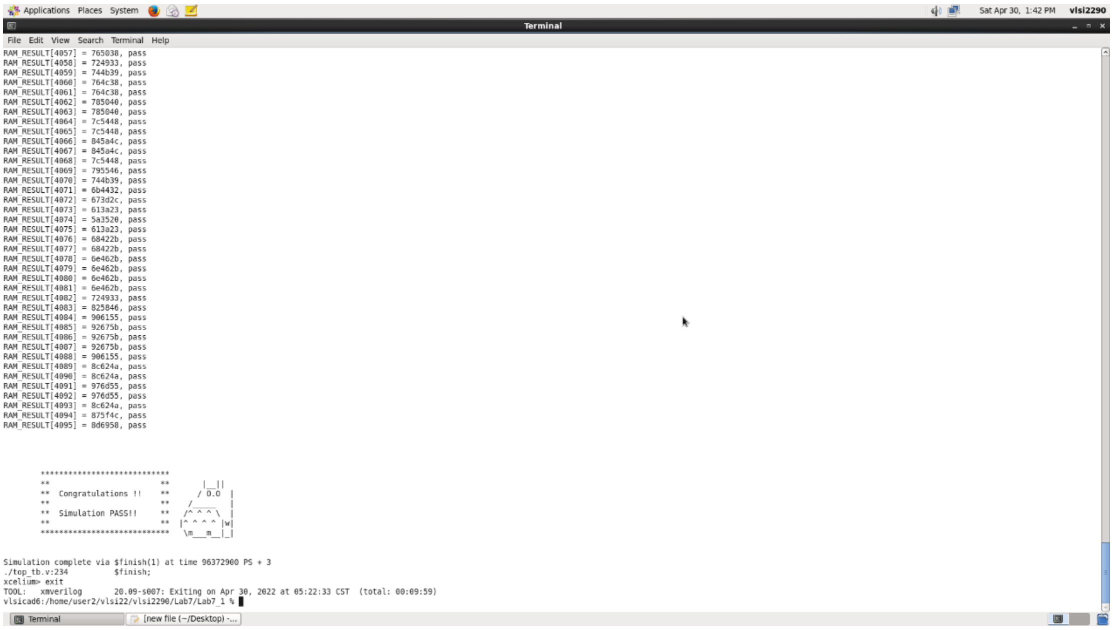


ii. Multiple picture (10)

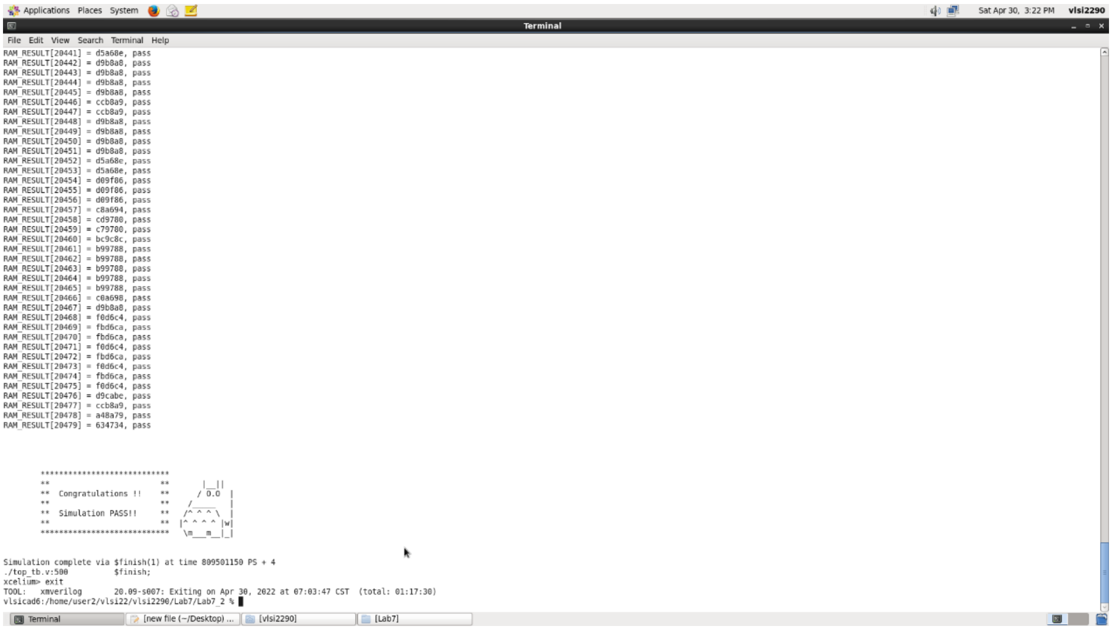


B. Postsim

i. Single picture

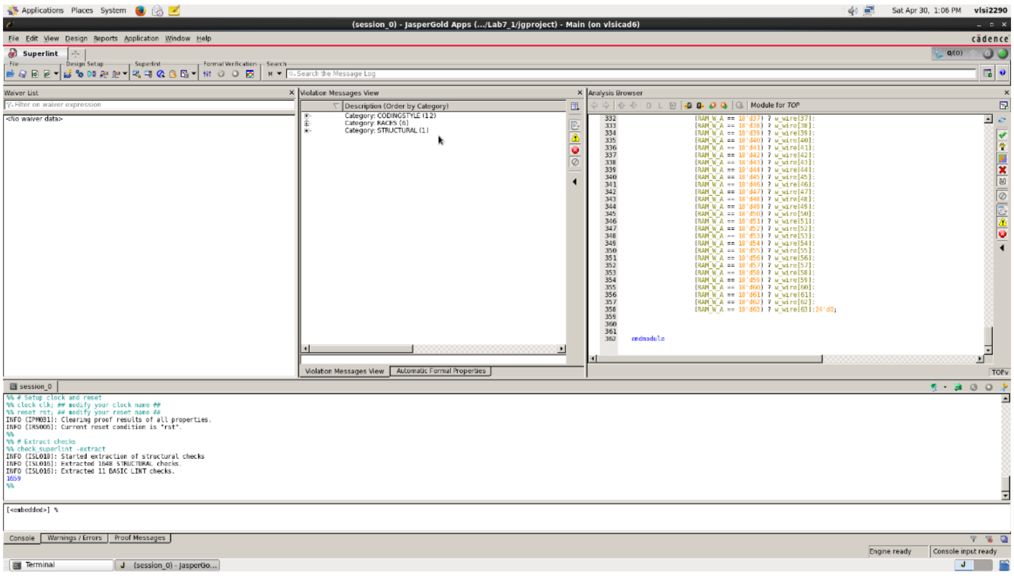


ii. Multiple picture (10)

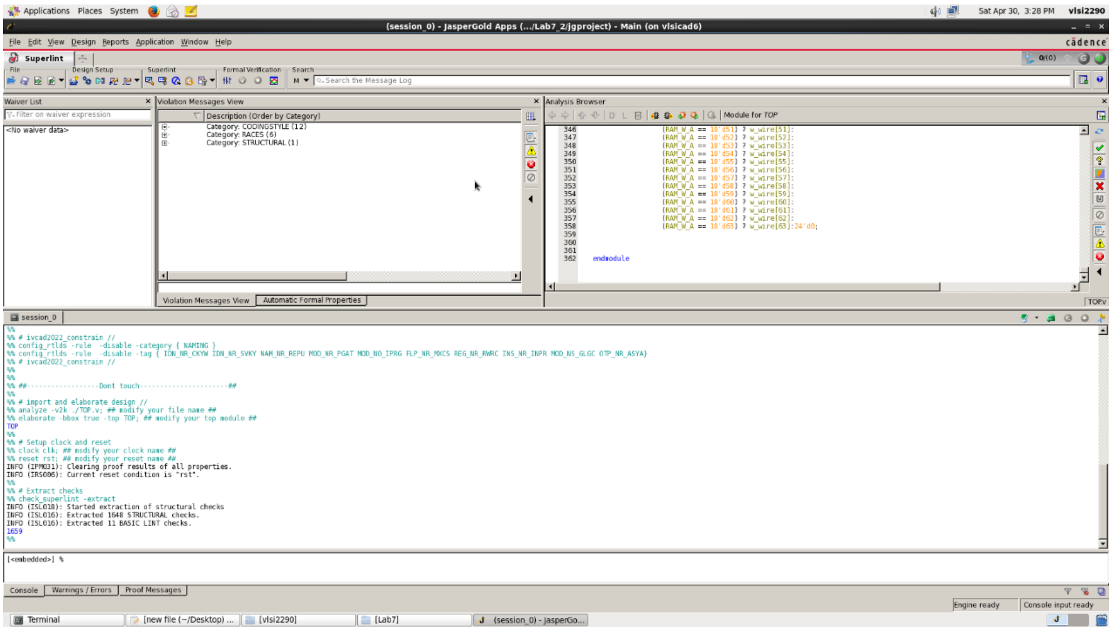


8. SuperLint coverage

A. Single picture SuperLint coverage = (1-19/342)*100% = 94.75%



B. Multiple picture SuperLint coverage = (1-19/342)*100% = 94.75%



9. Clock period, total cell area, post simulation time

A. Single picture

i. Clock period: 7.8 ns

ii. Total cell area: 22619.668687

iii. Post simulation time: 96372900ps

B. Multiple picture

i. Clock period: 7.9ns

ii. Total cell area: 22644.79897

iii. Post simulation time:809501150ps

10. **Design Optimization during Synthesis**

A. In the first phase, all modules except the VEP were implemented as purely combinational logic, which resulted in a slightly longer datapath. After experimentation, I found that keeping the period at **two cycles** was the best choice, as it balanced performance and simplified the controller design. Using only one cycle would cause the update signal to remain constantly enabled, making the system harder to control, so this approach was not adopted.

B. To reduce area, I reused the Manhattan distance calculation within the VEP instead of recalculating it in the neighbor function. By storing the results in registers, I was able to reuse them later, thereby avoiding redundant computations. Additionally, I simplified the number of cases in the USS module, which further reduced overall cell area.

C. In **Second phase**, nearly all files remained the same as in first phase, with the only major difference being modifications to the controller's state transitions. As a result, the implementation process for **Second phase** was much faster than for first phase. However, I observed that the number of cycles had a significant impact on simulation time. Since ten images had to be trained at once, the simulation time increased drastically-especially for post-synthesis simulation, where the process took almost two hours before producing waveform outputs.

D. From the experimental results, I also found that when the codebook was trained using only a single image, it contained a mix of various colors. In contrast, after training with ten images, the codebook was dominated by skin-tone colors. Although this caused greater color distortion for individual images, it allowed more effective compression across multiple images.

E. Overall, I appreciated the thoughtful design of this project, which not only helped us deepen our understanding of system behavior but also challenged us to address practical issues in simulation and synthesis.