30 Days of RTL Coding

-By T Tushar Shenoy

Day 28

Problem Statement: Implementing 8 BIT Even Down Counter in Structural Style.

Theory:

An 8-bit even down counter is a sequential circuit that counts down from 254 to 0 in even decrements. It is implemented using 8 D flip-flops, with the Q' output of each flip-flop connected to the D input of the same flip-flop. First D Flip-Flop is the LSB bit and the last D Flip-Flop is the MSB bit. The 8 bit even down counter can be used to count the number of even pulses in a signal, to measure the duration of an even interval, or to generate a sequence of even numbers. The counter can be implemented using a variety of logic gates, but it is most implemented using D flip-flops. This is because D flip-flops are easy to implement and they are very reliable. To test the functionality of the counter, OOP (Object Oriented Programming) based test bench is designed. The 8-bit even down counter consists of eight flip-flops, with each flip-flop representing one bit of the counter. These flip-flops are connected in a cascading fashion, with the output of each flip-flop being connected to the clock input of the next flip-flop in the sequence. When the counter receives a clock pulse, it decrements its current value by two, effectively counting down in even numbers.

For example, if the counter is initialized with a value of 10011010 in binary, it will sequence through the following states: 10011010→ 10011000→ 10010110→ 10010100→.....→00000010→0000000.

The 8-bit even down counter is a valuable component in digital systems, allowing for precise counting down in even numbers. Its versatility and ease of implementation make it a popular choice in various applications where controlled sequencing and timing are required. Whether in microcontrollers, digital signal processing, or control circuits, the even down counter plays a significant role in enhancing the efficiency and functionality of digital systems.

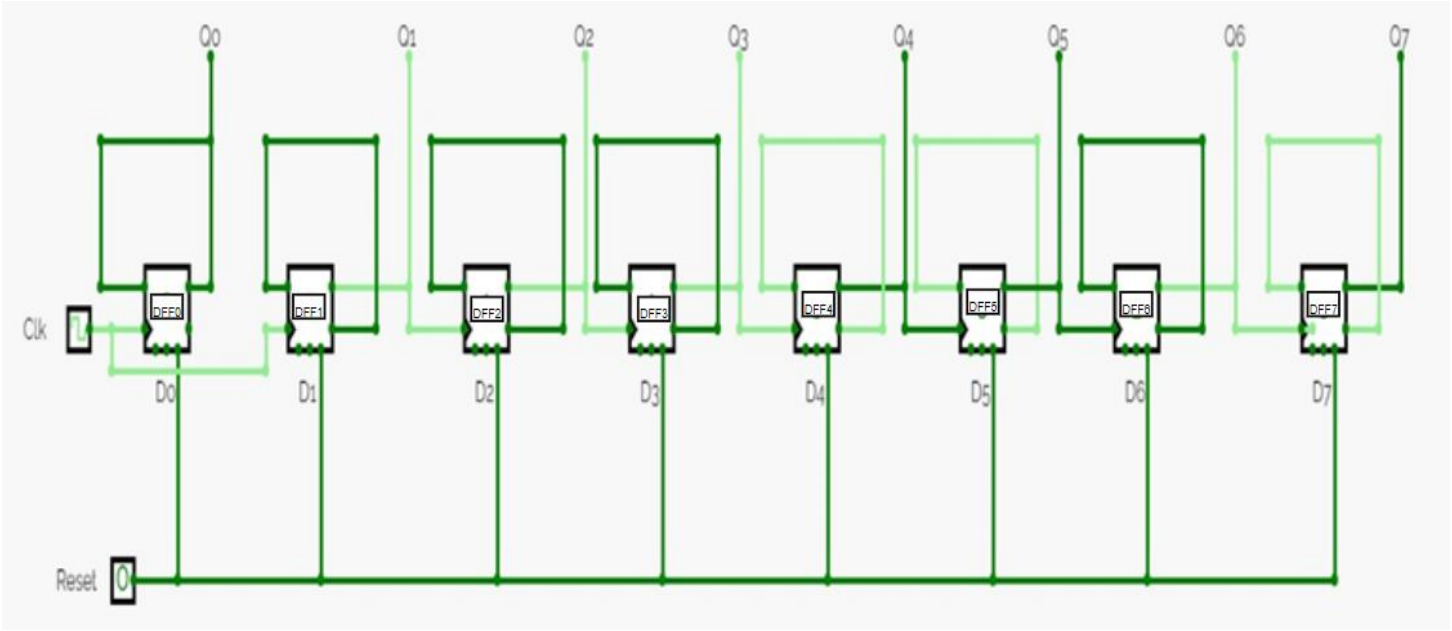


FIG: 8-bit even down counter

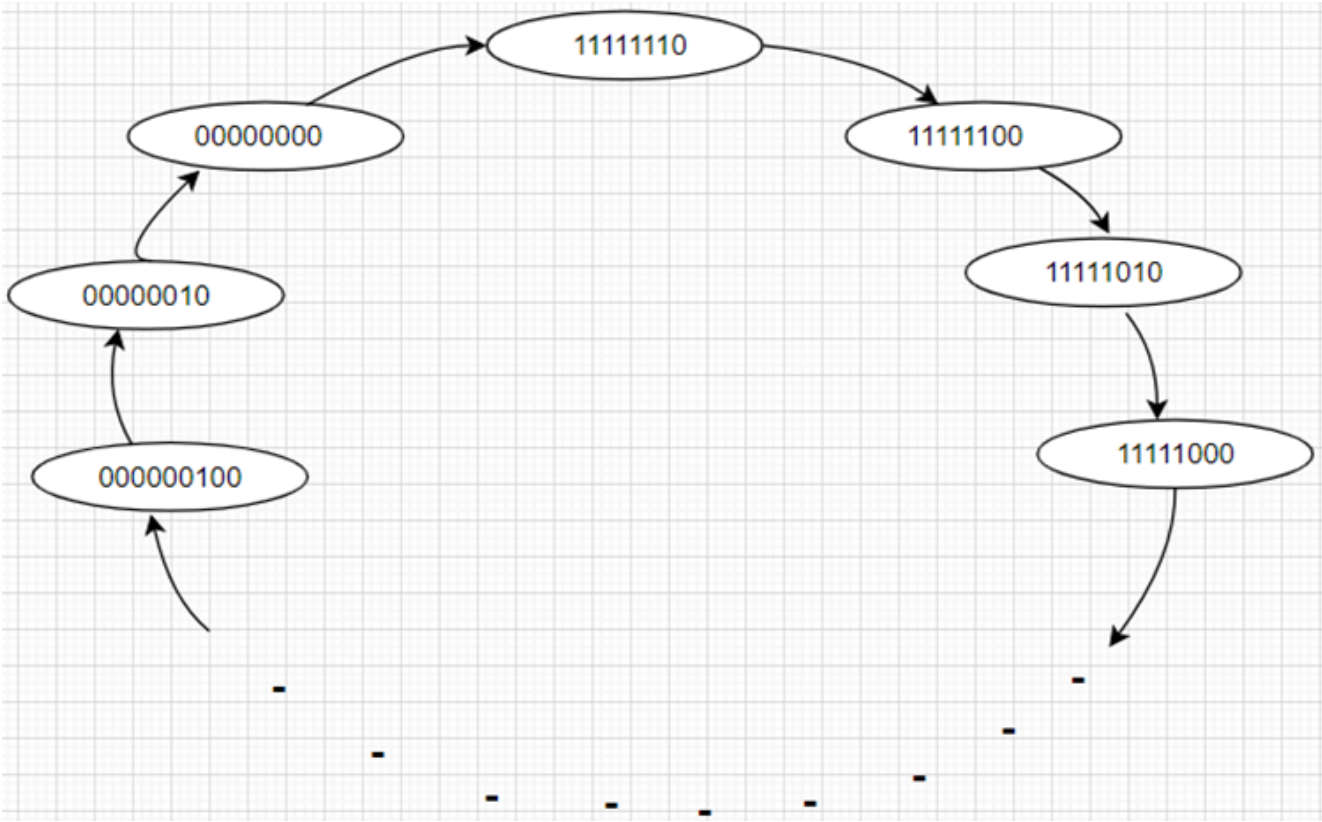


FIG: 8-BIT Even down counter state Diagram

Working:

8 bit even down counter will count numbers from 254 to 0 downwards. Since this is a 8 bit counter 8 D Flip-Flops are used. The D flip-flop is a type of sequential circuit with data input D and two outputs Q and Q'. Leftmost D Flip-Flop in block diagram is the LSB Bit and the rightmost D Flip-Flop is the MSB Bit. Clock and Reset are the inputs. Since this is an even counter the LSB bit should be zero so the output Q0 is given as data input. Clock is given to the second Flip-Flop and Q1' is given to data input of the same Flip-Flop. Q1 is given as clock for the next Flip-Flop. Hence it becomes asynchronous. Same process continues for the remaining Flip-Flop. Active high reset is given to the circuit and it is common for all the Flip-Flops. When the reset is high, output Q of all Flip-Flops goes low. When the reset goes low circuit starts operating as a down counter which utilizes D flip-flops as the fundamental building blocks to achieve this functionality. The output of first D Flip-Flop remains unchanged irrespective of the given reset whereas the output of other D Flip-Flops are changed with respect to positive edge of the clock given.

D Flip-flop truth table

Clock	D	Q+	$\bar{Q}$+
0	X	Q	$\bar{Q}$
↑	0	0	1
↑	1	1	0

8 BIT Even Down Counter Truth Table

Clk	Reset	Q7	Q6	Q5	Q4	Q3	Q2	Q1	Q0
⇧	1	0	0	0	0	0	0	0	0
⇧	0	1	1	1	1	1	1	1	0
⇧	0	1	1	1	1	1	1	0	0
⇧	0	1	1	1	1	1	0	1	0
-	-	-	-	-	-	-	-	-	0
-	-	-	-	-	-	-	-	-	0
-	-	-	-	-	-	-	-	-	0
-	-	-	-	-	-	-	-	-	0
-	-	-	-	-	-	-	-	-	0
⇧	0	0	0	0	0	0	0	1	0
⇧	0	0	0	0	0	0	0	0	0

Verilog Code:

```
//Verilog Code for D Flip Flop
module df(d,q,qb,clk,rst);
input d;
input rst;
input clk;
output q,qb;
reg q,qb;
    always@(posedge clk, posedge rst)
        begin
            if(rst)
                q=1'b0;
            else
                q=d;
                qb=~q;
            end
endmodule
```

```
//Verilog code for 8 Bit even Down Counter
module even_down_count(clk,rst,out,clkin);
input clk;
input rst;
input clkin;
wire [7:0]qb;
output [7:0]out;

df d0(.clk(clkin),.rst(rst),.d(out[0]),.q(out[0]),.qb(qb[0]));

df d1(.clk(clk),.rst(rst),.d(qb[1]),.q(out[1]),.qb(qb[1]));

df d2(.clk(out[1]),.rst(rst),.d(qb[2]),.q(out[2]),.qb(qb[2]));

df d3(.clk(out[2]),.rst(rst),.d(qb[3]),.q(out[3]),.qb(qb[3]));

df d4(.clk(out[3]),.rst(rst),.d(qb[4]),.q(out[4]),.qb(qb[4]));

df d5(.clk(out[4]),.rst(rst),.d(qb[5]),.q(out[5]),.qb(qb[5]));

df d6(.clk(out[5]),.rst(rst),.d(qb[6]),.q(out[6]),.qb(qb[6]));

df d7(.clk(out[6]),.rst(rst),.d(qb[7]),.q(out[7]),.qb(qb[7]));

endmodule
```

Testbench Code:

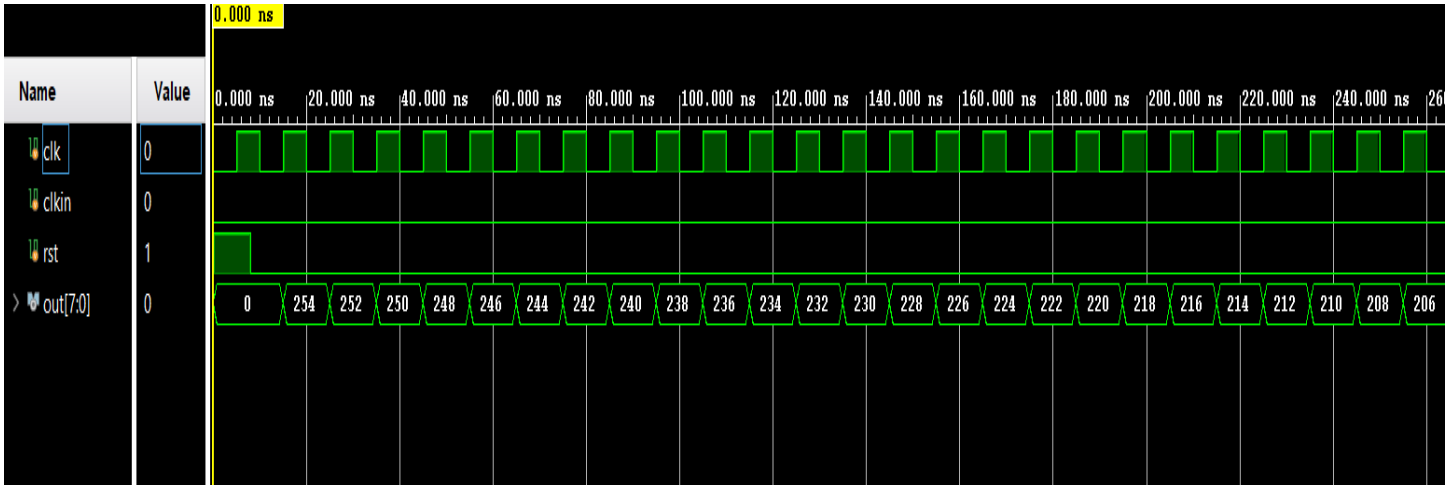
```
//TestBench for 8 Bit Even Down Counter
module Even_Down_Counter_tb();

reg clk,clkin,rst;
wire [7:0]out;

even_down_count dut(clk,rst,out,clkin);

initial begin
clkin=0;
clk=0;
rst=1;
#8 rst=0;
end
always #5 clk=~clk;
endmodule
```

Simulation Output:



GitHub Repository URL: https://github.com/tusharshenoy/RTL-Day-28-8-BIT-Even-Down-Counter