Computer-Aided VLSI System Design
Homework 3: Simple Convolution and Image Processing Engine

TA: 張力元 r11943006@ntu.edu.tw
TA: 蔡宇軒 f07943171@ntu.edu.tw

Due Tuesday, Apr. 25, 13:59

Data Preparation

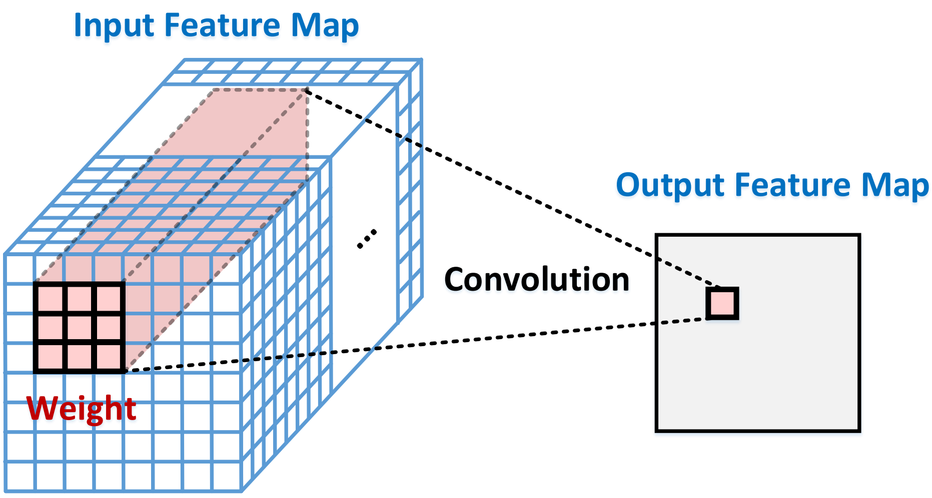
1. Decompress 1112_hw3.tar with following command

```
tar -xvf 1112_hw3.tar
```

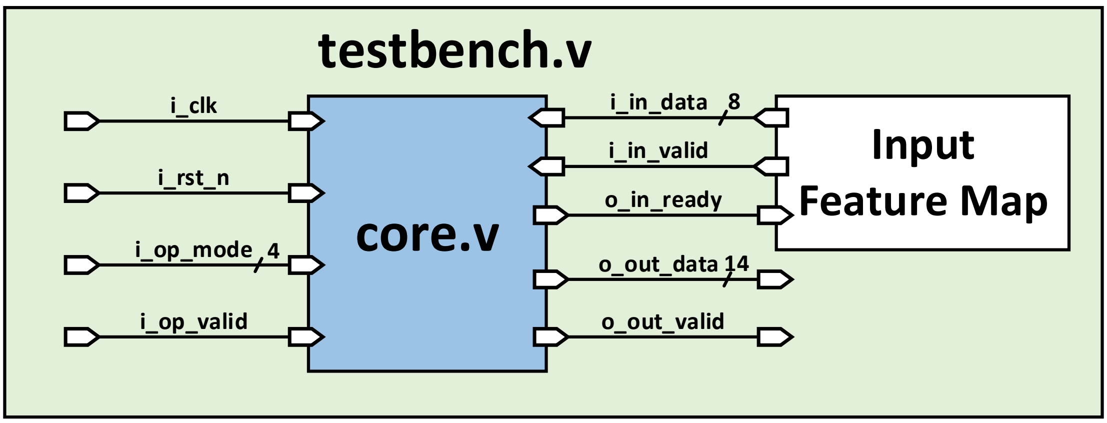
Folder	File	Description
00_TESTBED	testbench_temp.v	Testbench template
00_TESTBED/ PATTERN/	indata*.dat	Input image data
	opmode*.dat	Pattern of operation mode
	golden*.dat	Golden data of output
01_RTL	core.v	Your design
	rtl_01.f	File list for rtl simulation
	01_run	NCVerilog command
	99_clean_up	Command to clean temporary data
02_SYN	syn.tcl	Script for synthesis
	core_dc.sdc	Constraint file for synthesis
	02_run.dc	Command for DC
03_GATE	rtl_03.f	File list for gate-level simulation
	03_run	NCVerilog command for gate-level simulation
	99_clean_up	Command to clean temporary data
sram_****x8	sram_****x8.v	SRAM design file
	sram_****x8_slow_syn.db	Synthesis model
	sram_****x8_slow_syn.lib	Timing and power model
	sram_****x8.pdf	Datasheet for SRAM
top	report.txt	Design report form

Introduction

In this homework, you are going to implement a simplified convolution and image processing engine. An 8×8×32 feature map will be loaded first, and it will be processed with several functions. If you are not familiar with convolution, refer to [1] for some illustrations.



Block Diagram



Specifications

1. Top module name: **core**
2. Input/output description:

Signal Name	I/O	Width	Simple Description
i_clk	I	1	Clock signal in the system.
i_rst_n	I	1	Active **low** asynchronous reset.
i_op_valid	I	1	This signal is **high** if operation mode is valid
i_op_mode	I	4	Operation mode for processing
o_op_ready	O	1	Set **high** if ready to get next operation
i_in_valid	I	1	This signal is **high** if input pixel data is valid
i_in_data	I	8	Input pixel data (**unsigned**)
o_in_ready	O	1	Set **high** if ready to get next input data (only valid for i_op_mode = 4'b0000)
o_out_valid	O	1	Set **high** with valid output data
o_out_data	O	14	Pixel data or image processing result (**signed**)

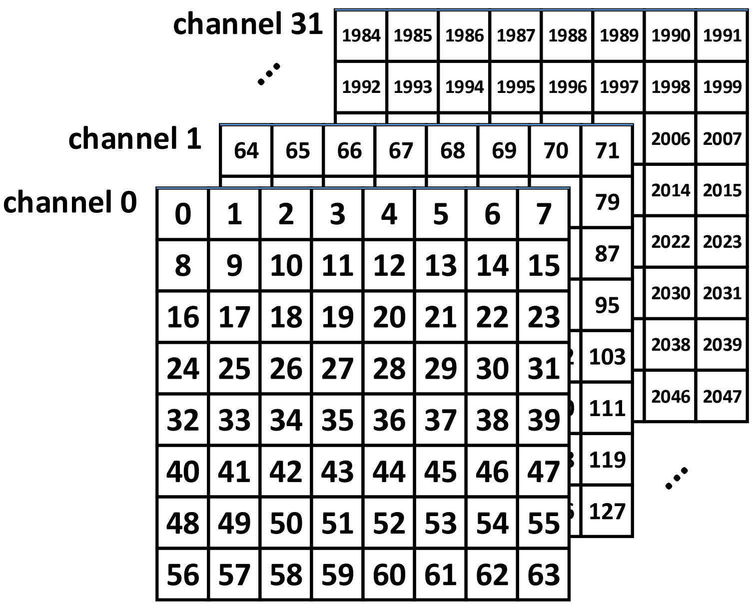
3. All inputs are synchronized with the **negative** edge clock.
4. All outputs should be synchronized at clock **rising** edge.
5. You should reset all your outputs when i_rst_n is **low**. Active low asynchronous reset is used and only once.
6. Operations are given by i_op_mode [3:0] when i_op_valid is **high**.
7. i_op_valid stays only **1 cycle**.
8. i_in_valid and o_op_ready can't be **high** in the same time.
9. i_op_valid and o_op_ready can't be **high** in the same time.
10. i_in_valid and o_out_valid can't be **high** in the same time.
11. i_op_valid and o_out_valid can't be **high** in the same time.
12. o_op_ready and o_out_valid can't be **high** in the same time.
13. Set o_op_ready to **high** to get next operation (only one cycle).
14. o_out_valid should be **high** for valid output results.
15. **At least one SRAM** is implemented in your design.
16. Only worst-case library is used for synthesis.
17. The synthesis result of data type should **NOT** include any **Latch**.
18. The slack for setup-time should be **non-negative**.
19. **No any timing violation and glitches** for the gate level simulation **after reset**.

Design Description

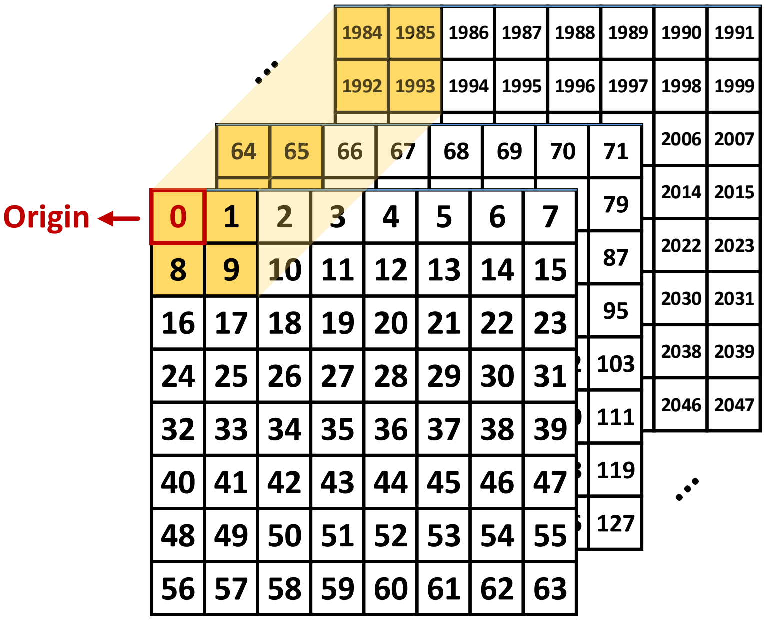
1. The followings are the operation modes you need to design for this homework:

i_op_mode	Meaning
4’b0000	Input feature map loading
4’b0001	Origin right shift
4’b0010	Origin left shift
4’b0011	Origin up shift
4’b0100	Origin down shift
4’b0101	Reduce the channel depth of the display region
4’b0110	Increase the channel depth of the display region
4’b0111	Output the pixels in the display region
4’b1000	Perform convolution in the display region
4’b1001	Median filter operation
4’b1010	Haar wavelet transform

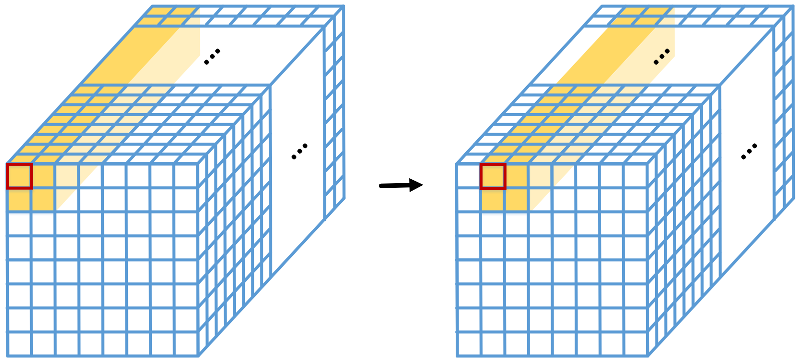
2. Input feature map loading:
   - An 8×8×32 feature map is loaded for 2048 cycles in **raster-scan** order.
   - The size of each pixel is 8 bits (unsigned).
   - Raise o_op_ready to 1 after loading all pixels.
   - If o_in_ready is 0, stop input data until o_in_ready is 1.
   - The input feature map will be loaded only once at the beginning.



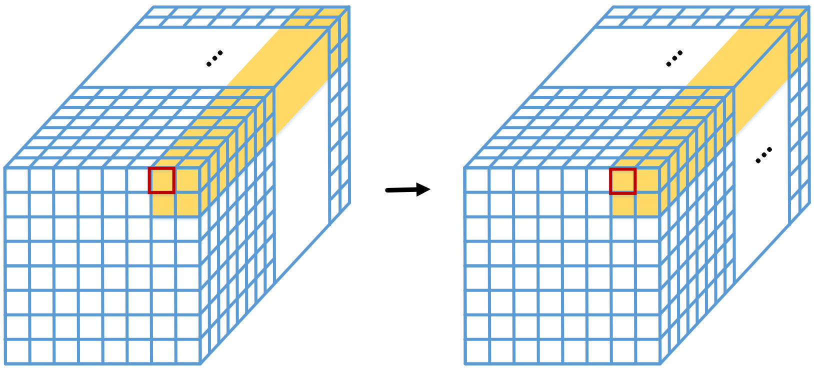
3. The first pixel in the display region is **origin**.
    - The default coordinate of the origin is at 0.
    - The size of the display region is 2×2×depth.



4. Origin shifting:
    - Ex. Origin right shift (i_op_mode = 4’b0001).



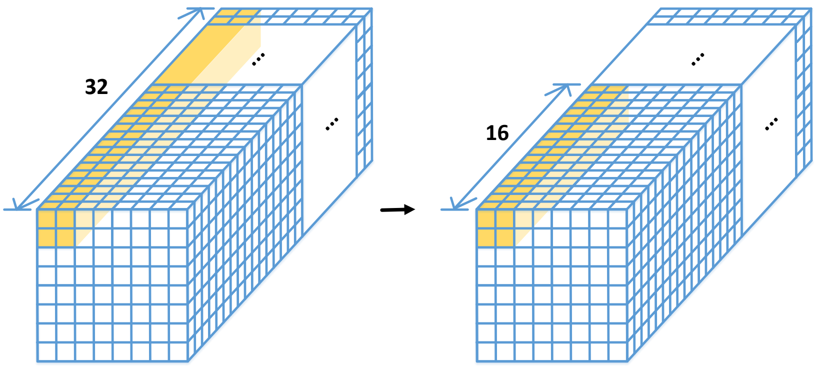
    - If output of display exceeds the boundary, retain the same origin point.



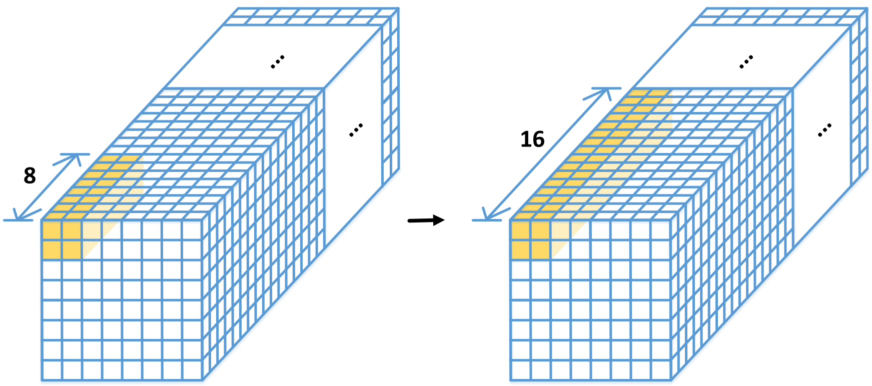
5. Channel depth:
   - 3 depths are considered in this design: 32, 16, and 8.
   - The display size will change according to different depth.

Depth	Display size
32	2 x 2 x 32
16	2 x 2 x 16
8	2 x 2 x 8

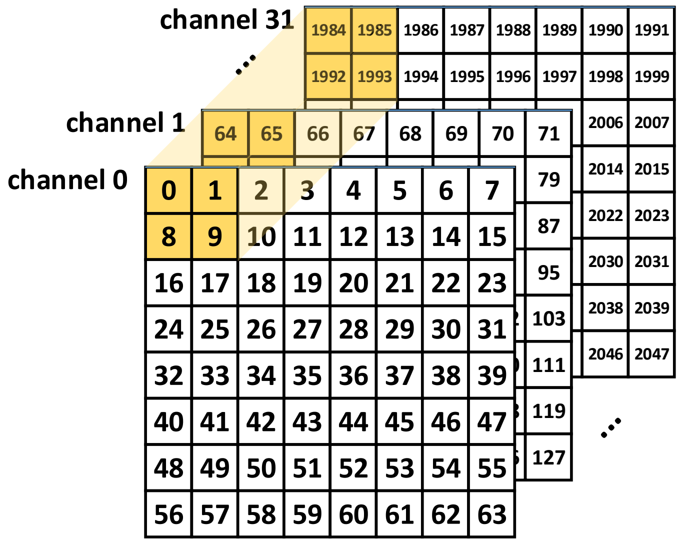
6. Scale-down:
   - Reduce the channel depth of the display region to next level.
     - Ex. For channel depth, 32 → 16 → 8
   - If the depth is 8, retain the same depth.



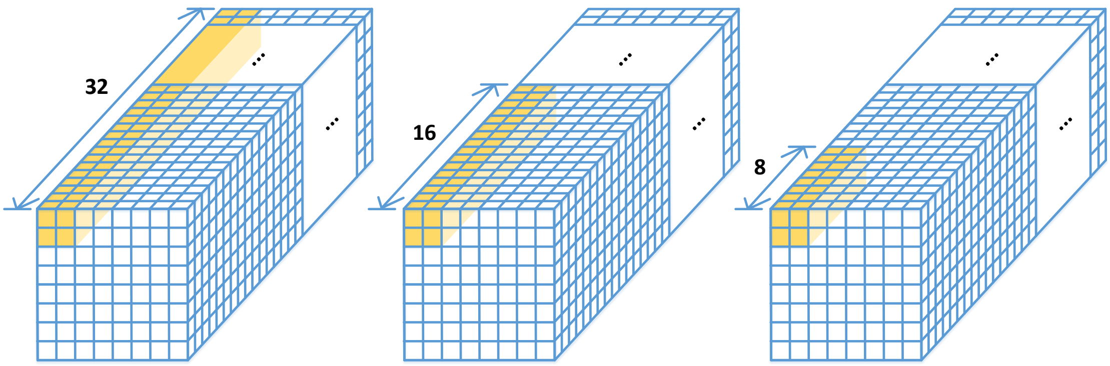
7. Scale-up:
   - Increase the channel depth of the display region to next level.
     - Ex. For channel depth, 8 → 16 → 32
   - If the depth is 32, retain the same depth.



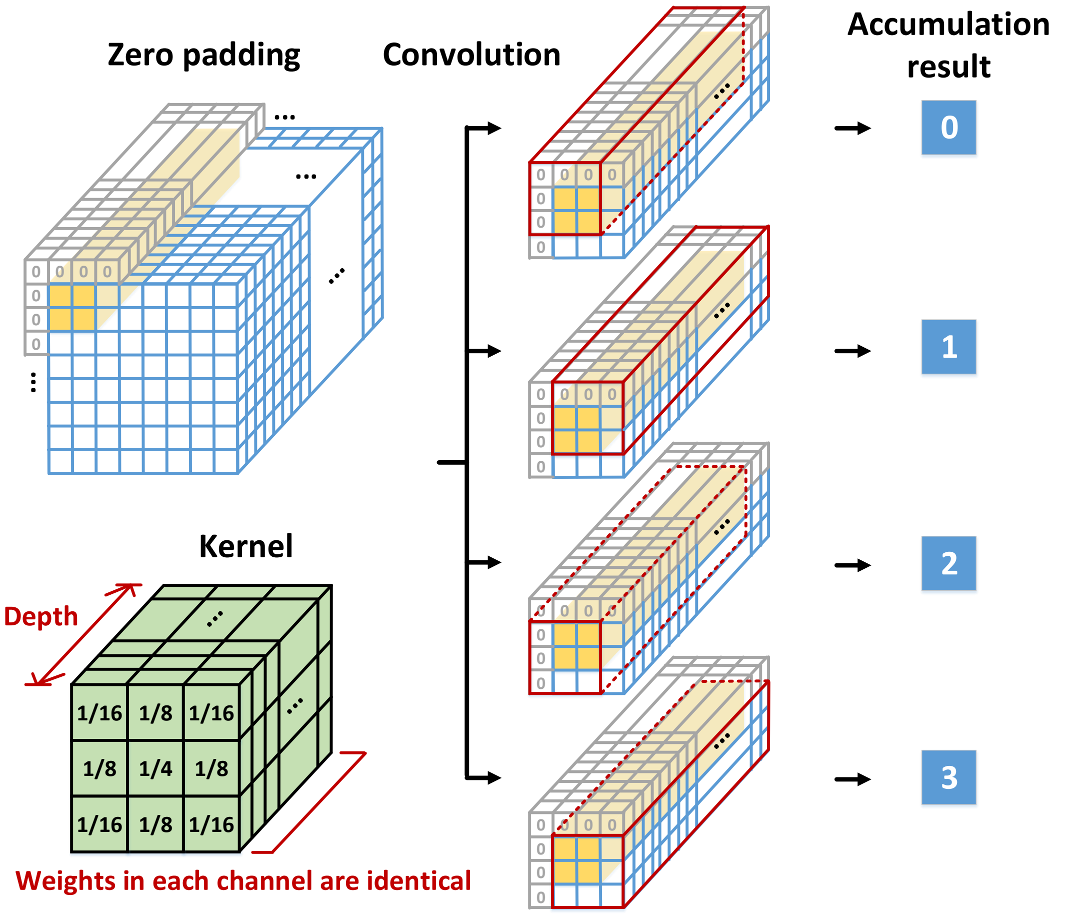
8. Display:
   - For this operation, you have to output the pixels in the display region.
   - Set **o_out_data [13:8]** to 0 and **o_out_data [7:0]** to the pixel data.
   - When i_op_mode = 4’b0111, the pixels are displayed in **raster-scan** order.
     (For example: 0 → 1 →8 → 9 → 64 → 65 → … → 1992 → 1993)



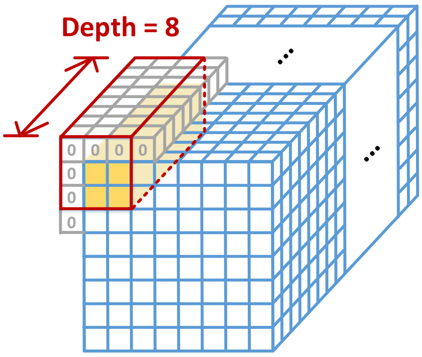
   - The size of display region changes according to the depth.



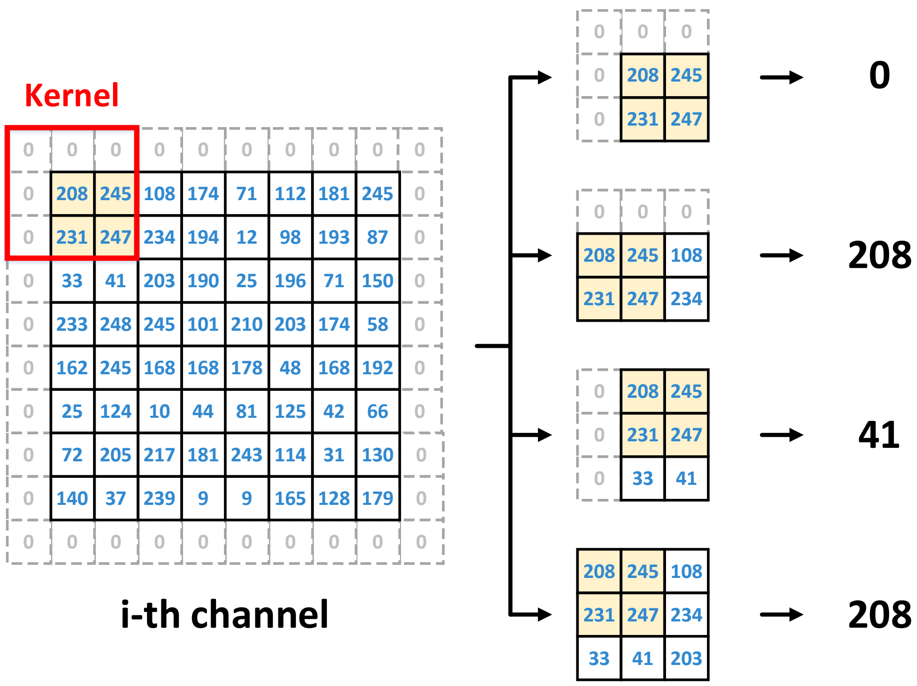
9. Convolution:
   - For this operation, you have to perform convolution **in the display region**.
   - The size of the kernel is 3×3×depth. The weights in each channel are identical.
   - The feature map needs to be zero-padded for convolution.
   - The accumulation results should be **rounded to the nearest integer** [2].
     - Do not truncate temporary results during computation.
   - After the convolution, you have to output the **4** accumulation results in **raster-scan** order.
   - The values of original pixels will not be changed.



   - The number of channels that are accumulated during convolution is determined by the depth. For example, accumulate 8 channels if the depth is 8.



10. Median filter operation:
    - For this operation, you have to perform median filtering **in the first 4 channels of the display region**.
    - The kernel size of the median filter is 3×3.
    - Perform median filtering on each channel **separately**.
    - The feature map needs to be zero-padded for median filter operation.
    - After median filtering, you have to output the **2×2×4** filtered results in **raster-scan** order.
    - Set **o_out_data [13:8]** to 0 and **o_out_data [7:0]** to pixel data.
    - The values of original pixels will not be changed.



11. Haar wavelet transform [3]:
    - For this operation, you have to perform Haar wavelet transform **in the first 4 channels of the display region**.
    - Note that the transform involves **signed arithmetic**.
    - Perform Haar wavelet transform on each channel **separately.**
    - The results of HWT should be r**ounded to the nearest integer (positive biased)** [2].
    - After the transform, you have to output the **2×2×4** results in **raster-scan** order.
    - The values of original pixels will not be changed.

- The orthonormal filters of 2-point Haar wavelet are defined as

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

- The 2-point Haar wavelet transform of a 2×2 region can be written as

$$B = HAH^T$$

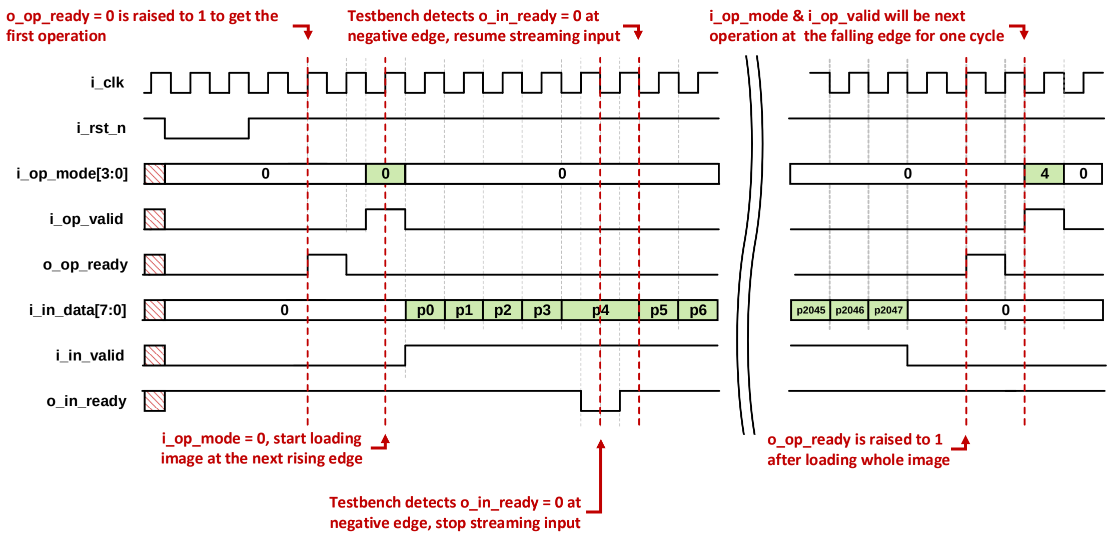
- An example:

8	32	108	174	71	112	181	245
4	64	234	194	12	98	193	87
33	41	203	190	25	196	71	150
233	248	245	101	210	203	174	58
162	245	168	168	178	48	168	192
25	124	10	44	81	125	42	66
72	205	217	181	243	114	31	130
140	37	239	9	16	165	128	179

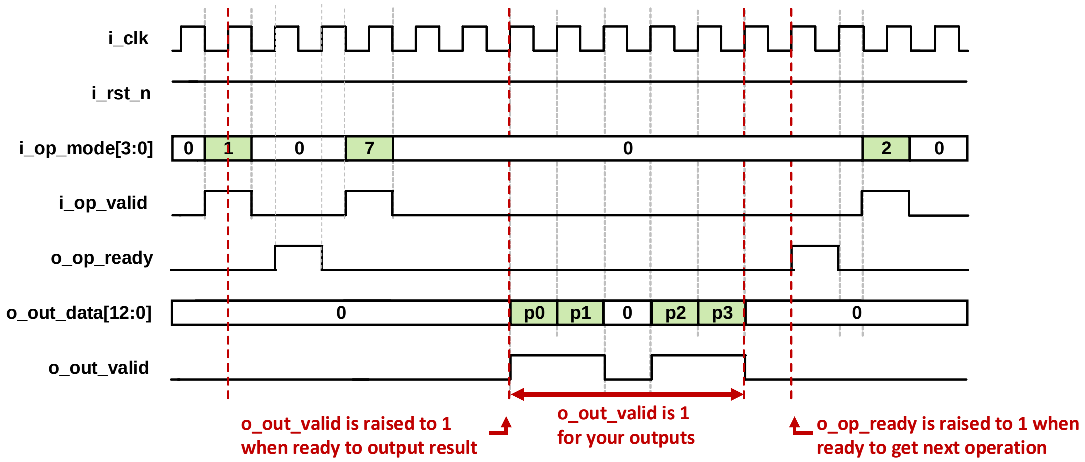
$$B = \frac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}\begin{bmatrix} 8 & 32 \\ 4 & 64 \end{bmatrix}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 54 & -42 \\ -14 & -18 \end{bmatrix}$$

Sample Waveform

1. Load Image Data (i_op_mode = 0)



2. Other operations



Submission

1. Create a folder named **studentID_hw3**, and put all below files into the folder
   - **core.v**
   - **core_syn.v**
   - **core_syn.sdf**
   - **core_syn.ddc**
   - **core_syn.area**
   - **core_syn.timing**
   - **report.txt**
   - **syn.tcl**
   - **rtl_01.f**
   - **rtl_03.f**
   - **all other design files** included in your design (optional)

   Note: Use **lower case** for the letter in your student ID. (Ex. r11943006_hw3)

2. Compress the folder **studentID_hw3** in a **tar file** named **studentID_hw3_v*k*.tar** **(*k* is the number of version, *k* =1,2,…)**

```
tar -cvf studentID_hw3_vk.tar studentID_hw3
```

   TA will only check the last version of your homework.

   Note: Use **lower case** for the letter in your student ID.

   (Ex. r11943006_hw3_v1.tar)

3. Submit to NTU COOL

Grading Policy

1. TA will run your code with following format of commands.
   a. RTL simulation (under **01_RTL**)

```
vcs -f rtl_01.f -full64 -R -debug_access+all +v2k +notimingcheck
+define+tb0
```

   b. Gate-level simulation (under **03_GATE**)

```
vcs -f rtl_03.f -full64 -R -debug_access+all +v2k +maxdelays -negdelay
+neg_tchk +define+SDF+tb0
```

2. Correctness of simulation: **70%** (follow our spec)

Pattern	Description	RTL simulation	Gate-level simulation
tb0	Load + shift + scale + display	5%	5%
tb1	Load + shift + scale + conv.	10%	10%
tb2	Load + shift + median filter	5%	5%
tb3	Load + shift + Haar	5%	5%
tb4	All operations (no display)	5%	5%
tbh	Hidden patterns	x	10%

3. Performance: **30%**
   - Performance = **Area * Time (μm^2 * ns)**
     - **Time = total simulation time of tb4**
     - The lower the value, the better the performance.
   - Performance score only counts if your design passes all the test patterns.
4. **No late submission**
   - 0 point for this homework
5. Lose **3 points** for any wrong naming rule or format for submission.
   - Don’t compress all homework folder and upload to NTU COOL
6. No plagiarism

References

[1] Illustrations for convolution
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1
[2] Rounding to the nearest
https://www.mathworks.com/help/fixedpoint/ug/rounding.html
[3] Haar wavelet transform
Haar wavelet transform - Wikipedia