ECEN 651 Lab Report1

Zhenlei Song 826009388

1. Objectives

- Environment setup
- Design and program in Verilog for a Ripple Carry Counter and a 4:1 Mux
- Setting testbench

2. Ripple Carry Counter

2.1 Ripple Carry Counter

We have 3 design source files and a simulation only file.

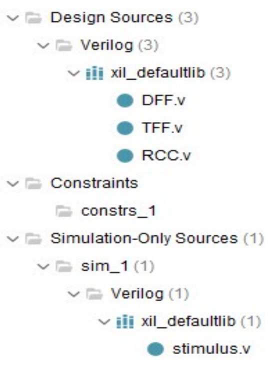


Fig 2.1-1 file list

2.1.1 DFF.v

```
module D_FF(q, d, clk, reset);
    output q;
    input d, clk, reset;
    reg q;
    always @(posedge reset or negedge clk)
    //if reset, set q as 0; else assign d to q
    if (reset)
        q = 1'b0;
    else
        q = d;
endmodule
```

2.1.2 TFF.v

```
module T_FF(q, clk, reset);
    //pin declarition
    output q;
    input clk, reset;
    wire d;
    //using a D flip flop to implement a T flip flop
    D_FF dff0(q, d, clk, reset);
    not nl(d, q);
endmodule
```

2.1.3 RCC.v

```
module RCC(q, clk, reset);
    //pins declareation
    output [3:0] q;
    input clk, reset;
    //combine 4 TFF in sequence,
    // output from last TFF serve as clk of the next TFF
    T_FF tff0(q[0], clk, reset);
    T_FF tff1(q[1], q[0], reset);
    T_FF tff2(q[2], q[1], reset);
    T_FF tff3(q[3], q[2], reset);
endmodule
```

2.2 Test Result

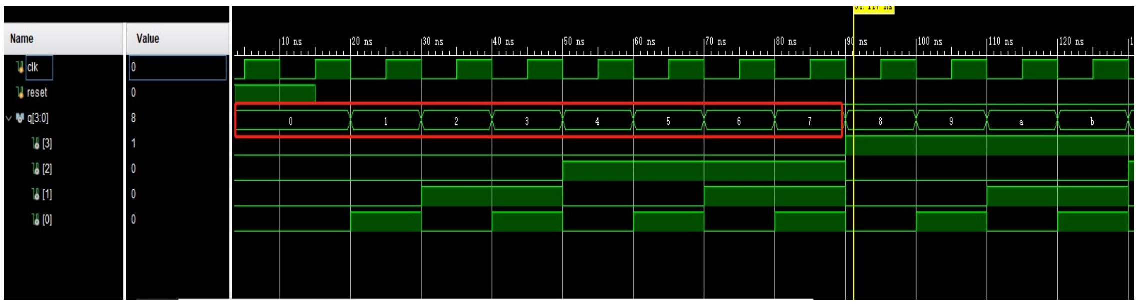
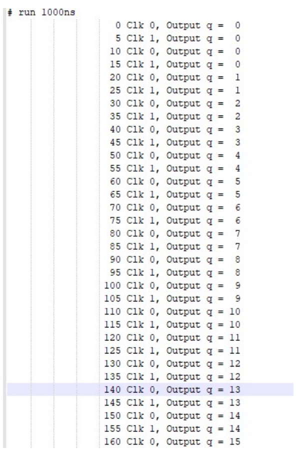


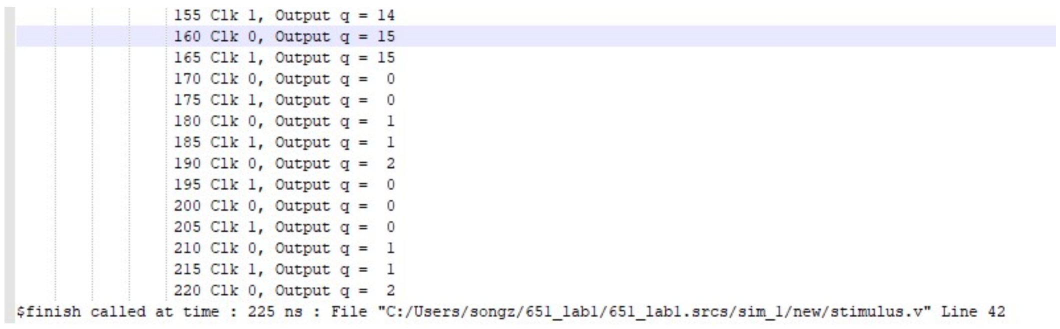
Fig 2.2-1 Wave Form

We can see output wave form in Fig 2.2-1. After reset goes down(0), at each negedge of clk, output of q adds 1.

Log printed:



```
# run 1000ns
  0 Clk 0, Output q =  0
  5 Clk 1, Output q =  0
 10 Clk 0, Output q =  0
 15 Clk 1, Output q =  0
 20 Clk 0, Output q =  1
 25 Clk 1, Output q =  1
 30 Clk 0, Output q =  2
 35 Clk 1, Output q =  2
 40 Clk 0, Output q =  3
 45 Clk 1, Output q =  3
 50 Clk 0, Output q =  4
 55 Clk 1, Output q =  4
 60 Clk 0, Output q =  5
 65 Clk 1, Output q =  5
 70 Clk 0, Output q =  6
 75 Clk 1, Output q =  6
 80 Clk 0, Output q =  7
 85 Clk 1, Output q =  7
 90 Clk 0, Output q =  8
 95 Clk 1, Output q =  8
100 Clk 0, Output q =  9
105 Clk 1, Output q =  9
110 Clk 0, Output q = 10
115 Clk 1, Output q = 10
120 Clk 0, Output q = 11
125 Clk 1, Output q = 11
130 Clk 0, Output q = 12
135 Clk 1, Output q = 12
140 Clk 0, Output q = 13
145 Clk 1, Output q = 13
150 Clk 0, Output q = 14
155 Clk 1, Output q = 14
160 Clk 0, Output q = 15
```



```
155 Clk 1, Output q = 14
160 Clk 0, Output q = 15
165 Clk 1, Output q = 15
170 Clk 0, Output q =  0
175 Clk 1, Output q =  0
180 Clk 0, Output q =  1
185 Clk 1, Output q =  1
190 Clk 0, Output q =  2
195 Clk 1, Output q =  0
200 Clk 0, Output q =  0
205 Clk 1, Output q =  0
210 Clk 0, Output q =  1
215 Clk 1, Output q =  1
220 Clk 0, Output q =  2
$finish called at time : 225 ns : File "C:/Users/songz/651_lab1/651_lab1.srcs/sim_1/new/stimulus.v" Line 42
```

Fig 2.2-2 Log

3. 4:1 Mux

3.1 MUX

MUX.v

In MUX.v, we implement a 4:1 mux, my way is using a 2 level select, to assign output.

```
module MUX(
    result, in0, in1, in2, in3, S0, S1
    );
    //pins declaration
    output result;
    input in0, in1, in2, in3, S0, S1;
    //S0    S1  result
    //0     0   in0
    //0     1   in1
    //1     0   in2
    //1     1   in3
    //do assign result value with 2 level select
    assign result = S1?(S0?in3:in2):(S0?in1:in0);
endmodule
```

3.2 TestBench

Mux_stimulus.v

```
module stimulus_mux;
    //pins declaration
    reg S0, S1, in0, in1, in2, in3;
    wire result;
    //add MUX
    MUX m1(result, in0, in1, in2, in3, S0, S1);
    initial
    //initial inputs
        begin
            in0 = 1'b0;
            in1 = 1'b0;
            in2 = 1'b0;
            in3 = 1'b0;
        end
    initial
        begin
            //in this part, we set inputs as that:
            //there is only one '1' or '0' at a time, and use mux to select this '1' or '0'
            //the output is supposed to be like all 1 in the first 40ns, all 0 in the next 40 ns
            #10 {in3, in2, in1, in0} = 4'b0001;
            {S1, S0} = 2'b00;
            #10 {in3, in2, in1, in0} = 4'b0010;
            {S1, S0} = 2'b01;
            #10 {in3, in2, in1, in0} = 4'b0100;
            {S1, S0} = 2'b10;
            #10 {in3, in2, in1, in0} = 4'b1000;
            {S1, S0} = 2'b11;

            #10 {in3, in2, in1, in0} = 4'b1110;
            {S1, S0} = 2'b00;
            #10 {in3, in2, in1, in0} = 4'b1101;
            {S1, S0} = 2'b01;
            #10 {in3, in2, in1, in0} = 4'b1011;
            {S1, S0} = 2'b10;
            #10 {in3, in2, in1, in0} = 4'b0111;
            {S1, S0} = 2'b11;
            #20 $finish;
        end
    //printing logs
    initial $monitor($time, "Input: %b%b%b%b, Select: %b%b, output: %b", in3, in2, in1, in0, S1, S0, result);
endmodule
```

In test bench, I set up input group as: this is only 1 different bit(0010, 1011). Use {S1, S0} to select the different one bit. The wave form is supposed to be: first 40ns result is high(1), next 40ns result is low(0).

3.3 Test result

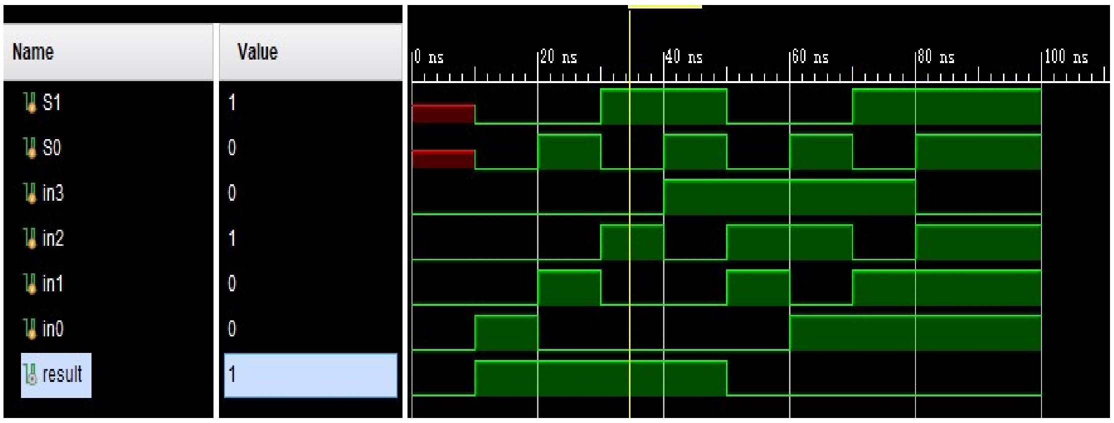
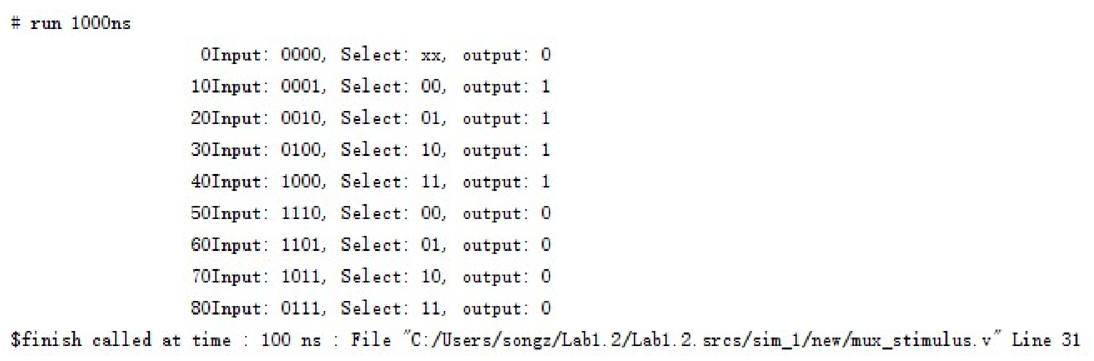


Fig 3.3-1 test result wave form



```
# run 1000ns
                  0Input: 0000, Select: xx, output: 0
                 10Input: 0001, Select: 00, output: 1
                 20Input: 0010, Select: 01, output: 1
                 30Input: 0100, Select: 10, output: 1
                 40Input: 1000, Select: 11, output: 1
                 50Input: 1110, Select: 00, output: 0
                 60Input: 1101, Select: 01, output: 0
                 70Input: 1011, Select: 10, output: 0
                 80Input: 0111, Select: 11, output: 0
$finish called at time : 100 ns : File "C:/Users/songz/Lab1.2/Lab1.2.srcs/sim_1/new/mux_stimulus.v" Line 31
```

Fig 3.3-2 test result logs

4. Questions

(a) $monitor is used to print specified log. $finish is to terminate whole simulation.

(b) '#' is to wait specified time units following. $time can be translated to current time stamp.

(c) The delay for 4:1 MUX built with gates only, is about the same as one for a 2:1 MUX. Which is much less than a 4:1 MUX built through three 2:1 MUX. Because the delay mainly come from not gate in MUX. In the first case, not gates are parallel, followed by and gate and or gate. But in the latter case, 2 layered MUX will add delay.