Experiment 4 (Implementation of a Sequence Detector)

Aim

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

Problems

In a custom home alarm system, there are two potential triggers for activating the alarm. The first is in the case of an intruder; when the sensor detects continued disturbance on the door. The second is in the case of fire, when the smoke detector detects heavy smoke inside the house. These sensors give outputs via a transmitter, and a receiver receives their output. The receiver organizes the arriving data, and processes the data sequentially. It changes the output of the system if certain sequences are detected.
You will design a circuit for detecting intruders and fire.

- If the sequence 10001 is detected, this means that there might be a fire, and the receiver should output 01.
- If the sequence 10101 is detected, there might be an intruder, and the receiver should output 11.
- In any other case, the receiver should output 00.

As long as these sequences are detected, output should be given. For example: if the data arriving from the transmitters for a certain time is 100010001, then the output should be 00000000000100000001. Another example: Data sequence 1010101 should give the output 0000000000110011.

Notes:

- The output is determined solely by the current state.
- Ignore the initial 0s 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. (Refer to the example.)

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 the your code, and your report under the name `<StudentID1>_<StudentID2>_PRE4.zip` through Moodle. One submission per group is enough.