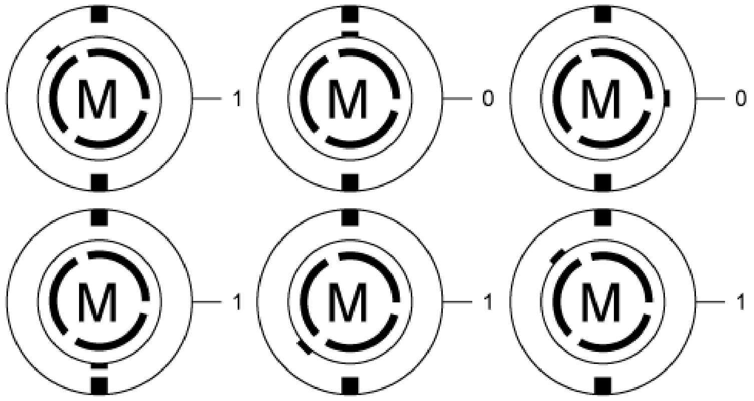
0.1 Experiment 3 (Implementation of a Sequence Detector)

0.1.1 Aim

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

0.1.2 Problems

In a custom rotation detection system, there are two receivers and one transmitter. The transmitter is located on the shaft of the motor and receivers are placed between the shaft and motor cover. Whenever a receiver gets data from the transmitter, it changes the output of the system.



You will design a circuit for detecting the speed of the motor. This circuit should detect input patterns such as 101,1001,10001 ... 10...01

- When the circuit detects more than three 0s between 1s (10001, 100001 etc.), it will give **01** as an output which means motor rotating slowly.
- When the circuit detects 1001 pattern, it will give **10** as an output which means motor rotating at medium speed.
- When the circuit detects 101 pattern, it will give **11** as an output which means motor rotating fast.
- For the other patterns, output should be **00**.

Note:

- The output is determined solely by the current state.

- Ignore the initial 0s of the sequence, if any.
- When **reset** is **HIGH**, the state should became initial state.
- The **reset** should be **synchronous** to the rising edge of the clock.
- There should not be any signals which have a **X or Z output**.

0.1.3 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 combinatorial 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 sequences for your testbench.
8. Verify the functionality of your implementation.
9. Verify that the sequence detector will also detect overlapping patterns.

Then, submit the your code, and your report under the name `<StudentID1>_<StudentID2>_PRE3.zip` through Moodle. One submission per group is enough.