Maze

Theoretical background

Pathfinding in grid-based structures such as mazes is a classic computational problem with many real-world applications in hardware systems. FPGA-based implementations are particularly useful in scenarios requiring low-latency decision-making and high-throughput environments, such as robotics (autonomous navigation), network routing, or game engines with real-time constraints.

Goal

Create an RTL module that receives a maze from the Judge, finds a path from the input to the output, and sends the correct answer back to the Judge. The size of the maze is provided in the preamble (maximum 64×64).

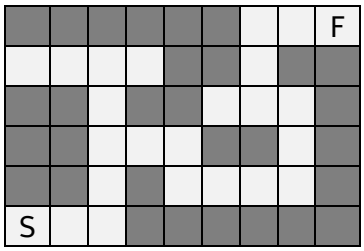
Interface

Signal name	Signal type	Bit length
i_clk	Input	1 bit
i_rst	Input	1 bit
i_valid	Input	1 bit
i_data	Input	8 bits
i_first	Input	1 bit
i_last	Input	1 bit
o_data	Output	8 bit
o_valid	Output	1 bits
o_last	Output	1 bit

Protocol

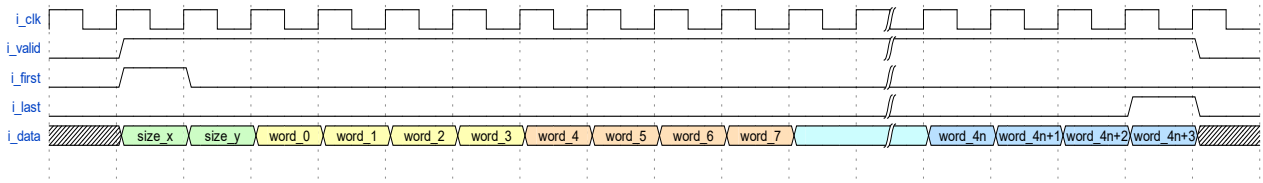
The Judge sends a preamble with the maze size and the maze content. The preamble consists of the first two bytes indicating the size of the maze (size_x - number of columns, size_y - number of rows). The maze content is encoded as follows: 1 - wall, 0 - path. The start is always in the bottom-left corner, and the finish is always in the top-right corner.

Transfer from the Judge:



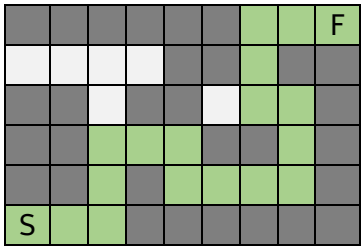
1	1	1	1	1	1	0	0	0
0	0	0	0	1	1	0	1	1
1	1	0	1	1	0	0	0	1
1	1	0	0	0	1	1	0	1
1	1	0	1	0	0	0	0	1
0	0	0	1	1	1	1	1	1

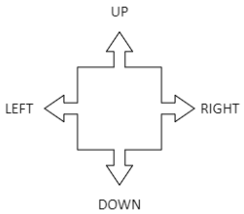
	word_0 – 8 bits				word_1 – 8 bits				word_2 – 8 bits				word_3 – 8 bits			
	MSB_3			LSB_3	MSB_2			LSB_2	MSB			LSB_1	MSB_0			LSB_0
row_0 (word_3210)																F
row_1 (word_7654)																
row_n (word_4n+3/4n+2/4)	S															



Response to the Judge:

Your solution should find the correct path from start to finish and send it back to the Judge.



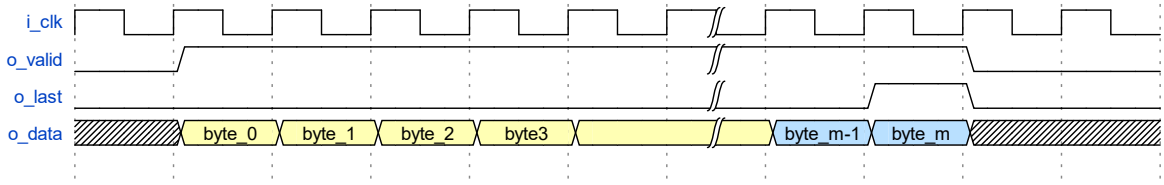


UP – 00

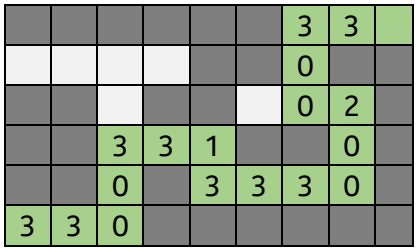
DOWN – 01

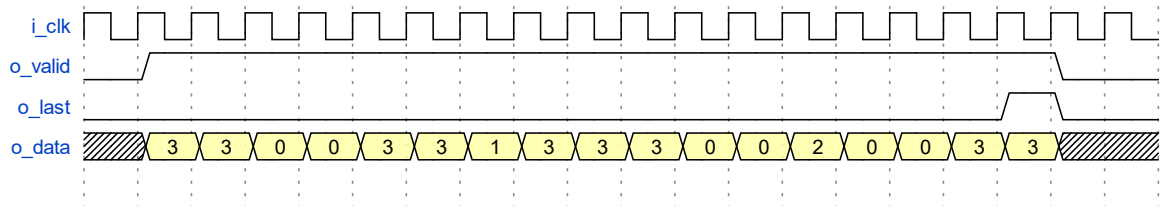
LEFT – 02

RIGHT – 03



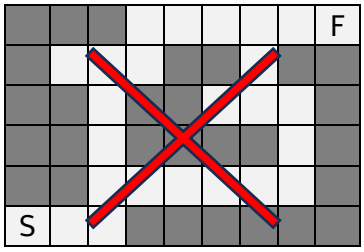
Example response:

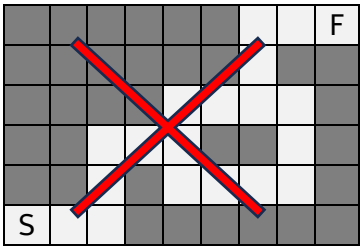




Assumptions

1. Only one valid path (a single solution).
2. No loops.



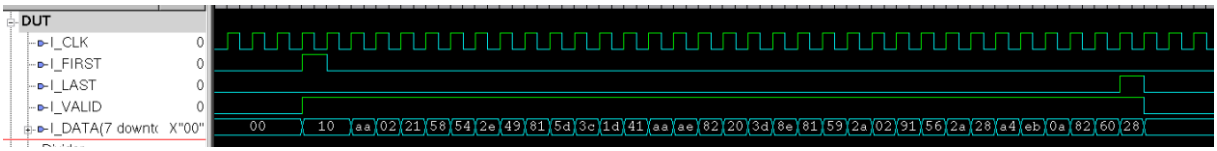


Example

Example 16x16 maze:

1	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	0	0	0	0	1	0	1	1	1	0
0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	1
0	1	0	1	1	1	0	1	0	0	1	1	1	1	0	0
0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0
1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	0
1	0	0	0	0	0	0	1	0	1	0	1	1	0	0	1
0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1	0	1	0	1	0	1	1	0
0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0
1	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1
0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0
0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0

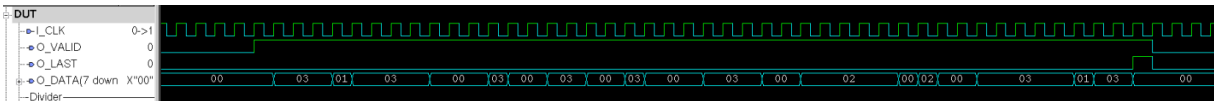
Data transfer:



Solution:

1	0	1	0	1	0	1	0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	1	0	1	0	1	1	0	0	0
0	1	0	1	0	1	0	0	0	0	1	0	1	1	1	0
0	1	0	0	1	0	0	1	1	0	0	0	0	0	0	1
0	1	0	1	1	1	0	1	0	0	1	1	1	1	0	0
0	0	0	1	1	1	0	1	0	1	0	0	0	0	0	1
1	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0
1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0
0	0	1	1	1	1	0	1	1	0	0	0	1	1	1	0
1	0	0	0	0	0	0	1	0	1	0	1	1	0	0	1
0	0	1	0	1	0	1	0	0	0	0	0	0	0	1	0
1	0	0	1	0	0	0	1	0	1	0	1	0	1	1	0
0	0	1	0	1	0	1	0	0	0	1	0	1	0	0	0
1	0	1	0	0	1	0	0	1	1	1	0	1	0	1	1
0	0	0	0	1	0	1	0	1	0	0	0	0	0	1	0
0	1	1	0	0	0	0	0	0	0	1	0	1	0	0	0

Solution transfer:



Evaluation:

Evaluation is done by comparing the output of the task with the value calculated by the reference model in the Judge software. If the values match, the task is considered correctly resolved.

In Development mode, the task will be tested using a single set of data that is randomized on each run. In Evaluation mode, the task will be tested using 3 fixed sets of data, which remain the same across all runs and are identical for every team (for more information, check chapter 2.3.1 **Testing modes** in the **FPGA_Hackathon_2025_Project_Guide** document).

You earn **7 points** for correctly processing each test vector. In Development mode, this value simply indicates that the task passed the test. Evaluation mode, your total base score is calculated by multiplying 7 by the number of test vectors, so **you can earn up to 21 points**.

Additional points may be awarded for **resource utilization and latency** (for more information, check chapter 2.3.2 **Bonus Points** in the **FPGA_Hackathon_2025_Project_Guide** document).