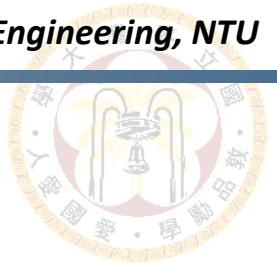
Computer-Aided VLSI System Design

Homework 3: Simple Image Processing and Display Controller

Graduate Institute of Electronics Engineering, National Taiwan University

NTU GIEE

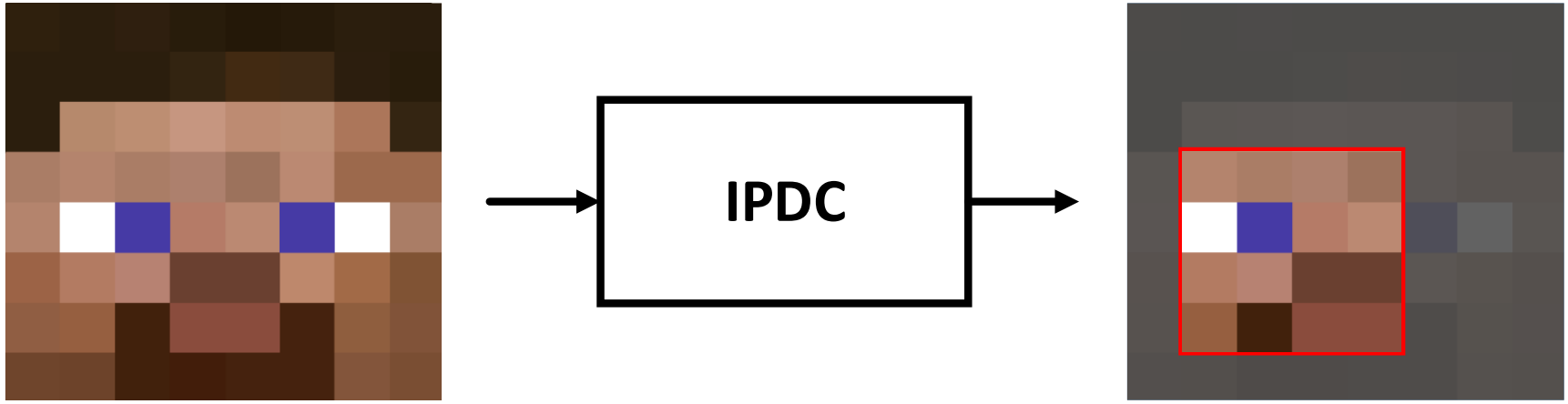


Goal

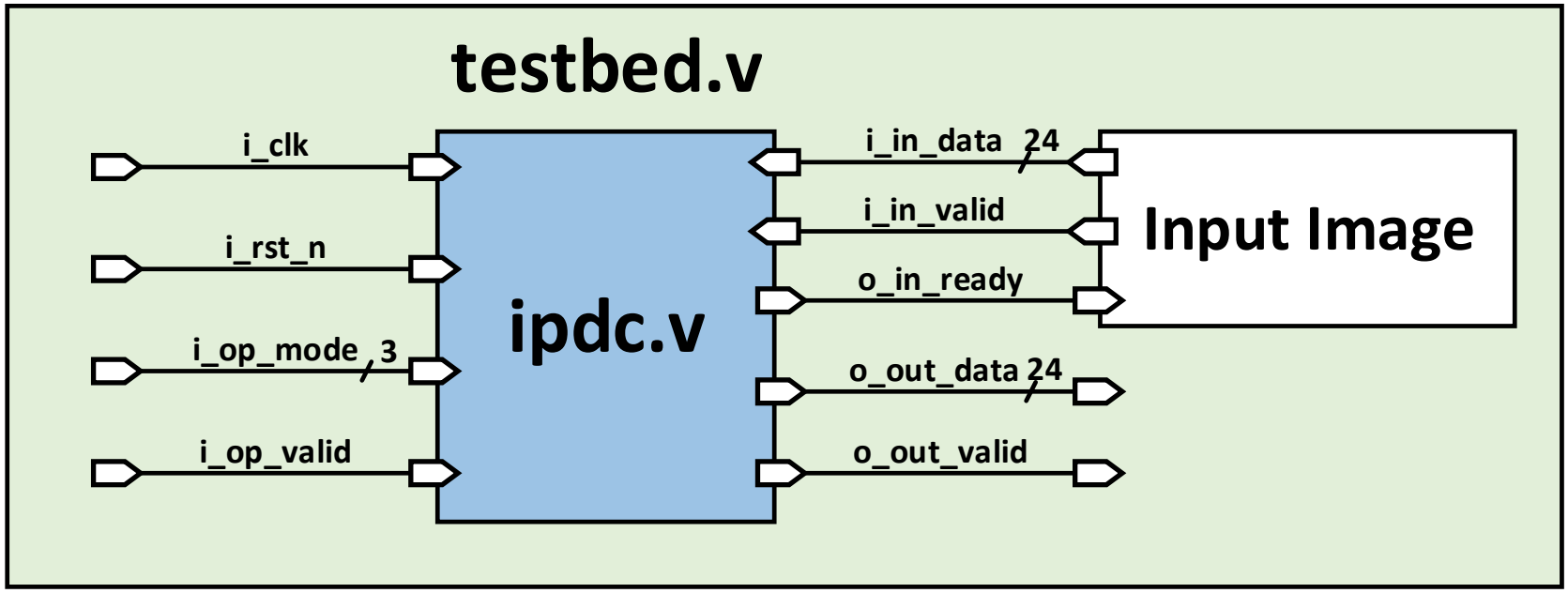
- In this homework, you will learn
  - How to synthesis your design
  - How to run gate-level simulation
  - How to use SRAM

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 8×8 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	3	Operation mode for processing
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 [7:0] →R i_in_data [15:8] →G i_in_data [23:16] →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 if ready to output result
o_out_data	O	24	Output pixel data (RGB or YCbCr, unsigned) o_out_data [7:0] →R or Y o_out_data [15:8] →G or Cb o_out_data [23:16] →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 set all your outputs and registers to be zero when **i_rst_n is low**
  - Active low asynchronous reset is used and only once

Specification(2)

- Operations are given by i_op_mode [2:0] 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.
- o_out_valid should retain **16 cycles** when displaying output image.
- **At least one 256×8 SRAM** is implemented in your design.

Specification(3)

- 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.

Input Image

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

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

Output Display

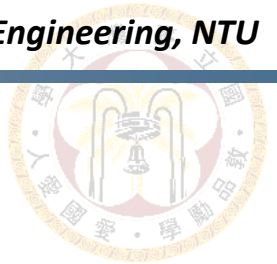
- The size of output image is 4×4×3. When output data is ready, 16 continuous sequence pixels are displayed in **raster-scan** order
  - For example: 8 → 9 → 10 → 11 → 16 → 17 → … → 32 → 33 → 34 → 35

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

Origin

- The first output pixel of the display is **origin**
- The default coordinate of the origin is at 0
  - For example: 8 is the origin for following output display

	0	1	2	3	4	5	6	7
Origin ←	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



Operation Modes

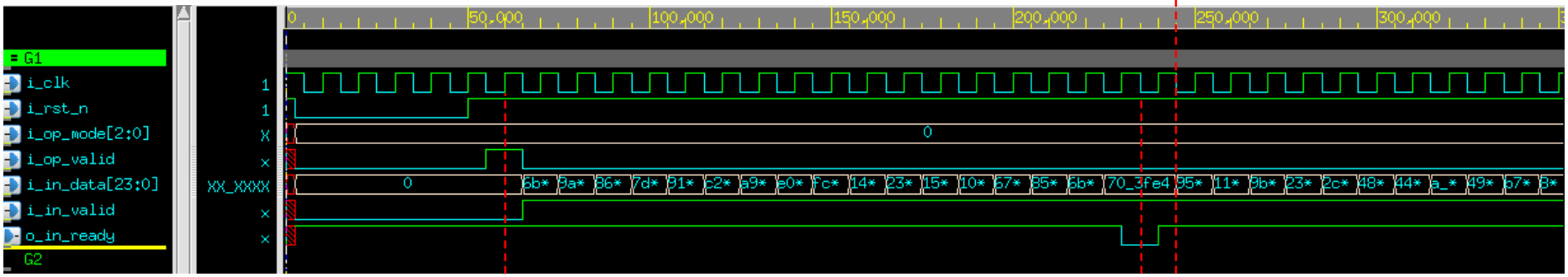
Operation Mode i_op_mode	Meaning	Need to display?
3'b000	Input image loading	No
3'b001	Origin right shift	Yes
3'b010	Origin down shift	Yes
3'b011	Default origin	Yes
3'b100	Zoom-in	Yes
3'b101	Median filter operation	No
3'b110	YCbCr display	No
3'b111	RGB display	No

Input Image Loading

- An 8×8×3 image is loaded for 64 cycles in **raster-scan** order.
- The pixel is in RGB type, and the size of each pixel is 24 bis.
- Raise o_out_valid to 1 after loading all image pixels.
- If o_in_ready is 0, stop input data until o_in_ready is 1.

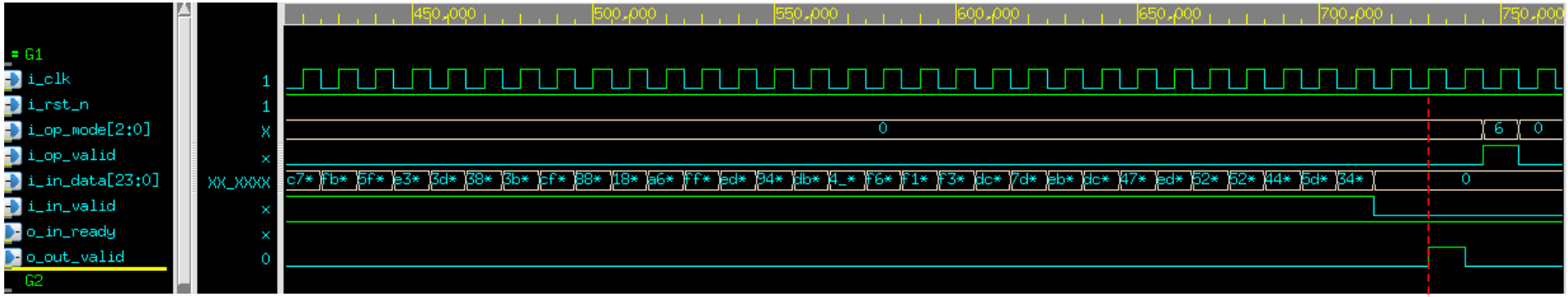
Sample Waveform

Testbench detects o_in_ready = 1 at negative edge, resume streaming input



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

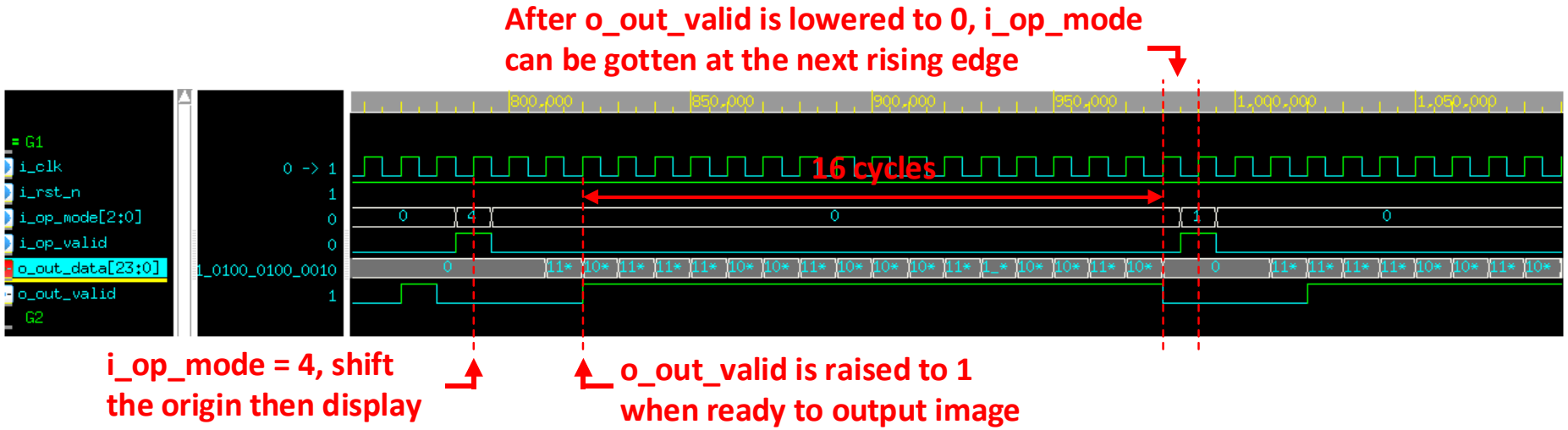
Testbench detects o_in_ready = 0 at negative edge, stop streaming input



o_out_valid is raised to 1 after loading whole image

Display Mode

- Default output is in RGB type
- If output of display exceeds the image boundary, retain the same origin point
  - i_op_mode: 1~4



Origin Right Shift

- Right shift the origin's coordinate with one pixel
  - i_op_mode = 1

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

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

Origin Down Shift

- Down shift the origin's coordinate with one pixel
  - i_op_mode = 2

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

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

Default origin

- Shift the origin's coordinate to 0
  - i_op_mode = 3

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

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

Zoom-In

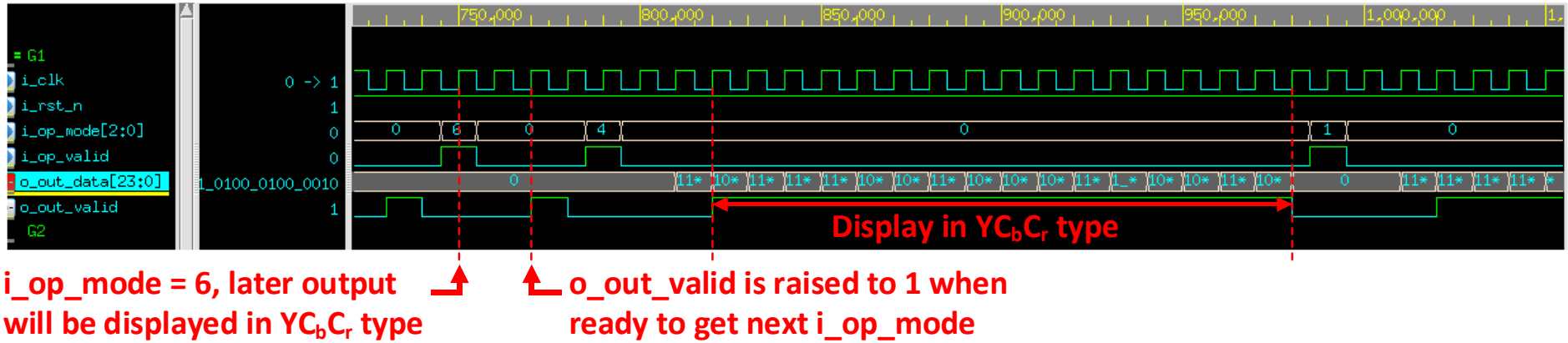
- Shift the origin's coordinate to 18
  - i_op_mode = 4

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

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

Change Display Type

- Two operation mode to change display type
  - i_op_mode = 6: change to YCbCr
  - i_op_mode = 7: change back to RGB
- The later output will be displayed in corresponding type



YCbCr Display

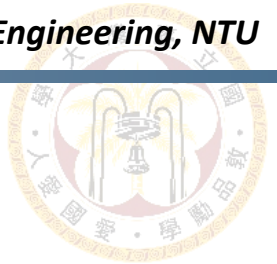
- Estimated YCbCr calculation

$$\mathbf{Y} = 0.25\mathbf{R} + 0.625\mathbf{G}$$

$$\mathbf{Cb} = -0.125\mathbf{R} - 0.25\mathbf{G} + 0.5\mathbf{B} + 128$$

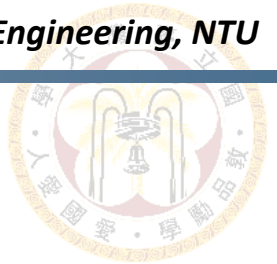
$$\mathbf{Cr} = 0.5\mathbf{R} - 0.375\mathbf{G} - 0.125\mathbf{B} + 128$$

- For YCbCr, only rounding the result after accumulation, i.e. do not truncate temporal result during shifting



RGB Display

- At RGB mode, the output should be the same as the input RGB data, or RGB data after median filtering



Median Filter Operation

- Median filtering is a nonlinear operation often used in image processing to reduce "salt and pepper" noise
- Do median filter processing on **RGB image**
- 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 to 10×10×3 first

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	208	174	71	71	71	98	0
41	208	194	174	112	98	150	87
41	233	203	194	190	174	150	71
41	203	190	178	178	174	168	71
124	168	168	168	125	168	125	58
72	162	168	168	125	114	114	42
37	124	124	81	114	114	125	42
0	72	37	9	9	31	114	0

Testbench

```
`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

golden*.dat

```
1   101000100011000011001101
2   011001011000101010100001
3   011001110110100101100001
4   110000110101100110001001
5   100011011101000000011001
6   100101110110110011001010
7   110001001001010101000001
8   101101000111011010100100
9   101100001011001101101100
10  010100001011000110110110
11  011010011001010001000010
12  000111011011110010000110
13  010011010111101110110001
14  100111010100000010110000
15  011100001011100100111100
16  100111111100000010010001
17  011001011000101010100001
18  011001110110100101100001
```

First display output

01_RTL

ipdc.v

```
`include "../sram_256x8/sram_256x8.v"

module ipdc (                       //Don't modify interface
    input         i_clk,
    input         i_rst_n,
    input         i_op_valid,
    input  [ 2:0] i_op_mode,
    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 testbed.v ipdc.v +notimingchecks +access+r +define+tb0
```

Modify by yourself

tb0, tb1, tb2

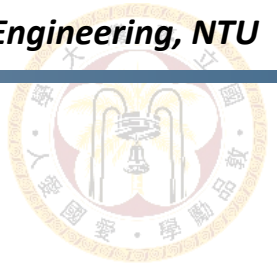
02_SYN

lpdc_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 testbed.v ipdc_syn.v tsmc13_neg.v \
          +ncmaxdelays +define+SDF+tb1 +access+r
```

sram_256x8

sram_256x8.pdf

Process Technology:
TSMC CL013G

High-Speed Single-Port Synchronous Flex-Repair™ SRAM with Redundancy

sram_256x8
256X8, Mux 8, Drive 6

Features

- Precise Optimization for TSMC's Eight-Layer Metal 0.13µm CL013G CMOS Process
- High Density (area is 0.018mm^2)
- Fast Access Time (1.20ns at fast@0C process, 1.32V, 0˚C)
- Fast Cycle Time (1.31ns at fast@0C process, 1.32V, 0˚C)
- One Read/Write Port
- Completely Static Operation
- Near-Zero Hold Time (Data, Address, and Control Inputs)

Memory Description

The 256X8 SRAM is a high-performance, synchronous single-port, 256-word by 8-bit memory designed to take full advantage of TSMC's eight-layer metal, 0.13µm CL013G CMOS process.

The SRAM's storage array is composed of six-transistor cells with fully static memory circuitry. The SRAM operates at a voltage of 1.2V ± 10% and a junction temperature range of -40˚C to +125˚C.

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 design (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: **60%**
  - **Level A**: Pass gate-level simulation (**60%**)
  - **Level B**: Pass RTL simulation, but gate-level simulation failed (**30%**)
  - **Level C**: RTL simulation failed (**0%**)
- Performance: **40%**
  - Performace = **Area (μm^2) × Area (μm^2) × Timing (μs)**
    (Lower number is better performance)
- Delay submission
  - In one week: (original score)*__0.7__
  - More than one week: **0** point for this homework
- Lose **5 point** for any wrong naming rule or format for submission

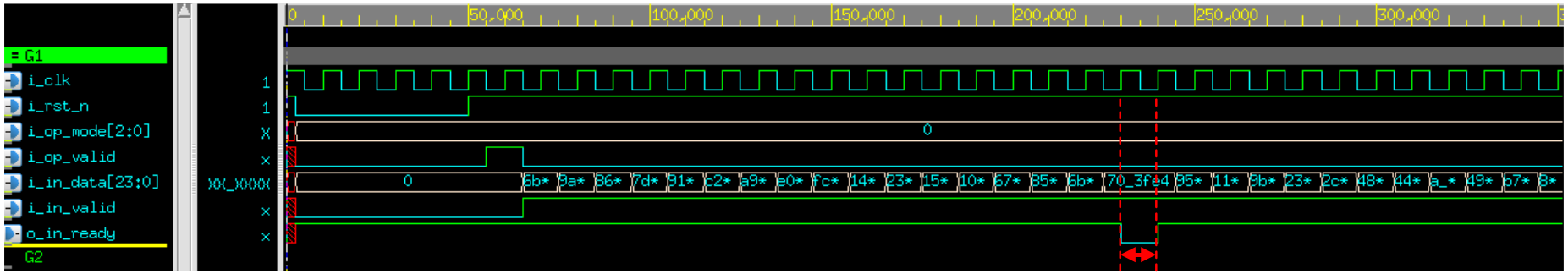
Area

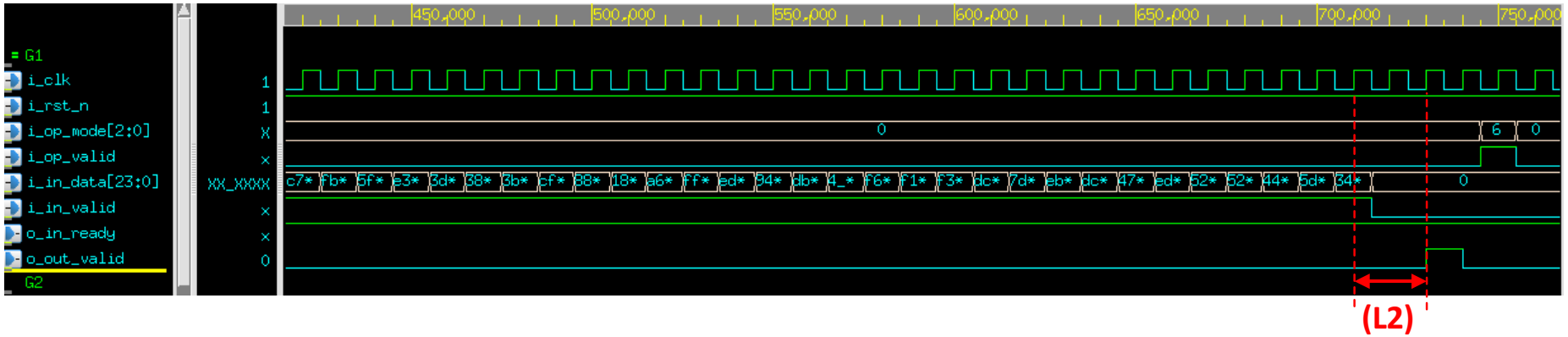
ipdc_syn.area

```
****************************************
Report : area
Design : ipdc
Version: N-2017.09-SP2
Date   : Mon Oct 26 00:25:49 2020
****************************************

Library(s) Used:

    slow (File: /home/raid7_2/course/cvsd/CBDK_IC_Contest/CIC/SynopsysDC/db/slow.db)
    sram_256x8 (File: /home/raid7_1/userd/d06027/CVSD/109/hw3/TA/HW3_v1/sram_256x8/sram_256x8_slow_syn.db)

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
```

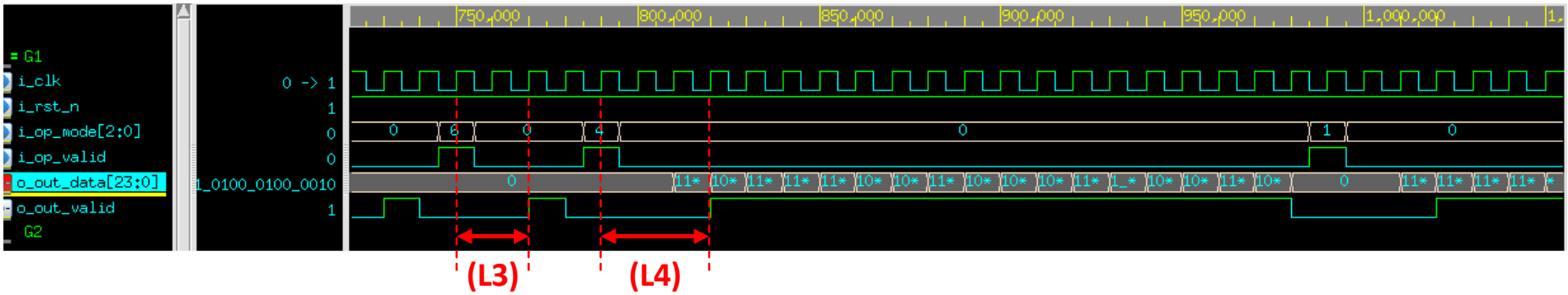
93408.960198 μm²

Timing


G1
i_clk
i_rst_n
i_op_mode[2:0]
i_op_valid
i_in_data[23:0]
i_in_valid
o_in_ready
G2
(L1)


G1
i_clk
i_rst_n
i_op_mode[2:0]
i_op_valid
i_in_data[23:0]
i_in_valid
o_in_ready
o_out_valid
G2
(L2)

Timing



- One tb timing = (L1)*n1 + (L2) + (L3)*n3 + (L4)*n4
  - Total: (t0 timing) + (t1 timing) + (t2 timing)

Performance

- Rank with your performance

Rank range	Score
1	40 %
2 – 5	38 %
6 – 15	30 %
16 – 30	20 %
31 – 50	10 %
51 – 100	5 %
101 ~	0 %

Report

- TAs will run your design with your clock period

report.txt

```
StudentID:

Clock period: (ns)

Area :  (um2)

Latency:
    tb0:  (cycle)
    tb1:  (cycle)
    tb2:  (cycle)
```