Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan, Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 loc. 2121 / 2221
E-mail Address: ceafa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Engineering – Department of Electrical Engineering

COMPUTER ENGINEERING PROGRAM
INTRODUCTION TO HDL
LABORATORY REPORT 4
FLIP-FLOPS

Name: Marfa, Jonard T. **Section: CpE - 3101**
Sr-Code: 23-03651

1. **Activity's Intended Learning Outcomes (ILOs)**
   At the end of this activity, the student shall be able to:
   a. **Implement and simulate basic flip-flops (D, SR, JK) in Verilog** to observe their state changes under clock pulses.
   b. **Analyze and verify flip-flop behavior** using truth tables and timing diagrams, ensuring outputs match theoretical expectations.

2. **Objectives of the Activity**
   The activity aims to:
   - **Introduce** students to sequential storage elements as the foundation of counters and state machines.
   - **Demonstrate** how different flip-flops respond to inputs and clock edges.
   - **Guide** students in writing Verilog modules and testbenches for flip-flops.
   - **Develop** student skills in interpreting simulation waveforms and connecting them to truth tables.

Sample Verilog Code:

FOR DESIGN

```
// Flip-Flop
module <flipflop_name> (
  input clk,
  input rst,
  // add inputs here (e.g., D, S, R, J, K)
  output reg Q
);
  always @(posedge clk or posedge rst) begin
    if (rst)
      Q <= 0;       // reset output
    else begin
      // write the behavior of your chosen flip-flop here
      // Example for D FF: Q <= D;
      // Example for SR FF: case({S,R}) ...
      // Example for JK FF: case({J,K}) ...
    end
  end
endmodule
```

College of Engineering – Department of Electrical Engineering

FOR TEST BENCH

```
module tb_<flipflop_name>;
 reg clk, rst;
 // declare inputs here (e.g., D, S, R, J, K)
 wire Q;

 // Instantiate the flip-flop
 <flipflop_name> uut (
  .clk(clk),
  .rst(rst),
  // connect inputs here
  .Q(Q)
 );

 // Clock generation
 initial begin
  clk = 0;
  forever #5 clk = ~clk; // 10 time units period
 end

 // Stimulus
 initial begin
  $dumpfile("<flipflop_name>.vcd"); $dumpvars;
  $display("Time | clk rst | Inputs | Q");

  // Reset
  rst = 1;
  // set inputs to 0
  #10 rst = 0;

  // Apply test cases here
  // Example: D=1; #10; D=0; #10;
  // Example: S=1; R=0; #10; S=0; R=1; #10;
  // Example: J=1; K=1; #20;

  $finish;
 end
endmodule
```

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan, Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 loc. 2121 / 2221
E-mail Address: ceafa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Engineering – Department of Electrical Engineering

----------*ASSESSMENT STARTS HERE*----------

ASSESSMENT 1: **D FLIP-FLOP**

Write the Verilog module for a D flip-flop. Simulate its behavior using a testbench and verify the output against the truth table.

A. Complete the truth tables:

Note: This is based on the problem, not from the Verilog. This table will be used for checking if your code works. This will also show if you really understand the problem.

TRUTH TABLE			
Input	Clock	Output	
D	clk	Q (previous state)	Qn (next state)
0	1	0	0
0	1	1	0
1	1	0	1
1	1	1	1

EXCITATION TABLE		
Input		Output
Q	Qn	D
0	0	0
0	1	1
1	0	0
1	1	1

B. Verilog Code *(paste the actual code):*

Design Code:

```
module marfa_D_FF(input D, input clk, output reg Q);
  initial Q = 0;
  always @(posedge clk)
  Q <= D;
endmodule
```

College of Engineering – Department of Electrical Engineering

Testbench Code:

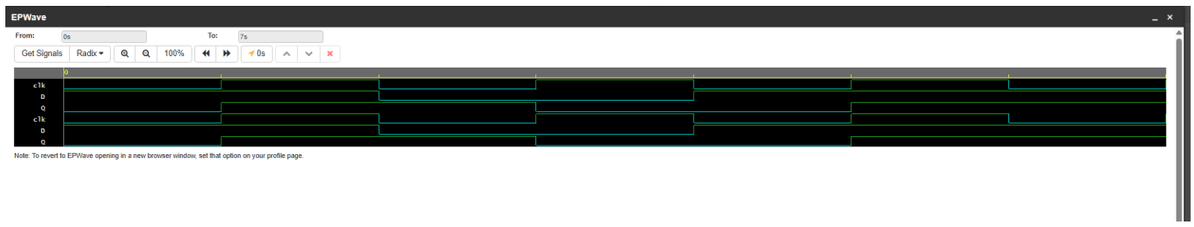
```
module Testbench_D;
 reg D, clk;
 wire Q;

 marfa_D_FF uut (.D(D), .clk(clk), .Q(Q));
 initial begin
 $dumpfile("dff.vcd");
 $dumpvars(0, Testbench_D);
  $display("Time |  Edge  | D |  Q  | Clock");
  $display("-----+----------+---+-----+-----");

 D = 0; clk = 0;
 #0 D = 1;
 #2 D = 0;
 #2 D = 1;
 #3 $finish;
 end
 always #1 clk = ~clk;

 always @(posedge clk) begin
  $strobe("%4t | (+) edge | %b |  %b  |  %b", $time, D, Q, clk);
 end

 always @(negedge clk) begin
  $strobe("%4t | (-) edge | %b |  %b  |  %b", $time, D, Q, clk);
 end
endmodule
```

C. Output Log (the output of your verilog code)

Log | Share

```
[2025-11-30 14:43:25 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dff.vcd opened for output.
Time |   Edge   | D |  Q  | Clock
-----+----------+---+-----+-----
   1 | (+) edge | 1 |  1  |  1
   2 | (-) edge | 0 |  1  |  0
   3 | (+) edge | 0 |  0  |  1
   4 | (-) edge | 1 |  0  |  0
   5 | (+) edge | 1 |  1  |  1
   6 | (-) edge | 1 |  1  |  0
testbench.sv:16: $finish called at 7 (1s)
   7 | (+) edge | 1 |  1  |  1
Finding VCD file...
./dff.vcd
[2025-11-30 14:43:26 UTC] Opening EPWave...
Done
```

D. EPWave



E. **Activity Conclusion/Takeaways** *(Discuss what happened to the circuit and the EPWave, and your conclusions)*

In this D Flip-Flop activity, I constructed the Truth Table by focusing specifically on the active clock state (Logic 1 or the rising edge) to ensure that the Next State (Qn+1) would directly reflect the input changes. I chose this approach because including the inactive clock states (Logic 0) results in "Hold" conditions that make deriving the Excitation Table unnecessarily complex. By utilizing a consistent active clock in the table, I could clearly observe that the required input D equates directly to the desired Next State. Although the Verilog simulation and EPWaveform display an alternating clock signal, which visually differs from the static inputs in my table, the underlying logic remains consistent. The simulation simply illustrates the dynamic behavior of the circuit, showing that it preserves the previous state when the clock is low and updates only during the active edge. Therefore, the EPWaveform serves as a reliable confirmation of my tabulated data.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan, Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 loc. 2121 / 2221
E-mail Address: ceafa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Engineering – Department of Electrical Engineering

ASSESSMENT 2: SR FLIP-FLOP

Write the Verilog module for an SR flip-flop. Simulate its behavior using a testbench and verify the output against the truth table.

A. Complete the truth tables:

Note: This is based on the problem, not from the Verilog. This table will be used for checking if your code works. This will also show if you really understand the problem.

TRUTH TABLE				
Input		Clock	Output	
S	R	clk	Q (previous state)	Qn (next state)
0	0	1	0	0
0	0	1	1	1
0	1	1	0	0
0	1	1	1	0
1	0	1	0	1
1	0	1	1	1
1	1	1	0	x
1	1	1	1	x

EXCITATION TABLE			
Input		Output	
Q	Qn	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

B. Verilog Code *(paste the actual code):*

Design Code:

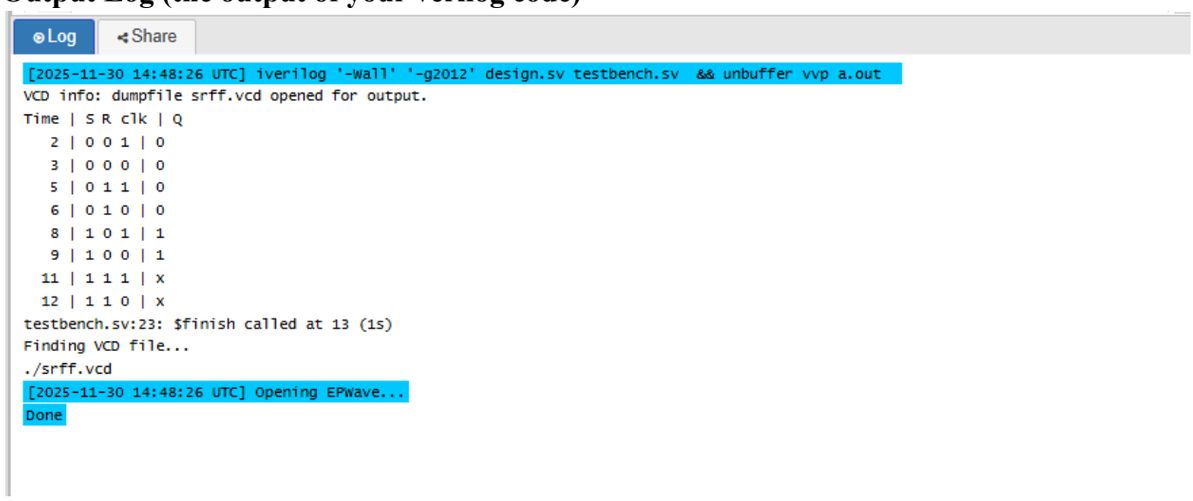
```
module marfa_SR_FLIPFLOP (
  input S, R,
  input clk,
  output reg Q
);
```

```
  initial Q = 0;
  always @(posedge clk) begin
  case ({S, R})
  2'b00: Q <= Q;
  2'b01: Q <= 0;
  2'b10: Q <= 1;
  2'b11: Q <= 1'bx;
  endcase
  end
endmodule
```

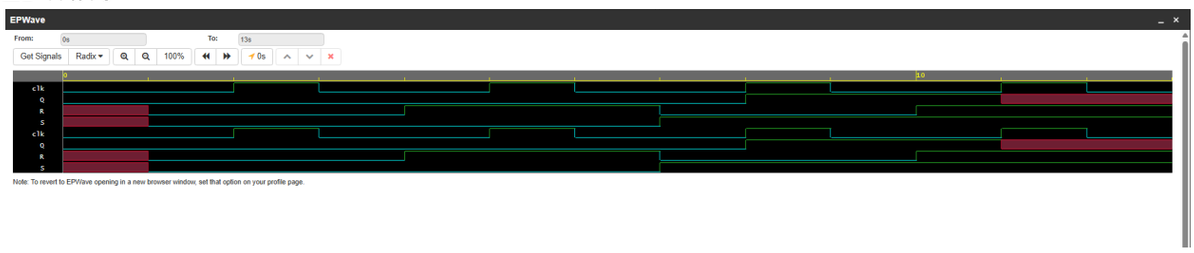
Testbench Code:

```
module Testbench_SR;
  reg S, R, clk;
  wire Q;

  marfa_SR_FLIPFLOP uut (
  .S(S),
  .R(R),
  .clk(clk),
  .Q(Q)
  );

  initial begin
  $dumpfile("srff.vcd");
  $dumpvars(0, Testbench_SR);
  $display("Time | S R clk | Q");

  clk = 0;

  #1 S = 0; R = 0; #1 clk = 1; #1 clk = 0;
  #1 S = 0; R = 1; #1 clk = 1; #1 clk = 0;
  #1 S = 1; R = 0; #1 clk = 1; #1 clk = 0;
  #1 S = 1; R = 1; #1 clk = 1; #1 clk = 0;
  #1 $finish;
  end

  always @(posedge clk or negedge clk)
  $strobe("%4t | %b %b %b | %b", $time, S, R, clk, Q);

endmodule
```

C. Output Log (the output of your verilog code)



D. EPWave



E. **Activity Conclusion/Takeaways** *(Discuss what happened to the circuit and the EPWave, and your conclusions)*

In this SR Flip-Flop activity, I applied the same analytical approach as the previous experiment by constructing the Truth Table with the Clock consistently set to Logic 1. This method isolates the active state behavior, allowing for a clear and direct derivation of the Next State (Qn+1) based on the Set and Reset inputs without the redundancy of the inactive "Hold" states found in the dynamic simulation. A critical observation in this specific circuit, which distinguishes it from the D Flip-Flop is the occurrence of the invalid state when both inputs are High or 1. My Truth Table theoretically identifies this as an indeterminate state X, and this is precisely validated by the simulation results, the Verilog log reports an X output, and the EPWaveform displays a distinct red block indicating an undefined value during this period. Furthermore, focusing on the active transitions allowed me to accurately populate the Excitation Table, correctly identifying the "Don't Care" conditions (marked as X) where the specific value of set and reset does not affect the outcome.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan, Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 loc. 2121 / 2221
E-mail Address: ceafa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Engineering – Department of Electrical Engineering

ASSESSMENT 3: **JK FLIP-FLOP**

Write the Verilog module for an JK flip-flop. Simulate its behavior using a testbench and verify the output against the truth table.

A. Complete the truth tables:

Note: This is based on the problem, not from the Verilog. This table will be used for checking if your code works. This will also show if you really understand the problem.

TRUTH TABLE				
Input		Clock	Output	
J	K	clk	Q (previous state)	Qn (next state)
0	0	1	0	0
0	0	1	1	1
0	1	1	0	0
0	1	1	1	0
1	0	1	0	1
1	0	1	1	1
1	1	1	0	1
1	1	1	1	0

EXCITATION TABLE			
Input		Output	
Q	Qn	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

B. Verilog Code *(paste the actual code)*:
Design Code:

```
module marfa_JK_FF (
  input J, K, clk,
  output reg Q
  );
```

College of Engineering – Department of Electrical Engineering

```
 initial Q = 0;

 always @(posedge clk) begin
 case ({J, K})
        2'b00: Q <= Q;
        2'b01: Q <= 0;
        2'b10: Q <= 1;
        2'b11: Q <= ~Q;
 endcase
 end
endmodule
```

Testbench Code:

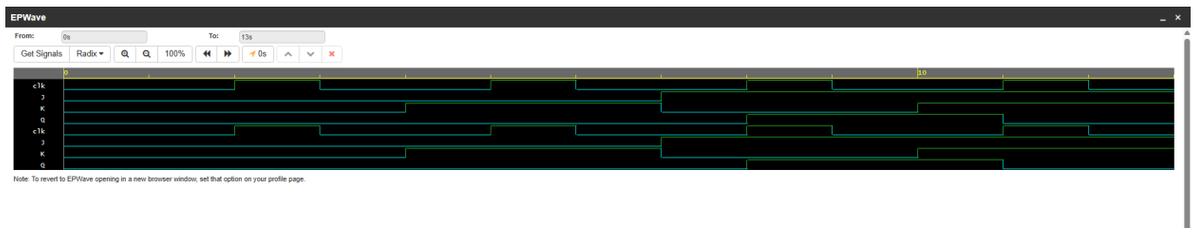
```
module Testbench_JK;
 reg J, K, clk;
 wire Q;

 marfa_JK_FF uut (
  .J(J),
  .K(K),
  .clk(clk),
  .Q(Q)
 );

 initial begin
  $dumpfile("jkff.vcd");
  $dumpvars(0, Testbench_JK);
  $display("Time | J K clk | Q");

  J = 0; K = 0; clk = 0;

  #1 J = 0; K = 0; #1 clk = 1; #1 clk = 0;
  #1 J = 0; K = 1; #1 clk = 1; #1 clk = 0;
  #1 J = 1; K = 0; #1 clk = 1; #1 clk = 0;
  #1 J = 1; K = 1; #1 clk = 1; #1 clk = 0;
  #1 $finish;
 end

 always @(posedge clk or negedge clk)
 $strobe("%4t | %b %b %b | %b", $time, J, K, clk, Q);

endmodule
```

C. Output Log (the output of your verilog code)

Log | Share

```
[2025-11-30 14:52:19 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile jkff.vcd opened for output.
Time | J K clk | Q
   2 | 0 0 1 | 0
   3 | 0 0 0 | 0
   5 | 0 1 1 | 0
   6 | 0 1 0 | 0
   8 | 1 0 1 | 1
   9 | 1 0 0 | 1
  11 | 1 1 1 | 0
  12 | 1 1 0 | 0
testbench.sv:23: $finish called at 13 (1s)
Finding VCD file...
./jkff.vcd
[2025-11-30 14:52:20 UTC] Opening EPWave...
Done
```

D. EPWave



E. **Activity Conclusion/Takeaways** *(Discuss what happened to the circuit and the EPWave, and your conclusions)*

In this JK Flip-Flop activity, I continued with the established methodology of constructing the Truth Table using a consistent Logic 1 for the clock. This approach allowed me to focus purely on the state transitions without the interference of the "Hold" states present in the dynamic simulation. The most significant observation in this experiment distinguishing the JK from the SR Flip-Flop is the behavior when both inputs are High. Unlike the SR circuit which enters an invalid state, my Truth Table indicates a "Toggle" mode where the output inverts (switches from 0 to 1, or 1 to 0). This theoretical behavior is perfectly supported by the simulation results, specifically the Verilog log at time unit 11 shows that when J=1 and K=1, the output Q successfully toggles from its previous high state to a low state, rather than becoming undefined. Consequently, I was able to derive a complete Excitation Table without any invalid output states, while still utilizing "Don't Care" conditions on the input side to simplify the design logic. The EPWaveform visually confirms this stable operation, verifying that the JK Flip-Flop resolves the indeterminate condition of the SR latch by correctly toggling on the active clock edge.

Republic of the Philippines
BATANGAS STATE UNIVERSITY
The National Engineering University
Alangilan Campus
Golden Country Homes, Alangilan, Batangas City, Batangas, Philippines 4200
Tel Nos.: (+63 43) 425-0139 loc. 2121 / 2221
E-mail Address: ceafa@g.batstate-u.edu.ph | Website Address: http://www.batstate-u.edu.ph

College of Engineering – Department of Electrical Engineering

CRITERIA	SCORE
Code	
The code is completely functional and responds correctly producing the correct outputs. **(5 Points)**	
The code is correct regarding syntax but required output is incorrect. **(3 Points)**	
The code has several syntax errors. **(1 Points)**	
Truth Table Construction	
Truth table is complete and accurate, showing correct mapping from inputs to Sum, Carry, and decoder output. **(5 Points)**	
Truth table is mostly correct but contains minor errors or omissions. **(3 Points)**	
Truth table is incomplete or shows poor understanding of input-output relationships. **(1 points)**	
Overall Work	
Student's work shows complete understanding of problem and all requirements. Able to explain entire program and design correctly as it is. **(10 Points)**	
Student's work shows understanding of problem and most requirements. Able to explain some program and design. **(7 Points)**	
Student's work shows slight understanding of problem and requirements. Unable to explain program and design. **(5 Points)**	
TOTAL SCORE:	
Percentage Score= (Total Score/20) * 100	
Other Comments:	