**Homework 9: MAC kernel, Convolutional kernels**

**Issued:** May 9 (Tuesday), 2023 **Due:** May 15 (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 (10p): MAC (Channel-wise accumulation)**

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

What you have to do:

1. Complete the missing codes in mac.v.
2. Do a simulation and capture the waveform.

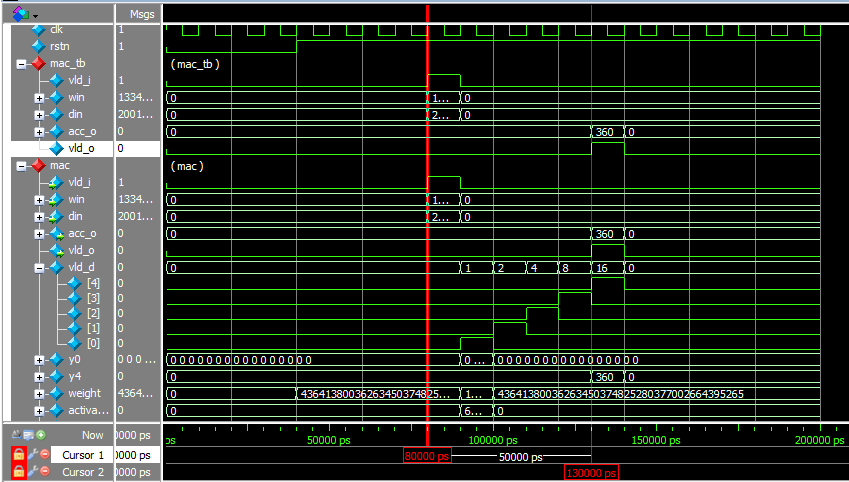
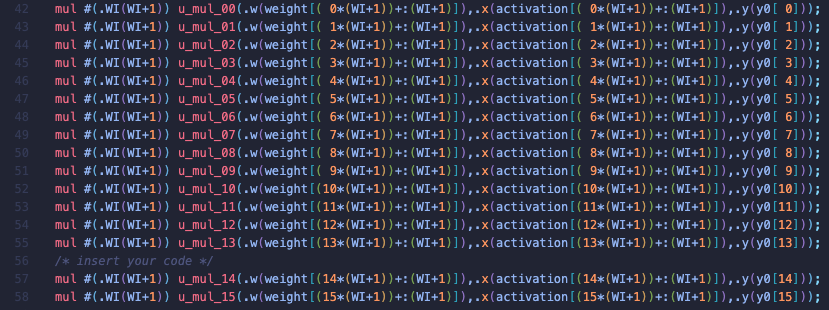
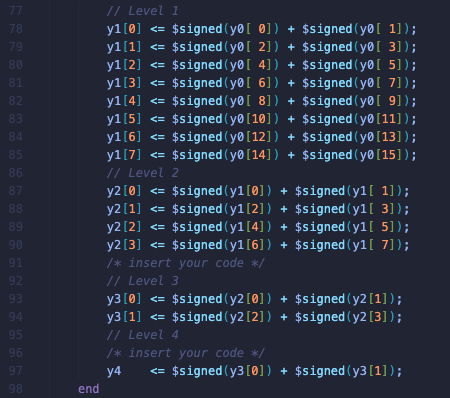


Fig. 1-1: Captured waveform of mark\_kern\_tb, mac\_kern, and mac.

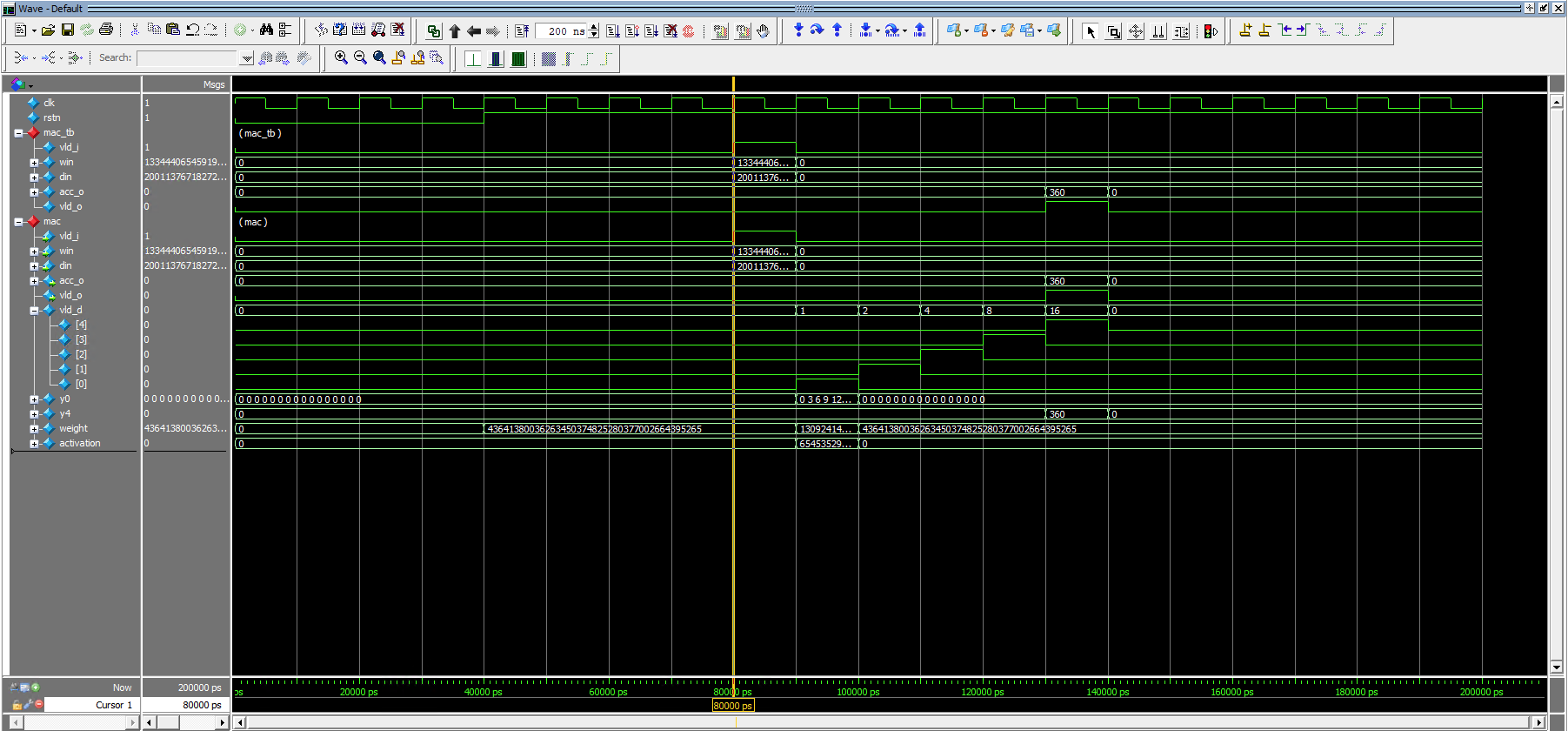
1. (1p) Explain why the final result is 360.
2. (1p) Explain why the output valid signal (vld\_o) delays 5 cycles after the input valid signal (vld\_i).

**Solution 1 : MAC (Channel-wise accumulation)**

****

****

****

****

**(c) : From pre-processing step, mac.v converts win to 2\*w+1, which makes all the 16 weights to 3. And activation is from 0 to 15. Then the convolutional calculation is as same as the first 16 terms of arithmetic sequence, whose difference is 3. And total summation of sequence is 360. This is the reason why the result is 360.**

**(d) : 1 cycle of total 5 cycle is for multiplication, and left 4 cycles are for the accumulation, which is same as the height of accumulation tree.**

**Problem 2 (10p): MAC kernel (Filter-wise accumulation)**

Implement a MAC kernel in Verilog. Please see the description in the lecture note for details.

What you have to do:

1. Reuse the implemented mac.v in Problem 1.
2. Complete the missing codes in mac\_kern.v.
3. Do a simulation and capture the waveform.

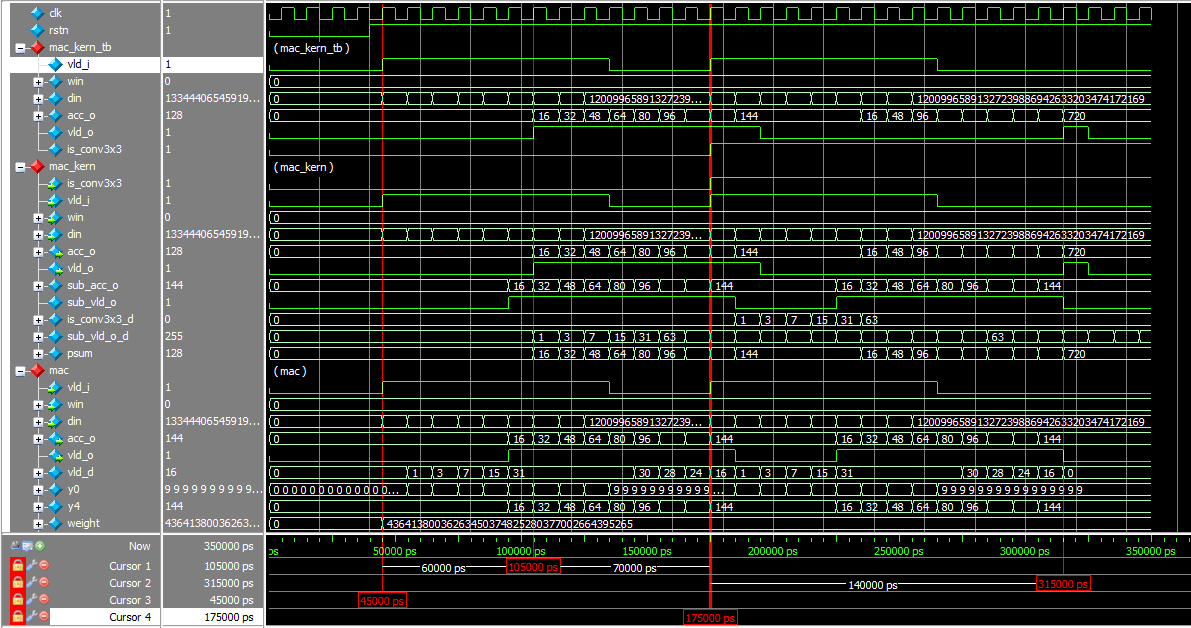
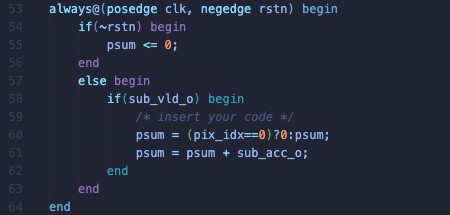
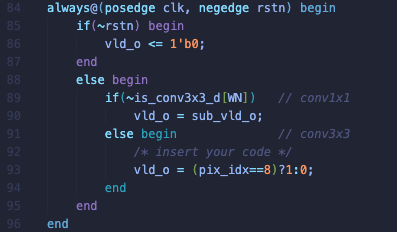


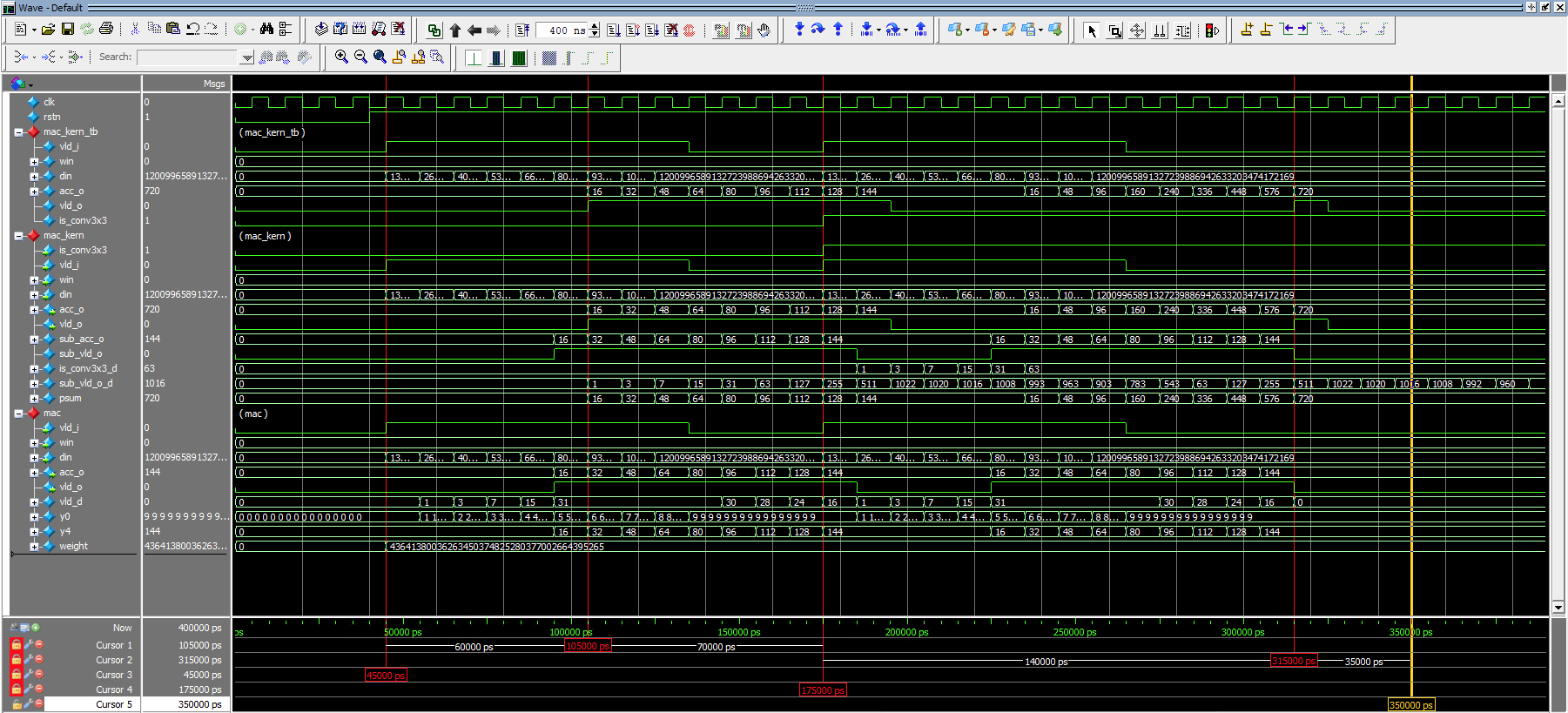
Fig. 2-1: Captured waveform of mark\_kern\_tb, mac\_kern, and mac.

1. (2p) Explain the values of the output port acc\_o.
2. (2p) According to the waveform, the time intervals between vld\_i and vld\_o of mac\_kern are 6 and 14 cycles for conv1x1 and conv3x3, respectively. Explain those numbers.

**Problem 2 : MAC Kernel**

****

****

****

**(d) : The first testcase has win=16{8’b0}(which will be converted to 1 in pre-processing step), increasing din from 16{8’b1} to 16{8’b9}. So, the output of the testcases will be from 16 to 144 increasing by 16 respectively since this testcase is for conv1x1.**

**The second testcase has same value with the first testcase but is\_conv3x3 is 1’b1, which means the testcase is for conv3x3. The output of conv3x3 will be the summation of conv1x1 for each input which is summation of outputs from the conv1x1 since the input is same. So, final output is 720=(16+144)\*9/2.**

**(e) :**

**Conv1x1 : 1 cycle of total 6 cycles is for multiplication, 4 cycles of total 6 cycles are for accumulation and last 1 cycle of total 6 cycles is for partial sum.**

**Conv3x3 : 6 cycle of total 14 cycles are for conv1x1, and remaining 8 cycles is for partial sum of remaining 8 conv1x1 computations.**

**Problem 3 (10p): Convolutional Kernel**

Implement a convolutional kernel and its test bench in Verilog. Please see the description in the lecture note for details.

What you have to do:

1. Reuse the implemented mac.v and mac\_kern.v in Problems 1 and 2.
2. Complete the missing codes in bias\_shifter.v, act\_shifter.v and conv\_kern\_tb.v.
3. Do a simulation and capture the waveform.

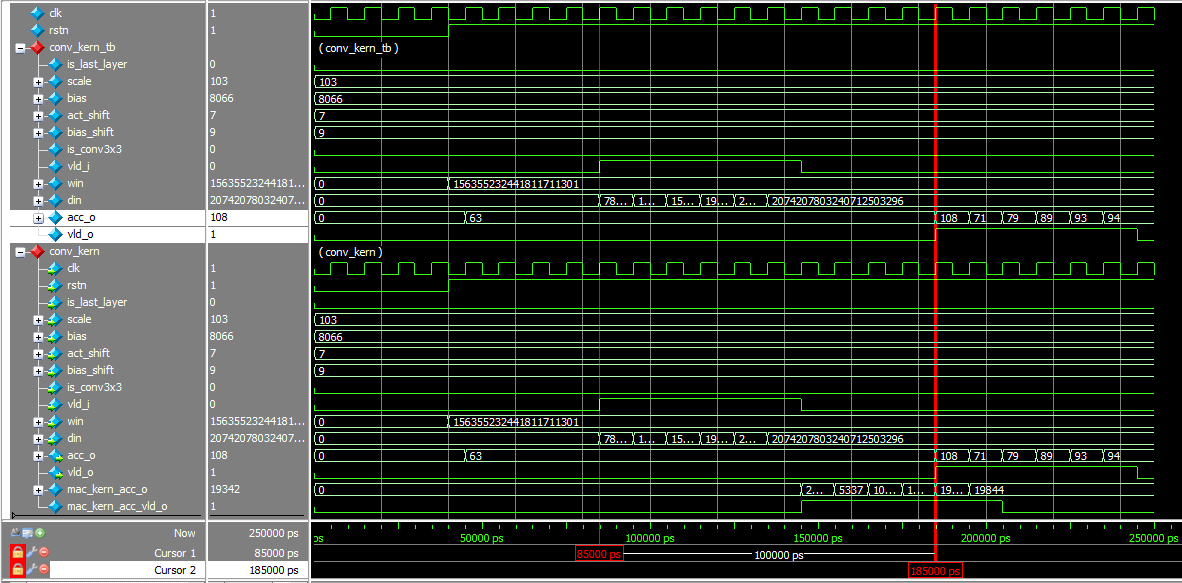
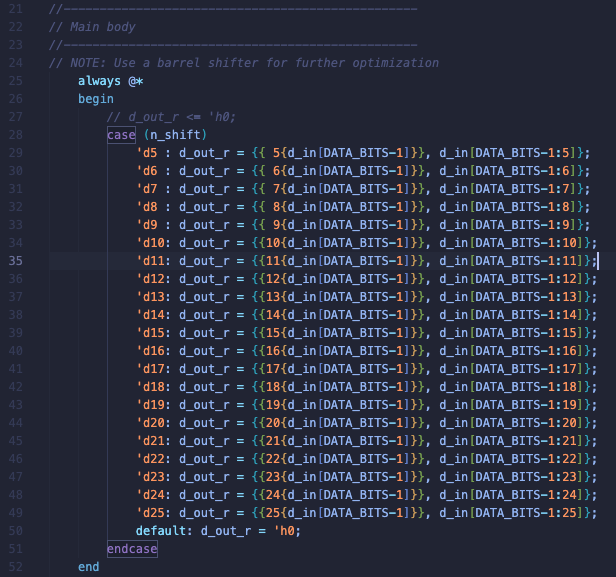
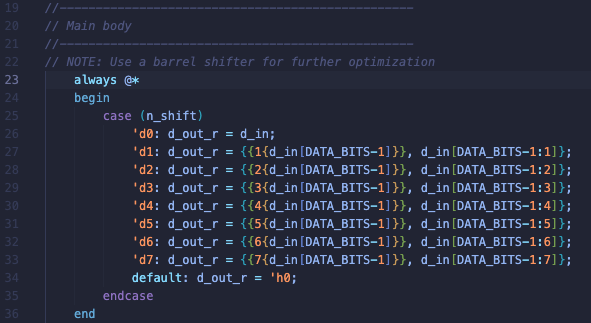
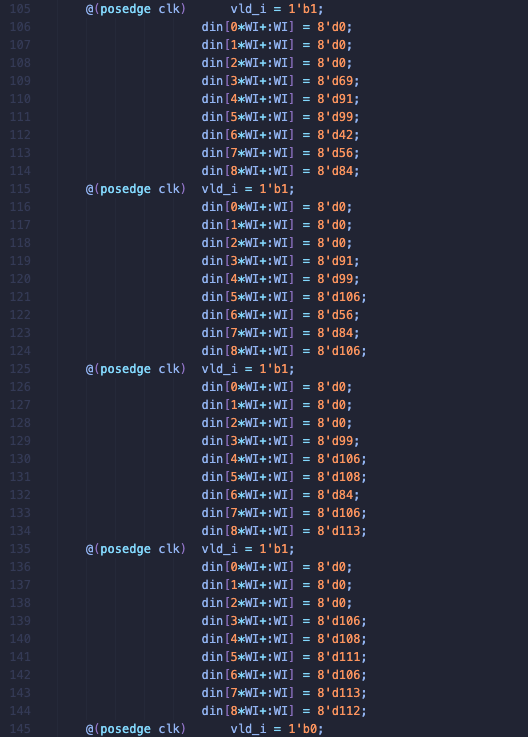


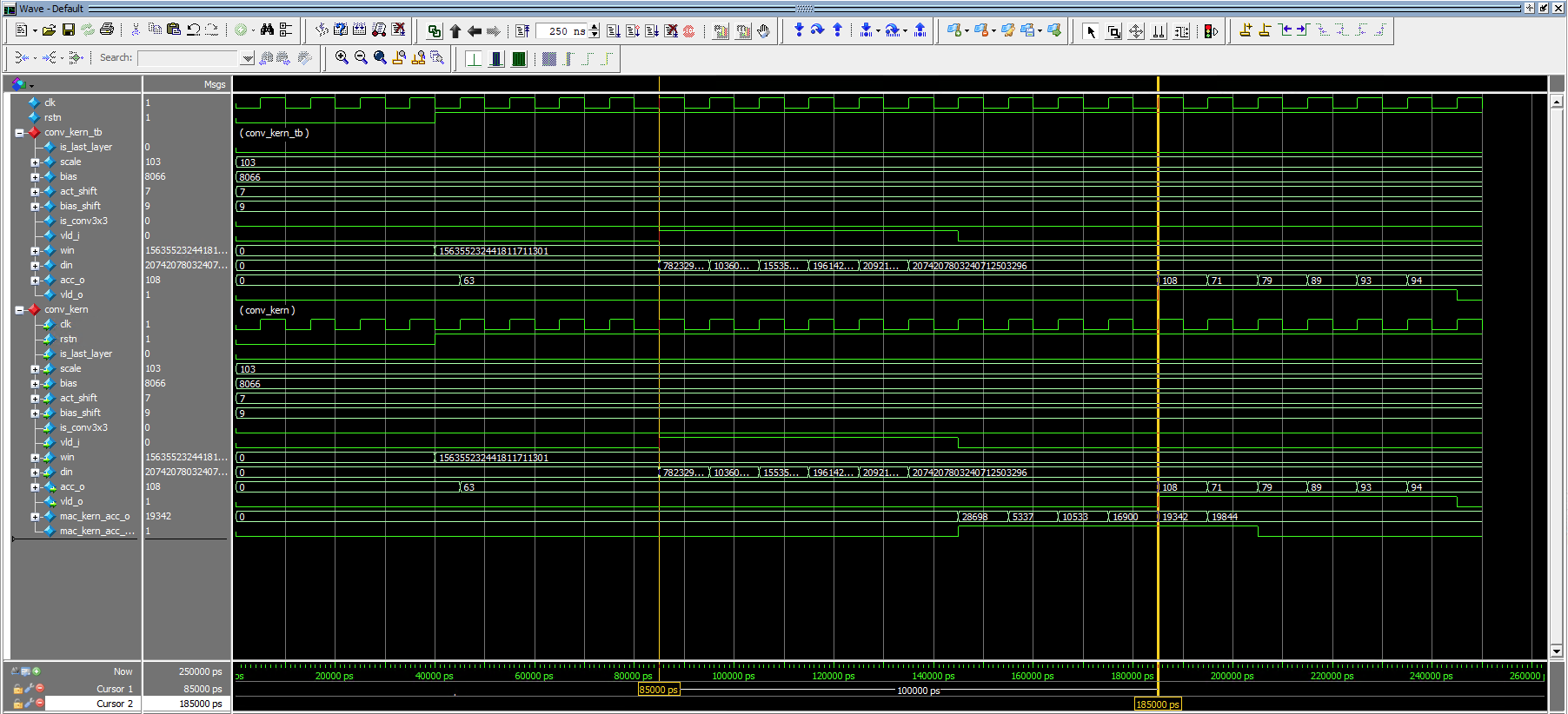
Fig. 3-1: Simulation results.

**Solution 3 : Convolutional Kernel**

****

****

****

****

**Problem 4: (Optional) Sliding window (2p)**

Let H and W be the input image’s height and weight, respectively. In our case, H = 128 and W = 128. Assume an input image is stored in an array in\_img[H][W] whose element has 8 bits. We aim to compute an output feature map stored in an array out\_img[H][W]. Like Problem 3, let’s assume that weights (win) are pre-defined. Your task is to complete the following pseudocode:

for h = 0 to H-1

for w = 0 to W-1

// Generate din from in\_img

//{{{

// Insert your code

//}}}

// Compute an output pixel which corresponds to acc\_o of conv\_kern in Problem 3

out\_img[h][w] = conv\_kern(win,din)

end for

end for

**Solution 4 : Sliding Window**

**// Insert your code**

**din[0][0]=(h==0)?8’b0:((w==0)?8’b0:in\_img[h-1][w-1])**

**din[0][1]=(h==0)?8’b0:in\_img[h-1][w]**

**din[0][2]=(h==0)?8’b0:((w==W-1)?8’b0:in\_img[h-1][w+1])**

**din[1][0]=(w==0)?8’b0:in\_img[h][w-1]**

**din[1][1]=in\_img[h][w]**

**din[1][2]=(w==W-1)?8’b0:in\_img[h][w+1]**

**din[2][0]=(h==H-1)?8’b0:((w==0)?8’b0:in\_img[h+1][w-1])**

**din[2][1]=(h==H-1)?8’b0:in\_img[h+1][w]**

**din[2][2]=(h==H-1)?8’b0:((w==W-1)?8’b0:in\_img[h+1][w+1])**

**[About a timing issue in Vivado]** If you use Vivado for simulation, please follow the template below to fix the timing synchronization issue. In particular, a delay (vld\_d) is generated from a signal (vld\_i\_s) that is assigned to an input (vld\_i).

