

NYCU-EE IC LAB – Fall 2024

Lab03 Exercise

Design: Tetris

Data Preparation

1. Extract files from TA's directory:

```
% tar xvf ~iclabTA01/Lab03.tar
```

2. The extracted LAB directory contains:

- a. **00_TESTBED**
- b. **01_RTL**
- c. **02_SYN**
- d. **03_GATE**
- e. **09_SUBMIT**

Introduction

Tetris is a classic and influential video game. The game features a series of geometric shapes called "Tetriminoes," which are composed of four-square blocks each. These shapes fall from the top of the screen, and the player's task is to rotate and position them to create complete horizontal lines without any gaps. When a line is completed, it disappears, and the player earns points.

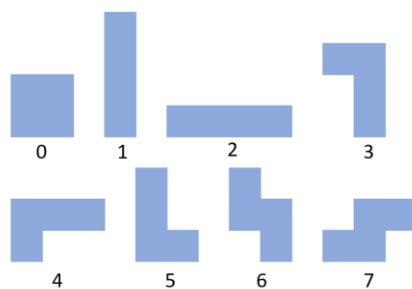
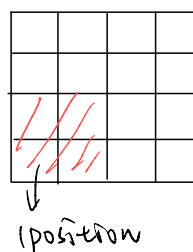
In today's lab, your task is to construct an easy version of Tetris and complete a pattern that validates your design. Your Tetris design has to correctly place the tetriminoes in a 6x12 map according to their position and their type which are given as the inputs of your design, and then eliminate the rows filled up with blocks as well as calculate the score. As for the pattern, it should integrate the functions of stimulus generation, basic environment check, and answer check altogether.

Tetris.v

There are 16 tetriminoes in each round of game. The steps demonstrated below would repeat for 16 or less than 16 times in certain situation successively.

- Description of the inputs

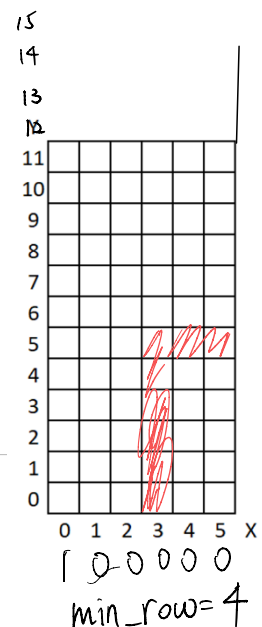
The pattern would deliver a tetrimino as well as its X position(column) to the design once at a time. There are 8 types of tetrimino, and the variable position is used to denote the X coordinate of the leftmost block of the tetrimino.

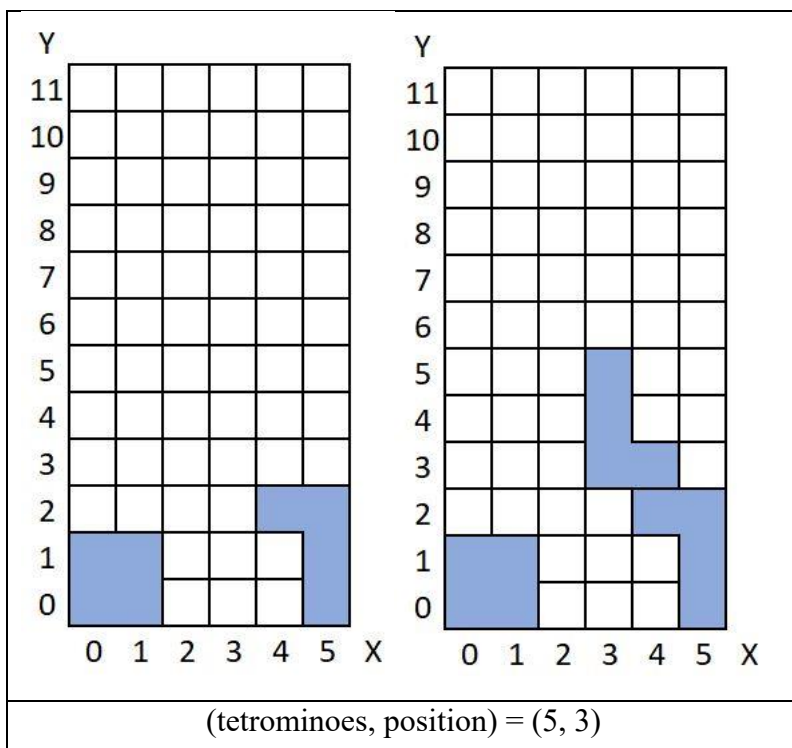
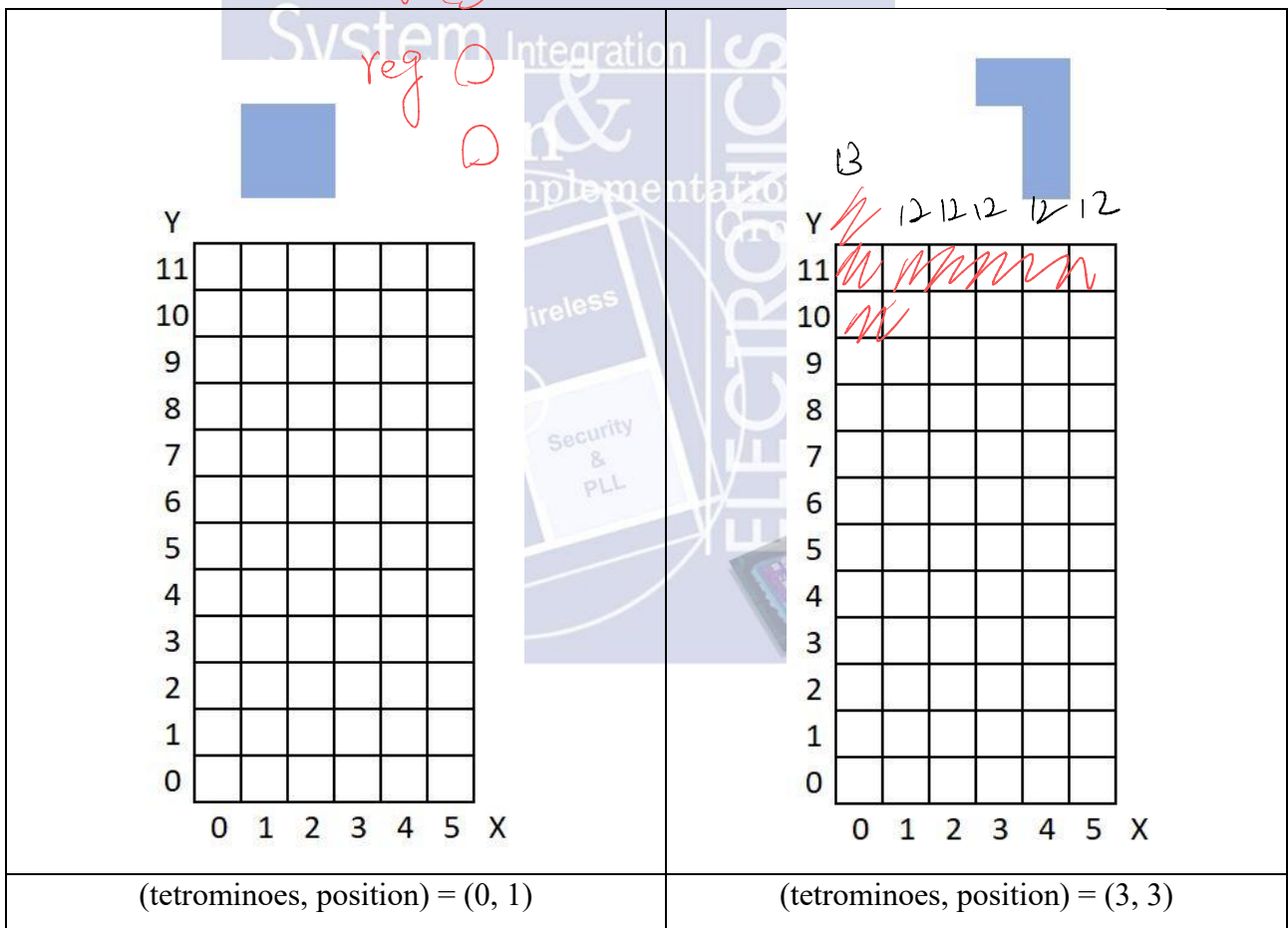


all of them
are 4 blocks

0 1 1 3 4 4

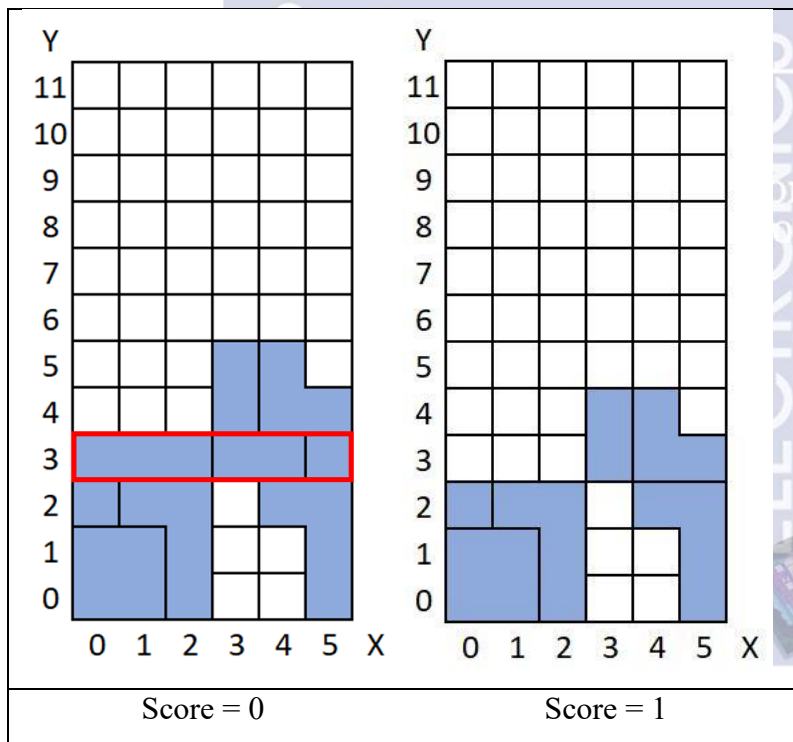
- 1 -





- Description of the placing

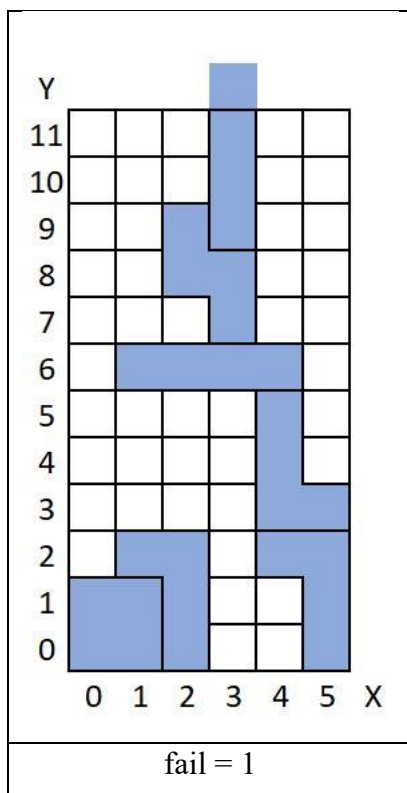
You are asked to determine where the tetrimino should be placed on the Y axis(row). The previous tetriminoes stacks may block the current tetrimino's way down to the bottom, and the current tetrimino would pile on the top of the stack.



- Description of line clearing

When a horizontal line is filled without any gap, it should be cleared from the map. The player can earn a point for each line cleared, which indicates that clearing two rows earns yourself two points. Then, all the tetriminoes stacks above the cleared row **move down by one row simultaneously.**

*check every row
determine how many drop*



- Description of the failing condition of the game

If the tetrimino cannot fit in the map **after eliminating fully filled horizontal lines**, the game is over, which implies the current round of the Tetris terminates early. A new round of game will start, the remaining tetriminoes stacks in the map removed and the score is calculated from zero again (结束)

- Description of the Outputs

For each input set of (tetriminoes, position), you should output the accumulated score in current game round. And output the final 6x12 Tetris map at the end of each round of the game. If the game terminate earlier, not only the Tetris map is outputted, but also a fail signal is outputted to denote the failure of the current round of the game.

Problem Description

Read Pattern

In this lab, you need to read the patterns from the file **Input.txt**. The format of **Input.txt** is outlined below:

2 (2 patterns in this file)

0 (Pattern 0)
 2 1 (a type 2 tetromino enters into the 6(row)x12(col) Tetris map from the second row)
 3 0 (a type 3 tetromino enters into the 6(row)x12(col) Tetris map from the first row)
 .
 . (a series of 16 instructions)
 .
 1 1 (a type 1 tetromino enters into the 6(row)x12(col) Tetris map from the second row)
 1 (Pattern 1)
 7 3 (a type 7 tetromino enters into the 6(row)x12(col) Tetris map from the fourth row)
 .
 . (a series of 16 instructions)
 .
 5 4 (a type 5 tetromino enters into the 6(row)x12(col) Tetris map from the fifth row)

The first number at the beginning of the file is the number of patterns, and the rest of the file is divided into groups of 17 lines.

1. The first line in each group is the index of the pattern.
2. The rest of the 16 lines are the tetrominoes and their initial position that will be delivered into your module respectively.

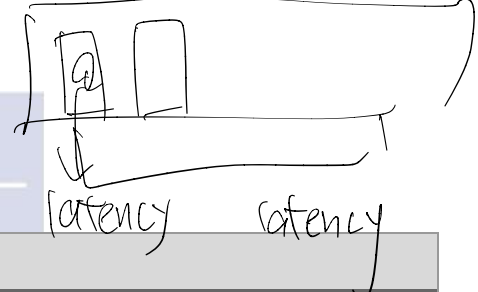
Please note that you should calculate the golden answer in the PATTERN. The golden answer from external files is NOT allowed.

All the above files are stored in 00_TESTBED.

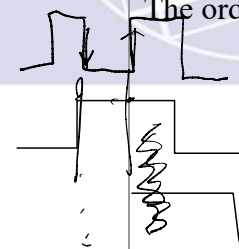
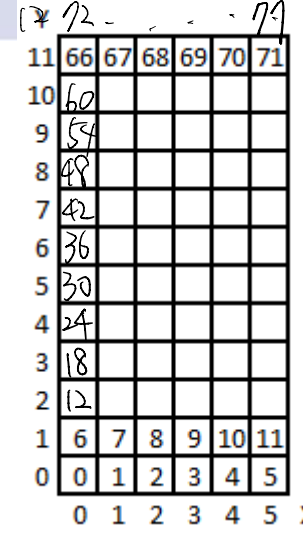
Inputs/Outputs (Tetris.v)

Input Signals

Signal	Bit Width	Description
clk	1	Global clock signal. All signals are sampled on the Positive edge of the global clock.
rst_n	1	Global reset signal. This signal is active LOW.
in_valid	1	High when the input is valid.
tetrominoes	3	Denote the type of the tetromino. Please refer to the Introduction part to check the index of different type of tetromino.
position	3	Mark the location of the leftmost block of the tetromino.
		tetrominoes
		position
		0, 3, 5, 6
		3'd0~3'd4
		1
		3'd0~3'd5
		2
		3'd0~3'd2
		4, 7
		3'd0~3'd3



Output Signals

Signal	Bit Width	Description
score_valid	1	HIGH when the score is valid. The score must be valid for every (tetrominoes, position) set in 1 game round.
score	4	The score the player earns in each round of game.
tetris_valid	1	HIGH when the tetris is valid. Must be HIGH at the end of the current round, which is exactly the same cycle of the last score_valid of the current game round. 只要最後再出現就好
tetris	72	A 12x6 unpacked array. Record the tetris map of the current round. The order of the tetris map is described below:   [0:11] tetris [0:5]
fail	1	A flag that indicates the current round has terminated early. 1: the player loses and the game terminates early. 0: the player wins.

1. All inputs will be changed at the clock negative edge.
2. All input signals are synchronized at the **negative edge** of the clock.
3. For each pattern, **in_valid** should keep high for 1 cycle and **score_valid/tetris_valid** should keep high for 1 cycle as well.
4. The next input pattern will come in 1~4 negative edge of the clock after your **score_valid** is pulled down.
5. Each round of game is composed of 16 input sets. However, **if the game fail and terminate early at the nth input set**, it is **the first input set of the next round of game** that will be provided to your design rather than the (n+1)th input set of the current round.
6. All output signals are checked at the clock negative edge. The **tetris_valid** is HIGH at the cycle

as same as the last cycle of the score_valid in current round.

Specifications

1. Top module name: TETRIS (design file name: TETRIS.v)
2. It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock starting) in your design, you may fail to reset signals.
3. The clock period must less than **40 ns**.
4. The **reset signal (rst_n)** would be given only once at the beginning of simulation. **All output signals should be reset.** The pattern will check the output signal 100ns after the reset signal is pulled low.
5. **The signals score, fail, and tetris_valid must be 0 when the score_valid is low. And the tetris must be reset when tetris_valid is low.**
6. **The latency of each inputs set is limited in 1000 cycles.** The latency is the time of the clock cycles between the **falling edge of the in_valid and the rising edge of the score_valid**.
7. **The score and fail should be correct when score_valid is high. The tetris must be correct when the tetris_valid is high.**
8. **The score_valid and the tetris_valid cannot be high for more than 1 cycle. They must be pulled down immediately in the next cycle.**
9. The input delay is set to **0.5*(clock period)**.
10. The output delay is set to **0.5*(clock period)**, and the output loading is set to **0.05**.
11. The synthesis result of the data type **cannot** include any **latches**.
12. Gate-level simulation **cannot** include any **timing violations** without the notimingcheck command.
13. After synthesis, The slack at the end of the timing report should be **non-negative (MET)**.
14. Any words with “error”, or “congratulation” can’t be used as variable name.

Grading Policy

1. **Function Validity:** 60% (Demo: Use your TETRIS.v, and use the input.txt and pattern.v we provided.)
2. **Pattern:** 20% (Demo: Use your pattern.v, and use the input.txt and TETRIS_encrypted.v we provided.)
 - (3%) Pattern correctness. (Complete the simulation when design is correct.)
 - (3%) SPEC 4
 - (3%) SPEC 5
 - (3%) SPEC 6
 - (5%) SPEC 7

- (3%) SPEC 8
3. **Performance: 20%**
 - $\text{Area} \times \text{total execution latency} \times \text{cycle period}$
 4. The grade of the 2nd demo would be **30% off**.
 5. The design we provided will also check the pattern correctness. If the simulation cannot complete because of the error of pattern, the pattern demo will fail.

Check the result

Check Pattern

1. Include `TETRIS_encrypted.v` in your `TESTBED.v`.
2. Check each specification in terminal.

Specification	Command	Check the result on screen
Pattern correctness	<code>./01_run_vcs_rtl CORRECT</code>	Congratulations
SPEC 4	<code>./01_run_vcs_rtl SPEC_4_{1~5}</code>	SPEC-4 FAIL
SPEC 5	<code>./01_run_vcs_rtl SPEC_5_{1~5}</code>	SPEC-5 FAIL
SPEC 6	<code>./01_run_vcs_rtl SPEC_6_1</code>	SPEC-6 FAIL
SPEC 7	<code>./01_run_vcs_rtl SPEC_7_{1~10}</code>	SPEC-7 FAIL
SPEC 8	<code>./01_run_vcs_rtl SPEC_8_{1~2}</code>	SPEC-8 FAIL

- You can run `./07_check_pattern` to check all above specifications.
- Make sure the keyword printed on the screen is EXACTLY THE SAME as the one listed in the table above. You can print out other information according to your preference, but only ONE keyword from the table above is permissible and it is indispensable.

Check Design

1. Include your `TETRIS.v` in your `TESTBED.v`.
2. Check each specification in terminal.

Specification	Command	Check the result on screen
Pattern correctness	<code>./01_run_vcs_rtl</code>	Congratulations

- You can run `./08_check_design` to check the design functionality.

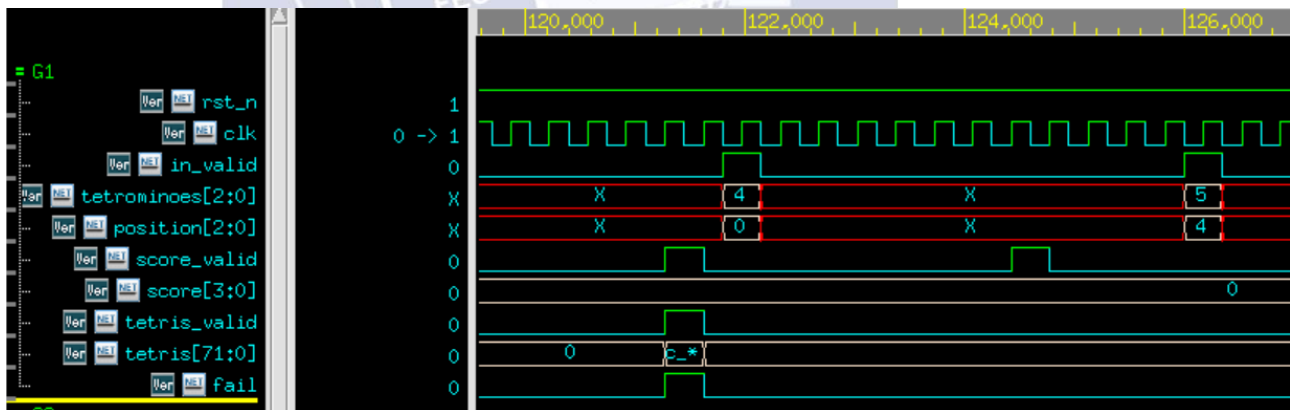
Note

1. Please submit `PATTERN.v`, `TETRIS.v` in **Lab03/EXERCISE/09_SUBMIT**
 - a. 1st_demo deadline: **2024/09/30**(Mon.) 12:00:00
 - b. 2nd_demo deadline: **2024/10/02**(Wed.) 12:00:00
2. **Please upload the following file under 09_SUBMIT:**
 - `PATTERN.v`, `TETRIS.v`

- If your file **violates the naming rule**, you will **lose 5 points**.
 - Encryption of any uploaded files is prohibited or the demo will fail.
3. We provide the TETRIS_encrypted.v for you. You only need to complete the pattern and check the correctness of each design we provided.
 4. You need to implement the TETRIS.v. We only check the functionality using our pattern.
 5. No hidden design and hidden pattern.

Example Waveform

Inputs and Outputs



1. The waveform above is an example of the inputs and the outputs.
2. The next inputs will come in the next 1~4 negative edge of the clock.
3. The two patterns above give the examples that the next pattern will appear after 1 cycle and after 4 cycles, respectively.