Rover navigation

Theoretical background

Navigation systems are one of the most critical components of autonomous vehicles. On Earth we use satellite systems such as GPS, however on the Moon it is not possible to use them.

In scenario, where the rover lost its tracking of current position relative to the base station, it will need to resolve this in the field.

Goal

Create an RTL module which will calculate the distance and angle between the lunar rover and the lunar base station, so the rover will be able to head towards it.

The lunar rover has an omnidirectional antenna, meaning it can transmit and receive signals from any direction. If the antenna transmits a pulse at the appropriate frequency, the lunar base will receive this signal and immediately send back a return pulse.

You have access to antenna module.

When you send the pulse to the *ping* port, antenna will send a radio signal to the base station. The station will reflect the pulse, and you will see this on the *pong* port. After that, the rover will move 1000m in one direction, then you can make a second measurement, then rover will turn 90 degrees counterclockwise and it will move one more time 1000m (as in the figure below), so you can grab third sample of time of flight between the rover and the station.

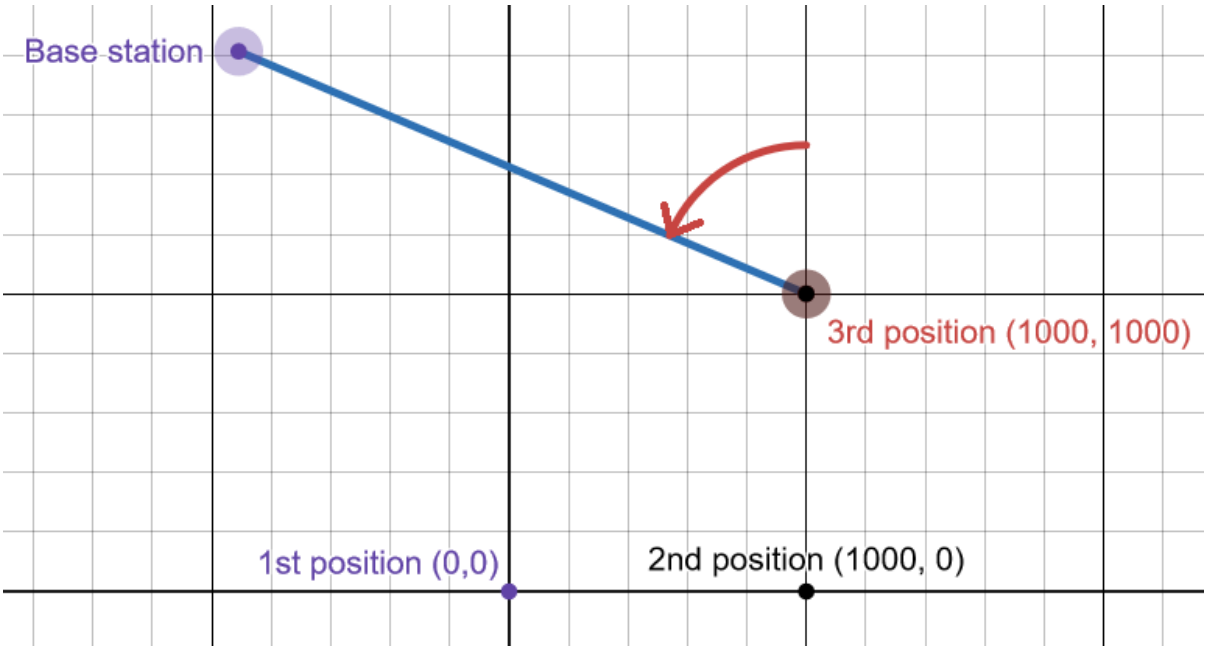


Figure 1 Example position of base station, but it could be anywhere!

Using **trilateration** find an angle (red arc in the figure) of how much lunar rover needs to rotate (counterclockwise) and what is the distance (blue segment in the figure) from the last position to the base station. Assume that the pulse is traveling with the speed of light c= 299 792 458 m/s, and there are no additional delays in the path of the signal. The clock speed is 100MHz.

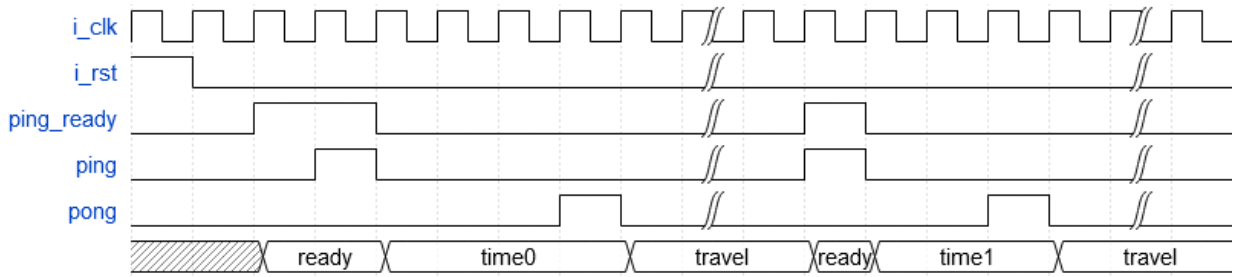
Provided module

You can edit only the *task_9.sv* file. If you wish, you may add other files for supportive submodules.

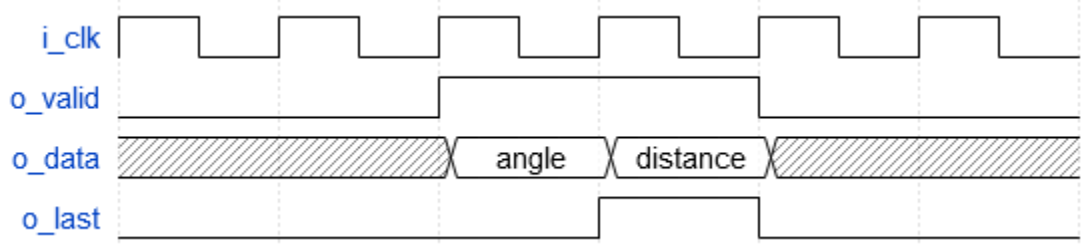
Signal name	Signal type	Bit length
i_clk	Input	1
i_rst	Input	1
ping	Output	1
ping_ready	Input	1
pong	Input	1
o_data	Output	32
o_valid	Output	1

After reset, wait for *ping_ready* signal to be high, then you can send pulse to *ping* and wait for response pulse on *pong*. After asserting *ping*, *ping_ready* deasserts, and when *pong* pulse arrives, rover travels to next measure point. When it is ready for another ping-pong, it will assert *ping_ready*.

Example waveform:



After gathering all three times, calculate the distance and angle, and send it to the output, asserting *o_valid* signal.



Output format:

- Distance: meters, 32-bit floating point (FP32)
- Angle: radians, 32-bit floating point (FP32).

Evaluating the task

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 **8 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 8 by the number of test vectors, so **you can earn up to 24 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).