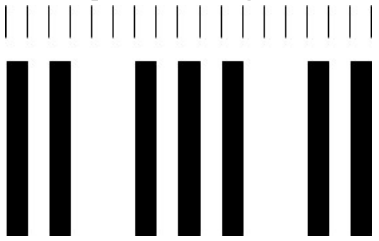
6.4 Experiment 4 (Implementation of a Bar Code Reader)

6.4.1 Aim

In this experiment, your knowledge to design a sequential circuit will be tested. Please refer to book chapter 7 for more details.

6.4.2 Problem

A bar code represents bits as alternating white and black bands. The black bands are of uniform width, while the white bands have width either equal to, or triple of, the width of the white bands. Below is an example of a code-word using the bar code. The tick marks on top show the single widths.



Assume that a bar-code reader translates the bands into symbols, B for black, W for white, one symbol per single width. Thus the symbol sequence for the code-word above would be

B W B W W W B W B W B W W W B W B

A bar pattern represents a binary sequence as follows: a 0 is encoded as BWB, while a 1 is encoded as BWWW. The code can start only with B so the finite state machine will idle until it receives B. After that, when it receives its first W, it knows that the code has started. The transducer will give output 0 or 1 as soon it has determined the next bit from the bar pattern. If the bit is not known yet, it will give output _ . Thus for the input sequence above, the machine will produce;

_ _ 0 _ _ 1 _ _ 0 _ 0 _ _ 1 _ _ 0

B W B W W W B W B W B W W W B W B

The machine will output the symbol end when it subsequently encounters two B's in row, at which point it will return to its initial state.

The machine will output _ if an unexpected sequence occurs. For example, see the following sequence;

_ _ 0 _ _ _ _ _ 1 _ _ 0

B W B W W B W W W B W B

Note: The machine starts with reset input. Active-high and synchronous reset is used.

Hint: You need to find and apply a suitable input/output representation to convert symbol inputs/outputs to binary inputs/outputs.

6.4.3 Preliminary Work

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 standard architecture by using a state register (of the appropriate width) and combinational logic.
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 regular expression 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 symbols long or longer sequences for your testbench.
8. Verify the functionality of your implementation.
9. Verify that the detector detects all the example inputs given above.

Submit your code and report considering the submission rules described in Section 3.1. One submission per group is required.

6.4.4 Lab Work

1. Repeat all the procedures done in the preliminary work for the problem given in the lab section. Fill the report, write the Verilog code, and demonstrate your implementation to the assistants.