Computer-Aided VLSI System Design
Final Project: Component Labeling Engine

TA: 李諭奇 d06943027@ntu.edu.tw
TA: 傅子興 r08943012@ntu.edu.tw

Due Friday, Jan. 15, 23:59

Data Preparation

1. Decompress 1091_final.tar with following command

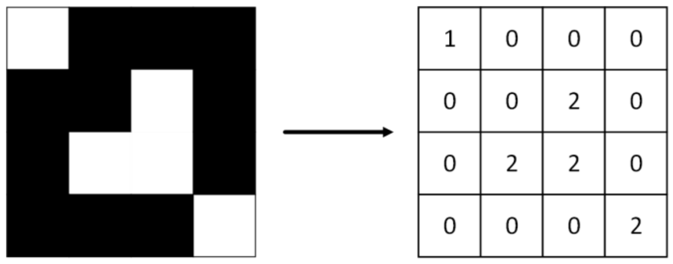
```
tar -xvf 1091_final.tar
```

File/Folder	Description
CLE.v	Your design.
testbed.v	Testbench for CLE.
rom_128x8/	ROM related files for different binary image patterns.
sram_1024x8/	SRAM related files for different golden label data. **Do not use sram_1024x8.v in your design.**
.synopsys_dc.setup	Configuration file for DC.
CLE_DC.sdc	Constraint file for synthesis.
CLE_APR.sdc	Constraint file for APR.
sram_for_design/	SRAM related files. You can use these memories in your design. Contain 3 types of SRAM: 1. sram_256x8 2. sram_512x8 3. sram_4096x8

All libraries needed for synthesis, simulation and APR can be found in previous homework. **Only worst-case library is used in this final project.**

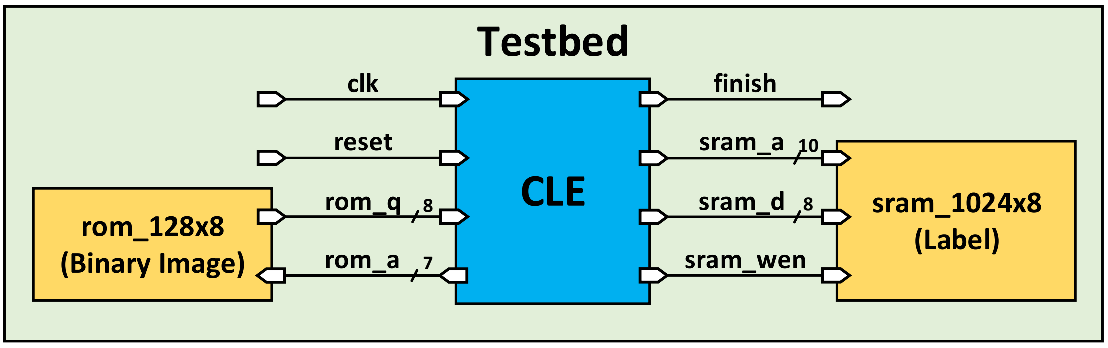
Introduction

In this final project, you are asked to design a **Component Labeling Engine (CLE)**, which can detect object segmentation from the binary image, and give the same ID number to the same object.



Binary image Label Data

Block Diagram



Specifications

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

Signal Name	I/O	Width	Simple Description
clk	I	1	Clock signal in the system. All inputs are synchronized with the **positive** edge clock. All outputs should be synchronized at clock **rising** edge
reset	I	1	Active **high** asynchronous reset.
rom_a	O	7	Address for binary image memory.
rom_q	I	8	Binary data from binary image memory.
sram_a	O	10	Address for storing to label memory.
sram_d	O	8	Label data for storing to label memory.
sram_wen	O	1	Set **low** if CLE needs to write data to label memory. (reading mode is not supported in this design)
finish	O	1	Set **high** if finish Labeling, and testbed will check the correctness of label data in sram_1024x8.

Design Description

1. The input image is a 32x32 binary image as shown in **Fig.1**. For the binary signal, 0 represents the background, and 1 represents the object for each pixel. You have to check if those pixels with value 1 are connected or not. The connected pixels represent to the same object. Those pixels are given with the same label ID from the same object. The number of label ID can be created by yourself.

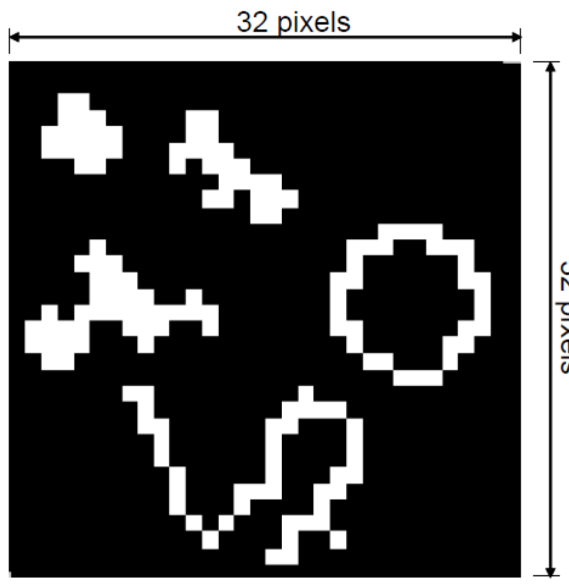
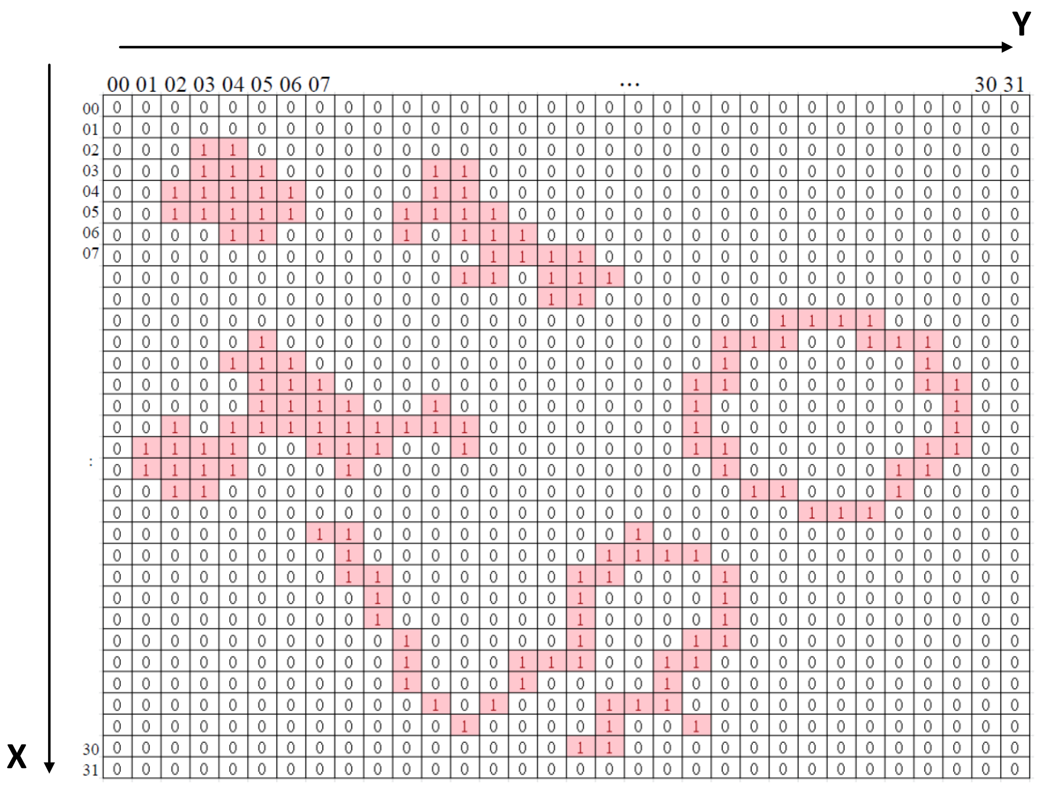


Fig.1. Binary image.



	00	01	02	03	04	05	06	07											…												30	31
00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	0	0	1	1	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	0	0	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0
	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	0	0	0
	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0
	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
	0	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
	0	0	1	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0
	0	1	1	1	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0
:	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	1	0	0	0
	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig.2. Actual value for each pixel in binary image.

2. The image is already stored in the 128x8 ROM. The storing order is shown in **Fig.3**. For example, if the address value is “0”, the corresponding 8-bit binary data represents the pixels [X=00, Y=00-07] in **Fig.2**. The MSB is related to [X=00, Y=00], and the LSB is related to [X=00, Y=07]. The number of times to read data from ROM is not constrained, and the signal CEN from ROM is always set to 0.

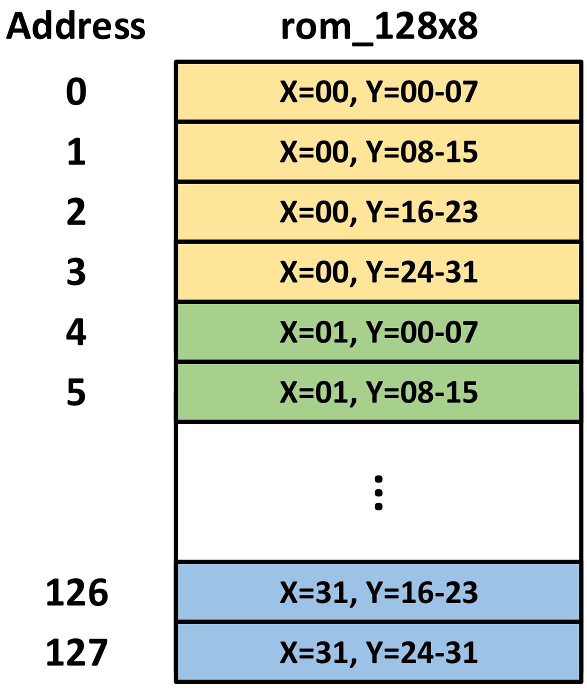


Fig.3. The storing order for the binary image in ROM.

3. The operations in CLE are shown below:
   A. First, CLE has to find the pixel equal to 1 in **Fig.2**, which means the object position. Then, it has to identify whether the pixel is connected to other pixels in a 3x3 block. **Fig.4**. shows the 8 connecting situations in a 3x3 block. If the pixels are connected, they are seen as the same object. Otherwise, the pixels which are not connected are seen as different object.

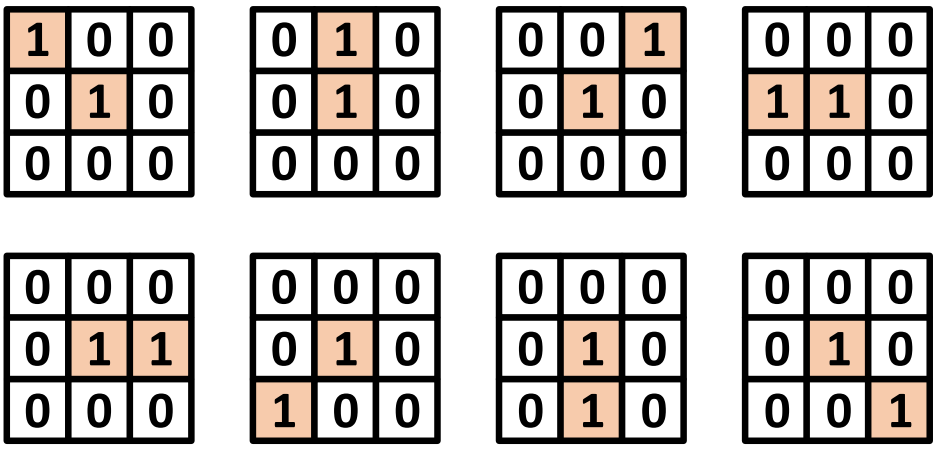
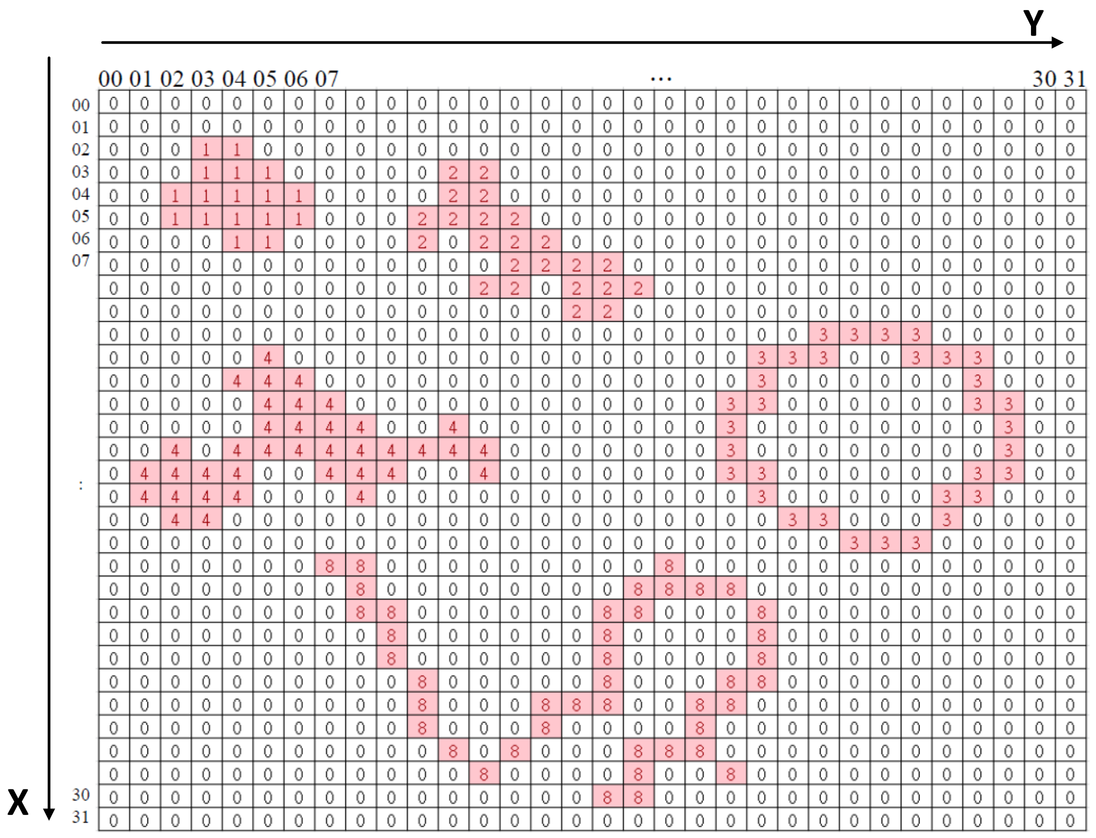


Fig.4. The 8 connecting situations in a 3x3 block.

B. Every pixel is required to be set with the same label ID for the same object. You can decide the number of label ID by yourself with the following naming rules:
   a. The number of label ID range you can use: **8’h01~8’hFB**
   b. The number of label ID range you cannot use: **8’hFC~8’hFF** (These numbers have other specific meanings)
   c. The number **8’h00** is for background. You can’t give this number to object.
   d. Label ID cannot be reused for different object.

4. **Fig.5** shows the example result after processing on **Fig.2**. Five label IDs are used in this example. All labels for the 32x32 image are required to be stored in the outside SRAM (the sram_1024x8 is not included in CLE). The storing method is shown in **Fig.6**. For example, the address value “0” stores the label for pixel [X=00, Y=00], and so on. If all labels are stored in sram_1024x8, CLE sets output “finish” to high, and the testbench will check the answer.



	00	01	02	03	04	05	06	07											…												30	31
00	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
01	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
02	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
03	0	0	0	1	1	1	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
04	0	0	1	1	1	1	1	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
05	0	0	1	1	1	1	1	0	0	0	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
06	0	0	0	0	1	1	0	0	0	0	2	0	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
07	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3	3	0	0	0	0	0
	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3	0	0	3	3	3	0	0	0
	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	3	0	0	0
	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0	0	0	0	3	3	0	0
	0	0	0	0	0	4	4	4	4	0	0	4	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	3	0	0
	0	0	4	0	4	4	4	4	4	4	4	4	4	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	3	0	0
:	0	4	4	4	4	0	0	4	4	4	0	0	4	0	0	0	0	0	0	0	3	3	0	0	0	0	0	0	3	3	0	0
	0	4	4	4	4	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	3	0	0	0
	0	0	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	0	0	0	3	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	3	3	0	0	0	0	0
	0	0	0	0	0	0	0	8	8	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	8	8	8	8	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	8	8	0	0	0	0	0	0	8	8	0	0	0	8	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	8	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	8	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	8	0	0	0	8	8	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	8	0	0	0	8	8	8	0	0	8	8	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	8	0	0	0	8	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	8	0	8	0	0	0	8	8	8	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	8	0	0	0	0	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig.5. Labels for each pixel to store in SRAM.

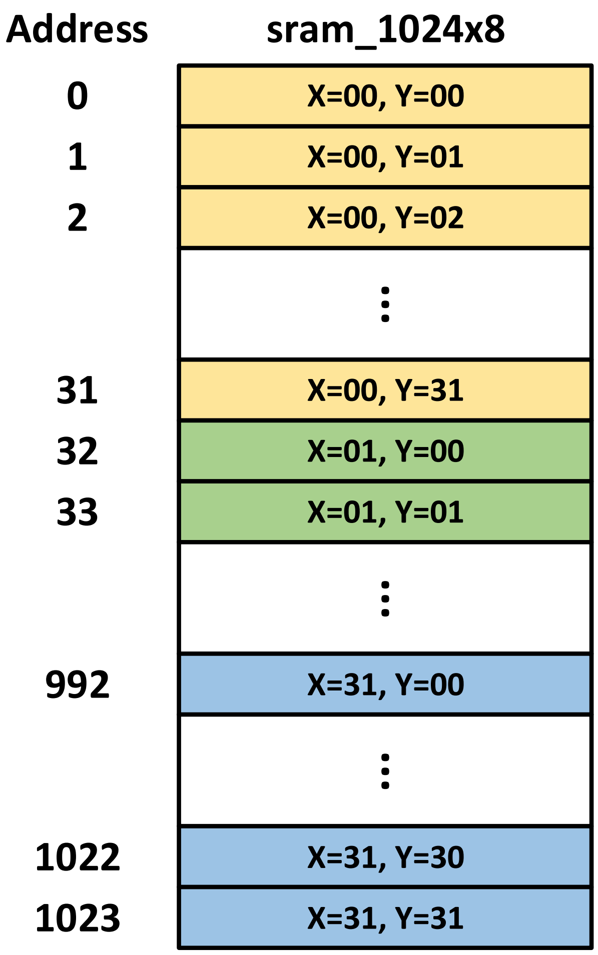
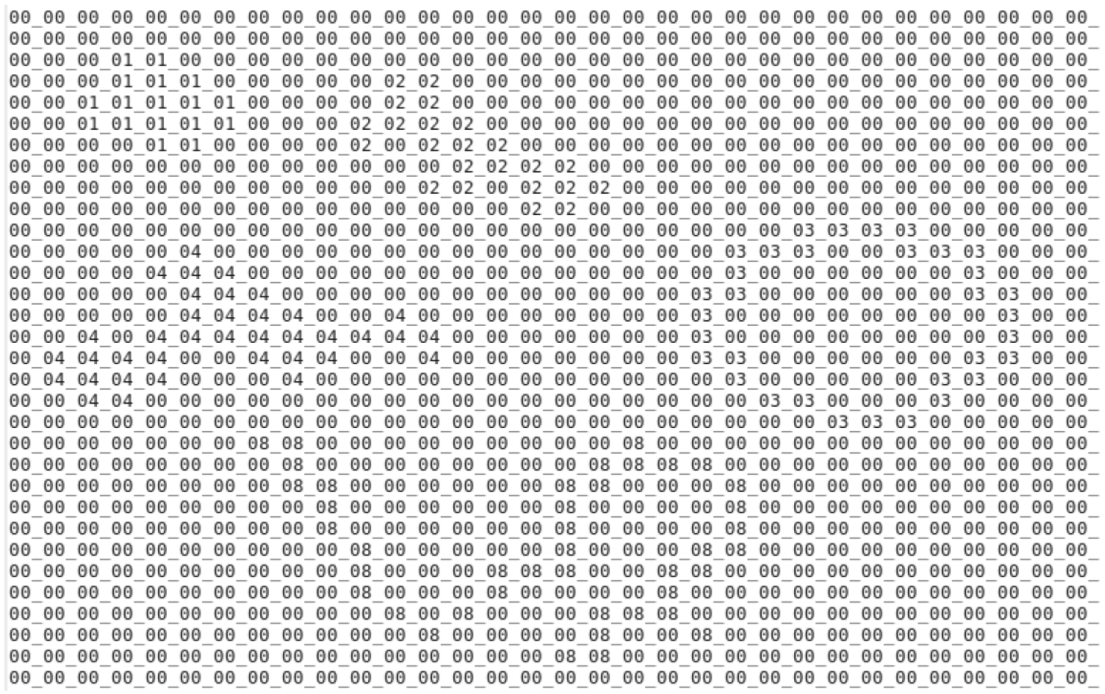


Fig.6. The storing order for the 8-bit label in SRAM.

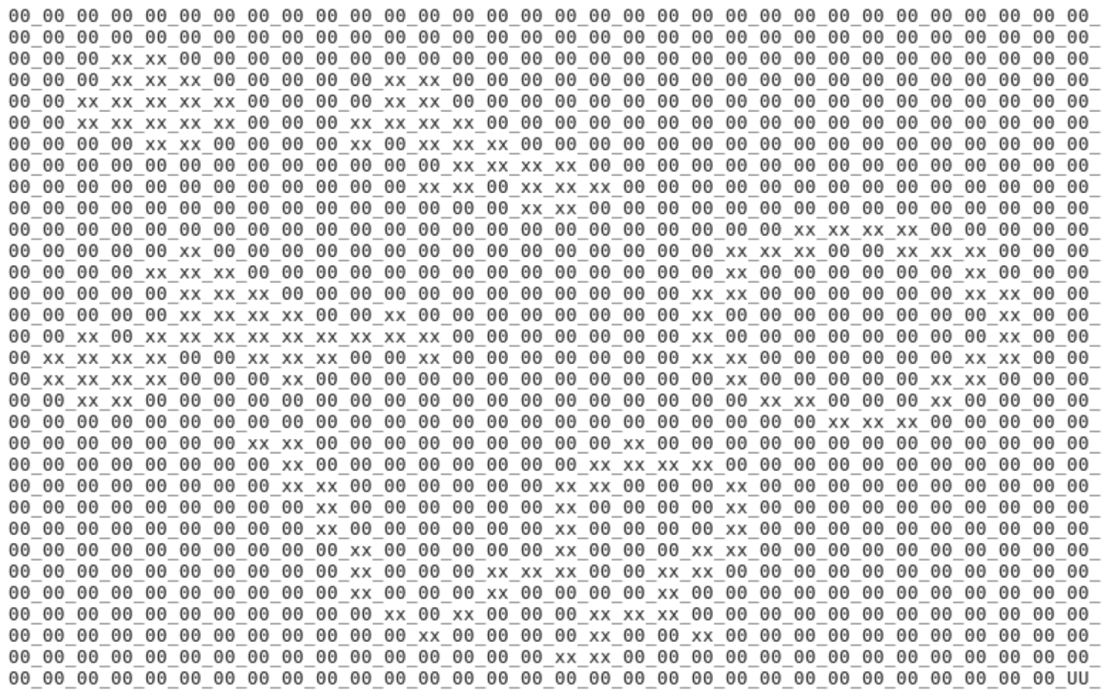
5. The testbench will show your result as **Fig.7**, and it will also compare with the golden result. If the result of CLE is different from golden result, it will show the specific label for pixels with wrong labels as **Fig.8**. The notations for specific label are shown below:

Specific label	**Description**
XX	The Pixel belongs to background. The CLE identifies it to wrong ID number.
UU	The Pixel belongs to background. The CLE identifies it to Unknown.
xx	The Pixel belongs to object. The CLE identifies it to wrong ID number.
uu	The Pixel belongs to object. The CLE identifies it to Unknown.



```
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_01_01_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_01_01_01_00_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_01_01_01_01_01_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_01_01_01_01_01_00_00_00_02_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_01_01_00_00_00_00_02_00_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_02_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_02_02_00_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_03_00_00_00_00_00_
00_00_00_00_00_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_00_00_03_03_03_00_00_00_
00_00_00_00_04_04_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_00_00_00_00_00_00_03_00_00_00_
00_00_00_00_00_04_04_04_00_00_00_00_00_00_00_00_00_00_00_00_03_03_00_00_00_00_00_00_03_03_00_00_
00_00_00_00_00_04_04_04_04_00_00_04_00_00_00_00_00_00_00_00_03_00_00_00_00_00_00_00_00_03_00_00_
00_00_04_00_04_04_04_04_04_04_04_04_04_00_00_00_00_00_00_00_03_00_00_00_00_00_00_00_00_03_00_00_
00_04_04_04_04_00_00_04_04_04_00_00_04_00_00_00_00_00_00_00_03_03_00_00_00_00_00_00_03_03_00_00_
00_04_04_04_04_00_00_00_04_00_00_00_00_00_00_00_00_00_00_00_00_03_00_00_00_00_00_03_03_00_00_00_
00_00_04_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_00_00_00_03_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_00_00_00_00_00_
00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_00_00_08_08_08_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_08_08_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_08_00_00_00_08_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_08_08_08_00_00_08_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_08_00_08_00_00_00_08_08_08_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_08_00_00_00_00_08_00_00_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
```

Fig.7. Log message of SRAM from function simulation.



```
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_xx_xx_xx_00_00_00_00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_xx_xx_xx_xx_xx_00_00_00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_xx_xx_xx_xx_xx_00_00_00_xx_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_xx_xx_00_00_00_00_xx_00_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_00_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_xx_xx_00_00_00_00_00_
00_00_00_00_00_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_xx_00_00_xx_xx_xx_00_00_00_
00_00_00_00_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_xx_00_00_00_00_00_00_xx_00_00_00_
00_00_00_00_00_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_00_00_00_00_00_00_xx_xx_00_00_
00_00_00_00_00_xx_xx_xx_xx_00_00_xx_00_00_00_00_00_00_00_00_xx_00_00_00_00_00_00_00_00_xx_00_00_
00_00_xx_00_xx_xx_xx_xx_xx_xx_xx_xx_xx_00_00_00_00_00_00_00_xx_00_00_00_00_00_00_00_00_xx_00_00_
00_xx_xx_xx_xx_00_00_xx_xx_xx_00_00_xx_00_00_00_00_00_00_00_xx_xx_00_00_00_00_00_00_xx_xx_00_00_
00_xx_xx_xx_xx_00_00_00_xx_00_00_00_00_00_00_00_00_00_00_00_00_xx_00_00_00_00_00_xx_xx_00_00_00_
00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_00_00_00_xx_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_xx_00_00_00_00_00_
00_00_00_00_00_00_00_xx_xx_00_00_00_00_00_00_00_00_00_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_xx_00_00_00_00_00_00_00_00_xx_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_xx_xx_00_00_00_00_00_00_xx_xx_00_00_00_xx_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_xx_00_00_00_00_00_00_xx_00_00_00_00_xx_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_xx_00_00_00_00_00_00_xx_00_00_00_00_xx_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_xx_00_00_00_00_00_xx_00_00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_xx_00_00_00_xx_xx_xx_00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_xx_00_00_00_xx_00_00_00_00_xx_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_xx_00_xx_00_00_00_xx_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_xx_00_00_00_00_xx_00_00_xx_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_xx_xx_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_UU_
```

Fig.8. Log message of SRAM with specific label result from function simulation.

Timing Diagram

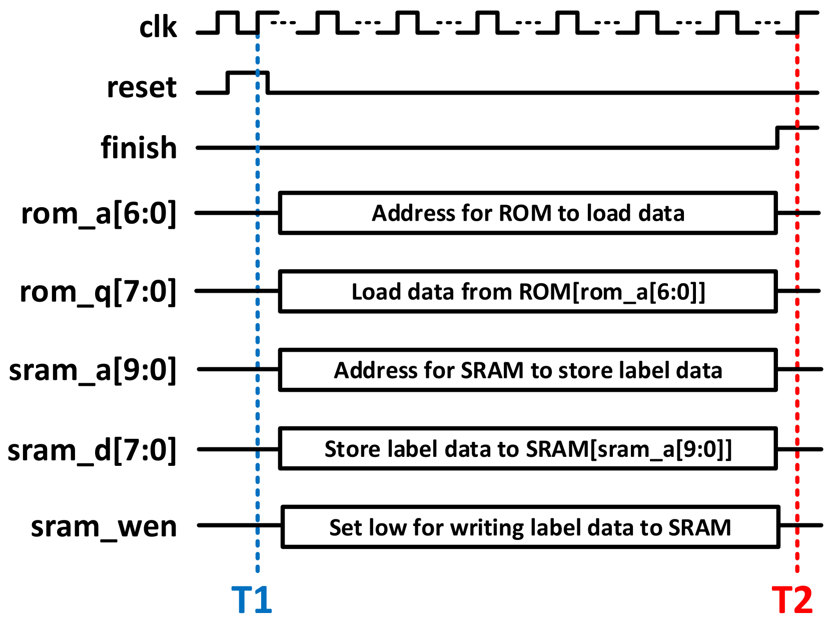
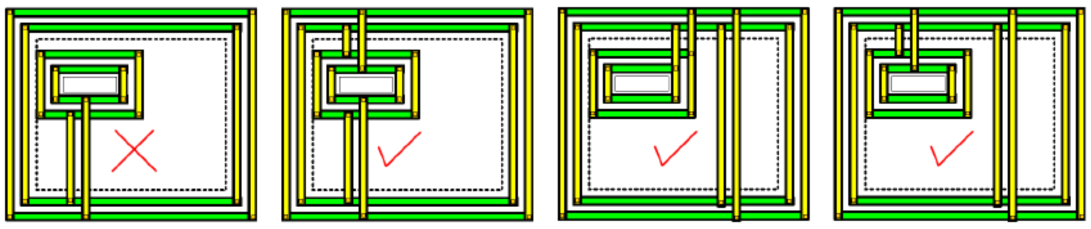

clk
reset
finish
rom_a[6:0]
Address for ROM to load data
rom_q[7:0]
Load data from ROM[rom_a[6:0]]
sram_a[9:0]
Address for SRAM to store label data
sram_d[7:0]
Store label data to SRAM[sram_a[9:0]]
sram_wen
Set low for writing label data to SRAM
T1
T2

Fig.9. Timing diagram of CLE

1. CLE is initialized with one cycle **reset**.
2. ROM is readable after **T1**.
3. Set finish to **high** at **T2** after store all labels for objects and background in sram_1024x8.

Specifications for APR

1. Macro layout only. (IO Pad and bonding pad are not required)
2. The width of power ring is set to **2um**, and **only one group** of VDD/VSS is required.
3. Do not add Dummy Metal in your design.
4. All VDD/VSS pins from SRAM are required to connect to Core Power Ring, and the width is set to **2um**.
5. At least one power stripe us added in your design, and the width of VDD/VSS is set to **2um**. (At least one stripes along vertical direction)
6. Power rail (follow pin) is required to add in your design.
7. Core Filler is required to add in your design.
8. Remember to generate **CLE.ioc**, and load it to allocate pin position.



Submission

1. Create a folder named **teamID_final**, and put all below files into the folder (If you have two designs for different conditions of performance, create folder named **A** and folder named **AT2** under folder **teamID_final**, and put following files in each folder)
   - **CLE.v**
   - **CLE_syn.v**
   - **CLE_syn.sdf**
   - **CLE_pr.v**
   - **CLE_pr.sdf**
   - **CLE.gds**
   - **CLE_final.tar (archive of the design database directory)**
   - **report.pdf**
   - **all other design files** included in your design for rtl simulation (optional)

   Note1: Use **lower case** for the letter in your TeamID. (Ex. team00_final)

   Note2: For **CLE_final.tar**, compress your final database from APR

```
tar -cvf CLE_final.tar CLE_final.dat
```

2. Compress the folder **teamID_final** in a **tar file** named **teamID_final_v*k*.tar (*k* is the number of version, *k* =1,2,…)**

```
tar -cvf teamID_final_vk.tar teamID_final
```

   TA will only check the last version of your homework.

   Note: Use **lower case** for the letter in your student ID. (Ex. team00_final_v1)

3. Submit to folder **final** on FTP server
   - IP: 140.112.175.68
   - Port: 21
   - Account: 1091cvsd_student
   - Password: ilovecvsd

Grading Policy

1. TA will run your code with following format of commands.
   a. RTL simulation (under **01_RTL**)

```
ncverilog testbed.v CLE.v \
          ../sram_for_design/sram_256x8/sram_256x8.v \
          ../sram_for_design/sram_512x8/sram_512x8.v \
          ../sram_for_design/sram_4096x8/sram_4096x8.v \
          +notimingchecks +define+tb0 +access+rw
```

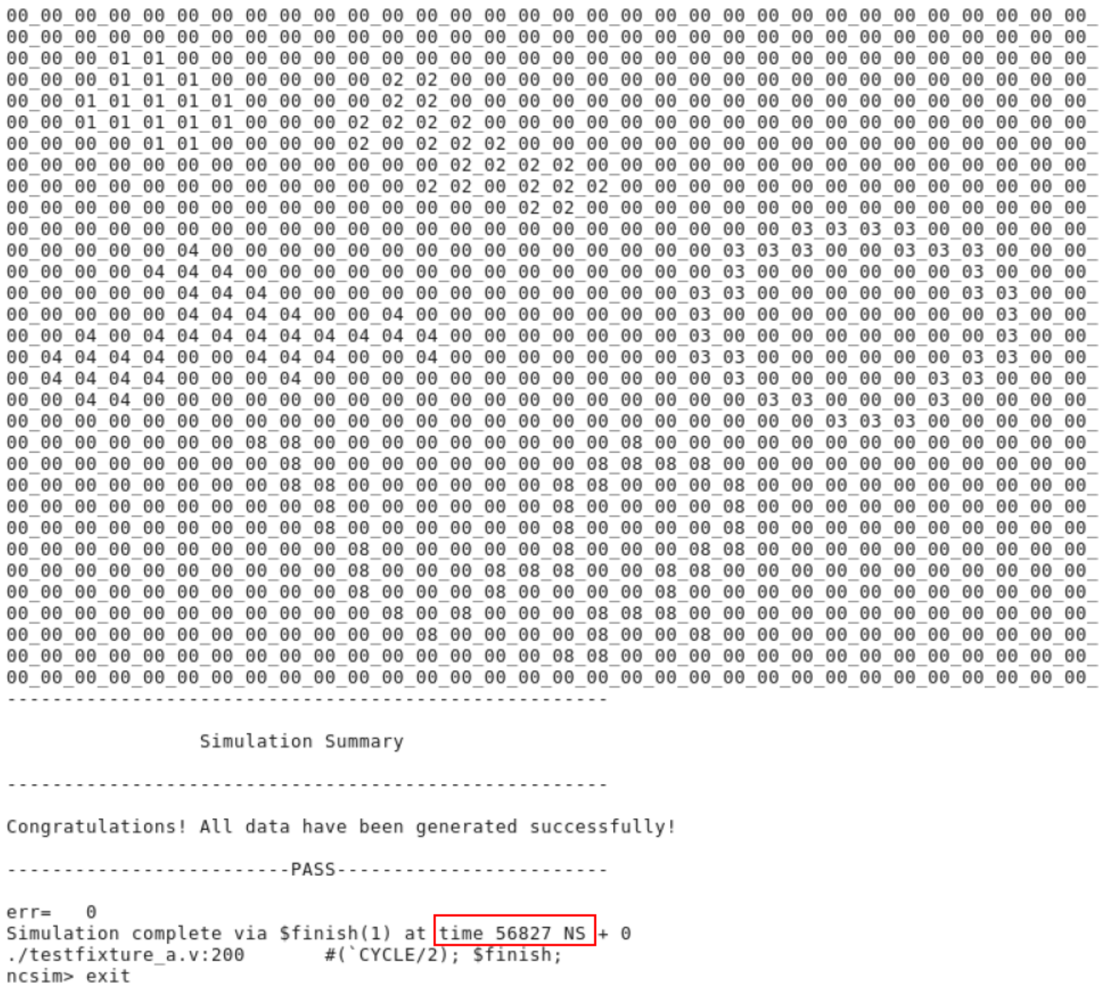
b. Gate-level simulation (under **03_GATE**)

```
ncverilog testbed.v CLE_syn.v tsmc13_neg.v \
          ../sram_for_design/sram_256x8/sram_256x8.v \
          ../sram_for_design/sram_512x8/sram_512x8.v \
          ../sram_for_design/sram_4096x8/sram_4096x8.v \
          +ncmaxdelays +define+SDF+tb0 +access+rw
```

c. Post-layout simulation (under **05_POST**)

```
ncverilog testbed.v CLE_pr.v tsmc13_neg.v \
          ../sram_for_design/sram_256x8/sram_256x8.v \
          ../sram_for_design/sram_512x8/sram_512x8.v \
          ../sram_for_design/sram_4096x8/sram_4096x8.v \
          +ncmaxdelays +define+SDF+tb0 +access+rw
```

2. TAs will rank you design with **Area** and **Time**
   - Time: processing time from simulation (ex. 56827ns below)



```
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_01_01_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_01_01_01_00_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_01_01_01_01_01_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_01_01_01_01_01_00_00_00_02_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_01_01_00_00_00_00_02_00_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_02_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_02_02_00_02_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_02_02_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_03_00_00_00_00_00_
00_00_00_00_00_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_00_00_03_03_03_00_00_00_
00_00_00_00_04_04_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_00_00_00_00_00_00_03_00_00_00_
00_00_00_00_00_04_04_04_00_00_00_00_00_00_00_00_00_00_00_00_03_03_00_00_00_00_00_00_03_03_00_00_
00_00_00_00_00_04_04_04_04_00_00_04_00_00_00_00_00_00_00_00_03_00_00_00_00_00_00_00_00_03_00_00_
00_00_04_00_04_04_04_04_04_04_04_04_04_00_00_00_00_00_00_00_03_00_00_00_00_00_00_00_00_03_00_00_
00_04_04_04_04_00_00_04_04_04_00_00_04_00_00_00_00_00_00_00_03_03_00_00_00_00_00_00_03_03_00_00_
00_04_04_04_04_00_00_00_04_00_00_00_00_00_00_00_00_00_00_00_00_03_00_00_00_00_00_03_03_00_00_00_
00_00_04_04_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_00_00_00_03_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_03_03_03_00_00_00_00_00_
00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_00_00_08_08_08_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_08_08_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_00_00_08_00_00_00_08_08_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_08_08_08_00_00_08_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_08_00_00_00_08_00_00_00_00_08_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_08_00_08_00_00_00_08_08_08_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_08_00_00_00_00_08_00_00_08_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_08_08_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_00_
-----------------------------------------------------

                 Simulation Summary

-----------------------------------------------------

Congratulations! All data have been generated successfully!

-------------------------PASS------------------------

err=    0
Simulation complete via $finish(1) at time 56827 NS + 0
./testfixture_a.v:200        #(`CYCLE/2); $finish;
ncsim> exit
```

   - Area: Use below command to analyze the area, **die area** is adopted when ranking (remember to save your design files first!!)

```
innovus #> analyzeFloorplan
```

3. Different score for different level design
   - **Level A**: achieve following request and get **50pts**
     a. Function work, and pass all patterns in RTL-simulation
     b. Finish synthesis, and pass all patterns in gate-level simulation
     c. Finish APR with given specifications, and pass all patterns in post-layout simulation
     d. Cycle time for gate-level simulation and post-layout simulation is less than **100ns**

     Two conditions of performance are considered: (You can hand in 2 designs)

     a. Score = Area × Time^2

     Since 7 patterns are provided, the **Total Score** needs to consider all simulation times of the patterns

     Total score = Score$_0$ + Score$_1$ + … + Score$_6$

     b. Score = Area

     20pts for each condition, and TAs will grade your design with your rank

Rank range	Points
1	20 %
2 – 5	18 %
6 – 10	16 %
11 – 15	14 %
16 – 20	12 %

Rank range	Points
21 – 25	10 %
26 – 30	8 %
31 – 35	6 %
36 – 40	4 %
41 – 45	2 %
45 –	0 %

   - **Level B**: Finish APR, but contain one of the following situations, get **40pts**
     a. Some specifications for APR are violated
     b. Contain less than 5 DRC/LVS errors

   - **Level C**: Contain one of the following situations, get **20pts**
     a. Finish synthesis and pass gate-level simulation, but post-layout simulation is failed
     b. Finish APR, but contain more than 5 DRC/LVS errors

4. Report: **10pts**

Final Projection Presentation

Date: **January 19, 2021**

Reference

[1] IC Design Contest, 2016.