CPE 300L
DIGITAL SYSTEM ARCHITECTURE AND DESIGN LABORATORY

LABORATORY 2
SEQUENTIAL CIRCUIT DESIGN

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
UNIVERSITY OF NEVADA, LAS VEGAS

OBJECTIVE
Learn on various implementations of latches and flip-flops, Verilog models, synchronous and asynchronous operation and asynchronous system design.

INTRODUCTION
In this lab Altera Quartus II software is used to verify the correctness of the circuits modeled in the Verilog. Both structural models and models with component instantiations are used. Experiments are done for both synchronous and asynchronous cases.

LAB DELIVERIES

PRELAB

1. SR latch
Write the structural Verilog codes for SR latch on NOR gates. Use a) gate logic level model and b) instances of Verilog gates.
Examples for 3-input AND:
   a) assign y = a & b & c
   b) and3 andgate(a, b, c, y)
Do the simulation in Quartus and get the waveform. Put the codes and waveforms in the prelab document.

2. Implementation of SR latch: Structural Verilog Models
Analyze the Verilog code implementing the gated SR latch (Fig. 1).
Implement both a) and b) codes and get waveforms. What is the difference between these two codes?

3. Enabled D latch
1. Generate a Verilog file using the style of code in Figure 3 for the gated D latch. Use the synthesis keep directive to ensure that separate logic elements are used to implement the signals $R, S_g, R_g, Q_a,$ and $Q_b$.

For latches in Fig. 3.1. and Fig. 3.2. write structural codes, simulate the designs and verify its correctness by checking the waveforms.
4. Asynchronous set/clear
Add asynchronous Set’ and Clear’ inputs to the design of Gated D latch, as shown in Fig. 4. Write the structural Verilog code.

![Gated D latch with asynchronous inputs](image)

Fig. 4. Gated D latch with asynchronous inputs

Simulate the Verilog code for the design to get the waveforms. Verify correctness of the design in the waveforms, make sure that all the cases are tested, including both synchronous and asynchronous signals.

<table>
<thead>
<tr>
<th>Prelab part</th>
<th>Delivery</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a. Waveform</td>
</tr>
<tr>
<td></td>
<td>b. Verilog code</td>
</tr>
<tr>
<td>2</td>
<td>c. Waveform for code 2.a) – SR latch with component instantiation</td>
</tr>
<tr>
<td></td>
<td>d. Waveform for code 2.b) – SR latch structural model</td>
</tr>
<tr>
<td>3</td>
<td>a. Verilog code and waveform for 3.1 – Gated D latch</td>
</tr>
<tr>
<td></td>
<td>b. Verilog code and waveform for 3.2 – Gated D latch with multiplexer</td>
</tr>
<tr>
<td>4</td>
<td>a. Verilog code and waveform</td>
</tr>
</tbody>
</table>
LAB EXPERIMENTS

1. D flip-flop
   1.1. Master-Slave D flip-flop
   Write a structural code for Master-Slave D flip-flop in Fig. 5.1 using Fig. 5.2 latches for Master and Slave construction.

   ![Fig. 5.1. D flip-flop with Master-Slave gated latches](image)

   ![Fig. 5.2. Gated D latch](image)

   Simulate the code, check the correctness using waveforms. Check the logic utilization and component count in the simulation report (Total utilization, Combinational ALUTs, Dedicated logic registers).

   1.2. 6-NAND D flip-flop
   Write a structural code for a smaller circuit of D flip-flop (6 NAND gates) in Fig. 6. Implement, check the correctness, and check the logic utilization and component count.

   ![Fig. 6. 6-NAND D flip-flop](image)

   1.3. 6-NAND D flip-flop with synchronous inputs
   Modify the circuit (structural Verilog model) from 1.2. to make the D latch with synchronous active low Set and Reset (sampled at Clock).
2. JK flip-flop construction

Fig. 7. shows the Truth Table and JK circuit based on D flip-flop

![JK flip-flop based on D](image)

**Fig. 7.1 JK flip-flop based on D**

<table>
<thead>
<tr>
<th>Q</th>
<th>Q_{next}</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>×</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>×</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>×</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>×</td>
<td>0</td>
</tr>
</tbody>
</table>

**Fig. 7.2. Truth table of JK flip-flop**

Implement three different Verilog models of JK flip provided below and verify the operation. Get waveforms for all the models.

```verilog
// #1 JK
module JKFF (Q, QN, J, K, Clock);
output Q, QN; // data output
input J, K; // data input
input Clock;
reg Q;
always @(posedge Clock)
begin
Q = J & (~Q) | (~K) & Q;
end
assign QN = ~Q;
endmodule

// #2 JK
module JKFF (Q, QN, J, K, Clock);
output Q, QN; // data output
input J, K; // data input
input Clock;
reg Q;
always @(posedge Clock)
begin
if ({J, K} == 2'b01) Q = 1'b0;
else if ({J, K} == 2'b10) Q = 1'b1;
else if ({J, K} == 2'b11) Q = ~Q;
else Q = Q;
end
assign QN = ~Q;
endmodule

// JK #3
module JKFF (Q, QN, J, K, Clock);
output Q, QN; // data output
input J, K; // data input
input Clock;
reg Q;
always @(posedge Clock)
begin
  case ({J, K})
  2'b01: Q = 1'b0;
  2'b10: Q = 1'b1;
  2'b11: Q = ~Q;
  default: Q = Q;
  endcase
end
assign QN = ~Q;
endmodule
```
3. JK with synchronous clear
Modify JK flip-flops from 2. to include synchronous ClearN. Draw the circuit and provide a behavioral model of your design. Synthesize, and verify the synthesized circuit in RTL viewer.

4. FSM Design
Model the FSM machine defined as follows:

<table>
<thead>
<tr>
<th>PS</th>
<th>NS</th>
<th>X=0</th>
<th>X=1</th>
<th>Z</th>
<th>X=0</th>
<th>X=1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>X=0</td>
<td>X=1</td>
<td></td>
<td>X=0</td>
<td>X=1</td>
</tr>
<tr>
<td>S0</td>
<td>S1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S1</td>
<td>S3</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S2</td>
<td>S4</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S3</td>
<td>S5</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S4</td>
<td>S5</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S5</td>
<td>S0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S6</td>
<td>S0</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Use always block to represent the combinational part of the circuit and generate the next state information and output. Use another always block to model the state register.

5. Asynchronous system design
In a car security system, we usually want to connect the siren in such a way that the siren will activate when it is triggered by one or more sensors. In addition, there will be a master switch to turn the system on or off. Let us assume that there is a car door switch D, a vibration detector switch V, and the master switch M. We will use the convention that when the door is opened D = 1, otherwise, D = 0. Similarly, when the car is being shaken, V = 1, otherwise, V = 0. Thus, we want the siren S to turn on, that is, set S = 1, when either D = 1 or V = 1, or when both D = 1 and V = 1, but only for when the system is turned on, that is, when M = 1. However, we turn off the system, and either enter or drive the car, we do not want the siren to turn on. Hence, when M = 0, it does not matter what values D and V have, the siren should remain off.
As a next step, what we really want is to have the siren remain on, even after both the door and vibration switches are off. In order to do so, we need to remember the state of the siren. In other words, for the siren to remain on, it should be dependent not only on whether the door or the vibration switch is on, but also on the fact that the siren is currently on. We can use the state of a SR latch to remember the state of the siren (i.e., the output of the latch will drive the siren). The state of the latch is driven by the conditions of the input switches.

Design an asynchronous circuit for the car alarm system with the above requirements and use NAND based SR latch. Start with the truth Table, obtain the logic equation, simplify, and incorporate the latch. Write the structural Verilog Module and synthesize. Obtain the circuit (RTL viewer). Demonstrate the operation on the board. Instead of Siren sound use the LED output.
POSTLAB REPORT:

Include the following elements in your postlab report:

1. Theory of operation
   a. Provide basic info about D and JK flip-flops
   b. Write few sentences: explain how waveforms are used to check the correctness of a circuit

2. Your prelab report – include as a section 2 of the postlab

3. Results of the experiments

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Delivery</th>
</tr>
</thead>
</table>
| 1.1        | a. Verilog code and waveform  
             b. Logic utilization and component count from the compilation report |
| 1.2        | a. Verilog code and waveform  
             b. Logic utilization and component count from the compilation report |
| 1.3        | a. Verilog code and waveform  |
| 2          | b. Waveforms for three JK models |
| 3          | a. Verilog codes and the waveforms for 3 JK flip-flops  
             b. Circuits from RTL viewer for 3 JK flip-flops |
| 4          | a. The diagram of the FSM machine  
             b. Verilog code  
             c. Simulation results |
| 5          | a. Truth table  
             b. Equation for the asynchronous design  
             c. Simplified equation  
             d. Asynchronous design with the latch  
             e. Structural Verilog code  
             f. Circuit from RTL viewer |

4. Answer the questions:
   a. What is the meaning of component count from the Quartus compilation report?
   b. What is the advantage of using asynchronous Set and Reset in case of D flip-flop? Give an example.
   c. Regarding Logic Utilization from the Quartus Compilation Report – what are ALUT and Dedicated Logic Registers?

5. Conclusions
   a. Write down your conclusions, things learned, problems encountered during the lab and how they were solved, etc.