Project: Optimizing a CNN Accelerator for Image Super Resolution

LABORDE-TASTET Antonin & Satyam

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:

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

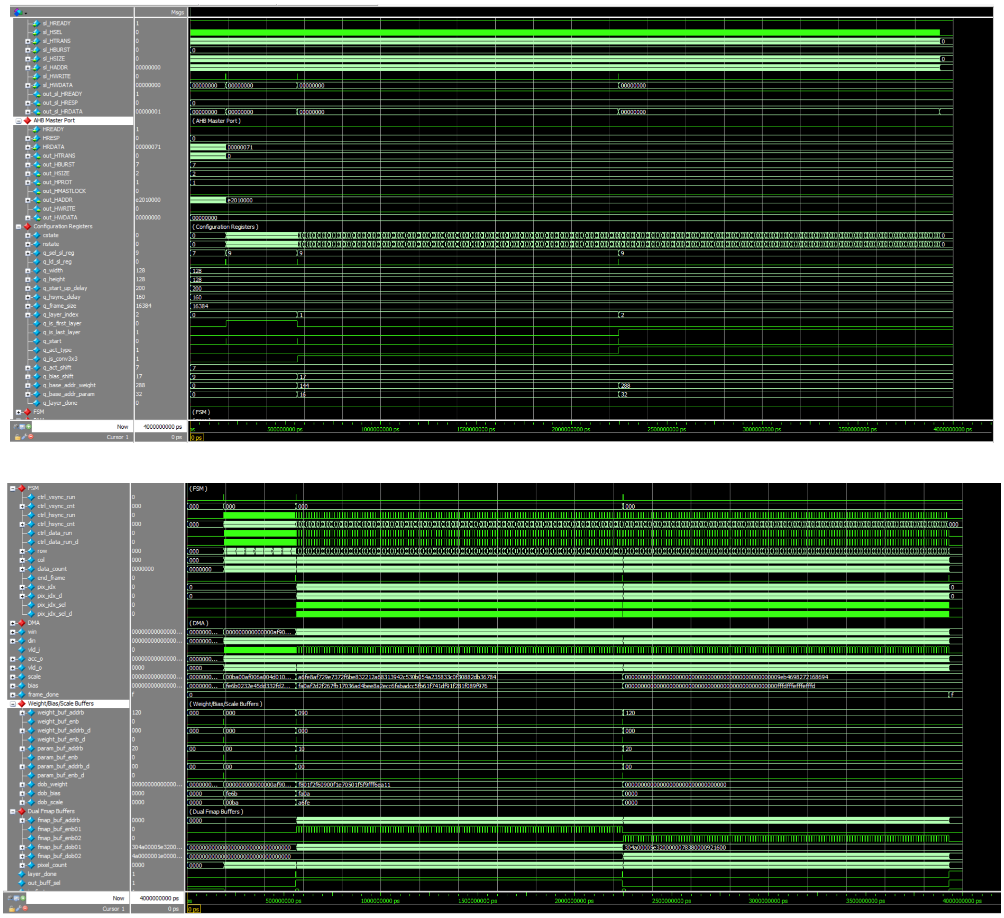
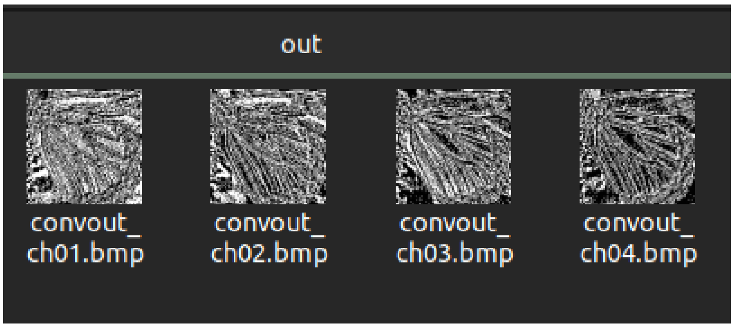


Figure 1-1: A captured waveform of cnn_accel.

b. Use check_hardware_results.m to verify the output images generated by the H/W simulation.



Fenêtre de Commandes

```
Results of the channel 01 are same!
Results of the channel 02 are same!
Results of the channel 03 are same!
Results of the channel 04 are same!
>>
```

2. 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:

a. 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	0	0	1	1	1	1	0	1
bias_shift	16	22	23	23	23	23	23	24
act_shift	1	1	1	1	1	1	0	0

b. 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.

```
%% Save the CNN model
n_layers = size(test_vector,1)-2;
bitwidth = 128;
Ti = 16;    % A CONV kernel computes 16 products at the same time (conv_kern.v)
To = 16;    % Run 16 CONV kernels at the same time (cnn_accel.v)

fid_all_weights     = fopen(sprintf('%s/%s/all_conv_weights.hex',outdir,model_name),'wt');
fid_all_biases      = fopen(sprintf('%s/%s/all_conv_biases.hex',outdir,model_name),'wt');
fid_all_scales      = fopen(sprintf('%s/%s/all_conv_scales.hex',outdir,model_name),'wt');

fprintf('The model is %s !!!\n\n', model_name);
for i = 1:n_layers
    fprintf('Exporting layer %d .......... ',i);
    conv_weights     = test_vector{i,2};
    conv_biases      = test_vector{i,3};
    conv_scales      = test_vector{i,4};
    conv_output      = test_vector{i,7};

    fid_weights     = fopen(sprintf('%s/%s/conv_weights_L%d.hex',outdir,model_name,i),'wt');
    fid_biases      = fopen(sprintf('%s/%s/conv_biases_L%d.hex',outdir,model_name,i),'wt');
    fid_scales      = fopen(sprintf('%s/%s/conv_scales_L%d.hex',outdir,model_name,i),'wt');
    fid_convout     = fopen(sprintf('%s/%s/convout_L%d.hex',outdir,model_name,i),'wt');
```

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

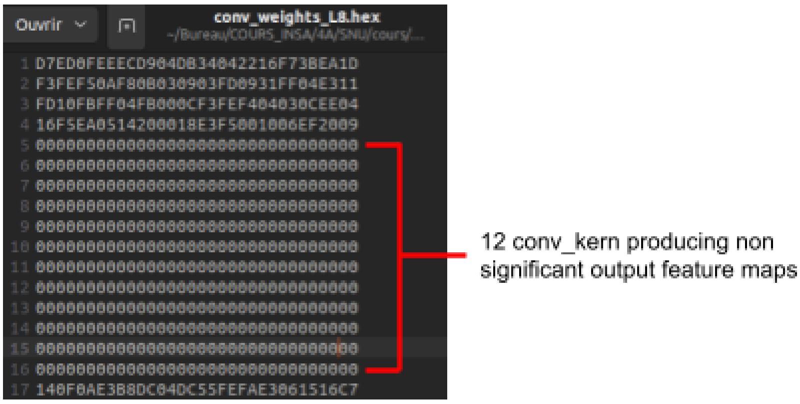
	No. of bit	Number of lines	Buffer Size in total

	per line	#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.

These values are **not optimal**.

As we can see with the weight configuration in the hex file for the last layer (Layer 8):



With this configuration, each 12 of the 16 conv_kern will produce useless outputs -> 0 because for this layer we only need to produce 4 output feature maps. Thus, these 12 conv_kern could be used to compute other pixels. Hence, in the **conv_weights_L8.hex** file, we could fill these ‘000… 00’ gap with other pixels’ weight and **reduce its size from 144 to 36**.

For conv_weights_L1.hex, similarly, the configuration is 16 words of 9 weight which is not optimal, a solution could be to set the weight file as follows: to 9 words of 16 weight. => Fully utilized.

3. **(20p) Sub-pixel layer (See Lecture 14 for the detailed description)**

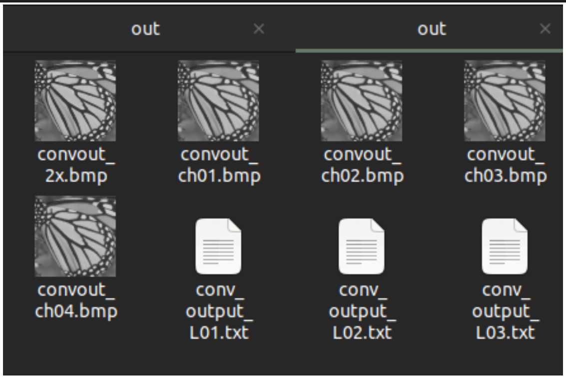
- Implement the code in accel_cnn.v to handle **the sub-pixel layer**. Basically, four pixels are generated at the sub-pixel layer. Each of the four pixels must be added with the corresponding input.

```
reg [7:0] rec1;
reg [7:0] rec2;
reg [7:0] rec3;
reg [7:0] rec4;

reg [8:0] px;
```

```
reg [8:0] pr1;
reg [8:0] pr2;
reg [8:0] pr3;
reg [8:0] pr4;

always@(*) begin
   if(q_is_last_layer && !layer_done) begin
       px = {1'b0,in_img[pixel_count]};
       pr1 = {1'b0,acc_o[7:0]};
       pr2 = {1'b0,acc_o[15:8]};
       pr3 = {1'b0,acc_o[15:8]};
       pr4 = {1'b0,acc_o[15:8]};

       if((px+pr1) & 9'b100000000)
           rec1 = 8'd255;
       else
           rec1 = px+pr1;
       if((px+pr2) & 9'b100000000)
           rec2 = 8'd255;
       else
           rec2 = px+pr2;
       if((px+pr3) & 9'b100000000)
           rec3 = 8'd255;
       else
           rec3 = px+pr3;
       if((px+pr4) & 9'b100000000)
           rec4 = 8'd255;
       else
           rec4 = px+pr4;
   end
end

assign out_pixel = {rec4,rec3,rec2,rec1};
```

- Make a new BMP writer that captures a 32-bit input and writes the high-resolution image.

```
out_img[2*row     * (2*WIDTH) + (2*col   )] <= din[7:0];     /*Your code*/
out_img[2*row     * (2*WIDTH) + (2*col+1)] <= din[15:8];    /*Your code*/
out_img[(2*row+1) * (2*WIDTH) + (2*col   )] <= din[23:16];   /*Your code*/
out_img[(2*row+1) * (2*WIDTH) + (2*col+1)] <= din[31:24];   /*Your code*/
```



4. (30p) H/W simulation and verification for SSAI2021

What you have to do:

a. Copy the weight/scale/bias hex files of SSAI2021 from Part 2b to input_data/

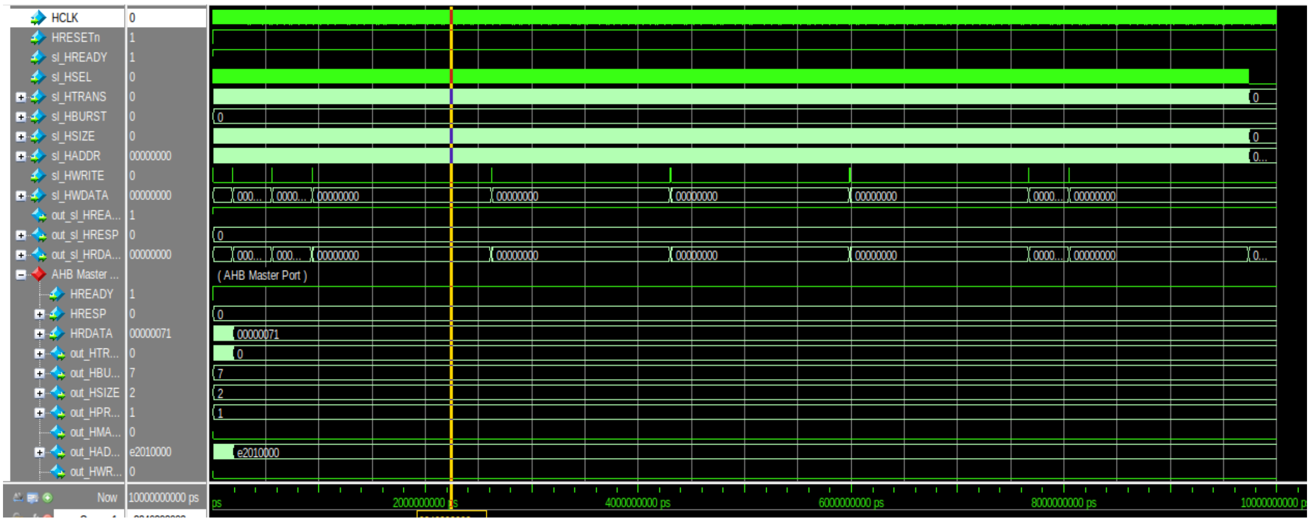
b. Update the **buffer** parameters for SSAI2021 in the top file (cnn_accel.v) and the test bench (top_system_tb.v).

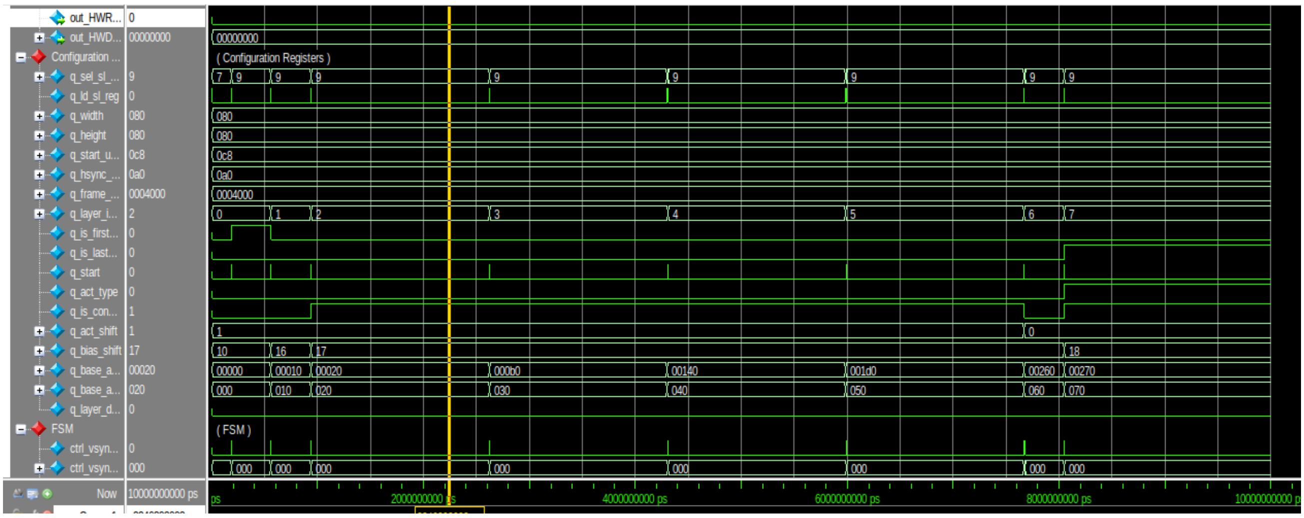
   Hint: Only changing the number of layers may work.

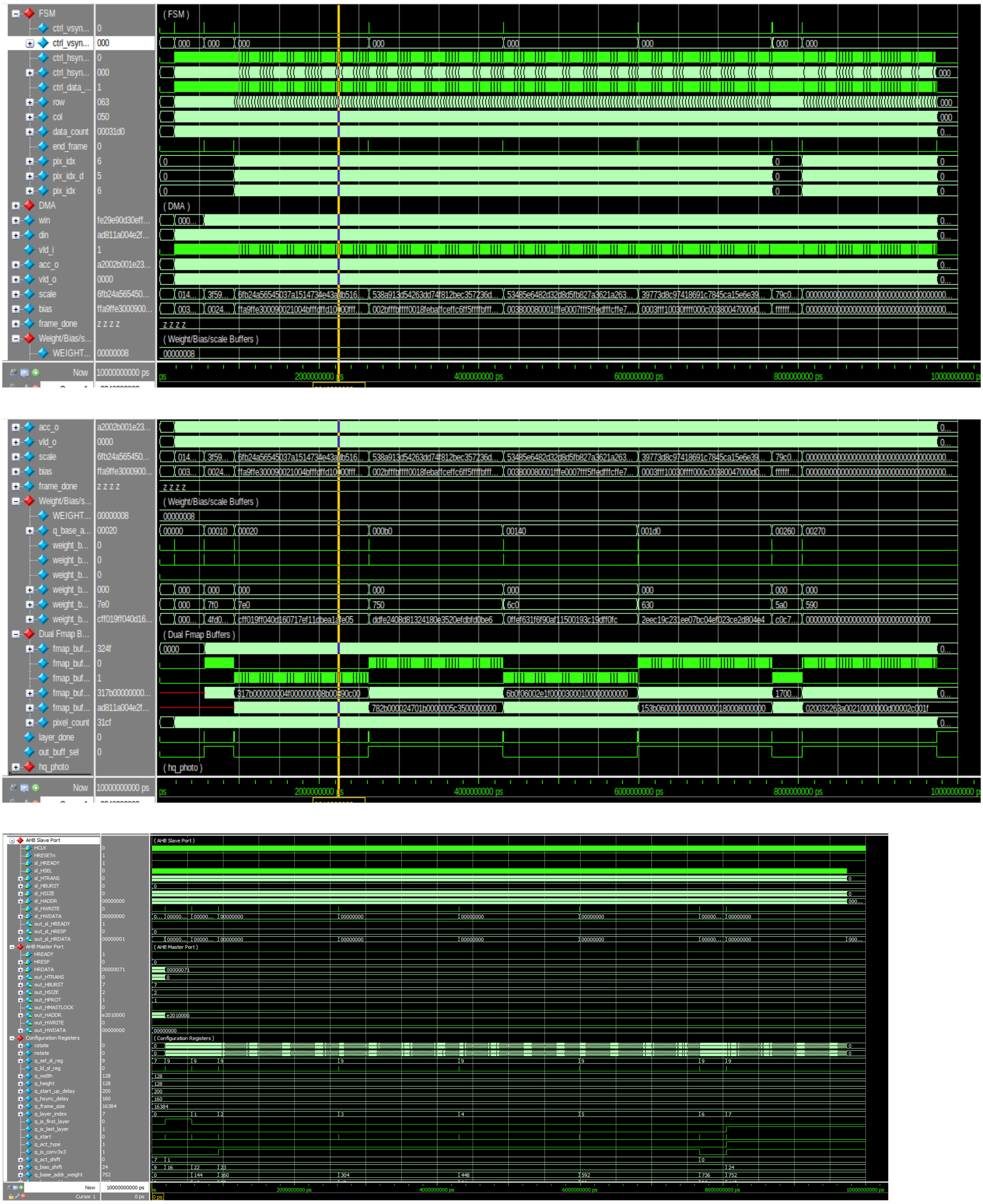
c. Modify the test bench (top_system_tb.v) to execute SSAI2021 on the CNN accelerator

   Hint: Use the parameters in Part 2a.

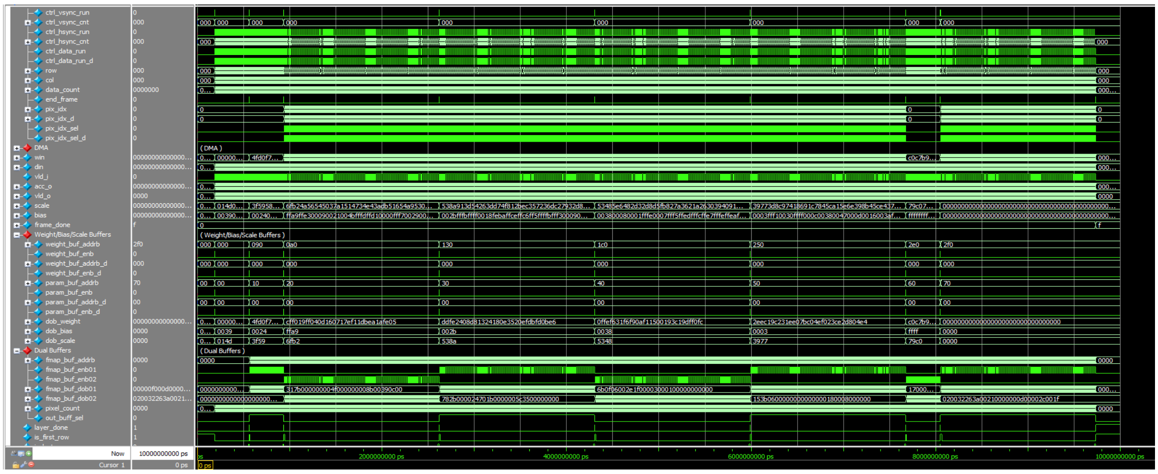
d. Do a simulation with time = 10ms and capture the waveform.






FSM
ctrl_vsyn...
ctrl_hsyn...
ctrl_data_...
row
col
data_count
end_frame
pix_idx
pix_idx_d
DMA
win
din
vld_i
acc_o
vld_o
scale
bias
frame_done
Weight/Bias/s...
WEIGHT...
Now
10000000000 ps
2000000000 ps
4000000000 ps
6000000000 ps
8000000000 ps
(Weight/Bias/scale Buffers)
q_base_a...
weight_b...
Dual Fmap B...
fmap_buf...
pixel_count
layer_done
out_buff_sel
hq_photo
(Dual Fmap Buffers)
(hq_photo)

(a)



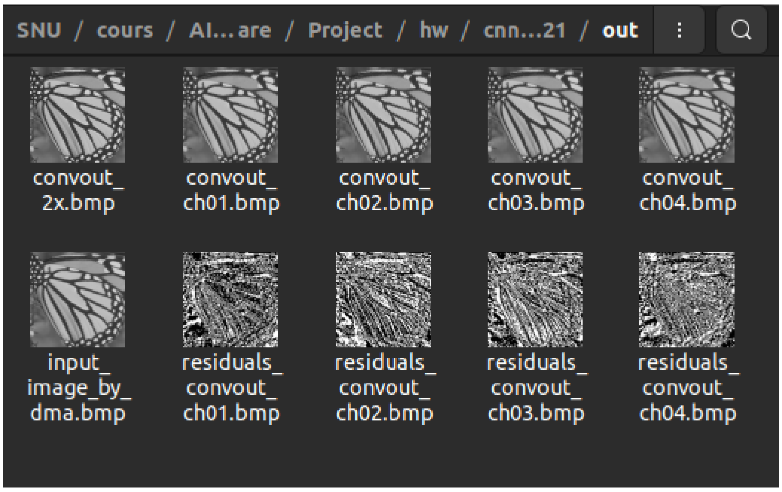
(b)

Figure 1-3: Figure 1-1: A captured waveform of cnn_accel.

Hints:

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

e. Use check_hardware_results.m to verify the output images generated by the H/W simulation.



Fenêtre de Commandes

```
Results of the channel 01 are same!
Results of the channel 02 are same!
Results of the channel 03 are same!
Results of the channel 04 are same!
>>
```

```
//// Debugging
//integer fp_output_L01;
//integer fp_output_L02;
//integer fp_output_L03;
//
//integer idx;
//always @(posedge clk or negedge rstn) begin
//  if(~rstn) begin
//      fp_output_L01= $fopen("out/conv_output_L01.txt", "w");
//      fp_output_L02= $fopen("out/conv_output_L02.txt", "w");
//      fp_output_L03= $fopen("out/conv_output_L03.txt", "w");
//      idx <= 0;
//  end
//  else begin
//      if(vld_o[0]) begin
//          for(idx = To*ACT_BITS/4-1; idx >= 0; idx=idx-1) begin
//              if(idx == 0) begin
//                  case(q_layer_index)
//                      3'd0: $fwrite(fp_output_L01,"%01h\n", acc_o[idx*4+:4]);
//                      3'd1: $fwrite(fp_output_L02,"%01h\n", acc_o[idx*4+:4]);
//                      3'd2: $fwrite(fp_output_L03,"%01h\n", acc_o[idx*4+:4]);
//                  endcase
//              end
//              else begin
//                  case(q_layer_index)
//                      3'd0: $fwrite(fp_output_L01,"%01h", acc_o[idx*4+:4]);
//                      3'd1: $fwrite(fp_output_L02,"%01h", acc_o[idx*4+:4]);
//                      3'd2: $fwrite(fp_output_L03,"%01h", acc_o[idx*4+:4]);
//                  endcase
//              end
//          end
//      end
//  end
//end
```

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

Problem 2 (100p): Optimization

1. **Optimization (90p)**

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

a. **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.

b. 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.

DMA Optimization:

```
        if(data_vld_o_ld) begin
            if(in_pixel_count == q_frame_size-1)
                in_pixel_count <= 0;
            else
                in_pixel_count <= in_pixel_count + 4;      // modif
        end
```

```
        if(data_vld_o_ld) begin
            in_img[in_pixel_count] <= data_o_ld[7:0];
            in_img[in_pixel_count+1] <= data_o_ld[15:8];// modif
            in_img[in_pixel_count+2] <= data_o_ld[23:16];// modif
            in_img[in_pixel_count+3] <= data_o_ld[31:24];// modif
        end
```

With the re-ordered file, we can load the pixel 4 by 4 instead of 1 by 1.

```
            if(data_last_o_ld) begin    // End of loading a line
                if(start_line_ld == q_height-4) begin   // Last line   //
                    start_line_ld <= 0;
                    start_addr_ld <= q_input_image_base_addr;
                    load_image_done <= 1'b1;
                end
                else begin  // Normal line
                    start_line_ld <= start_line_ld + 4;  // modif
                    start_addr_ld <= start_addr_ld + {q_width,2'b00};
                end
            end
```

cnn_accel.v

```
   while(!image_load_done) begin
                                           #(32*p)       @(posedge       HCLK)
u_top_system.u_riscv_dummy.task_AHBread(`CNN_ACCEL_INPUT_IMAGE_LOAD,image_loa
d_done);  // modif
   end
```

top_system_tb.v

- Fully utilizing the convolution kernels when executing Layer 3.
- Applying layer fusion

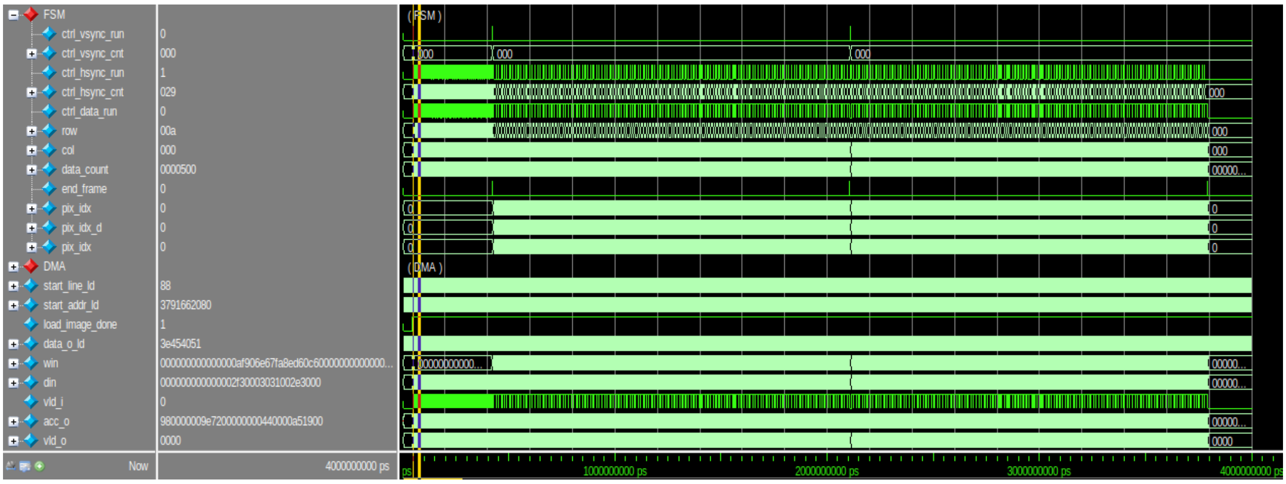
You should describe your modified code in the report.

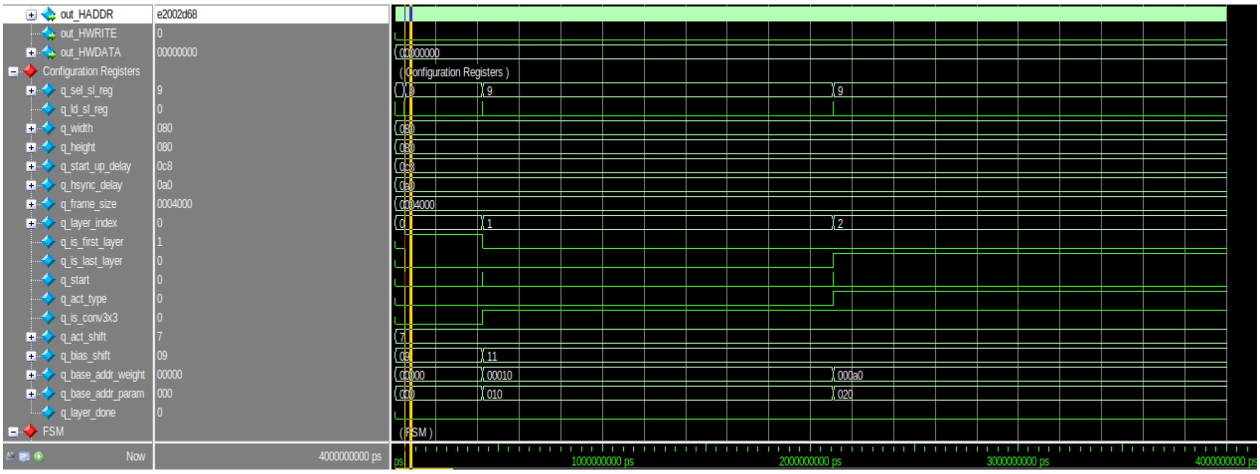
c. Evaluation

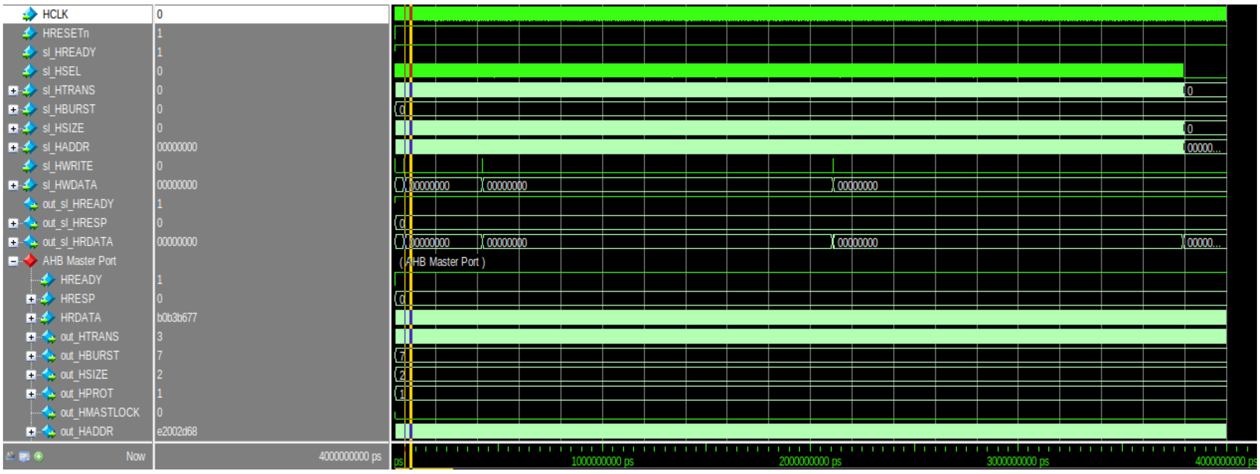
- Use check_hardware_results.m to verify the output images generated by H/W simulation. Please make sure that your optimized code functions correctly as the baseline.

- Report the execution time (t) and the buffer size (s) of your design. The overall improvement is measured by the following metric:

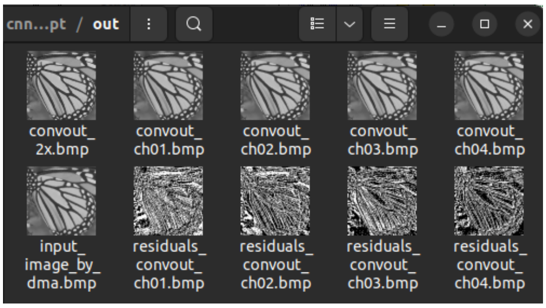
Overall improvement in time was not significant but **DMA's time reduced four fold. (From 188us to 47us)** and total running time from 3,930 to 3,790 us








```
# T=46685 ns: FAST loading the image by DMA is DONE!!!
#
# T=421765 ns: Layer 1 done!!!
#
# T=2106365 ns: Layer 2 done!!!
#
# T=3790965 ns: Layer 3 done!!!
#
```



Fenêtre de Commandes

```
Results of the channel 01 are same!
Results of the channel 02 are same!
Results of the channel 03 are same!
Results of the channel 04 are same!
>>
```

$$S_{overall} = \left(\frac{t0}{t}\right) \times \left(\frac{s0}{s}\right)$$

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

$$so\ =\ s\ ;\ S_{overall}\ =\ \left(\frac{to}{t}\right)\ =\frac{3{,}930}{3{,}790}=\ 1.04$$

2. 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*.