Lab 12 Finite State Machines

Ali Muhammad Asad   
aa07190

Top Level Module

`timescale 1ns / 1ps

module TopLevelModule\_FSM\_Lab12(clk,left,right,reset,La,Lb,Lc,Ra,Rb,Rc);

input clk, left, right, reset;

output La, Lb, Lc, Ra, Rb, Rc;

wire A, B, C, D, E, F, G;

wire clk\_div, res;

assign res = reset;

lab12\_clkdiv cd12(.clk(clk), .clk\_d(clk\_div));

D\_flipflop dff1(.D(A),.clk(clk\_div), .reset(res), .Q(B));

D\_flipflop dff2(.D(G),.clk(clk\_div), .reset(res), .Q(C));

D\_flipflop dff3(.D(F),.clk(clk\_div), .reset(res), .Q(D));

assign A = (B&&(~C)&&(~res))|| ((~C)&&(~D)&&(~res)&&right);

assign F = ((~C)&&(D)&&(~res))||((~B)&&(C)&&(~D)&&(~res))||((~B)&&(~C)&&(~res)&&(right)&&(left));

assign G = ((~B)&&(~D)&&(res)&&(left))||((~B)&&(~D)&&(C)&&(~res))||((~D)&&(~res)&&(~C)&&(B));

assign La = ((C)&&(D)&&(~res))||((~B)&&(C)&&(~res))||((~B)&&(D)&&(~res));

assign Lb = ((C)&&(D)&&(~res))||((~B)&&(D)&&(~res));

assign Lc = ((C)&&(D)&&(~res));

assign Ra = ((B)&&(~C));

assign Rb = ((B)&&(D)&&(~res))||((B)&&(C)&&(~res));

assign Rc = ((B)&&(C)&&(~res));

endmodule

Clock Divider:

`timescale 1ns / 1ps

module lab12\_clkdiv(input clk,

output reg clk\_d

);

parameter div\_val = 25'b1011111010111100001000000;

// input clk; output reg clk\_d;

reg[25:0] count;

initial begin

clk\_d = 0; count = 0;

end

always @(posedge clk)

begin

if(count == div\_val)

begin

clk\_d <= ~clk\_d;

count = 0;

end

else

count <= count + 1;

end

endmodule

D Flip Flop

`timescale 1ns / 1ps

module D\_flipflop(

input D,

input clk,

input reset,

output reg Q

);

always @(posedge clk)

begin

if(reset == 1'b1)

Q <= 1'b0;

else

Q <= D;

end

endmodule

Constraints

