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

TA: 林瑩昇 r10943114@ntu.edu.tw

TA: 羅宇呈 f08943129@ntu.edu.tw

Due Tuesday, Nov. 08, 14:00

Data Preparation

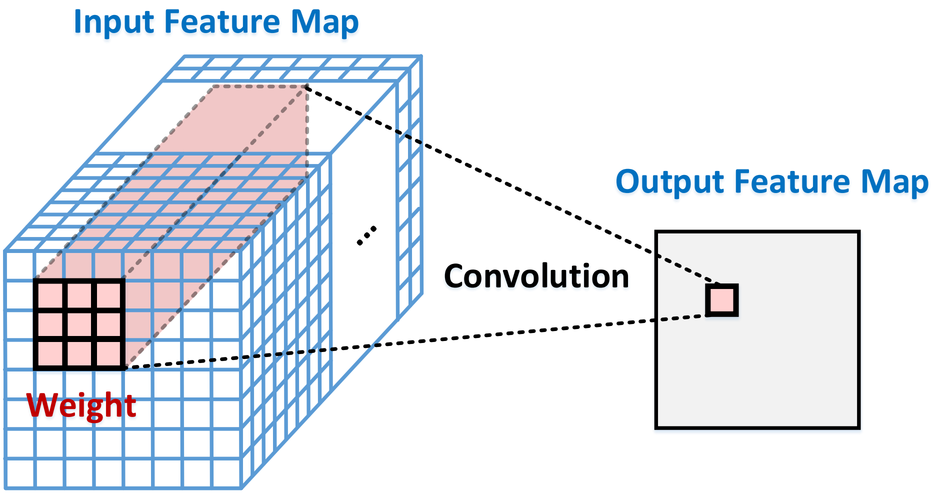
1. Decompress 1111_hw3.tar with following command

```
tar -xvf 1111_hw3.tar
```

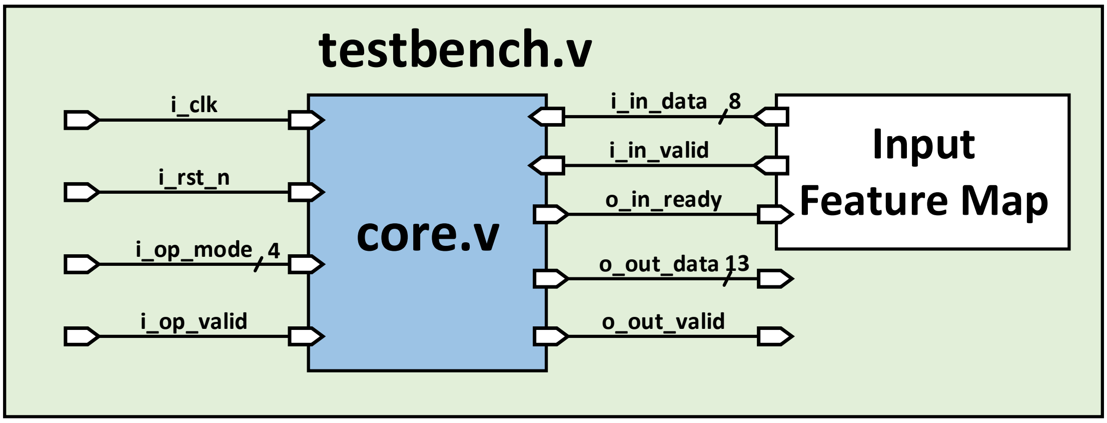
Folder	File	Description
00_TESTBED	testbench_temp.v	Testbench template
00_TESTBED/ PATTERN/	indata*.dat	Input image data
	opmode*.dat	Pattern of operation mode
	golden*.dat	Golden data of output
01_RTL	core.v	Your design
	rtl_01.f	File list for rtl simulation
	01_run	NCVerilog command
	99_clean_up	Command 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 engine with some simple functions. 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	13	Pixel data or convolution result (**unsigned**)

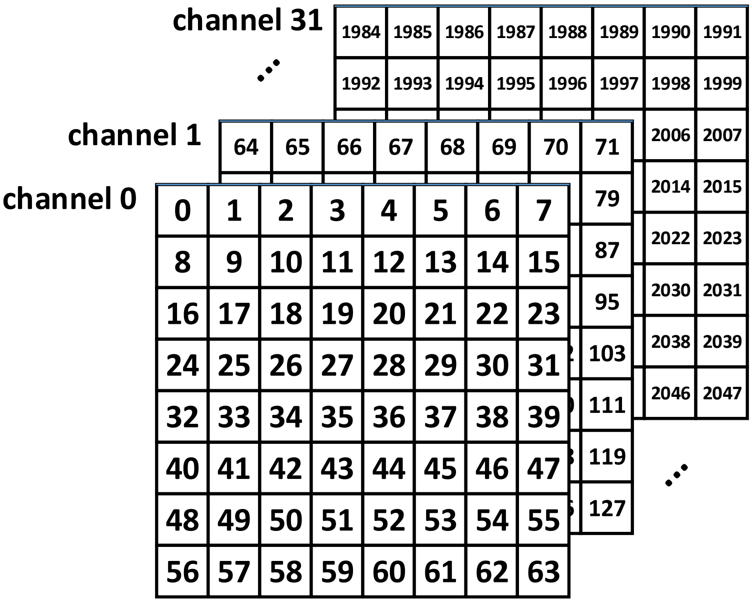
3. All inputs are synchronized with the **negative** edge clock.
4. All outputs should be synchronized at clock **rising** edge.
5. You should reset all your outputs when i_rst_n is **low**. Active low asynchronous reset is used and only once.
6. Operations are given by i_op_mode [3:0] when i_op_valid 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.

Design Description

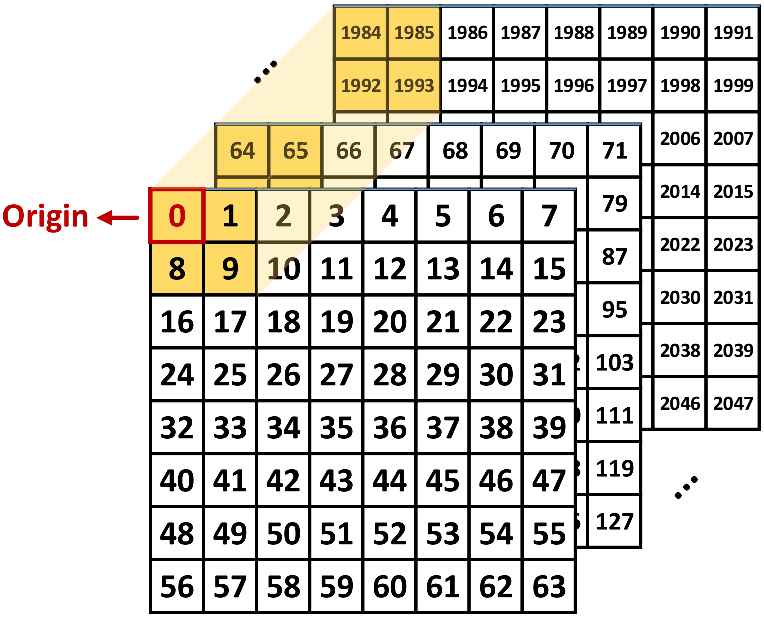
1. The followings are the operation modes you need to design for this homework:

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 number of channels for convolution
4’b0110	Increase the number of channels for convolution
4’b0111	Perform convolution in the display region
4’b1000	Output the pixels in the display region

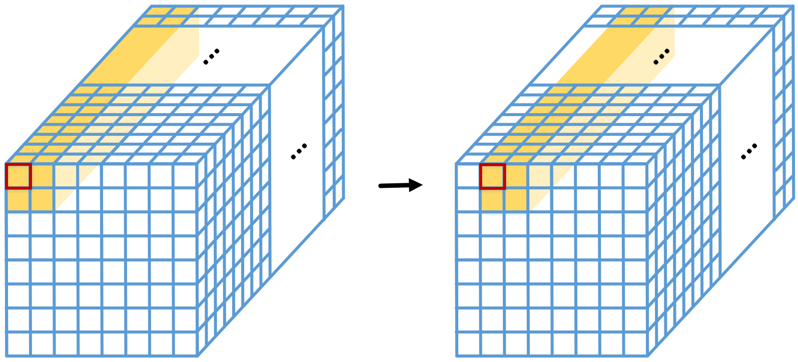
2. Input feature map loading:
   - An 8×8×32 feature map is loaded for 2048 cycles in **raster-scan** order.
   - The size of each pixel is 8 bits.
   - Raise o_op_ready to 1 after loading all pixels.
   - If o_in_ready is 0, stop input data until o_in_ready is 1.



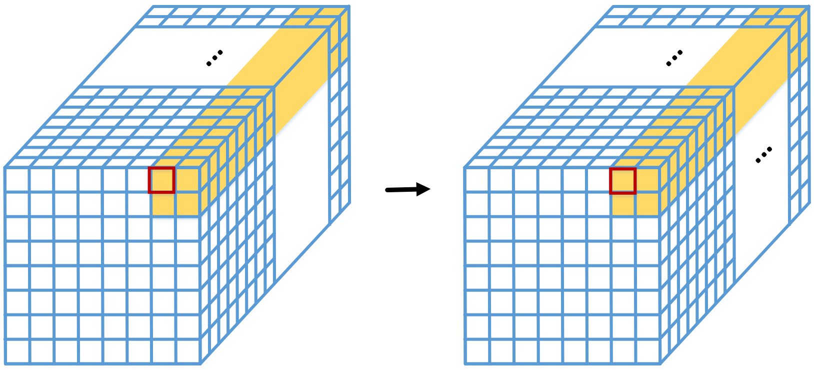
3. The first output pixel of the display is **origin**.
   - The default coordinate of the origin is at 0.
   - The size of the display region is 2×2×depth.



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



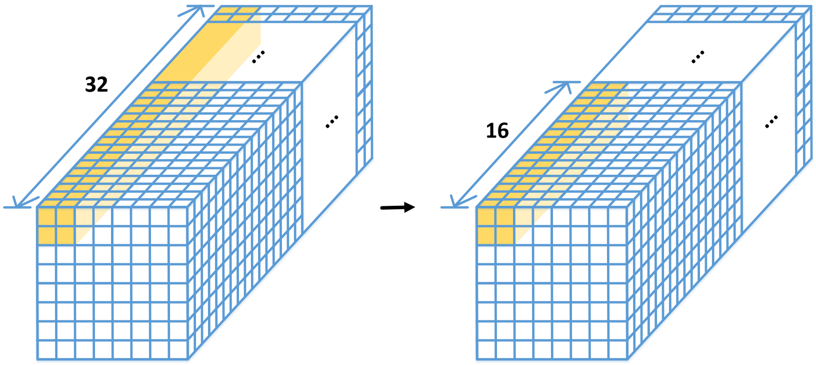
   - If output of display exceeds the boundary, retain the same origin point.



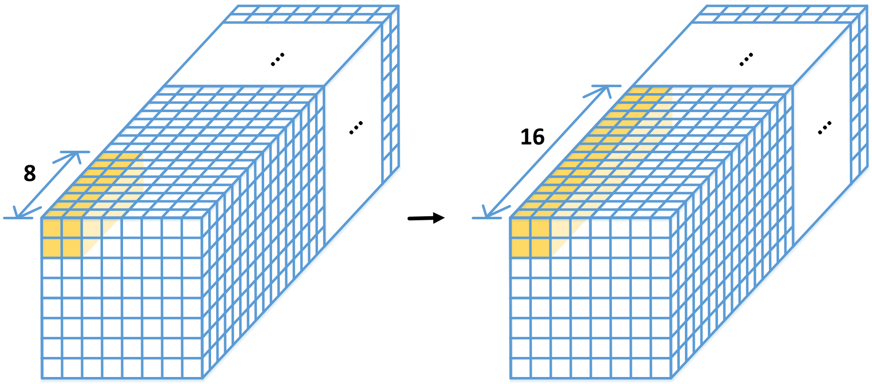
5. Channel depth:
   - 3 depths are considered in this design: 32, 16, and 8
   - The display size will change according to different depth

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

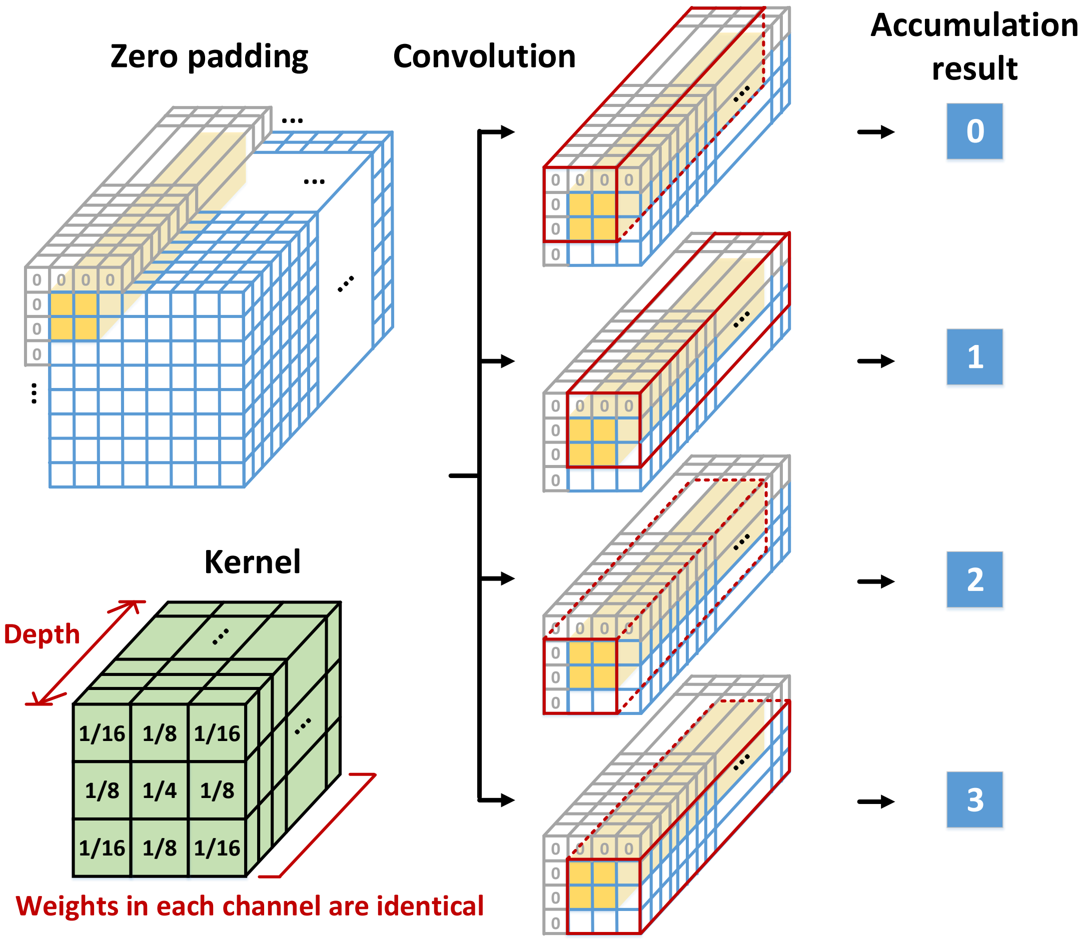
6. Scale-down:
   - Scale down the depth of the channel to next level
     - ■ Ex. For channel depth, 32 → 16 → 8
   - If the depth is 8, retain the same depth



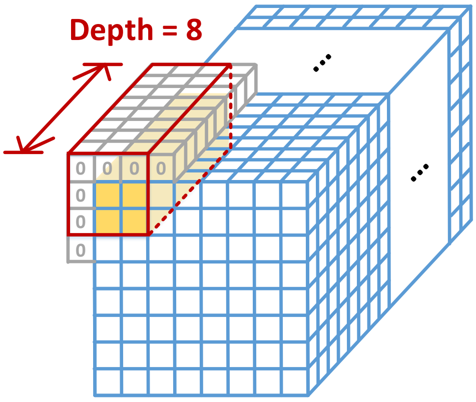
7. Scale-up:
   - Scale up the depth of the channel to next level
     - ■ Ex. For image size, 8 → 16 → 32
   - If the depth is 32, retain the same depth



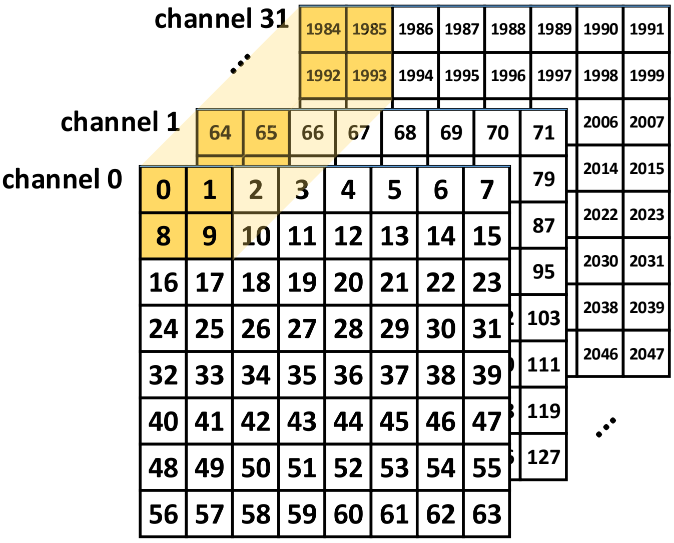
8. Convolution:
   - For this operation, you have to perform convolution **in the display region**.
   - The size of the kernel is 3×3×depth. The weights in each channel are identical.
   - The accumulation results should be **rounded (四捨五入) to integer.**
   - The feature map needs to be zero-padded for convolution.
   - The values of original pixels will not be changed.
   - After the convolution, you have to output the four accumulation results in **raster-scan** order.



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



9. Display:
    - For this operation, you have to output the pixels in the display region.
    - Set o_out_data [12:8] to 0 and o_out_data [7:0] to the pixel data.
    - When i_op_mode = 4’b1000, the pixels are displayed in **raster-scan** order. (For example: 0 → 1 →8 → 9 → 64 → 65 → … → 1992 → 1993)

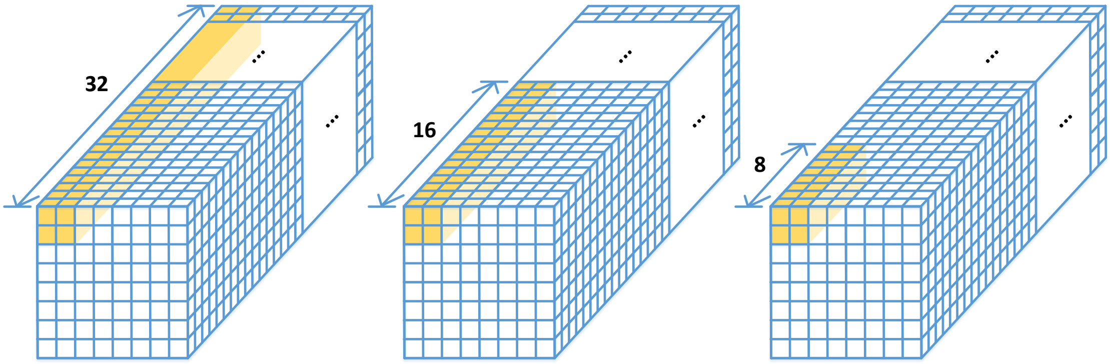


1984	1985	1986	1987	1988	1989	1990	1991
1992	1993	1994	1995	1996	1997	1998	1999
						2006	2007
						2014	2015
						2022	2023
						2030	2031
						2038	2039
						2046	2047

64	65	66	67	68	69	70	71
							79
							87
							95
							103
							111
							119
							127

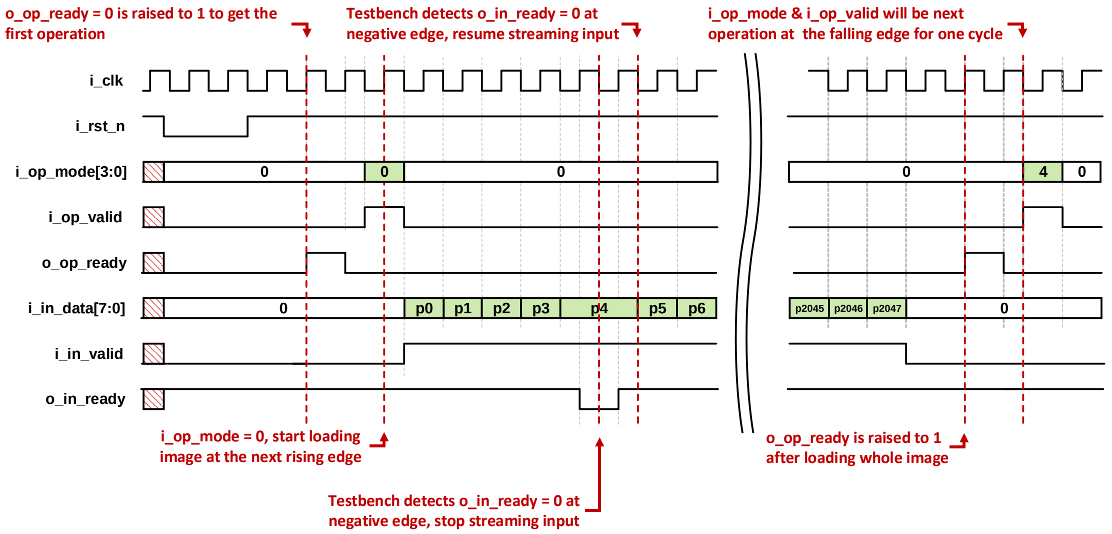
0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23
24	25	26	27	28	29	30	31
32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47
48	49	50	51	52	53	54	55
56	57	58	59	60	61	62	63

- The size of display region changes according to depth.



Sample Waveform

1. Load Image Data (i_op_mode = 0)



o_op_ready = 0 is raised to 1 to get the first operation

Testbench detects o_in_ready = 0 at negative edge, resume streaming input

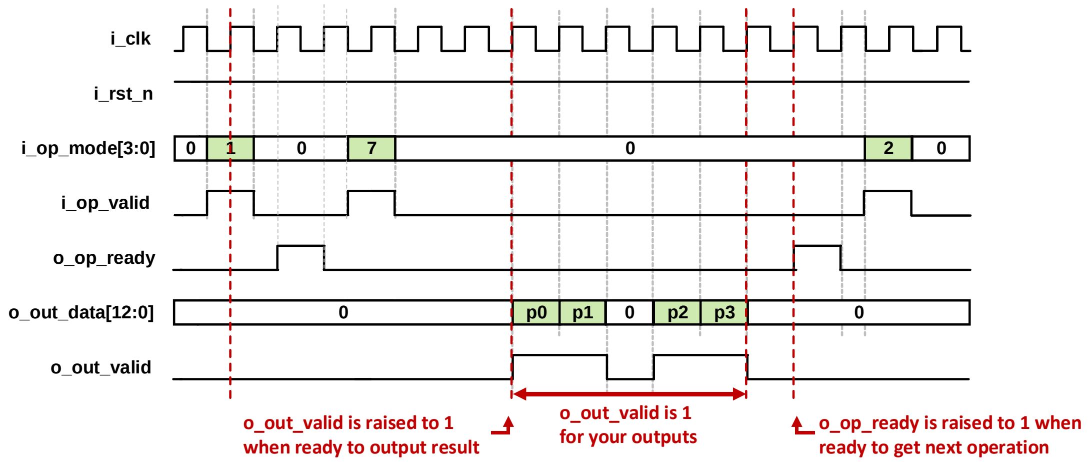
i_op_mode & i_op_valid will be next operation at the falling edge for one cycle

i_op_mode = 0, start loading image at the next rising edge

o_op_ready is raised to 1 after loading whole image

Testbench detects o_in_ready = 0 at negative edge, stop streaming input

2. Other operations



o_out_valid is raised to 1 when ready to output result

o_out_valid is 1 for your outputs

o_op_ready is raised to 1 when ready to get next operation

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. r06943027_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. r06943027_hw3_v1)

3. Submit to NTU COOL

Grading Policy

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

```
ncverilog -f rtl_01.f +notimingchecks +access+r +define+tb0
```

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

```
ncverilog -f rtl_03.f \
          +ncmaxdelays +define+SDF+tb0 +access+r
```

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

Pattern	Description	RTL simulation	Gate-level simulation
tb0	Load + shift + display	5%	5%
tb1	Load + shift + scale + display	5%	5%
tb2	Load + shift + convolution	10%	10%
tb3	All operations (no display)	10%	10%
hidden	10 hidden patterns	x	10%

3. Performance: **30%** (correct for all patterns)
   - **Area * Time (μm^2 * ns)**
   - (Lower number is better performance)
   - The latency of loading input feature map will be subtracted with 2048 cycles.
4. Delay submission
   - In one day: **(original score)*0.7**
   - In two days: **(original score)*0.4**
   - More than two days: 0 point for this homework
5. Lose **3 point** for any wrong naming rule or format for submission.
   - Don't compress all homework folder and upload to NTU COOL

Designware

1. Document

```
evince
/home/raid7_4/raid1_1/linux/synopsys/synthesis/cur/dw/doc/datasheets/*.pdf
```

2. Include designware IP in your **rtl_01.f** for RTL simulation

Example:

```
// DesignWare
// ----------------------------------------------------------------------
/home/raid7_4/raid1_1/linux/synopsys/synthesis/cur/dw/sim_ver/DW_norm.v
```

3. Change your RTL simulation command

```
ncverilog -f rtl_01.f -incdir
/home/raid7_4/raid1_1/linux/synopsys/synthesis/cur/dw/sim_ver/
+notimingchecks +access+r +define+tb0
```

Reference

[1] Illustrations for convolution: https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1