**Homework 3: IO, Convolutional neural network kernel**

**Issued:** January 10 (Tuesday), 2023 **Due: January 16 (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 to the screen, also include the plot and screen output your code generates. Submit either a \*.doc or \*.pdf file.

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

**Problem 1 (10p): LCD Drive Interface**

Implement an AHB interface of LCD Drive and integrate it into the top system. Please see the description in the lecture note for details.

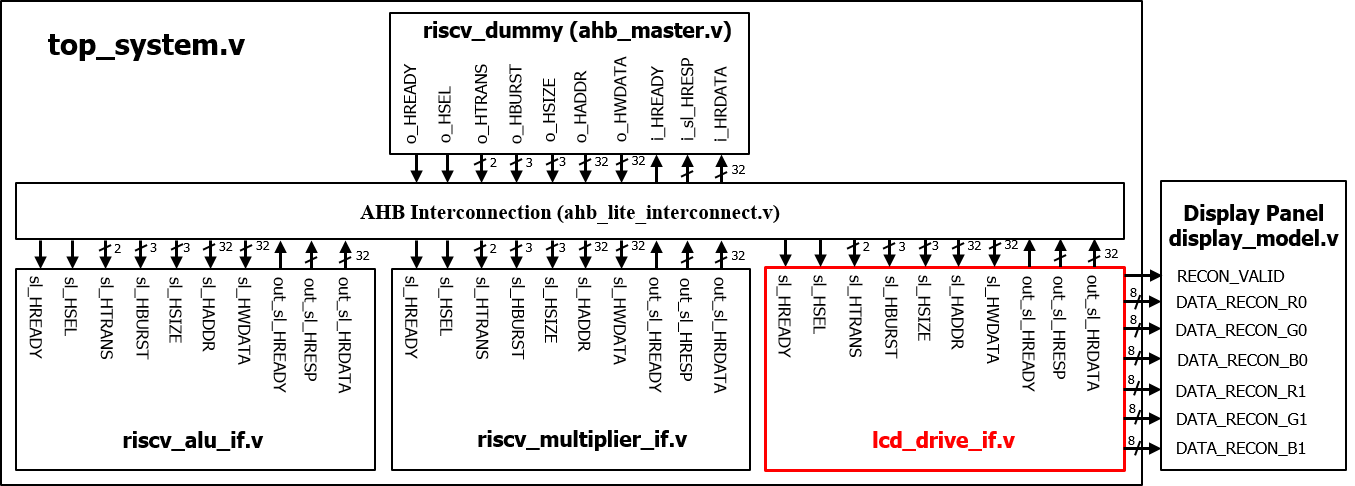
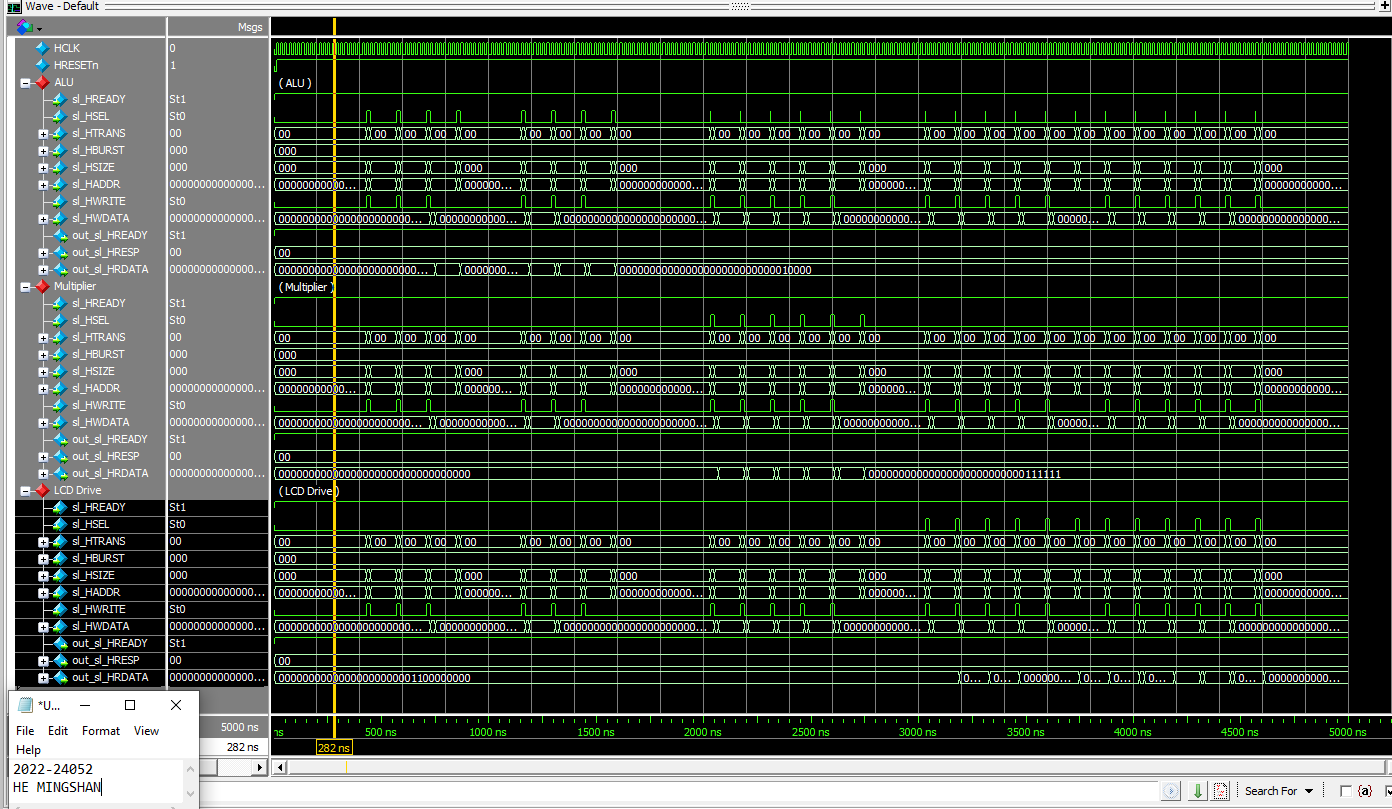


Fig. 1-1: A top system using an AHB bus.

What you have to do:

1. **Create a custom IP (lcd\_drive\_if.v) and add it to the top system**

* Complete the code to create the AHB slave interface of the LCD drive (lcd\_drive\_if.v).
* Add its base address and the register map (map.v).
* Complete the code to add the LCD drive IP to the Bus interconnect (top\_system.v).
* Do a simulation with time = 5.000 ns
* Capture the simulation result, i.e., waveform



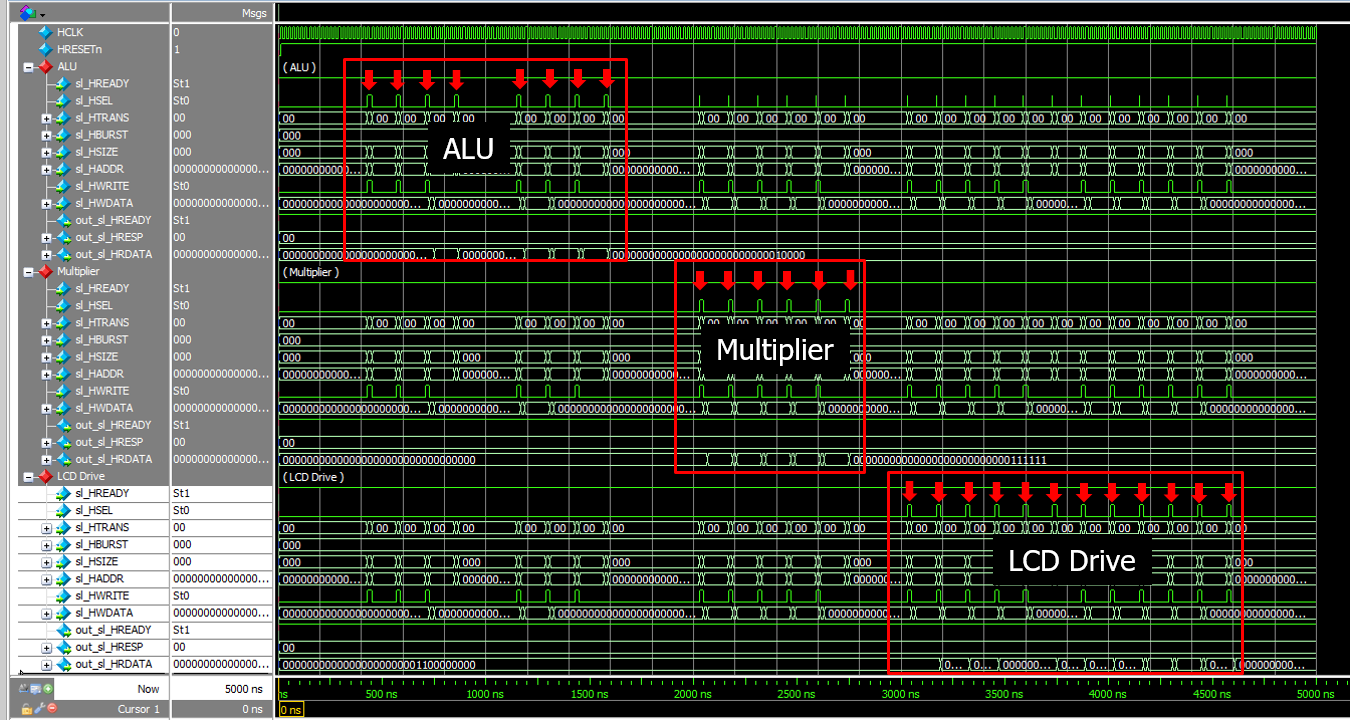
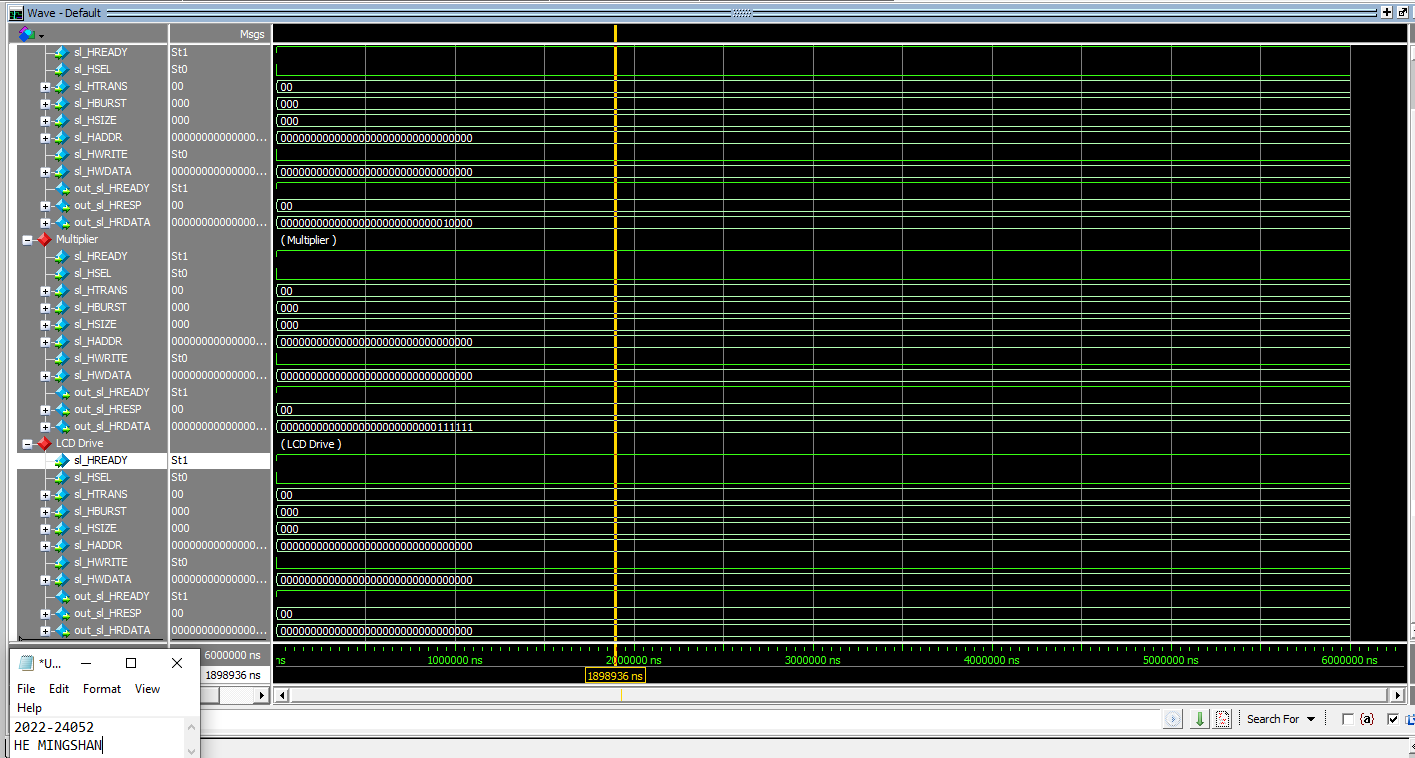


Fig. 1-2: Waveform with highlighted Select signals of three AHB slaves.

1. **Generate data signals (lcd\_drive\_if.v)**

* Complete the missing codes related Finite state machine, row and column counters, and the end-of-frame signal.
* Do a simulation with time = 6ms.
* Test the system with different modes.
* Capture the results, including the waveform and the output images.



|  |  |
| --- | --- |
|  |  |
|  |  |

**Submit your RTL codes**.

1. /\* Insert your code \*/
2. q\_height <= sl\_HWDATA[W\_SIZE-1 :0];
3. /\* Insert your code \*/
4. q\_data\_count <= sl\_HWDATA[W\_DELAY-1 :0];
5. /\* Insert your code \*/
6. q\_start <= sl\_HWDATA[W\_DELAY-1 :0];
7. /\* Insert your code \*/
8. q\_br\_mode <= sl\_HWDATA[W\_DELAY-1 :0];
9. /\* Insert your code \*/
10. q\_br\_value <= sl\_HWDATA[W\_DELAY-1 :0];
11. /\* Insert your code \*/
12. out\_sl\_HRDATA = q\_height;
13. /\* Insert your code \*/
14. out\_sl\_HRDATA = q\_data\_count;
15. /\* Insert your code \*/
16. out\_sl\_HRDATA = q\_start;
17. /\* Insert your code \*/
18. out\_sl\_HRDATA = q\_br\_mode;
19. /\* Insert your code \*/
20. out\_sl\_HRDATA = q\_br\_value;
21. **if**(q\_start/\*Insert your code\*/)
22. **if**(ctrl\_hsync\_cnt == HSYNC\_DELAY/\*Insert your code\*/)
23. **if**(col == q\_width - 2/\*Insert your code\*/)//end of line
24. **if**(col == q\_width - 2/\*Insert your code\*/)
25. assign end\_frame = (data\_count == 196607/\*Insert your code\*/)? 1'b1: 1'b0;



Fig. 1-3: Simulation results.

**Problem 2 (10p): BRAM Controller**

Implement an AHB interface of LCD Drive and integrate it into the top system. Please see the description in the lecture note for details.

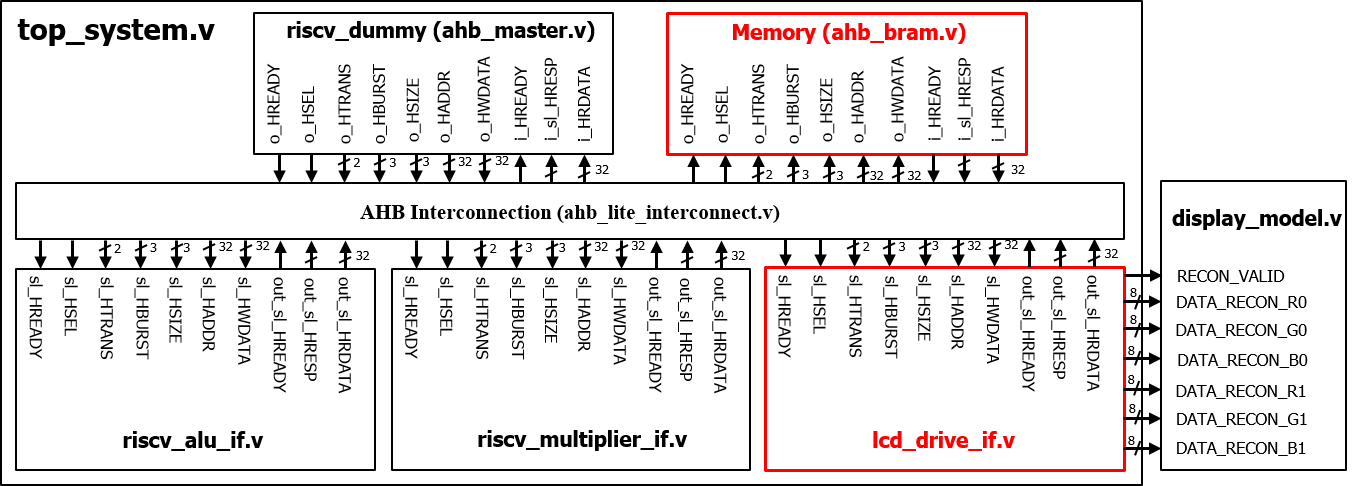
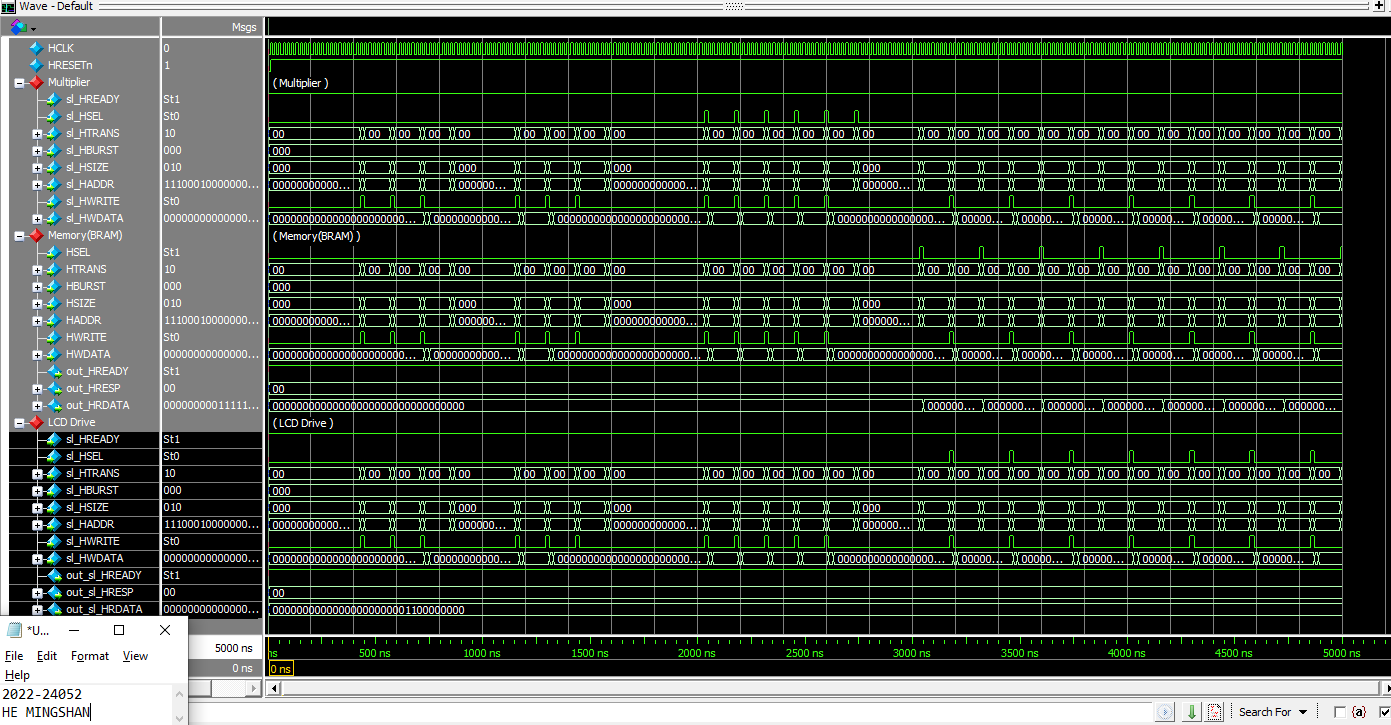


Fig. 2-1: A top system using an AHB bus.

What you have to do:

1. **Bus interface and IP integration (Similar to Problem 1)**

* Complete the code to create the AHB slave interface of LCD drive (lcd\_drive\_if.v).
* Complete the code to add the LCD drive IP to the Bus interconnect (top\_system.v).
* Do simulation with time = 5.000 ns
* Capture the simulation result, i.e., wave form



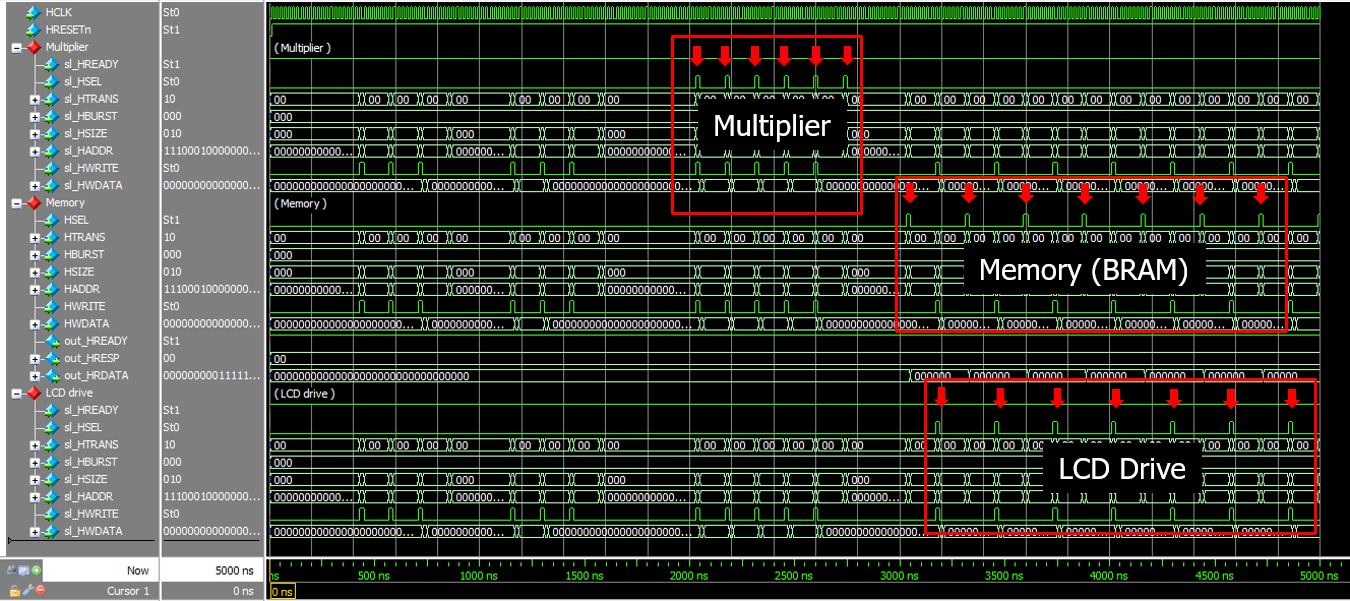
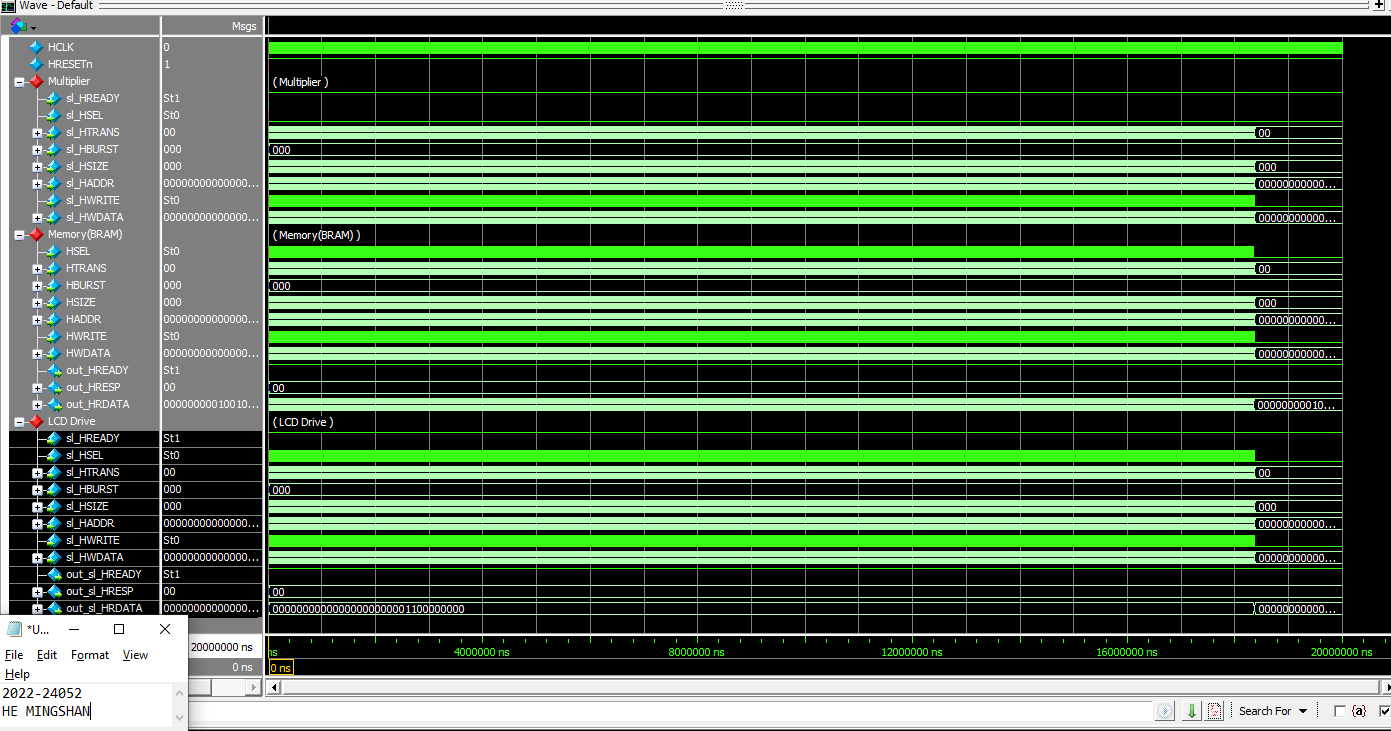


Fig. 2-2: Waveform with highlighted Select signals of three AHB slaves.

1. **Generate data signals (lcd\_drive\_if.v)**

* Complete the missing codes.
* Do a simulation with time = 20 ms.
* Test the system with different modes.
* Capture the results, including the waveform and the output images.



|  |  |
| --- | --- |
|  |  |
|  |  |

**Submit your RTL codes**.

1. //  3. AHB2LCDDRIVE
2. assign  lcd\_sl\_HSEL         =   w\_AHB\_IC\_out\_sl\_HSEL    [2]/\*Insert your code\*/;
3. assign  lcd\_sl\_HADDR        =   w\_AHB\_IC\_out\_sl\_HADDR   [2\*32+:32]/\*Insert your code\*/;
4. assign  lcd\_sl\_HTRANS       =   w\_AHB\_IC\_out\_sl\_HTRANS  [2\*2+:2]/\*Insert your code\*/;
5. assign  lcd\_sl\_HBURST       =   w\_AHB\_IC\_out\_sl\_HBURST  [2\*`W\_BURST+:`W\_BURST]/\*Insert your code\*/;
6. assign  lcd\_sl\_HSIZE        =   w\_AHB\_IC\_out\_sl\_HSIZE   [2\*3+:3]/\*Insert your code\*/;
7. assign  lcd\_sl\_HPROT        =   w\_AHB\_IC\_out\_sl\_HPROT   [2\*4+:4]/\*Insert your code\*/;
8. assign  lcd\_sl\_HWRITE       =   w\_AHB\_IC\_out\_sl\_HWRITE  [2]/\*Insert your code\*/;
9. assign  lcd\_sl\_HWDATA       =   w\_AHB\_IC\_out\_sl\_HWDATA  [2\*32+:32]/\*Insert your code\*/;
10. assign  lcd\_sl\_HREADY       =   w\_AHB\_IC\_out\_sl\_HREADY  [2]/\*Insert your code\*/;
11. assign  w\_AHB\_IC\_sl\_HREADY  [2] /\*Insert your code\*/=   out\_lcd\_sl\_HREADY;
12. assign  w\_AHB\_IC\_sl\_HRESP   [2\*2+:2]/\*Insert your code\*/=   out\_lcd\_sl\_HRESP;
13. assign  w\_AHB\_IC\_sl\_HRDATA  [2\*32+:32]/\*Insert your code\*/= out\_lcd\_sl\_HRDATA;
15. //  4. AHB2MEM
16. assign  mem\_sl\_HSEL         =   w\_AHB\_IC\_out\_sl\_HSEL    [3]/\*Insert your code\*/;
17. assign  mem\_sl\_HADDR        =   w\_AHB\_IC\_out\_sl\_HADDR   [3\*32+:32]/\*Insert your code\*/;
18. assign  mem\_sl\_HTRANS       =   w\_AHB\_IC\_out\_sl\_HTRANS  [3\*2+:2]/\*Insert your code\*/;
19. assign  mem\_sl\_HBURST       =   w\_AHB\_IC\_out\_sl\_HBURST  [3\*`W\_BURST+:`W\_BURST]/\*Insert your code\*/;
20. assign  mem\_sl\_HSIZE        =   w\_AHB\_IC\_out\_sl\_HSIZE   [3\*3+:3]/\*Insert your code\*/;
21. assign  mem\_sl\_HPROT        =   w\_AHB\_IC\_out\_sl\_HPROT   [3\*4+:4]/\*Insert your code\*/;
22. assign  mem\_sl\_HWRITE       =   w\_AHB\_IC\_out\_sl\_HWRITE  [3]/\*Insert your code\*/;
23. assign  mem\_sl\_HWDATA       =   w\_AHB\_IC\_out\_sl\_HWDATA  [3\*32+:32]/\*Insert your code\*/;
24. assign  mem\_sl\_HREADY       =   w\_AHB\_IC\_out\_sl\_HREADY  [3]/\*Insert your code\*/;
25. assign  w\_AHB\_IC\_sl\_HREADY  [3]/\*Insert your code\*/=    out\_mem\_sl\_HREADY;
26. assign  w\_AHB\_IC\_sl\_HRESP   [3\*2+:2]/\*Insert your code\*/=   out\_mem\_sl\_HRESP;
27. assign  w\_AHB\_IC\_sl\_HRDATA  [3\*32+:32]/\*Insert your code\*/= out\_mem\_sl\_HRDATA;



Fig. 2-3: Simulation results.

1. **Frame buffer**

* Explain the input and output signals of lcd\_frame\_buffer, i.e. size, meaning.

Answer: Inputs: `clk` means the clock input signal, `en` means the BRAM enable signal; `addr` ‘s size is clog2(width\*height) and means the address input to read image data by one clock cycle; `din` ‘s size is 24 and means the data input including 8bit R, 8bit G, and 8bit B; `we` means the write enable signal; `addr\_dual\_pixel` ‘s size is clog2(width\*height) and means the address input to read image data by one clock cycle. Outputs: `dout` ’s size is 24 and means the data output including 8bit R, 8bit G, and 8bit B; `dout\_dual\_pixel` ‘s size is 48 and means the data output of two pixel including 2\*8bit R, 2\*8bit G, and 2\*8bit B.

* What is the size of the frame buffer?

Answer: Frame buffer is store an image before display. One pixel needs 24-bit to store RGB data, and the whole image are the size of width\*height, thus the size of the frame buffer 24\*width\*height, which means q\_mem[width\*height-1:0][23:0]

**Problem 3 (10p): Compressed frame buffer**

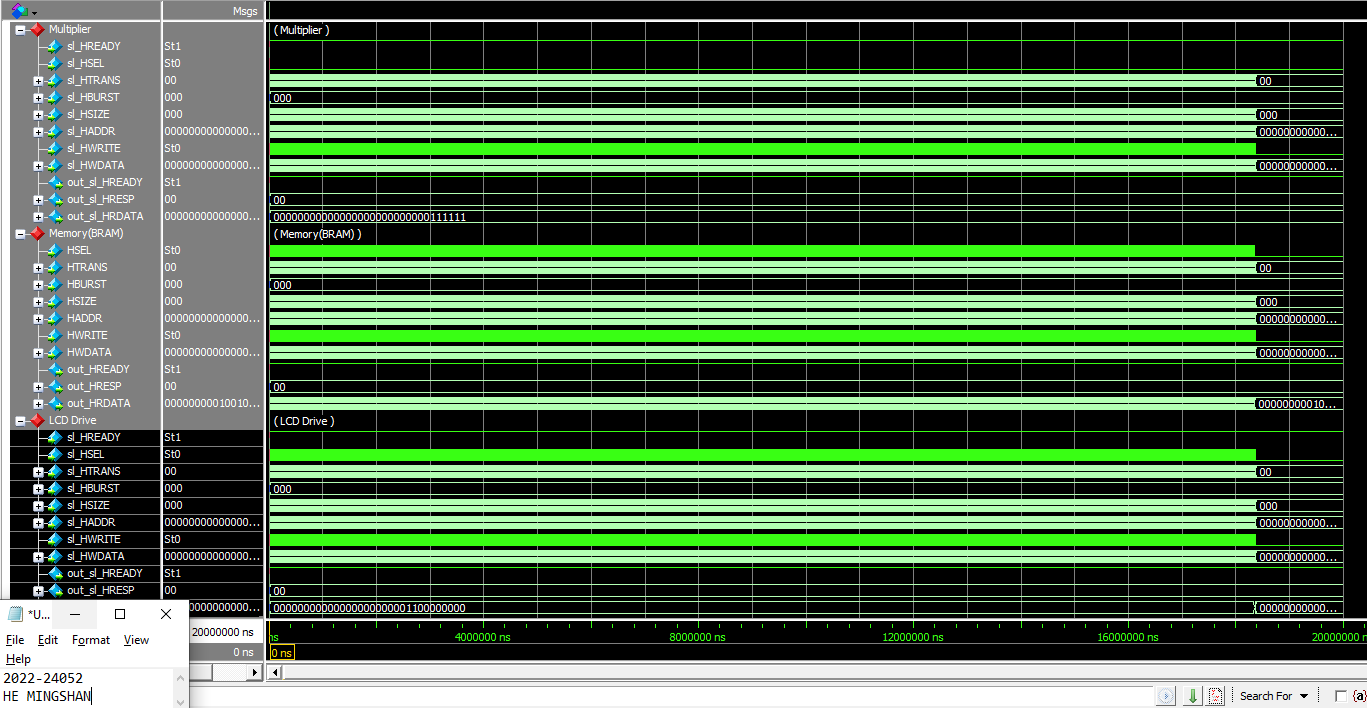
Implement an AHB interface of LCD Drive and integrate it to the top system. Please see the description in the lecture note for details.

1. **Bus interface and IP integration (Similar to Problems 1 and 2)**

* Complete the code to create the AHB slave interface of the LCD drive (lcd\_drive\_if.v).
* Complete the code to add the LCD drive IP to the Bus interconnect (top\_system.v).

1. **Generate data signals (lcd\_drive\_if.v)**

* Complete the missing codes related data signals (lcd\_drive\_if.v).
* Complete the missing codes of RCT modules (lcd\_forward\_rct.v and lcd\_inverse\_rct.v).
* Do simulation with time = 20 ms.
* Test the system with different modes.
* Capture the results including the waveform and the output images.



|  |  |
| --- | --- |
|  |  |
|  |  |



Fig. 3-1: Simulation results.

1. **Frame buffer**

* Explain the input and output signals of lcd\_frame\_buffer\_opt, i.e. size, meaning.

Answer: Inputs: `en` means the enable signal; `we` means the write enable signal with 0 means read and 1 means write; `din` ‘s size is 40 with including Y0, Y1, Cb, Cr and each is 10-bit data;`addr`’s size is clog2(WIDTH\*HEIGHT/2) and means the image is compressed and the `addr` also is compressed with half. `addr\_dual\_pixel`’s size is clog2(WIDTH\*HEIGHT/2), and it is different from `addr`, it can access two pixels per cycle. Outputs: `dout`’ size is 40, which including Y0, Y1, Cb, Cr.

* What is the size of the frame buffer?

Answer: Frame buffer is store an image before display. One pixel needs 40-bit to store Y0,Y1,Cb,Cr data, and the whole image are the size of width\*height/2, thus the size of the frame buffer 20\*width\*height, which means q\_mem[width\*height/2-1:0][39:0]

* Compare the buffer sizes of the baseline frame buffer in Problem 2 and the compressed one in Problem 3.

Answer: If the problem 3 and problem 2 will use the same image to display, the problem 3 will compress the image and using the smaller size of the frame buffer in problem 3.

**Problem 4 (15p): Convolutional Layer**

Implement a function to calculate a convolutional layer in Matlab. Please see the description in the lecture note for details.

What you have to do:

1. Complete the missing codes in convol2.m and test\_SR.m
2. Output the feature maps of all three layers.
3. For each convolutional layer, let NP be the number of multiplication operations for calculating **ONE output pixel**. Complete the following table:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Layer | Filter size | Number of input channels | Number of output channels | Input | Output | NP |
| 1 | 3x3 | 1 | 16 | 128x128x1 | 128x128x16 | 3x3x1 |
| 2 | 3x3 | 16 | 16 | 128x128x16 | 128x128x16 | 3x3x16 |
| 3 | 3x3 | 16 | 4 | 128x128x16 | 128x128x4 | 3x3x16 |

Calculate the total number of multiplication operations.

First Layer: 3\*3\*1 for one output pixel of one feature; 3\*3\*1\*16 for one output pixel of all features;

3\*3\*1\*16\*128\*128 for all output pixels of all features;

Second Layer: 3\*3\*16 for one output pixel of one feature; 3\*3\*16\*16 for one output pixel of all features; 3\*3\*16\*16\*128\*128 for all output pixels of all features;

Third Layer: 3\*3\*16 for one output pixel of one feature; 3\*3\*16\*4 for one output pixel of all features; 3\*3\*16\*4\*128\*128 for all output pixels of all features;

Thus the total number of multiplication operations are equals 2,359,296+37,748,736+9,437,184=49,545,216

**Problem 5 (10p): Quantization**

Implement a function to do weight and activation quantization in Matlab. Please see the description in the lecture note for details.

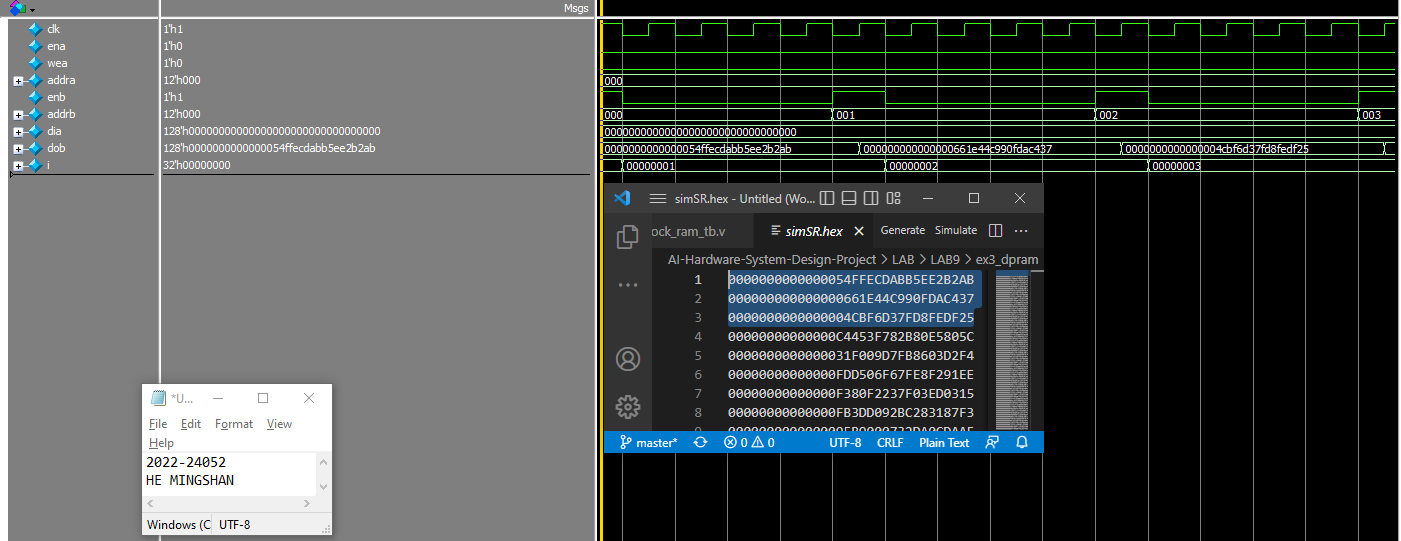
What you have to do:

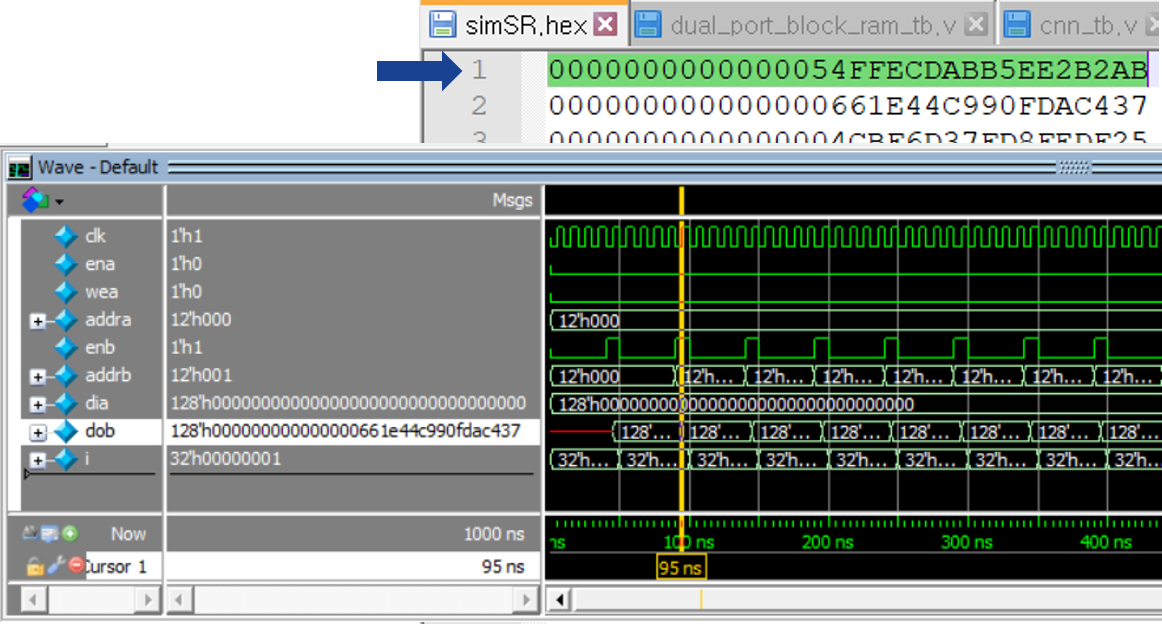
1. Reuse convol2.m in Problem 4.
2. Complete the missing codes in those files: uniform\_quantize.m, hwu\_relu\_quantize.v, hwu\_linear\_quantize and and test\_SR\_quant.m.
3. Output the feature maps of all three layers.

**Problem 6 (10p): DPRAM**

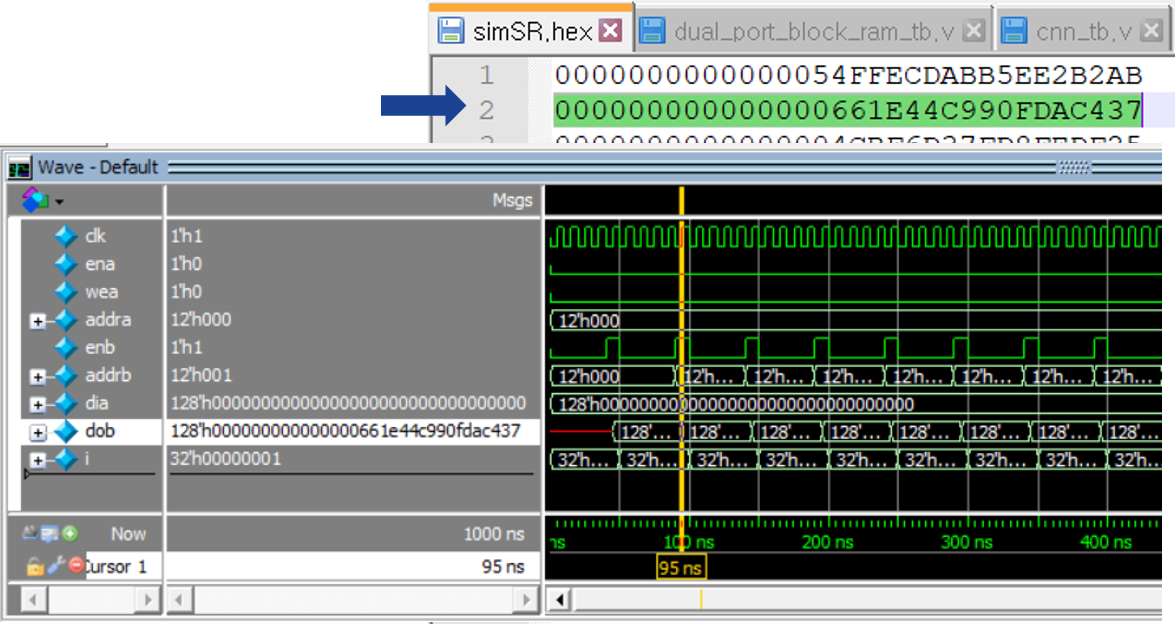
What you have to do:

1. Complete the missing code in dual\_port\_block\_ram\_tb.v to **load 16 conv. filters** (simSR.hex).
2. Capture the simulation result.





1. The first convolution filter



1. The 16-th convolution filter

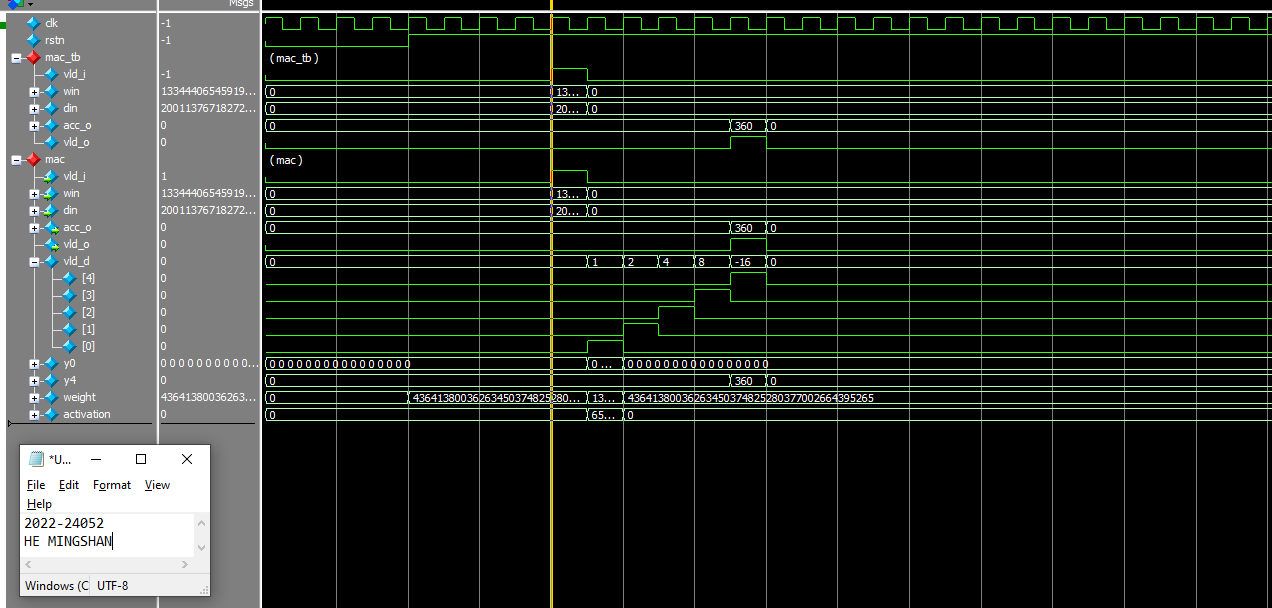
Fig. 6-1: Simulation results.

**Problem 7 (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.



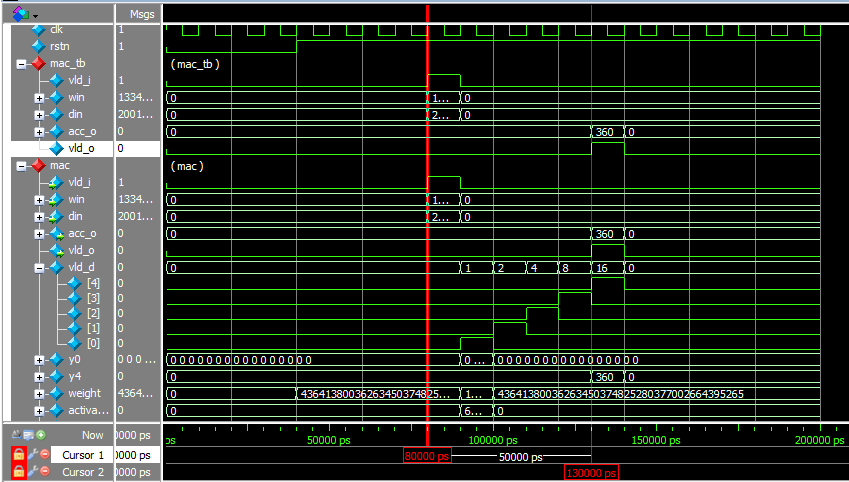


Fig. 7-1: Captured waveform of mark\_kern\_tb, mac\_kern, and mac.

1. (2p) Explain why the final result is 360.

Answer: The all weights are 3 and the activations are [0,1,2,…,15], and the final result equals

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

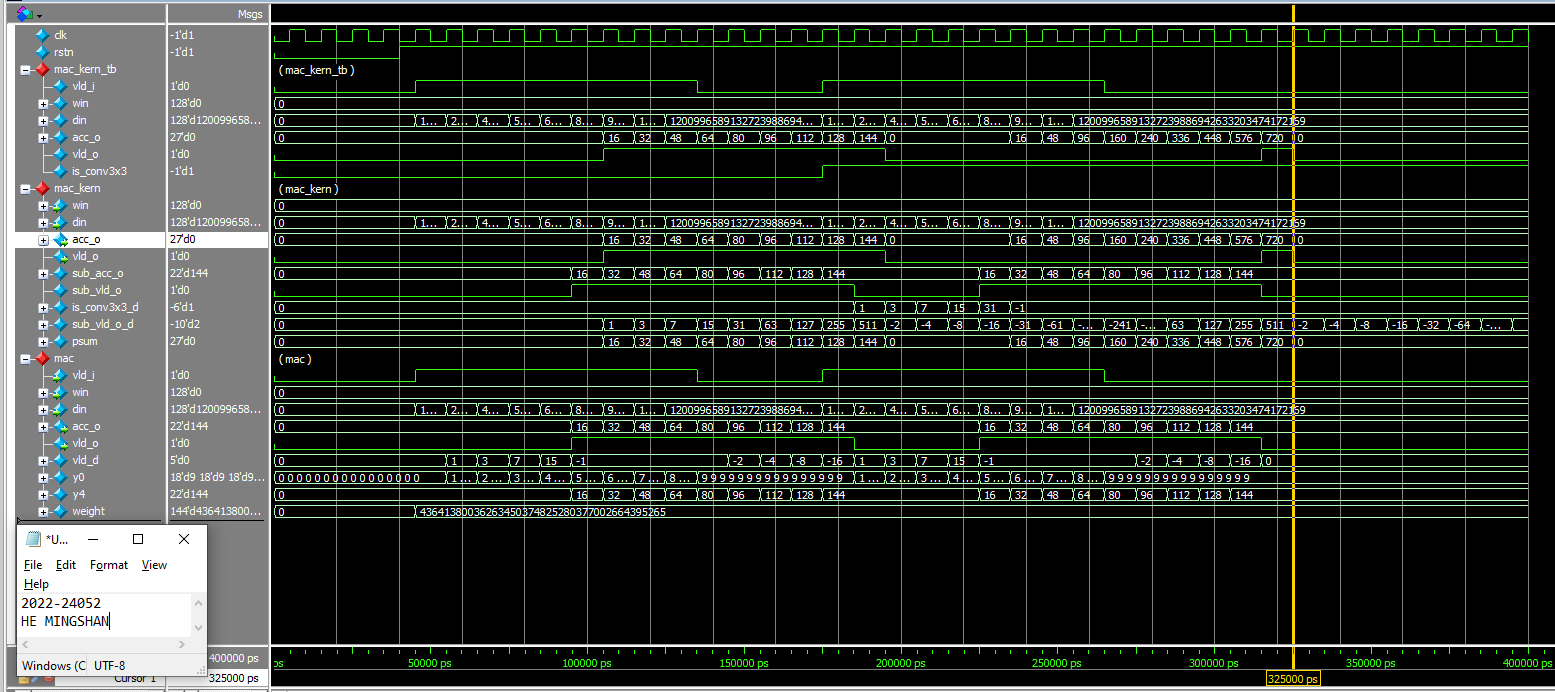
Because the `vld\_i` equals the `vld\_o[4]`, which 4 is the clog2(16). And the `vld\_o` is defined the delay step by step with `vld\_o[0] = vld\_i`, `vld\_o[1] = vld\_o[0]`, and `vld\_o[2] = vld\_o[1]`, Thus the output valid signal delays after delays 5 cycles the input valid signal, also means the y0, y1, y2, y3,y4 are all computed successfully.

**Problem 8 (15p): 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 7.
2. Complete the missing codes in mac\_kern.v.
3. Do a simulation and capture the waveform.



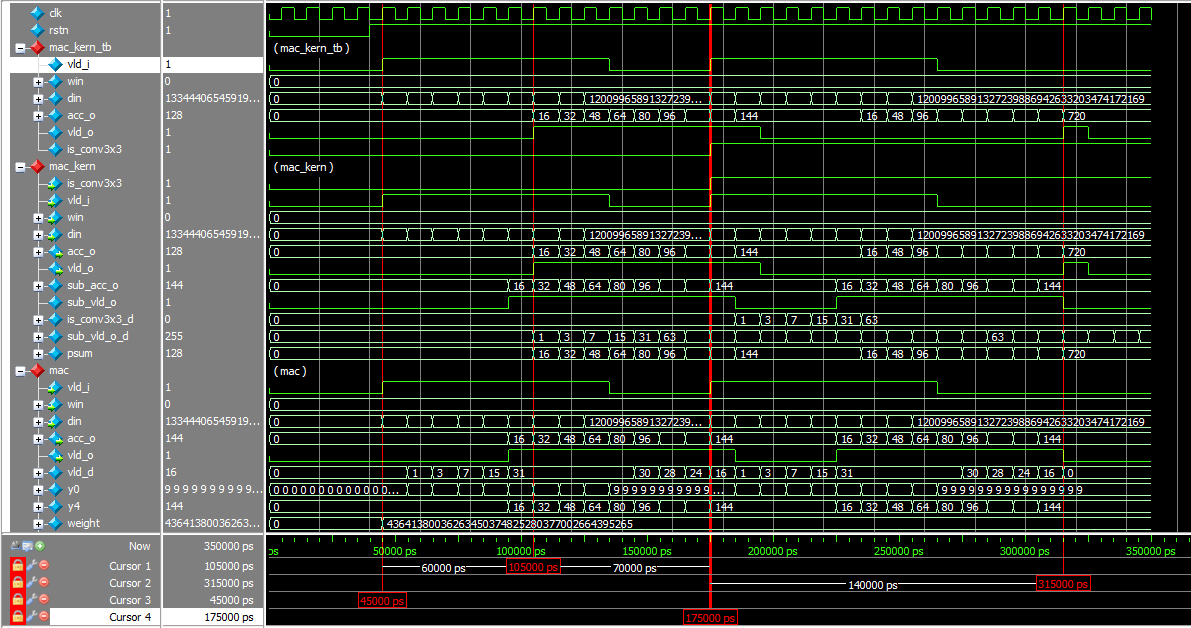


Fig. 8-1: Captured waveform of mark\_kern\_tb, mac\_kern, and mac.

1. (2p) Explain the values of the output port acc\_o.

Answer: First the test are divided by the filter. The first test is used conv 1\*1 filter and the whole `acc\_o` are the same from the value y4; The second test is used conv 3\*3 filter which means after the `mac\_kern` and the `mac\_kern\_tb`’s output `acc\_o` is the accumulation of the `mac` ‘s output. Thus the second `acc\_o` of the `mac\_kern\_tb` and `mac\_kern` is the sum of the `acc\_o` of the `mac`.

1. (2p) According to 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.

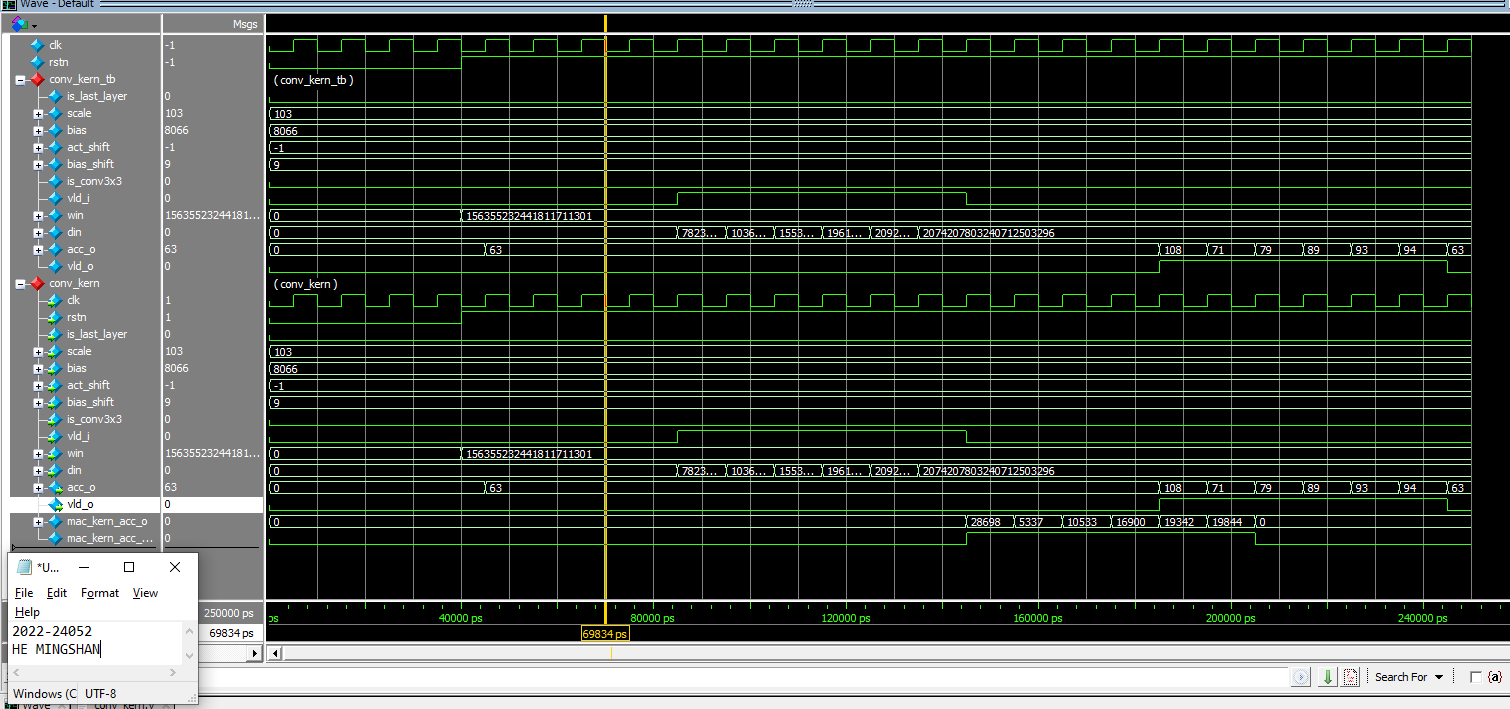
Answer: The first test is used conv 181 filter and the delay cycles are the same with the above problem are 5 cycles, but the `vld\_o` is get the value from the `sub\_vld\_o`, which makes the more 1 cycles. Thus between the `vld\_i` and `vld\_o` has the 6 cycles delay. About the second test which used conv 3\*3 filter, it will compute 9 pixel and it makes the more 8 cycles which exclude the 6 cycles for the first pixel. Thus between the `vld\_i` and `vld\_o` has the 6+8 = 14 cycles delay.

**Problem 9 (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 7 and 8.
2. Complete the missing codes in bias\_shifter.v, act\_shifter.v and conv\_kern\_tb.v.
3. Do simulation and capture the waveform.



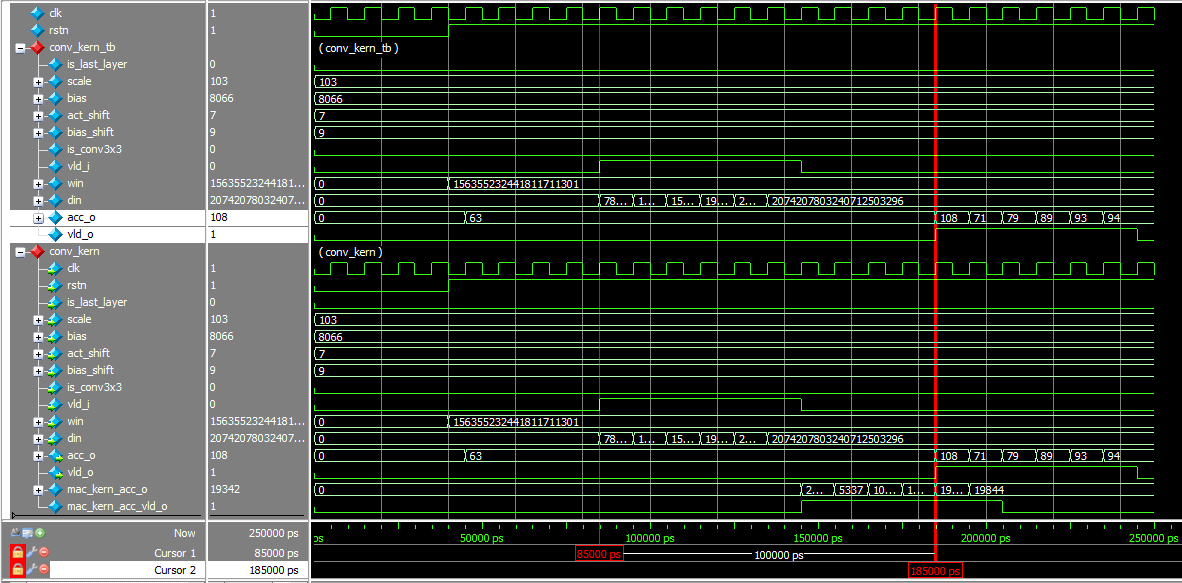


Fig. 9-1: Simulation results.

**Problem 10: (Optional) Sliding window (5p)**

Let H and W be the height and the weight of the input image. In our case, H = 128 and W = 128. Assume that 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]. Similar to 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

If (h ==0 && w ==0) {din[0]=0;din[1]=0;din[2]=0;

din[3]=0;din[4]=in\_img[h][w];din[5]=in\_img[h][w+1];

din[6]=0;din[7]=in\_img[h+1][w];din[8]=in\_img[h+1][w+1];}

else if (h ==0 && w ==W-1) {din[0]=0;din[1]=0;din[2]=0;

din[3]=in\_img[h][w-1];din[4]=in\_img[h][w];din[5]=0;

din[6]=in\_img[h+1][w-1];din[7]=in\_img[h+1][w]; din[8]=0;}

else if (h ==H-1 && w ==0) {din[0]=0;din[1]= in\_img[h-1][w];din[2]= in\_img[h-1][w+1];

din[3]=0;din[4]=in\_img[h][w];din[5]=in\_img[h][w+1];

din[6]=0;din[7]=0;din[8]=0;}

else if (h ==H-1 && w ==W-1) {din[0]= in\_img[h-1][w-1];din[1]= in\_img[h-1][w];din[2]=0;

din[3]= in\_img[h][w-1];;din[4]=in\_img[h][w];din[5]=0;

din[6]=0;din[7]=0;din[8]=0;}

If((h==0)&&(w>0||w<W-1)){ din[0]=0;din[1]=0;din[2]=0;

din[3]= in\_img[h][w-1];din[4]=in\_img[h][w];din[5]=in\_img[h][w+1];

din[6]= in\_img[h+1][w-1];din[7]=in\_img[h+1][w];din[8]=in\_img[h+1][w+1];}

Else if((h==H-1)&&(w>0||w<W-1)) {

din[0]= in\_img[h-1][w-1];din[1]=in\_img[h-1][w];din[2]=in\_img[h-1][w+1];

din[3]= in\_img[h][w-1];din[4]=in\_img[h][w];din[5]=in\_img[h][w+1];

din[6]=0;din[7]=0;din[8]=0; }

If((w==0)&&(h>0||h<H-1)){

din[0]=0;din[1]=in\_img[h-1][w];din[2]=in\_img[h-1][w+1];

din[3]=0;din[4]=in\_img[h][w];din[5]=in\_img[h][w+1];

din[6]=0;din[7]= in\_img[h+1][w];din[8]= in\_img[h+1][w+1];

}

Else if((w==W-1)&&(h>0||h<H-1)){

din[0]=in\_img[h-1][w-1];din[1]=in\_img[h-1][w];din[2]=0;

din[3]=in\_img[h][w-1];din[4]=in\_img[h][w];din[5]=0;

din[6]=in\_img[h-1][w-1];din[7]= in\_img[h+1][w];din[8]=0;

}

If((h>0||h<H-1)&&(w>0||w<W-1)){

din[0]=in\_img[h-1][w-1];din[1]=in\_img[h-1][w];din[2]=in\_img[h-1][w+1];

din[3]=in\_img[h][w-1];din[4]=in\_img[h][w];din[5]=in\_img[h][w+1];

din[6]=in\_img[h-1][w-1];din[7]= in\_img[h+1][w];din[8]=in\_img[h+1][w+1];

}

//}}}

// 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