ELEC-4200

# Digital System Design

FROM: Jacob Howard

TO: Prof. Ujjwal Guin

DUE DATE: 3/4/21

## Lab 7

## Introduction

The goal of this lab was to familiarize ourselves with various types of registers and various types of counters. A register stores bits of information in such a way that systems can write to or read out all the bits simultaneously. Most of the tasks were verified as functional by the TA.

## Task 1

In Task 1, we were asked to design a 4-bit register with synchronous reset and load. This code was provided for us so I won't show it in the report, but we still needed to develop a testbench and simulate the design. The simulation code can be found in *Testbench 1* and the simulation can be seen in *Figure 1* below.

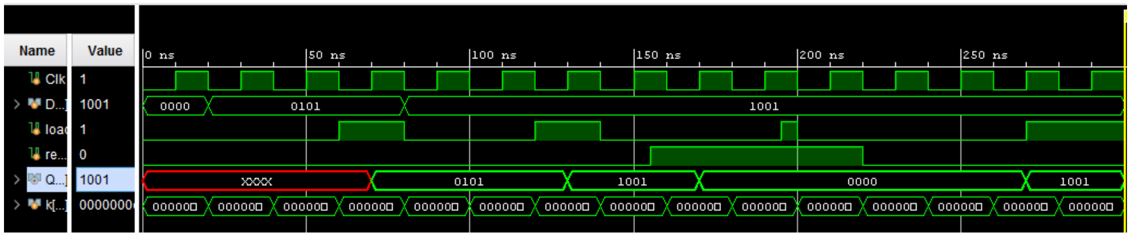

*Figure 1*

```
initial begin
        #300 $finish; //runs simulation 100 times
        end

        initial begin
        load = 0;
        reset = 0;
        D = 0;
        #60 load = 1; //at 60ns
        #20 load = 0; //80ns
        #40 load = 1; //120 ns
        #20 load = 0; //140 ns
        #15 reset = 1; //122 ns
        #40 load = 1; //195 ns
        #5 load = 0; //215 ns
        #20 reset = 0; //240 ns
        #50 load = 1; //280ns
        end

        initial begin
        D = 0;
        #20 D = 4'b0101;
        #60 D = 4'b1001;
        end

         //clock
           initial begin
           for (k = 0; k <= 100; k = k+1)
           begin
               Clk = 0;
               #10 Clk = 1;
               #10;
               Clk = 0;
               end

end
endmodule
```

*Testbench 1*

## Task 2

In Task 2, we were asked to design Model a 4-bit register with synchronous reset, set, and load signals. No testbench was required for Task 2. Only physical testing was required. The code for this task can be seen below in *Code 2*.

```
always @(posedge Clk)
if (reset)
begin
Q <= 4'b0;
end
else if (set)
begin
Q <= 4'b1111;
end
else if (load)
begin
Q <= D;
end
endmodule
```

*Code 2*

## Task 3

In Task 3, we were asked to design a 1-bit delay line shift register. We were asked to develop a testbench to simulate the design. Simulation can be seen below in *Figure 2*, code can be seen in *Code 3*, and testbench code can be seen in *Testbench 3* below.

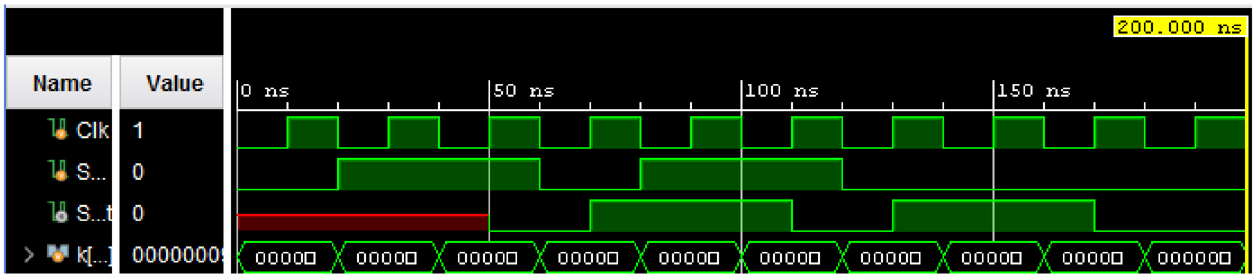

*Figure 2*

```
always @(posedge Clk)

shift_reg <= {shift_reg[1:0], ShiftIn};
assign ShiftOut = shift_reg[2];

endmodule
```

*Code 3*

```
initial begin
        #200 $finish; //runs simulation 100 times
        end
        //clock
        initial begin
        for (k = 0; k <= 1000; k = k+1)
        begin
        Clk = 0;
        #10 Clk = 1;
        #10;
        Clk = 0;
        end
        end
      initial begin
      ShiftIn = 0;
      #20 ShiftIn = 1;
      #40 ShiftIn = 0;
      #20 ShiftIn = 1;
      #40 ShiftIn = 0;
      end

endmodule
```

*Testbench 3*

## Task 4

In Task, we were asked to model Model a 4-bit parallel in left shift register. We then were asked to create a testbench to simulate the design. The simulation can be seen in *Figure 3*, the code can be seen in *Code 4*, and the testbench can be seen in *Testbench 4* below.

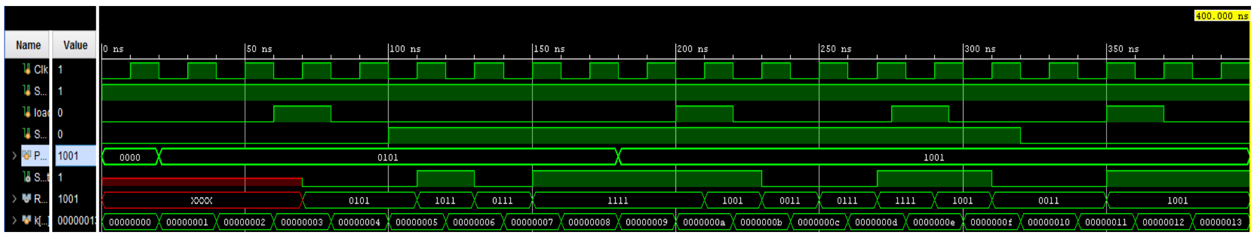

*Figure 7*

```
always @(posedge Clk)
if(load)
shift_reg <= ParallelIn;
else if (ShiftEn)
shift_reg <= {shift_reg[2:0], ShiftIn};
assign ShiftOut = shift_reg[3];
assign RegContent = shift_reg;
endmodule
```

*Code 4*

```
initial begin
        #400 $finish; //runs simulation 100 times
        end
        //clock
        initial begin
        for (k = 0; k <= 1000; k = k+1)
        begin
        Clk = 0;
        #10 Clk = 1;
        #10;
        Clk = 0;
        end
        end
        //load
      initial begin
      load = 0;
      #60 load = 1;
      #20 load = 0;
      #120 load = 1;
      #20 load = 0;
      #55 load = 1;
      #20 load = 0;
      #55 load = 1;
      #20 load = 0;
      end
      //ShiftEn and SHiftIn
      initial begin
      ShiftIn = 1;
      ShiftEn = 0;
      #100 ShiftEn = 1;
      #220 ShiftEn = 0;
      end
      //ParallelIn
      initial begin
      ParallelIn = 0;
      #20 ParallelIn = 4'b0101;
      #160 ParallelIn = 4'b1001;
      end
endmodule
```

*Testbench 4*

## Task 5

In Task 5, we were asked to write a model for a 4-bit serial-in parallel-out shift register. We were also required to write a testbench for this. The code for this task can be seen in Code 5 below and the testbench can be seen in Testbench 5.

```
always @(posedge Clk)

if(ShiftEn)
shift_reg <= {shift_reg[2:0],
ShiftIn};
assign ShiftOut = shift_reg[3];
assign ParallelOut = shift_reg;

endmodule
```

*Code 5*

```
initial begin
        #400 $finish; //runs simulation 100 times
        end
        //clock
        initial begin
        for (k = 0; k <= 1000; k = k+1)
        begin
        Clk = 0;
        #10 Clk = 1;
        #10;
        Clk = 0;
        end
        end

        initial begin
        ShiftEn = 0;
        #40 ShiftEn = 1;
        #40 ShiftEn = 0;
        #40 ShiftEn = 1;
        #40 ShiftEn = 0;
        #80 ShiftEn = 1;
        #40 ShiftEn = 0;
        #40 ShiftEn = 1;
        #40 ShiftEn = 0;
        end
        initial begin
        ShiftIn = 1;
        #200 ShiftIn = 0;
        end
endmodule
```

*Testbench 5*

## Task 6

In task 6, we were asked to design an 8-bit counter using T flip-flops, extending the above structure to 8-bits. The code for this task can be seen in Code 6. We were also required to write a testbench which can be seen below in Testbench 6.

```
TFlipFlop U1 (.t(t), .enable(enable), .Clk(Clk),
.clr(clr), .q(q1));
and (t2, t1, q1);
TFlipFlop U2 (.t(t2), .enable(enable), .Clk(Clk),
.clr(clr), .q(q1));
and (t3, t2, q2);
TFlipFlop U3 (.t(t3), .enable(enable), .Clk(Clk),
.clr(clr), .q(q3));
and (t4, t3, q3);
TFlipFlop U4 (.t(t2), .enable(enable), .Clk(Clk),
.clr(clr), .q(q1));


endmodule
```

*Code 6*

```
initial begin
        #400 $finish; //runs simulation 100 times
        end
        //clock
        initial begin
        for (k = 0; k <= 1000; k = k+1)
        begin
        Clk = 0;
        #5 Clk = 1;
        #5;
        Clk = 0;
        end
        end
        initial begin
        enable = 0;
        #20 enable = 1;
        #80 enable = 0;
        #40 enable = 1;
        end
        initial begin
        clr = 0;
        #60
        clr = 1;
        end
endmodule
```

*Testbench 6*

## Task 7

In task 7, we were asked to modify our code from task 6 so the 8-bit counter uses D flip-flops. Sadly I did not complete this part of the lab so I do not have any code to provide for this task.

## Task 8

In task 8, we were asked to model a 4-bit down-counter with synchronous load, enable, and clear as given in the code above. We were also required to write a testbench to simulate the design. This task was quite challenging and I never could complete the code for this task. Therefore, I do not have any code or design to provide for this task.

## Conclusion

In conclusion, this lab was very helpful for us to understand how to model different types of registers and counters. I did not feel like the lab as a whole was difficult, although I did run into some challenges. Some challenges I was able to solve with help of the Ta, while others, I could not figure out how to complete. Overall, this lab was not too bad, but the workload of having 8 steps can be difficult, especially when some tasks are quite harder than others.