Digital Design and Computer Organisation Laboratory

3rd Semester, Academic Year 2025

Date:29-09-2025

Name: Prajwal M	SRN:PES2UG24CS910	Section:C

Week Number: 6 Program Number: 6-11

1. **Design and implement 2 bit up synchronous counter using JK flipflop**

counter_2bit_up_sync_jk.v U

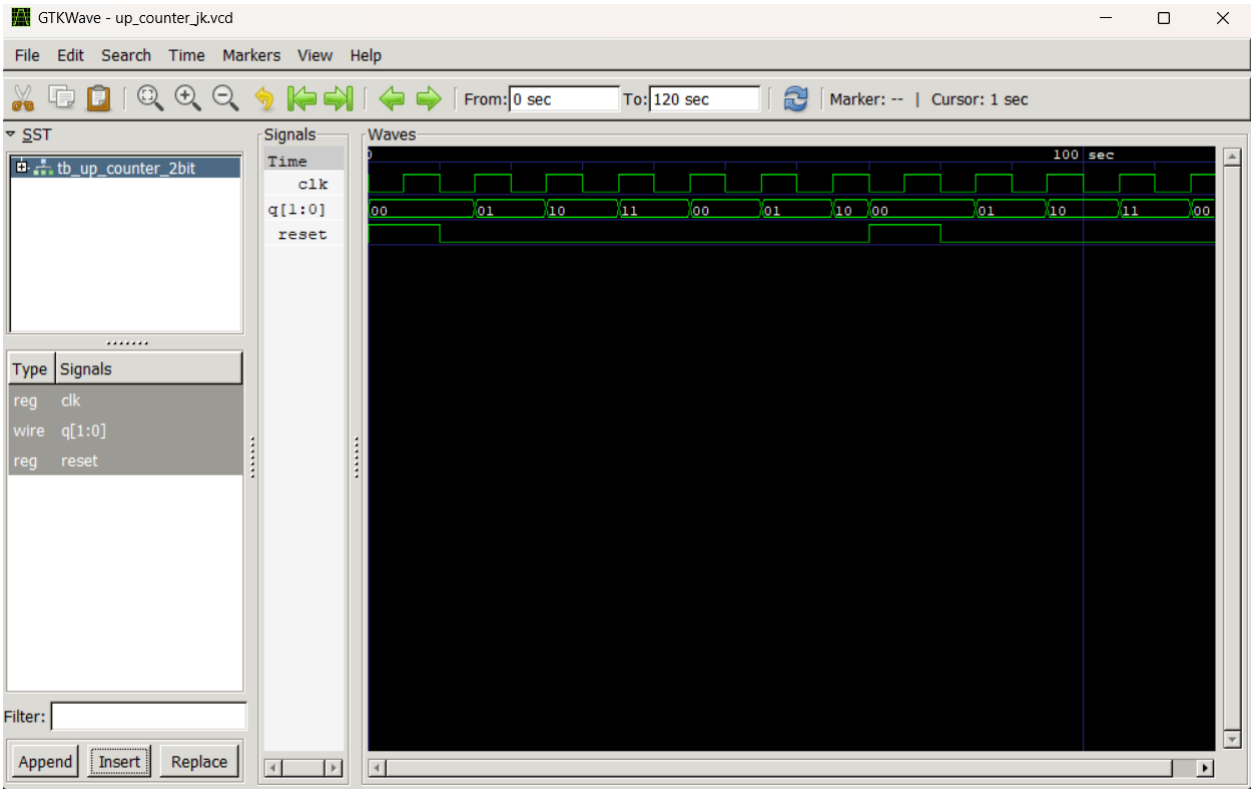
counter_2bit_up_sync_jk.v

```
module jk_ff (input wire j, input wire k, input wire clk, input wire rst, output reg q);
always @(posedge clk or posedge rst)
begin
 if (rst)
     q <= 1'b0;                // Reset to 0
 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

// 2-bit synchronous up counter using JK flip-flops
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];
// Instantiate two JK flip-flops
jk_ff jkff0 (.clk(clk), .rst(reset), .j(j0), .k(k0), .q(q[0]));
jk_ff jkff1 (.clk(clk), .rst(reset), .j(j1), .k(k1), .q(q[1]));
endmodule
```

upcounter_tb.v

```v
module tb_up_counter_2bit;
reg clk, reset;
wire [1:0] q;
// Instantiate counter
up_counter_2bit uut (.clk(clk),
.reset(reset), .q(q));
// Clock generation
initial clk = 0;
always #5 clk = ~clk;
// Clock period = 10
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;
#60; // Run counter for a while
reset = 1; #10; // Reset again
reset = 0;
#40;
$finish;
end
endmodule
```

```
D:\iverilog lab\week 7>iverilog -o upcounter counter_2bit_up_sync_jk.v upcounter_tb.v

D:\iverilog lab\week 7>vvp upcounter
VCD info: dumpfile up_counter_jk.vcd opened for output.
Time    Clk     Reset   Q1Q0
0       0       1       00
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       00
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

D:\iverilog lab\week 7>gtkwave up_counter_jk.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[120] end time.
```


GTKWave - up_counter_jk.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 120 sec
Marker: -- | Cursor: 1 sec
SST
tb_up_counter_2bit
Type Signals
reg clk
wire q[1:0]
reg reset
Filter:
Append Insert Replace
Signals
Time
clk
q[1:0]
reset
Waves
100 sec
00 01 10 11 00 01 10 00 01 10 11 00

2.Design and implement 2 bit down synchronous counter using JK flipflop

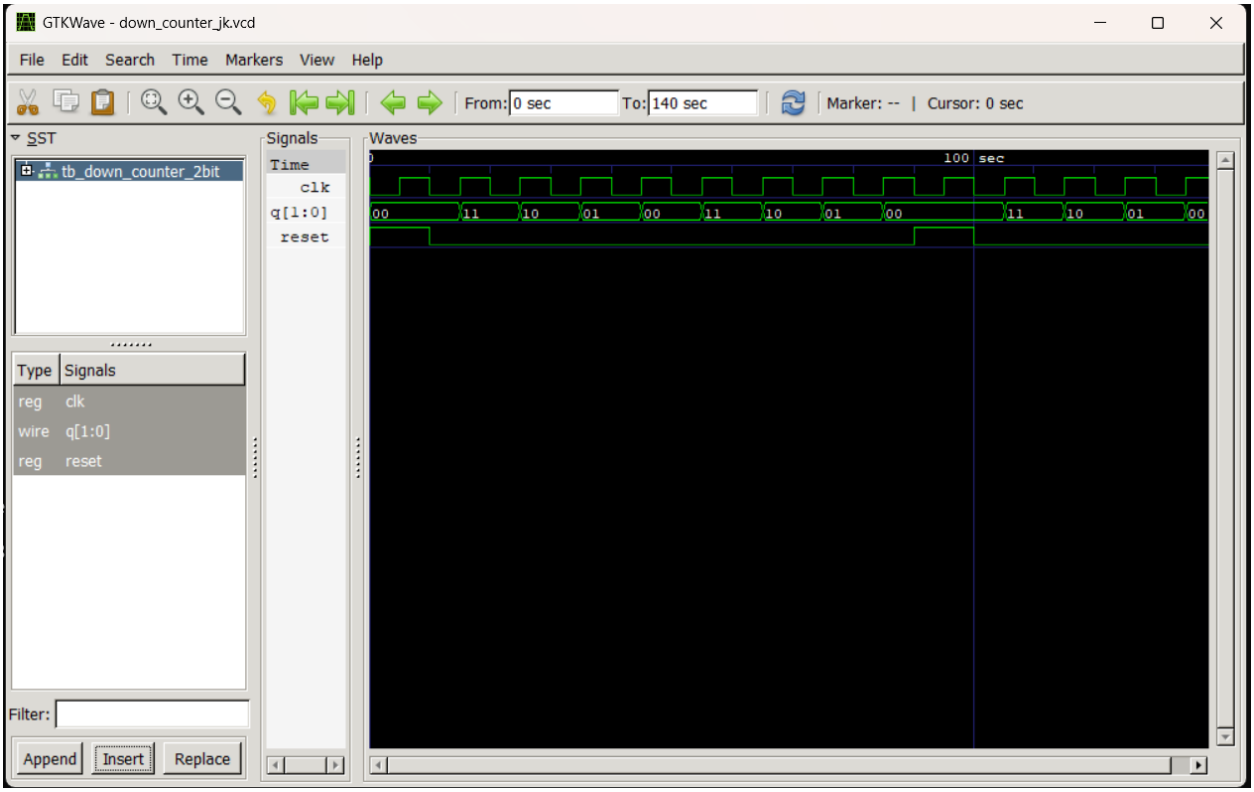
counter_2bit_down_sync_jk.v

```v
module jk_ff (input wire j, input wire k, input wire clk, input wire rst, output reg q);
always @(posedge clk or posedge rst)
begin
 if (rst)
     q <= 1'b0;                // Reset to 0
 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

// 2-bit synchronous down counter using JK flip-flops
module down_counter_2bit ( input clk, input reset, output [1:0] q);
// JK inputs
wire j0, k0, j1, k1;
assign j0=1'b1;
assign k0=1'b1;
assign j1=~q[0];
assign k1=~q[0];
// Instantiate flip-flops
jk_ff jkff0(.clk(clk), .rst(reset), .j(j0), .k(k0), .q(q[0]));
jk_ff jkff1(.clk(clk), .rst(reset), .j(j1), .k(k1), .q(q[1]));
endmodule
```

downcounter_tb.v

```v
module tb_down_counter_2bit;
reg clk, reset; wire [1:0] q;
// Instantiate counter
down_counter_2bit uut (.clk(clk),
.reset(reset), .q(q));
// Clock generation
initial begin
clk = 0;
forever #5 clk = ~clk; // Clock period = 10
end
initial begin
$dumpfile("down_counter_jk.vcd");
$dumpvars(0, tb_down_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; #80; // Run counter for a while
reset = 1; #10; // Reset again
reset = 0; #40;
$finish;
end
endmodule
```

```
D:\iverilog lab\week 7>iverilog -o downcounter counter_2bit_down_sync_jk.v downcounter_tb.v

D:\iverilog lab\week 7>vvp downcounter
VCD info: dumpfile down_counter_jk.vcd opened for output.
Time     Clk      Reset    Q1Q0
0        0        1        00
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        0        10
75       1        0        01
80       0        0        01
85       1        0        00
90       0        1        00
95       1        1        00
100      0        0        00
105      1        0        11
110      0        0        11
115      1        0        10
120      0        0        10
125      1        0        01
130      0        0        01
135      1        0        00
140      0        0        00

D:\iverilog lab\week 7>gtkwave down_counter_jk.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
```



3.Design and implement 2 bit up down synchronous counter using JK flipflop

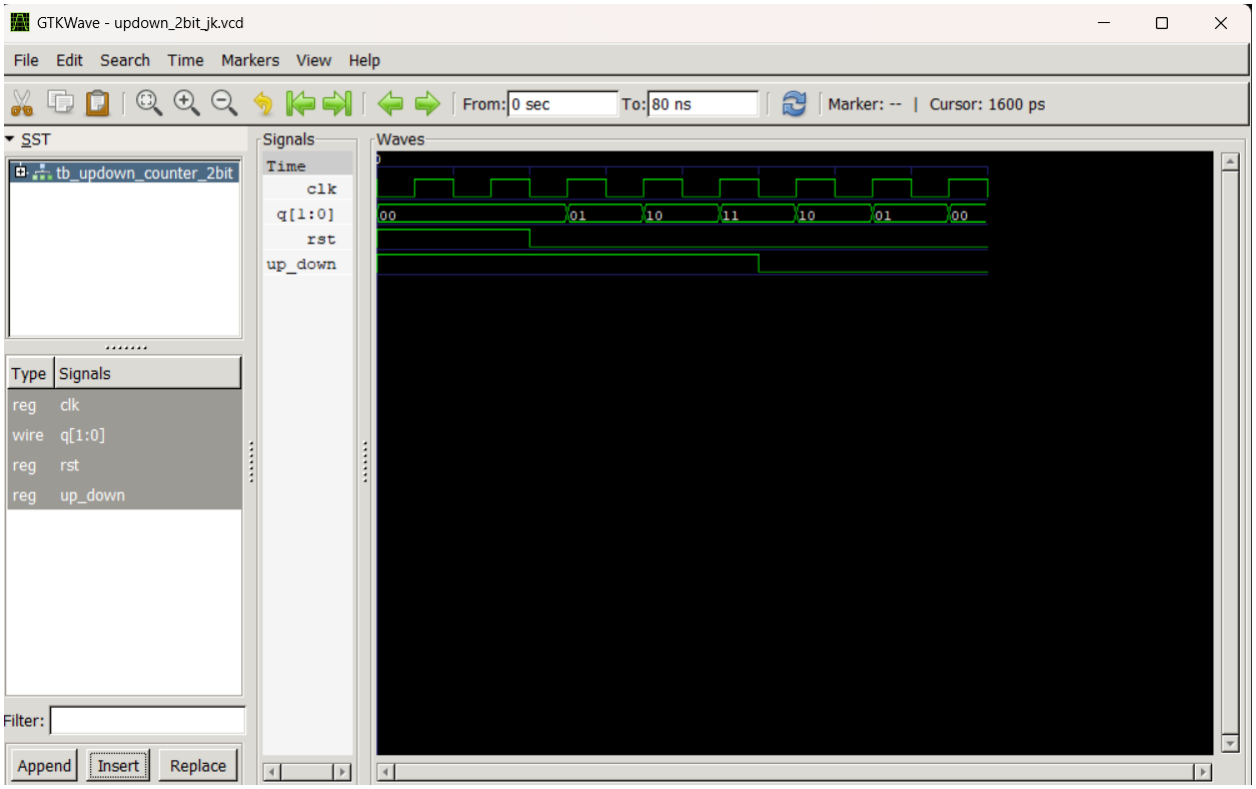
counter_2bit_updown_sync_jk.v

```v
module jk_ff (input wire j, input wire k, input wire clk, input wire rst, output reg q);
always @(posedge clk or posedge rst)
begin
 if (rst)
     q <= 1'b0;                // Reset to 0
 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 updown_counter_2bit ( input wire clk, input wire rst, input wire up_down, // 1: up, 0: do
output wire [1:0] q);
wire j0, k0, j1, k1;
// J0, K0 - Q0 always toggles
assign j0=1'b1;
assign k0=1'b1;
// J1, K1 - Q1 toggles based on direction
assign j1=up_down ? q[0] : ~q[0];
assign k1=up_down ? q[0] : ~q[0];
// Flip-flops instantiation
jk_ff jkff0(.clk(clk), .rst(rst), .j(j0), .k(k0), .q(q[0]));
jk_ff jkff1(.clk(clk), .rst(rst), .j(j1), .k(k1), .q(q[1]));
endmodule
```

updowncounter_tb.v

```
`timescale 1ns/1ps
module tb_updown_counter_2bit;
reg clk = 0;
reg rst = 1;
reg up_down = 1;
wire [1:0] q;
updown_counter_2bit dut (.clk(clk), .rst(rst), .up_down(up_down), .q(q));
// 10 ns period clock
always #5 clk = ~clk;
// Print after each synchronous update
always @(posedge clk) begin
$display("%0t clk=%0b rst=%0b upDn=%0b Q=%0b%0b",
$time, clk, rst, up_down, q[1], q[0]);
end
initial begin
$dumpfile("updown_2bit_jk.vcd");
$dumpvars(0, tb_updown_counter_2bit);
// Keep reset high for one edge -> Q becomes 00 at first posedge
@(negedge clk); rst = 1;
@(negedge clk); rst = 0; // release just before a posedge
// Count UP to reach 11: 00 -> 01 -> 10 -> 11
up_down = 1;
repeat (3) @(negedge clk); // change only on negedge
// Now count DOWN: 11 -> 10 -> 01 -> 00
up_down = 0;
repeat (3) @(negedge clk);
$finish;
end
endmodule
```

```
D:\iverilog lab\week 7>iverilog -o updown counter_2bit_updown_sync_jk.v updowncounter_tb.v

D:\iverilog lab\week 7>vvp updown
VCD info: dumpfile updown_2bit_jk.vcd opened for output.
5000 clk=1 rst=1 upDn=1 Q=00
15000 clk=1 rst=1 upDn=1 Q=00
25000 clk=1 rst=0 upDn=1 Q=00
35000 clk=1 rst=0 upDn=1 Q=01
45000 clk=1 rst=0 upDn=1 Q=10
55000 clk=1 rst=0 upDn=0 Q=11
65000 clk=1 rst=0 upDn=0 Q=10
75000 clk=1 rst=0 upDn=0 Q=01

D:\iverilog lab\week 7>gtkwave updown_2bit_jk.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[80000] end time.
```



Assignment:

1. Implement 3 bit ring counter using D flipflop

ring_counter_3bit.v

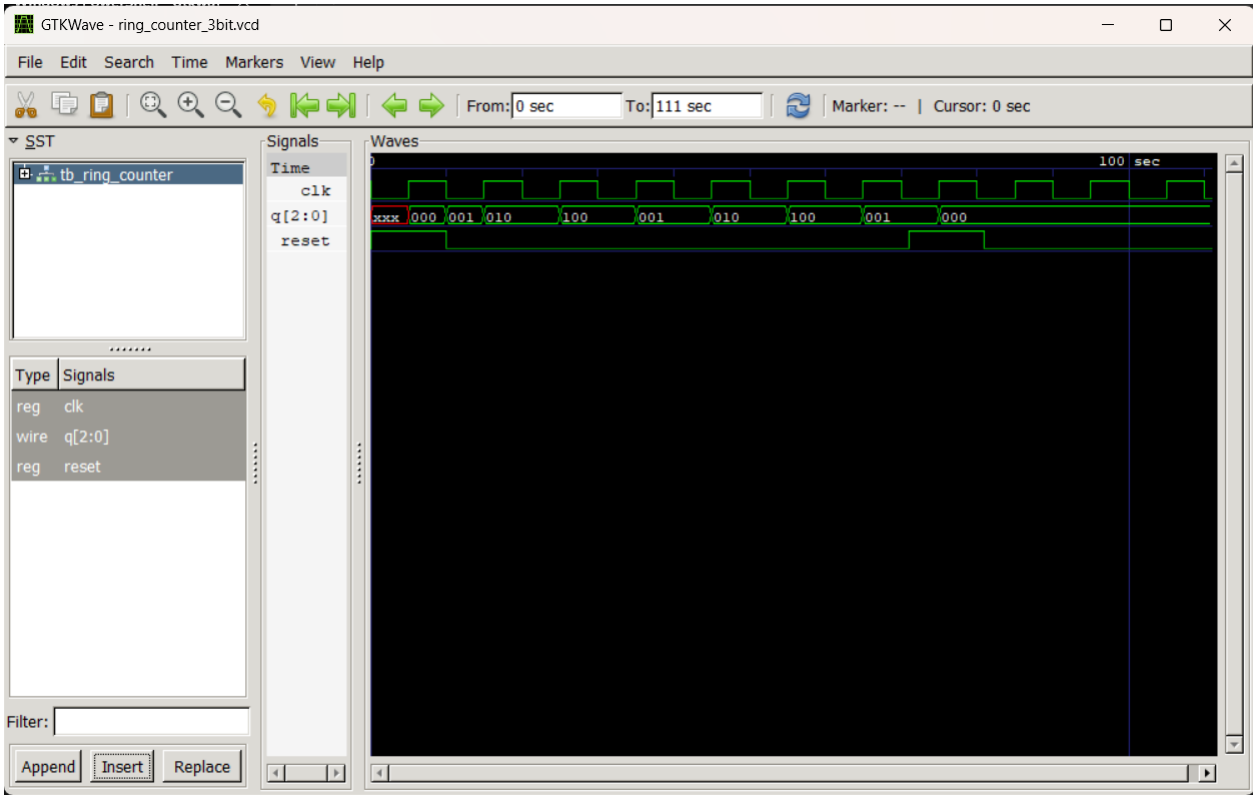
```
1   module d_ff (input wire D, input wire clk, input wire rst, output reg Q);
2   always @(posedge clk)
3   begin
4     if (rst)      Q<=1'b0;     // Asynchronous reset to 0
5    else      Q<=D;        // Sample D on rising edge of clock
6   end
7   endmodule
8
9   // 3-bit Ring Counter using D Flip-flop
10  module ring_counter_3bit(input clk, input reset, output [2:0] q);
11
12  // Ring counter: 001->010->100->001...
13  // Each D input gets the previous Q output (circular shift)
14  d_ff dff0(.D(q[2]), .clk(clk), .rst(reset), .Q(q[0]));
15  d_ff dff1(.D(q[0]), .clk(clk), .rst(reset), .Q(q[1]));
16  d_ff dff2(.D(q[1]), .clk(clk), .rst(reset), .Q(q[2]));
17
18  endmodule
```

ring_counter_tb.v

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

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

initial begin
clk = 0;
forever #5 clk = ~clk;
end

initial begin
$dumpfile("ring_counter_3bit.vcd");
$dumpvars(0, tb_ring_counter);
$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;
// Force initial state to 001 for ring counter
force uut.dff0.Q = 1'b1;
force uut.dff1.Q = 1'b0;
force uut.dff2.Q = 1'b0;
#1;
release uut.dff0.Q;
release uut.dff1.Q;
release uut.dff2.Q;
#60; // Run for 6 cycles to see full ring pattern
reset = 1; #10;
reset = 0; #30;
$finish;
end
endmodule
```

Do y

```
D:\iverilog lab\week 7>iverilog -o ring ring_counter_3bit.v ring_counter_tb.v

D:\iverilog lab\week 7>vvp ring
VCD info: dumpfile ring_counter_3bit.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
71       0        1        001
75       1        1        000
80       0        1        000
81       0        0        000
85       1        0        000
90       0        0        000
95       1        0        000
100      0        0        000
105      1        0        000
110      0        0        000

D:\iverilog lab\week 7>gtkwave ring_counter_3bit.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[111] end time.
WM Destroy

D:\iverilog lab\week 7>
```



2. Implement 3 bit Johnson counter using D flipflop

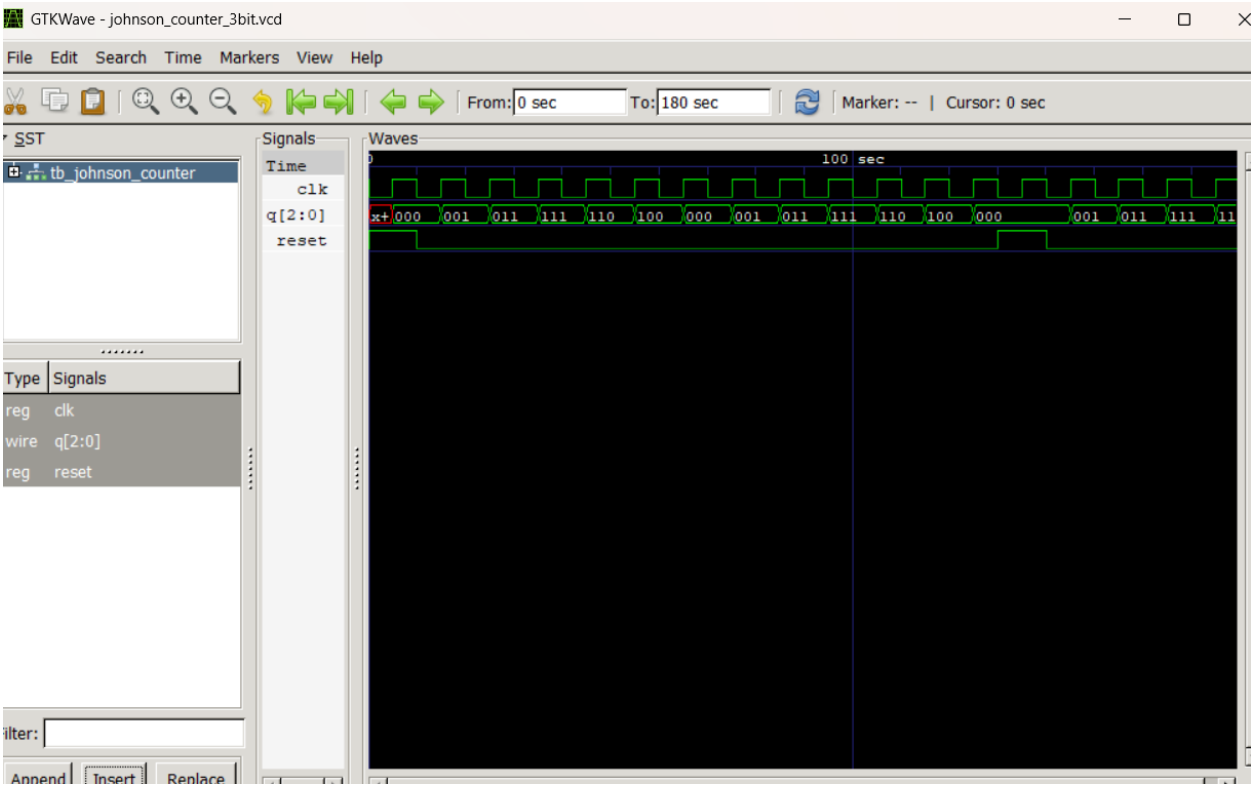
johnson_counter_3bit.v

```
D:\iverilog lab\week 7\johnson_counter_3bit.v • Untracked
, input wire rst, output reg Q);
always @(posedge clk)
begin
  if (rst)      Q<=1'b0;     // Asynchronous reset to 0
 else      Q<=D;        // Sample D on rising edge of clock
end
endmodule

// 3-bit Johnson Counter using D Flip-flop
module johnson_counter_3bit(input clk, input reset, output [2:0] q);

// Johnson counter: 000->001->011->111->110->100->000...
// Each D input gets the previous Q, except first gets inverted last Q
d_ff dff0(.D(~q[2]), .clk(clk), .rst(reset), .Q(q[0]));
d_ff dff1(.D(q[0]), .clk(clk), .rst(reset), .Q(q[1]));
d_ff dff2(.D(q[1]), .clk(clk), .rst(reset), .Q(q[2]));

endmodule
```

johnson_counter_tb.v

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

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

initial begin
clk = 0;
forever #5 clk = ~clk;
end

initial begin
$dumpfile("johnson_counter_3bit.vcd");
$dumpvars(0, tb_johnson_counter);
$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; #120; // Run for 12 cycles to see full Johnson sequence
reset = 1; #10;
reset = 0; #40;
$finish;
end
endmodule
```

```
D:\iverilog lab\week 7>iverilog -o john johnson_counter_3bit.v johnson_counter_tb.v

D:\iverilog lab\week 7>vvp john
VCD info: dumpfile johnson_counter_3bit.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
115     1       0       100
120     0       0       100
125     1       0       000
130     0       1       000
135     1       1       000
140     0       0       000
145     1       0       001
150     0       0       001
155     1       0       011
160     0       0       011
165     1       0       111
170     0       0       111
175     1       0       110
180     0       0       110

D:\iverilog lab\week 7>gtkwave johnson_counter_3bit.vcd

GTKWave Analyzer v3.3.48 (w)1999-2013 BSI

[0] start time.
[180] end time.
```


GTKWave - johnson_counter_3bit.vcd
File Edit Search Time Markers View Help
From: 0 sec
To: 180 sec
Marker: -- | Cursor: 0 sec
SST
tb_johnson_counter
Type Signals
reg clk
wire q[2:0]
reg reset
Filter:
Append Insert Replace
Signals
Time
clk
q[2:0]
reset
Waves
100 sec
000 001 011 111 110 100 000 001 011 111 110 100 000 001 011 111