CPE 300L
DIGITAL SYSTEM ARCHITECTURE AND DESIGN LABORATORY

LABORATORY 4
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.

Behavioral models are typically used to model flip-flops in Verilog. An always procedural block with event control in the form of a sensitivity list is used to determine when the state of the flip-flop changes. If-else or case statements in the procedural block are then used to determine the next state. Latches can also be modeled in Verilog, either by a structural model based on gates or behaviorally as we do in the following with flip-flops. However, the use of latches is typically avoided when synthesizing Verilog for an FPGA as the timing is difficult to analyze.

Verilog keywords **`posedge`** and **`negedge`** are qualifiers which can be used to specify the active edge of a clock signal. An example of an always block which models a simple **`D`** flip-flop is shown is the following

```
always @(posedge Clock)
begin
    Q=D;
end
```

This always statement will update **`Q`** with the value of **`D`** on the rising edge of the clock signal.

A synchronous active low clear signal can be added to this flip-flop model to reset the flip-flop output to 0. This is done with an if-else statement in the always block as follows

```
always @(posedge Clock)
begin
    if (~ClearN)
        Q=1'b0;
    else
        Q=D;
end
```

Here **ClearN** is a control signal which is sampled on the rising edge of the clock. If **ClearN** is 0 the argument in the if-else is 1 corresponding to true and the output **Q** of the flip-flop is set to 0, otherwise the output is updated with the value of **D**.

An asynchronous clear signal could be used instead of a synchronous clear. In this case the always statement modeling the flip-flop is

```
always @(posedge Clock, negedge ClearN)
begin
   if (~ClearN)
       Q=1'b0;
   else
       Q=D;
end
```

Here the falling edge of **ClearN** initiates the evaluation of the always block. In contrast the synchronous clear does not initiate the evaluation, that is determined by the clock signal.

Any output wires updated inside the always statement has to be defined as a variable with the reg statement. Example: in #1 JK in prelab 3, **Q** is an output updated inside an **always** statement. Hence, **Q** is defined as a reg before the **always** statement begins.

LAB DELIVERIES

PRELAB

1. Implementation of SR latch: Structural Verilog Models

Analyze the Verilog code (Fig. 1.1) implementing the gated SR latch (Fig. 1).

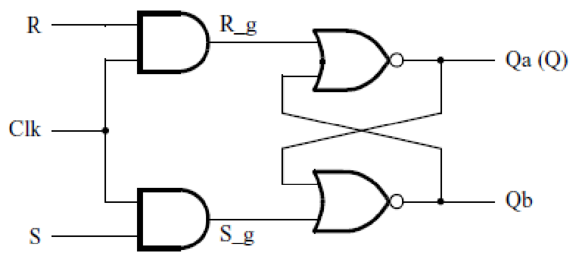


Figure 1. Gated SR latch

```verilog
// A gated RS latch
module part1 (Clk, R, S, Q);
    input Clk, R, S;
    output Q;
    wire R_g, S_g, Qa, Qb; /*
synthesis keep */
    assign R_g = R & Clk;
    assign S_g = S & Clk;
    assign Qa = ~(R_g | Qb);
    assign Qb = ~(S_g | Qa);
    assign Q = Qa;
endmodule
```

a) with component instantiation

```verilog
// A gated RS latch
module part1 (Clk, R, S, Q);
    input Clk, R, S;
    output Q;
    wire R_g, S_g, Qa, Qb; /*
synthesis keep */
    and (R_g, R, Clk);
    and (S_g, S, Clk);
    nor (Qa, R_g, Qb);
    nor (Qb, S_g, Qa);
    assign Q = Qa;
endmodule
```

b) structural model

Fig. 1.1. Verilog code for SR latch

Implement both a) and b) codes and get waveforms in Quartus. What is the difference between these two codes?

2. Asynchronous set/clear

Add asynchronous **Set′** and **Clear′** inputs to the design of Gated **D** latch, as shown in Fig. 2. Write the structural Verilog code.

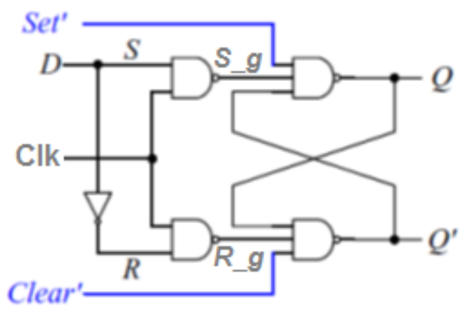


Fig. 2. Gated D latch with asynchronous inputs

Simulate the Verilog code for the design to get the waveforms. Verify correctness of the design in the waveforms in Quartus, make sure that all the cases are tested.

3. JK flip-flop construction

Fig. 3. shows the Truth Table and JK circuit based on D flip-flop

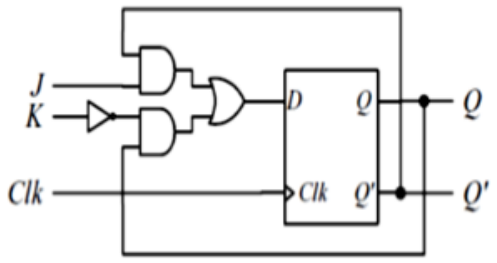


Fig. 3.1 JK flip-flop based on D

Q	Q_{next}	J	K
0	0	0	×
0	1	1	×
1	0	×	1
1	1	×	0

Fig. 3.2. Truth table of JK flip-flop

Fig. 3. JK flip-flop

Implement three different Verilog models of JK flip provided below and verify the operation. Get waveforms for all the models in Quartus. Check the logic utilization and component count in the simulation report (Total utilization, Combinational ALUTs, Dedicated logic registers).

```
//#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
```

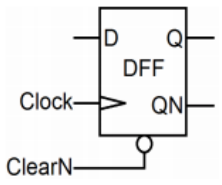
4. JK with synchronous clear
Modify JK flip-flops from prelab 3. to include synchronous ClearN. Write the verilog code and verify the operation in Quartus waveforms.

PRELAB DELIVERIES

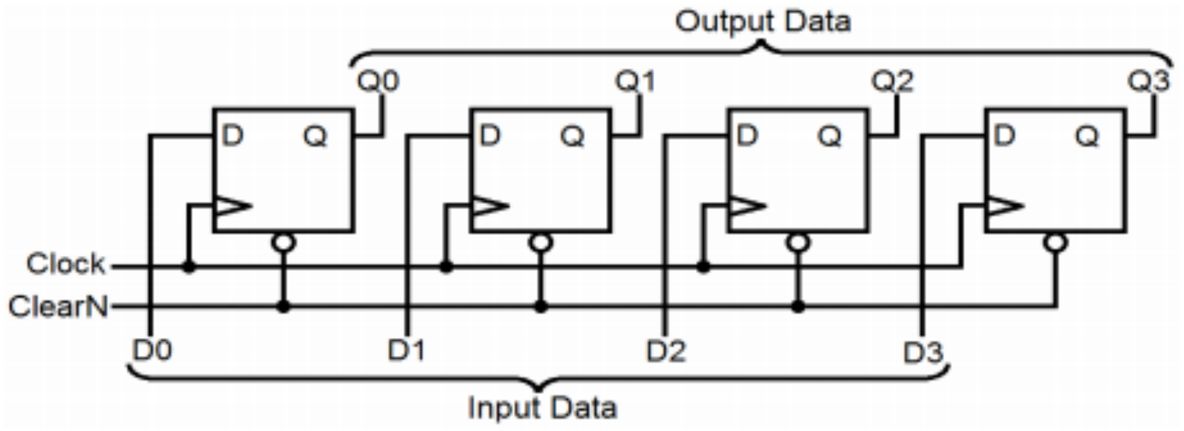
1. Waveform and answer to the question.
2. Verilog code and waveform.
3. Verilog code and waveform. Report Total utilization, Combinational ALUTs, Dedicated logic registers
4. Verilog code and waveform.

LAB EXPERIMENTS

1. Design a D flip flop with asynchronous low clear and complementary output as shown in block below. Verify the operation in Quartus waveform. Verify your design in RTL viewer.



2. A register consists of two or more flip-flops that share the clock and other control signals. A simple 4-bit register which uses D flip-flops is shown in figure below. This register will store the 4-bits of input data in the flip-flops on the rising edge of a clock signal. The output then appears as a 4-bit output. The register also has an asynchronous clear signal which will clear the 4-bit output to 0000. Write the Verilog code and test it using Quartus waveform. Verify your design in RTL viewer.



3. Write a Verilog Code for 4-bit mod-13 counter, simulate in ModelSim with the testbench and upload your design in DE2 board with the counter running on the 7 seg. Use the onboard clock to generate 1hz clock for the design. (use the clock divider if necessary). Verify your design in RTL viewer.

4. Design a 4-bit Pseudo Random Number Generator (PRNG) using a linear feedback shift register. You can refer to http://en.wikipedia.org/wiki/Linear_feedback_shift_register You can choose any generator polynomial (eg. x^4+x+1). Verify using Modelsim with a testbench. Also, the output of the PRNG should be displayed on a 7seg display in DE2 board. Your module should have clock, reset and output number. Use the onboard clock to generate 1hz clock for the design.

5. One of the applications of LFSR is to generate an automatic test pattern. Use the LFSR designed in experiment 4 on the ALU design from lab3 experiment4. Check the RTL generated and upload your design in DE2 board. Use different seed for input A and B, Cin is always 0. Find a way to pause your design to verify the result of the ALU.

POSTLAB REPORT:

Include the following elements in your postlab report:

Section	Element	
1	Theory of operation *Include a brief description of every element and phenomenon that appears during the experiments.* *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	Prelab report	
3	Results of the experiments	
	Experiment	**Experiment Results**
	1	a. Verilog code, waveform and RTL View
	2	a. Verilog code, waveform and RTL View
	3	a. Verilog code b. Testbench code c. Simulation waveforms and RTL View d. Picture of DE2 board
	4	a. Verilog code b. Testbench code c. Simulation waveforms d. Picture of DE2 board
	5	a. Verilog Code and RTL View b. Picture of DE2 board
4	Answer the questions	
	Question no.	**Question**
	1	What is the meaning of *component count* from the Quartus compilation report?
	2	What is the advantage of using asynchronous *Set* and *Reset* in case of D flip-flop? Give an example.
	3	Regarding *Logic Utilization* from the *Quartus Compilation Report* – what are *ALUT* and *Dedicated Logic Registers?*
5	Conclusions *Write down your conclusions, things learned, problems encountered during the lab and how they were solved, etc.*	