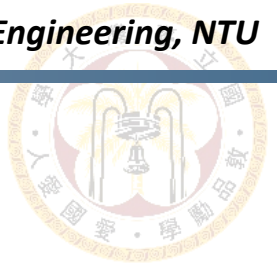
Computer-Aided VLSI System Design

Homework 3: Simple Convolution and Image Processing Engine

Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE

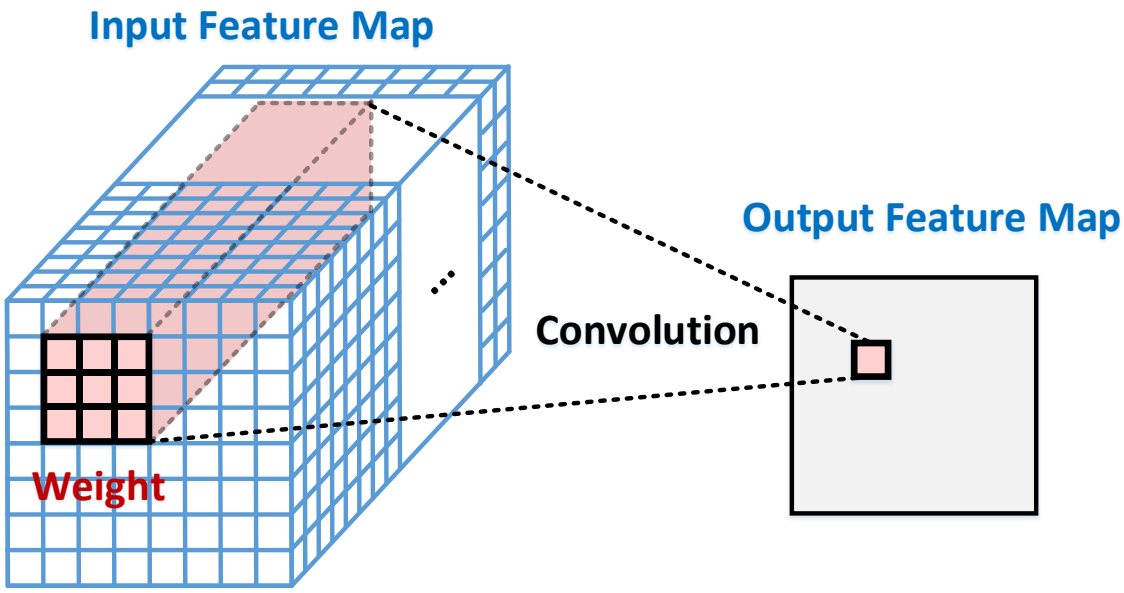


Goal

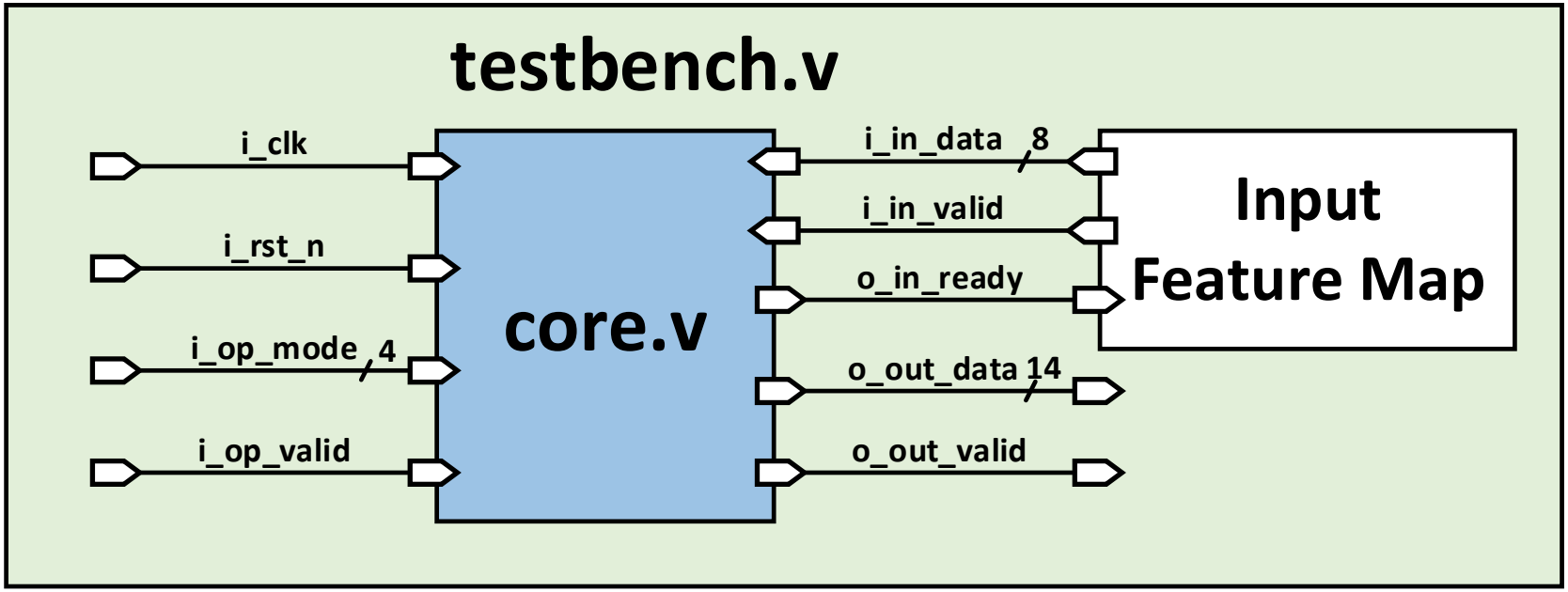
- In this homework, you will learn
  - How to synthesis your design
  - How to run gate-level simulation
  - How to use SRAM

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.

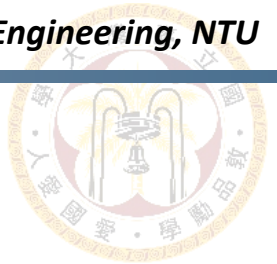


Block Diagram



Input/Output

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**)



Specification(1)

- All inputs are synchronized with the **negative** edge clock
- All outputs should be synchronized at clock **rising** edge
- You should reset all your outputs when i_rst_n is **low**
  - Active low asynchronous reset is used and only once

Specification(2)

- Operations are given by i_op_mode when i_op_valid is **high**
- i_op_valid stays only **1** cycle
- i_in_valid and o_out_valid can't be **high** in the same time
- i_op_valid and o_out_valid can't be **high** in the same time
- i_in_valid and o_op_ready can't be **high** in the same time
- i_op_valid and o_op_ready can't be **high** in the same time
- o_op_ready and o_out_valid can't be **high** in the same time

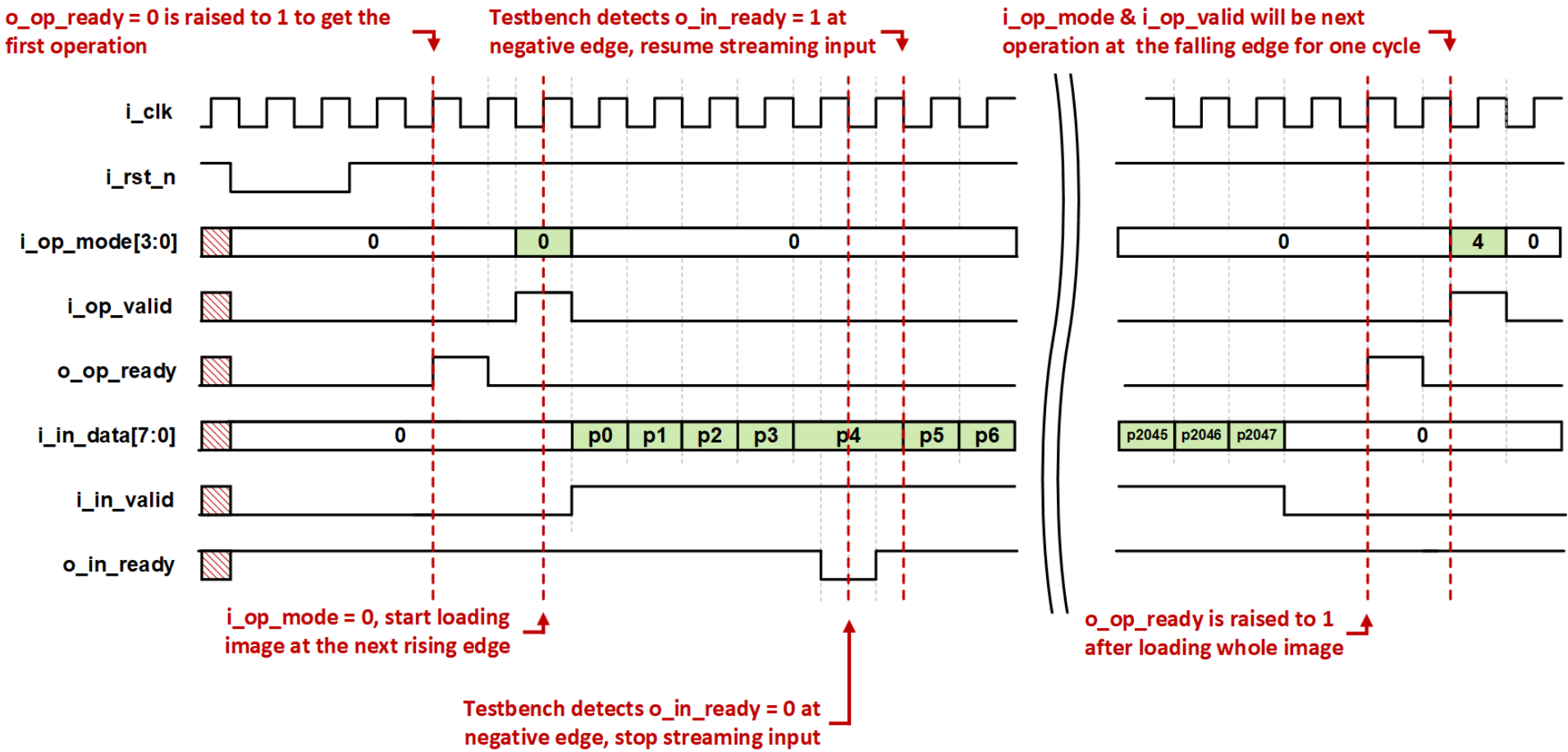
Specification(3)

- Set o_op_ready to **high** to get next operation (only one cycle)
- o_out_valid should be **high** for valid output results
- **At least one SRAM** is implemented in your design

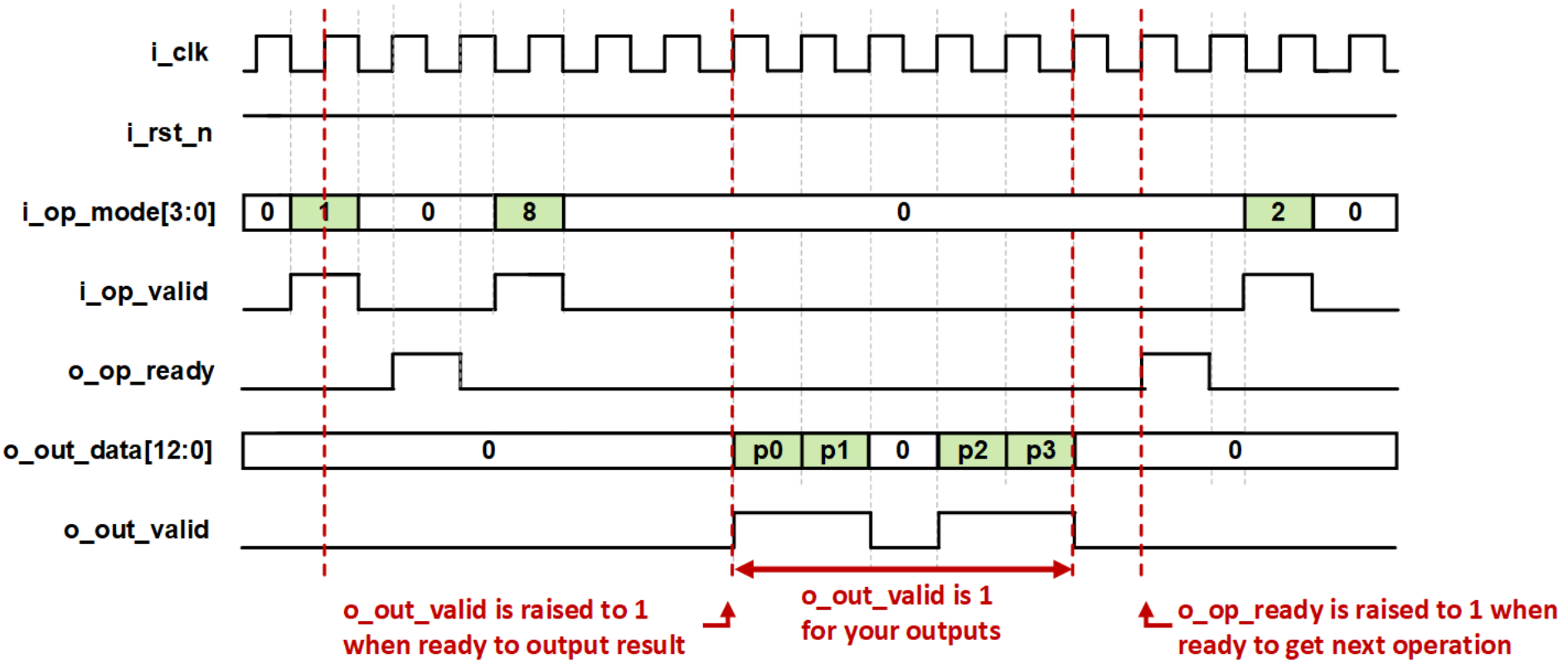
Specification(4)

- Only worst-case library is used for synthesis.
- The synthesis result of data type should **NOT** include any **Latch**.
- The slack for setup-time should be **non-negative**.
- **No any timing violation and glitches** for the gate level simulation **after reset**.

Waveform: Loading Image



Waveform: Other Operations

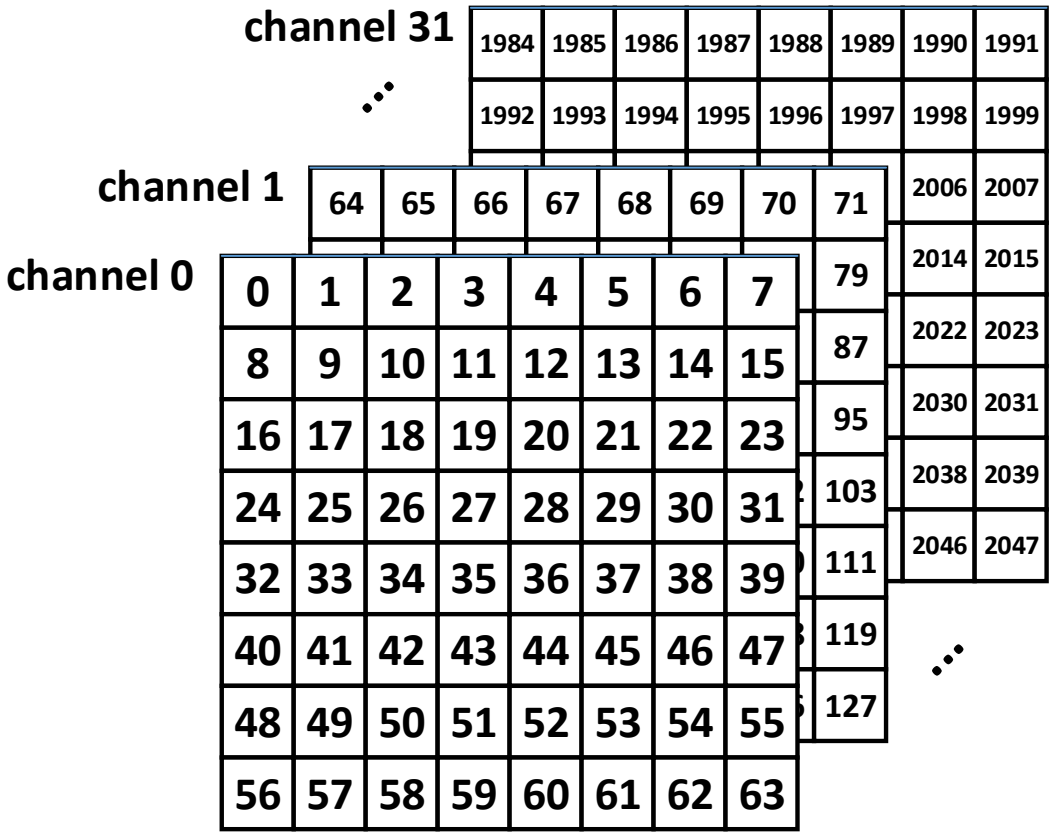


Operation Modes

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

Input Image

- The input image is given in **raster-scan** order

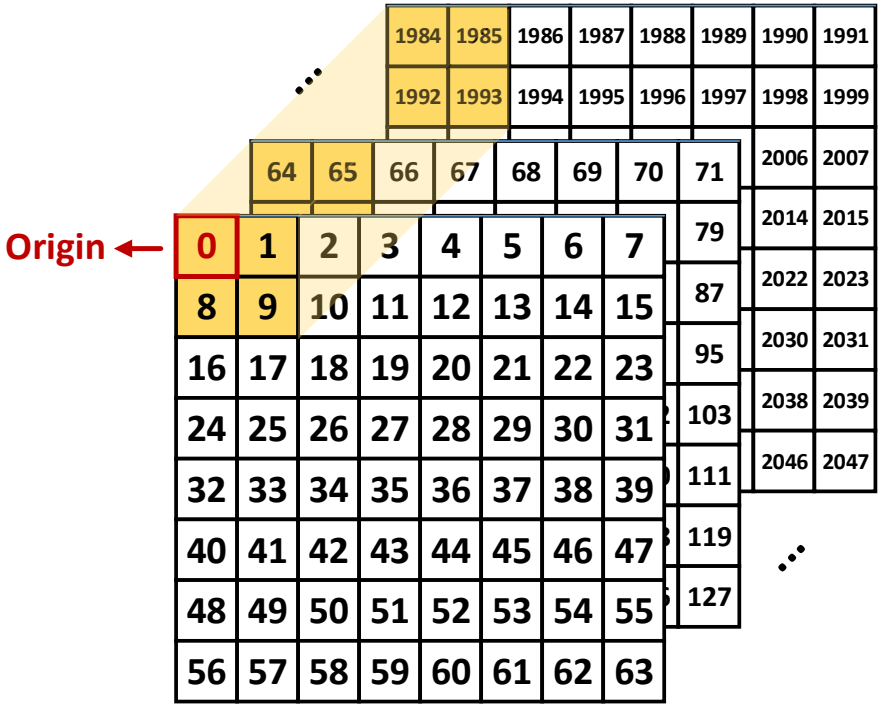


Input Image Loading

- An 8x8x32 image 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 image 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

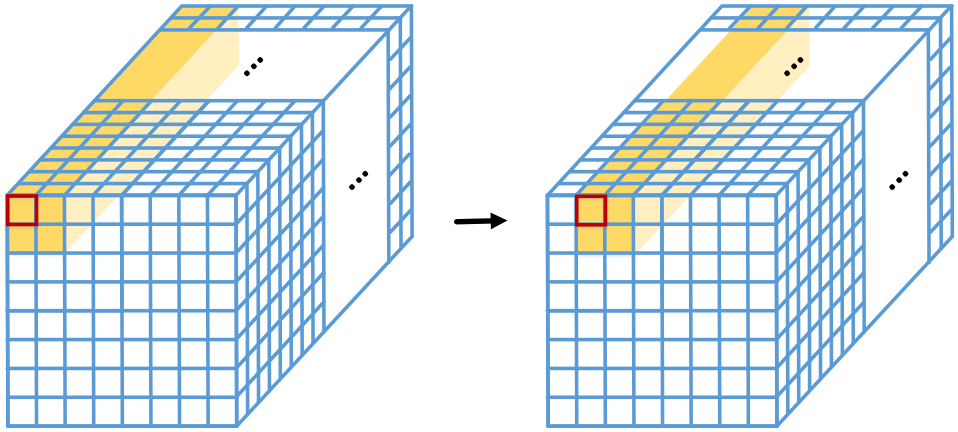
Origin

- 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 $2 \times 2 \times$ depth

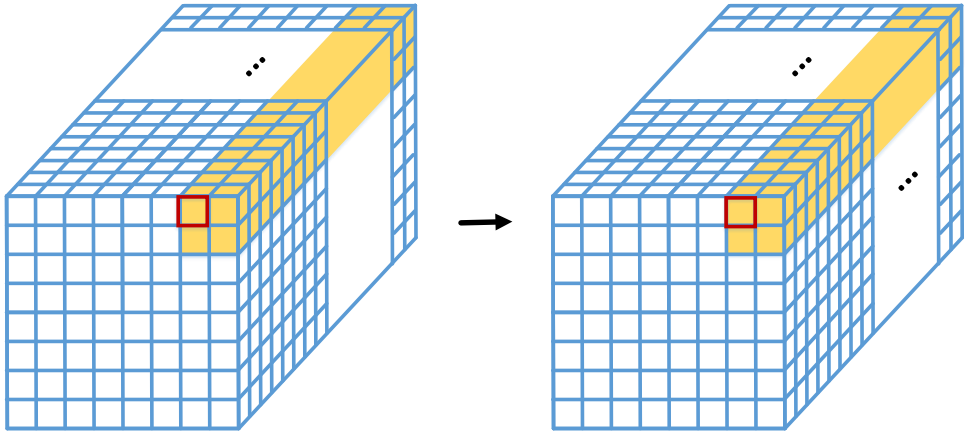


Origin Shifting

- Origin right shift



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



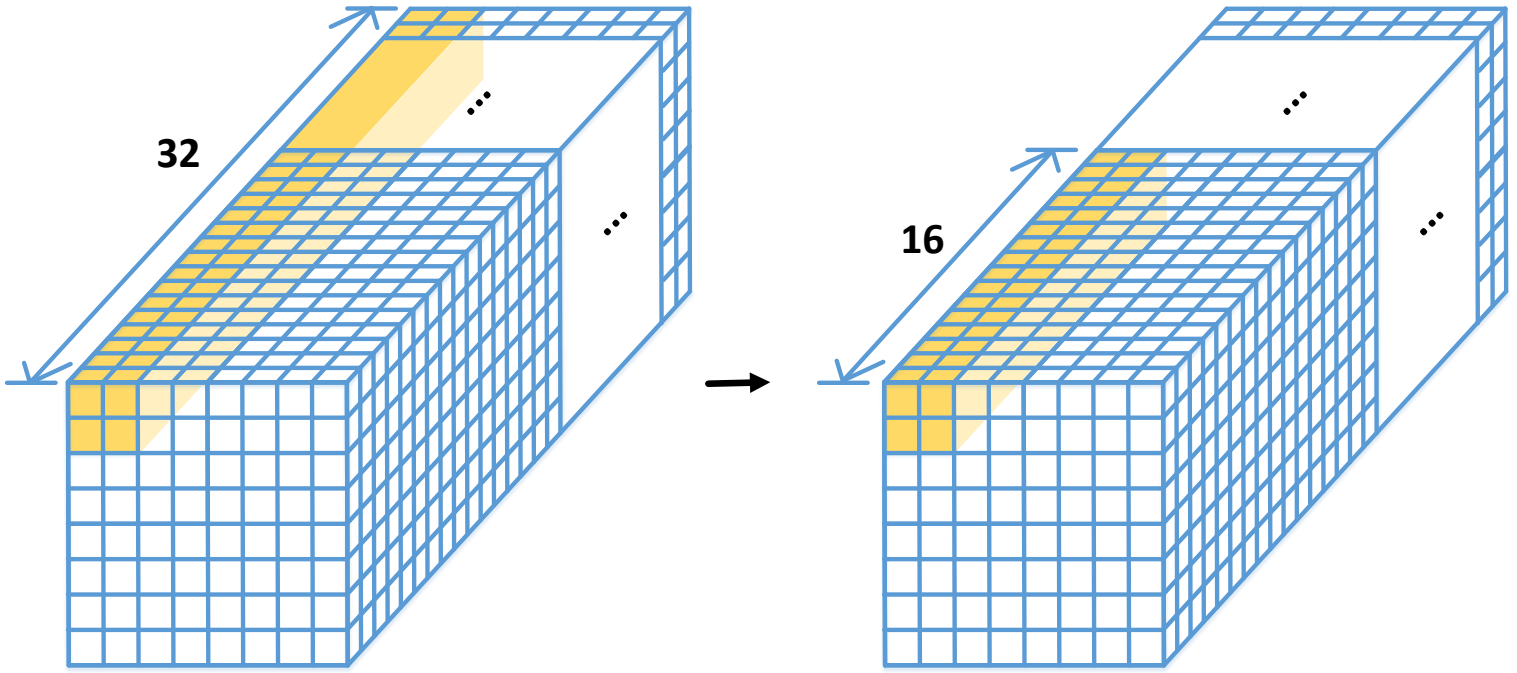
Channel Depth

- 3 depths are considered in this design
  - 32, 16, 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

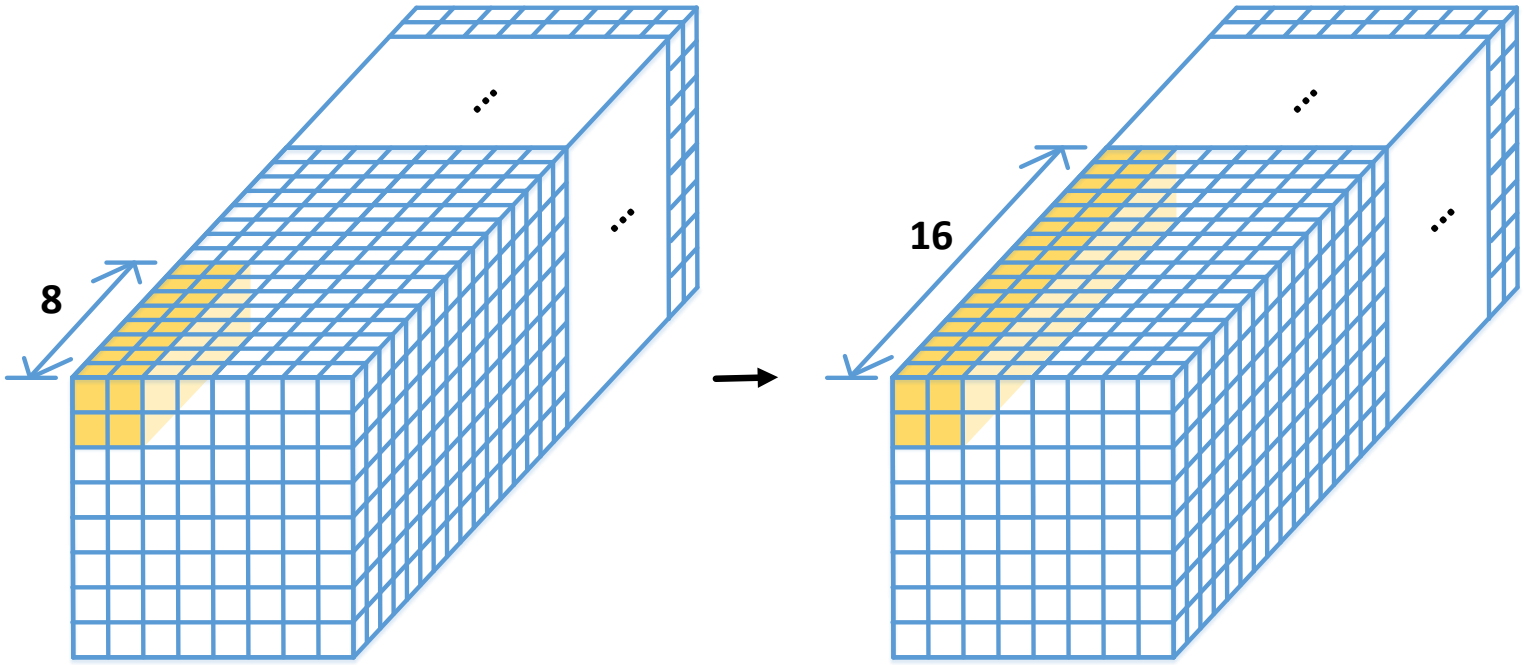
Scale-down

- Reduce the channel depth of the display region
  - Ex. For channel depth, 32 → 16 → 8
- If the depth is 8, retain the same depth



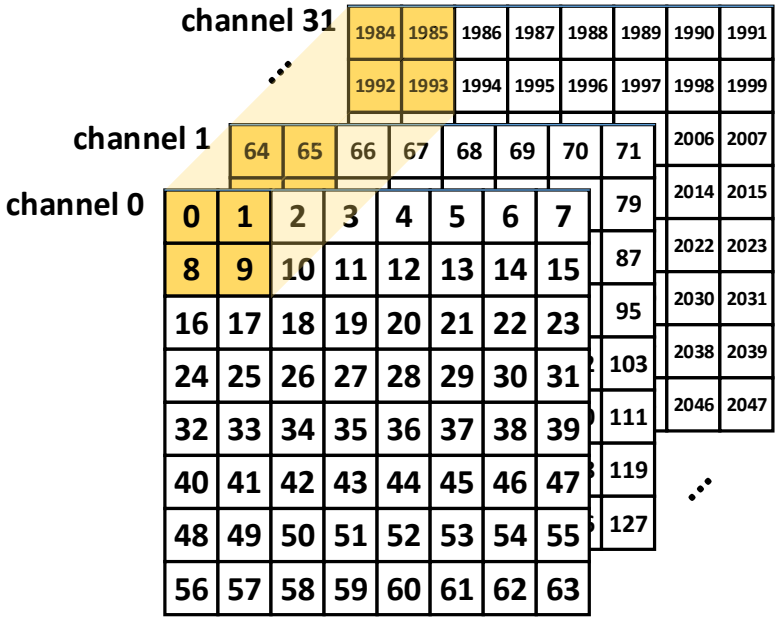
Scale-up

- Increase the channel depth of the display region
  - Ex. For channel depth, 8 → 16 → 32
- If the depth is 32, retain the same depth



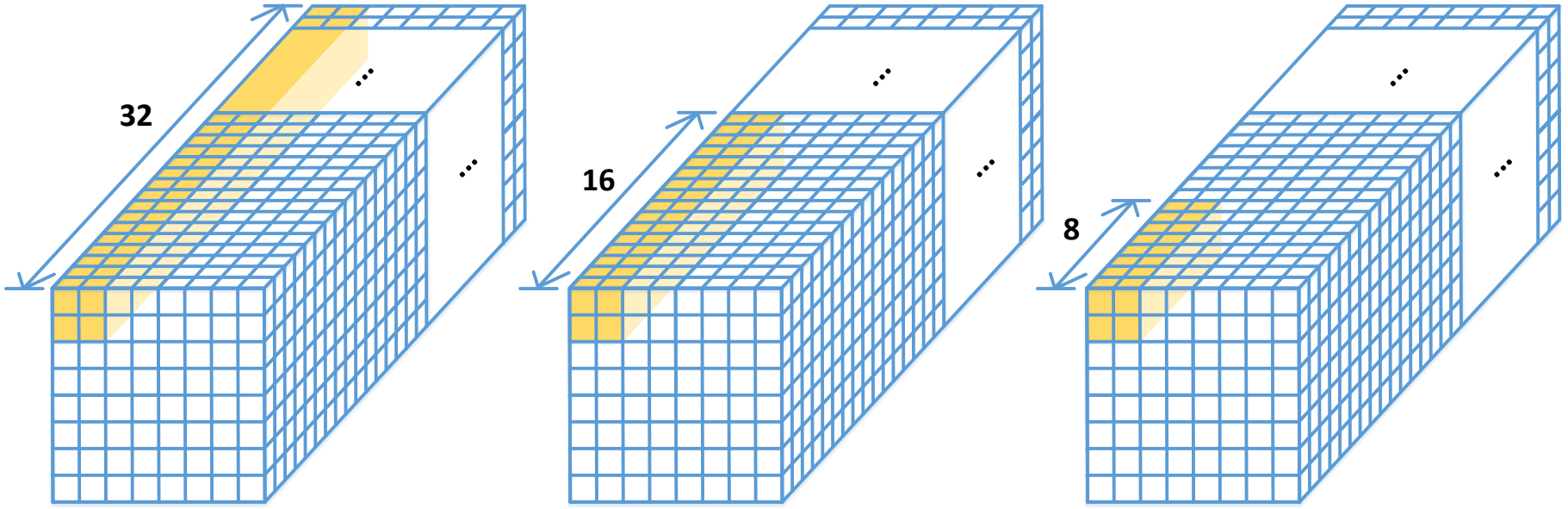
Display

- 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 pixel data
- The pixels are displayed in **raster-scan** order
  - For example: 0 → 1 →8 → 9 → 64 → 65 → … → 1992 → 1993



Display

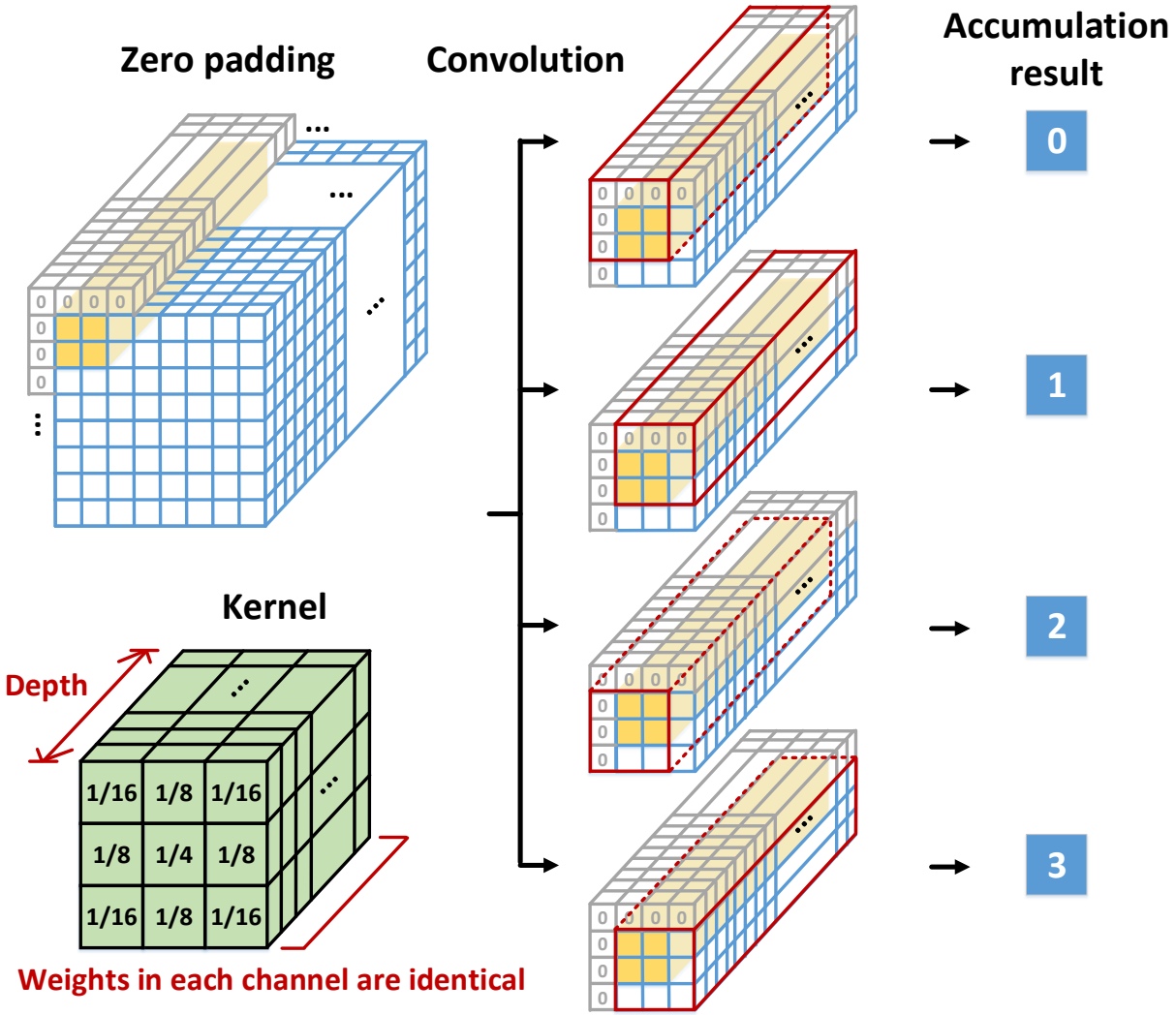
- For display, the display size changes according to the depth



Convolution

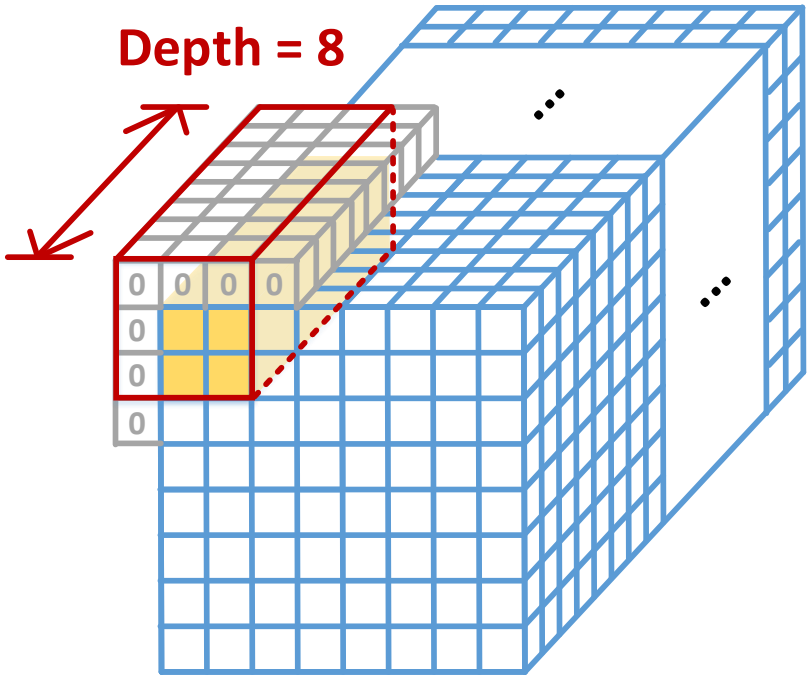
- For this operation, you have to perform convolution **in the display region**
- The size of the kernel is a 3 x 3 x depth and the weights in each channel are identical
- The feature map needs to be zero-padded for convolution
- The accumulation results should be **rounded to the nearest integer** [1]
  - 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

Example of Convolution



Example of Convolution

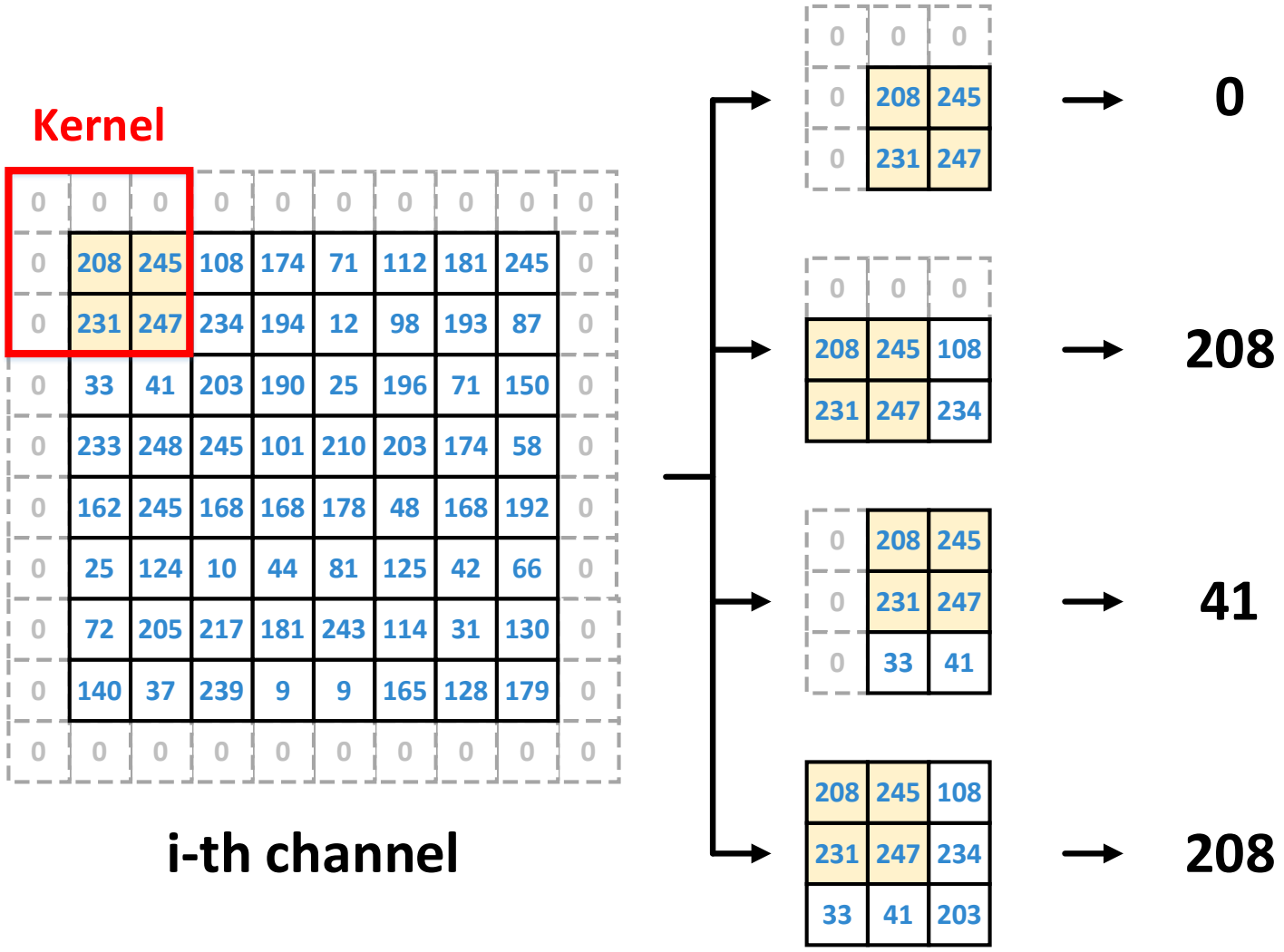
- The number of channels that are accumulated during convolution is determined by the depth.
  - For example, accumulate 8 channels if the depth is 8.



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 x 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 x 2 x 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

Example of Median Filter Operation



Haar Wavelet Transform

- 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 **rounded to the nearest integer (positive biased)** [1]
- After the transform, you have to output the **2 x 2 x 4** results in **raster-scan** order
- The values of original pixels will not be changed

Haar Wavelet Transform

- 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 x 2 region can be written as

$$B = HAH^T$$

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

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

$$= \begin{bmatrix} 54 & -42 \\ -14 & 18 \end{bmatrix}$$

Testbench

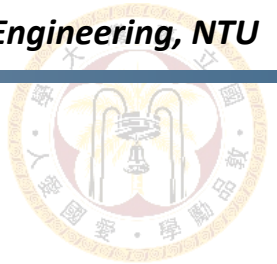
```
`timescale 1ns/100ps
`define CYCLE       10.0     // CLK period.
`define HCYCLE      (`CYCLE/2)
`define MAX_CYCLE   10000000
`define RST_DELAY   2


`ifdef tb1
    `define INFILE "../00_TESTBED/PATTERN/indata1.dat"
    `define OPFILE "../00_TESTBED/PATTERN/opmode1.dat"
    `define GOLDEN "../00_TESTBED/PATTERN/golden1.dat"
`elsif tb2
    `define INFILE "../00_TESTBED/PATTERN/indata2.dat"
    `define OPFILE "../00_TESTBED/PATTERN/opmode2.dat"
    `define GOLDEN "../00_TESTBED/PATTERN/golden2.dat"
`elsif tb3
    `define INFILE "../00_TESTBED/PATTERN/indata3.dat"
    `define OPFILE "../00_TESTBED/PATTERN/opmode3.dat"
    `define GOLDEN "../00_TESTBED/PATTERN/golden3.dat"
```

```
`elsif tb4
    `define INFILE "../00_TESTBED/PATTERN/indata4.dat"
    `define OPFILE "../00_TESTBED/PATTERN/opmode4.dat"
    `define GOLDEN "../00_TESTBED/PATTERN/golden4.dat"
`elsif tbh
    `define INFILE "../00_TESTBED/PATTERN/indatah.dat"
    `define OPFILE "../00_TESTBED/PATTERN/opmodeh.dat"
    `define GOLDEN "../00_TESTBED/PATTERN/goldenh.dat"
`else
    `define INFILE "../00_TESTBED/PATTERN/indata0.dat"
    `define OPFILE "../00_TESTBED/PATTERN/opmode0.dat"
    `define GOLDEN "../00_TESTBED/PATTERN/golden0.dat"
`endif

`define SDFFILE "core_syn.sdf"  // Modify your sdf file name
```

```
// For gate-level simulation only
`ifdef SDF
    initial $sdf_annotate(`SDFFILE, u_core);
    initial #1 $display("SDF File %s were used for this simulation.", `SDFFILE);
`endif
```



Pattern

indata*.dat

```
11010000
10101101
10001011
01010110
00101110
11110001
11110110
11001101
01000110
11001110
00001011
10101111
10001000
00111111
00001100
11110100
11100100
00101100
11000100
11001000
```

opmode*.dat

```
0000
0010
1000
0100
1000
0011
1000
0100
1000
0110
1000
0001
1000
0110
1000
0110
1000
0011
1000
0100
```

golden*.dat

```
00100111111001
00110011000100
00110001011010
01000001010101
00110001011010
01000001010101
00101110110101
00111111110110
00100111111001
00110011000100
00110001011010
01000001010101
00110001011010
01000001010101
00101110110101
00111111110110
00110001011010
01000001010101
00101110110101
00111111110110
```

01_RTL

- **core.v**

```
module core (                       //Don't modify interface
    input         i_clk,
    input         i_rst_n,
    input         i_op_valid,
    input  [ 3:0] i_op_mode,
    output        o_op_ready,
    input         i_in_valid,
    input  [ 7:0] i_in_data,
    output        o_in_ready,
    output        o_out_valid,
    output [13:0] o_out_data
);
```

01_RTL

- Run the RTL simulation under 01_RTL folder

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

tb0, tb1, tb2, tb3, tb4

or

```
./01_run tb0 5.0
```

clock period

rtl_01.f

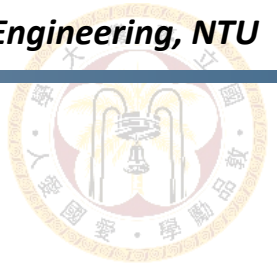
- **rtl_01.f**

```
// ------------------------------------------------------------------------------
// Simulation: HW3
// ------------------------------------------------------------------------------

// testbench
// ------------------------------------------------------------------------------
../00_TESTBED/testbench.v

// memory file
// ------------------------------------------------------------------------------
//../sram_256x8/sram_256x8.v
//../sram_512x8/sram_512x8.v
//../sram_4096x8/sram_4096x8.v


// design files
// ------------------------------------------------------------------------------
./core.v
```



02_SYN

- **core_dc.sdc**

```
# operating conditions and boundary conditions #
set cycle  5.0;  # modify your clock cycle here #
```

- Run the command to do synthesis

```
dc_shell-t -f syn.tcl | tee syn.log
```

03_GATE

- Run gate-level simulation under 03_GATE folder

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

or

```
./03_run tb0 5.0
```

clock period

sram_256x8

Pin Description

Pin	Description
A[7:0]	Addresses (A[0] = LSB)
D[7:0]	Data Inputs (D[0] = LSB)
CLK	Clock Input
CEN	Chip Enable
WEN	Write Enable
Q[7:0]	Data Outputs (Q[0] = LSB)

SRAM Logic Table

CEN	WEN	Data Out	Mode	Function
H	X	Last Data	Standby	Address inputs are disabled; data stored in the memory is retained, but the memory cannot be accessed for new reads or writes. Data outputs remain stable.
L	L	Data In	Write	Data on the data input bus D[n-1:0] is written to the memory location specified on the address bus A[m-1:0], and driven through to the data output bus Q[n-1:0].
L	H	SRAM Data	Read	Data on the data output bus Q[n-1:0] is read from the memory location specified on the address bus A[m-1:0].

Submission

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

Grading Policy

- 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%

- 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

Grading Policy

- **No late submission**
  - 0 point for this homework
- Lose **3 points** for any wrong naming rule or format for submission
  - Do not compress all homework folder and upload to NTU COOL
- No plagiarism

Area

- core_syn.area

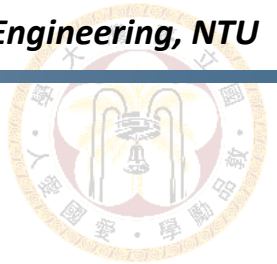
```
Number of ports:                          103
Number of nets:                          2808
Number of cells:                         2540
Number of combinational cells:           2319
Number of sequential cells:               215
Number of macros/black boxes:               1
Number of buf/inv:                        481
Number of references:                     231

Combinational area:                24400.124917
Buf/Inv area:                       2914.435785
Noncombinational area:              7529.666225
Macro/Black Box area:             131906.968750
Net Interconnect area:            312201.571442

Total cell area:                  163836.759892
Total area:                       476038.331334
```

Number of macros/black boxes should not be 0

163836.759892 μm^2



Report

- TAs will run your design with your clock period

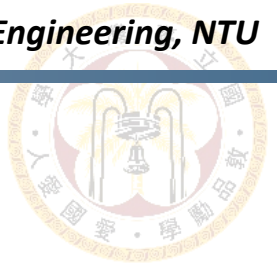
- **report.txt**

```
StudentID:

Clock period: (ns)

Area :  (um^2)
```

The clock period that can pass all gate-level simulations without any timing violations



References

- [1] Rounding to the nearest
  - Rounding - MATLAB & Simulink (mathworks.com)
- [2] Haar wavelet transform
  - Haar wavelet transform - Wikipedia