Project: CNN Accelerator for Image Super Resolution

Issued: January 23 (Tuesday), 2023 **Due:** **1 pm, January 31 (Tuesday), 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 (100p): CNN Accelerator for SR

1. Baseline code (10p)

What you have to do:

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

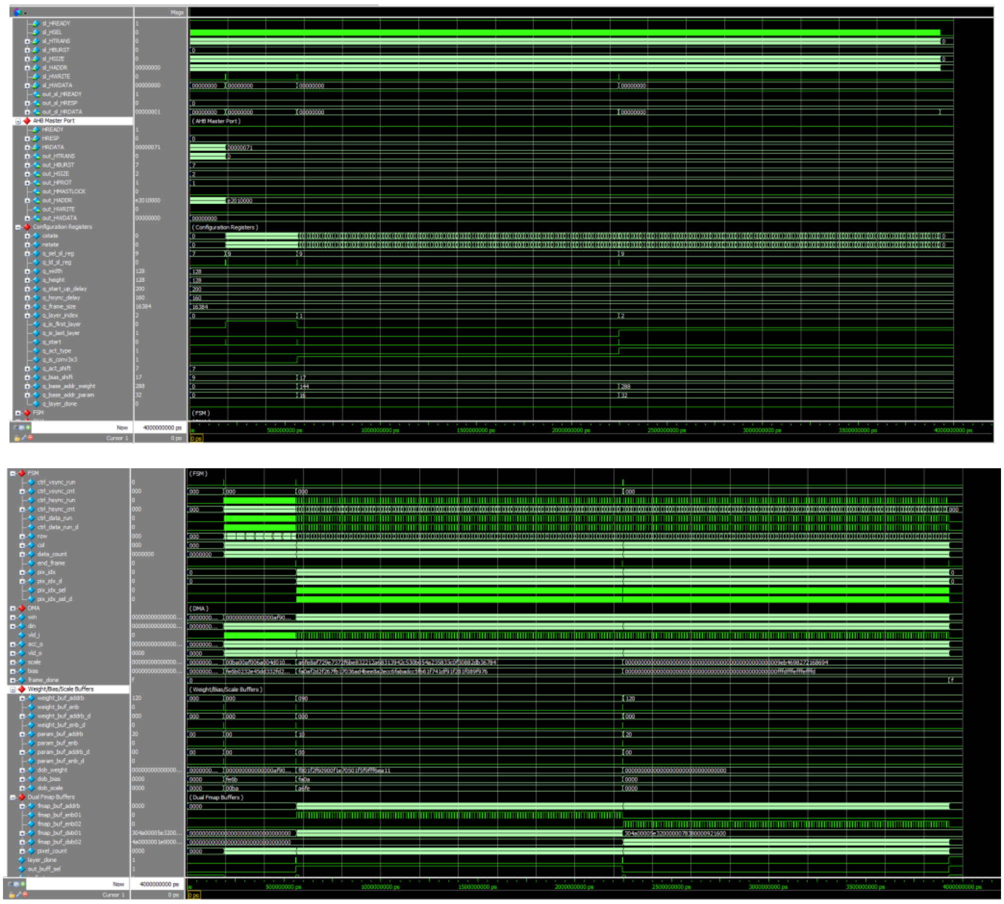


Figure 1-1: A captured waveform of cnn_accel.

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

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								
q_is_last_layer								
q_is_conv3x3								
bias_shift								
act_shift								

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

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

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

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/

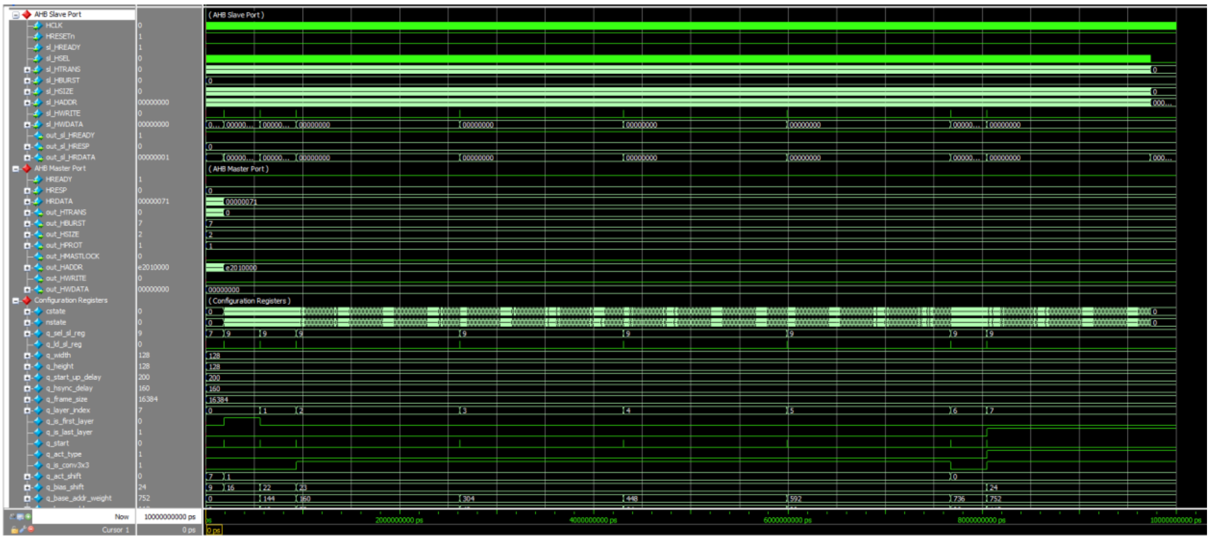
b. Update the **buffer** parameters for SSAI2021 in the top file (cnn_accel.v) and the test bench (top_system_tb.v).

   Hint: Only changing the number of layers may work.

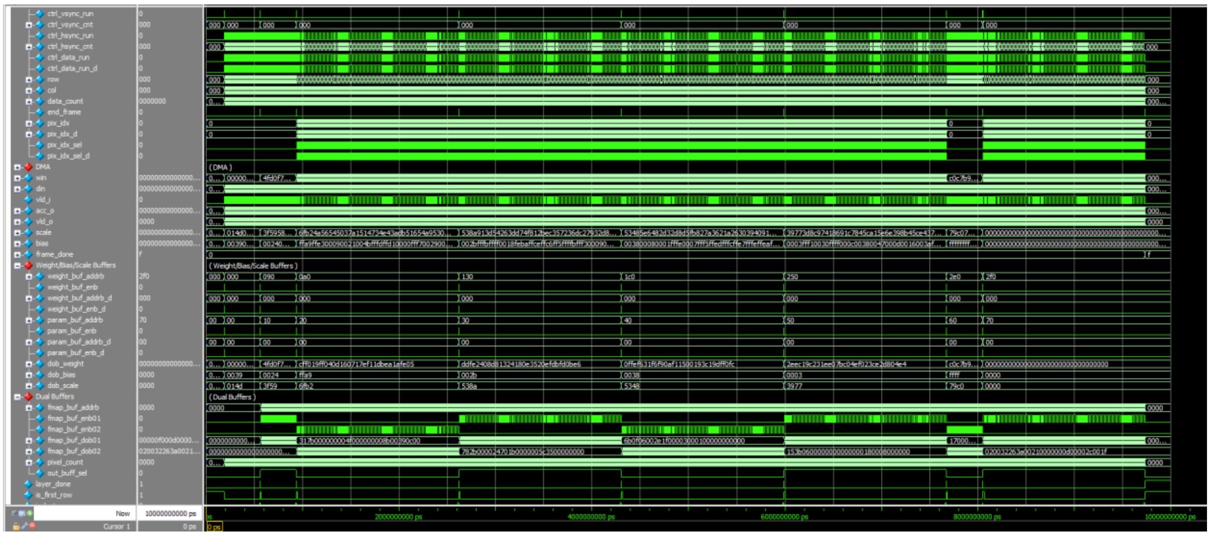
c. Modify the test bench (top_system_tb.v) to execute SSAI2021 on the CNN accelerator

   Hint: Use the parameters in Part 2a.

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



(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 early verify the outputs of some first CONV layers.

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

```
//// 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.
- 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:

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

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

2. Optimality analysis (10p)

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