**Project: Optimizing a CNN Accelerator for Image Super Resolution**

**Issued:** June 8 (Thursday), 2023 **Due: June 22nd (Thursday), 2023**

**What to turn in**: **Copy the text from your MODIFIED codes and paste it into a document**. If a question asks you to plot or display something on the screen, also include the plot and screen output your code generates. Submit either a \*.doc or \*.pdf file.

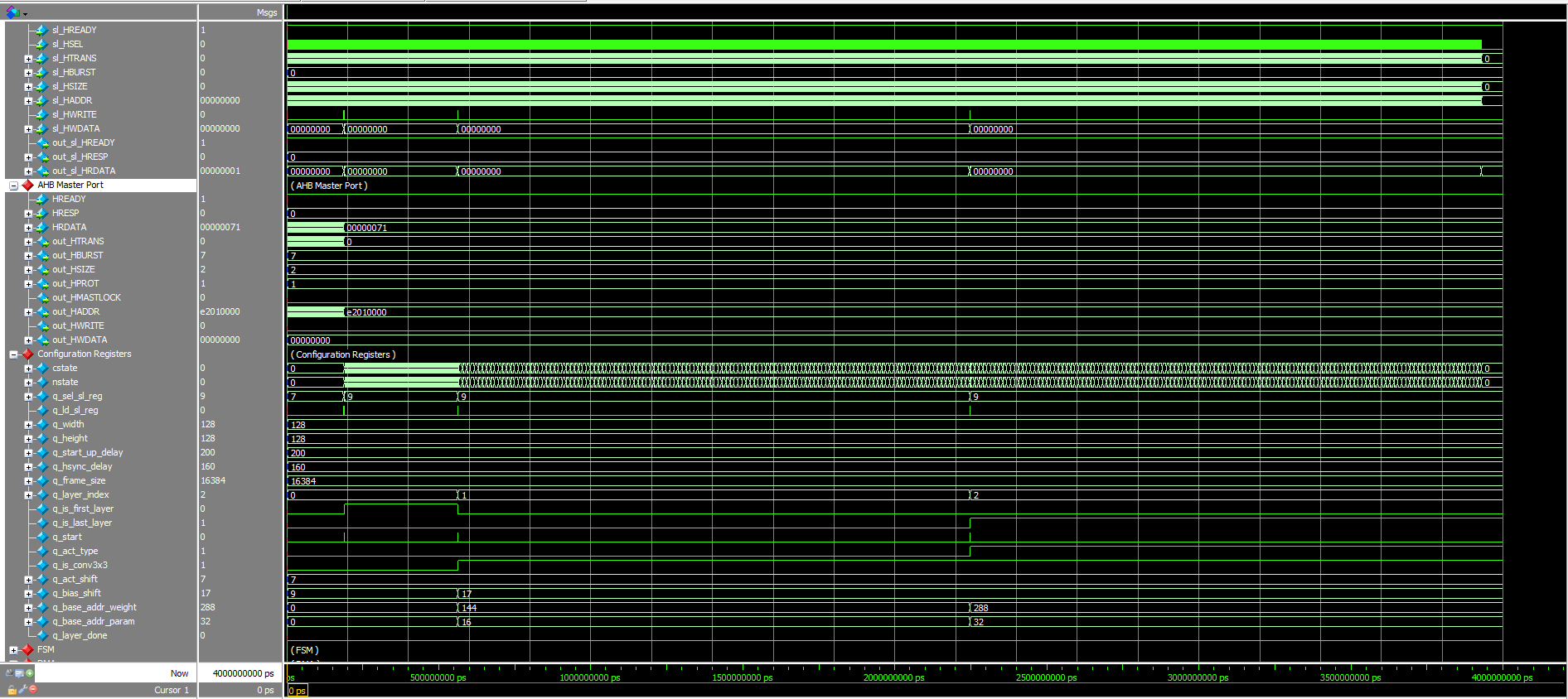
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Problem 1 (100p): CNN Accelerator for SR**

1. **Baseline code (10p)**

What you have to do:

1. Do a simulation with time = 4ms and capture the waveform.

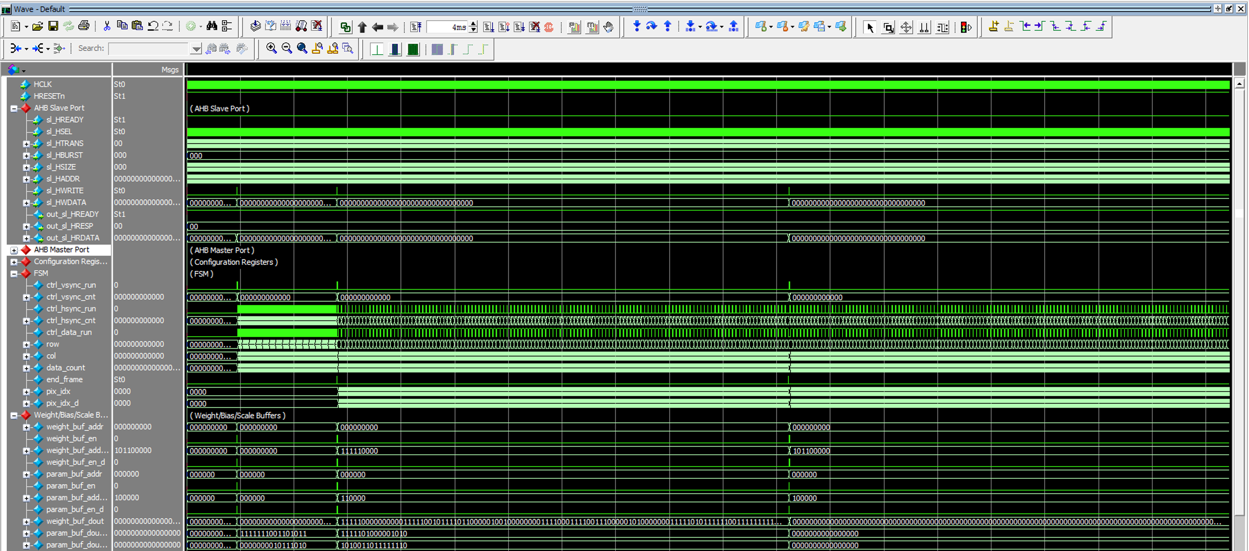
****

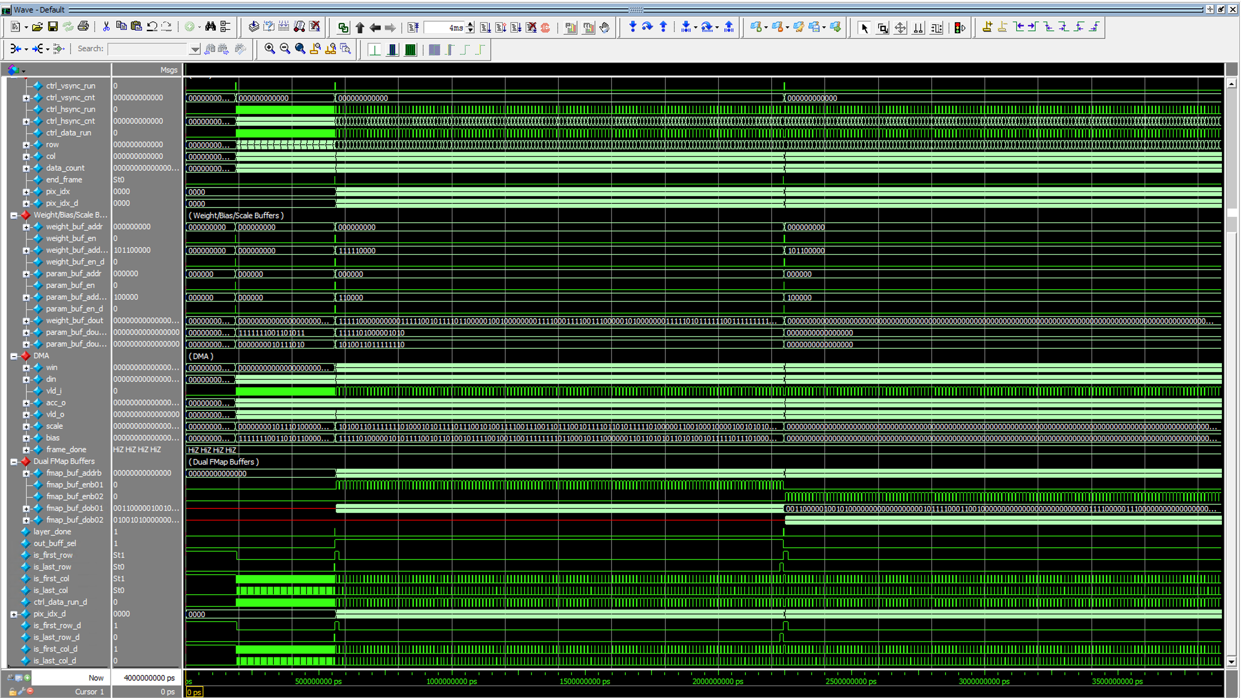
****

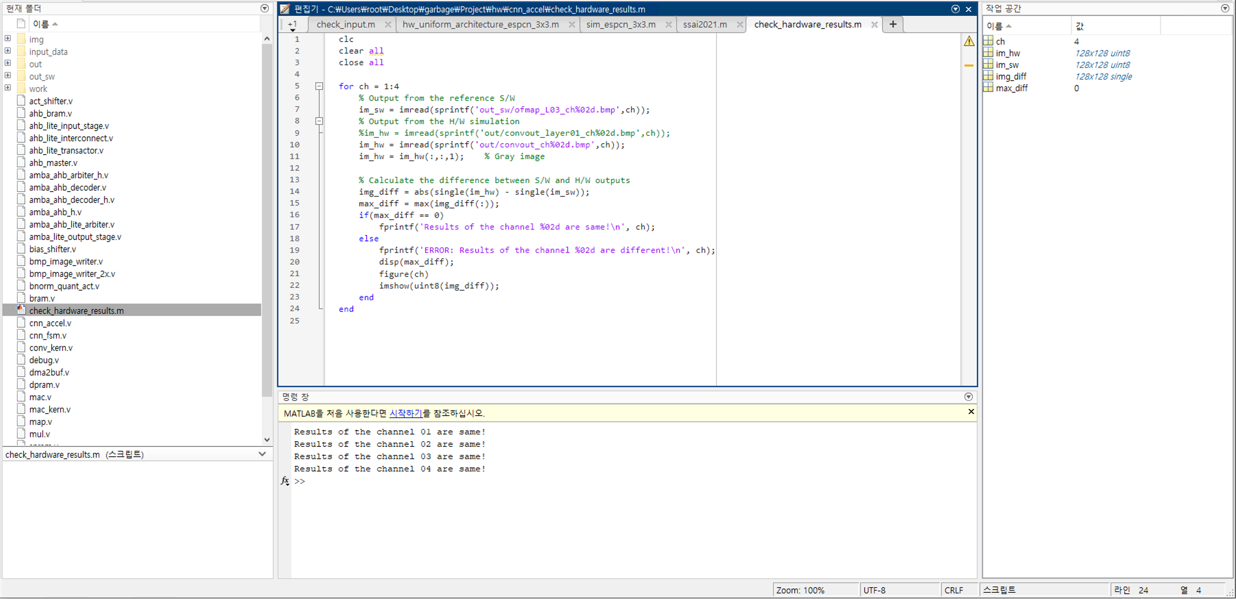
Figure 1-1: A captured waveform of cnn\_accel.

1. Use check\_hardware\_results.m to verify the output images generated by the H/W simulation.

**Solution :**

****

****

****

1. **Reference S/W (40p)**

Modify the codes to run a deeper network for SR named SSAI2021 which has 8 convolutional layers in Table I.

Table I: SSAI2021 Network Architecture for Image Super-Resolution

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer | Filter size | Input channels | Output channels | Input | Output |
| 1 | 3×3 | 1 | 16 | 128×128×1 | 128×128×16 |
| 2 | 1×1 | 16 | 16 | 128×128×16 | 128×128×16 |
| 3 | 3×3 | 16 | 16 | 128×128×16 | 128×128×16 |
| 4 | 3×3 | 16 | 16 | 128×128×16 | 128×128×16 |
| 5 | 3×3 | 16 | 16 | 128×128×16 | 128×128×16 |
| 6 | 3×3 | 16 | 16 | 128×128×16 | 128×128×16 |
| 7 | 1×1 | 16 | 16 | 128×128×16 | 128×128×16 |
| 8 | 3×3 | 16 | 4 | 128×128×16 | 128×128×4 |

What you have to do:

1. **Configuration (20p)**

The configuration parameters of all eight layers are defined in Table II. Run the reference S/W code (hw\_uniform\_architecture\_ssai2021.m) and complete Table II.

Table II: SSAI2021’s configuration parameters

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Layer | | | | | | | |
| #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 |
| q\_is\_first\_layer | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| q\_is\_last\_layer | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| q\_is\_conv3x3 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| bias\_shift | 16 | 22 | 23 | 23 | 23 | 23 | 23 | 24 |
| act\_shift | 00 | 01 | 01 | 01 | 01 | 01 | 00 | 00 |

1. **Data preparation (20p)**

* Run write\_cnn\_model\_to\_hex\_file.m to generate all hexadecimal files, including input, weights, scales, biases, and outputs at the folder /output\_hex\_file/ssai2021. Note that weights, biases, scales, and outputs of a layer can be stored separately for verification.

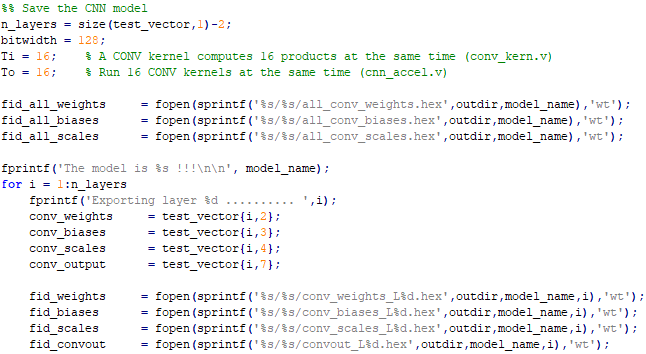


Figure 1-2: Matlab code used to define the file identifiers for writing.

* Based on the hexadecimal files, determine the buffer sizes required for weights, biases, and scales and the buffer sizes by completing Table III.

Table III: The buffer requirements

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | No. of bit  per line | Number of lines | | | | | | | | | Buffer Size in total | |
| #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 | Word  (bit) | | No. of  words |
| Weight | 128 | 16 | 16 | 144 | 144 | 144 | 144 | 16 | 144 | 128 | | 768 |
| Scale | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | | 128 |
| Bias | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | | 128 |

* (5p) Are those values in Table III optimal? Explain your answer.

1. **(20p) Sub-pixel layer (See Lecture 14 for the detailed description)**

* Implement the code in accel\_cnn.v to handle **the sub-pixel layer**. Basically, four pixels are generated at the sub-pixel layer. Each of the four pixels must be added with the corresponding input.
* Make a new BMP writer that captures a 32-bit input and writes the high-resolution image.

1. **(30p) H/W simulation and verification for SSAI2021**

What you have to do:

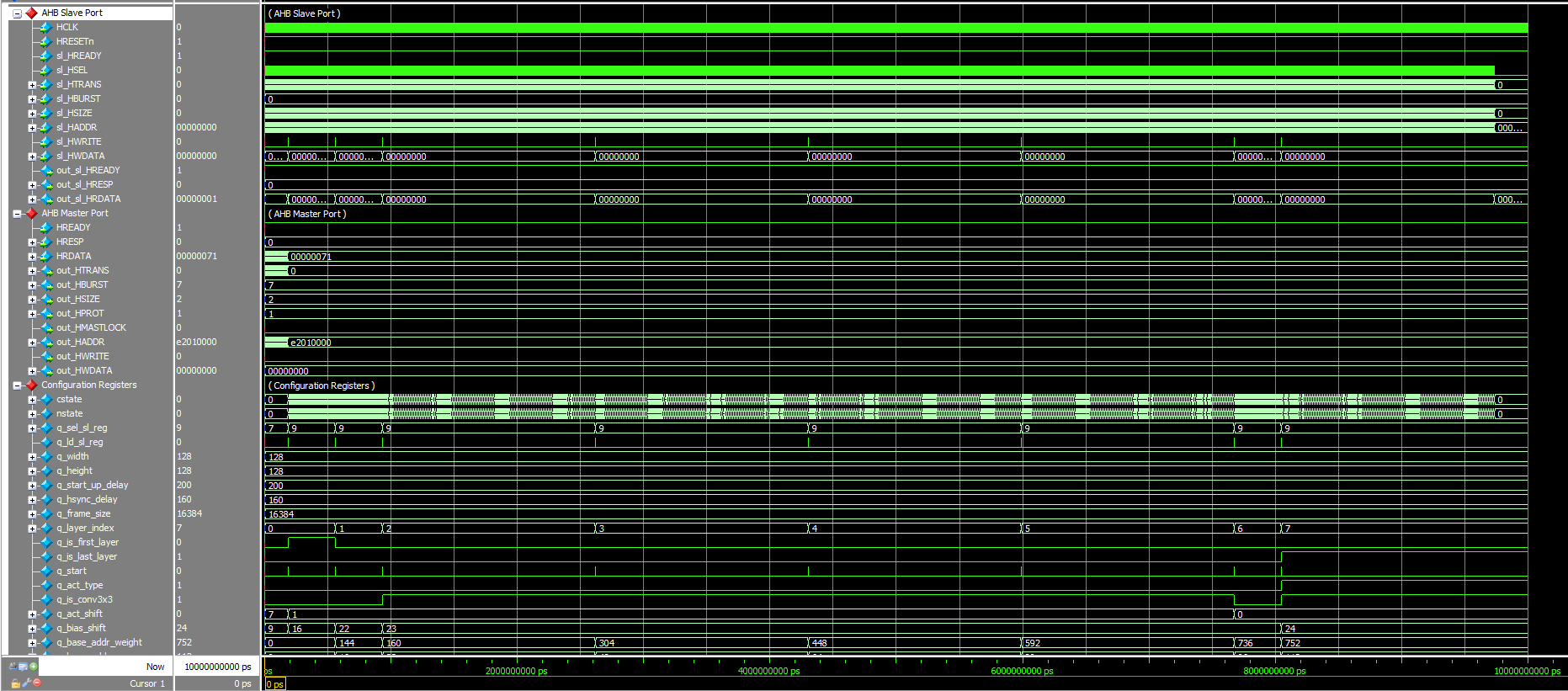
1. Copy the weight/scale/bias hex files of SSAI2021 from Part 2b to input\_data/
2. Update the **buffer** parameters for SSAI2021 in the top file (cnn\_accel.v) and the test bench (top\_system\_tb.v).

Hint: Only changing the number of layers may work.

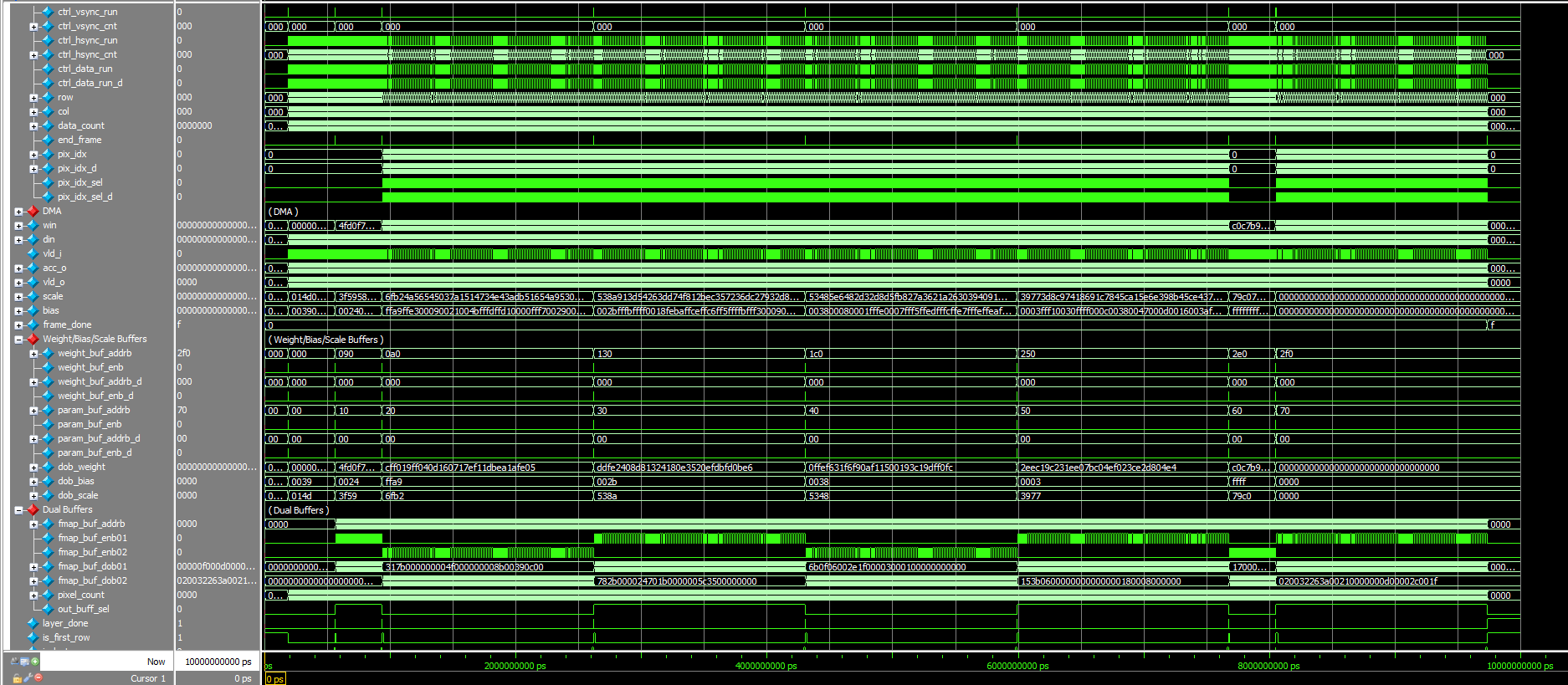
1. Modify the test bench (top\_system\_tb.v) to execute SSAI2021 on the CNN accelerator

Hint: Use the parameters in Part 2a.

1. Do a simulation with time = 10ms and capture the waveform.



(a)



(b)

Figure 1-3: Figure 1-1: A captured waveform of cnn\_accel.

Hints:

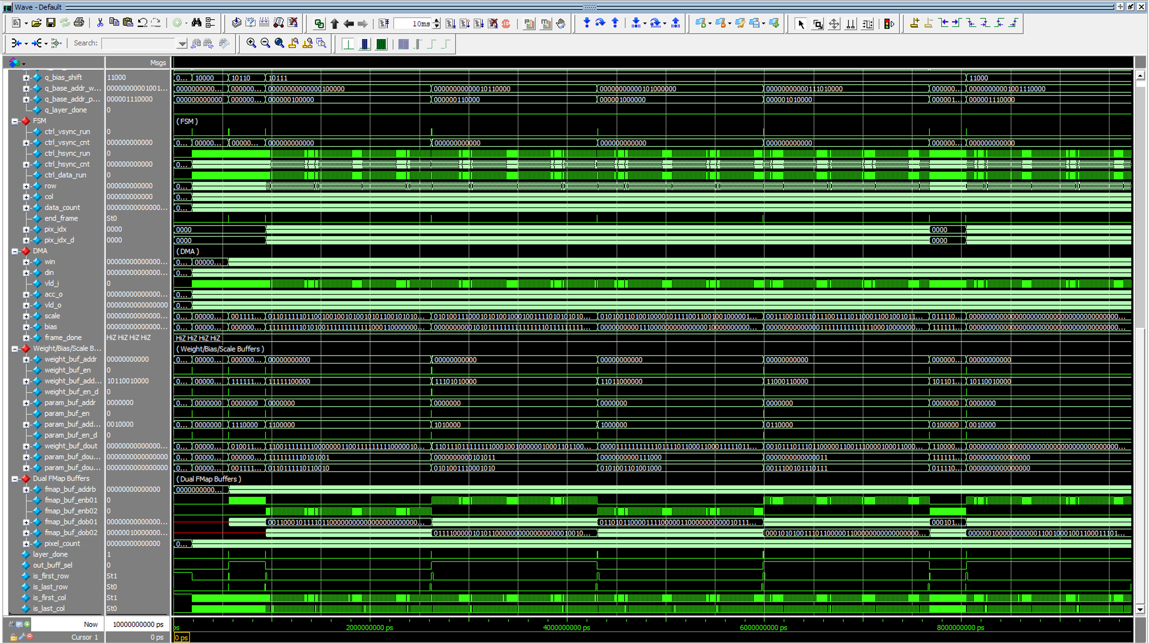
* **You should check the configuration registers carefully**.
* To speed up the simulation time, the following code in the top module (cnn\_accel.v) is **commented out,** as shown in Fig. 1-4. During debugging, you may **uncomment** them to verify the outputs of some first CONV layers early.

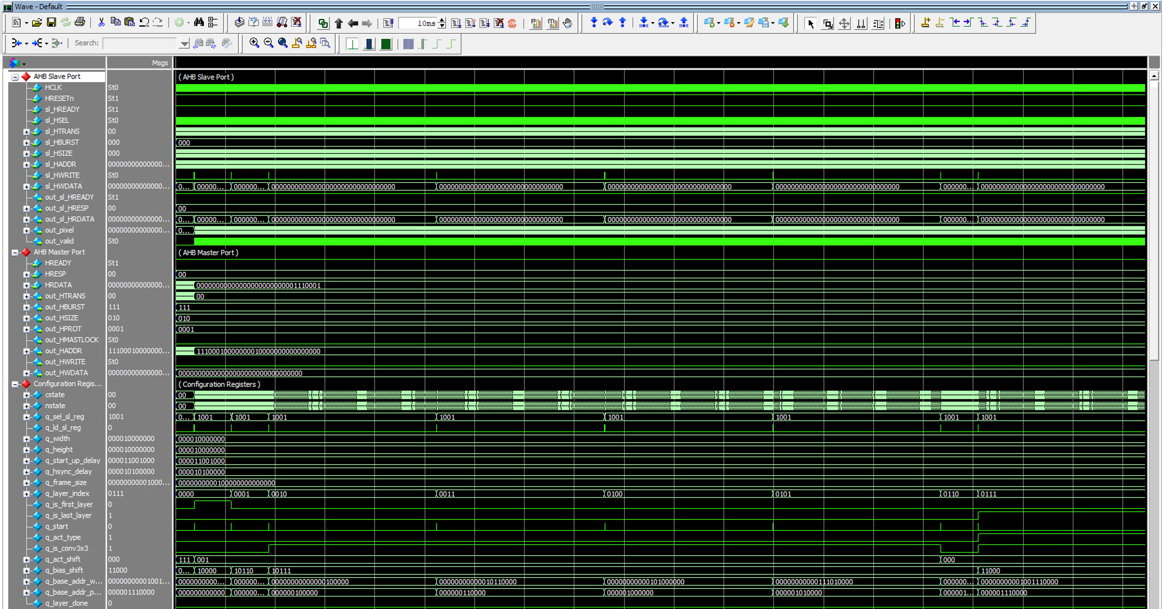
1. Use check\_hardware\_results.m to verify the output images generated by the H/W simulation.

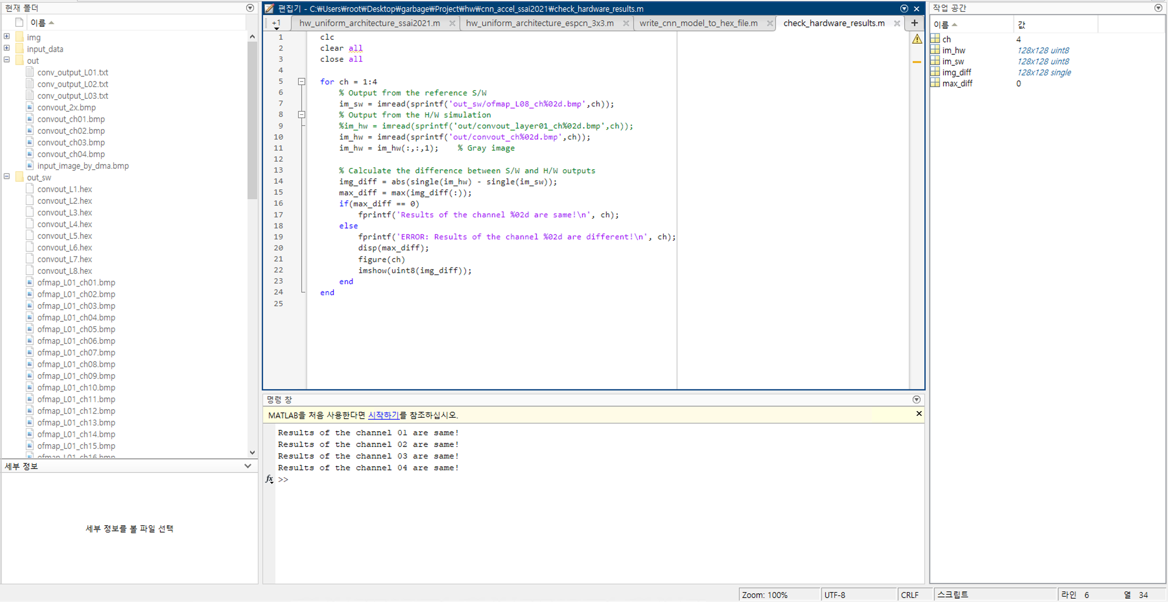


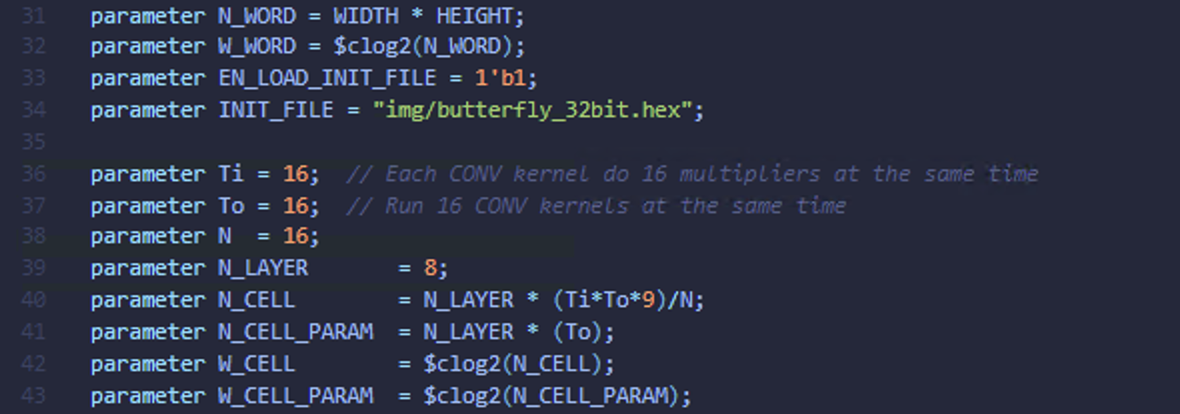
Figure 1-4: Disable the file logging to speed up the simulation time.

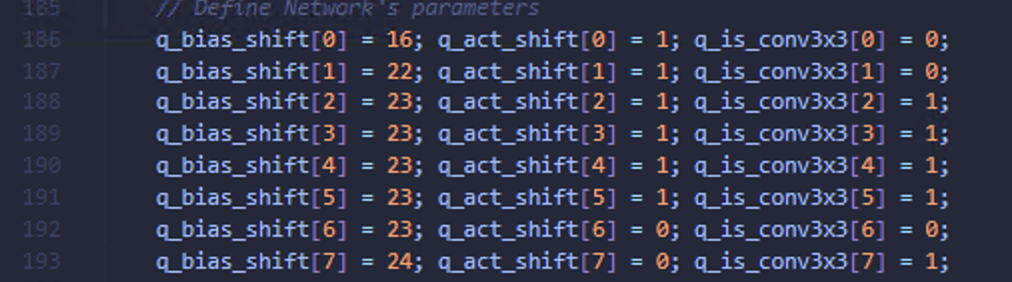
**Solution :**

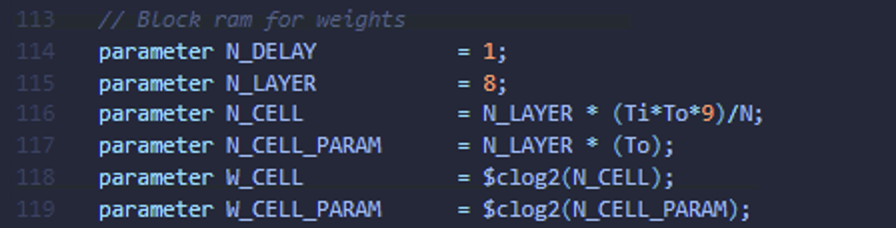
****

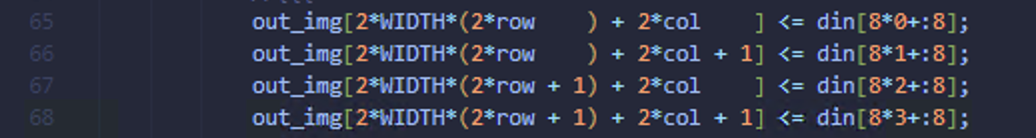
****

****

****

****

****

****

**Problem 2 (100p): Optimization**

1. **Optimization (90p)**

Improve the CNN accelerator design for time and buffer reduction. Check Lecture 14b for details.

1. **Problem and scopes**

The goal is to improve the CNN accelerator by reducing or minimizing the number of cycles and the buffer size.

Scopes and constraints:

* The baseline code is cnn\_accel\_opt/, which executes **the three-layer CNN** as in the class.
* Ti and To are fixed to 16. Do NOT increase the number of convolution kernels or the number of multipliers in a kernel.
* Weights, scales, and biases are quantized to 8-bit, 16-bit, and 16-bit numbers, respectively.
* The clock frequency is fixed to 100MHz. Do NOT increase the clock frequency to speed up the system.
* WIDTH, HEIGHT, and FRAME\_SIZE are fixed.
* The running time and the buffer size of the baseline are **t0=3,930** us and **s0=4,280** Kbits.

What you can modify:

* Hex files: You can reorganize the input file (img/ butterfly\_32bit.hex) or the weight/bias/scale files.
* The CNN accelerator top module (cnn\_accel.v, cnn\_fsm.v).
* The test bench (top\_system\_tb.v).
* The image writer (bmp\_image\_writer.v) may be modified if you define a new output order.

1. **Optimization methods**

As described in the class, we can reduce execution and buffer size by:

* Reordering the input data and the number of transactions on DMA.
* Pipelining the DMA and the convolution computation to reduce the input buffer.
* Fully utilizing the convolution kernels when executing Layer 3.
* Applying layer fusion

You should describe your modified code in the report.

1. **Evaluation**

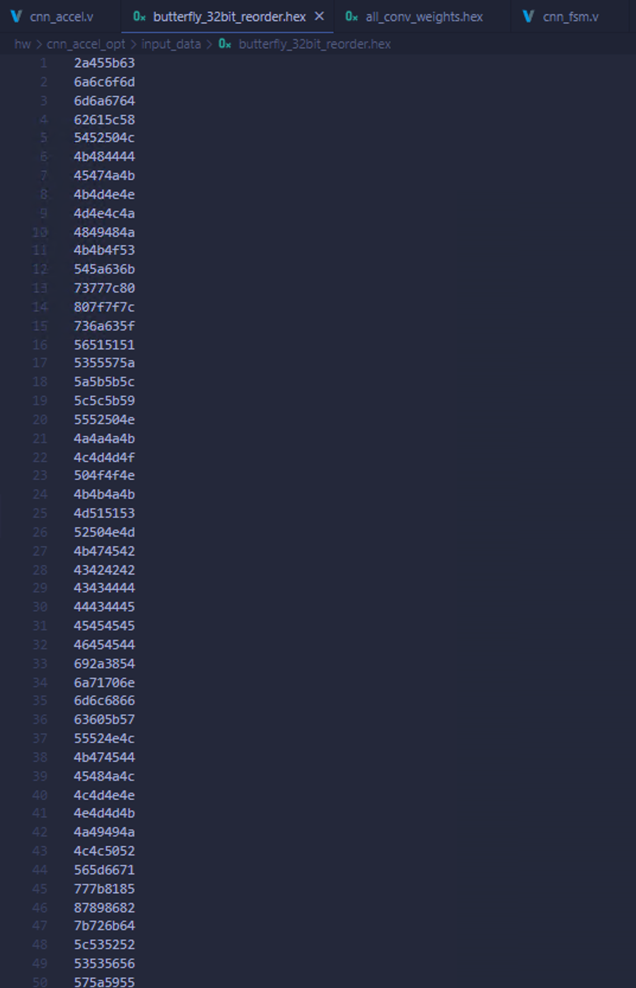
* Use check\_hardware\_results.m to verify the output images generated by H/W simulation. Please make sure that your optimized code functions correctly as the baseline.
* Report the execution time (t) and the buffer size (s) of your design. The overall improvement is measured by the following metric:

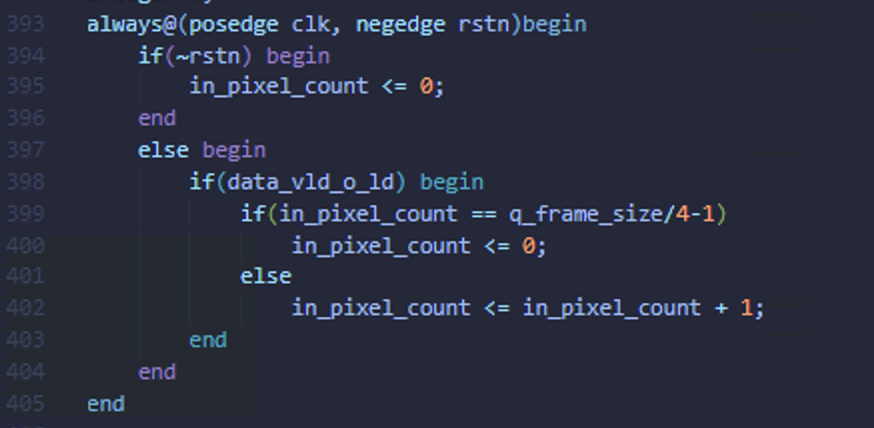
Where t0 and s0 are the execution time and the buffer size of the baseline version, respectively.

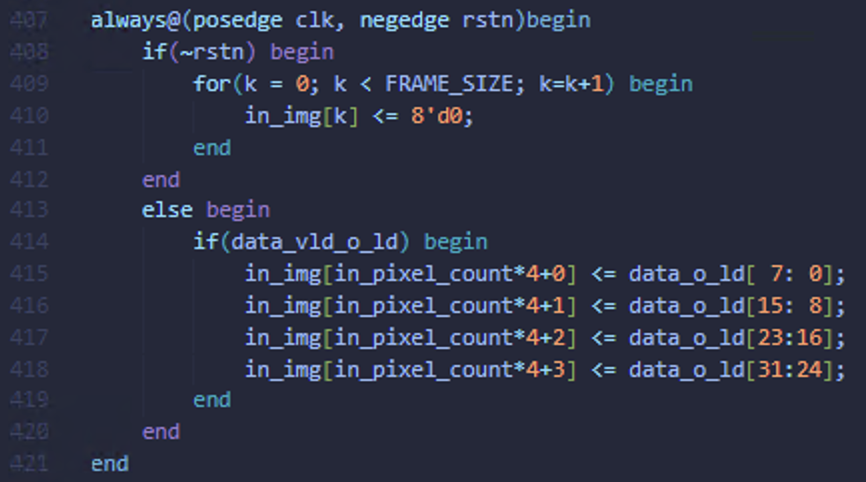
1. **Optimality analysis (10p)**

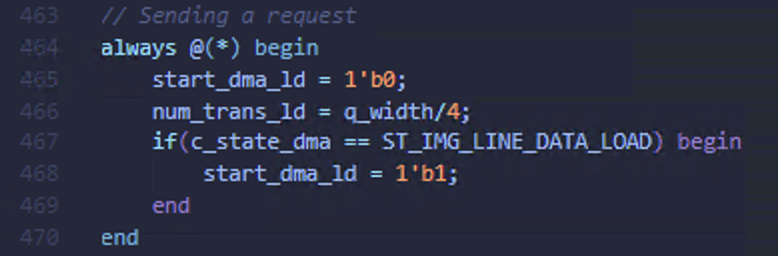
Explain why you choose parameters for your approach. For example, to reduce the input buffer size, you only preload a few image lines, i.e., *n*, from Memory and then pipeline the DMA and the convolution computation. Then, you should explain the choice of *n*.

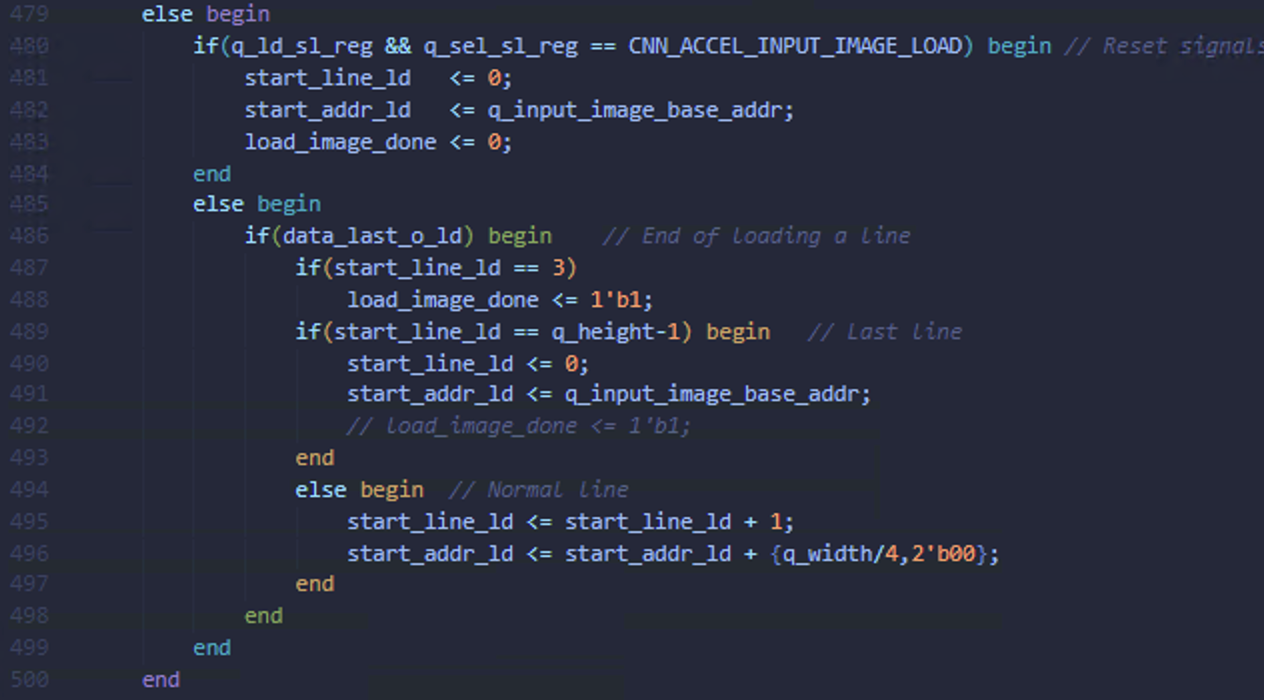
**Solution :**

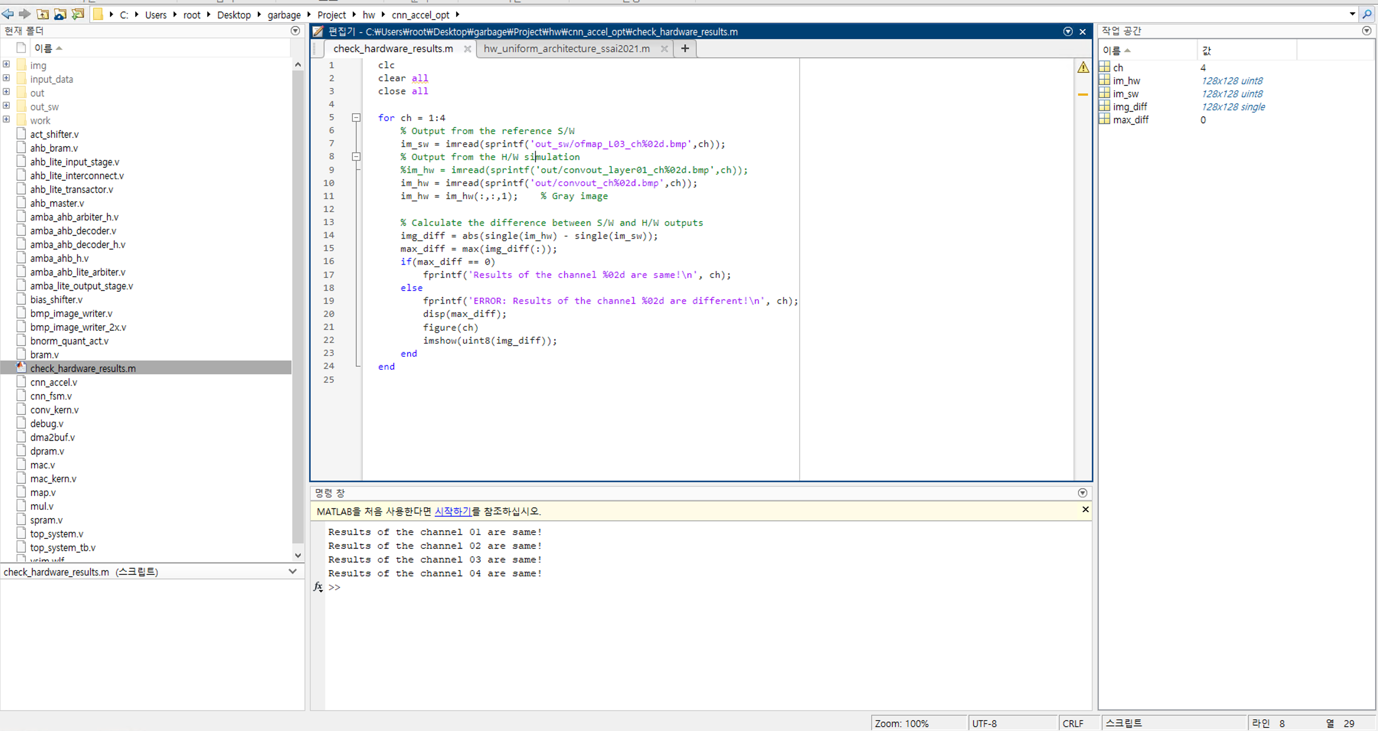
****

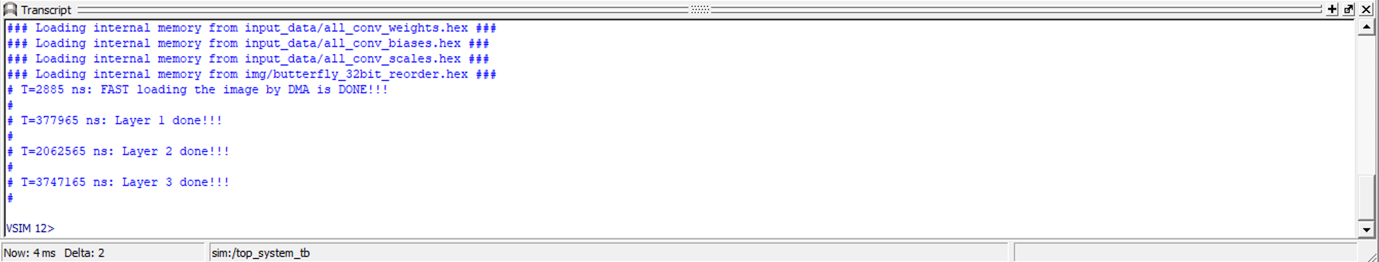
****

****

****

****

****

****

**Optimization : Input Image Re-ordering & DMA Pipelining**

1. **Reorder input image and speed up the DMA x4**
2. **DMA Pipelining with the first layer to start early**

**Evaluation : Total time t -> 3740 us / Total Memory s -> 4280Kbits / S\_overall = 1.05**

**Analysis :**

1. **n for DMA loading ( Re-ordering ) => 4 / Reason : each word can store at most 4 pixels at the same time. Use bandwidth at most we can use**
2. **n for DMA pipelining => 3 / Reason : only need 3 lines of image to start the layer 1**