Computer-Aided VLSI System Design
Homework 3: Simple Image Processing and Display Controller

TA: 李諭奇 d06943027@ntu.edu.tw **Due Tuesday, Dec. 07, 14:00**

TA: 羅宇呈 f08943129@ntu.edu.tw

Data Preparation

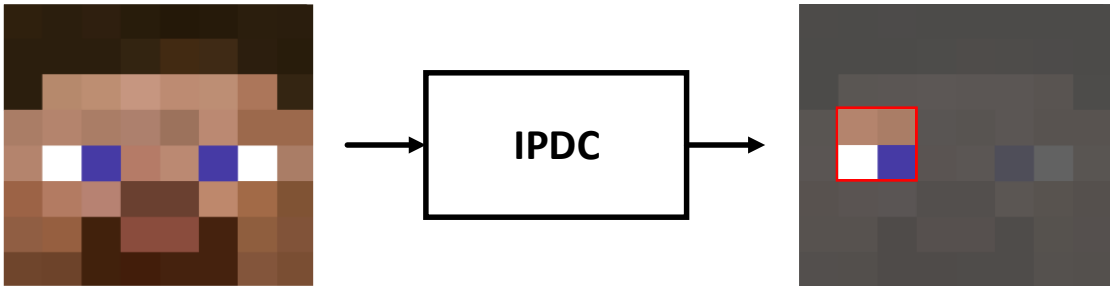
1. Decompress 1101_hw3.tar with following command

```
tar -xvf 1101_hw3.tar
```

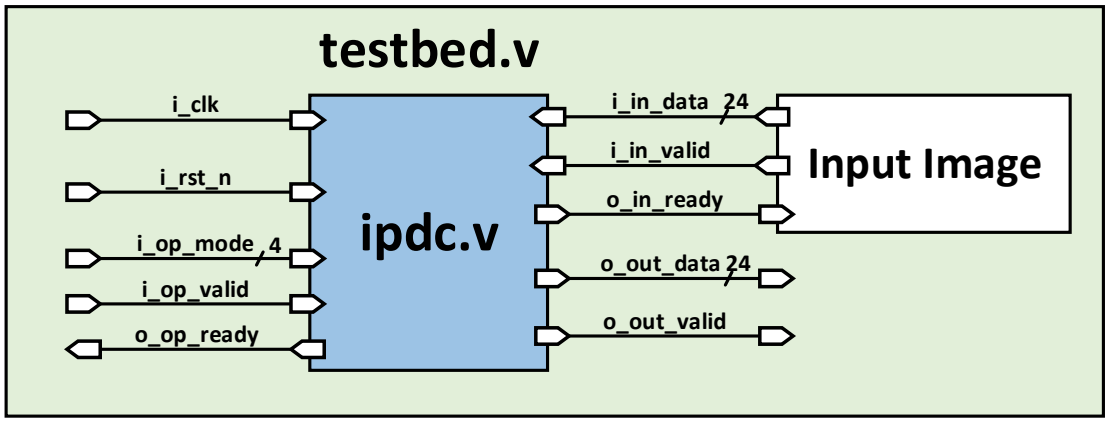
Folder	File	Description
00_TESTBED	testbed_temp.v	Testbench template
00_TESTBED/ PATTERN/	Indata*.dat	Input image data
	opmode*.dat	Pattern of operation mode
	golden*.dat	Golden data of display image
01_RTL	ipdc.v	Your design
	rtl_01.f	File list for rtl simulation
	01_run	NCVerilog command
	99_cleaan_up	Command to clean temporary data
02_SYN	syn.tcl	Script for synthesis
	ipdc_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_cleaan	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

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. A 16×16 image will be loaded first, and it will be processed with several functions.



Block Diagram



Specifications

1. Top module name: **ipdc**
2. Input/output description:

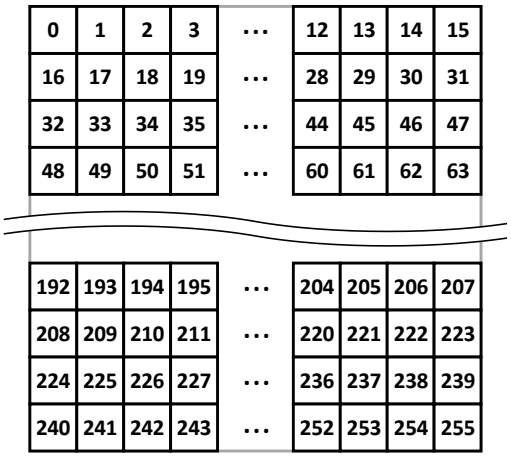
Signal Name	I/O	Width	Simple Description
i_clk	I	1	Clock signal in the system.
i_rst_n	I	1	Active **low** asynchronous reset.
i_op_valid	I	1	This signal is **high** if operation mode is valid
i_op_mode	I	4	Operation mode for processing
o_op_ready	O	1	Set **high** if ready to get next operation
i_in_valid	I	1	This signal is **high** if input pixel data is valid
i_in_data	I	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 = 4'b0000)
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

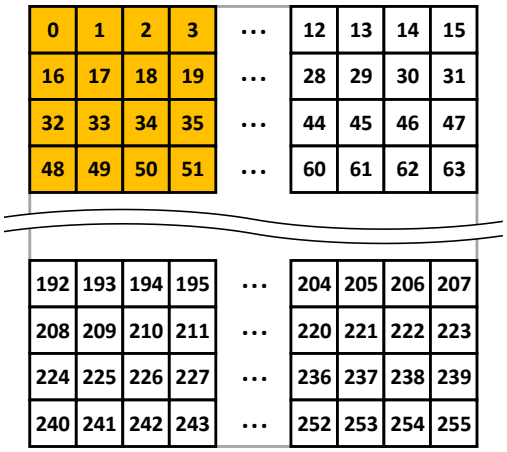
3. All inputs are synchronized with the **negative** edge clock.
4. All outputs should be synchronized at clock **rising** edge.
5. You should reset all your outputs when i_rst_n is **low**. Active low asynchronous reset is used and only once.
6. Operations are given by i_op_mode [3:0] when i_op_valid 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. Latency between i_op_valid and o_op_ready should be less than 1000ns (except i_op_mode = 4'b0000)
17. Only worst-case library is used for synthesis.
18. The synthesis result of data type should **NOT** include any **Latch**.
19. The slack for setup-time should be **non-negative**.
20. **No any timing violation and glitches** for the gate level simulation.

Design Description

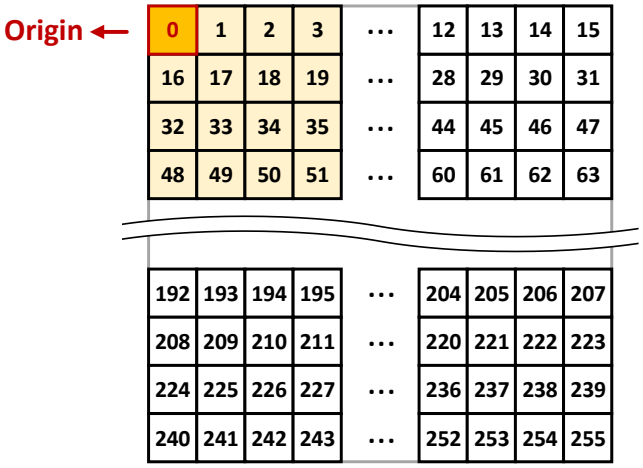
1. The input image is given in **raster-scan** order.



2. 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)



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

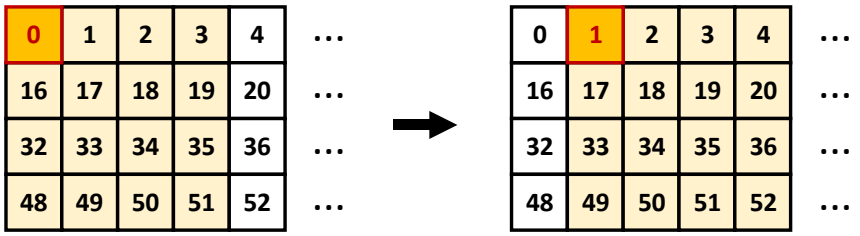


4. The followings are the operation modes you need to design for this homework:

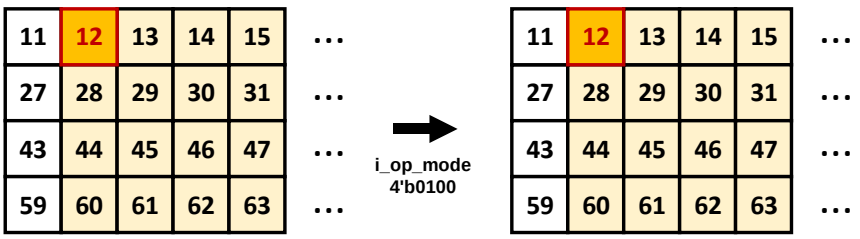
Operation Mode i_op_mode	Meaning	Need to display?
4'b0000	Input image loading	No
4'b0100	Origin right shift	Yes
4'b0101	Origin left shift	Yes
4'b0110	Origin up shift	Yes
4'b0111	Origin down shift	Yes
4'b1000	Scale-down	Yes
4'b1001	Scale-up	Yes
4'b1100	Median filter operation	Yes
4'b1101	YCbCr display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

5. 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 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.

6. Origin shifting:
   - EX. Origin right shift (i_op_mode = 4'b0100).



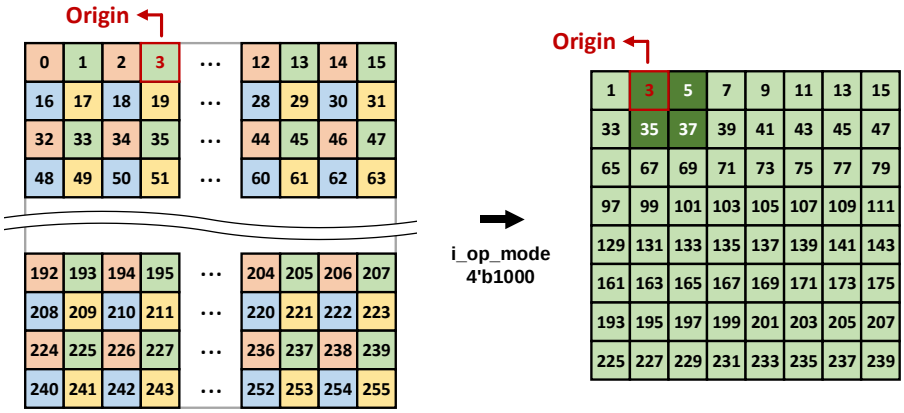
   - If output of display exceeds the image boundary, retain the same origin point.



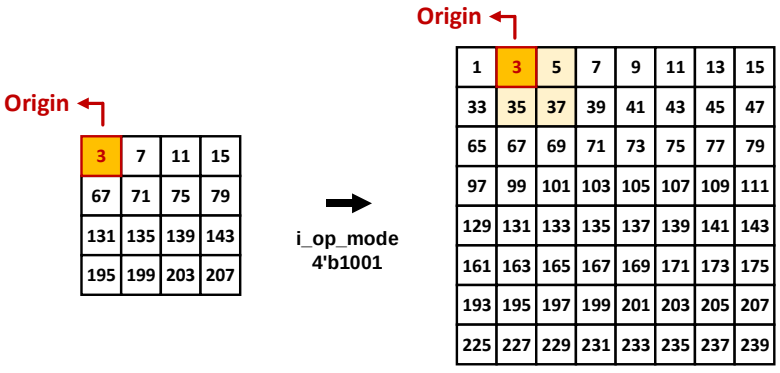
7. Image size:
   - 3 image sizes are considered in this design: 16x16, 8x8, 4x4
   - For output display, the display size will change with different image size

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

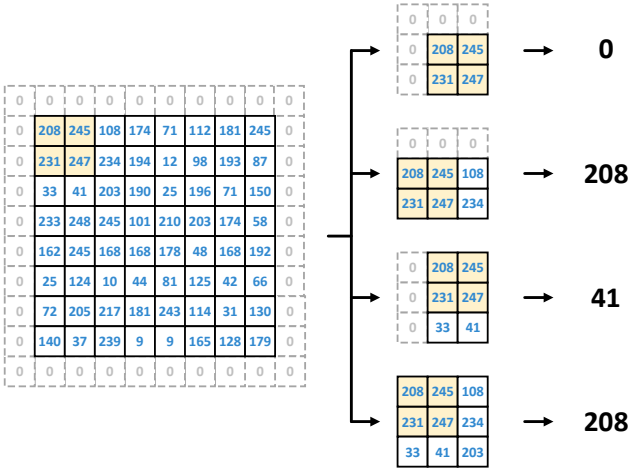
8. Scale-down:
   - Scale down both image size and display size to next level
     - Ex. For image size, 16x16 -> 8x8, 8x8 -> 4x4
   - If the image size is 4x4, retain the same image size and display size
   - Scale down the size with the location of the origin



9. 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
   - If the display region is located furthest to the right, retain the same image size and display size

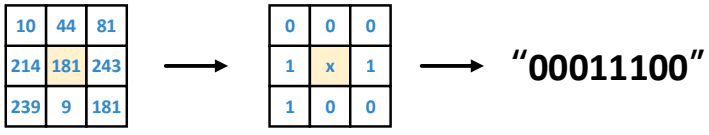


10. Median filter operation:
    - For this operation, you have to perform median filtering on the display region.
    - The filter is a 3x3 kernel. It results in a median of the set of pixel value.
    - Operate median filtering to R-channel, G-channel, B-channel, separately.
    - The image needs to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



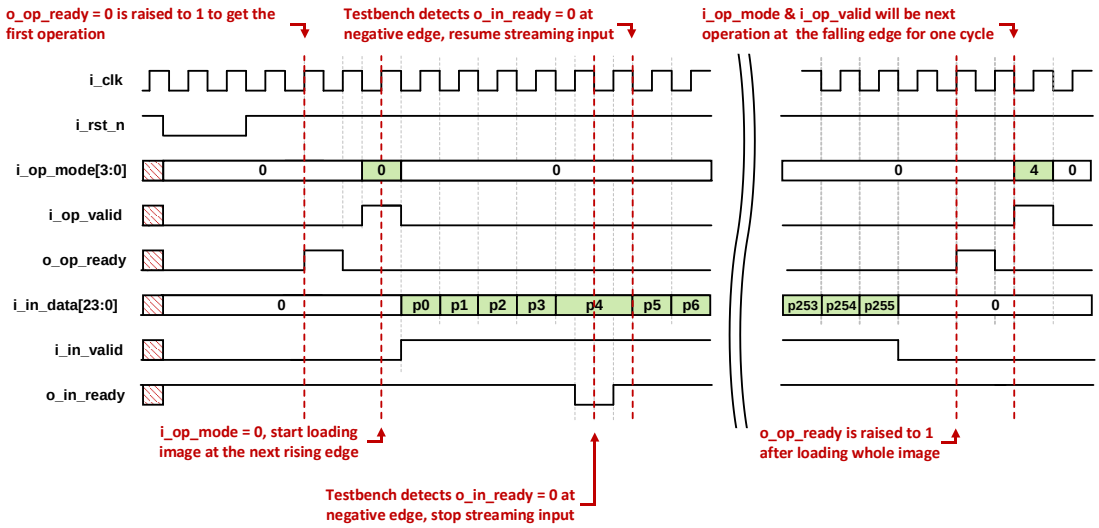
11. YCbCr display:
    - For this operation, you have to perform YCbCr transform on the display region.
    - Estimated YCbCr calculation
        - $Y = 0.25R + 0.625G$
        - $C_b = -0.125R - 0.25G + 0.5B + 128$
        - $C_r = 0.5R - 0.375G - 0.125B + 128$
    - For YCbCr, only **rounding** the result after accumulation, i.e. do not truncate temporal result during shifting
    - The values of original pixels will not be changed

12. 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.

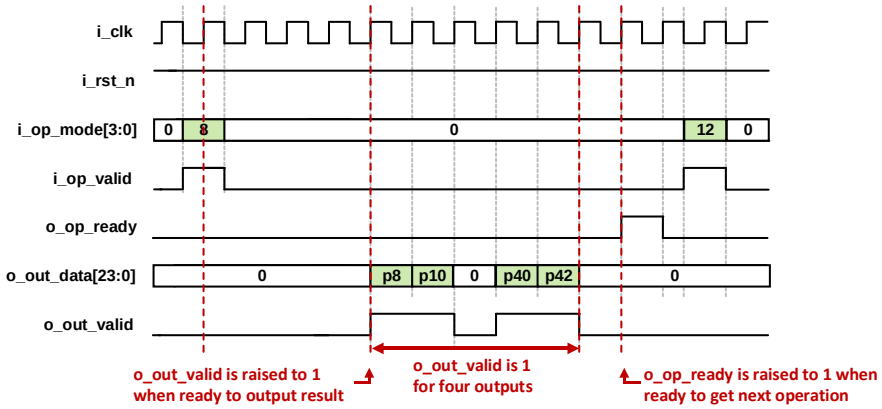


Sample Waveform

1. Load Image Data (i_op_mode = 0)



2. Other operations



Submission

1. 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**
   - **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 folder **HW3** on FTP server
   - IP: 140.112.175.68
   - Port: 21
   - Account: 1101cvsd_student
   - Password: ilovecvsd

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: **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%

3. Performance: **20%** (correct for all patterns)
   - **Area (μm^2)**
   - (Lower number is better performance)
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 FTP server.

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
```