Please attach this cover sheet to your completed homework

ECE 324 Homework 4: Shift Register Simulation

Name: Alex Blake

Exercise	**Course outcome**	**Grade**
Homework4	2.a, 7.b	/30

2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.

7.b. Employ appropriate learning strategies such as communicating with an expert, using external resources, experimentation, simulation, etc.

Introduction

In this assignment, students were asked to create a testbench that uses an existing universal shift register module given by the professor. It requires students to learn and verify an existing module along with sequential logic in a testbench.

Procedure

The first step is to create a testbench module for the register and define the logic required for the inputs and outputs of the module. Once those are done, it is time to instantiate a universal shift register module within the testbench and have the logic in the testbench assigned to the corresponding inputs and outputs.

Since the shift register requires a clock input (clk), this must be generated. To do this, a simple always block is used, and every T/2, the clock is inverted. Thus, there is a clock cycle with period T generated.

The rest of the test is done within an initial loop because it does not require more than one loop through. The loop tests the following functionalities:

- Load
- No operation (hold)
- Shift left 4 places
- Shift right 4 places
- Asynchronous reset

Each of these functionalities are in their own (commented) sections of the block which expresses how they accomplish the functions above. This completes the construction of the testbench. Simulate and verify the outputs match the expected values.

Results

The testbench worked exactly as predicted and gave the output below in Figure 1. The plot can be broken down into its functions explained in the procedure above.

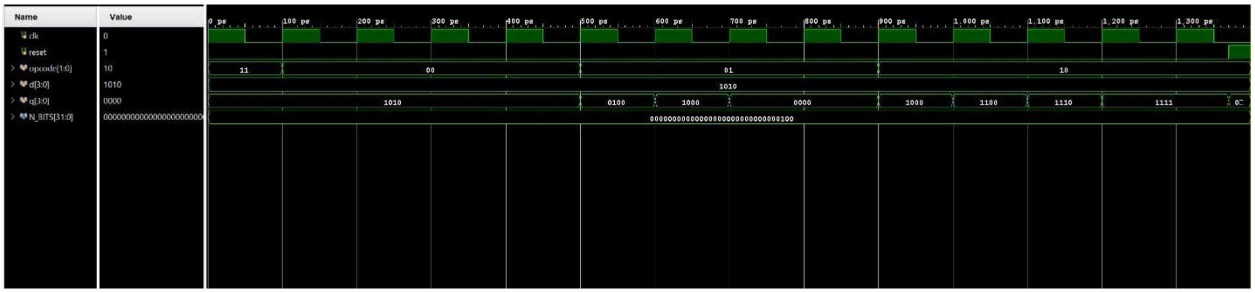

Figure 1: Waveform Output of Testbench (zipped in case unreadable)

The first function used here is between 0 and 100ps. The opcode is set to 2'b11 which indicates a load. The value coded to d is 4'b1010, so during this clock cycle, d is loaded into q, thus q=d. This is exactly what happens in Figure 1.

The second function occurs from 100ps to 500ps (4 clock cycles). The opcode is set to 2’b00 which means it should do no operation. As expected, q remains unchanged throughout these 4 cycles.

The third function is from 500ps to 900ps (4 clock cycles). The opcode is set to 2’b01 which means q should shift left. Since the assignment asked to shift left 4 times, 4 clock cycles were used; the register shifts once a clock cycle. The values below are shown and expected outcomes:

1010 → 0100 → 1000 → 0000 → 0000

In the shift register logic, the least significant value of d is loaded in which means 0’s are shifted in.

The fourth function is from 900ps to 1400ps (4 clock cycles). The opcode is set to 2’b10 which means the register should be shifting right. Since d is 4’b1010, the most significant bit is 1 which is shifted in each cycle giving the following:

0000 → 1000 → 1100 → 1110 → 1111

This output is exactly what Figure 1 shows, so this function works as expected.

The final function is an asynchronous reset. To prove it is asynchronous, the reset input is driven high while the clock is low, which is seen in the last clock cycle. Unlike the rest of the functions that require the clock to be high, this function immediately reset the value of q to 4’d0. This concludes the requirements of the assignment.

Conclusion

This assignment was a great way to practice creating sequential testbenches to test existing modules. It requires students to read and understand how a module works from code and incorporate its functionality into a working testbench and understanding how the logic works in sequential order.