# ECEN 651 Lab Report1

## Zhenlei Song 826009388

### Objectives

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

### Ripple Carry Counter

* 1. Ripple Carry Counter

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

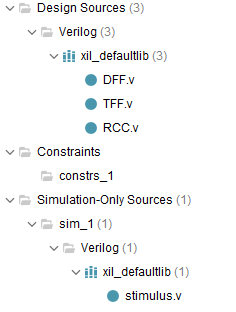


Fig 2.1-1 file list

* + 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

* + 1. 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

* 1. Test Result

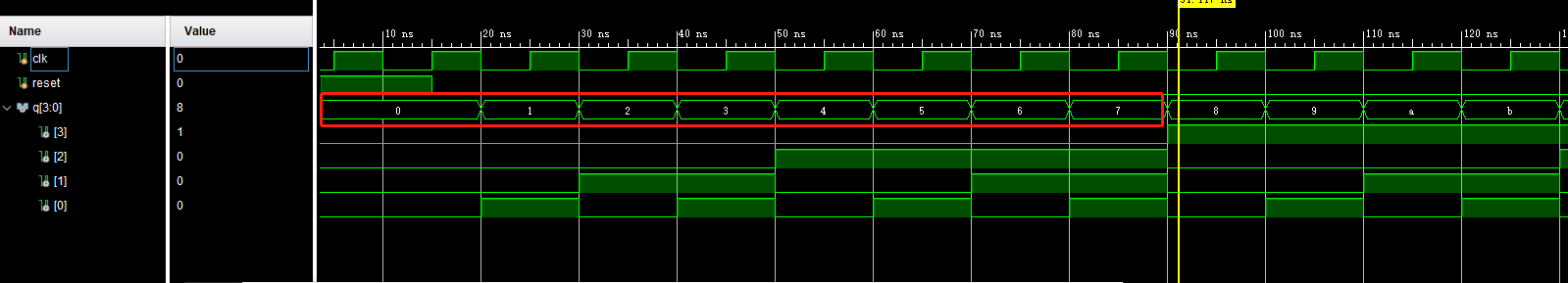


Fig 2.2-1 Wave Form

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:

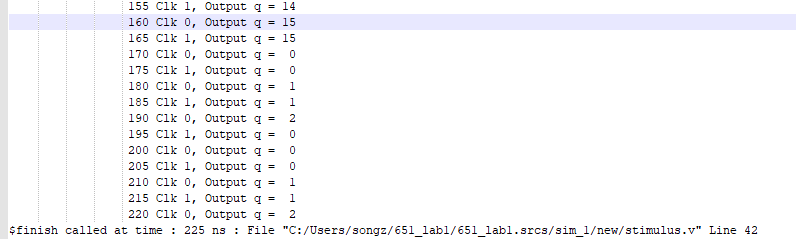
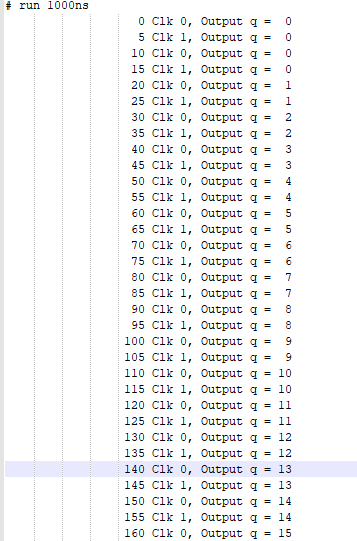


Fig 2.2-2 Log

### 4:1 Mux

* 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

* 1. 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

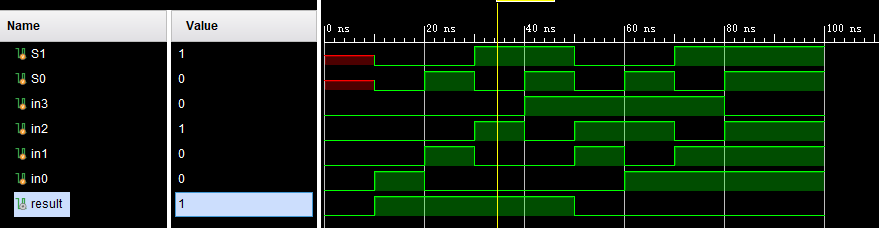


Fig 3.3-1 test result wave form

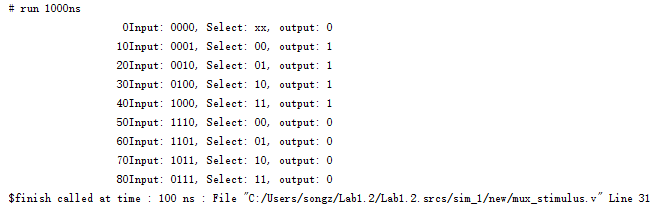


Fig 3.3-2 test result logs

1. Questions
2. $monitor is used to print specified log. $finish is to terminate whole simulation.
3. ‘#’ is to wait specified time units following. $time can be translated to current time stamp.
4. 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.