NCTU-EE IC LAB – Fall 2019

Lab02 Exercise

Design: Rat in a Maze

Data Preparation

1. Extract test data from TA's directory:

```
% tar xvf ~iclabta01/Lab02.tar
```

2. The extracted LAB director contains:
   a. Exercise
   b. Practice
   c. nLint

Design Description

"Rat in a Maze" is a game, given a N*N binary maze of blocks where source block is the upper left most block i.e., maze[0][0] and destination block is lower rightmost block i.e., maze[N-1][N-1]. A rat starts from source and has to reach the destination. The rat can move only in two directions: down and right. In the maze matrix, 0 means the block is a dead end and 1 means the block can be used in the path from source to destination.

You will get the 8*8 binary maze, then you should output the path that the rat went through. If there are many solutions, so the output direction should be down first, then right.

Fig. 1

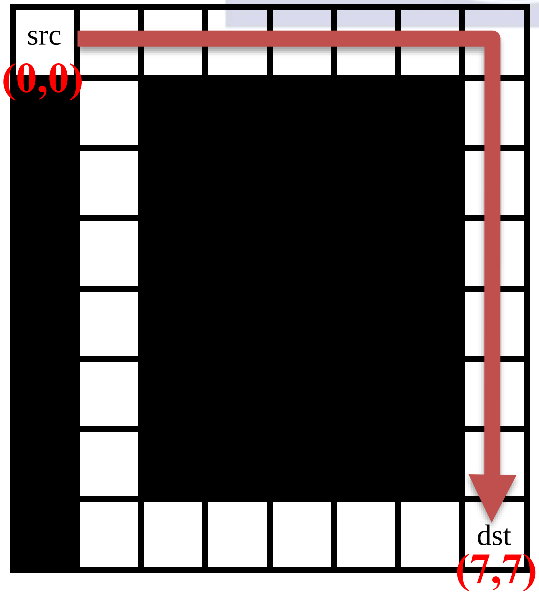
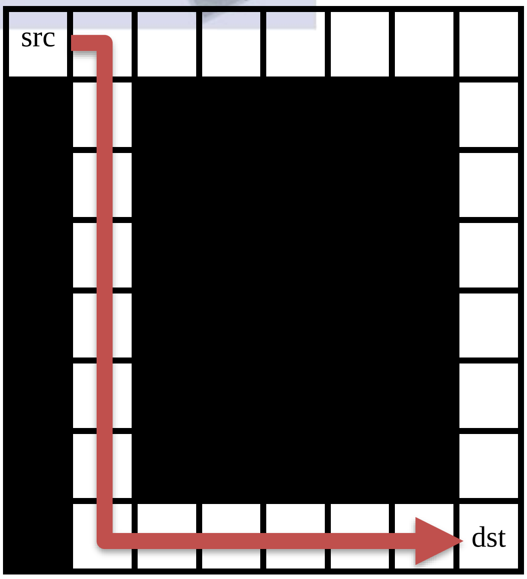


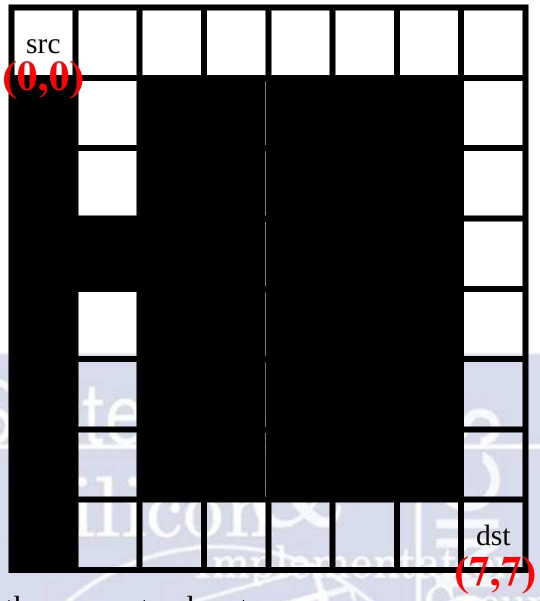
Fig. 2



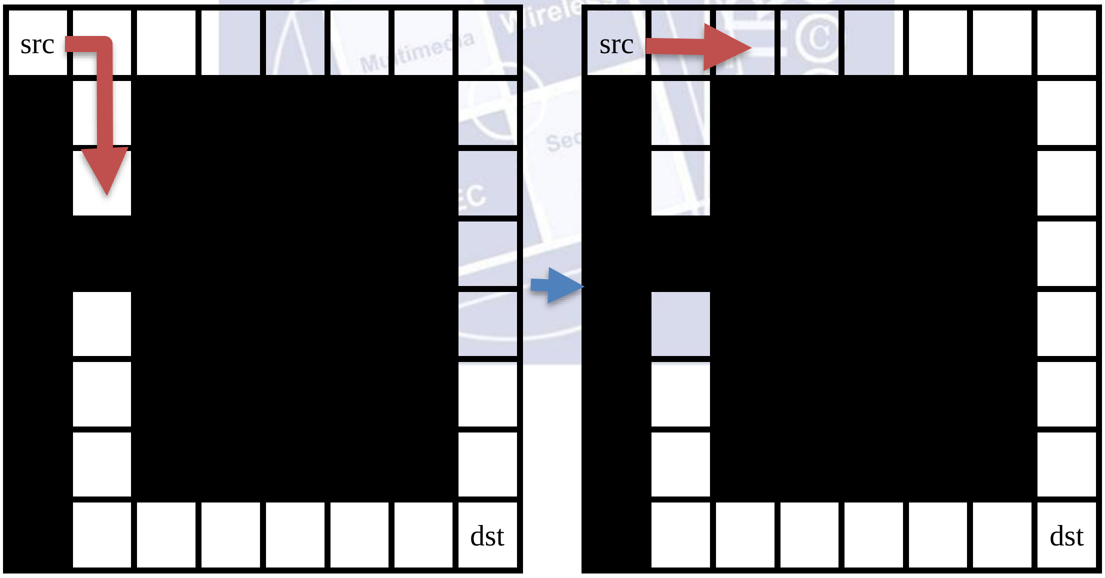
For example, the user gets the maze such as Fig. 1. There are two paths from source to destination i.e., Fig. 1 and Fig. 2. The user must choose path of Fig. 2 because at the position (0, 1), the rat goes down to (1, 1), but not goes right to (0, 2).

Sample solution

The user gets the maze as the below.



Rat goes through the maze step by step.



When the rat goes to the dead road, it should be backtracking to the branch and then goes right. Follow this rule, the rat can finally get the solution.

Inputs

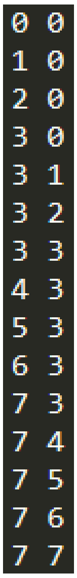
Input	Bit Width	Definition and Description
clk	1	Clock.
rst_n	1	Asynchronous active-low reset.
in_valid	1	High when input signals are valid.
maze	8	An 8-bit maze of blocks row by row from row 0 to row 7 for 8 cycles. 0 means the block is a dead end and 1 means the block can be used.

1. You will receive an 8-bit numbers maze[7:0] for 8 cycles which mean the 8x8 maze. **in_valid** is high simultaneously for 8 cycles.
2. There is ***only 1 reset*** before the first pattern, thus, your design must be able to reset automatically.
3. All inputs will be changed at clock ***negative*** edge.
4. The next input pattern will come in 1~3 cycles after **out_valid** falls.
5. All patterns have solutions.

Outputs

Output	Bit Width	Definition and Description
out_valid	1	High when out is valid. It cannot be overlapped with **in_valid** signal.
out_row	3	Row positions of path step by step.
out_col	3	Column positions of path step by step.

1. Output **out_row[2:0] and out_col[2:0]** are positions form (0, 0) to (7, 7) for 15 cycles.



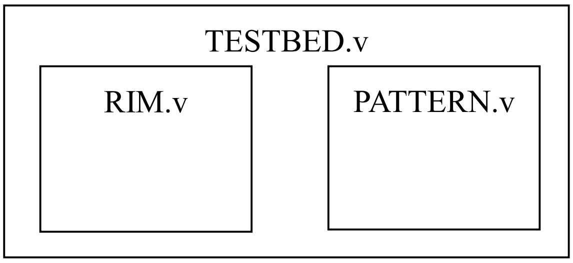
```
0 0
1 0
2 0
3 0
3 1
3 2
3 3
4 3
5 3
6 3
7 3
7 4
7 5
7 6
7 7
```

2. All outputs are synchronized at clock ***positive*** edge.
3. **out_valid, out_row and out_col** should be low after initial reset.
4. **out_valid** should not be raised when **in_valid** is high.
5. **out_valid** is set to high when the output value is valid.
6. The test pattern will check whether your output sequence is correct or not at clock ***negative*** edge.

Specifications

1. Top module name: **RIM** (File name: **RIM.v**)
2. Input pins: **clk, rst_n, in_valid, maze[7:0]**
3. Output pins: **out_valid, out_row[2:0], out_col[2:0]**
4. **It is asynchronous reset and active-low architecture. If you use synchronous reset (considering reset after clock staring) in your design, you may fail to reset signals should be reset after the reset signal is asserted.**
5. The latency of your design in each pattern should not be larger than **200** cycles. The latency is defined in Fig. 3.
6. You can adjust your clock period by yourself, but the maximum period is 5 ns. The precision of clock period is 0.1, for example, 4.5 is allowed, 4.55 is not allowed.
7. The input delay is set to **0.5*(clock period).**
8. The output delay is set to **0.5*(clock period)**, and the output loading is set to 0.05.
9. The synthesis result of data type cannot include any **LATCH** (in syn.log).
10. After synthesis, you can check RIM.area, RIM.timing and RIM.resource.
11. The slack in the end of RIM.timing should be **non-negative** and the result should be **MET**.
12. The design **cannot** be completed by the look-up table approach.

Block Diagram



Grading Policy

1. The performance is determined by the area and latency of your design. The smaller the area and latency is, the higher score you can get.

Functionality: 1de: 75%
Performance: (area) 15%
(simulation time) 10%

2. **The grade of 2nd demo would be 30% off**.
3. The pattern is attached in the archives.

Note

1. Please upload the following file on newE3 platform before **12:00 at noon** on **Sep.30**.
2. The file name should be **RIM_iclab??.v**(?? is your account number) and **cycle_time_iclab??.txt.**(cycle_time should be less than 5.0, and the precision of clock period is 0.1, e.x., 5.0_iclab99.txt.) If the uploaded file violating the naming rule, you will get **5 deduct points** on this Lab.
3. Template folders and reference commands:
   01_RTL/ (RTL simulation) **./01_run**
   02_SYN/ (Synthesis) **./01_run_dc**
   (Check the design which contains **Latch** or not in **syn.log**)
   (Check the design's timing in /Report/RIM.timing)
   03_GATE_SIM/ (GL simulation) **./01_run**
   You can key in **./09_clean_up** to clear all log files and dump files in each folder
4. Sample waveform:

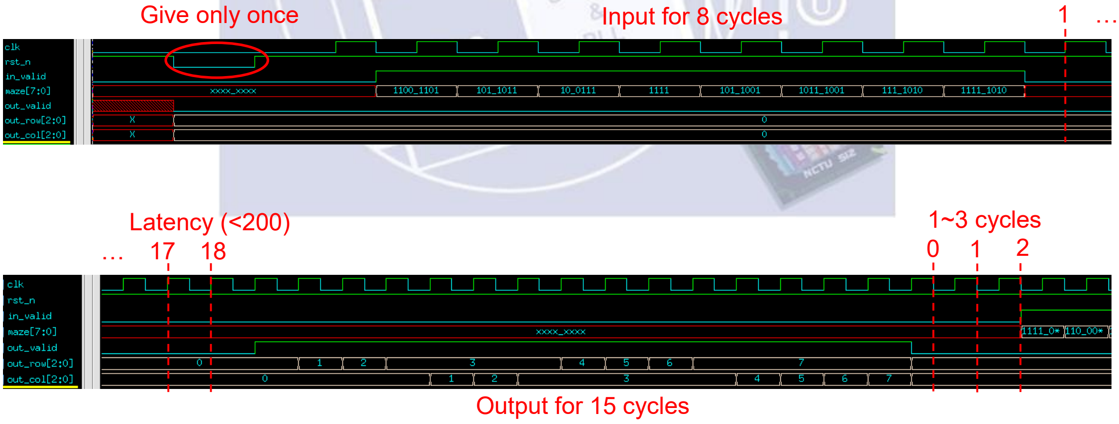


Fig. 3 Latency definition