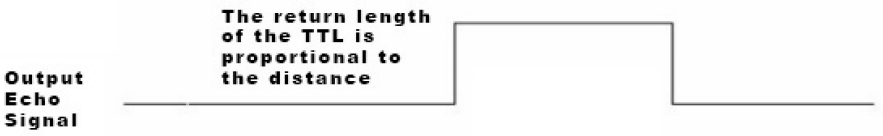
Experiment 3 (Implementation of a Sequence Detector)

Aim

In this experiment, your knowledge to design a Sequence Detector using Finite State Machines will be tested.

Problems

You will design a circuit for obstacle detection which gets the data from an ultrasonic sensor.



You will design a circuit for detecting input patterns such as 010, 0110, 01110 ... 01...10

- When the circuit detects the 010 pattern, it will give 01 as an output which means that the ultrasonic sensor detected an obstacle between 0 and 1 meters.
- When the circuit detects the 0110 pattern, it will give 10 as an output which means that ultrasonic sensor detected an obstacle between 1 and 2 meters.
- When the circuit detects three or more ones between zeros (01110, 011110 etc.), it will give 11 as an output which means that ultrasonic sensor detected an obstacle further from 2 meters.

Notes:

- The output is determined solely by the current state.
- Ignore the initial 1s of the sequence, if any.
- The sensor starts with reset input. Active-high and synchronous reset is used.
- Make sure that the sequence detector also detects overlapping patterns.

Preliminary Work

Before the experiment, you should apply and report 5 step controller process explained in the class as follows:

1. Capture the FSM: Create finite state machine that describes the desired behavior of the controller.
2. Create the architecture: Create a standard architecture by using a state register of the appropriate width and combinational logic with inputs being the state register bits and the finite state machine inputs and outputs being the next state bits and the finite state machine engine.
3. Encode the states: Assign a unique binary number to each state. Each binary number representing a state is known as an encoding. Any encoding is acceptable as long as each state has a unique encoding.
4. Create the state table: Create a truth table for the combinational logic such that the logic will generate the correct FSM outputs and next state signals. Ordering the inputs with state bits first makes this truth table describe the state behavior, so the table is a state table.
5. Implement the combinational logic: Implement the combinational logic using any method.
6. Write the Verilog code of the sequence detector. You are free to write in behavioral or gate level code.
7. Write the Verilog code for the testbench waveform in order to test possible input sequences. Use at least five different 32-bit or longer sequences for your testbench.
8. Verify the functionality of your implementation.

Then, submit your code, and report under the name `<StudentID>_PRE3.zip` through Moodle.