Lab04

Altay İlker Yiğitel

CS223 Sec:01

22203024

D Flip Flop

module (

input logic d,

input logic clk,

input logic reset,

output logic q

);

always\\_ff @(posedge clk) begin

if (reset)

q <= 0;

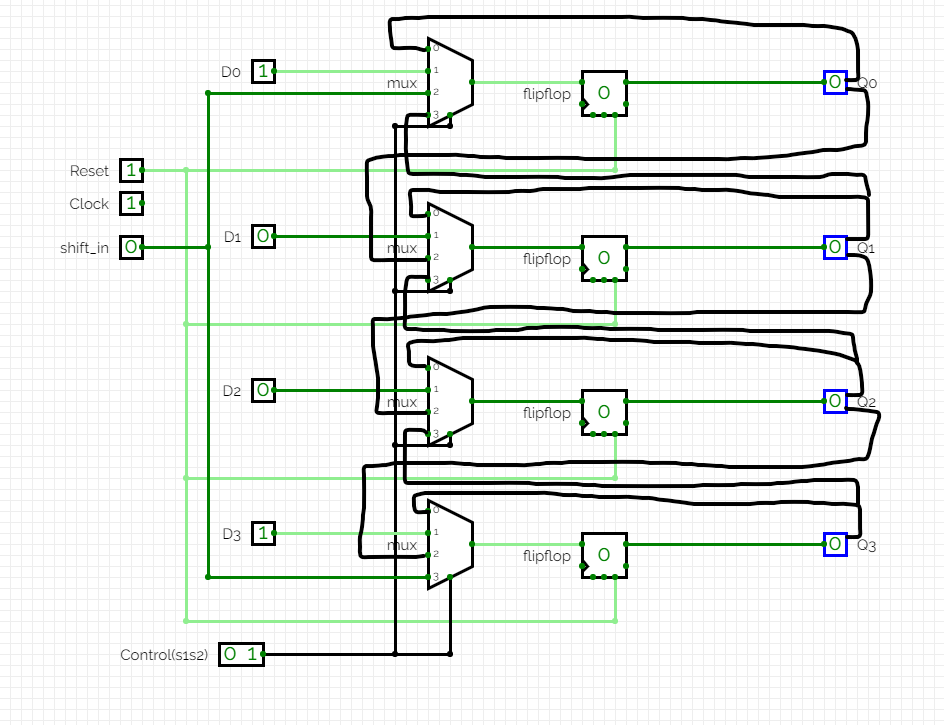
else

q <= d;

end

endmodule

Circuit schematic (block diagram) for the internal design of the multifunction register by using 4:1 multiplexers and synchronously resettable D flip-flops



Structural SystemVerilog module for the multifunction register designed and a testbench for it

module multifunction\_register (

input logic [3:0] d,

input logic [1:0] s,

input logic clk,

input logic reset,

input logic shift\_in,

output logic [3:0] q

);

logic [3:0] q\_temp;

always\_ff @(posedge clk) begin

if (reset) begin

q\_temp <= 4'b0000;

end

else begin

case(s)

2'b00: q\_temp <= q\_temp;

2'b01: q\_temp <= d;

2'b10: q\_temp <= {q\_temp[2:0], shift\_in};

2'b11: q\_temp <= {shift\_in, q\_temp[3:1]};

default: q\_temp <= 4'b0000;

endcase

end

end

assign q = q\_temp;

endmodule

module testbench;

localparam CLK\_PERIOD = 10;

logic [3:0] d;

logic [1:0] s;

logic clk;

logic reset;

logic shift\_in;

logic [3:0] q;

multifunction\_register uut (

.d(d),

.s(s),

.clk(clk),

.reset(reset),

.shift\_in(shift\_in),

.q(q)

);

always #CLK\_PERIOD/2 clk = ~clk;

initial begin

d = 4'b0000;

s = 2'b00;

reset = 1'b0;

shift\_in = 1'b0;

#5 reset = 1'b1;

#5 reset = 1'b0;

#10;

d = 4'b1010;

s = 2'b01;

#10;

d = 4'b0101;

s = 2'b10;

shift\_in = 1'b1;

#10;

d = 4'b1100;

s = 2'b11;

shift\_in = 1'b0;

#10;

$display("All test cases passed!");

$finish;

end

endmodule

SystemVerilog module for BCD-to-7-segment code converter

module logic\_gate (

input wire w3, w2, w1, w0,

output reg a, b, c, d, e, f, g

);

always @\* begin

if (!w3 && !w2 && !w1 && !w0) begin

{a, b, c, d, e, f, g} = 7'b0000001;

end

else if (!w3 && !w2 && !w1 && w0) begin

{a, b, c, d, e, f, g} = 7'b1001111;

end

else if (!w3 && !w2 && w1 && !w0) begin

{a, b, c, d, e, f, g} = 7'b0010010;

end

else if (!w3 && !w2 && w1 && w0) begin

{a, b, c, d, e, f, g} = 7'b0000110;

end

else if (!w3 && w2 && !w1 && !w0) begin

{a, b, c, d, e, f, g} = 7'b1001100;

end

else if (!w3 && w2 && !w1 && w0) begin

{a, b, c, d, e, f, g} = 7'b0100100;

end

else if (!w3 && w2 && w1 && !w0) begin

{a, b, c, d, e, f, g} = 7'b0100000;

end

else if (!w3 && w2 && w1 && w0) begin

{a, b, c, d, e, f, g} = 7'b0001111;

end

else if (w3 && !w2 && !w1 && !w0) begin

{a, b, c, d, e, f, g} = 7'b0000000;

end

else begin

{a, b, c, d, e, f, g} = 7'b1111110;

end

end

endmodule

Testbench

module testbench;

reg w3, w2, w1, w0;

wire a, b, c, d, e, f, g;

logic\_gate dut (

.w3(w3),

.w2(w2),

.w1(w1),

.w0(w0),

.a(a),

.b(b),

.c(c),

.d(d),

.e(e),

.f(f),

.g(g)

);

initial begin

w3 = 0; w2 = 0; w1 = 0; w0 = 0;

#10;

for (int i = 0; i < 16; i++) begin

w3 = i[3];

w2 = i[2];

w1 = i[1];

w0 = i[0];

#10;

end

$finish;

end

endmodule

What are the disadvantages of using BCD to represent numbers? List two or three points and explain them with a few short sentences.

Inefficient Use of Memory: Because a 4-bit binary code is used to store each decimal digit, BCD takes more memory than binary representation. for example while max 99 can be represented with 8bits in BCD classic binary representation can display up to 256 with 8bits.   
  
Arithmetic Operation Complexity: Compared to binary representation, performing arithmetic operations on BCD values directly, such as addition and multiplication, can be slower and more difficult. The need to manage carries between decimal digits causes this complexity, which results in more complicated and slowly operating arithmetic algorithms.

Research how the full four-digit seven-segment display on the Basys3 board works. Think about which sequential components you need to drive it. Explain with a few short sentences.

A multiplexer is used to display the digits. One digit is displayed at the same time because of optical afterglow effect and the principle of visual residue we can see all of them at the same time. Also, outputs are in reversed order which is probably because common anode displays typically require fewer external connections.