30 Days of RTL Coding

-By T Tushar Shenoy

Day 22

Problem Statement: Implementing 4-bit synchronous up/down counter in Structural Style of Modelling.

Theory:

The counters which use clock signal to change their transition are called "Synchronous counters". This means the synchronous counters depends on their clock input to change state values. In synchronous counters, all flip flops are connected to the same clock signal and all flip flops will trigger at the same time. The up and down counters can be implemented in a single counter called up/down counter. This can be selected from its input. The design of up/ down counter with JK flip flops is shown below.

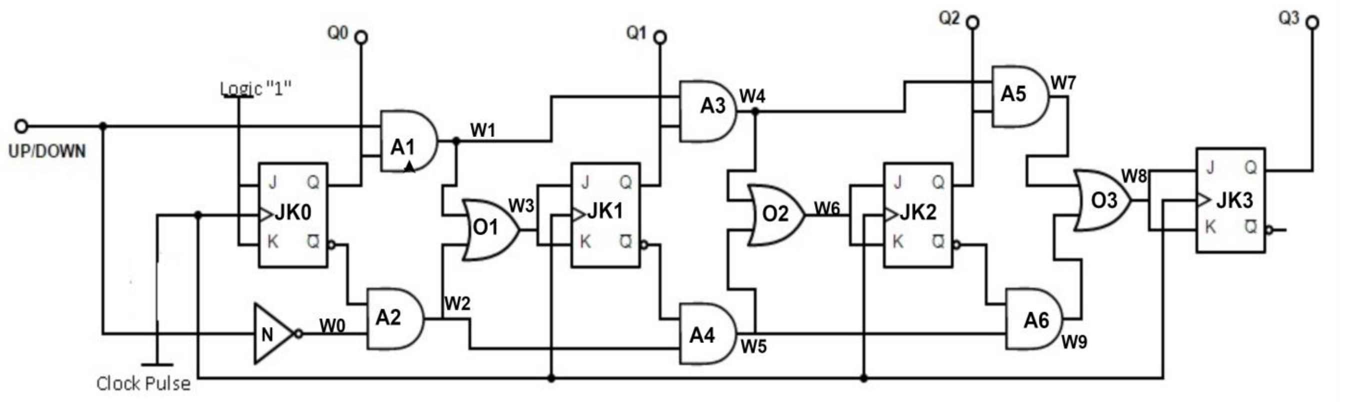


FIG: Block Diagram of UP/DOWN Counter using JK FF

The up/ down counter has "Up" and "Down" count modes by having 2 input AND gates, which are used to detect the appropriate bit conditions for counting operation. OR gates are used to combine the outputs of AND gate, from each JK flip flop. We provide a up/ down control line which enables upper or lower series of AND gates to pass the outputs of JK flip flops, Q , Q' to the next stage of flip flop, in the cascaded arrangement.

If the up /down control line is set to HIGH, then the top AND gates are in enable state and the circuit acts as UP counter. If the up /down control line is set to low, then the bottom AND gates are in enable state and the circuit acts as DOWN counter.

Verilog Code:

```
//JK Flip Flop Code Using Behavioural modelling
module JK_Flip_Flop(J,K,clk,reset,Q,Qb);

input J,K,clk,reset;
output reg Q,Qb;

always@(posedge clk,posedge reset)
begin
if(reset)
   Q=1'b0;
   else
   begin
        if(J==1'b0&&K==1'b0)
            Q=Q;
        else if(J==1'b0&&K==1'b1)
            Q=1'b0;
        else if(J==1'b1&&K==1'b0)
            Q=1'b1;
        else
            Q=~Q;
    end
   Qb<=~Q;
end

endmodule
```

```
//Verilog Code for 4 Bit UP/DOWN Counter
module up_down_counter(ud,reset,clk,Q);

input ud,reset,clk;
output [3:0]Q;

wire [3:0]Qb;
wire [9:0]w;

JK_Flip_Flop
JK0(.J(1'b1),.K(1'b1),.clk(clk),.reset(reset),.Q(Q[0]),.Qb(Qb[0]));
not N(w[0],ud);
and A1(w[1],ud,Q[0]);
and A2(w[2],Qb[0],w[0]);
or O1(w[3],w[1],w[2]);

JK_Flip_Flop
JK1(.J(w[3]),.K(w[3]),.clk(clk),.reset(reset),.Q(Q[1]),.Qb(Qb[1]));
and A3(w[4],w[1],Q[1]);
and A4(w[5],Qb[1],w[2]);
or O2(w[6],w[4],w[5]);

JK_Flip_Flop
JK2(.J(w[6]),.K(w[6]),.clk(clk),.reset(reset),.Q(Q[2]),.Qb(Qb[2]));
and A5(w[7],w[4],Q[2]);
and A6(w[9],Qb[2],w[5]);
or O3(w[8],w[7],w[9]);
```

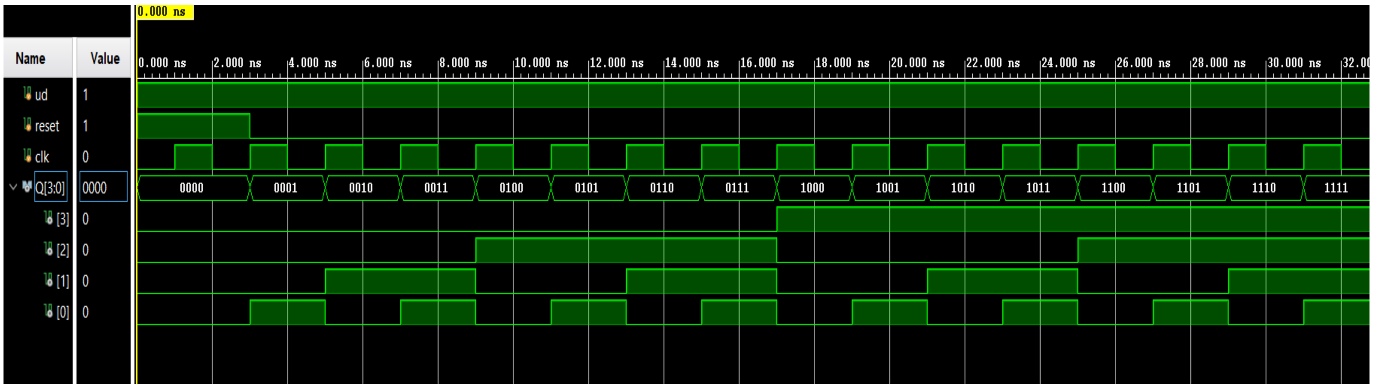
```
JK_Flip_Flop
JK3(.J(w[8]),.K(w[8]),.clk(clk),.reset(reset),.Q(Q[3]),.Qb(Qb[3]));
endmodule
```

Testbench Code:

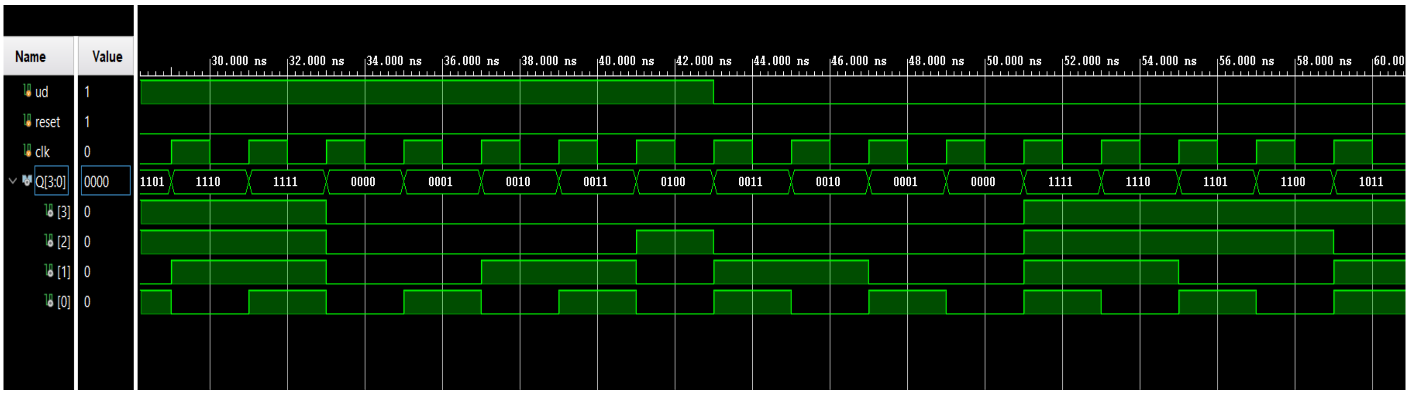
```
///Testbench for 4 Bit UP/DOWN Counter
module up_down_counter_tb();

reg ud,reset,clk;
wire [3:0]Q;

up_down_counter dut(.ud(ud),.reset(reset),.clk(clk),.Q(Q));

initial begin
ud=1'b1; //Up Counting
clk=1'b0;
reset=1'b1;
#3 reset=1'b0;
#40 ud=1'b0; //Down Counting
#40 $finish;
end
//The Above Code can be manipulated to obtain Up count from 0 to 255 and
Down count from 255 to 0
always #1 clk=~clk;
endmodule
```

Simulation Output:

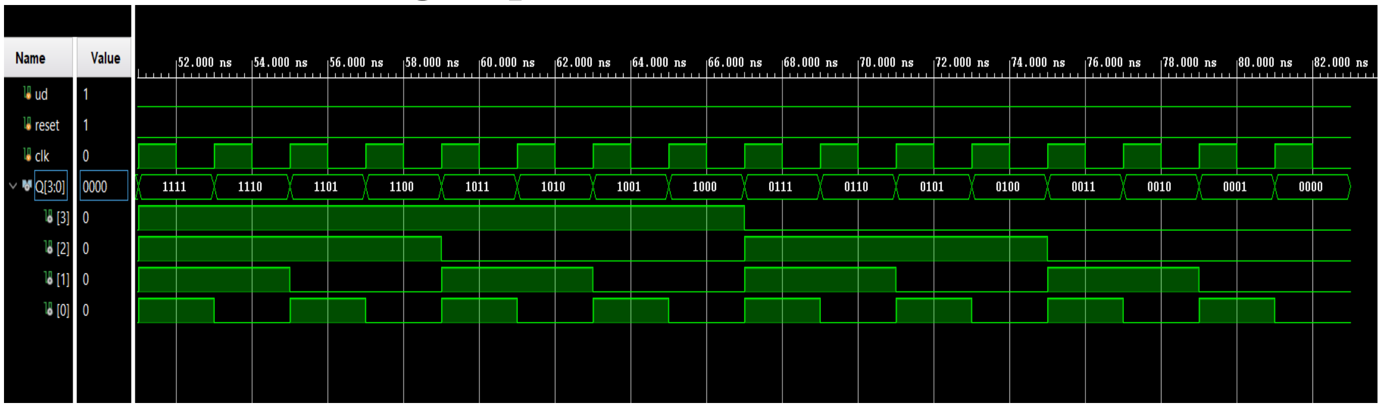
- **UP Counting Sequence, ud=1**



- **Control signal ud changes to 0,Down Counting starts**



- **DOWN Counting Sequence, ud=0**



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-22-Up-Down-Counter-Structural