Experiment 5 (Implementation of a Sequence Detector)

Aim

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

Problem

You will design a circuit for a sequence detector which will detect repetitive input patterns of 3 in the form of 000 or 111. The sequence detector will also detect overlapping inputs. As an example the input sequence of X should yield the result shown in Y.

- X: 0011 0001 1011 1110 1000 0101 0111 1000
- Y: 0000 0010 0000 1110 0001 1000 0001 1001

Notes:

- The output is determined solely by the current state.
- The output becomes 1 in the same clock cycle as the last input of the pattern is read.
- The detector starts with reset input. Active-high and synchronous reset is used. Reset input takes the current state to the starting state, giving a 0 output, which is not shown in the above X/Y sequence. (You should have a start state which is used as the beginning and is also used when there is a reset input.)
- 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.
9. **Draw the combinational logic on Logisim and add the .circ file to your submission.**

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