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 5
COUNTERS

Name: Bughao, Andrei Dennise R. **Section:** CpE-3101
Sr-Code: 23-09845

1. **Activity's Intended Learning Outcomes (ILOs)**
   At the end of this activity, the student shall be able to:
   a. **Implement and simulate** a 3-bit UP asynchronous counter and a 3-bit DOWN synchronous counter in Verilog to observe their counting sequences under clock pulses.
   b. **Analyze and verify** counter behavior using output logs and timing diagrams, highlighting differences between asynchronous and synchronous operation.
2. **Objectives of the Activity**
   The activity aims to:
   a. To reinforce understanding of **sequential circuits** and how counters are realized using flip-flops.
   b. To practice **Verilog coding** for both asynchronous and synchronous counter designs.
   c. To use a **testbench and simulation tools** to observe counting sequences and wrap-around behavior.
   d. To analyze differences in timing between asynchronous and synchronous counters.

Sample Verilog Code:

FOR DESIGN

```
module ff_ff (
  input clk,
  input rst,
  input A,
  input B,
  output reg X
);
  always @(posedge clk or posedge rst) begin
     case ({A, B})
       //Conditions
     endcase
   end
  end
endmodule
```

College of Engineering – Department of Electrical Engineering

```
module counter_des (
  input clk,
  input rst,
  output [3:0] X
);
  //Conditions
endmodule
```

FOR TEST BENCH

```
module tb_counter;
  reg clk, rst;
  wire [3:0] X;

  counter_des uut (.clk(clk), .rst(rst), .Q(Q));

  initial begin
    $dumpfile("counter.vcd"); $dumpvars;
    $display("table");

//rst

    // Generate clock pulses
//Displaying
      end
    end

    $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: ***3-BIT UP ASYNCHRONOUS COUNTER***

A **3-bit UP asynchronous counter** that increments its count on each clock pulse, with outputs changing in a ripple fashion.

A. CIRCUIT DIAGRAM:

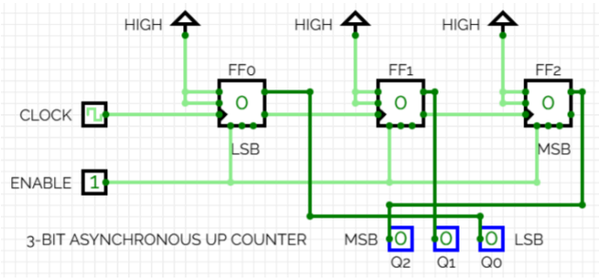


Figure 1.1 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 000

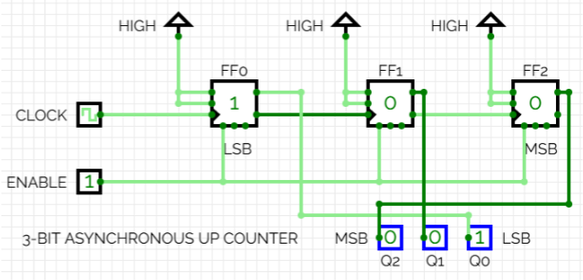


Figure 1.2 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 001

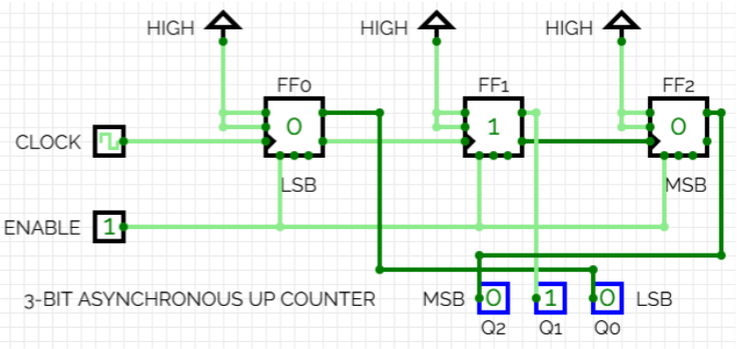


Figure 1.3 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 010

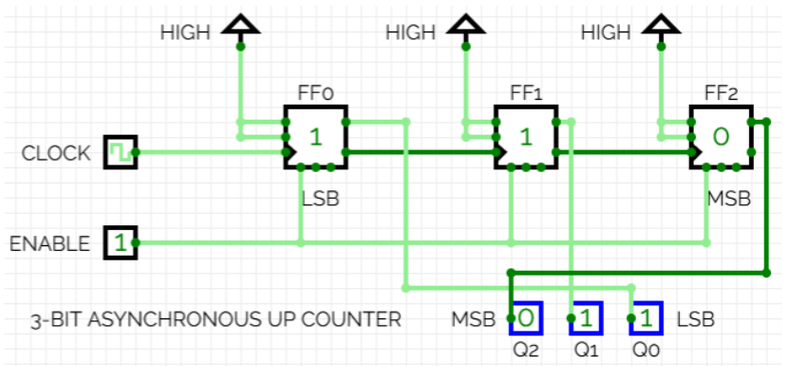


Figure 1.4 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 011

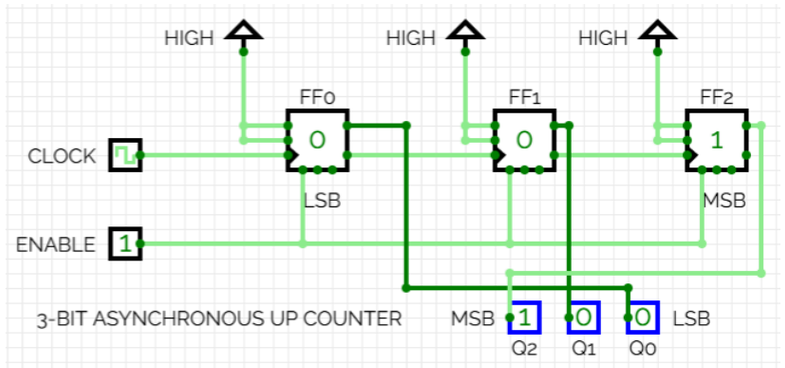


Figure 1.5 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 100

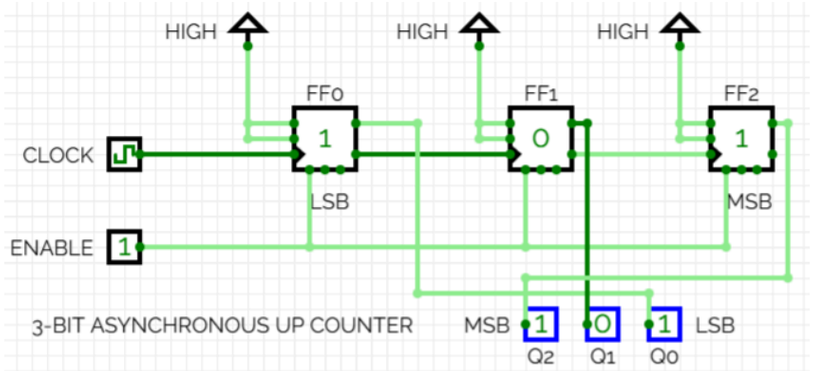


Figure 1.6 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 101

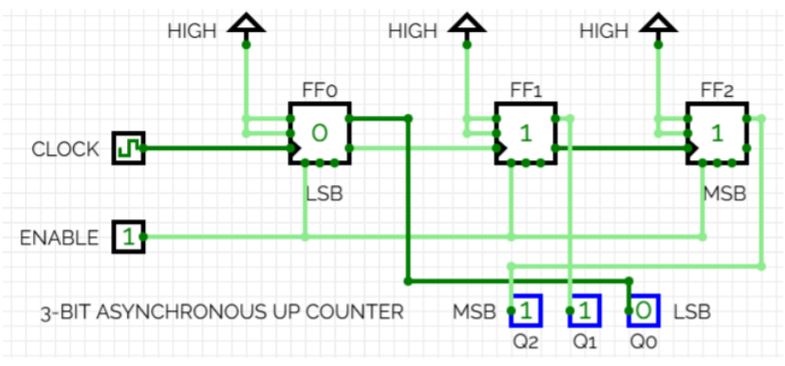


Figure 1.7 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 110

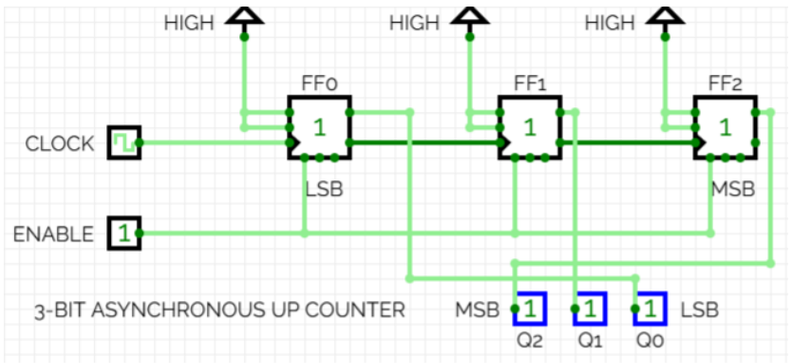


Figure 1.8 – 3-bit Asynchronous Up Counter: Q2 Q1 Q0 = 111

B. VERILOG CODE *(paste the actual code):*

Design Code:

```
// JK Flip-Flop module
module jk_ff(
    input clk,
    input rst,
    input J,
    input K,
    output reg Q
);
    always @(posedge clk or posedge rst) begin
        if (rst)
            Q <= 0;
        else begin
            case ({J, K})
                2'b00: Q <= Q;      // No change
                2'b01: Q <= 0;      // Reset
                2'b10: Q <= 1;      // Set
                2'b11: Q <= ~Q;     // Toggle
            endcase
        end
    end
endmodule

// 3-bit Asynchronous Up Counter
module async_up_counter_3bit(
    input clk,
    input rst,
    output [2:0] Q
```

```
);
   // JK inputs tied to 1 for toggle mode
   jk_ff jk0 (.clk(clk),      .rst(rst), .J(1'b1), .K(1'b1), .Q(Q[0]));
   jk_ff jk1 (.clk(~Q[0]),    .rst(rst), .J(1'b1), .K(1'b1), .Q(Q[1]));
   jk_ff jk2 (.clk(~Q[1]),    .rst(rst), .J(1'b1), .K(1'b1), .Q(Q[2]));
endmodule
```

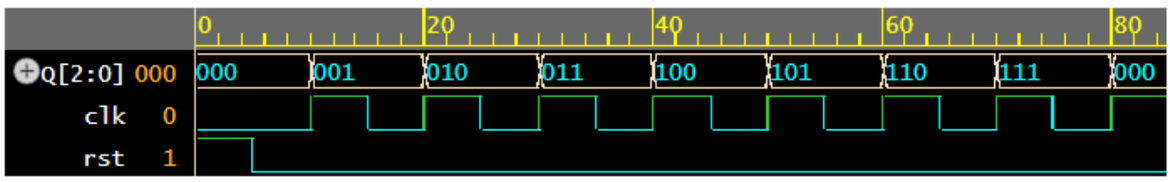
Testbench Code:

```
// Code your testbench here
// or browse Examples
// Testbench for 3-bit Asynchronous Up Counter
module tb_async_up_counter_3bit;
   reg clk, rst;
   wire [2:0] Q;

   async_up_counter_3bit uut (
      .clk(clk),
      .rst(rst),
      .Q(Q)
   );

   initial begin
      $dumpfile("counter_jk_3bit.vcd");
      $dumpvars(0, tb_async_up_counter_3bit);
      $display("Time | clk rst | Q2 Q1 Q0");
      $display("-------------------------");

      clk = 0;
      rst = 1;
      #5 rst = 0; // release reset

      // Generate clock pulses
      repeat (16) begin
         #5 clk = ~clk; // toggle clock
         if (clk) begin // display only at positive edge
            $display("%4t | %b %b | %b %b %b", $time, clk, rst, Q[2], Q[1], Q[0]);
         end
      end

      $finish;
   end
endmodule
```

C. OUTPUT LOG (the output of your verilog code)

```
[2025-12-03 11:22:12 UTC] iverilog
VCD info: dumpfile counter_jk_3bit.
Time | clk rst | Q2 Q1 Q0
-------------------------
  10 | 1 0 | 0 0 0
  20 | 1 0 | 0 0 1
  30 | 1 0 | 0 1 0
  40 | 1 0 | 0 1 1
  50 | 1 0 | 1 0 0
  60 | 1 0 | 1 0 1
  70 | 1 0 | 1 1 0
  80 | 1 0 | 1 1 1
testbench.sv:32: $finish called at
Finding VCD file...
./counter_jk_3bit.vcd
[2025-12-03 11:22:13 UTC] Opening
Done
```

D. EPWave



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

For this experiment, I built a 3-bit asynchronous up counter using JK flip-flops. In an asynchronous counter, only the first flip-flop receives the clock directly, and each of the next flip-flops is triggered by the output of the previous one. This setup makes the counter increase its value with every clock pulse, counting from 000 to 111. I first created the circuit in CircuitVerse, using three JK flip-flops. The clock was connected to the first flip-flop, while the outputs Q0 and Q1 were used as clocks for the next flip-flops. Since JK flip-flops automatically toggle, I did not need extra logic gates. I checked each output combination from 000 to 111 and captured screenshots, which are shown above.

After making the circuit, I wrote a Verilog code for the same 3-bit asynchronous up counter. Each flip-flop had J = K = 1 so it always toggles. The first flip-flop got the external clock, and the others were triggered by the inverted output of the previous flip-flop. I ran a simulation in Icarus Verilog, and the log showed the outputs Q2, Q1, and

Q0 at each clock pulse. The values I got were 000, 001, 010, 011, 100, 101, 110, 111. These values matched what I observed in CircuitVerse and EPWave.

Comparing all three results, the counter worked correctly. Each method showed the same counting sequence. This proves that the 3-bit asynchronous up counter works as expected. It also shows that the Verilog code matches the actual circuit and the simulations. Overall, this experiment helped me understand how asynchronous counters work and how flip-flops toggle to produce a counting sequence.

ASSESSMENT 2: ***3-BIT DOWN SYNCHRONOUS COUNTER***

A **3-bit DOWN synchronous counter** that decrements its count on each clock pulse, with all flip-flops triggered simultaneously.

A. CIRCUIT DIAGRAM:

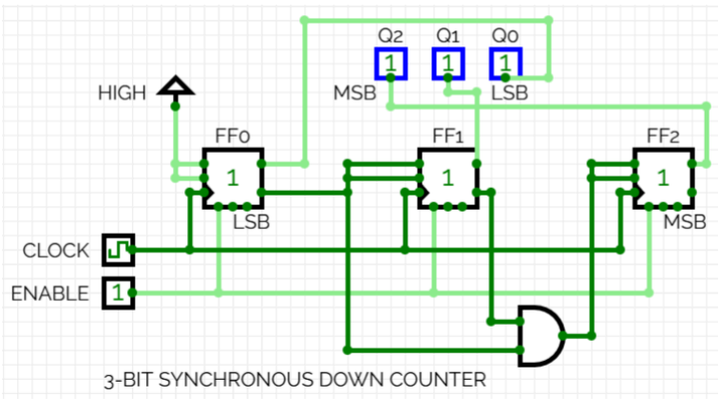


Figure 1.1 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 111

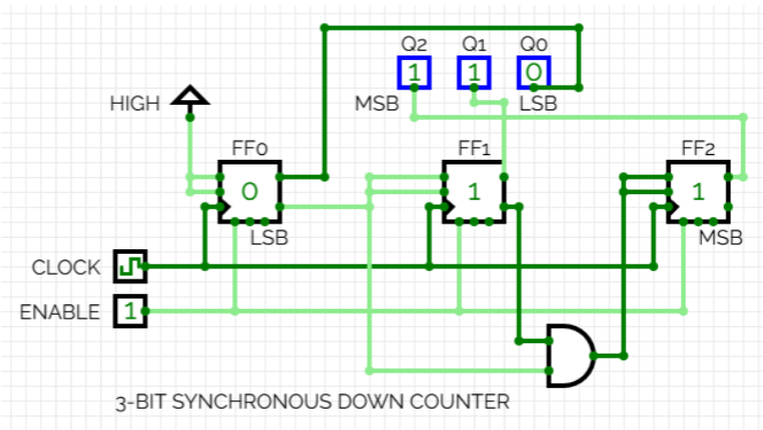


Figure 1.2 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 110

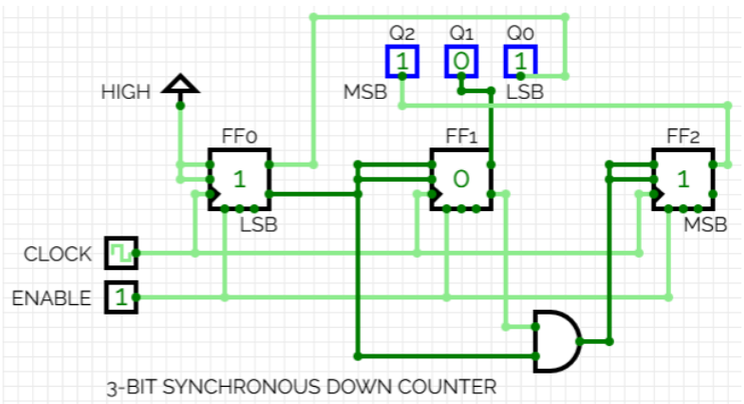


Figure 1.3 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 101

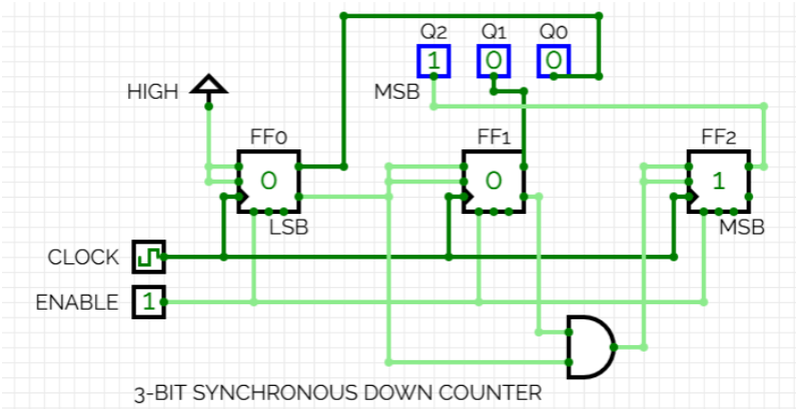


Figure 1.4 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 100

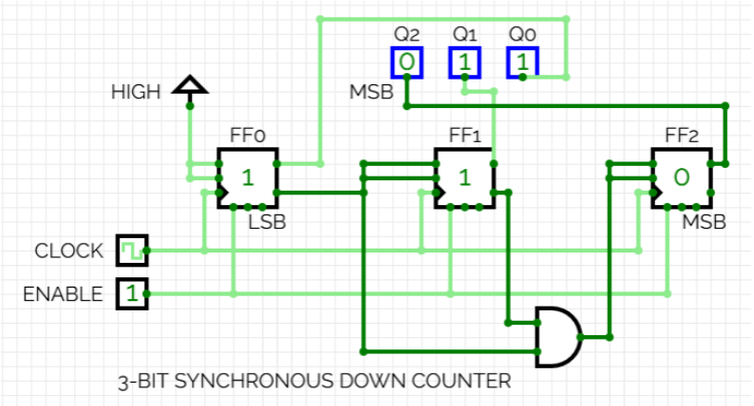


Figure 1.5 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 011

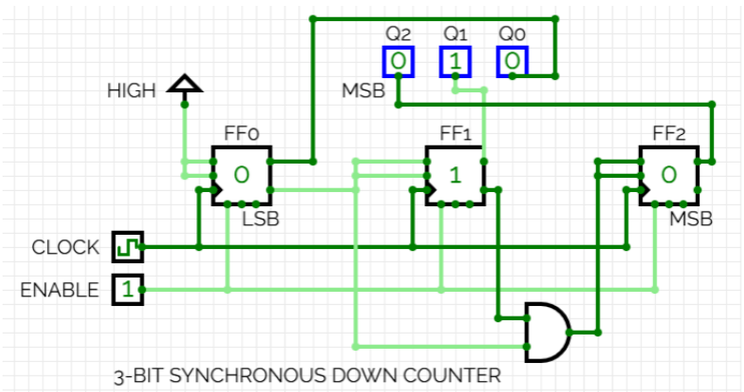


Figure 1.6 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 010

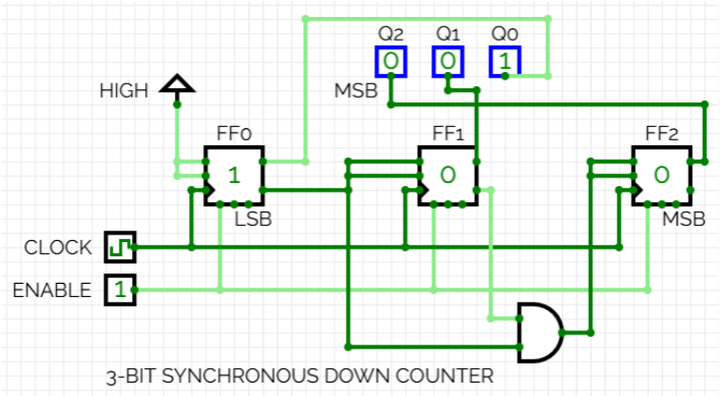


Figure 1.7 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 001

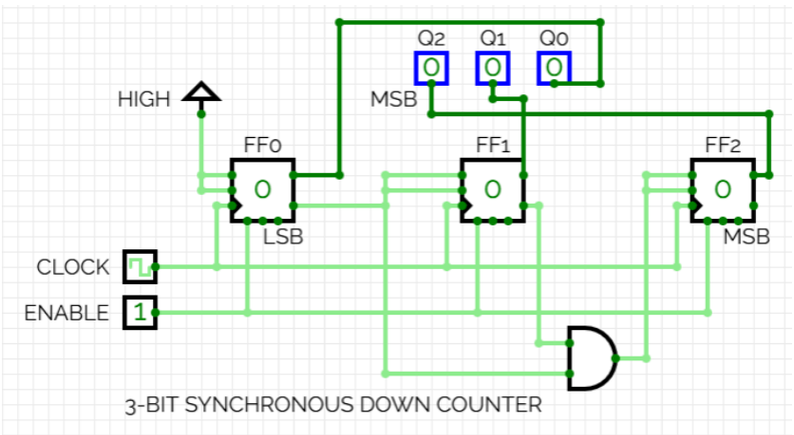


Figure 1.8 – 3-bit Synchronous Down Counter: Q2 Q1 Q0 = 000

B. VERILOG CODE *(paste the actual code):*

Design Code:

```
// JK Flip-Flop
module jk_ff (
```

College of Engineering – Department of Electrical Engineering

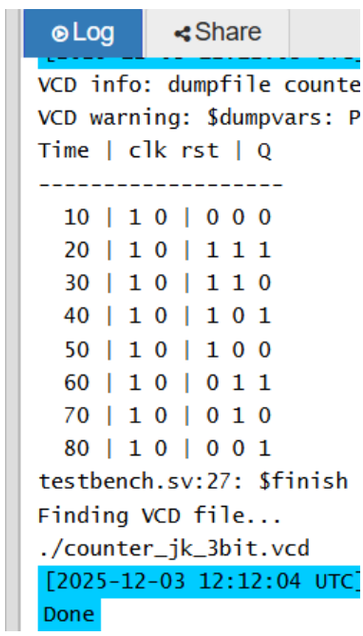
```
    input clk,
    input rst,
    input J,
    input K,
    output reg Q
);

    always @(posedge clk or posedge rst) begin
        if (rst)
            Q <= 0;
        else begin
            case ({J, K})
                2'b00: Q <= Q;      // no change
                2'b01: Q <= 0;      // reset
                2'b10: Q <= 1;      // set
                2'b11: Q <= ~Q;     // toggle
            endcase
        end
    end
endmodule

// Synchronous Down Counter (3-bit)
module sync_down_counter_3bit (
    input clk, input rst,
    output [2:0] Q
);

    // Toggle conditions for DOWN counter
    jk_ff ff0 (.clk(clk), .rst(rst), .J(1'b1), .K(1'b1), .Q(Q[0]));
    jk_ff ff1 (.clk(clk), .rst(rst), .J(~Q[0]), .K(~Q[0]), .Q(Q[1]));
    jk_ff ff2 (.clk(clk), .rst(rst), .J(~Q[0] & ~Q[1]), .K(~Q[0] & ~Q[1]), .Q(Q[2]));

endmodule
```

Testbench Code:

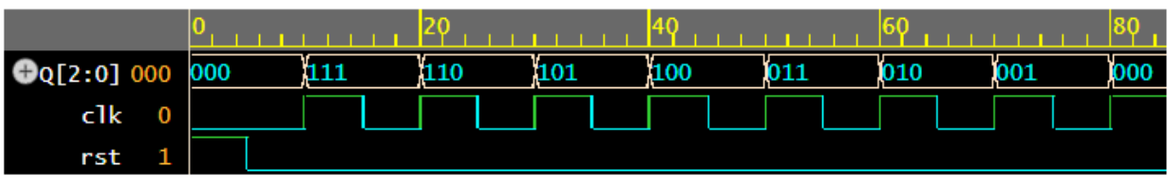
```
module tb_sync_down_counter_3bit;
    reg clk, rst;
    wire [2:0] Q;

    sync_down_counter_3bit uut (.clk(clk), .rst(rst), .Q(Q));

    initial begin
        $dumpfile("counter_jk_3bit.vcd"); $dumpvars;
        $display("Time | clk rst | Q");
        $display("-------------------");
```

```
        clk = 0; rst = 1;
        #5 rst = 0; // release reset

        // Generate clock pulses
        repeat (16) begin
            #5 clk = ~clk; // toggle clock
            if (clk) begin // only display at posedge
                $display("%4t | %b %b | %b %b %b",
                    $time, clk, rst, Q[2], Q[1], Q[0]);
            end
        end

        $finish;
    end
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)*

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

For this experiment, I built a 3-bit synchronous down counter using JK flip-flops. Unlike the asynchronous counter, all flip-flops in a synchronous counter share the same clock, which makes them change their outputs at the same time. I first designed the counter using the toggle conditions in the JK flip-flops, where each flip-flop toggles based on the combination of outputs from the previous flip-flops. The first flip-flop always toggles on every clock pulse, the second flip-flop toggles when the first output is low, and the third flip-flop toggles when both the first and second outputs are low.

I then implemented the counter in Verilog, following the same logic. Each JK flip-flop was set with J = K = 1 to enable toggling. I ran a testbench that generated clock pulses and recorded the outputs at every positive edge of the clock. The outputs counted down in sequence from 111 to 000, and the spacing of the outputs in the display was clear and easy to read.

Comparing the theoretical operation, the Verilog simulation, and the testbench outputs, all results matched perfectly. Each clock pulse produced the expected sequence, showing that the counter works correctly. This confirms that the 3-bit synchronous down counter behaves as designed, and the Verilog code accurately represents the logic of the synchronous circuit. Overall, the experiment helped me understand how synchronous counters operate and how JK flip-flops can be configured to produce a controlled down-counting sequence.

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)**	
Circuit Diagram	
Diagram is complete, clear, and accurately reflects counter logic and flip-flop connections. **(5 Points)**	
Diagram is mostly correct but has minor errors or missing labels. **(3 Points)**	
Diagram is unclear, incomplete, or incorrect. **(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:	