Maximum Finder

Theoretical background

The capability of identifying the highest value in the data stream has a wide range of applications, from real-time monitoring in industrial settings to financial data analysis and network optimization in telecommunications. By identifying peak values, digital modules can enable crucial insights for decision-making, resource allocation, and performance optimization across various fields. This document explores the design and implementation of such a module.

Goal

Create an RTL module that returns the highest number from the input data stream.

Provided module

You can edit only the *task_1.sv* file. You may add other files for supportive submodules, if needed.

You only get the outline of the ***task_1*** module. Inputs and outputs are declared as shown below in the Table 1.

Table 1 List of inputs and outputs of the task_1 module.

Signal name	Signal type	Bit length
i_clk	Input	1 bit
i_rst	Input	1 bit
i_valid	Input	1 bit
i_data	Input	16-bit (signed, encoded as two's complement)
i_first	Input	1 bit
i_last	Input	1 bit
o_data	Output	16-bit (signed, encoded as two's complement)
o_valid	Output	1 bit
o_last	Output	1 bit

Input and output interfaces

The task receives **a packet of 16-bit signed samples** through the *i_data* interface. The packet contains between 16 and 1024 samples. To detect the last sample, rely on the *i_last* impulse.

Task receives data in the way shown in the figure below:

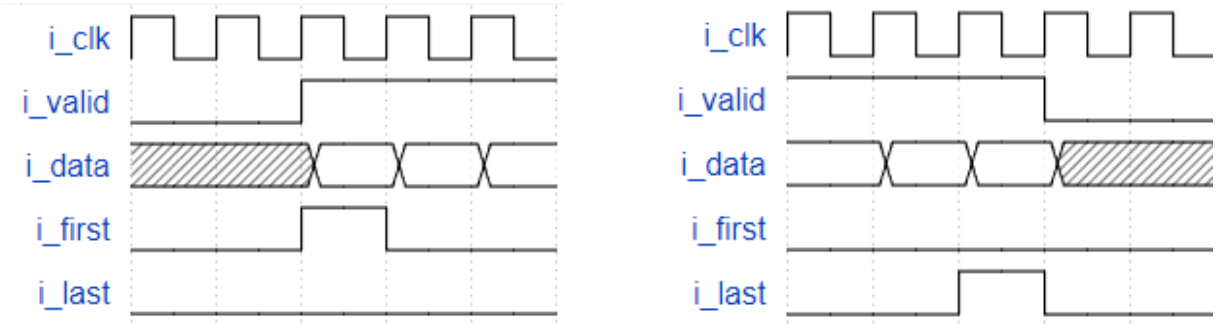

Figure 1 Input waveforms – the start of the packet (left) and the end of the packet (right).

The task should return the highest value that occurred in the input packet.

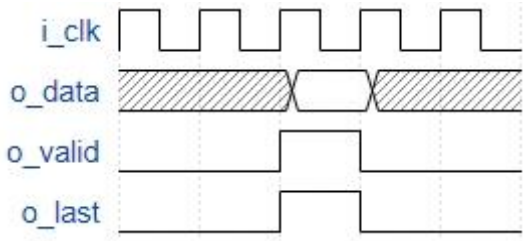

Figure 2 Output waveforms.

Evaluating the task

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 1 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 1 point by the number of test vectors, so you can earn up to 3 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).