College of Engineering – Department of Electrical Engineering

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

Name: Gamboa, Jave Danielle M. **Section:** CpE - 3101
Sr-Code: 23-03647

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
```

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
```

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)**
1	1	0	1
1	1	1	1
0	1	0	0
0	1	1	0

When the clk is 0, Qn = Q

Excitation Table		
Input		**Output**
Q	**Qn**	**D**
0	0	0
0	1	1
1	0	0
1	1	1

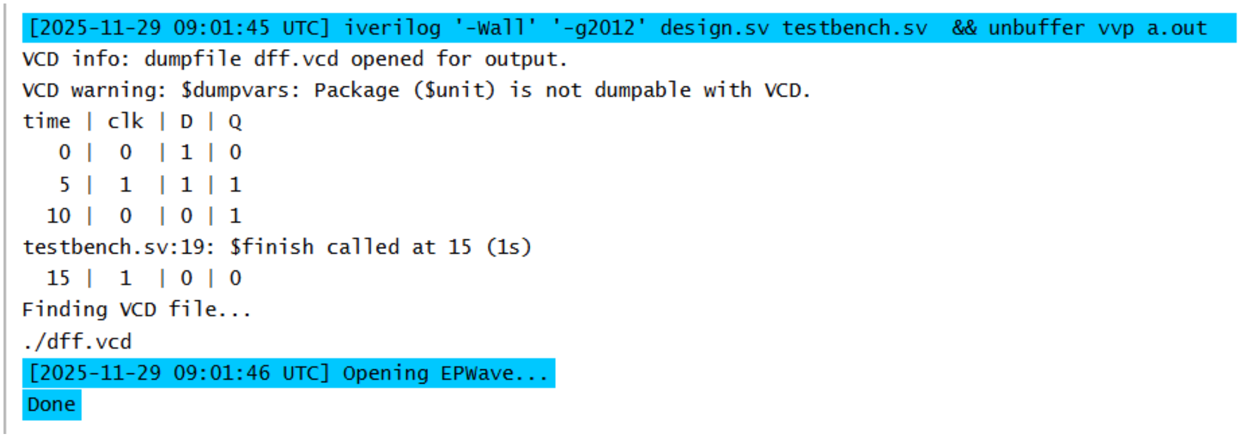
B. **Verilog Code** *(paste the actual code):*
Design Code:

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

Testbench Code:

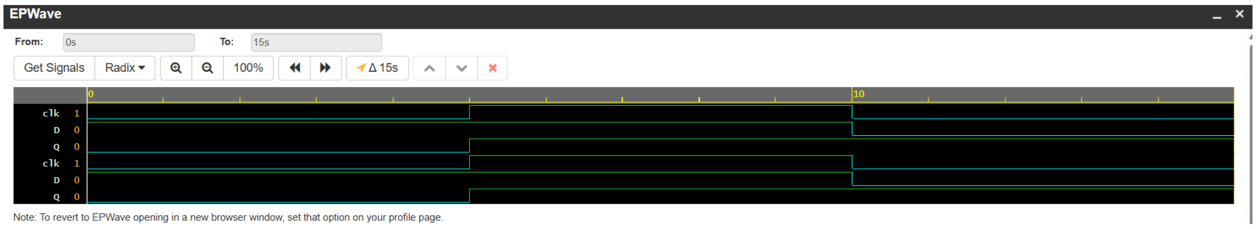
```
module Testbench_dff;
  reg D, clk;
  wire Q;

  d_flipflop uut(.D(D), .clk(clk), .Q(Q));

  initial begin
    clk = 0;
    forever #5 clk = ~clk; // 10 time unit period
  end

  initial begin
    $dumpfile("dff.vcd");
    $dumpvars;
    $display("time | clk | D | Q");

    D = 1;  #10;
    D = 0;  #5;
    $finish;
  end

  always @(clk)
    $strobe("%4t |  %b  | %b | %b", $time, clk, D, Q);

endmodule
```

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



```
[2025-11-29 09:01:45 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dff.vcd opened for output.
VCD warning: $dumpvars: Package ($unit) is not dumpable with VCD.
time | clk | D | Q
   0 |  0  | 1 | 0
   5 |  1  | 1 | 1
  10 |  0  | 0 | 1
testbench.sv:19: $finish called at 15 (1s)
  15 |  1  | 0 | 0
Finding VCD file...
./dff.vcd
[2025-11-29 09:01:46 UTC] Opening EPWave...
Done
```

D. EPWave



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.

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

The way a D Flipflop works is that when the clock is at high edge or 1, it will get the value of D and assign it to Q. Now we can see in the output Q updates and gets the value of D when the clock is at 1. The output corresponds with the truth table.

In the EPWave, we can see that D and Q are the same when the clk is at high meaning it holds the last value of Q when the clk becomes 0. I also learned here that we should use $strobe and not $display because when we use display, it will immediately get the value of D and Q even if Q has not yet updated and will cause it to display a wrong output. Lastly, in the excitation table the value of D will depend on what the value of Qn is. In conclusion, this experiment successfully demonstrates the edge-triggered behavior of a D flip-flop, where Q reliably captures and holds the value of D at each clock's rising edge.

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

When the clk is 0, Qn = Q

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

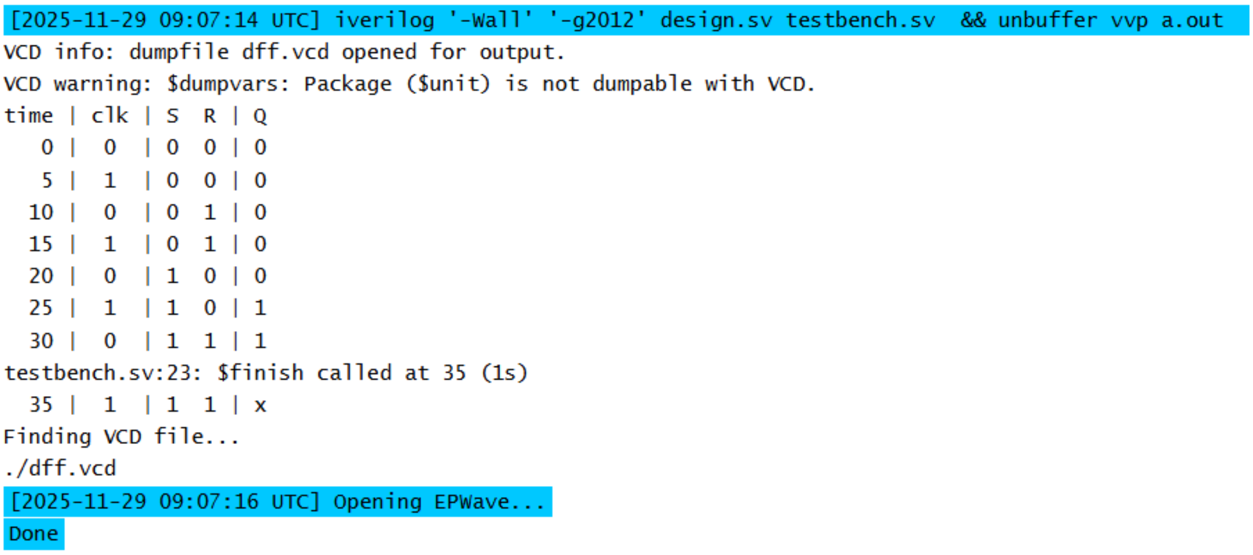
B. Verilog Code *(paste the actual code):*
Design Code:

```
module senior_flipflop (
    input S, R,
    input clk,
    output reg Q
);
initial Q=0;

always @(posedge clk) begin
    case ({S, R})
        2'b00: Q <= Q;    // No change
        2'b01: Q <= 0;    // Reset
        2'b10: Q <= 1;    // Set
        2'b11: Q <= 1'bx; // Invalid condition
    endcase
end
endmodule
```

Testbench Code:

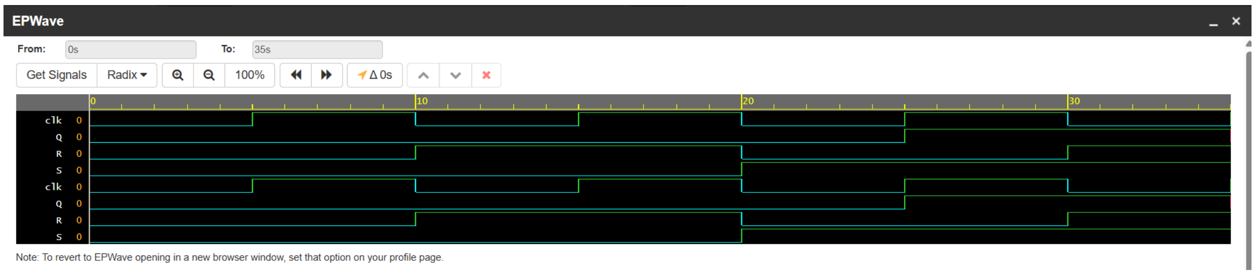
```
module Testbench_srff;
  reg S, R , clk;
  wire Q;

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

  initial begin
    clk = 0;
    forever #5 clk = ~clk; // 10 time unit period
  end

  initial begin
    $dumpfile("dff.vcd");
    $dumpvars;
    $display("time | clk | S  R | Q");

    S = 0; R = 0;

    #10 S = 0; R = 1;
    #10 S = 1; R = 0;
    #10 S = 1; R = 1;
    #5;
    $finish;
  end

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

endmodule
```

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

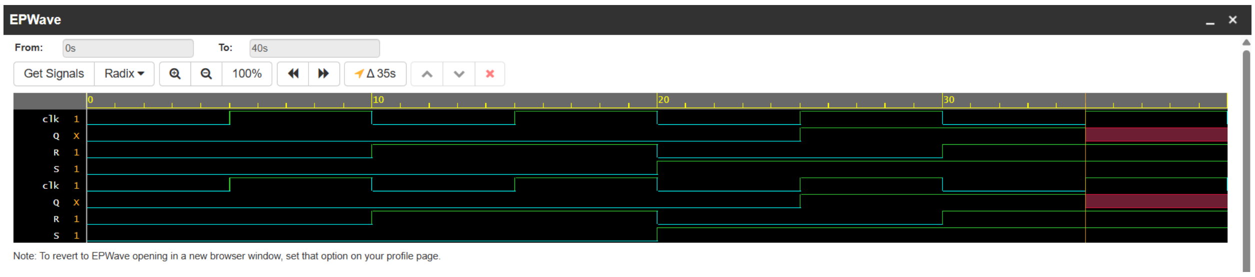


```
[2025-11-29 09:07:14 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dff.vcd opened for output.
VCD warning: $dumpvars: Package ($unit) is not dumpable with VCD.
time | clk | S  R | Q
   0 |  0  | 0  0 | 0
   5 |  1  | 0  0 | 0
  10 |  0  | 0  1 | 0
  15 |  1  | 0  1 | 0
  20 |  0  | 1  0 | 0
  25 |  1  | 1  0 | 1
  30 |  0  | 1  1 | 1
testbench.sv:23: $finish called at 35 (1s)
  35 |  1  | 1  1 | x
Finding VCD file...
./dff.vcd
[2025-11-29 09:07:16 UTC] Opening EPWave...
Done
```

D. EPWave



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.

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

In an SR Flipflop, when the value of both S and R is 0 it will hold that last value of Q. When the value of S is 0 and R is 1, it will reset the value of Q to 0. When the value of S is 1 and R is 0, it will set the value of Q to 1. Lastly, when both S and R have a value of 1, it will give an invalid output for Q.

We can also see in the output (Qn) of S=1 and R=1 that it gives an invalid output denoted by x. In the second EPWave, I added another instance of S = 1; R = 1; clk = 1; to show that it will highlight red when it is invalid rather than having a high(green) or low(blue) line. Lastly, in the excitation table the x denotes don't care meaning it can either be 0 or 1. In this part, we will look at how the Q changed to Qn. If it changed from 0 to 1 it used the set inputs of S = 1 and R = 0 and when it changed from 1 to 0 it used the reset inputs of S = 0 and R = 1. When Q = 0 and Qn = 0, it either held the last value(00) or reset it to 0(01) meaning a 0 in S and a don't care in R. When the Q = 1 and Qn = 1, it either set it to 1(10) or held the last value(00) meaning a don't care in S and a 0 in Q.

Overall, the SR flip-flop operates by using its S (set) and R (reset) inputs to control the state of Q. It correctly performs the hold, set, and reset functions depending on the input values, but the combination S = 1 and R = 1 leads to an invalid or undefined output, which is why the truth table shows x for that case and why EPWave highlights it in red. Since Q is initialized to 0 in the code, this becomes the first stored value used in the waveform. The excitation table also confirms how the flip-flop transitions between states: transitions from 0 to 1 require the set input (10), transitions from 1 to 0 require the reset input (01), and cases where Q remains the same correspond to hold or don’t-care input conditions.

College of Engineering – Department of Electrical Engineering

ASSESSMENT 3: JK FLIP-FLOP

Write the Verilog module for a 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

When the clk is 0, Qn = Q

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

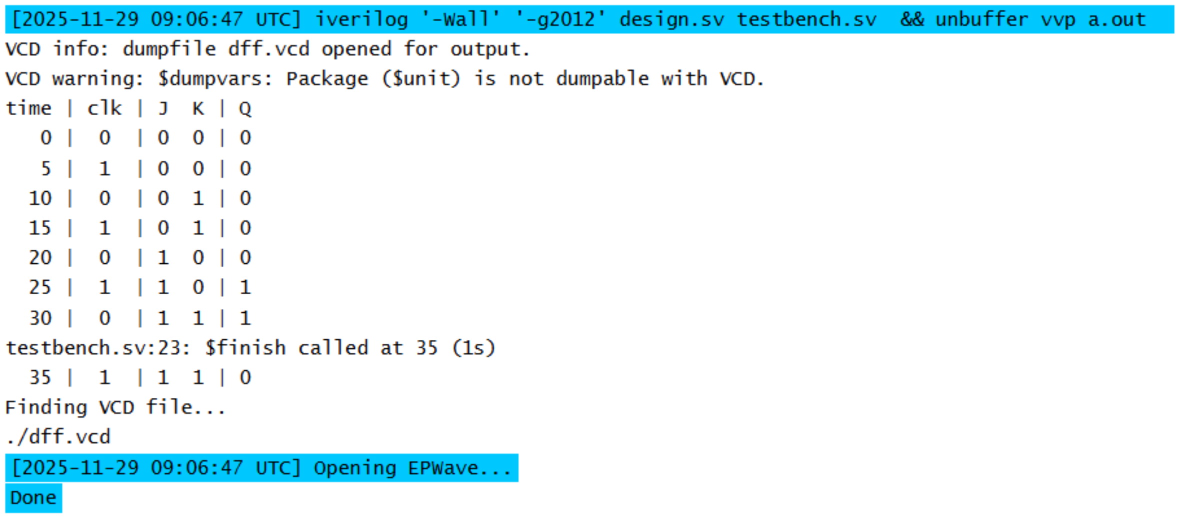
B. Verilog Code *(paste the actual code):*
Design Code:

```
module joke_flipflop (
    input J, K,
    input clk,
  output reg Q
);
initial Q=0;

always @(posedge clk) begin
 case ({J, K})
     2'b00: Q <= Q;             // No change
     2'b01: Q <= 0;    // Reset
     2'b10: Q <= 1;    // Set
     2'b11: Q <= ~Q;  // Invert
  endcase
end
endmodule
```

Testbench Code:

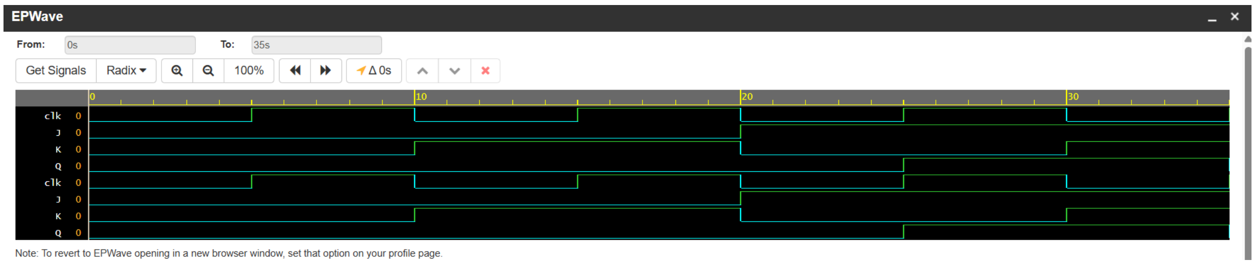
```
module Testbench_jkff;
 reg J, K , clk;
 wire Q;

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

 initial begin
  clk = 0;
  forever #5 clk = ~clk; // 10 time unit period
 end

 initial begin
  $dumpfile("dff.vcd");
  $dumpvars;
  $display("time | clk | J  K | Q");

  J = 0; K = 0;

  #10 J = 0; K = 1;
  #10 J = 1; K = 0;
  #10 J = 1; K = 1;
  #5;
  $finish;
 end

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

endmodule
```

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



```
[2025-11-29 09:06:47 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
VCD info: dumpfile dff.vcd opened for output.
VCD warning: $dumpvars: Package ($unit) is not dumpable with VCD.
time | clk | J  K | Q
   0 |  0  | 0  0 | 0
   5 |  1  | 0  0 | 0
  10 |  0  | 0  1 | 0
  15 |  1  | 0  1 | 0
  20 |  0  | 1  0 | 0
  25 |  1  | 1  0 | 1
  30 |  0  | 1  1 | 1
testbench.sv:23: $finish called at 35 (1s)
  35 |  1  | 1  1 | 0
Finding VCD file...
./dff.vcd
[2025-11-29 09:06:47 UTC] Opening EPWave...
Done
```

D. EPWave



Note: To revert to EPWave opening in a new browser window, set that option on your profile page.

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

JK Flipflop and SR Flipflop are almost the same, the only difference will be the inputs being J and K rather than S and R, and when the inputs are both 1. In a JK Flipflop, when the value of both J and K are 0 it will hold that last value of Q. When the value of J is 0 and K is 1, it will reset the value of Q to 0. When the value of J is 1 and K is 0, it will set the value of Q to 1. Lastly, when both J and K have a value of 1, it will invert the last value of Q.

In the excitation table, when it changed from 0 to 1 it used the set inputs of J = 1 and don't care in K and when it changed from 1 to 0 it used the reset inputs of don't care in J and K = 1. When Q = 0 and Qn = 0, it either held the last value(00) or reset it to 0(01) meaning a 0 in J and a don't care in K. When the Q = 1 and Qn = 1, it either set it to 1(10) or held the last value(00) meaning a don't care in J and a 0 in R.

The JK flip-flop functions very similarly to the SR flip-flop, with the key difference occurring when both inputs are 1. For JK, the input combination J = 1 and K = 1 causes the output to toggle, meaning it inverts the previous value of Q. Just like other edge-triggered flip-flops, the JK stores and updates its output based on the previous state, which we initialized as Q = 0 in the code.

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:	