Computer-Aided VLSI System Design

Homework 3: Simple Image Processing and Display Controller

Graduate Institute of Electronics Engineering, National Taiwan University

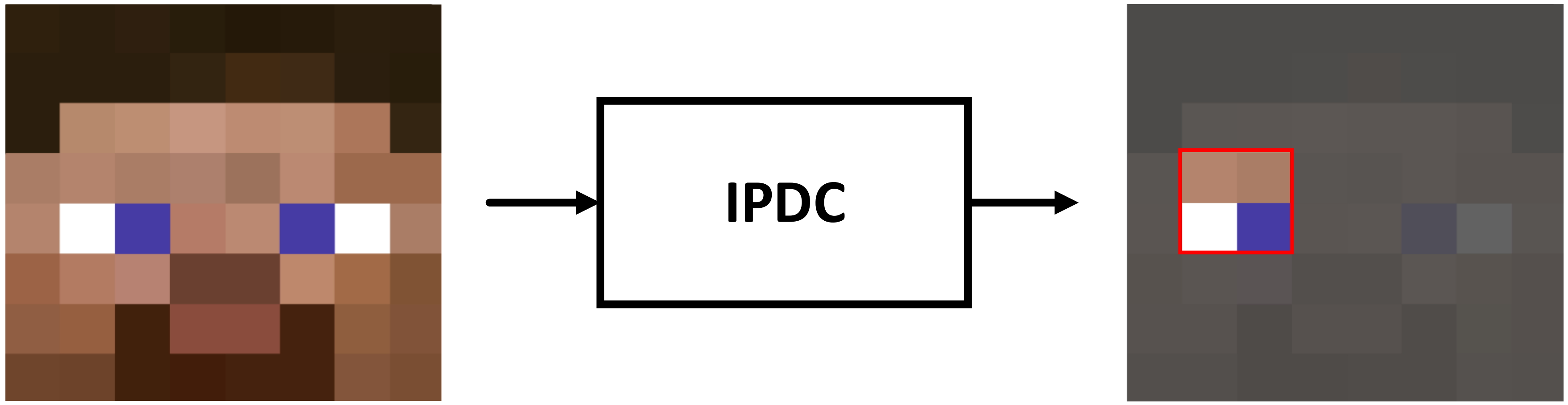
NTU GIEE

Goal

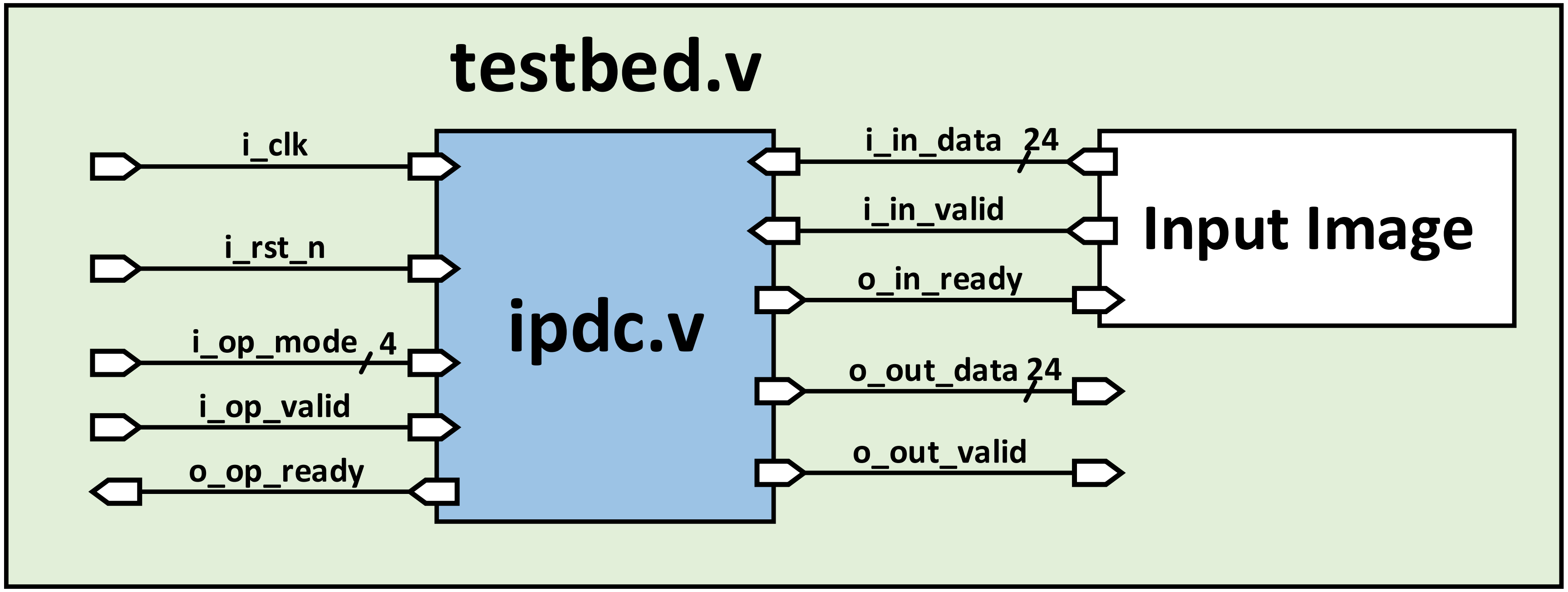
- In this homework, you will learn
  - How to synthesis your design
  - How to run gate-level simulation
  - How to use SRAM

Introduction

- Image display is a useful feature for the consumer electronics. In this homework, you are going to implement an image display controller with some simple functions. An 16×16 image 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	24	Input pixel data (RGB, **unsigned**) i_in_data [23:16] →R i_in_data [15:8] →G i_in_data [7:0] →B
o_in_ready	O	1	Set **high** if ready to get next input data (only valid for i_op_mode = 3'b000)
o_out_valid	O	1	Set **high** with valid output data
o_out_data	O	24	Output pixel data (RGB or YCbCr, **unsigned**) o_out_data [23:16] →R or Y o_out_data [15:8] →G or Cb o_out_data [7:0] →B or Cr

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 [3:0] 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

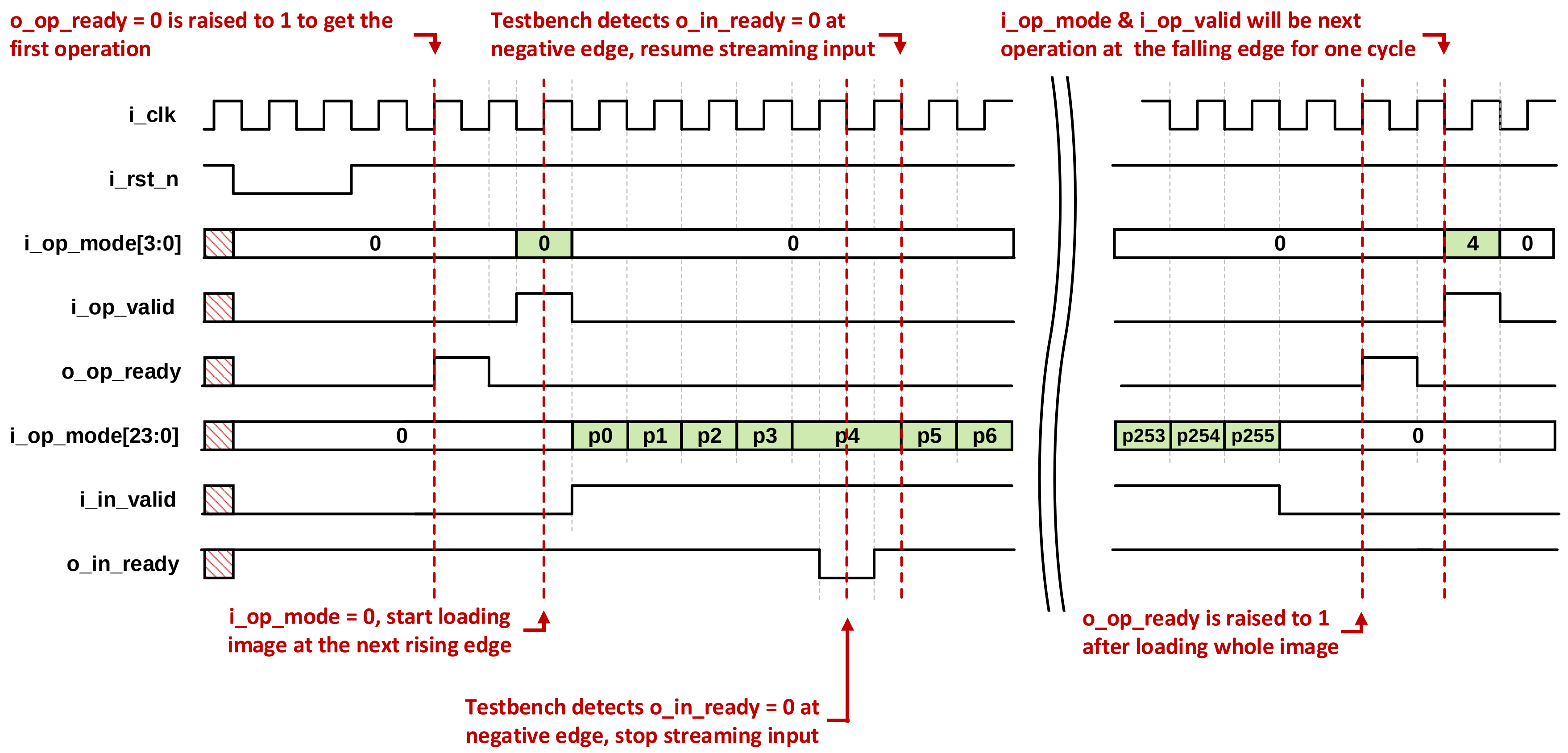
Specification(3)

- Set o_op_ready to **high** to get next operation (only one cycle)
- o_out_valid should be **high** for output results
- Latency between i_op_valid and o_op_ready should be less than **1000ns** (except i_op_mode = 4’b0000)
- **At least one SRAM** is implemented in your design

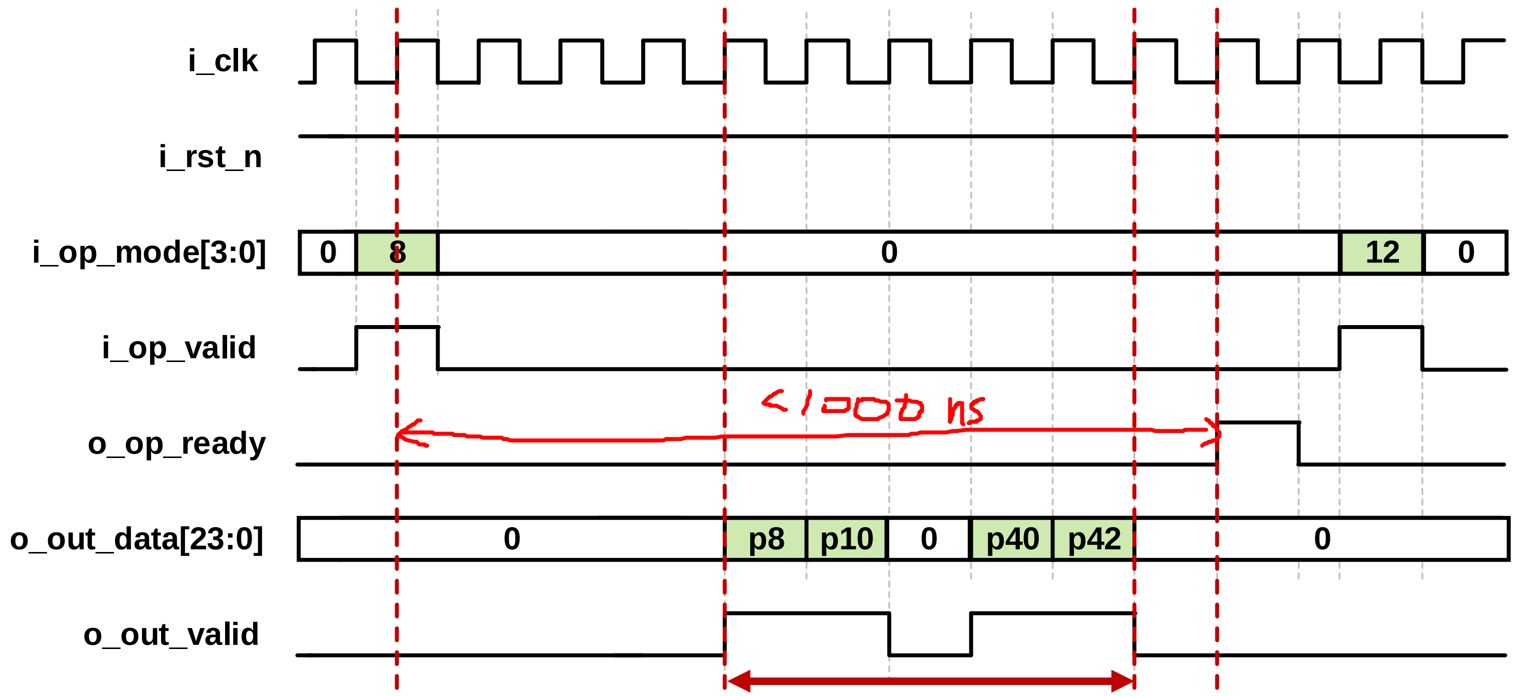
Specification(4)

- 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_op_mode[23:0]
0
p0
p1
p2
p3
p4
p5
p6
p253
p254
p255
0
i_in_valid
o_in_ready
i_op_mode = 0, start loading image at the next rising edge
Testbench detects o_in_ready = 0 at negative edge, stop streaming input
o_op_ready is raised to 1 after loading whole image

Waveform: Other Operations



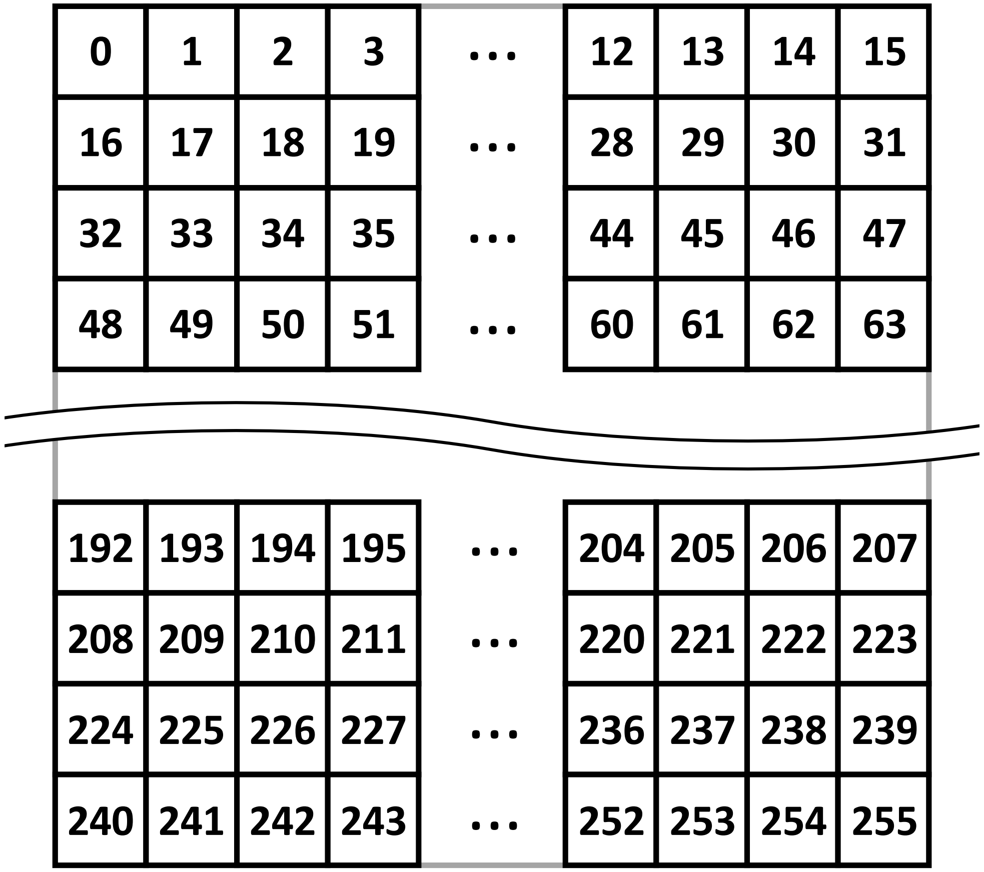
o_out_valid is raised to 1 when ready to output result

o_out_valid is 1 for four outputs

o_op_ready is raised to 1 when ready to get next operation

Input Image

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



0	1	2	3	…	12	13	14	15
16	17	18	19	…	28	29	30	31
32	33	34	35	…	44	45	46	47
48	49	50	51	…	60	61	62	63
192	193	194	195	…	204	205	206	207
208	209	210	211	…	220	221	222	223
224	225	226	227	…	236	237	238	239
240	241	242	243	…	252	253	254	255

Operation Modes

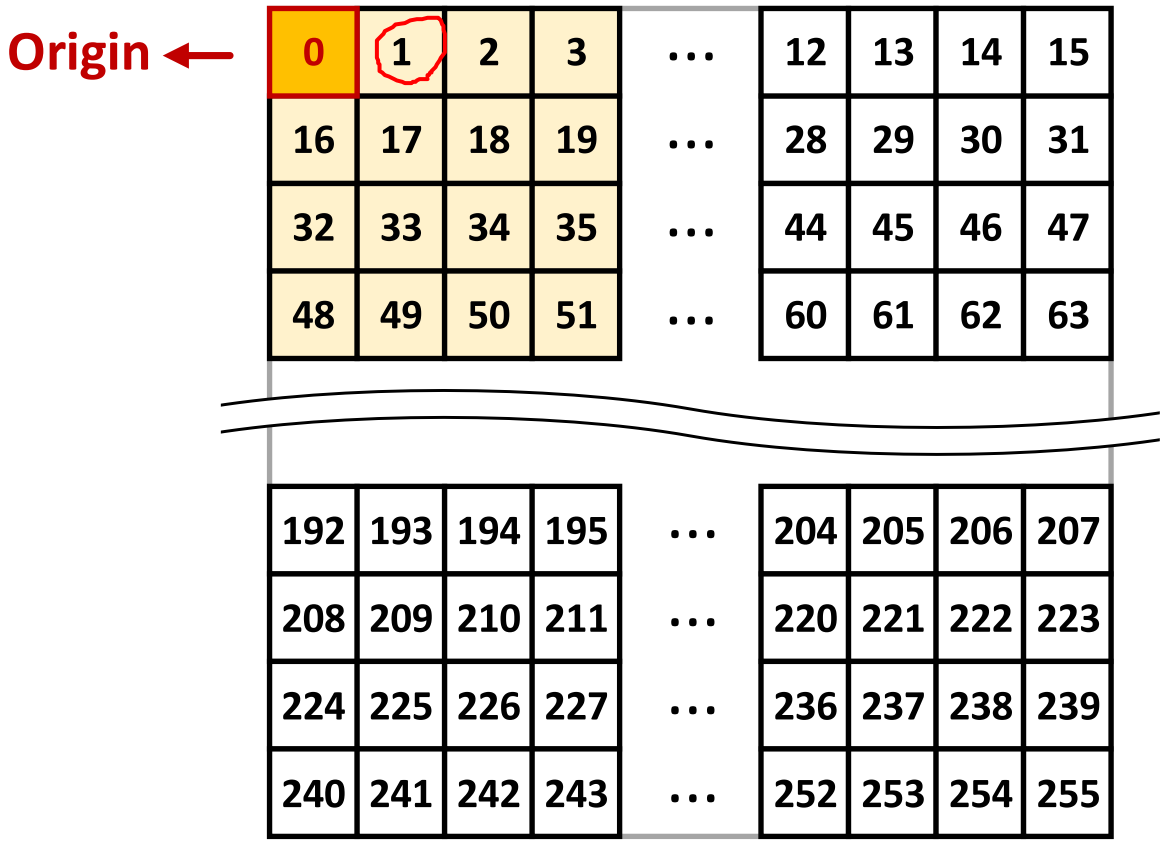
Operation Mode i_op_mode	Meaning	Need to display?
4’b0000	Input image loading	No
4’b0100	Origin right shift	Yes
4’b0101	Origin left shift	Yes
4’b0110	Origin up shift	Yes
4’b0111	Origin down shift	Yes
4’b1000	Scale-down	Yes
4’b1001	Scale-up	Yes
4’b1100	Median filter operation	Yes
4’b1101	YCbCr display	Yes
4’b1110	Census transform	Yes

Input Image Loading

- An 16×16×3 image is loaded for 256 cycles in **raster-scan** order.
- The pixel is in RGB type, and the size of each pixel is 24 bis.
- 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.

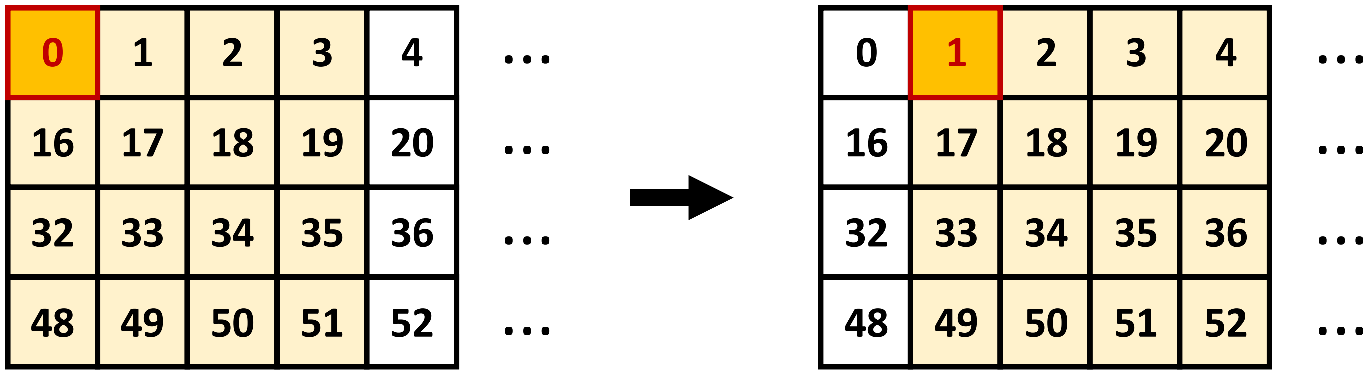
Origin

- The first output pixel of the display is **origin**
- The default coordinate of the origin is at 0



Origin Shifting

- Origin right shift (i_op_mode = 4’b0100)



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

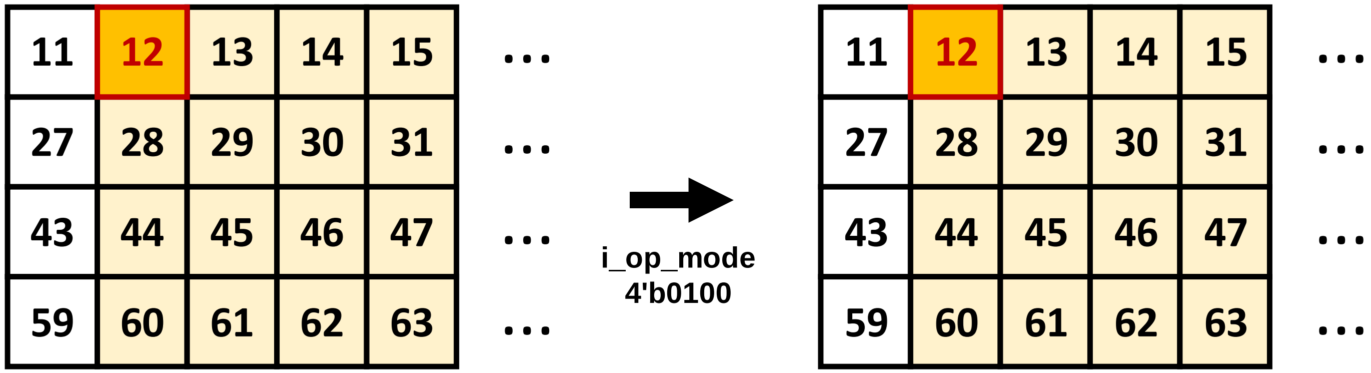


Image Size

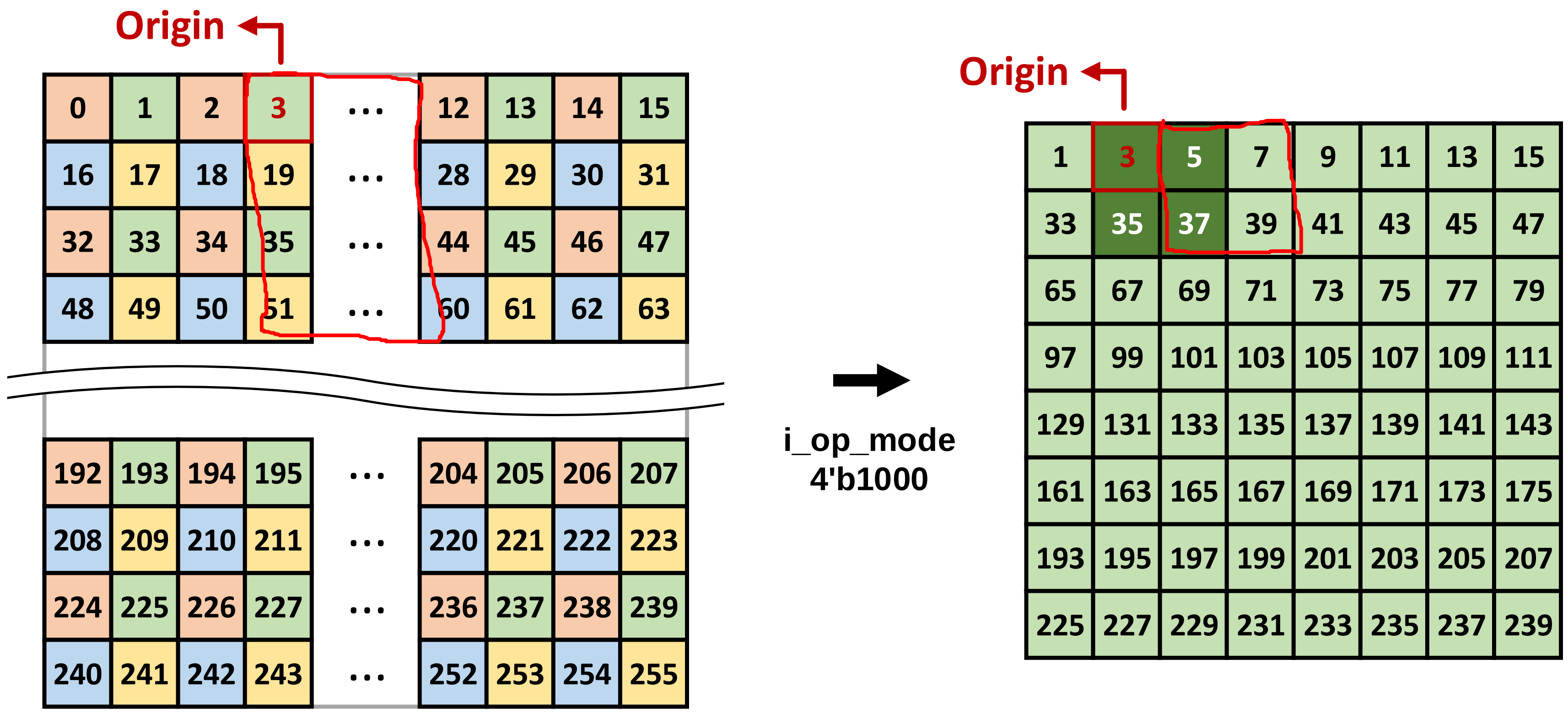
- 3 image sizes are considered in this design
  - 16x16, 8x8, 4x4
- For output display, the display size will change with different image size

Image size	Display size
16 x 16	4 x 4
8 x 8	2 x 2
4 x 4	1 x 1

Scale-down

- Scale down both image size and display size to next level
  - Ex. For image size,
    - 16x16 -> 8x8
    - 8x8 -> 4x4
- If the image size is 4x4, retain the same image size and display size
- Scale down the size with the location of the origin

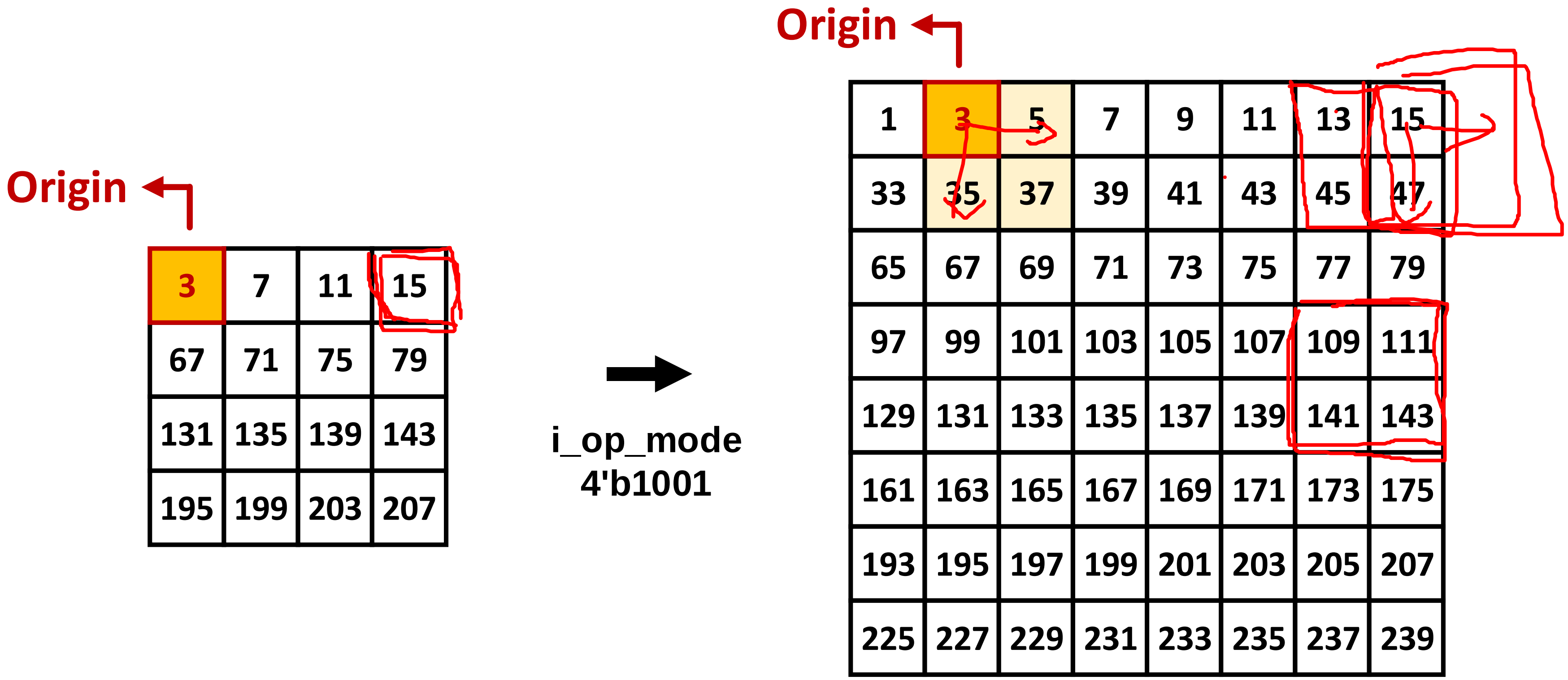
Example of Scale-down



Scale-up

- Scale up both image size and display size to next level
  - Ex. For image size,
    - 4x4 -> 8x8
    - 8x8 -> 16x16
- If the image size is 16x16, retain the same image size and display size
- Scale up the image with the related image when scaling down
- Scale up the size of display with the location of the origin

Example of Scale-up



3	7	11	15
67	71	75	79
131	135	139	143
195	199	203	207

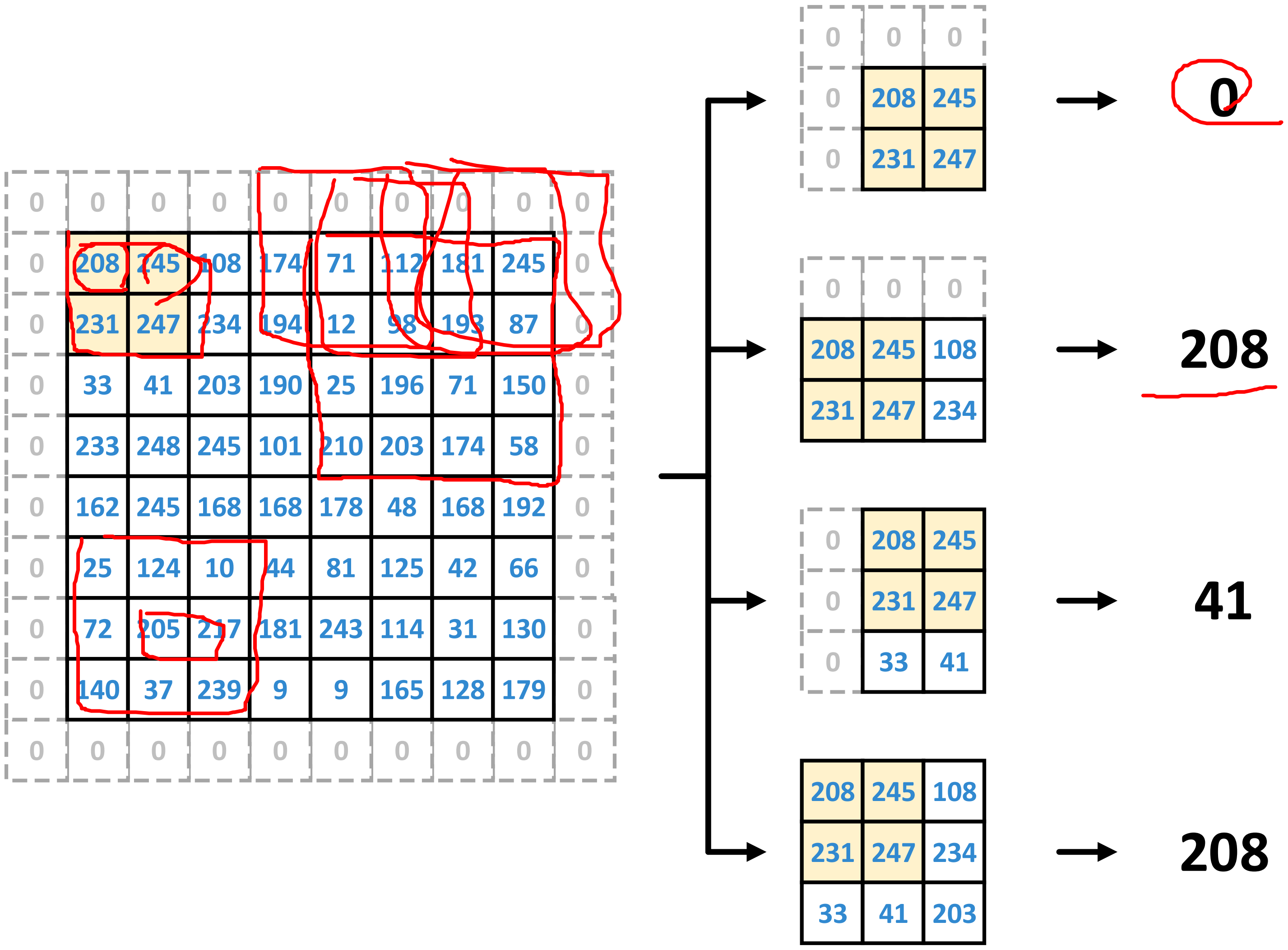
1	3	5	7	9	11	13	15
33	35	37	39	41	43	45	47
65	67	69	71	73	75	77	79
97	99	101	103	105	107	109	111
129	131	133	135	137	139	141	143
161	163	165	167	169	171	173	175
193	195	197	199	201	203	205	207
225	227	229	231	233	235	237	239

- If the display region is located furthest to the right, retain the same image size and display size

Median Filter Operation

- For this operation, you have to perform median filtering on the display region
- The filter is a 3x3 kernel. It results in a median of the set of pixel value
- Operate median filtering to R-channel, G-channel, B-channel, separately
- The image needs to be zero-padded for median filter operation
- The values of original pixels will not be changed

Example of Median Filter Operation



0	0	0	0	0	0	0	0	0	0
0	208	245	108	174	71	112	181	245	0
0	231	247	234	194	12	98	193	87	0
0	33	41	203	190	25	196	71	150	0
0	233	248	245	101	210	203	174	58	0
0	162	245	168	168	178	48	168	192	0
0	25	124	10	44	81	125	42	66	0
0	72	205	217	181	243	114	31	130	0
0	140	37	239	9	9	165	128	179	0
0	0	0	0	0	0	0	0	0	0

0	0	0
0	208	245
0	231	247

→ 0

0	0	0
208	245	108
231	247	234

→ 208

0	208	245
0	231	247
0	33	41

→ 41

208	245	108
231	247	234
33	41	203

→ 208

YCbCr Display

- For this operation, you have to perform YCbCr transform on the display region
- Estimated YCbCr calculation

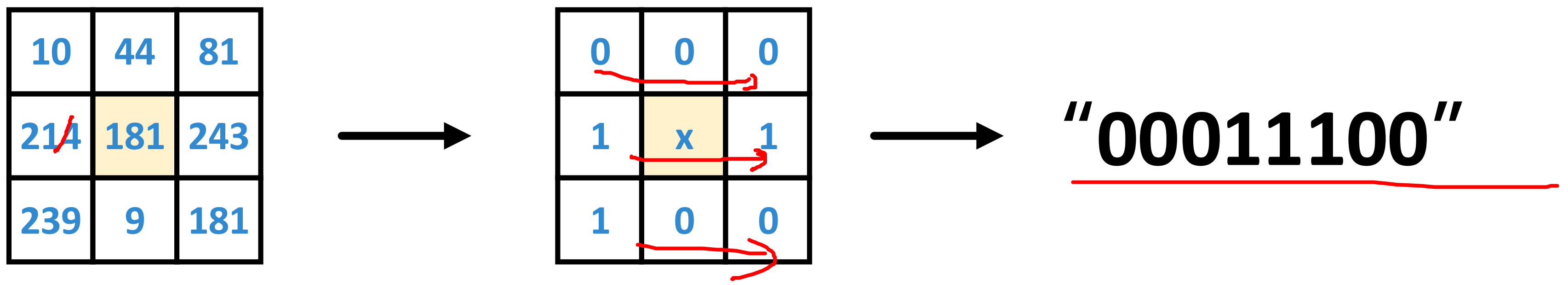
$$\boxed{\begin{aligned} \mathbf{Y} &= 0.25\mathbf{R} + 0.625\mathbf{G} \\ \mathbf{Cb} &= \underline{-0.125\mathbf{R} - 0.25\mathbf{G} + 0.5\mathbf{B} + 128} \\ \mathbf{Cr} &= 0.5\mathbf{R} - 0.375\mathbf{G} - 0.125\mathbf{B} + 128 \end{aligned}}$$

- For YCbCr, only **rounding** the result after accumulation, i.e. do not truncate temporal result during shifting
- The values of original pixels will not be changed

Census Transform

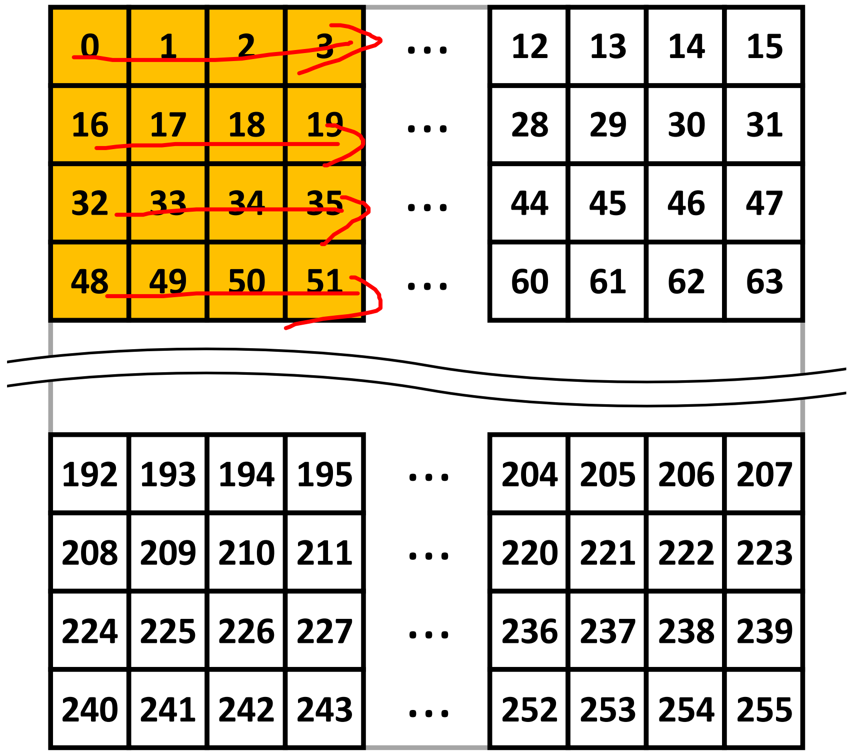
- For this operation, you have to perform Census transform on the display region
- The filter is a 3x3 kernel. It identifies the pixels with higher intensity than the center pixel
- Operate Census transform to R-channel, G-channel, B-channel, separately
- The image needs to be zero-padded for Census Transform
- The values of original pixels will not be changed

Example of Census Transform

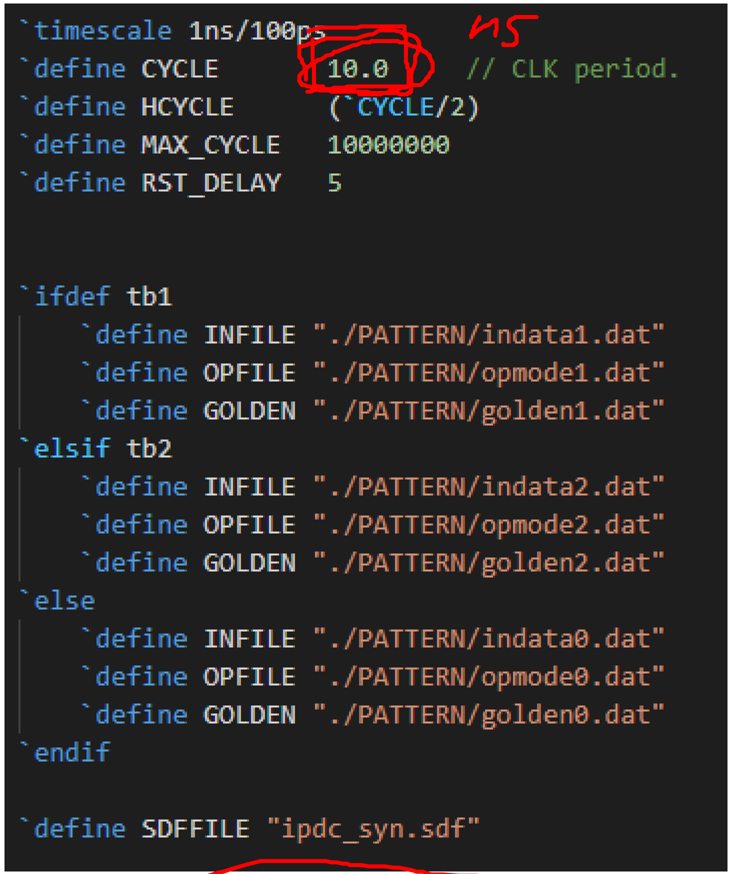


Output Display

- When output data is ready, the pixels are displayed in **raster-scan** order
  - For example: 0 → 1 → 2 → 3 → 16 → 17 → … → 48 → 49 → 50 → 51

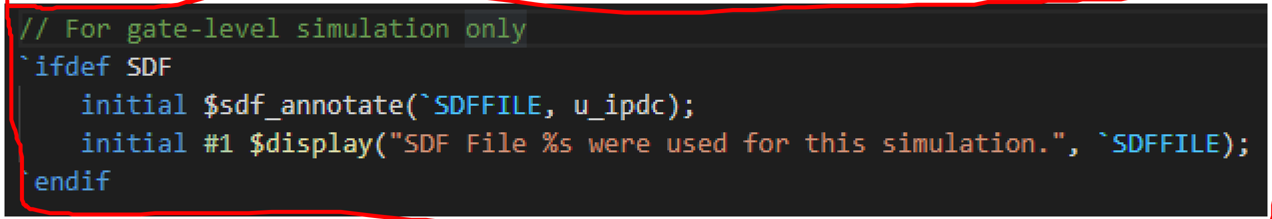


Testbench



ns

```
`timescale 1ns/100ps
`define CYCLE       10.0    // CLK period.
`define HCYCLE      (`CYCLE/2)
`define MAX_CYCLE   10000000
`define RST_DELAY   5


`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"
`else
    `define INFILE "./PATTERN/indata0.dat"
    `define OPFILE "./PATTERN/opmode0.dat"
    `define GOLDEN "./PATTERN/golden0.dat"
`endif

`define SDFFILE "ipdc_syn.sdf"
```



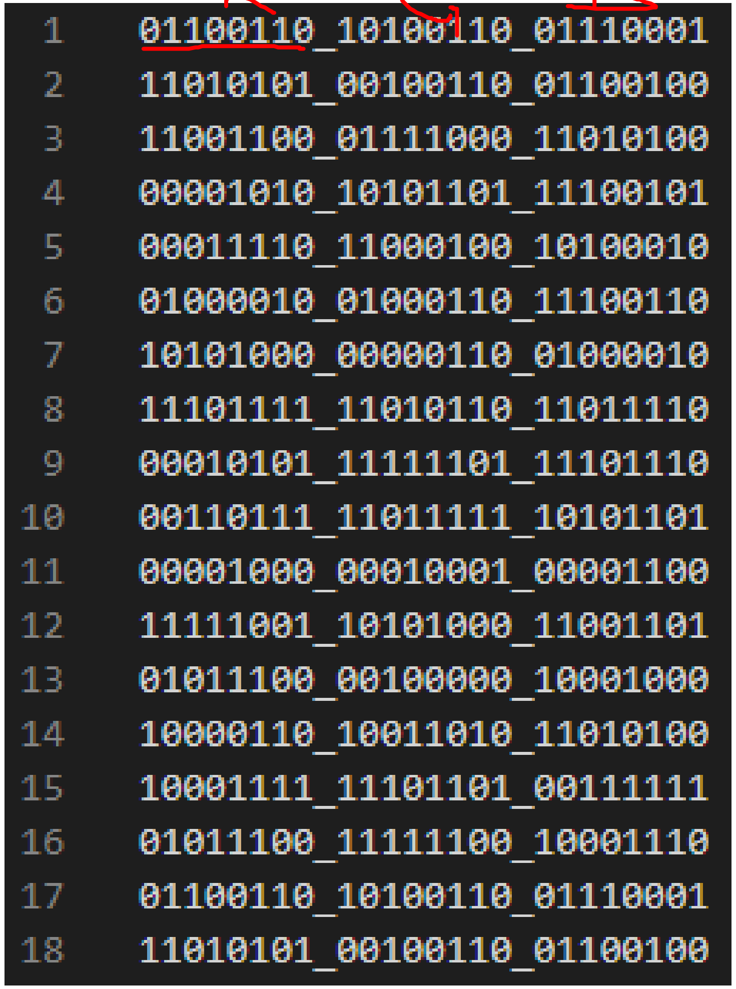
```
// For gate-level simulation only
`ifdef SDF
    initial $sdf_annotate(`SDFFILE, u_ipdc);
    initial #1 $display("SDF File %s were used for this simulation.", `SDFFILE);
`endif
```

Pattern

input

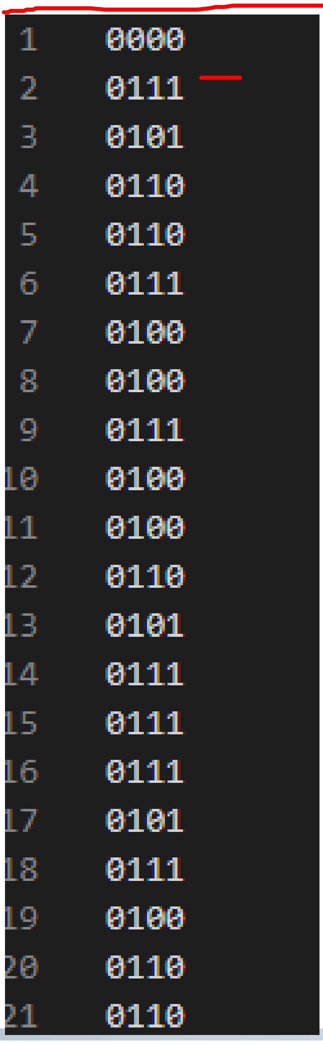
golden*.dat

R G B



```
01100110_10100110_01110001
11010101_00100110_01100100
11001100_01111000_11010100
00001010_10101101_11100101
00011110_11000100_10100010
01000010_01000110_11100110
10101000_00000110_01000010
11101111_11010110_11011110
00010101_11111101_11101110
00110111_11011111_10101101
00001000_00010001_00001100
11111001_10101000_11001101
01011100_00100000_10001000
10000110_10011010_11010100
10001111_11101101_00111111
01011100_11111100_10001110
01100110_10100110_01110001
11010101_00100110_01100100
```

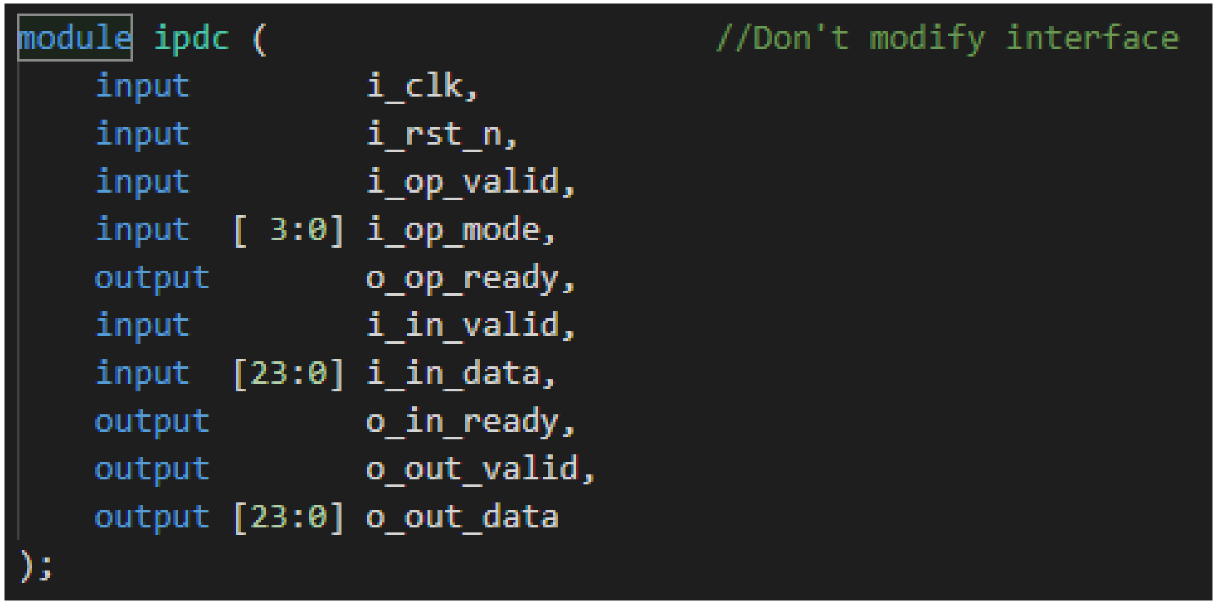
opmode*.dat



```
0000
0111
0101
0110
0110
0111
0100
0100
0111
0100
0100
0110
0101
0111
0111
0111
0101
0111
0100
0110
0110
```

01_RTL

ipdc.v



```
module ipdc (                          //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   [23:0] i_in_data,
    output         o_in_ready,
    output         o_out_valid,
    output  [23:0] o_out_data
);
```

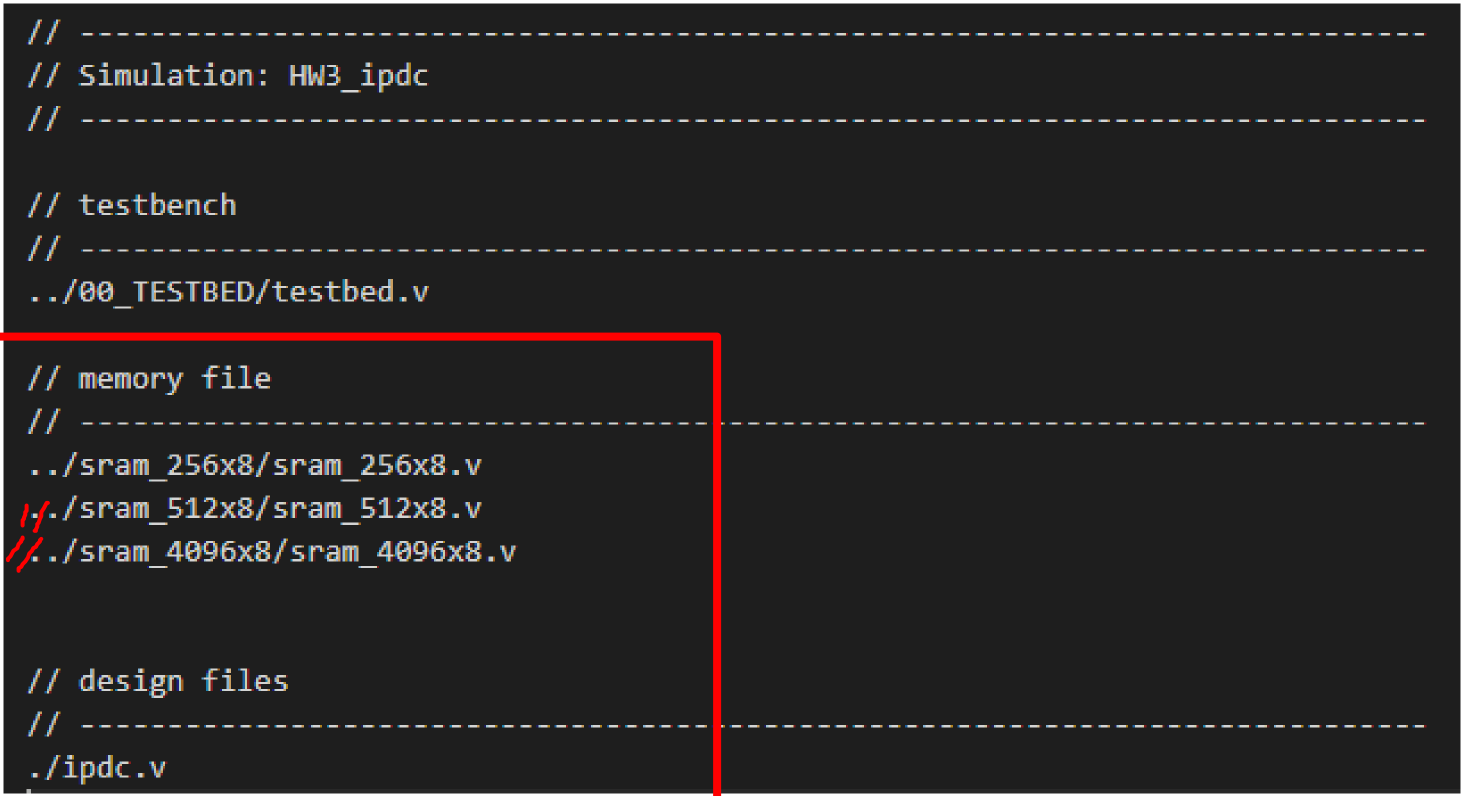
- Run the RTL simulation under 01_RTL folder

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

tb0, tb1, tb2,tb3

rtl_01.f

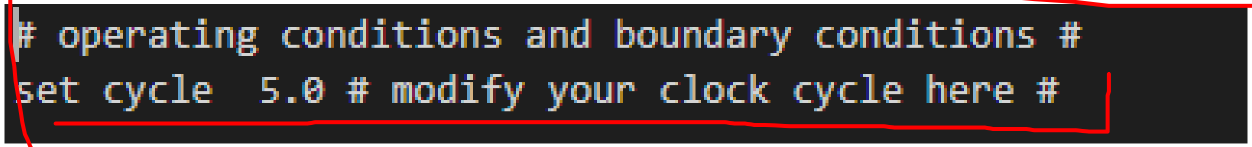
rtl_01.f



```
// -------------------------------------------------------------------------------
// Simulation: HW3_ipdc
// -------------------------------------------------------------------------------

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

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


// design files
// -------------------------------------------------------------------------------
./ipdc.v
```

02_SYN

ipdc_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

```
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
  - **ipdc.v**
  - **ipdc_syn.v**
  - **ipdc_syn.sdf**
  - **ipdc_syn.ddc**
  - **ipdc_syn.area**
  - **ipdc_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: **80%** (follow our spec)

Pattern	Description	RTL simulation	Gate-level simulation
tb0	Load+ shift	5%	5%
tb1	Load+ shift + scale	10%	10%
tb2	Load+ shift + filtering	10%	10%
tb3	All operations	10%	10%
hidden	10 hidden patterns	x	10%

- Performance: **20%** (correct for all patterns)
  - Performace = **Area (μm^2)**

    (Lower number is better performance)

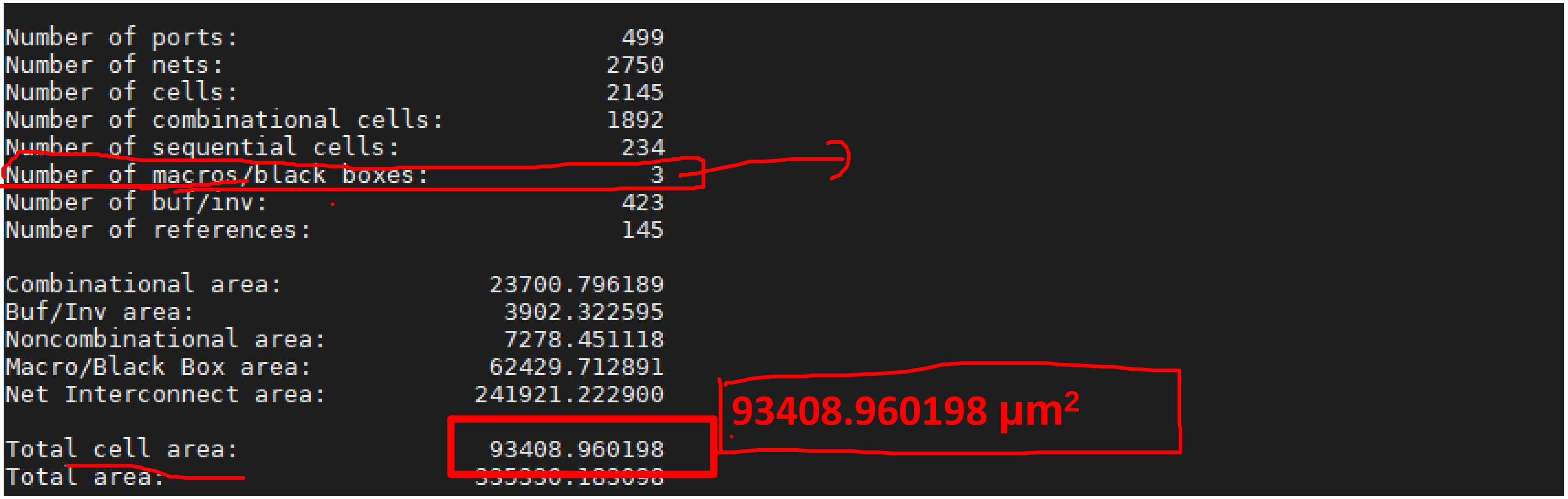
Grading Policy

- 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

ipdc_syn.area



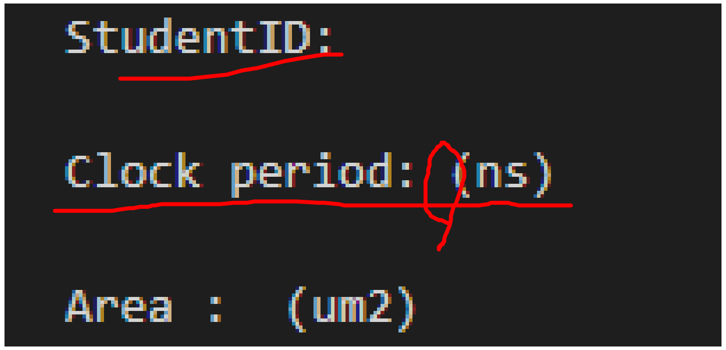
```
Number of ports:                          499
Number of nets:                          2750
Number of cells:                         2145
Number of combinational cells:           1892
Number of sequential cells:               234
Number of macros/black boxes:               3
Number of buf/inv:                        423
Number of references:                     145

Combinational area:              23700.796189
Buf/Inv area:                     3902.322595
Noncombinational area:            7278.451118
Macro/Black Box area:            62429.712891
Net Interconnect area:          241921.222900

Total cell area:                 93408.960198
Total area:                     335330.183098
```

93408.960198 μm^2

Report

- TAs will run your design with your clock period

report.txt



```
StudentID:

Clock period: (ns)

Area :  (um2)
```

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
```