Digital Design and Computer Organisation Laboratory

3rd Semester, Academic Year 2025

Date:29-09-2025

Name: Aakash Desai	SRN:PES1UG24CS006	Section 3A

Week Number: 6 Program Number: 1

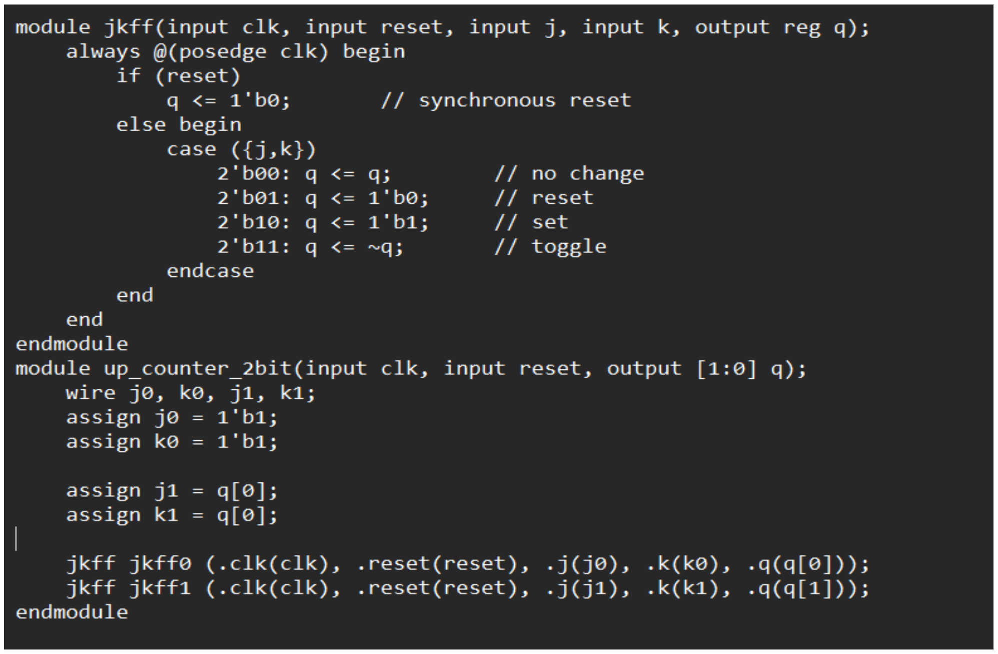
TITLE: Write an iverilog program to design and implement a 2-bit up counter using JK flip flops.Generate the VVP output and simulation waveform using GTKWave.

Deliverables

I. Verilog Code Screenshot
II. Verilog VVP Output Screen Shot
III. GTKWAVE Screenshot
IV. Output Table to be completed and included

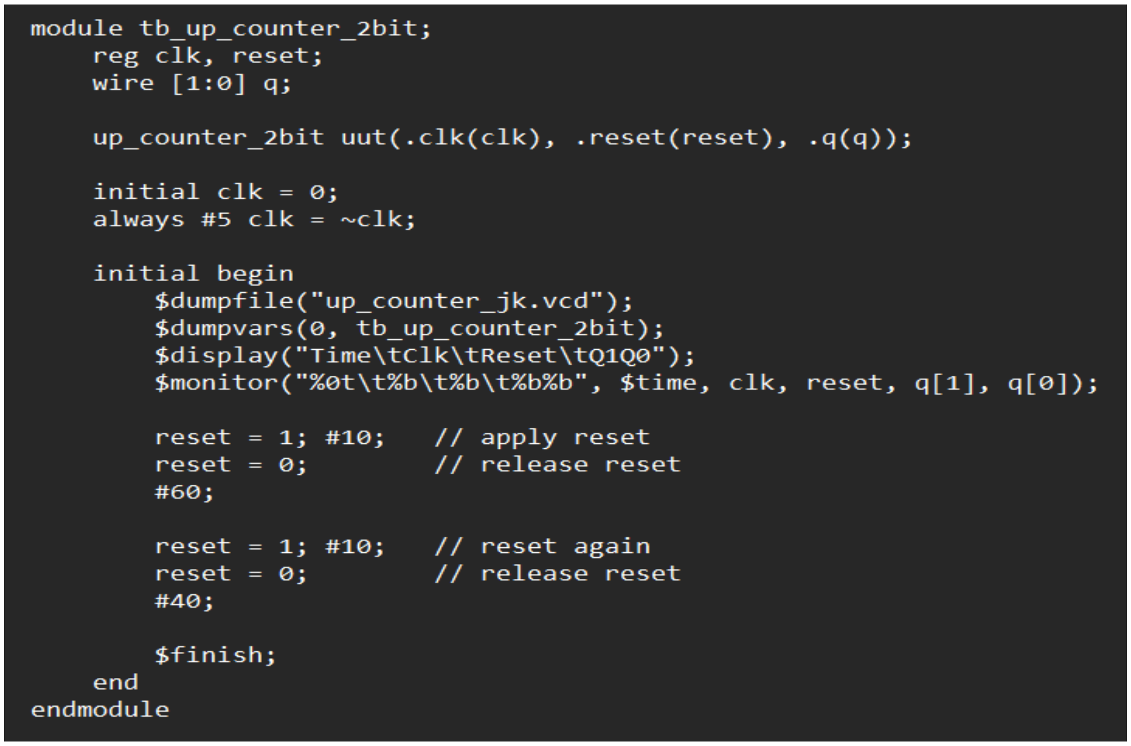
I.Verilog Code Snippet

- Main code



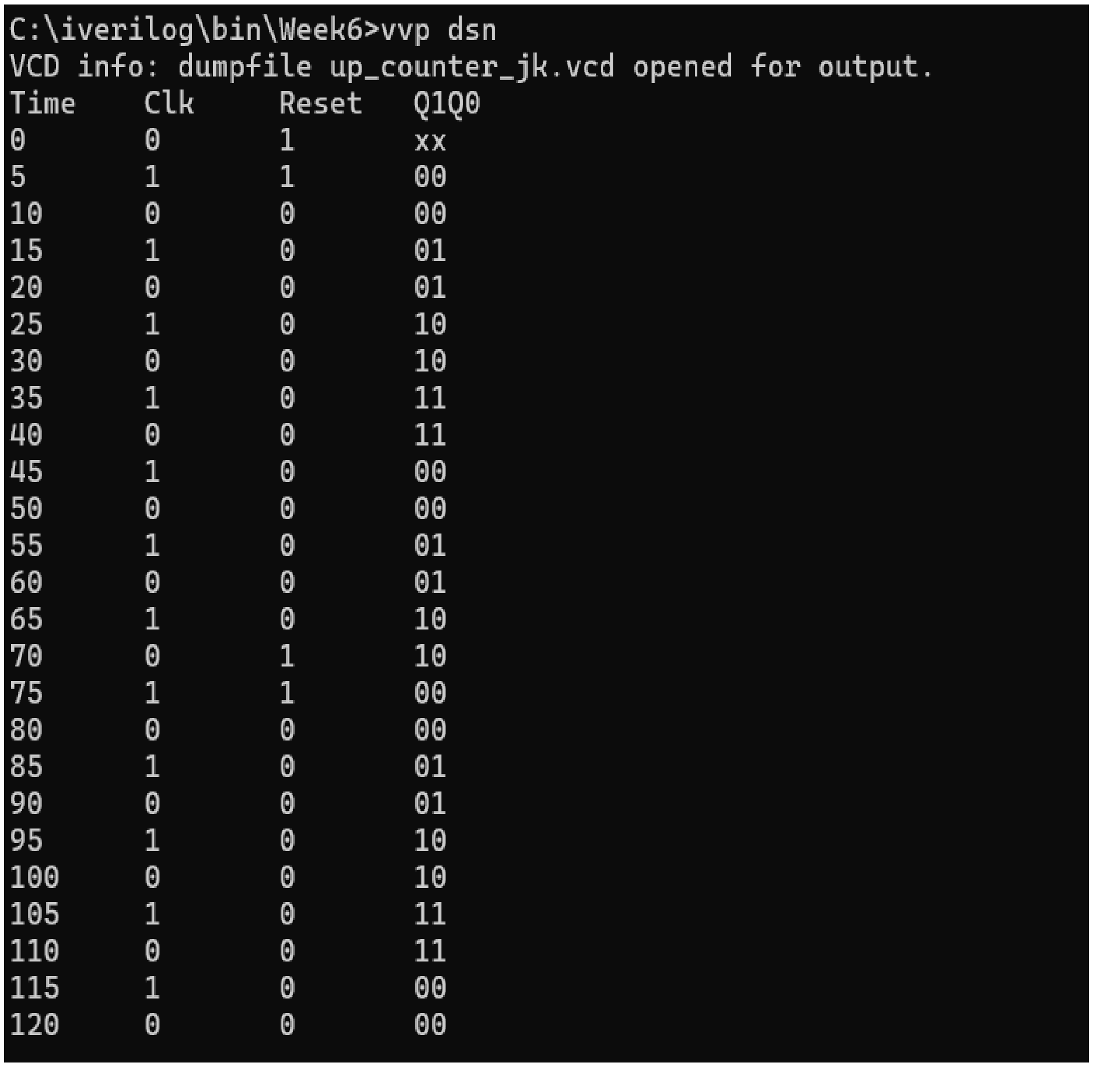
```verilog
module jkff(input clk, input reset, input j, input k, output reg q);
    always @(posedge clk) begin
        if (reset)
            q <= 1'b0;       // synchronous reset
        else begin
            case ({j,k})
                2'b00: q <= q;        // no change
                2'b01: q <= 1'b0;     // reset
                2'b10: q <= 1'b1;     // set
                2'b11: q <= ~q;       // toggle
            endcase
        end
    end
endmodule
module up_counter_2bit(input clk, input reset, output [1:0] q);
    wire j0, k0, j1, k1;
    assign j0 = 1'b1;
    assign k0 = 1'b1;

    assign j1 = q[0];
    assign k1 = q[0];

    jkff jkff0 (.clk(clk), .reset(reset), .j(j0), .k(k0), .q(q[0]));
    jkff jkff1 (.clk(clk), .reset(reset), .j(j1), .k(k1), .q(q[1]));
endmodule
```

- Test Bench



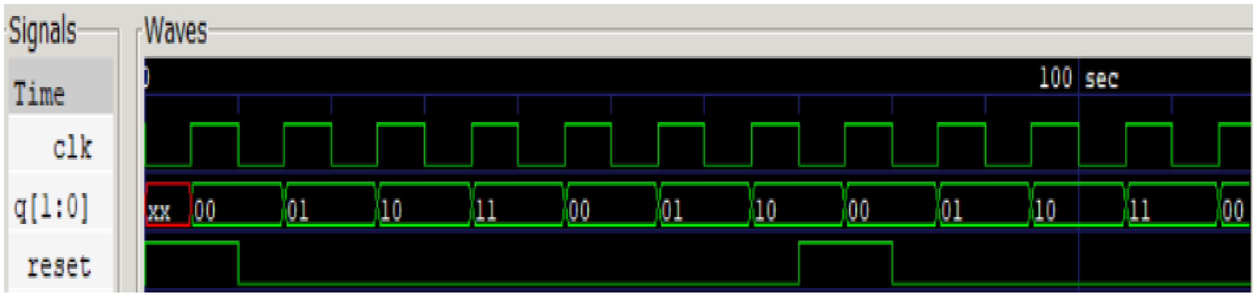
```verilog
module tb_up_counter_2bit;
    reg clk, reset;
    wire [1:0] q;

    up_counter_2bit uut(.clk(clk), .reset(reset), .q(q));

    initial clk = 0;
    always #5 clk = ~clk;

    initial begin
        $dumpfile("up_counter_jk.vcd");
        $dumpvars(0, tb_up_counter_2bit);
        $display("Time\tClk\tReset\tQ1Q0");
        $monitor("%0t\t%b\t%b\t%b%b", $time, clk, reset, q[1], q[0]);

        reset = 1; #10;   // apply reset
        reset = 0;        // release reset
        #60;

        reset = 1; #10;   // reset again
        reset = 0;        // release reset
        #40;

        $finish;
    end
endmodule
```

II Verilog VVP output ScreenShot



```
C:\iverilog\bin\Week6>vvp dsn
VCD info: dumpfile up_counter_jk.vcd opened for output.
Time    Clk     Reset   Q1Q0
0       0       1       xx
5       1       1       00
10      0       0       00
15      1       0       01
20      0       0       01
25      1       0       10
30      0       0       10
35      1       0       11
40      0       0       11
45      1       0       00
50      0       0       00
55      1       0       01
60      0       0       01
65      1       0       10
70      0       1       10
75      1       1       00
80      0       0       00
85      1       0       01
90      0       0       01
95      1       0       10
100     0       0       10
105     1       0       11
110     0       0       11
115     1       0       00
120     0       0       00
```

III GTKWave ScreenShot



IV Excitation Table

Current State		Next State		Flip Flops			
Q1	Q0	Q1	Q0	J1	K1	J0	K0
0	0	0	1	0	X	1	X
0	1	1	0	1	X	X	1
1	0	1	1	X	0	1	X
1	1	0	0	X	1	X	1

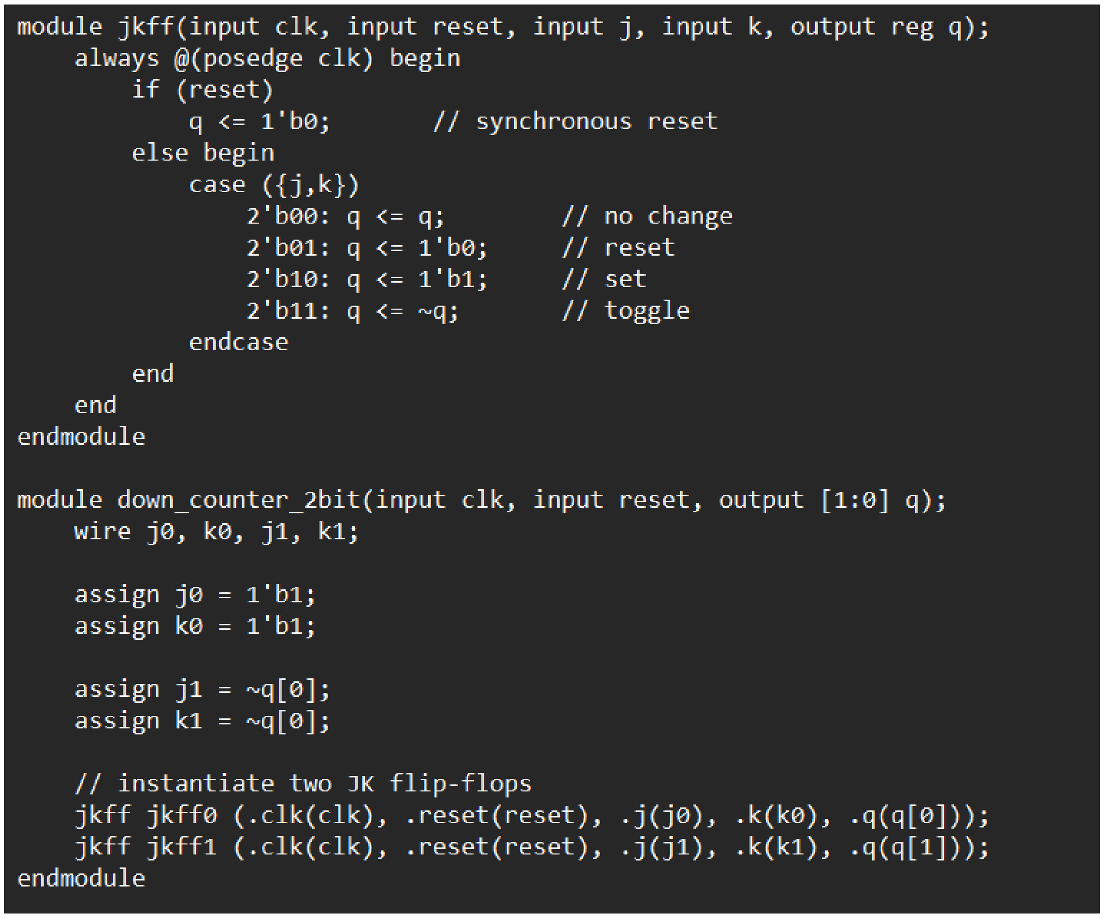
TITLE: Write an iverilog program to design and implement a 2-bit down counter using JK flip flops.Generate the VVP output and simulation waveform using GTKWave.

Deliverables

I. Verilog Code Screenshot
II. Verilog VVP Output Screen Shot
III. GTKWAVE Screenshot
IV. Output Table to be completed and included

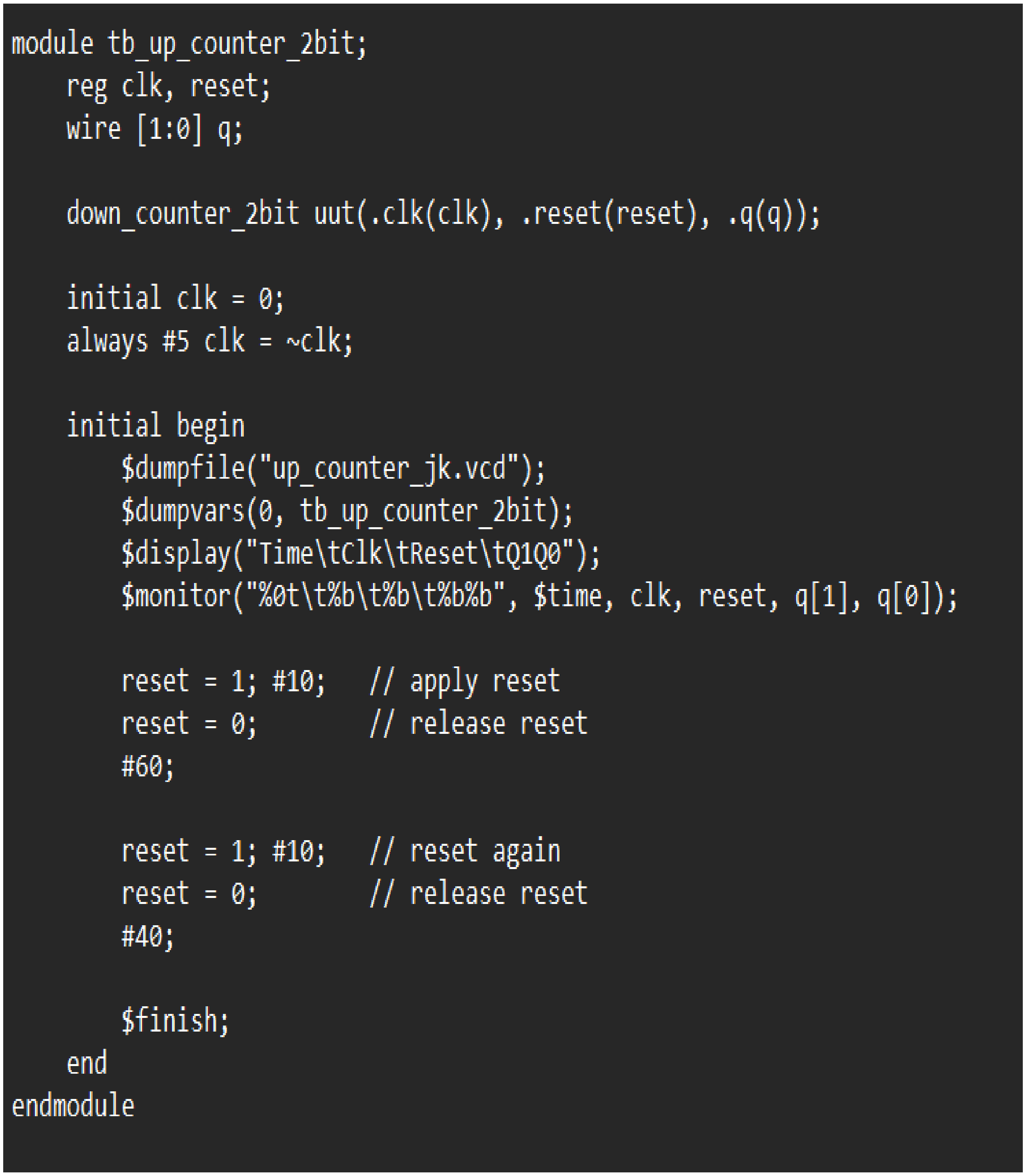
I.Verilog Code Snippet

- Main Code



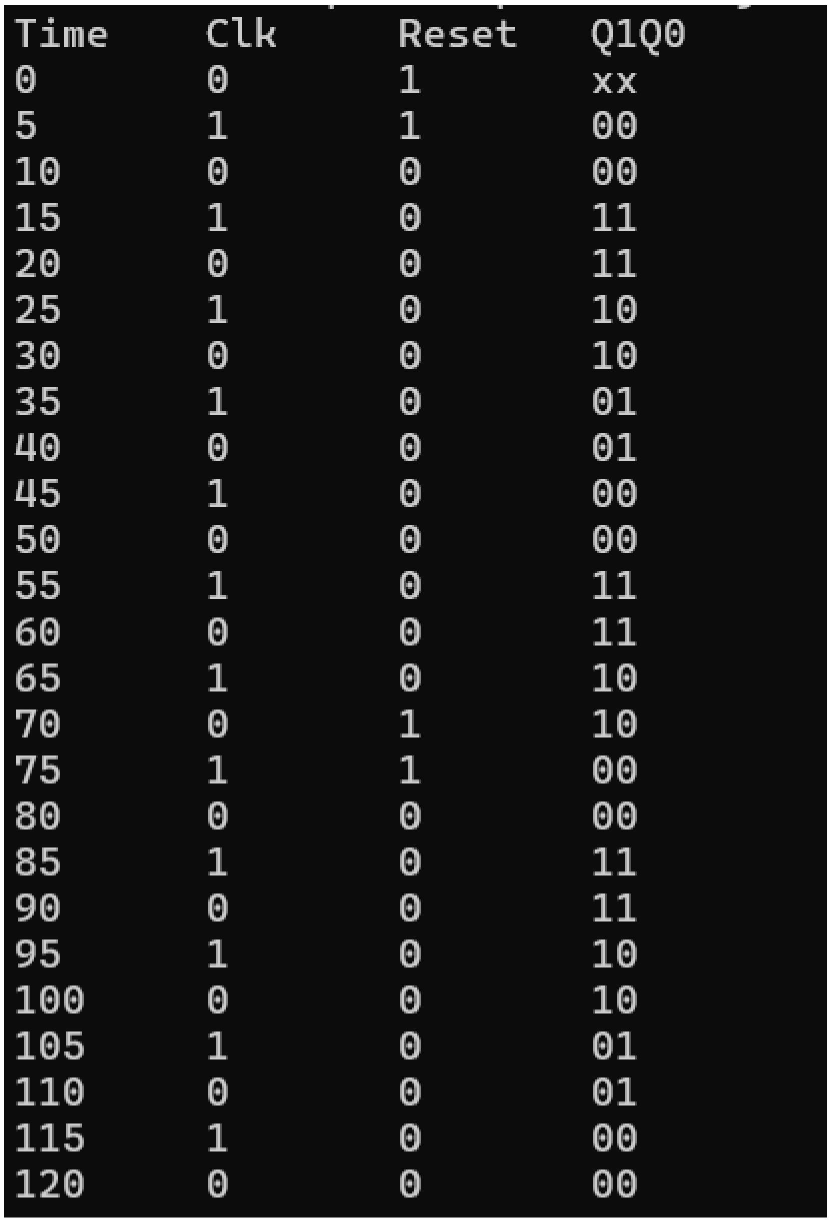
```
module jkff(input clk, input reset, input j, input k, output reg q);
    always @(posedge clk) begin
        if (reset)
            q <= 1'b0;       // synchronous reset
        else begin
            case ({j,k})
                2'b00: q <= q;        // no change
                2'b01: q <= 1'b0;     // reset
                2'b10: q <= 1'b1;     // set
                2'b11: q <= ~q;       // toggle
            endcase
        end
    end
endmodule

module down_counter_2bit(input clk, input reset, output [1:0] q);
    wire j0, k0, j1, k1;

    assign j0 = 1'b1;
    assign k0 = 1'b1;

    assign j1 = ~q[0];
    assign k1 = ~q[0];

    // instantiate two JK flip-flops
    jkff jkff0 (.clk(clk), .reset(reset), .j(j0), .k(k0), .q(q[0]));
    jkff jkff1 (.clk(clk), .reset(reset), .j(j1), .k(k1), .q(q[1]));
endmodule
```

- Test Bench



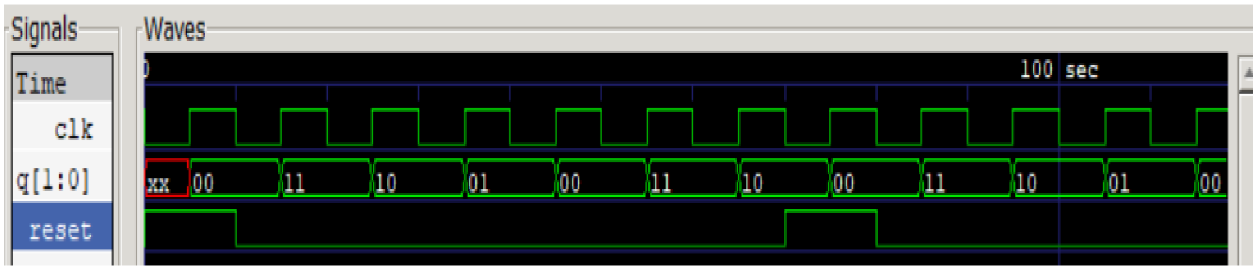
```verilog
module tb_up_counter_2bit;
    reg clk, reset;
    wire [1:0] q;

    down_counter_2bit uut(.clk(clk), .reset(reset), .q(q));

    initial clk = 0;
    always #5 clk = ~clk;

    initial begin
        $dumpfile("up_counter_jk.vcd");
        $dumpvars(0, tb_up_counter_2bit);
        $display("Time\tClk\tReset\tQ1Q0");
        $monitor("%0t\t%b\t%b\t%b%b", $time, clk, reset, q[1], q[0]);

        reset = 1; #10;   // apply reset
        reset = 0;        // release reset
        #60;

        reset = 1; #10;   // reset again
        reset = 0;        // release reset
        #40;

        $finish;
    end
endmodule
```

II Verilog VVP ScreenShot



```
Time      Clk       Reset     Q1Q0
0         0         1         xx
5         1         1         00
10        0         0         00
15        1         0         11
20        0         0         11
25        1         0         10
30        0         0         10
35        1         0         01
40        0         0         01
45        1         0         00
50        0         0         00
55        1         0         11
60        0         0         11
65        1         0         10
70        0         1         10
75        1         1         00
80        0         0         00
85        1         0         11
90        0         0         11
95        1         0         10
100       0         0         10
105       1         0         01
110       0         0         01
115       1         0         00
120       0         0         00
```

III GTK wave Screenshot



IV Excitation Table

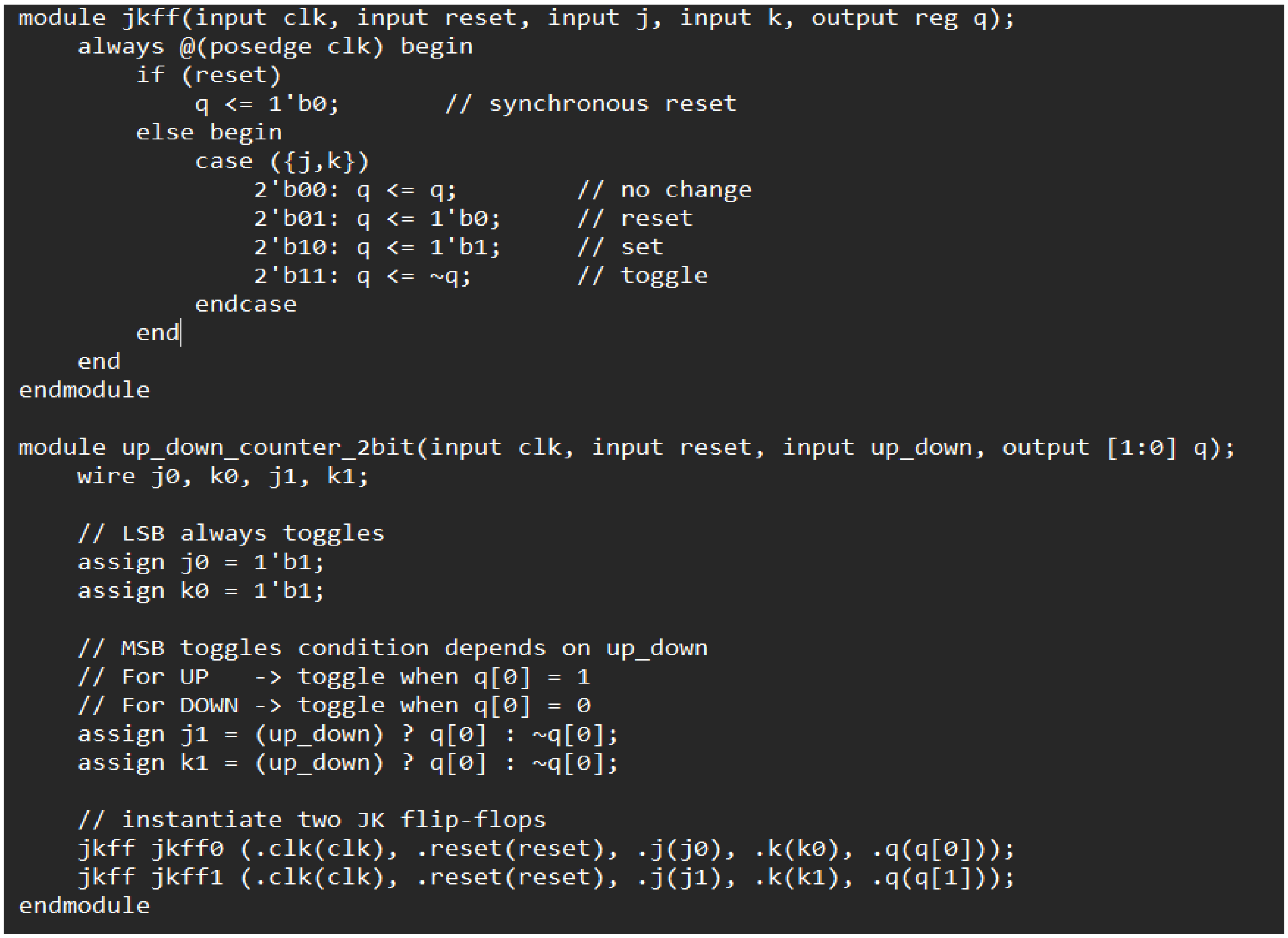
Current State		Next State		Flip Flops			
Q1	Q0	Q1	Q0	J1	K1	J0	K0
0	0	1	1	1	X	1	X
0	1	0	0	0	X	X	1
1	0	0	1	X	1	1	X
1	1	1	0	X	0	X	1

TITLE: Write an iverilog program to design and implement a 2-bit up-down counter using JK flip flops.Generate the VVP output and simulation waveform using GTKWave.

Deliverables

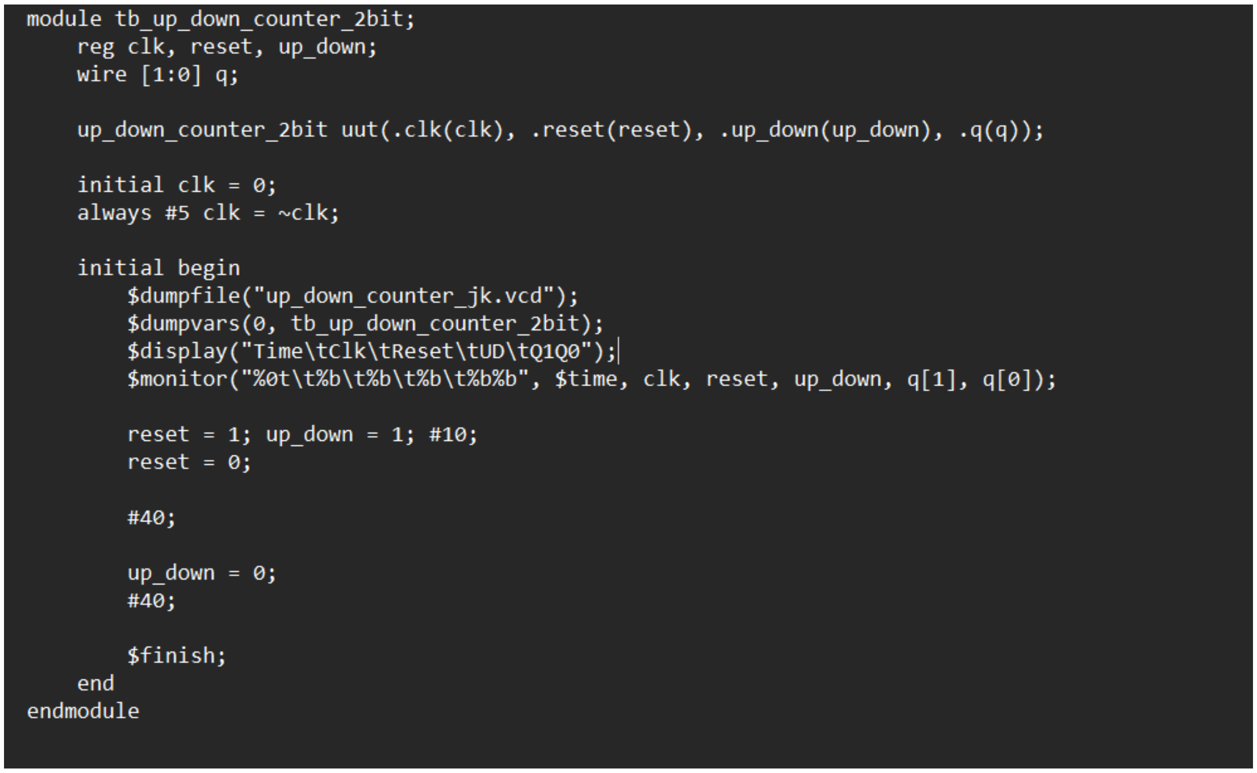
I. Verilog Code Screenshot
II. Verilog VVP Output Screen Shot
III. GTKWAVE Screenshot
IV. Output Table to be completed and included

- I Iverilog code



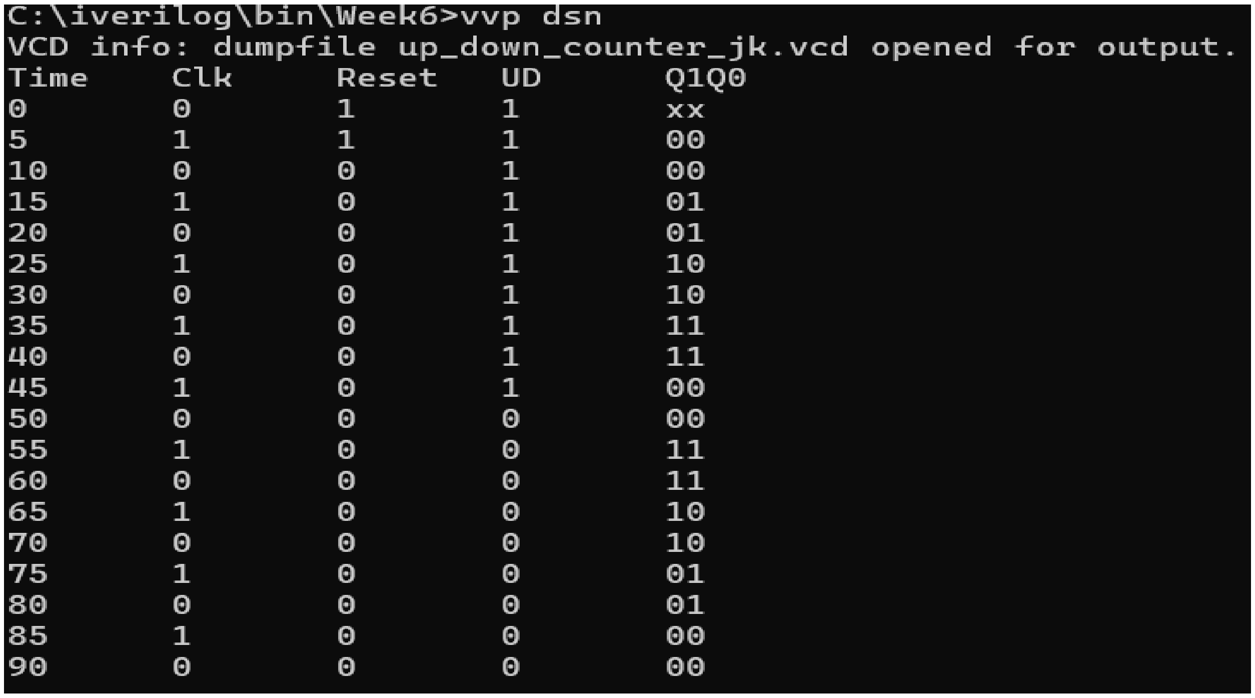
```
module jkff(input clk, input reset, input j, input k, output reg q);
    always @(posedge clk) begin
        if (reset)
            q <= 1'b0;       // synchronous reset
        else begin
            case ({j,k})
                2'b00: q <= q;        // no change
                2'b01: q <= 1'b0;     // reset
                2'b10: q <= 1'b1;     // set
                2'b11: q <= ~q;       // toggle
            endcase
        end
    end
endmodule

module up_down_counter_2bit(input clk, input reset, input up_down, output [1:0] q);
    wire j0, k0, j1, k1;

    // LSB always toggles
    assign j0 = 1'b1;
    assign k0 = 1'b1;

    // MSB toggles condition depends on up_down
    // For UP   -> toggle when q[0] = 1
    // For DOWN -> toggle when q[0] = 0
    assign j1 = (up_down) ? q[0] : ~q[0];
    assign k1 = (up_down) ? q[0] : ~q[0];

    // instantiate two JK flip-flops
    jkff jkff0 (.clk(clk), .reset(reset), .j(j0), .k(k0), .q(q[0]));
    jkff jkff1 (.clk(clk), .reset(reset), .j(j1), .k(k1), .q(q[1]));
endmodule
```

- Testbench



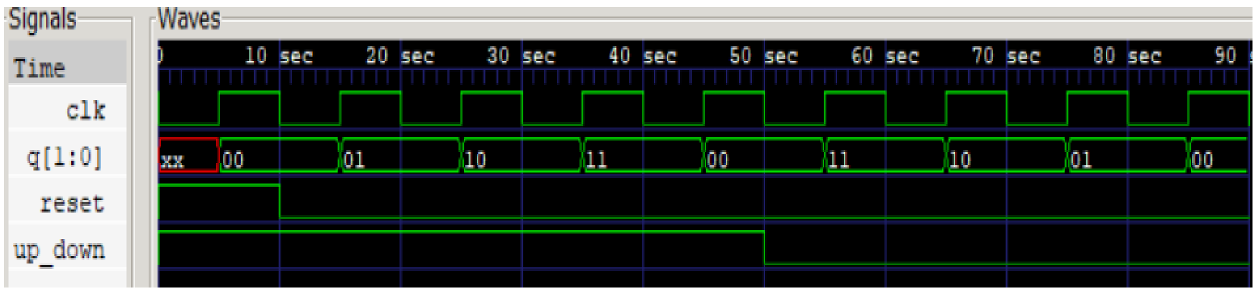
```
module tb_up_down_counter_2bit;
    reg clk, reset, up_down;
    wire [1:0] q;

    up_down_counter_2bit uut(.clk(clk), .reset(reset), .up_down(up_down), .q(q));

    initial clk = 0;
    always #5 clk = ~clk;

    initial begin
        $dumpfile("up_down_counter_jk.vcd");
        $dumpvars(0, tb_up_down_counter_2bit);
        $display("Time\tClk\tReset\tUD\tQ1Q0");
        $monitor("%0t\t%b\t%b\t%b\t%b%b", $time, clk, reset, up_down, q[1], q[0]);

        reset = 1; up_down = 1; #10;
        reset = 0;

        #40;

        up_down = 0;
        #40;

        $finish;
    end
endmodule
```

II VVP screenshot



```
C:\iverilog\bin\Week6>vvp dsn
VCD info: dumpfile up_down_counter_jk.vcd opened for output.
Time    Clk     Reset   UD      Q1Q0
0       0       1       1       xx
5       1       1       1       00
10      0       0       1       00
15      1       0       1       01
20      0       0       1       01
25      1       0       1       10
30      0       0       1       10
35      1       0       1       11
40      0       0       1       11
45      1       0       1       00
50      0       0       0       00
55      1       0       0       11
60      0       0       0       11
65      1       0       0       10
70      0       0       0       10
75      1       0       0       01
80      0       0       0       01
85      1       0       0       00
90      0       0       0       00
```

III GTKwave screenshot



IV Characteristic Table

Mode Select	Current State		Next State		Flip Flops			
	Q1	Q0	Q1	Q0	J1	K1	J0	K0
0	0	0	0	1	0	X	1	X
1	0	0	1	1	1	X	1	X
0	0	1	1	0	1	X	X	1
1	0	1	0	0	0	X	X	1
0	1	0	1	1	X	0	1	X
1	1	0	0	1	X	1	1	X
0	1	1	0	0	X	1	X	1
1	1	1	1	0	X	0	X	1

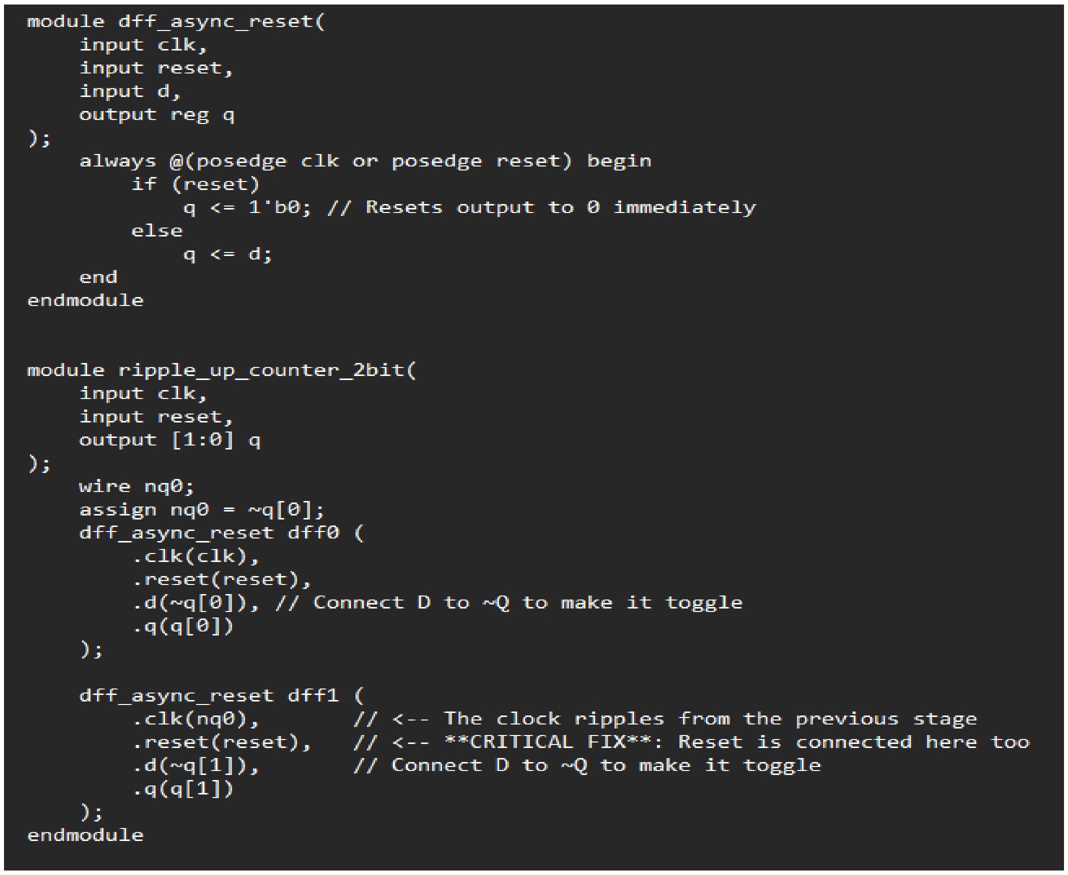
TITLE: Write a verilog program to design a 2 bit ripple up counter using D flip flop. Generate the VVP output and simulation waveform using GTKWave.

Deliverables

I. Verilog Code Screenshot
II. Verilog VVP Output Screen Shot
III. GTKWAVE Screenshot
IV. Output Table to be completed and included

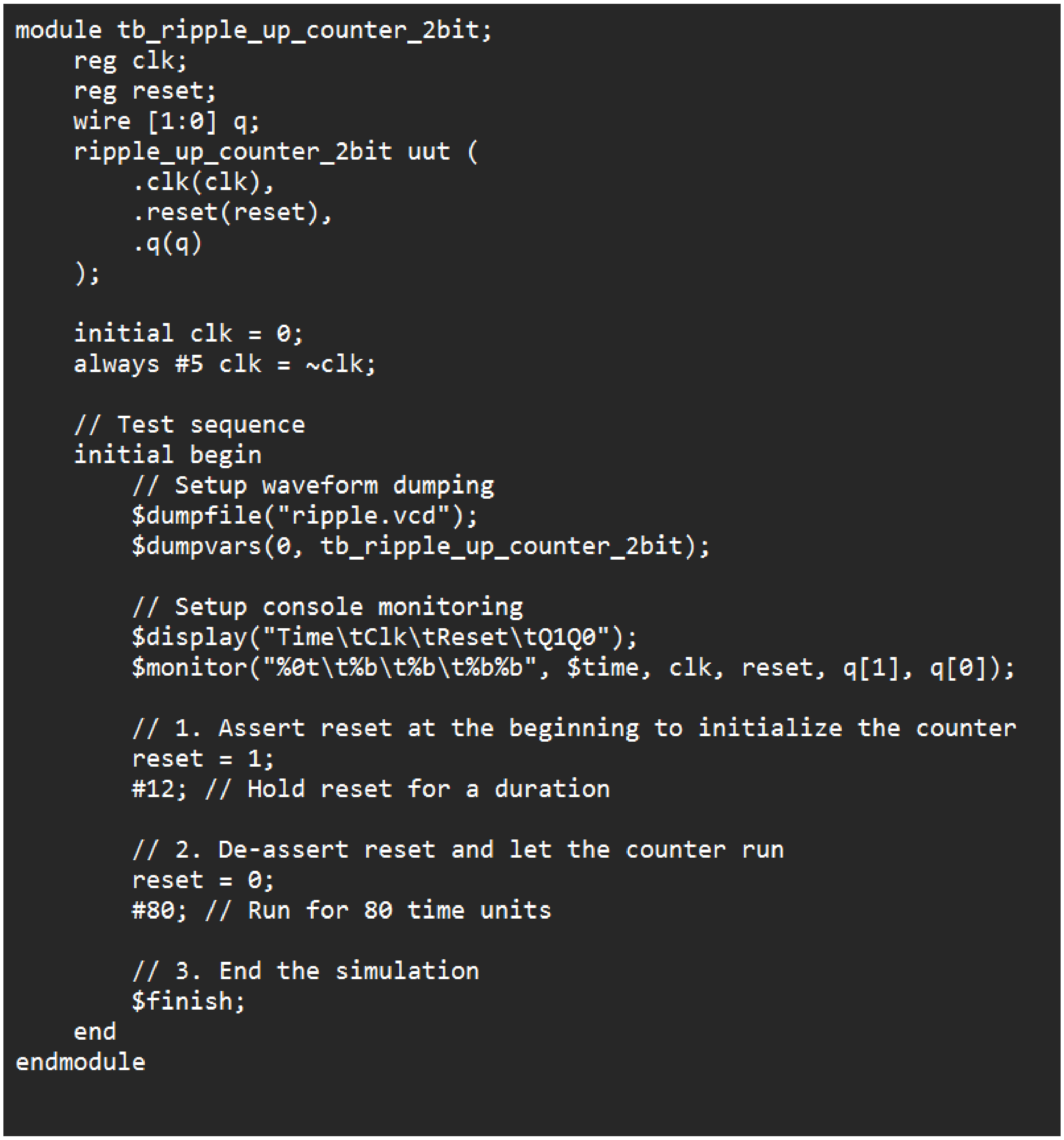
I.Verilog code

- Main code



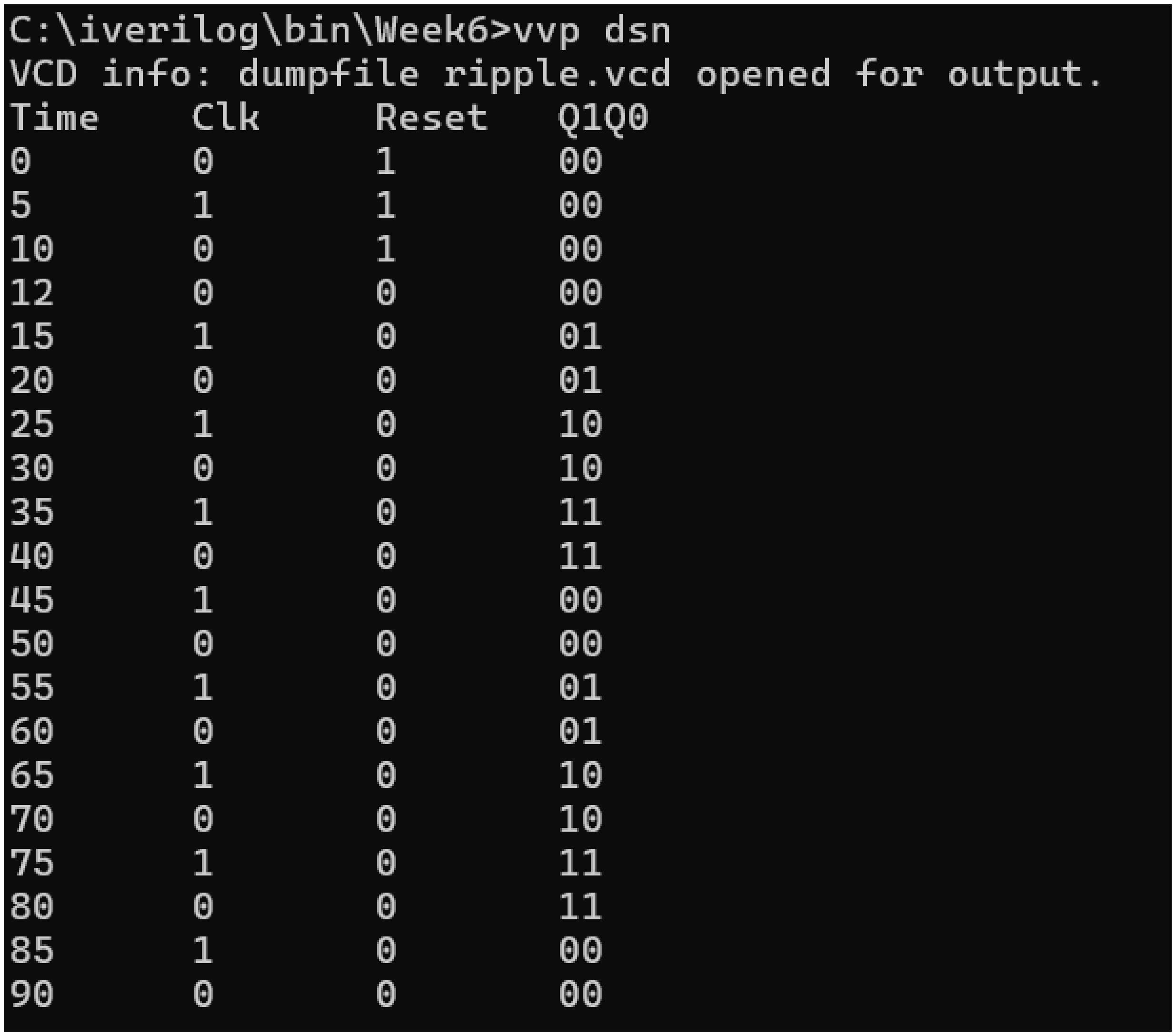
```
module dff_async_reset(
    input clk,
    input reset,
    input d,
    output reg q
);
    always @(posedge clk or posedge reset) begin
        if (reset)
            q <= 1'b0; // Resets output to 0 immediately
        else
            q <= d;
    end
endmodule


module ripple_up_counter_2bit(
    input clk,
    input reset,
    output [1:0] q
);
    wire nq0;
    assign nq0 = ~q[0];
    dff_async_reset dff0 (
        .clk(clk),
        .reset(reset),
        .d(~q[0]), // Connect D to ~Q to make it toggle
        .q(q[0])
    );

    dff_async_reset dff1 (
        .clk(nq0),       // <-- The clock ripples from the previous stage
        .reset(reset),   // <-- **CRITICAL FIX**: Reset is connected here too
        .d(~q[1]),       // Connect D to ~Q to make it toggle
        .q(q[1])
    );
endmodule
```

- Test bench



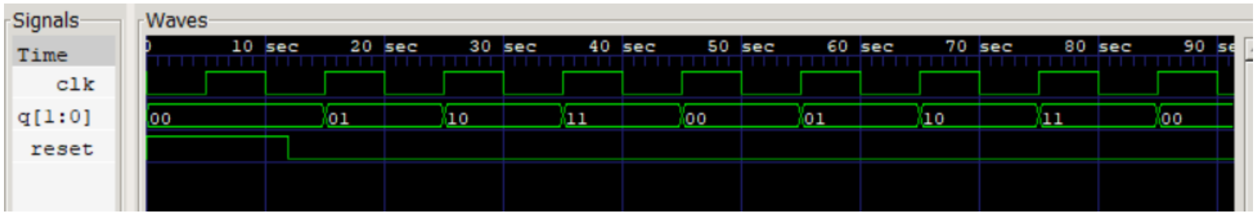
```
module tb_ripple_up_counter_2bit;
    reg clk;
    reg reset;
    wire [1:0] q;
    ripple_up_counter_2bit uut (
        .clk(clk),
        .reset(reset),
        .q(q)
    );

    initial clk = 0;
    always #5 clk = ~clk;

    // Test sequence
    initial begin
        // Setup waveform dumping
        $dumpfile("ripple.vcd");
        $dumpvars(0, tb_ripple_up_counter_2bit);

        // Setup console monitoring
        $display("Time\tClk\tReset\tQ1Q0");
        $monitor("%0t\t%b\t%b\t%b%b", $time, clk, reset, q[1], q[0]);

        // 1. Assert reset at the beginning to initialize the counter
        reset = 1;
        #12; // Hold reset for a duration

        // 2. De-assert reset and let the counter run
        reset = 0;
        #80; // Run for 80 time units

        // 3. End the simulation
        $finish;
    end
endmodule
```

II VVP Screenshot



```
C:\iverilog\bin\Week6>vvp dsn
VCD info: dumpfile ripple.vcd opened for output.
Time    Clk     Reset   Q1Q0
0       0       1       00
5       1       1       00
10      0       1       00
12      0       0       00
15      1       0       01
20      0       0       01
25      1       0       10
30      0       0       10
35      1       0       11
40      0       0       11
45      1       0       00
50      0       0       00
55      1       0       01
60      0       0       01
65      1       0       10
70      0       0       10
75      1       0       11
80      0       0       11
85      1       0       00
90      0       0       00
```

III GTKWave output



IV State Transition Table

Current State		Next State	
Q1	Q0	Q1	Q0
0	0	0	1
0	1	1	0
1	0	1	1
1	1	0	0

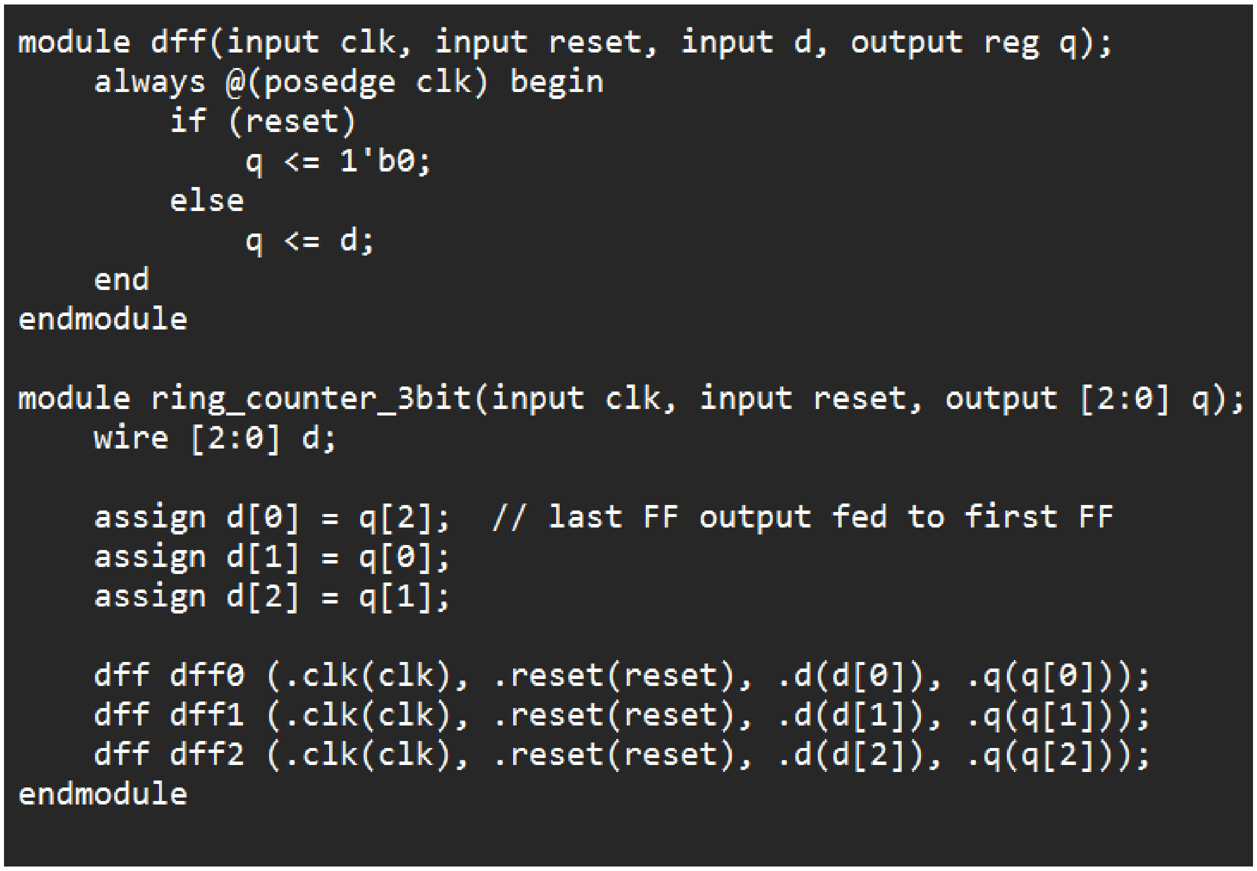
TITLE: Write a verilog program to design and implement a 3 bit Ring Counter using D Flip flop. Generate the VVP output and simulation waveform using GTKWave.

Deliverables

I. Verilog Code Screenshot
II. Verilog VVP Output Screen Shot
III. GTKWAVE Screenshot
IV. Output Table to be completed and included

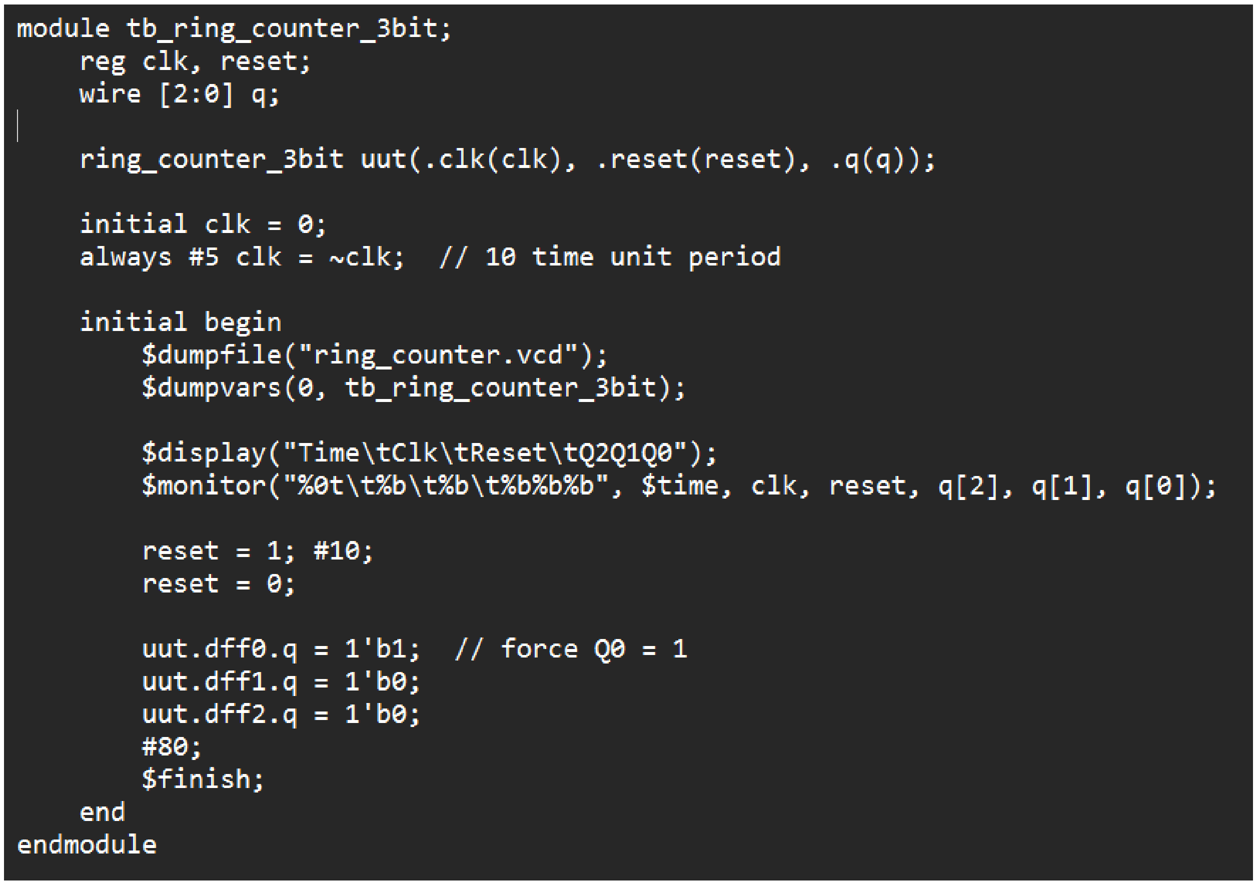
I Verilog Code

- Main code



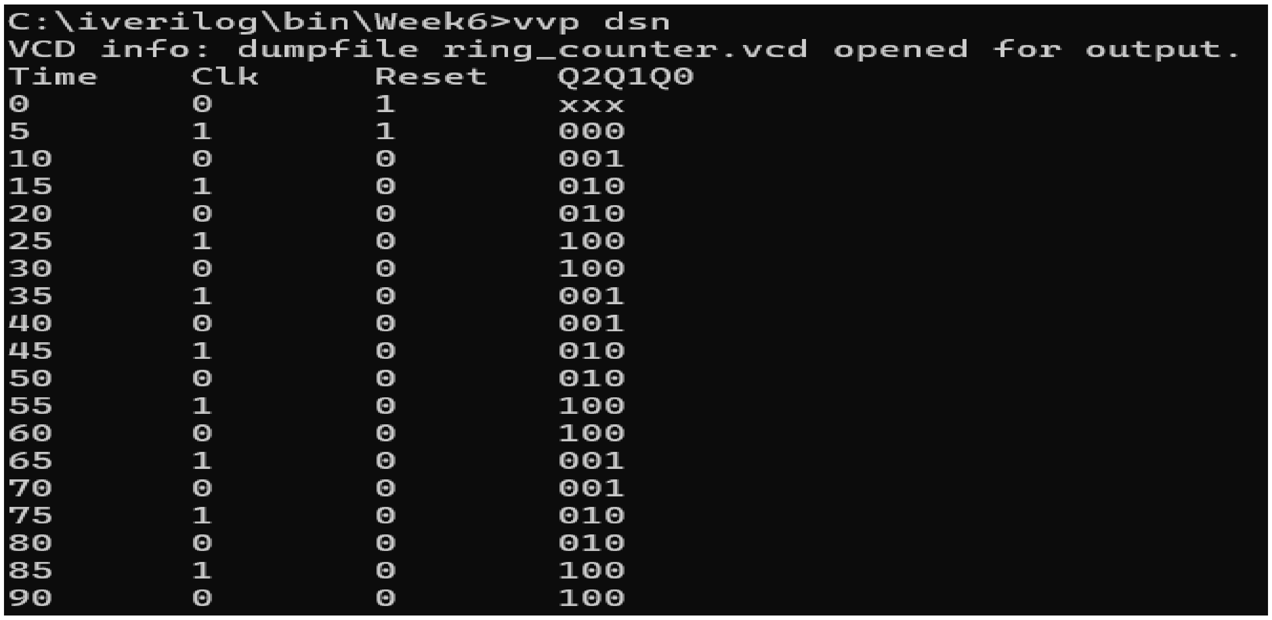
```
module dff(input clk, input reset, input d, output reg q);
    always @(posedge clk) begin
        if (reset)
            q <= 1'b0;
        else
            q <= d;
    end
endmodule

module ring_counter_3bit(input clk, input reset, output [2:0] q);
    wire [2:0] d;

    assign d[0] = q[2];  // last FF output fed to first FF
    assign d[1] = q[0];
    assign d[2] = q[1];

    dff dff0 (.clk(clk), .reset(reset), .d(d[0]), .q(q[0]));
    dff dff1 (.clk(clk), .reset(reset), .d(d[1]), .q(q[1]));
    dff dff2 (.clk(clk), .reset(reset), .d(d[2]), .q(q[2]));
endmodule
```

- Test bench



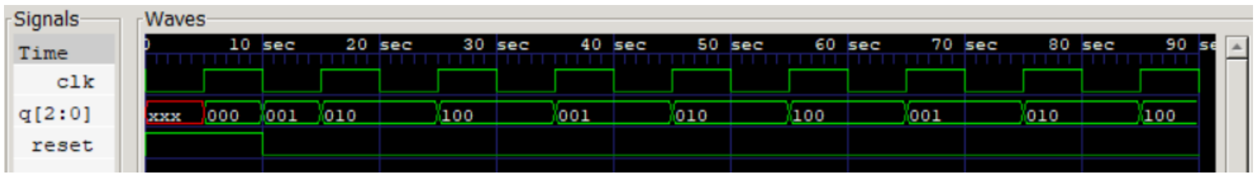
```
module tb_ring_counter_3bit;
    reg clk, reset;
    wire [2:0] q;

    ring_counter_3bit uut(.clk(clk), .reset(reset), .q(q));

    initial clk = 0;
    always #5 clk = ~clk;  // 10 time unit period

    initial begin
        $dumpfile("ring_counter.vcd");
        $dumpvars(0, tb_ring_counter_3bit);

        $display("Time\tClk\tReset\tQ2Q1Q0");
        $monitor("%0t\t%b\t%b\t%b%b%b", $time, clk, reset, q[2], q[1], q[0]);

        reset = 1; #10;
        reset = 0;

        uut.dff0.q = 1'b1;  // force Q0 = 1
        uut.dff1.q = 1'b0;
        uut.dff2.q = 1'b0;
        #80;
        $finish;
    end
endmodule
```

II VVP output Screenshot



```
C:\iverilog\bin\Week6>vvp dsn
VCD info: dumpfile ring_counter.vcd opened for output.
Time    Clk     Reset   Q2Q1Q0
0       0       1       xxx
5       1       1       000
10      0       0       001
15      1       0       010
20      0       0       010
25      1       0       100
30      0       0       100
35      1       0       001
40      0       0       001
45      1       0       010
50      0       0       010
55      1       0       100
60      0       0       100
65      1       0       001
70      0       0       001
75      1       0       010
80      0       0       010
85      1       0       100
90      0       0       100
```

III GTKwave output screenshot



IV Transition Table

Current State			D inputs			Next State		
Q2	Q1	Q0	D2	D1	D0	Q2	Q1	Q0
0	0	1	0	1	0	0	1	0
0	1	0	1	0	0	1	0	0
1	0	0	0	0	1	0	0	1
0	0	1	0	1	0	0	1	0

The counter has been initialized to a "one-hot" state of **001**.

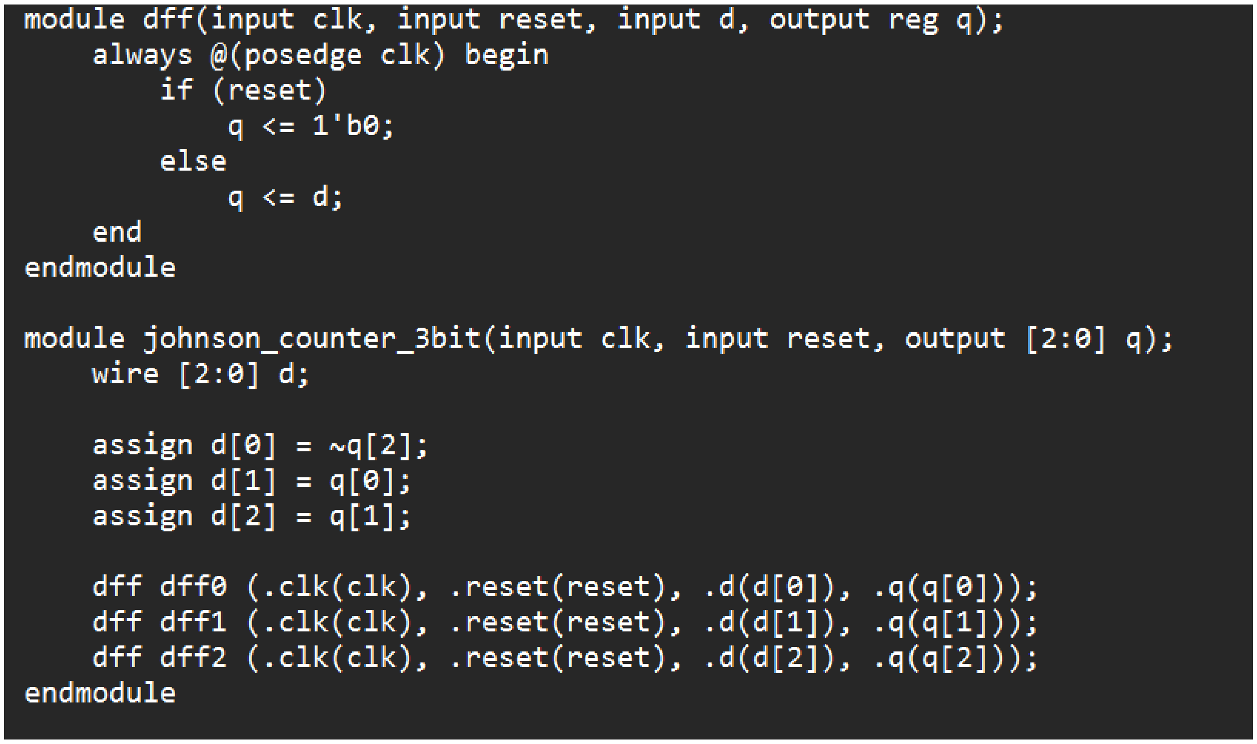
TITLE: Write a verilog program to design and implement a 3 bit Johnson Counter using D Flip flop. Generate the VVP output and simulation waveform using GTKWave.

Deliverables

I. Verilog Code Screenshot
II. Verilog VVP Output Screen Shot
III. GTKWAVE Screenshot
IV. Output Table to be completed and included

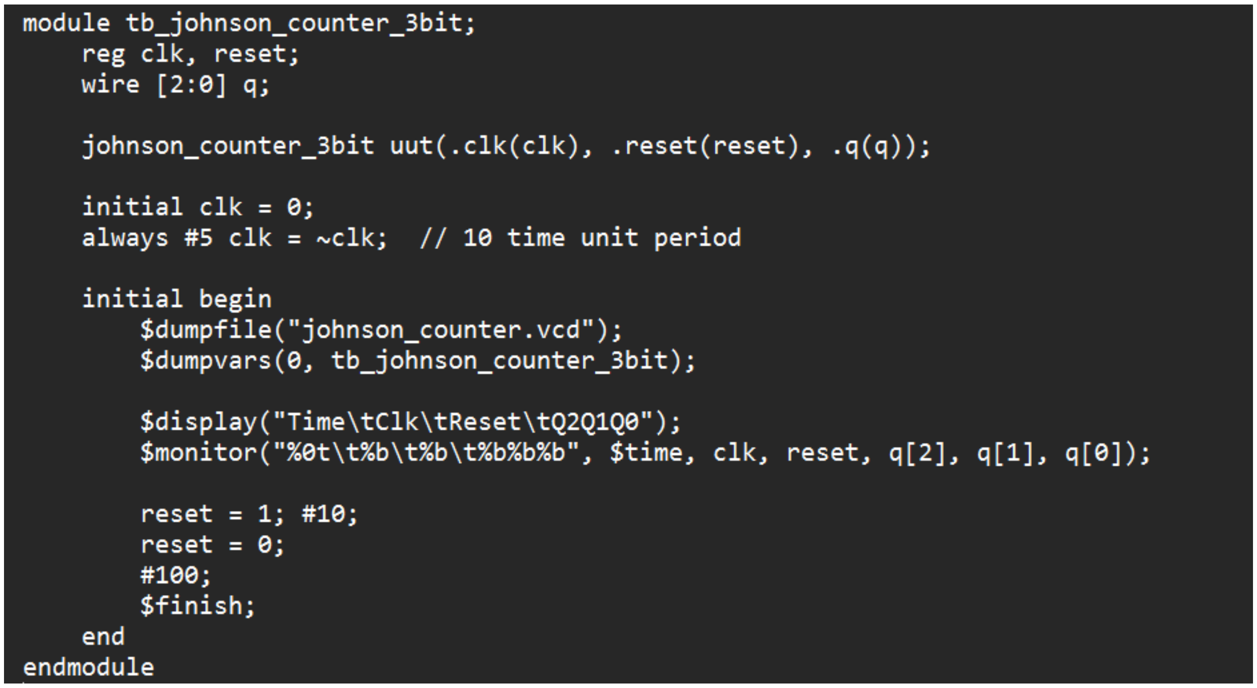
I Verilog Code

Main code



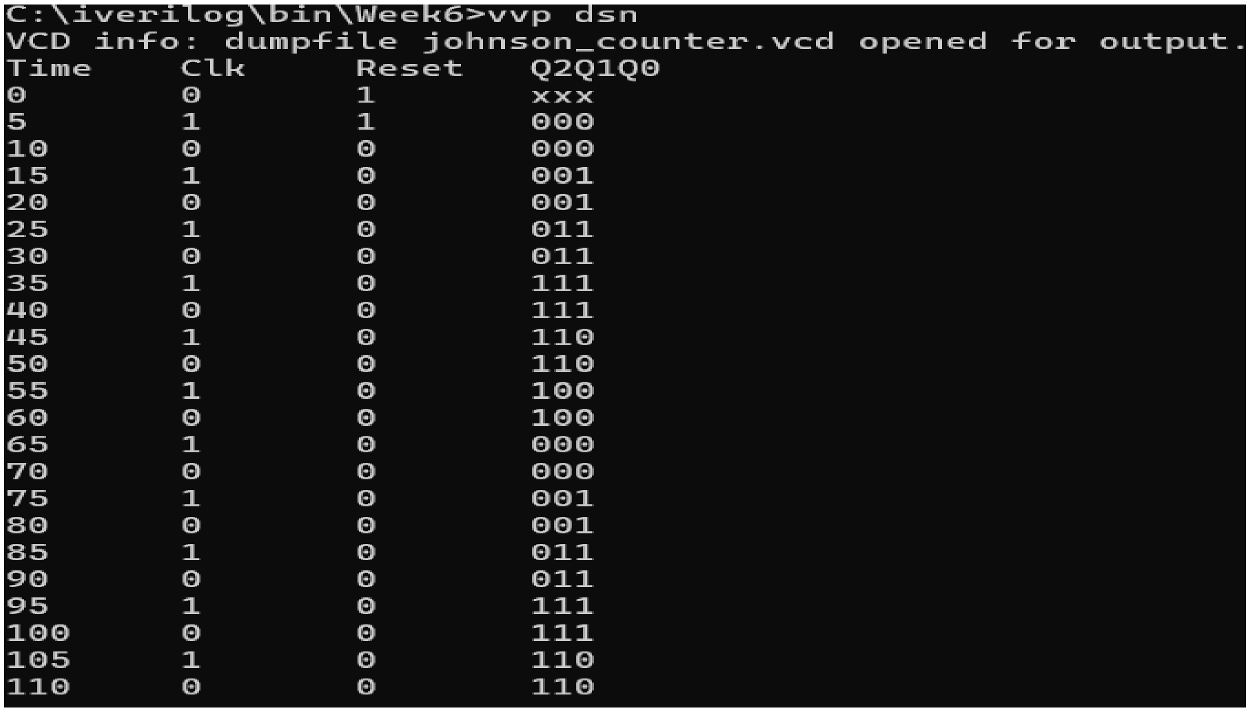
```
module dff(input clk, input reset, input d, output reg q);
    always @(posedge clk) begin
        if (reset)
            q <= 1'b0;
        else
            q <= d;
    end
endmodule

module johnson_counter_3bit(input clk, input reset, output [2:0] q);
    wire [2:0] d;

    assign d[0] = ~q[2];
    assign d[1] = q[0];
    assign d[2] = q[1];

    dff dff0 (.clk(clk), .reset(reset), .d(d[0]), .q(q[0]));
    dff dff1 (.clk(clk), .reset(reset), .d(d[1]), .q(q[1]));
    dff dff2 (.clk(clk), .reset(reset), .d(d[2]), .q(q[2]));
endmodule
```

Test Bench File



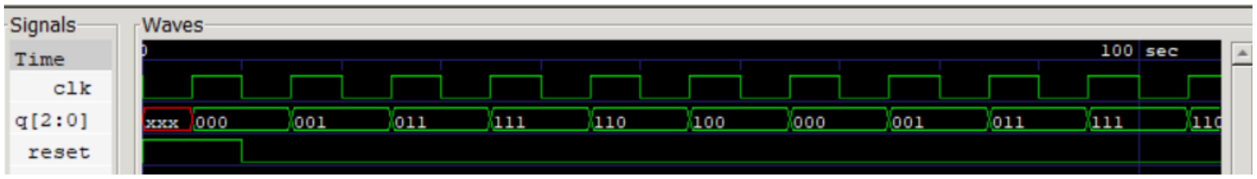
```
module tb_johnson_counter_3bit;
    reg clk, reset;
    wire [2:0] q;

    johnson_counter_3bit uut(.clk(clk), .reset(reset), .q(q));

    initial clk = 0;
    always #5 clk = ~clk;  // 10 time unit period

    initial begin
        $dumpfile("johnson_counter.vcd");
        $dumpvars(0, tb_johnson_counter_3bit);

        $display("Time\tClk\tReset\tQ2Q1Q0");
        $monitor("%0t\t%b\t%b\t%b%b%b", $time, clk, reset, q[2], q[1], q[0]);

        reset = 1; #10;
        reset = 0;
        #100;
        $finish;
    end
endmodule
```

II VVP Output Screenshot



```
C:\iverilog\bin\Week6>vvp dsn
VCD info: dumpfile johnson_counter.vcd opened for output.
Time    Clk     Reset   Q2Q1Q0
0       0       1       xxx
5       1       1       000
10      0       0       000
15      1       0       001
20      0       0       001
25      1       0       011
30      0       0       011
35      1       0       111
40      0       0       111
45      1       0       110
50      0       0       110
55      1       0       100
60      0       0       100
65      1       0       000
70      0       0       000
75      1       0       001
80      0       0       001
85      1       0       011
90      0       0       011
95      1       0       111
100     0       0       111
105     1       0       110
110     0       0       110
```

III GTKwave output screenshot



IV Table

The counter is initialized to an all-zero state.

Current State			D inputs			Next State		
Q2	Q1	Q0	D2	D1	D0	Q2	Q1	Q0
0	0	0	0	0	1	0	0	1
0	0	1	0	1	1	0	1	1
0	1	1	1	1	1	1	1	1
1	1	1	1	1	0	1	1	0
1	1	0	1	0	0	1	0	0
1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	1