**Homework 12: CNN Accelerator and DMA**

**Issued:** May 30 (Tuesday), 2023 **Due:** June 5 (Monday), 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 (30p): CNN Accelerator**

Implement a CNN accelerator in Verilog. Please see the description in the lecture note for details.

Note that the same baseline code is used for **Parts 1 and 2**.

1. **Weight/bias/scale buffers (10p)**

What you have to do:

* 1. Complete the missing codes to access weight/bias/scale buffers in cnn\_accel.v.
  2. Do simulation with time = 4,000us and capture the waveform.

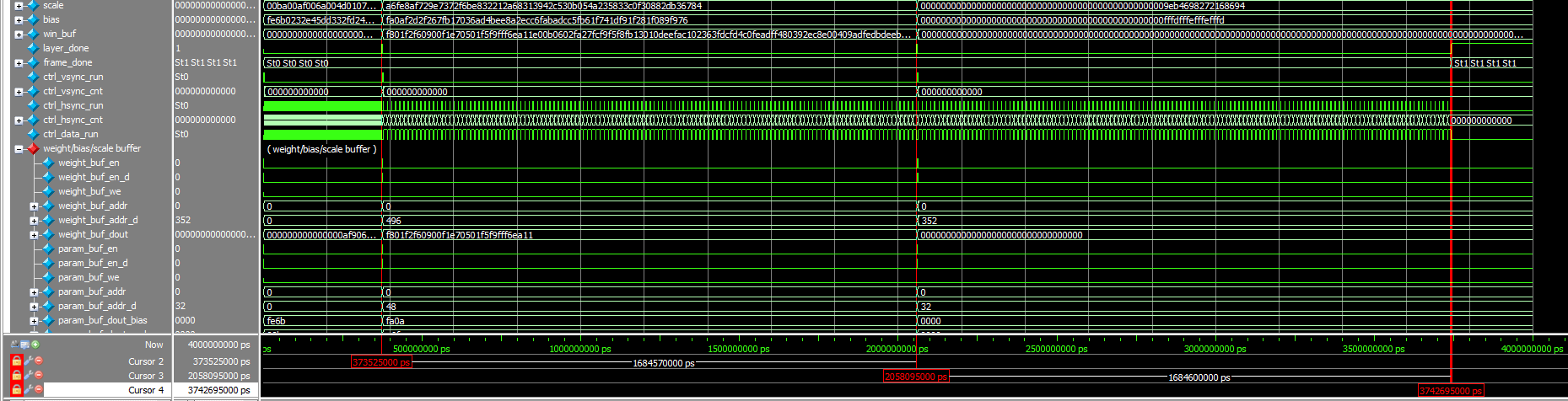


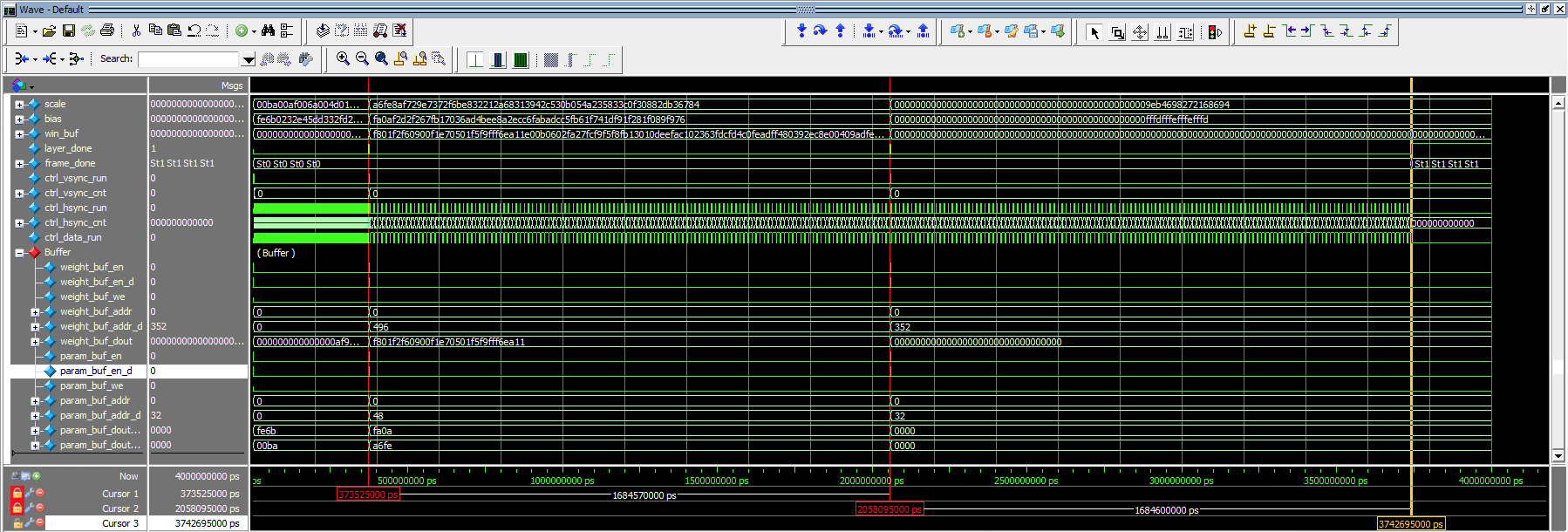
Fig. 1-1: A captured waveform of cnn\_accel.

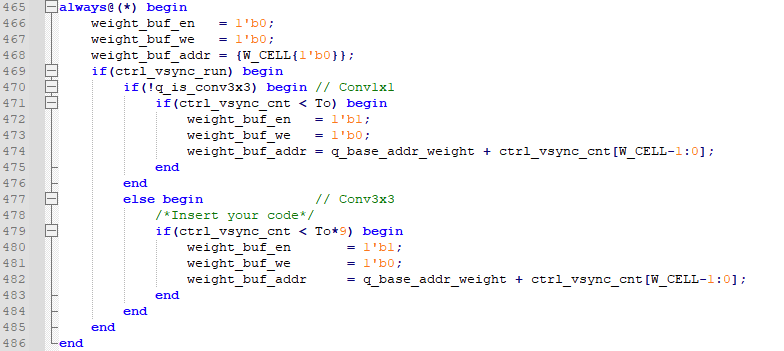
* 1. (3p) Determine the buffer sizes and the minimum number of cycles required for weight/bias/scale preloading by completing the following table:

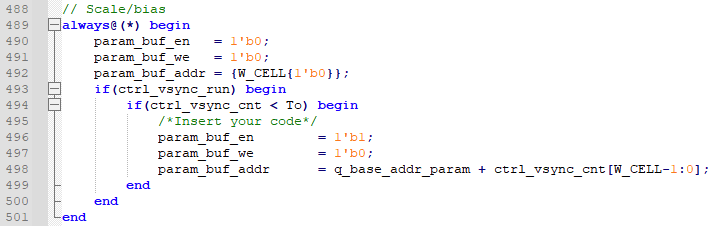
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Buffer | The number of cycles | | | The size of buffers | | | |
| Layer 1 | Layer 2 | Layer 3 | Word (bit) | No. of words | Size (bit) |
| Weight | 9 | 144 | 144 | 256 | 432 | 110592 |
| Bias | 16 | 16 | 4 | 16 | 48 | 768 |
| Scale | 16 | 16 | 4 | 16 | 48 | 768 |

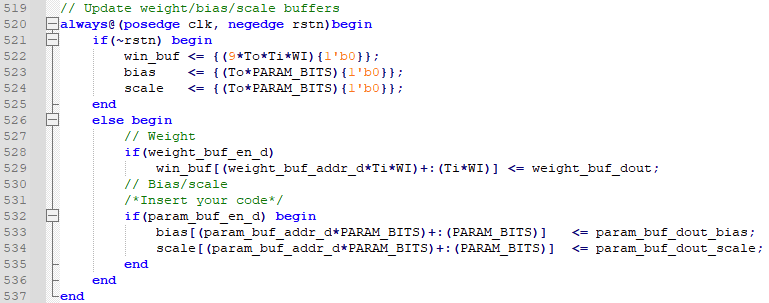
1. (2p) Explain how the base addresses q\_base\_addr\_weight and q\_base\_addr\_param are used.

**Solution 1-1 : Weight/bias/scale Buffers**

****

****

****

****

**(d) : Explain how the base addresses q\_base\_addr\_weight and q\_base\_addr\_param are used.**

**q\_base\_addr\_weight and q\_base\_addr\_param are used for calculating the offset to determine where the output of the buffer should locate since the output from the single ram needs an extra clock to give an output.**

1. **Dual buffers for feature maps (20p)**

What you have to do:

1. Complete the missing codes to access the dual buffers in cnn\_accel.v.
2. Do a simulation with time = 4,000us and capture the waveform.

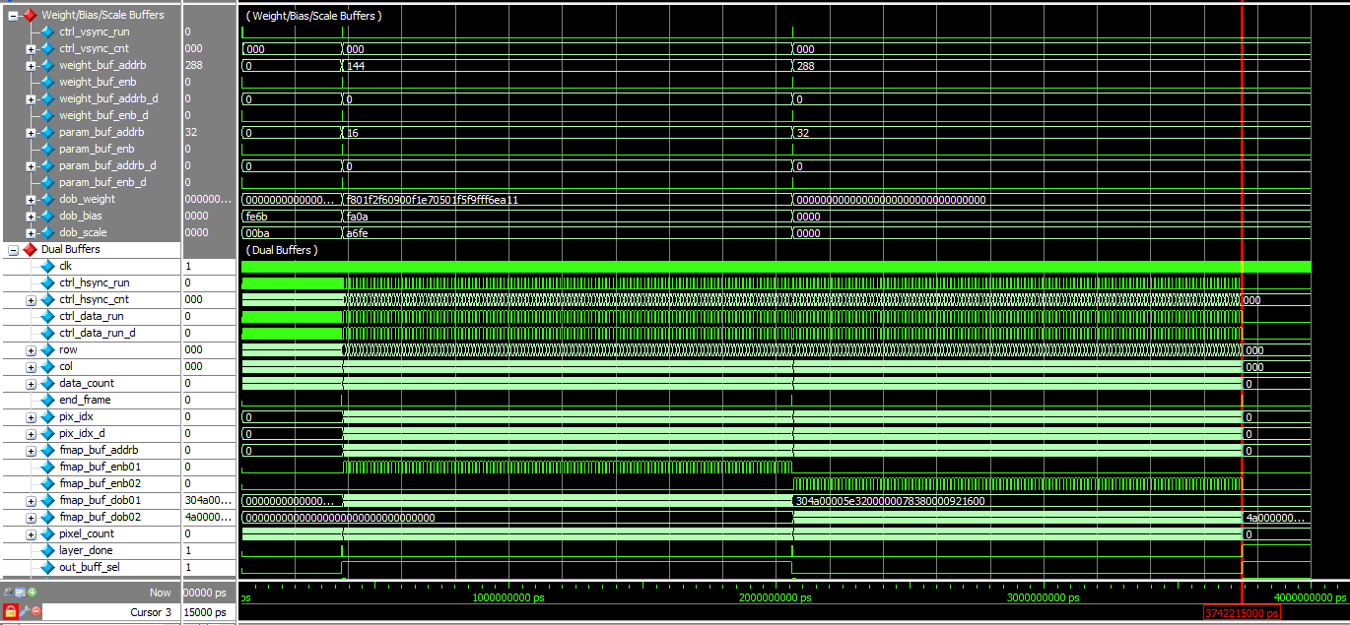


Fig. 7-2: A captured waveform of cnn\_accel.

1. Use check\_hardware\_results.m to verify the output images generated by the H/W simulation.
2. (2p) Explain how the flags q\_is\_first\_layer and out\_buff\_sel are used in this code.

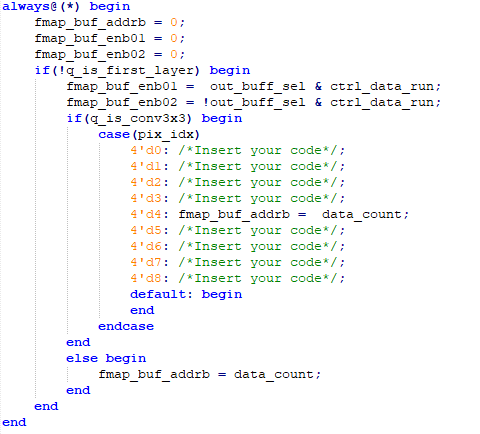


Fig. 1-4: The code used to access dual buffers

1. (3p) Determine the buffer sizes by completing the following table:

|  |  |  |  |
| --- | --- | --- | --- |
| Buffer | Buffer size | | |
| Word (bit) | Number of words | Size (bit) |
| Input buffer (in\_img) | 8 | 16384 | 131072 |
| Buffer 1 | 128 | 16384 | 2097152 |
| Buffer 2 | 128 | 16384 | 2097152 |

1. (2p) As explained in the class, in our CNN accelerator (cnn\_accel.v), each convolution kernel (conv\_kern.v) is used to generate an output feature map. In each convolution kernel, Ti multipliers are used to do convolution on the pixels from multiple input feature maps. Fig. 1-5 shows the captured waveform of convolution kernels 0 and 8 when doing the simulation with time = 375us. Explain why the outputs of multipliers 9~15 are zero.

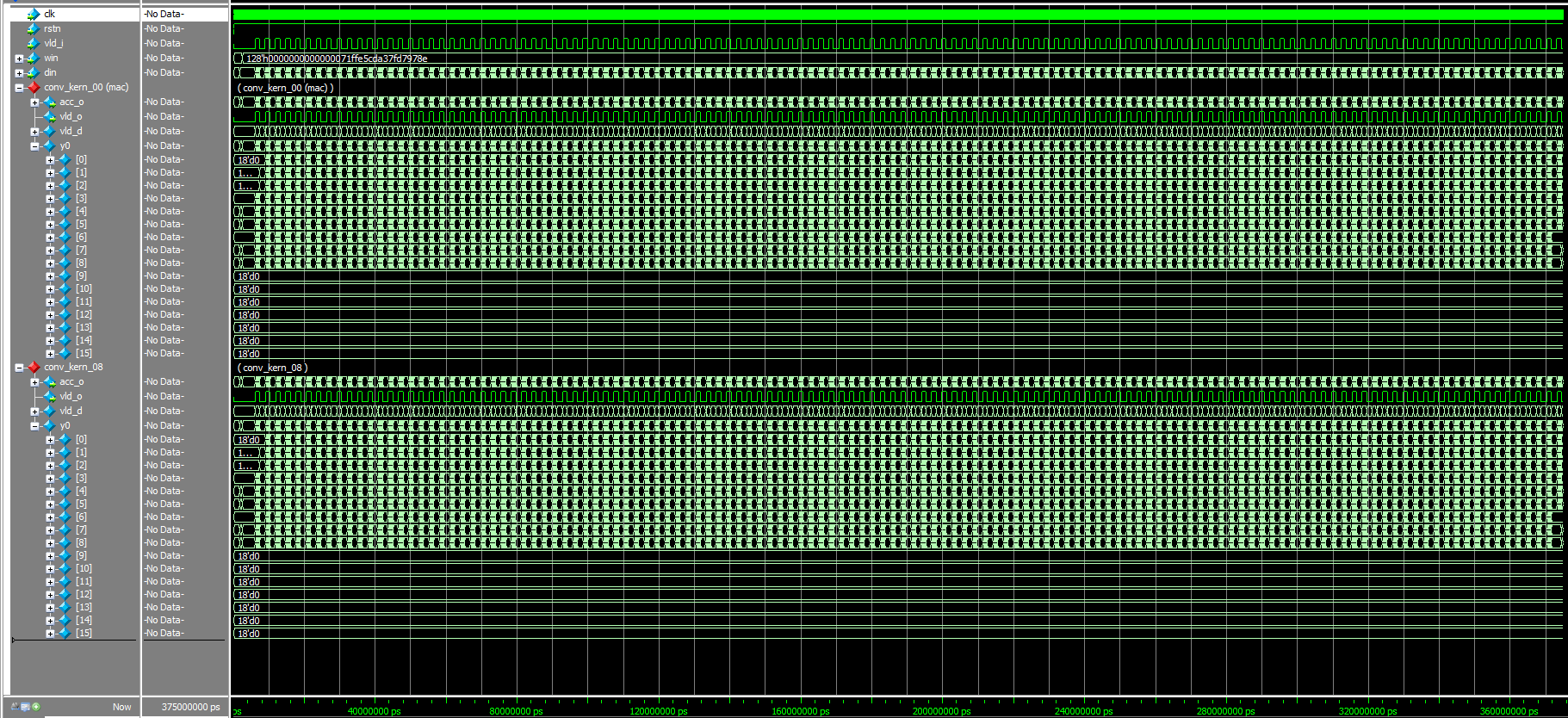


Fig. 1-5: A captured waveform of convolution kernel 0 and 8.

1. (2p) After the simulation, the output of the last layer is stored in conv\_output\_L03.txt, as shown in Fig. 6. Explain the file’s data format.

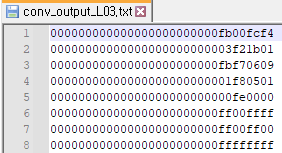
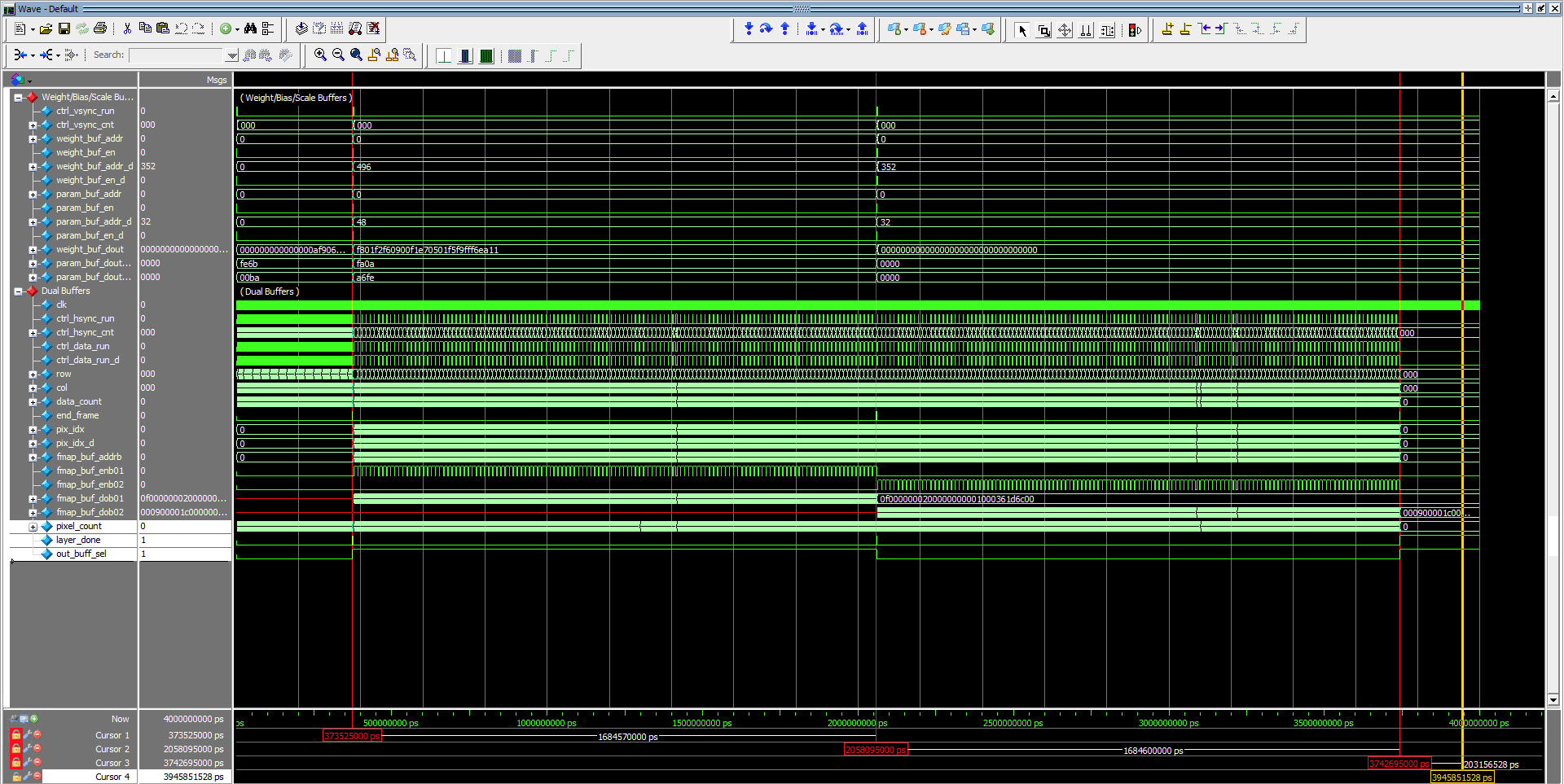
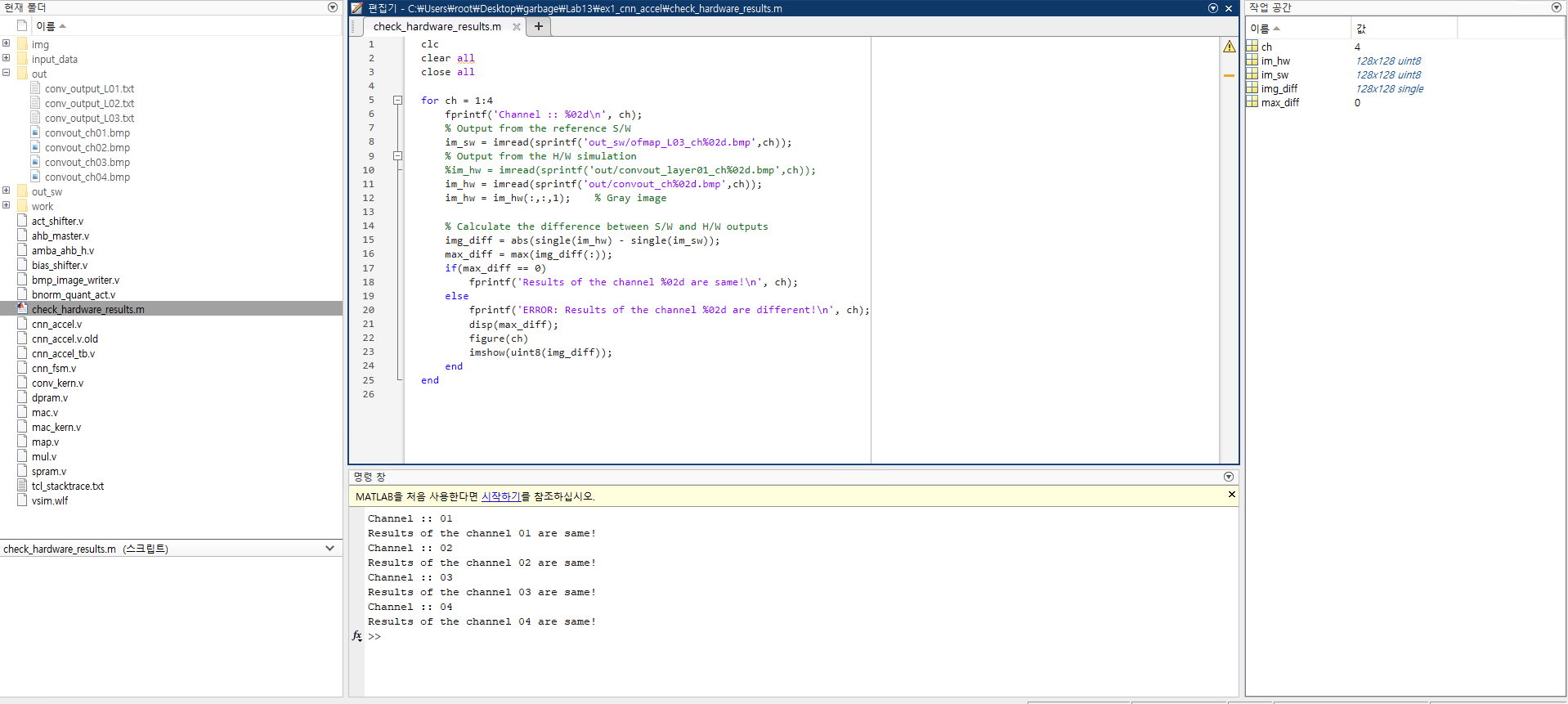
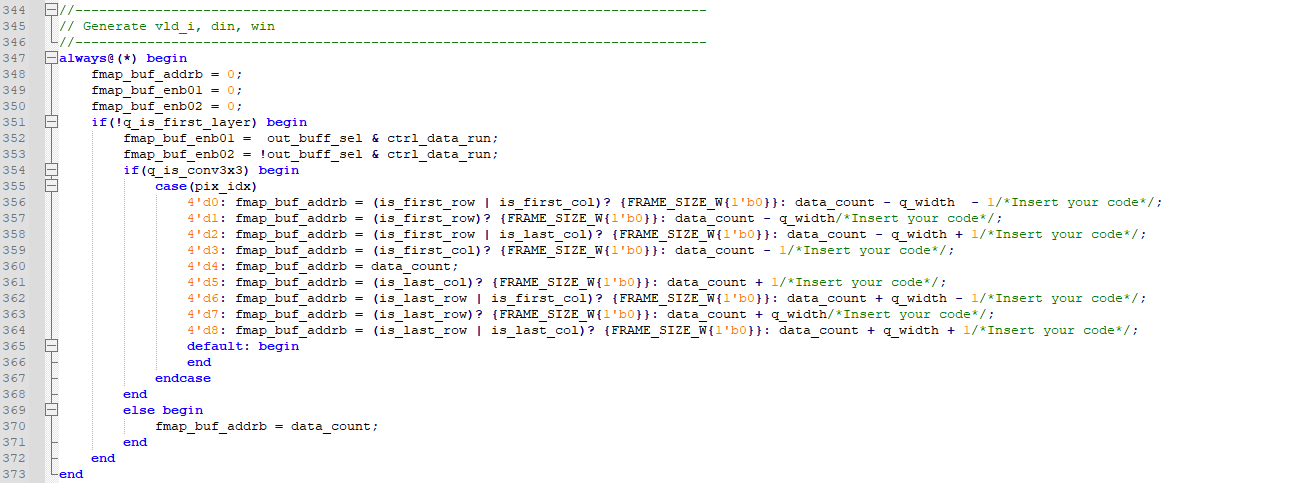


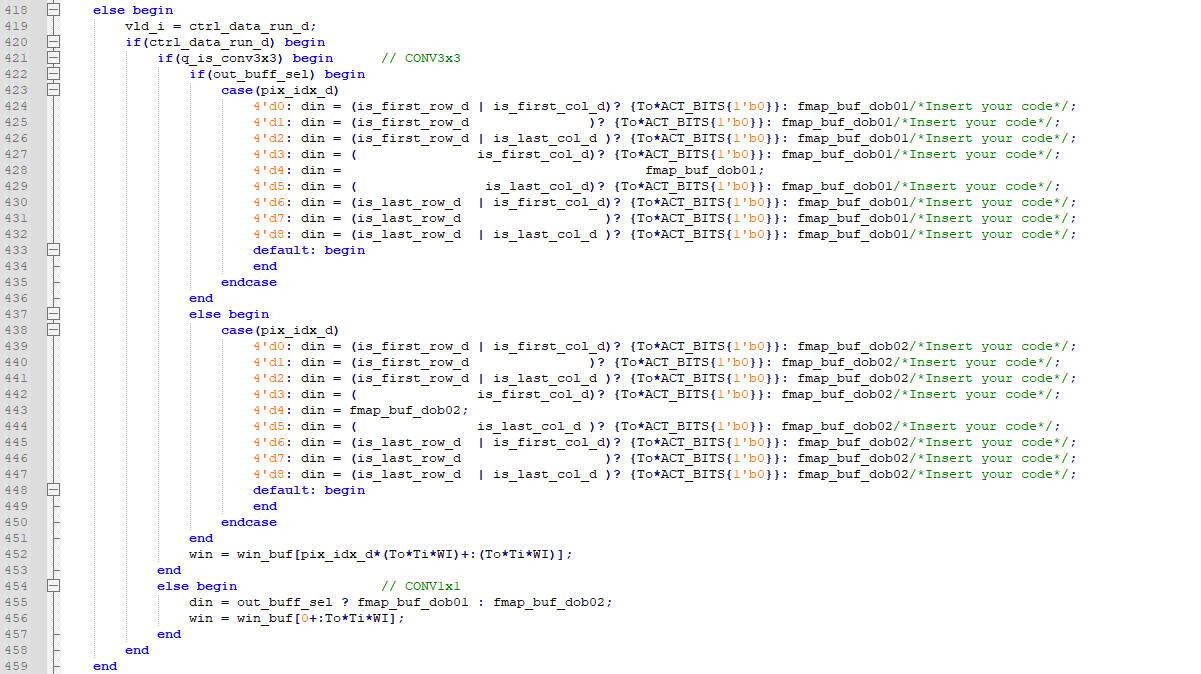
Fig. 1-6: The captured output file of the network after simulation.

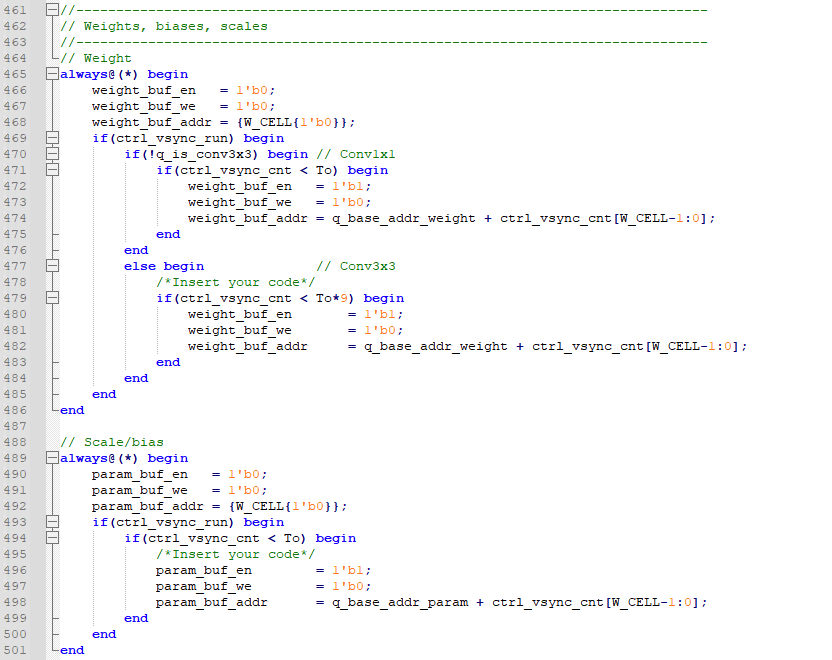
**Solution 1-2 : Dual buffers for feature maps**

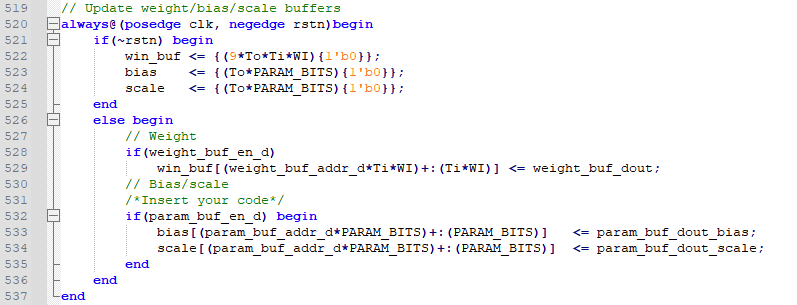
****

****

****

****

****

****

**(f): The reason why the 9~15th multipliers have 0 as result is that we had given the din with only 9 inputs to the mac module. This can be seen in the generation of din signal for the first layer in cnn\_accel.v**

**(g): Final output of the 3rd layer consists of total 4 images. Each image has 128x128x1 dimension and each pixel points are given line by line. In each line, the rightmost two numbers represent the pixel of the first image in hexadecimal representation. And the next 3rd and 4th numbers from the right side of a line represent the pixel of the second image. This notation keeps going through the numbers until the 8th number from the right side.**

**Problem 2 (10p): DMA**

Implement a CNN accelerator in Verilog. Please see the description in lecture 13 for details.

What you have to do:

1. Complete the missing codes in cnn\_accel.v
2. Do a simulation and capture the waveform.
3. Use check\_input.m to verify the output images generated by the H/W simulation.

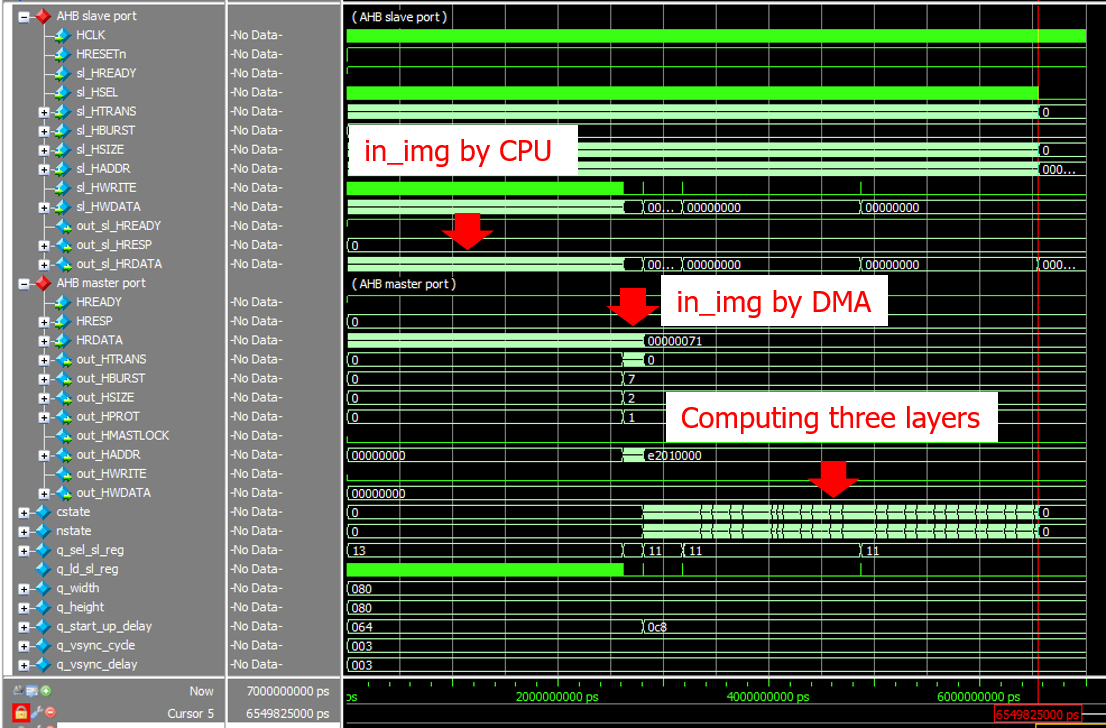
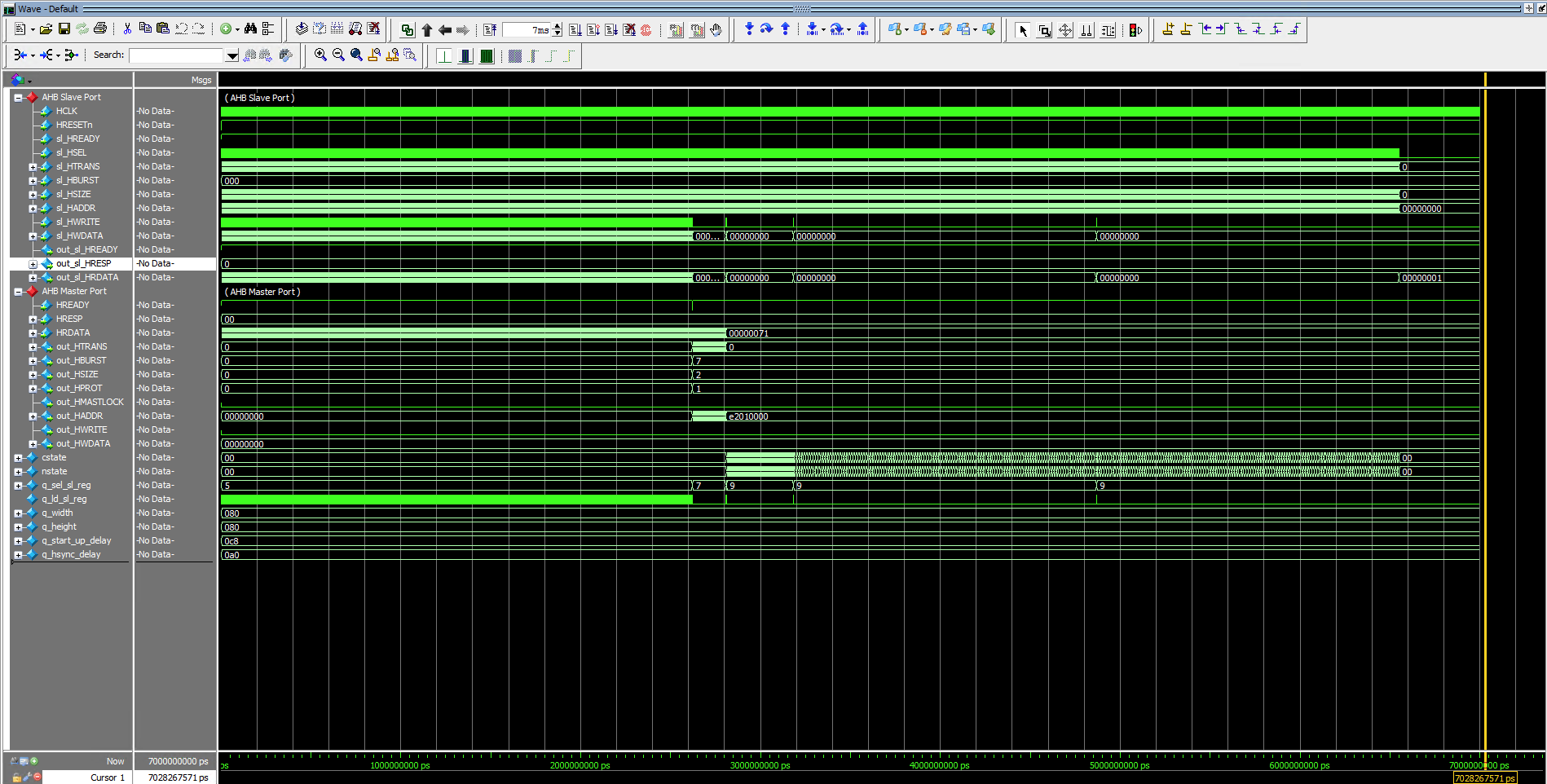
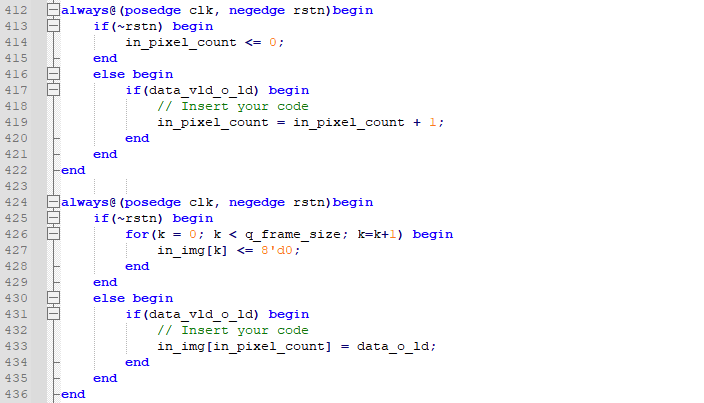
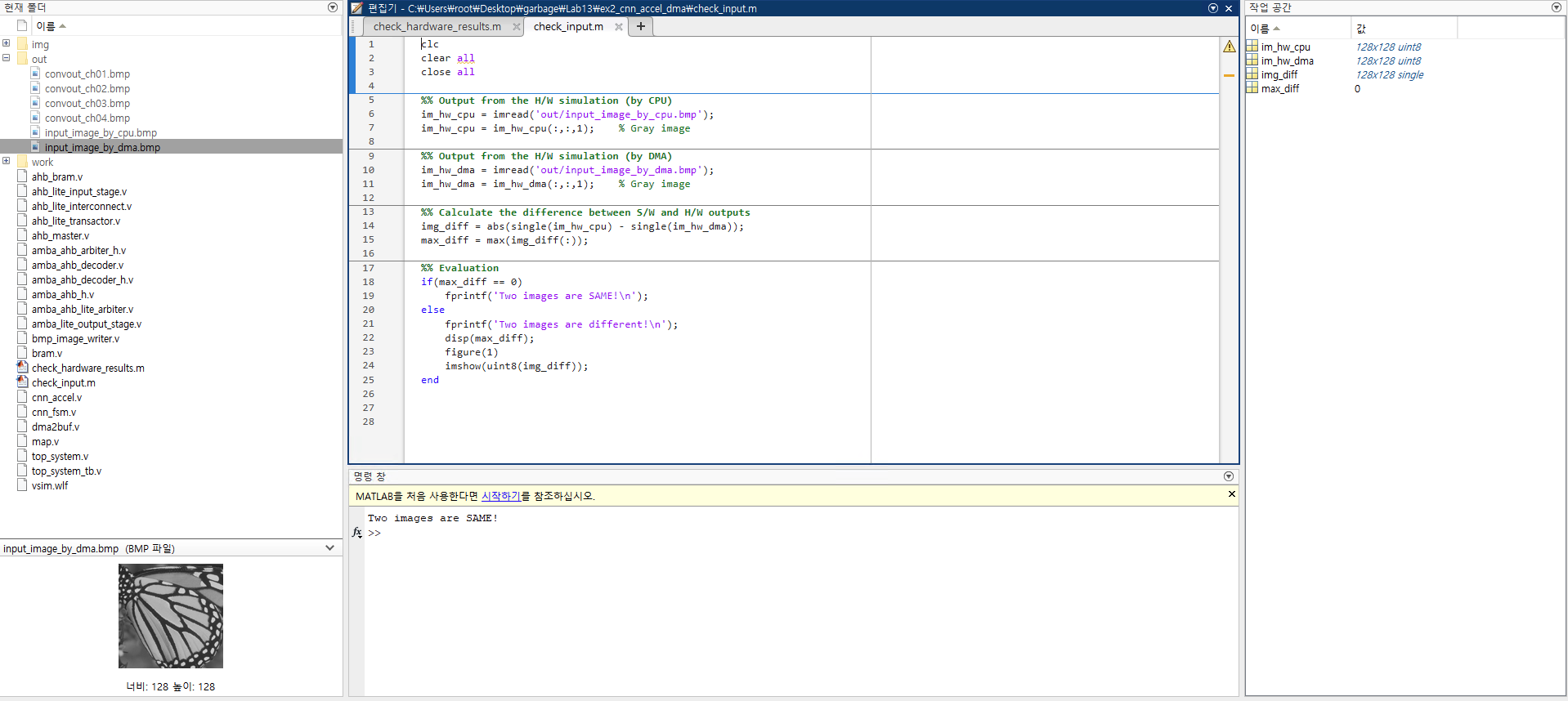


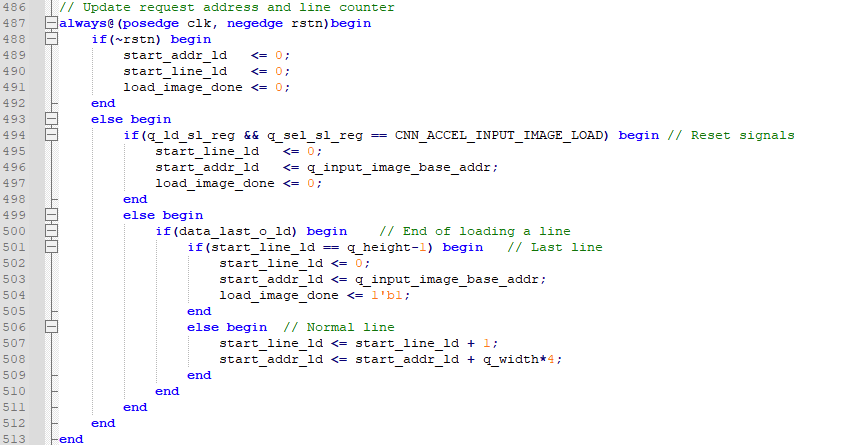
Fig. 8-1: The captured waveform of cnn\_accel.

1. (2p) By analyzing the data patterns of the AHB master port and the AHB slave port of cnn\_accel in Problem 2, explain how DMA can speed up the data reading process compared to the CPU.

**Solution 2 : DMA**

****

****

****

**(d) : When reading the image data from memory through the CPU indirectly, CNN accelerator needs total four arbitrary steps. Request to CPU from CNN accel. – Request to Memory from CPU – Retrieve to CPU from Memory – Retrieve to CNN accel. from CPU But, using AHB interface, all operations need to be executed on AHB ports, which will take extra time. But, using DMA, we don’t need to go through the CPU for the image data from memory. In addition, since we don’t use HWRITE signal, we can significantly reduce the execution time.**