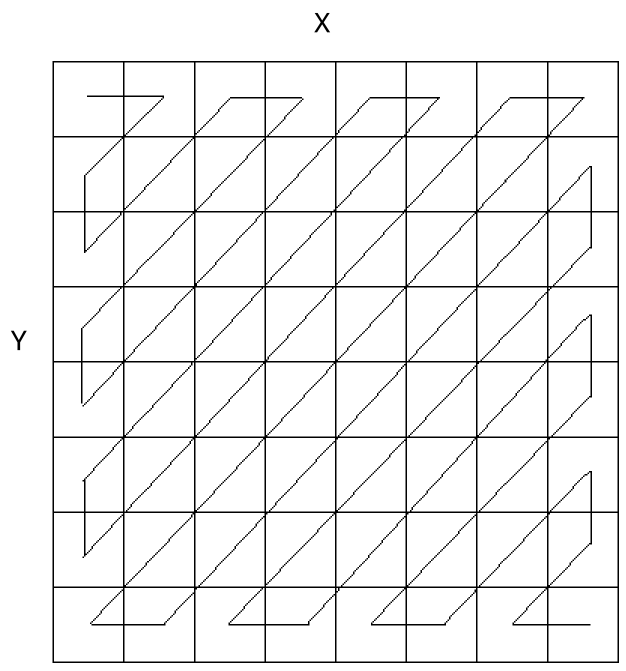
Traverse to the far side of the Moon

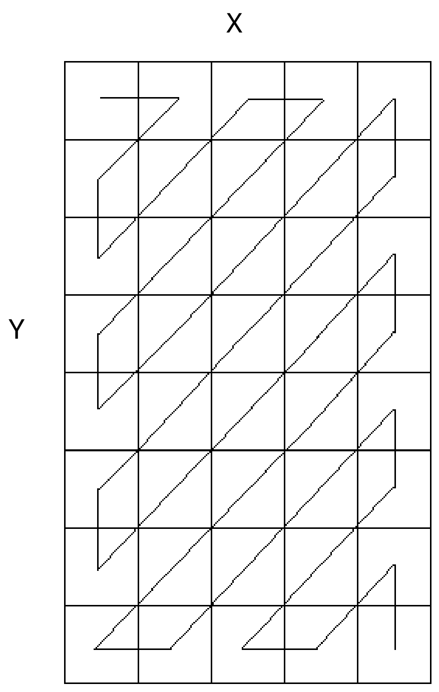
Theoretical background

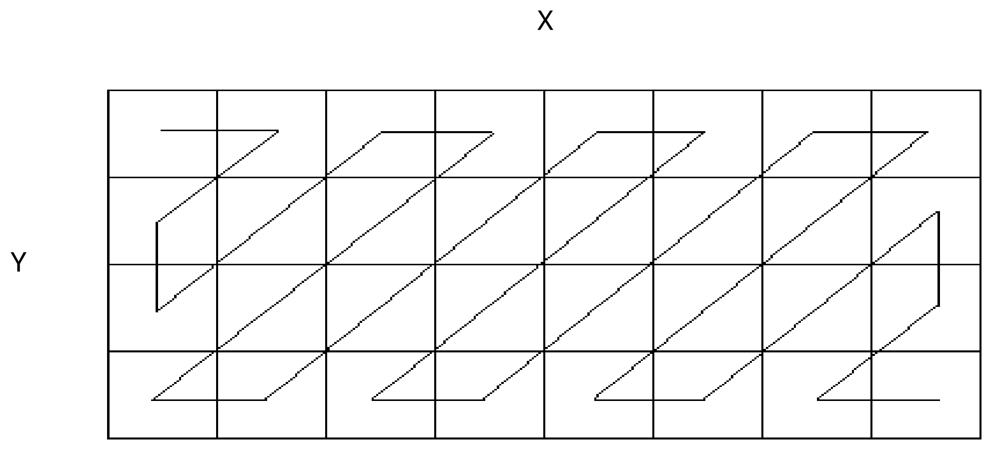
When working with matrices, one of the most fundamental operations is traverse — visiting each element of the matrix in a specific order. Matrix traverse is crucial for solving a variety of computational problems, such as searching, pathfinding, and data manipulation.

Goal

Your task is to create a module that receives a matrix and traverse it according to the pattern shown in the pictures below. Then, it should send the matrix back to the judge according to the new order. The received matrix can have dimensions x*y such as x*y =< (4090)







Provided module

You can edit only the *task_2.sv* file. If you wish, you may add other files for supportive submodules.

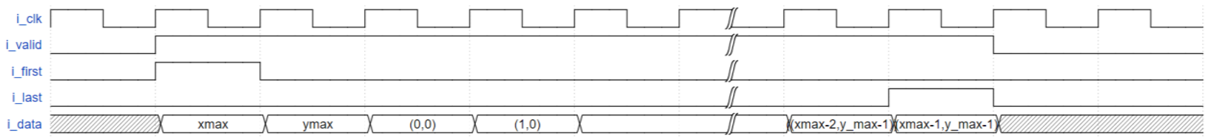
The module has the following inputs and outputs:

Table 1 List of inputs and outputs of the task_2 module.

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

Input and output interfaces

In one packet the task receives a sequence of data on the *i_data* port. The maximal amount of data is 4KB. The beginning of the data block is marked by the *i_first* signal, and the end by the *i_last* signal. The first two bytes are header bytes which contain x_max and y_max values, they define the number of columns and the number of rows. After the header, bytes with the value of the matrix are sent. Each byte represents the value of a different cell of the matrix. They are sent in row-by-row order. Throughout the data transfer, the *i_valid* signal is high.



Task should send response sequence using *o_data* port. Along with the data the *o_valid* signal should be set, and on the last cycle of data the *o_last* signal should go high.

Evaluating the task

The evaluation is done by comparing the sequence returned by the task with the expected order generated by the reference model in the Judge. If both are the same, the task is marked as correctly solved.

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 3 point 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 3 points by the number of test vectors, so you can earn up to 9 points.

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