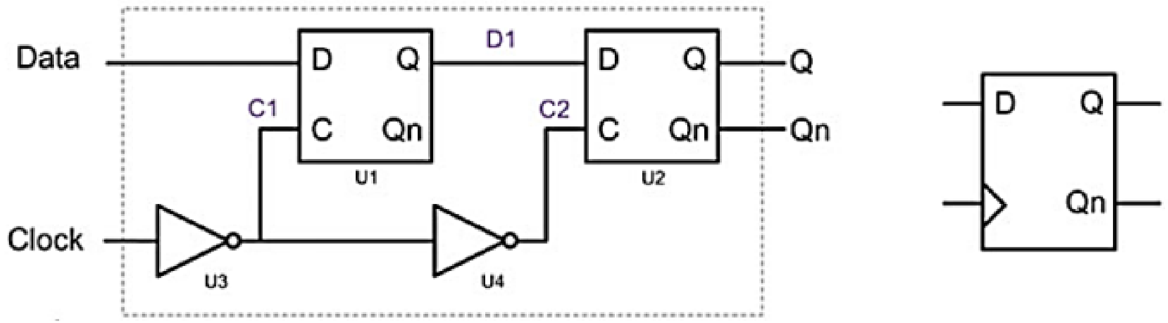
Professor Bhavnagarwala Anik Barua

CS 2204 - Section C 10-22-2021

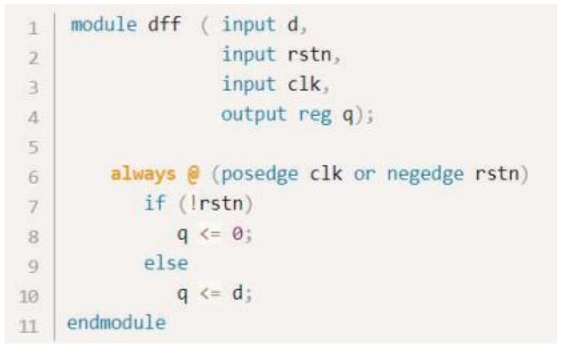
Lab 6: 3-Bit Ripple Counter using D Flip Flop

Today's lab is about building a 3-Bit Ripple Counter using D Flip Flop with a reset input and we will verify the correct operation by simulating a testbench. Before we talk about ripple counter, we need to understand the function of a D Flip Flop. D Flip Flop is one of the most widely used storage device in a sequential logic. D Flip Flop is basically a Flip Flop with one data input that stores the value of that input signal in its memory at the clock edges. It is similar to the D-Latch except that the store mode is triggered by a transition or the edge on the clock signal. With D Flip Flop, we can implement higher frequency system because the outputs are updated in a short amount of time and it restricts all the noises from corrupting the data stored by excluding all the input transitions at all other times.



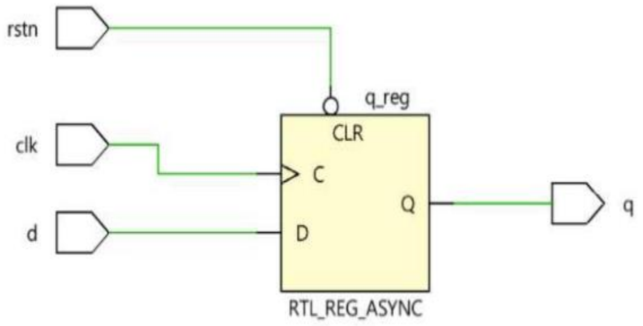
So in this lab first we will use this D Flip Flop to build a 3-Bit Ripple Counter with a reset input.

dff Module: From the given dff module we can see we have input d, reset, clk, and output reg q. Next, we are using an always block at to write the D Flip Flop function that states that at every positive edge of clock or negative reset, if the reset is not active low, we assign output to 0 else we assign the output to d which is our input. We can see from the dff diagram that our reset is active low, which means the Flip Flop goes to the reset state when the reset signal goes to a low level else we assign q to our input d.

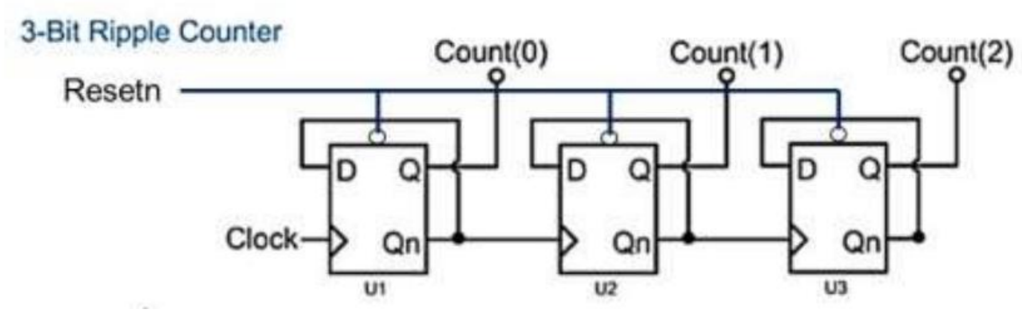


```
module dff  ( input d,
              input rstn,
              input clk,
              output reg q);

   always @ (posedge clk or negedge rstn)
      if (!rstn)
         q <= 0;
      else
         q <= d;
endmodule
```

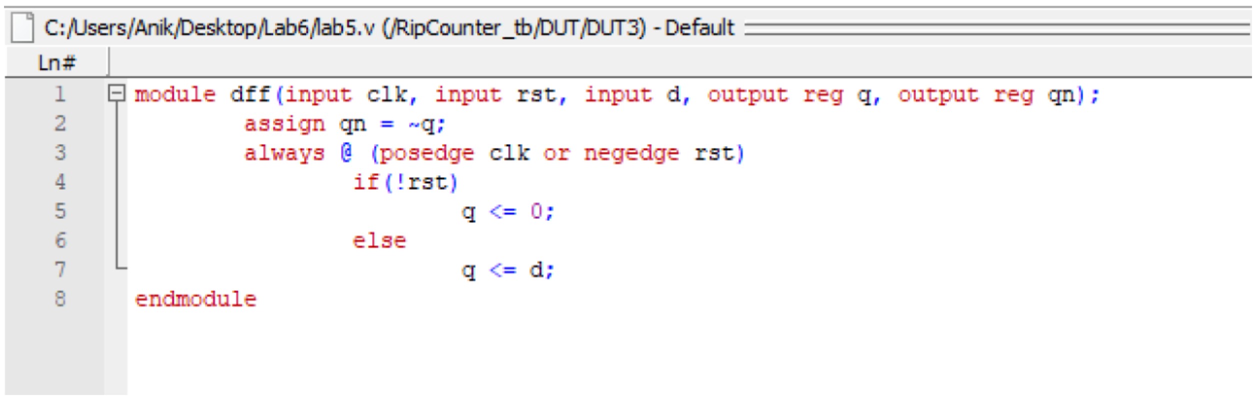


3 Bit Ripple Counter with 3 D-Flip Flop:



First we will write a Verilog module that modifies the given D Flip-Flop module by adding a complementary output Qn, which will be a clk value for next D Flip Flop. We can do this my adding a extra output Qn and assign it the negated value of q.

Modified dff:

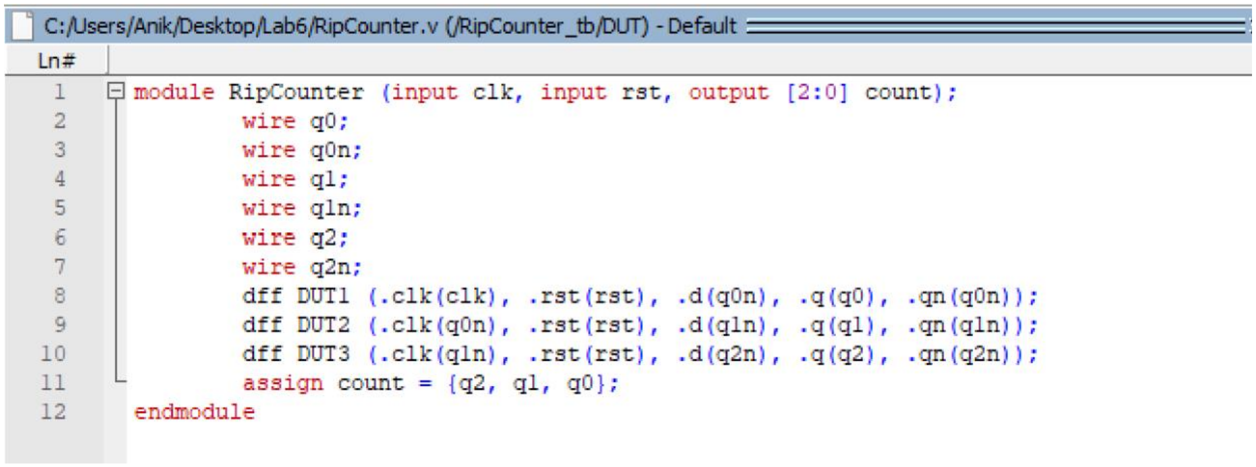


C:/Users/Anik/Desktop/Lab6/lab5.v (/RipCounter_tb/DUT/DUT3) - Default

```
module dff(input clk, input rst, input d, output reg q, output reg qn);
        assign qn = ~q;
        always @ (posedge clk or negedge rst)
                if(!rst)
                        q <= 0;
                else
                        q <= d;
endmodule
```

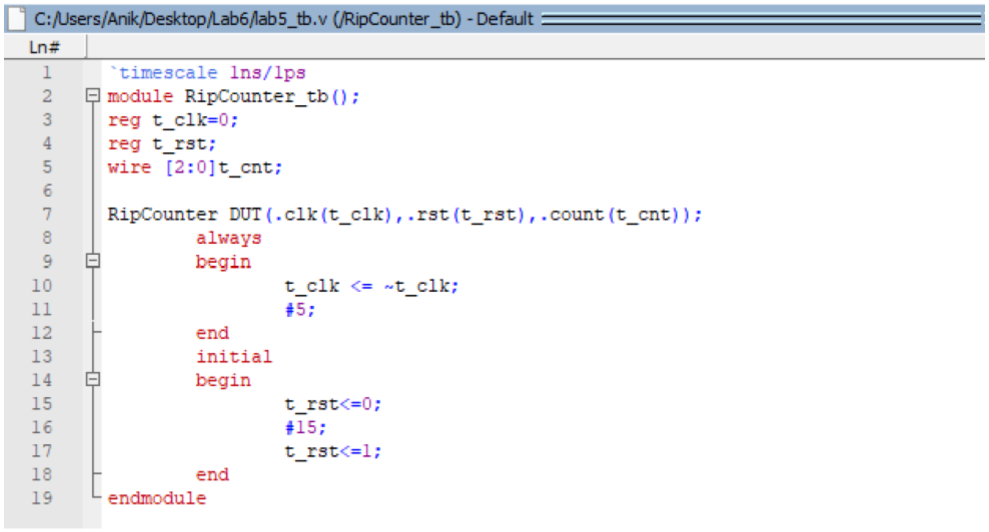
Next, we use this modified D Flip-Flop module to implement the schematic of a 3-bit ripple counter that loops back to ‘000’. First we have clk, rst as input and 3 counters as output. Since we need to create 3 dffs using the dff module we need 2 wires, q and qn for each dff. Then using the dff module, we create 3 dff’s by assigning the input and output according to the schemetic of 3 bit ripple counter. As we can see the first dff takes the clk, rst value but for d input it takes the value of q0n, and output as q0 and q0 bar (q0n). For the next dff the clk takes the value from the first dff q0n, rst stays the same, d takes q1n and q as q1 and qn as q1n. Next dff takes the clk value from the last dff q1n, rst same and d takes q2n and output q as q2 and qn as q2n. In the end, we assign count as the output of three dff outputs q2, q1, and q0.

Module for 3-bit Ripple Counter:


C:/Users/Anik/Desktop/Lab6/RipCounter.v (/RipCounter_tb/DUT) - Default

```
module RipCounter (input clk, input rst, output [2:0] count);
        wire q0;
        wire q0n;
        wire q1;
        wire q1n;
        wire q2;
        wire q2n;
        dff DUT1 (.clk(clk), .rst(rst), .d(q0n), .q(q0), .qn(q0n));
        dff DUT2 (.clk(q0n), .rst(rst), .d(q1n), .q(q1), .qn(q1n));
        dff DUT3 (.clk(q1n), .rst(rst), .d(q2n), .q(q2), .qn(q2n));
        assign count = {q2, q1, q0};
endmodule
```

Next, we will write the Verilog Testbench that that sweeps through enough Clock input pulses and we can verify 3 bit ripple counter is operating correctly. First we create the clk, rst, and 3 counters in our testbench. Then using the Ripple Counter module we map the values with its inputs and outputs. Then with the always block, we make the clock toggle (0 to 1 and 1 to 0) after every #5 seconds delay and the reset to begin with 0 and after #15 sec delay we assign it to 1. This gives us the testbench for our ripple counter and we can simulate it to get the waveforms.

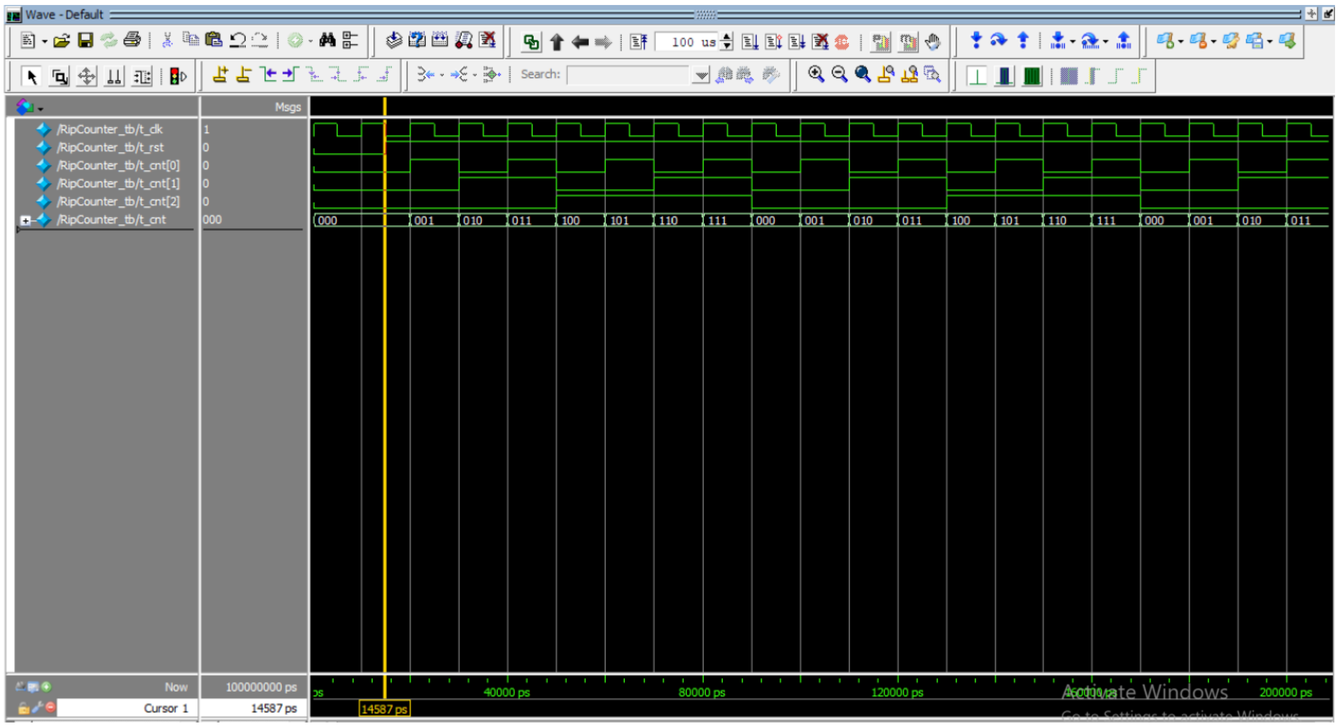


C:/Users/Anik/Desktop/Lab6/lab5_tb.v (/RipCounter_tb) - Default

```verilog
`timescale 1ns/1ps
module RipCounter_tb();
reg t_clk=0;
reg t_rst;
wire [2:0]t_cnt;

RipCounter DUT(.clk(t_clk),.rst(t_rst),.count(t_cnt));
        always
        begin
                t_clk <= ~t_clk;
                #5;
        end
        initial
        begin
                t_rst<=0;
                #15;
                t_rst<=1;
        end
endmodule
```

Waveforms Explanation:



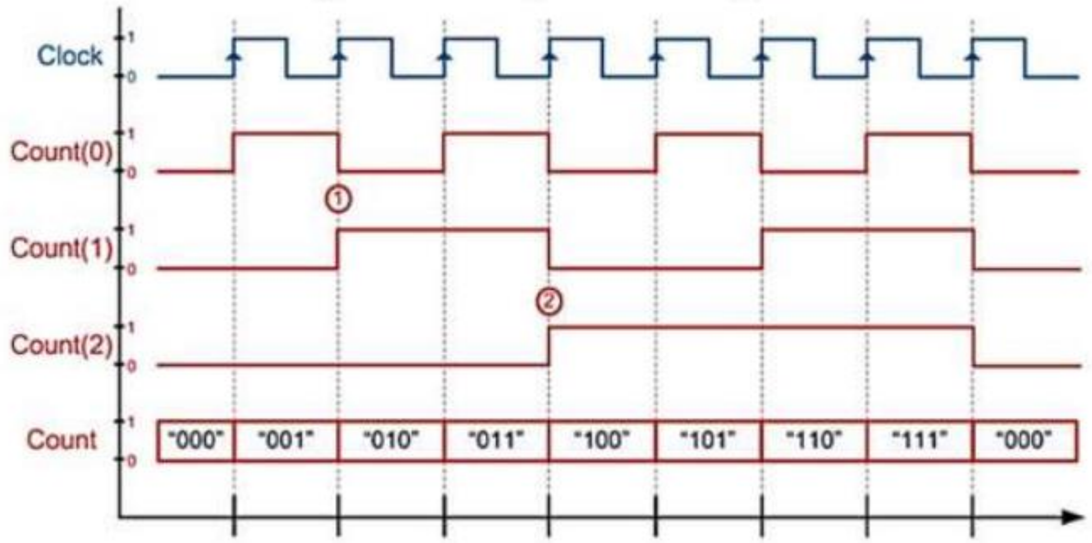
Wave - Default

```
/RipCounter_tb/t_clk      1
/RipCounter_tb/t_rst      0
/RipCounter_tb/t_cnt[0]   0
/RipCounter_tb/t_cnt[1]   0
/RipCounter_tb/t_cnt[2]   0
/RipCounter_tb/t_cnt      000

t_cnt: 000 001 010 011 100 101 110 111 000 001 010 011 100 101 110 111 000 001 010 011

Now        100000000 ps
Cursor 1   14587 ps
```

First we can see the counter works as it begins from 000 and increments by 1 until 111 and loops back to 000. As the reset goes to 1, at the positive edge of clock, the count(0) state changes to 1. Since we know D-Flip Flop only occurs at the positive edge of the clk, we can see even though clk goes to 0 the count stays to 1 until the next positive edge arrives and then it goes to 0. Next for count(1), since it takes the value of q_bar (Opposite of q) as it's clk value, at the count(0) negative edge, count(1) goes to 1 and until the next falling edge it stays at 1 and then goes back to 0. Then for count(2), it takes q1_bar as its clk value, so at the negative edge of count(2), it goes to 1 and until the next negative edge of count(1) it stays the same. We can see from the waveform that Count(2) is MSB and Count(0) is the LSB, and we get 000 when all of the count states are 0, and when count(0) goes to 1, we get count value as 001, then count(0) goes to 0 and count(1) goes to 1 and we get count value as 010. Following this same pattern, we get the count values increment to 111 and it loops back to 000 as the assertion of the reset brings the counter outputs to 000.



If we compare the waveforms with the given 3-bit Ripple Counter waveform in the lab assignment, we can see we get the same waveforms. So we have the correct operation for 3-bit Ripple counter.