Computer-Aided VLSI System Design

Homework 3: Simple Convolution Engine

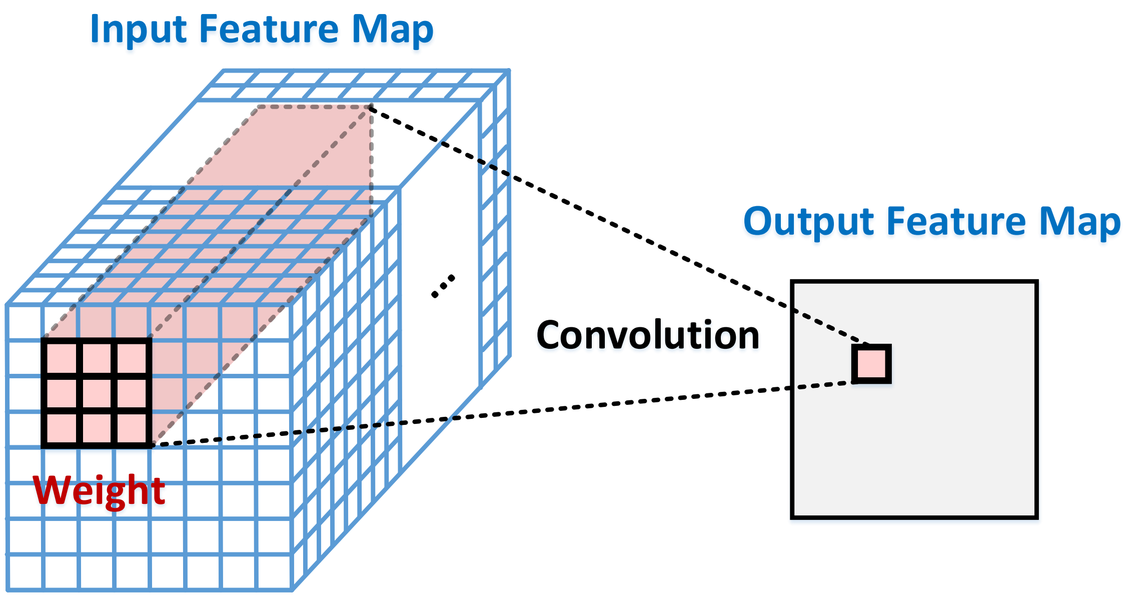
Graduate Institute of Electronics Engineering, National Taiwan University

Goal

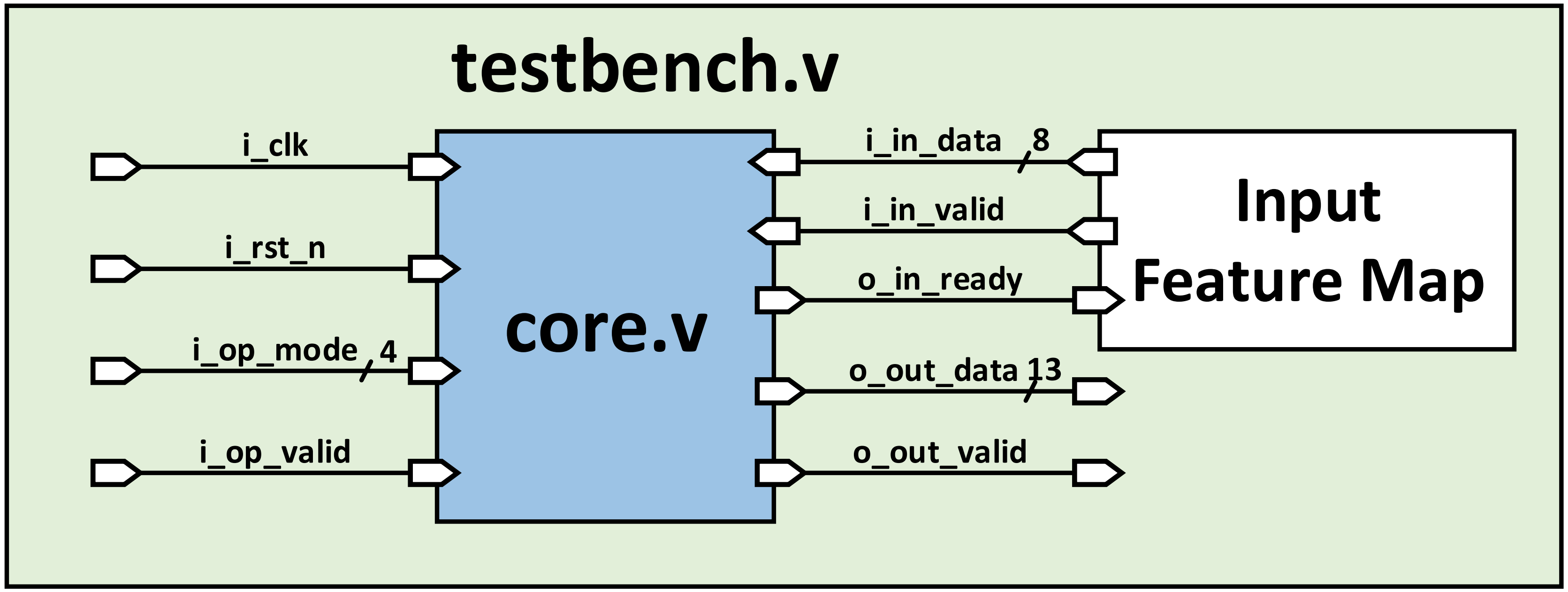
- In this homework, you will learn
  - How to synthesis your design
  - How to run gate-level simulation
  - How to use SRAM

Introduction

- In this homework, you are going to implement a simplified convolution engine with some simple functions. 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	13	Pixel data or convolution result (**unsigned**)

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_ready can't be **high** in the same time

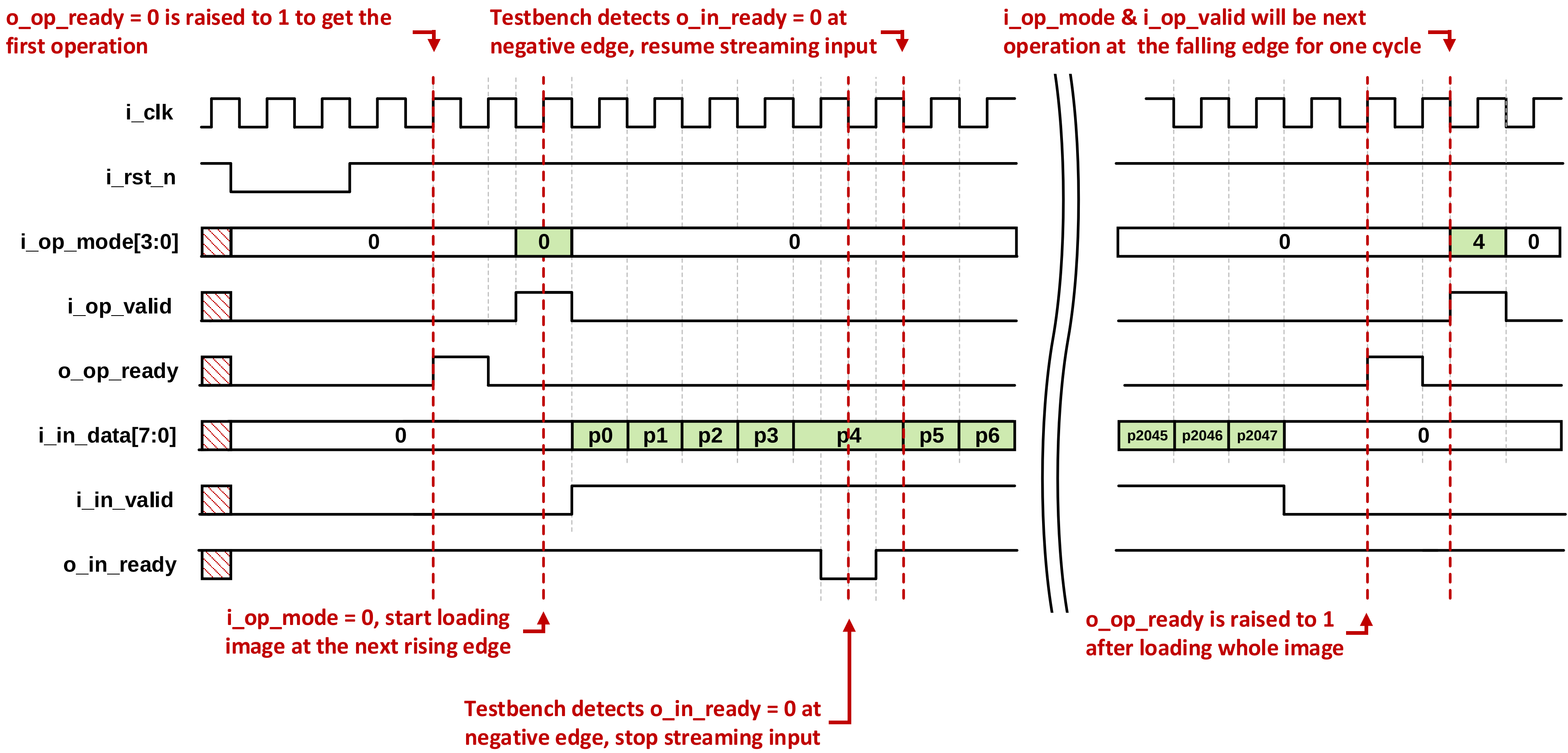
Specification(3)

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

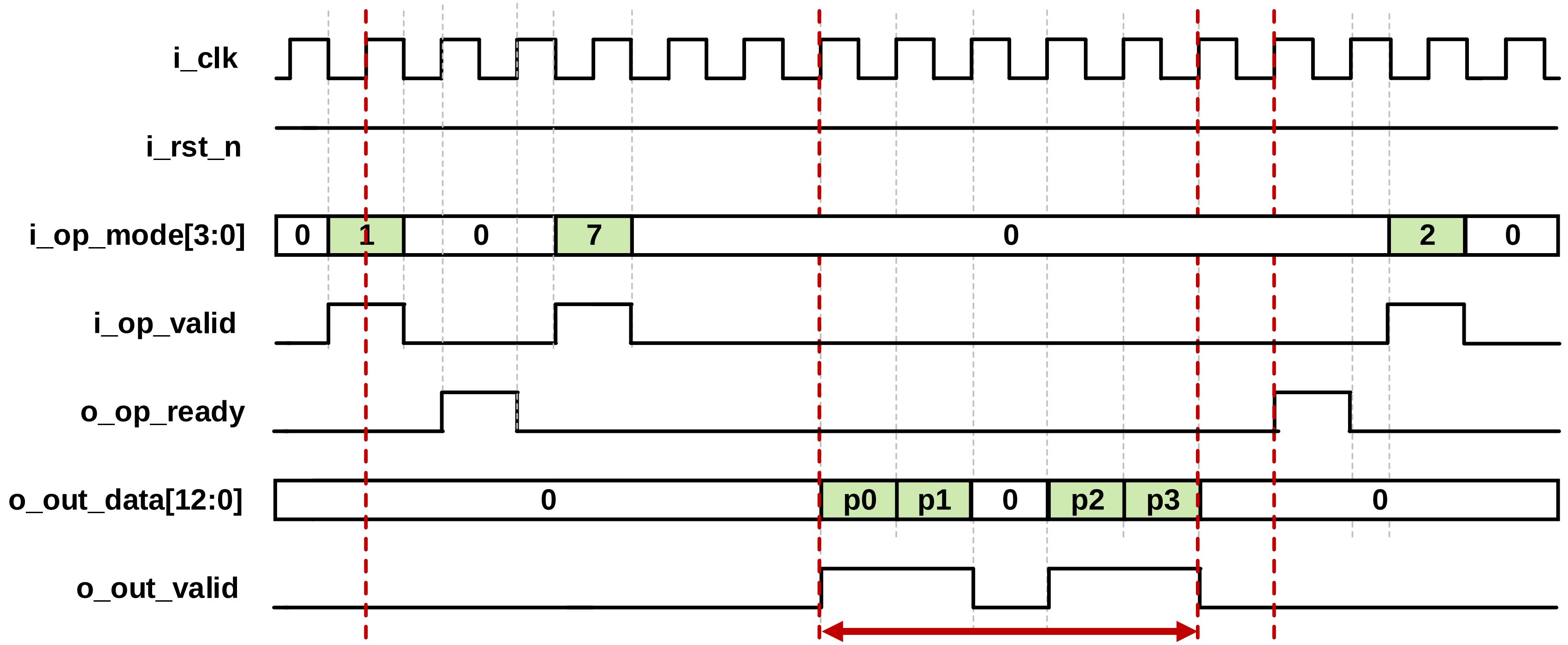
Specification(4)

- 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.

Waveform: Loading Image


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_clk
i_rst_n
i_op_mode[3:0]
0
0
0
0
4
0
i_op_valid
o_op_ready
i_in_data[7:0]
0
p0
p1
p2
p3
p4
p5
p6
p2045
p2046
p2047
0
i_in_valid
o_in_ready
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

Waveform: Other Operations



o_out_valid is raised to 1 when ready to output result

o_out_valid is 1 for your outputs

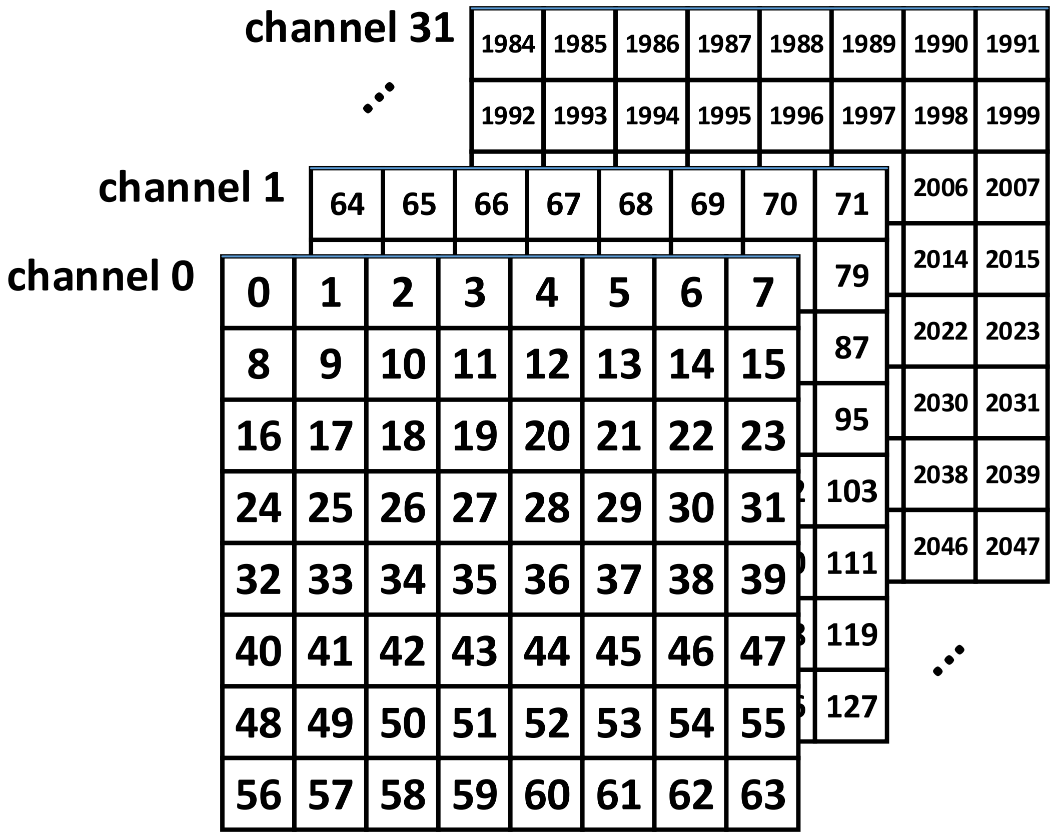
o_op_ready is raised to 1 when ready to get next operation

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 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

Input Image

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

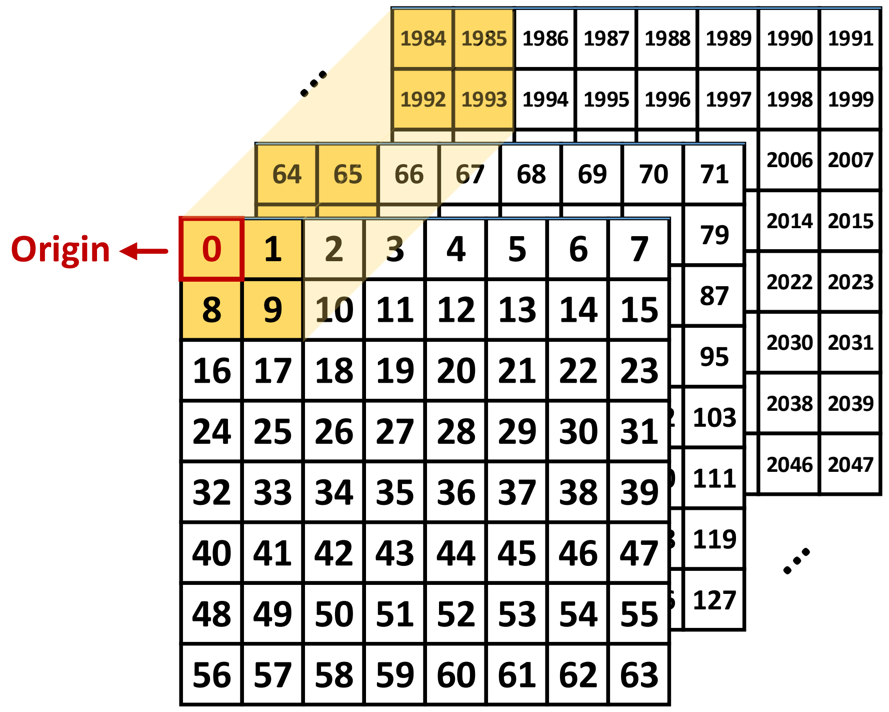


Input Image Loading

- An 8x8x32 image is loaded for 2048 cycles in **raster-scan** order
- The size of each pixel is 8 bits
- 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

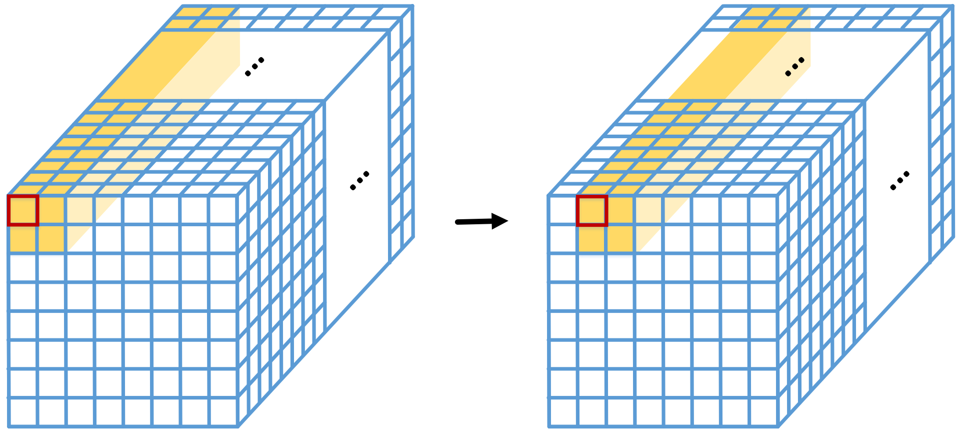
Origin

- 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 × 2 × depth

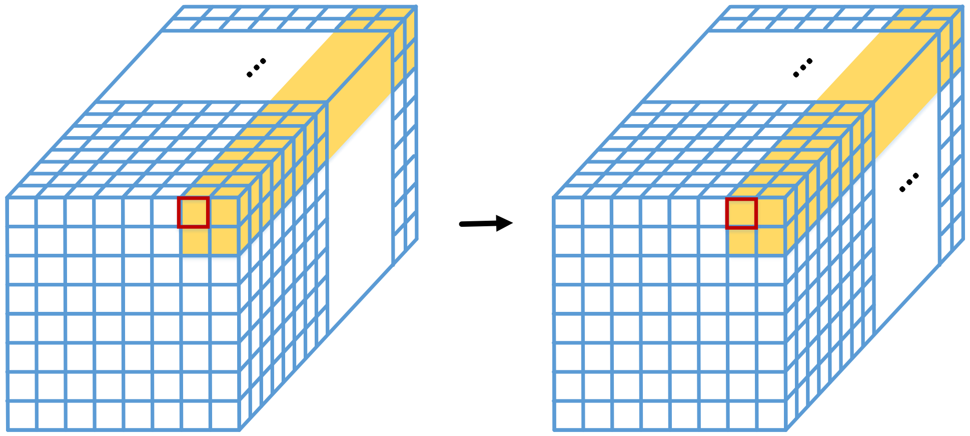


Origin Shifting

- Origin right shift (i_op_mode = 4’b0100)



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



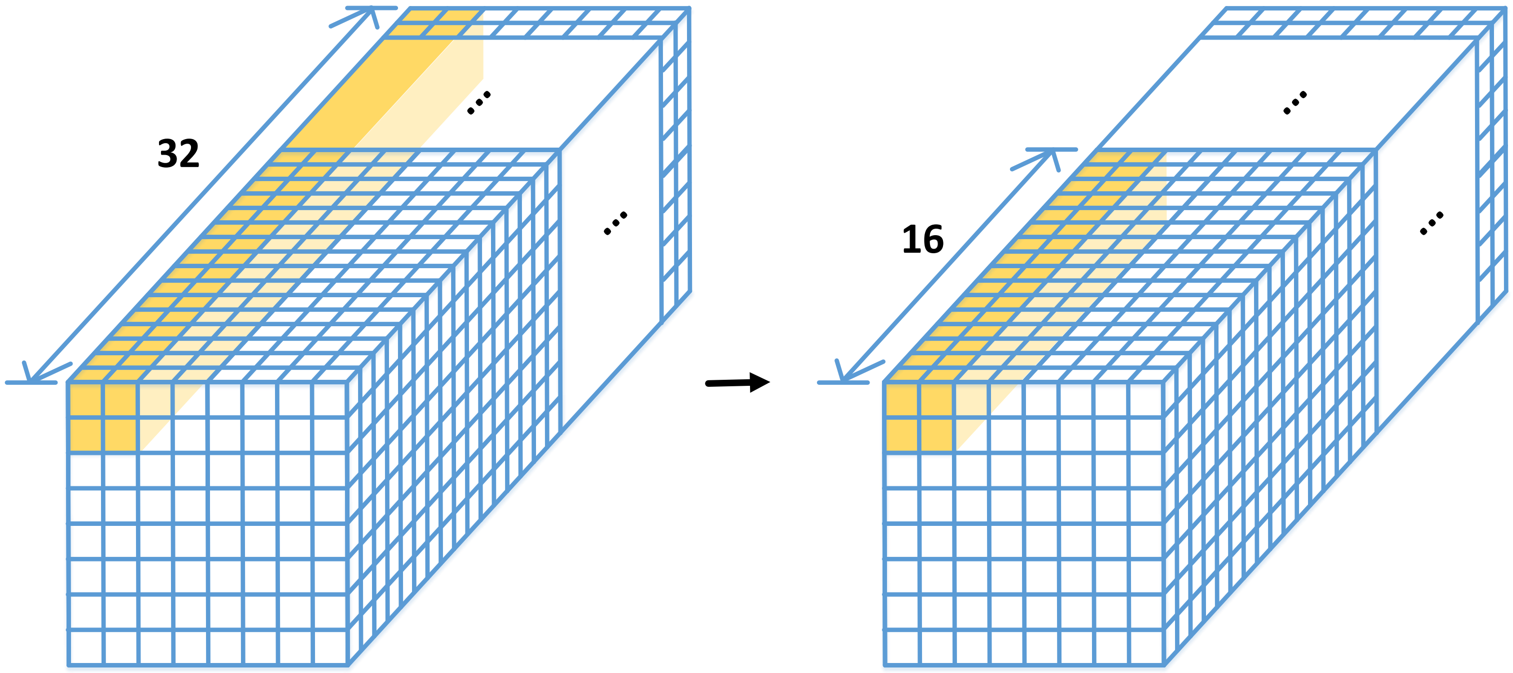
Channel Depth

- 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

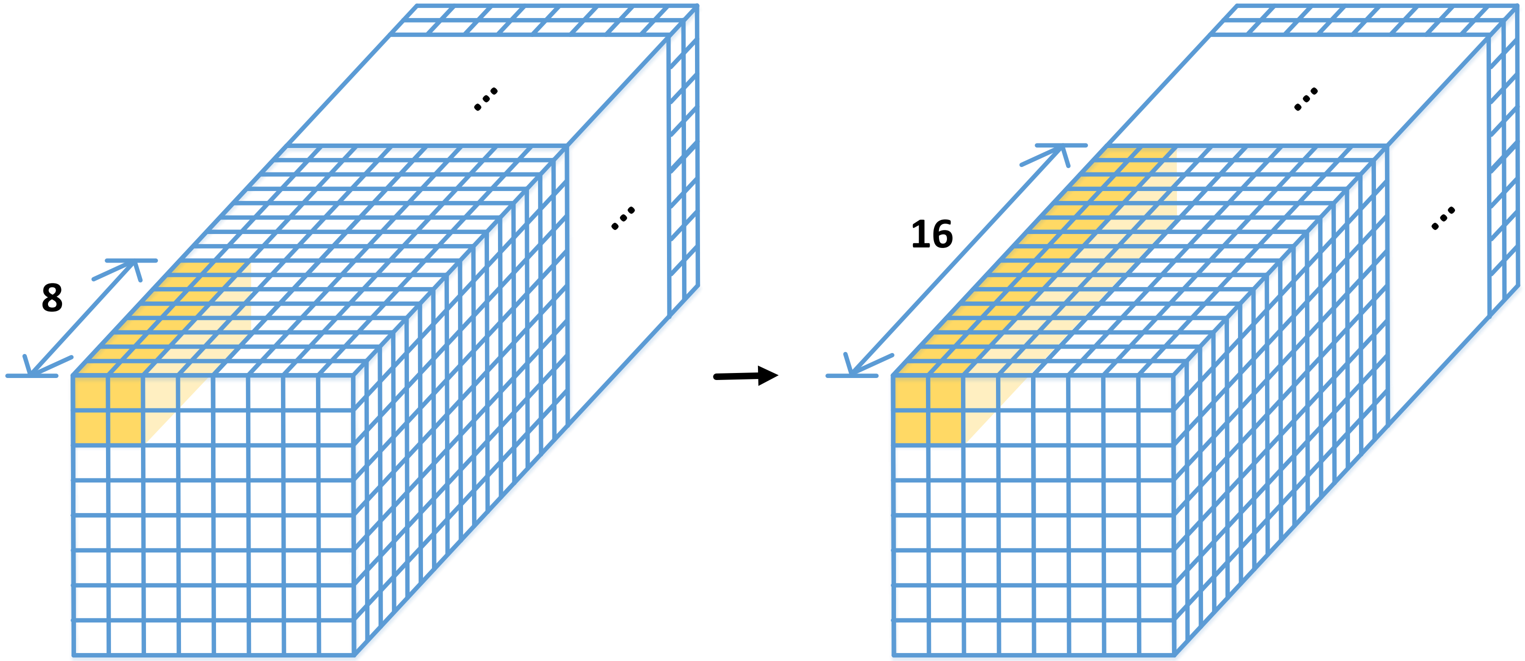
Scale-down

- Reduce the number of channels for convolution
  - Ex. For channel depth, 32 → 16 → 8
- If the depth is 8, retain the same depth



Scale-up

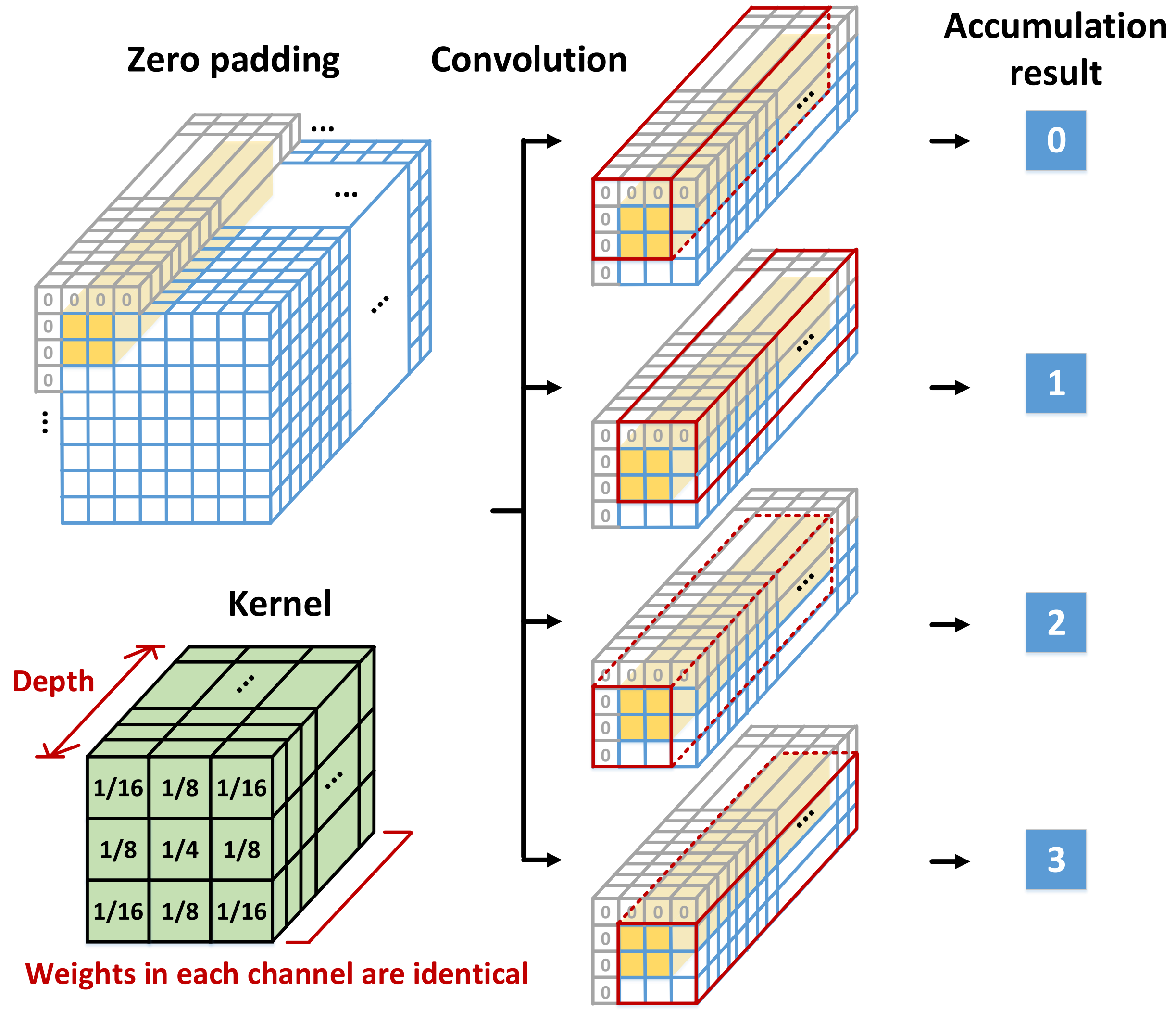
- Increase the number of channels for convolution
  - Ex. For channel depth, 8 → 16 → 32
- If the depth is 32, retain the same depth



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 (四捨五入) to integer.**
- After the convolution, you have to output the **4** accumulation results in **raster-scan** order
- The values of original pixels will not be changed

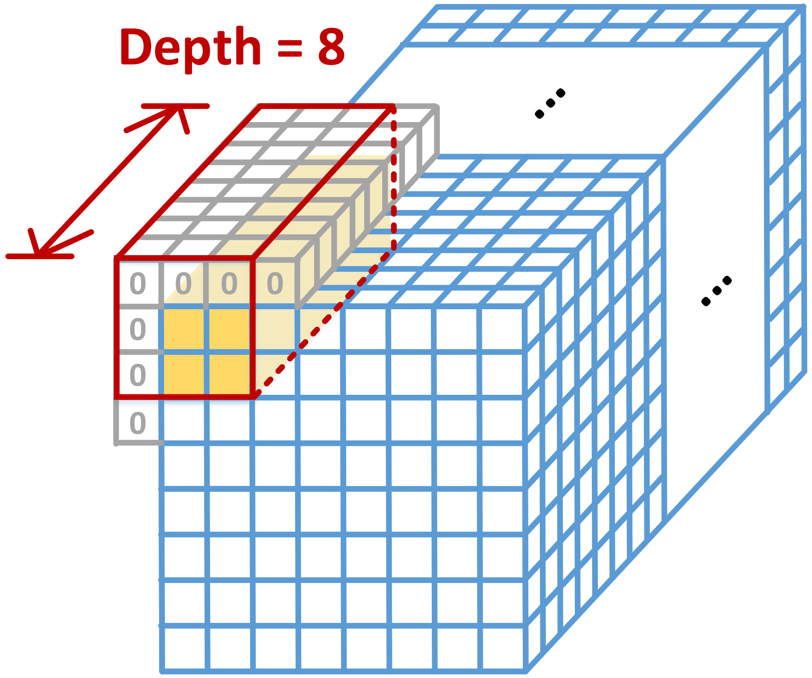
Example of Convolution



Weights in each channel are identical

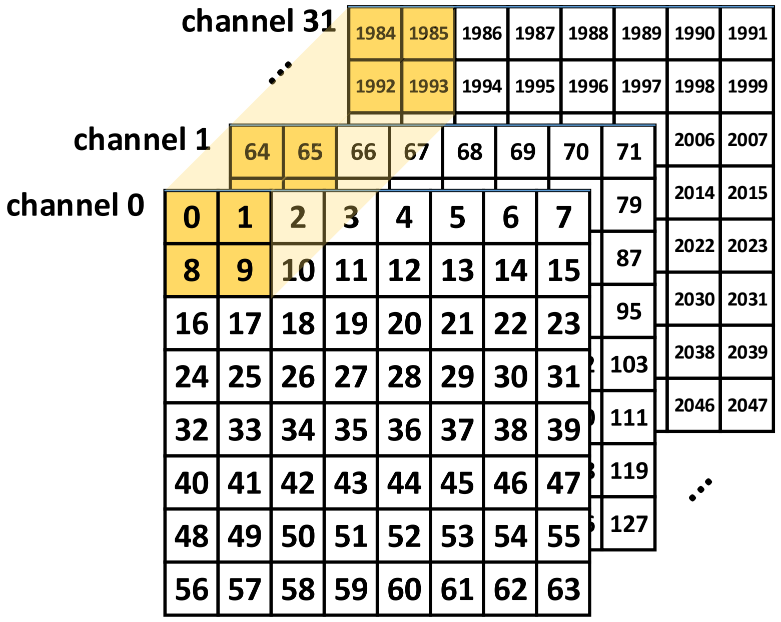
Example of Convolution

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



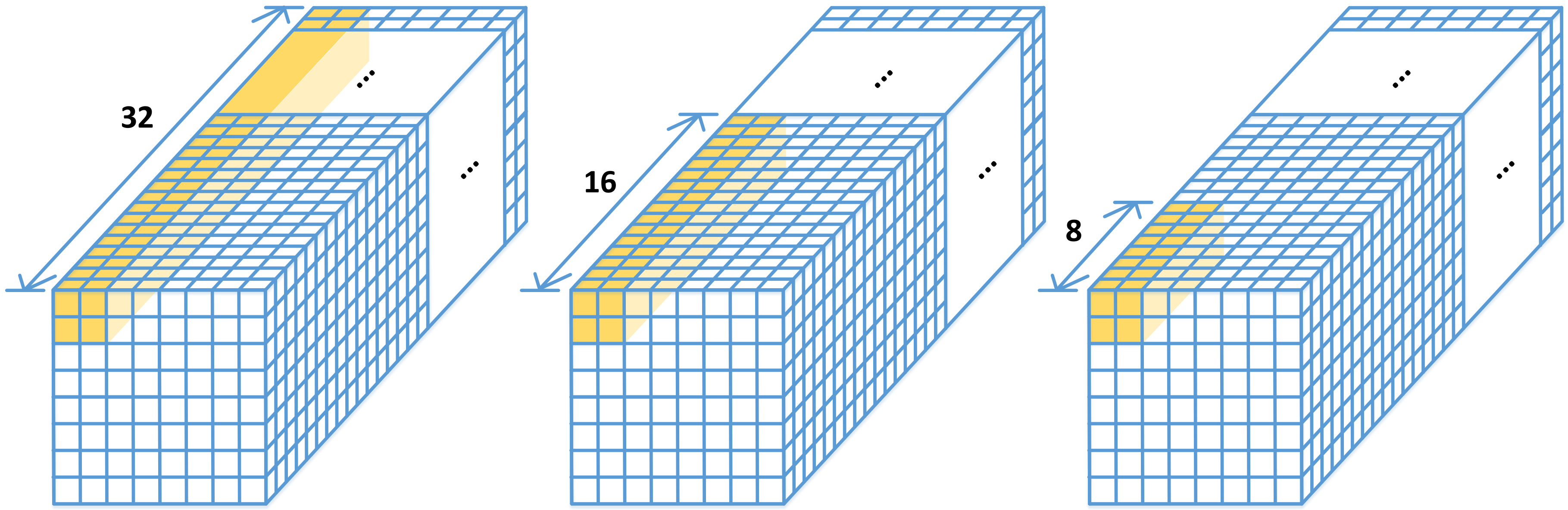
Display

- 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 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 depth



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 "./PATTERN/indata1.dat"
    `define OPFILE "./PATTERN/opmode1.dat"
    `define GOLDEN "./PATTERN/golden1.dat"
`elsif tb2
    `define INFILE "./PATTERN/indata2.dat"
    `define OPFILE "./PATTERN/opmode2.dat"
    `define GOLDEN "./PATTERN/golden2.dat"
`elsif tb3
    `define INFILE "./PATTERN/indata3.dat"
    `define OPFILE "./PATTERN/opmode3.dat"
    `define GOLDEN "./PATTERN/golden3.dat"
`else
    `define INFILE "./PATTERN/indata0.dat"
    `define OPFILE "./PATTERN/opmode0.dat"
    `define GOLDEN "./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

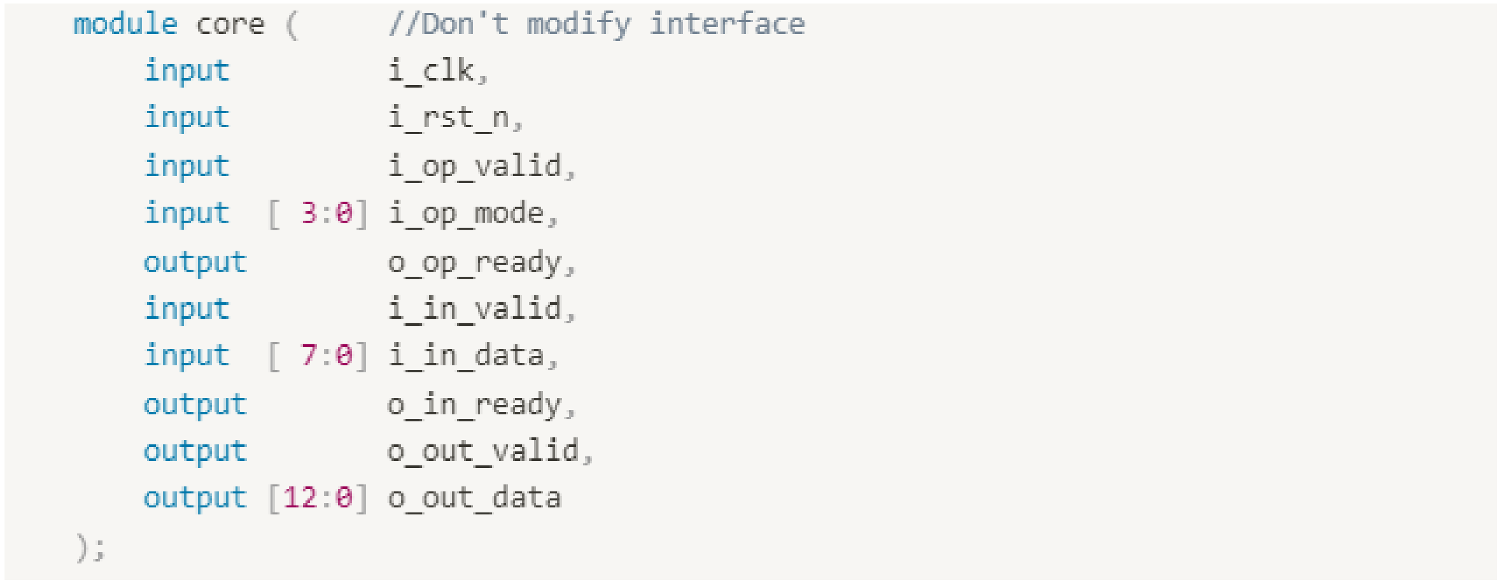
```
0000
0111
0111
0001
0111
0100
0001
0010
0011
0111
0111
0001
0111
0111
0011
0111
0001
0011
0111
0100
```

golden*.dat

```
0000000001000
0000000001001
0000000010000
0000000010001
0000001001000
0000001001001
0000001010000
0000001010001
0000010001000
0000010001001
0000010010000
0000010010001
0000011001000
0000011001001
0000011010000
0000011010001
0000000001000
0000000001001
0000000010000
0000000010001
```

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    [12:0]  o_out_data
);
```

- Run the RTL simulation under 01_RTL folder

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

tb0, tb1, tb2, tb3

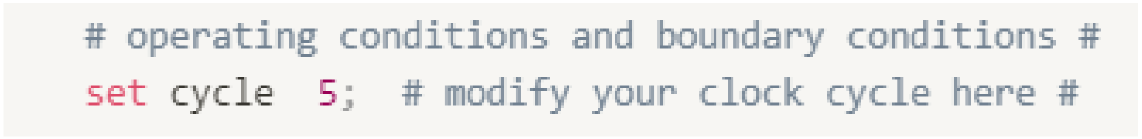
rtl_01.f

- **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;  # 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

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

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**
  - **all other design files** included in your file list (optional)
  - **rtl_01.f**
  - **rtl_03.f**
- 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 + 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%

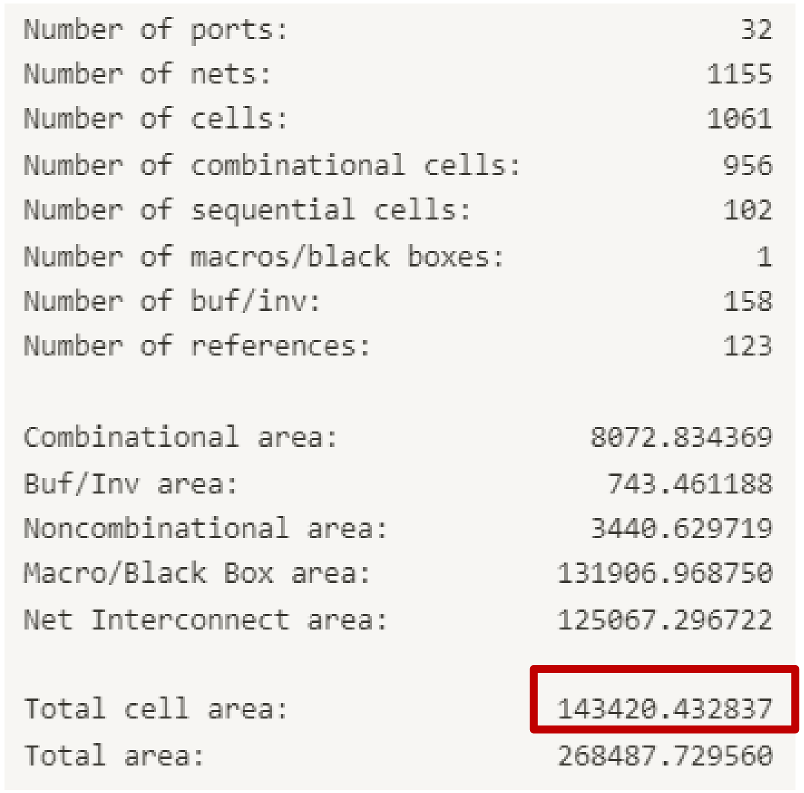
- Performance: **30%** (correct for all patterns)
  - Performance = **Area * Time (μm^2 * ns)**
    (Lower number is better performance)
  - The latency will be subtracted with **2048 cycles**

Grading Policy

- 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
- Lose **3 point** for any wrong naming rule or format for submission

Area

- **core_syn.area**



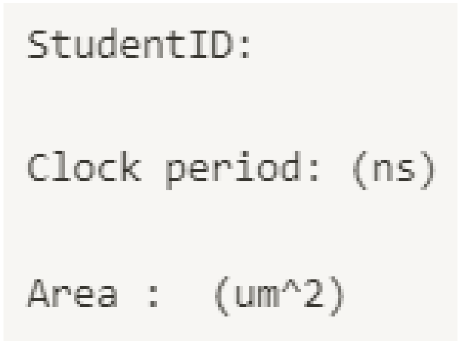
```
Number of ports:                          32
Number of nets:                         1155
Number of cells:                        1061
Number of combinational cells:           956
Number of sequential cells:              102
Number of macros/black boxes:              1
Number of buf/inv:                       158
Number of references:                    123

Combinational area:              8072.834369
Buf/Inv area:                     743.461188
Noncombinational area:           3440.629719
Macro/Black Box area:          131906.968750
Net Interconnect area:         125067.296722

Total cell area:               143420.432837
Total area:                    268487.729560
```

143420.432837 μm^2

Report

- TAs will run your design with your clock period

- **report.txt**



```
StudentID:

Clock period: (ns)

Area :  (um^2)
```

DesignWare

- Document

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

- 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
```

- 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
```