CS-223

Digital Design

Section: 3

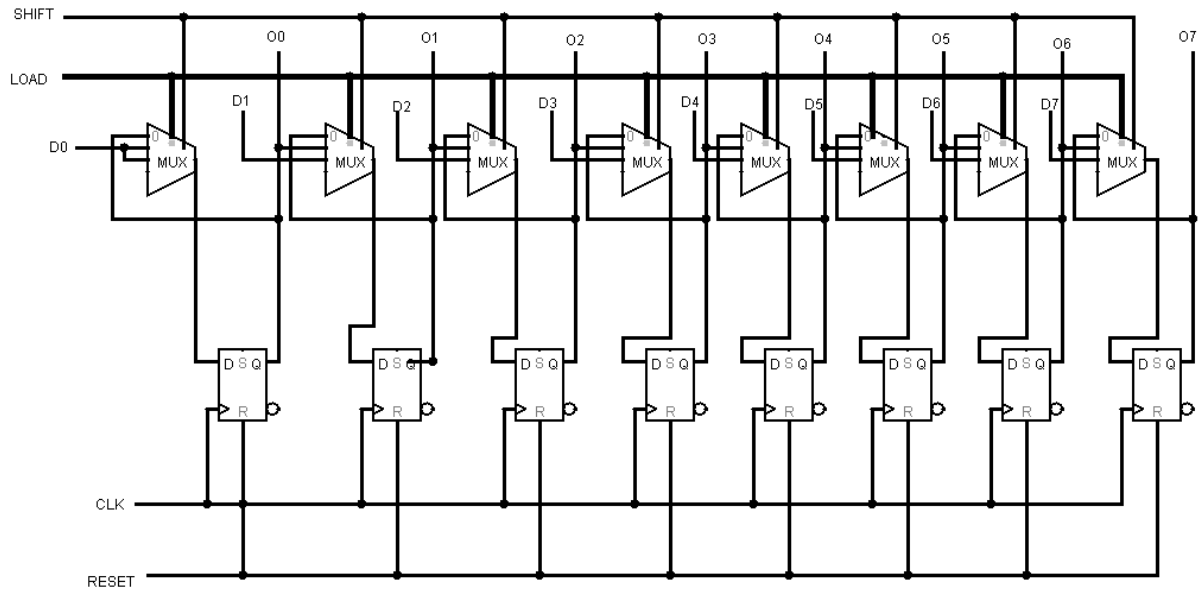
LAB 4

Mehmet Hasat Serinkan

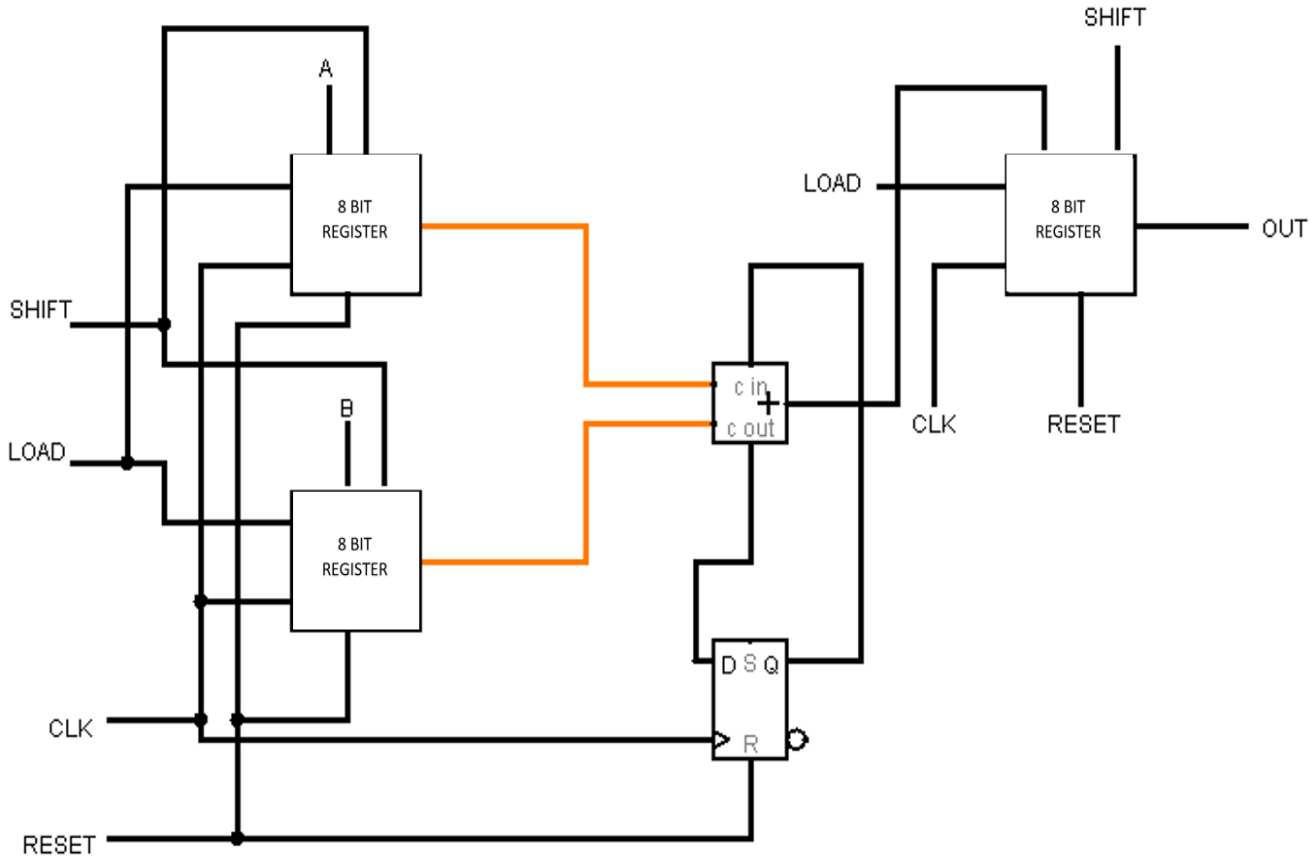
21901649

11.04.2022

B) Circuit schematic for your shift register design using D flip-flops.



C) Circuit schematic for your serial adder using the shift registers, full adder, and D flip-flop.



D) SystemVerilog module for synchronously resettable D flip-flop.

```
`timescale 1ns / 1ps

module dFlipFlop( input logic d, clk, reset, output logic q );

always_ff@(posedge clk, posedge reset)
begin
    if(reset)
        q <= 0;
    else
        q <= d;
end



endmodule
```

E) Structural SystemVerilog module for your shift register using the D flip-flop module along with the testbench.

```
`timescale 1ns / 1ps

module shiftRegister( input logic clk, input logic reset,
input logic shift, input logic load,
input logic[7:0] in,
output logic[7:0] q
    );

logic d0, d1, d2, d3, d4, d5, d6, d7;

dFlipFlop dff0(d0, clk, reset, q[0] );
dFlipFlop dff1(d1, clk, reset, q[1] );
dFlipFlop dff2(d2, clk, reset, q[2] );
dFlipFlop dff3(d3, clk, reset, q[3] );
dFlipFlop dff4(d4, clk, reset, q[4] );
dFlipFlop dff5(d5, clk, reset, q[5] );
dFlipFlop dff6(d6, clk, reset, q[6] );
dFlipFlop dff7(d7, clk, reset, q[7] );

always_ff @(posedge clk)
    begin
       if(reset)
        begin
            d0 = 0;
            d1 = 0;
            d2 = 0;
            d3 = 0;
            d4 = 0;
            d5 = 0;
            d6 = 0;
           d7 = 0;
       end

       else if(shift)
       begin
             d0 = 0;
             d1 = q[0];
             d2 = q[1];
             d3 = q[2];
             d4 = q[3];
             d5 = q[4];
             d6 = q[5];
             d7 = q[6];
         end
         else if(load)
         begin
             d0 = in[0];
             d1 = in[1];
             d2 = in[2];
             d3 = in[3];
             d4 = in[4];
             d5 = in[5];
             d6 = in[6];
             d7 = in[7];
         end
    end

endmodule
```

```
`timescale 1ns / 1ps

module testbench1();
logic clk, reset, shift, load;
logic [7:0] in, q;
shiftRegister dut(clk, reset, shift, load, in, q);
initial begin
    reset <= 1;
    reset <= 0;
    shift <= 0;
    load <= 1;
    in[0] <= 0; #5;
    in[1] <= 1; #5;
    in[2] <= 1; #5;
    in[3] <= 0; #5;
    in[4] <= 1; #5;
    in[5] <= 0; #5;
    in[6] <= 1; #5;
    in[7] <= 0; #5;
    load <= 0;
    shift <= 1;
end
always
begin
    clk <= 1; #1;
    clk <= 0; #1;
end


endmodule
```

F) Structural SystemVerilog module for your serial adder using the shift register, full adder, and D flip-flop modules along with the testbench.

This module is for the last register that is register out.

```
module sumShiftRegister( input logic clk, input logic reset,
input logic shift, input logic load,
input logic in,
output logic[7:0] q
);

logic d0, d1, d2, d3, d4, d5, d6, d7;

dFlipFlop dff0(d0, clk, reset, q[0] );
dFlipFlop dff1(d1, clk, reset, q[1] );
dFlipFlop dff2(d2, clk, reset, q[2] );
dFlipFlop dff3(d3, clk, reset, q[3] );
dFlipFlop dff4(d4, clk, reset, q[4] );
dFlipFlop dff5(d5, clk, reset, q[5] );
dFlipFlop dff6(d6, clk, reset, q[6] );
dFlipFlop dff7(d7, clk, reset, q[7] );

always_ff @(posedge clk)
   begin
      if(reset)
      begin
           d0 = 0;
           d1 = 0;
           d2 = 0;
           d3 = 0;
           d4 = 0;
           d5 = 0;
           d6 = 0;
           d7 = 0;
      end

      else if(shift)
      begin
             d0 = in;
             d1 = q[0];
             d2 = q[1];
```

```
                d2 = q[1];
                d3 = q[2];
                d4 = q[3];
                d5 = q[4];
                d6 = q[5];
                d7 = q[6];
            end

        end

    endmodule

```

This is the structural serial adder and its testbench.

```
`timescale 1ns / 1ps

module halfadder( input logic a, b, output logic sum, carry);

    assign sum = a ^ b;
    assign carry = a & b;

endmodule

module fulladder( input logic a, b, cin, output logic sum, cout);

    logic sum1, carry1, carry2;

    //Halfadders
    halfadder ha1(a, b, sum1, carry1);
    halfadder ha2(sum1, cin, sum, carry2);

    //Or gate
    or( cout, carry1, carry2);

endmodule



module serialAdder( input logic clk, input logic reset,
input logic shift, input logic load,
input logic[7:0] a, input logic[7:0] b,
output logic[7:0] aOut,
output logic[7:0] bOut,
output logic[7:0] registerSum
);

logic d;
logic cin;

logic q;
logic sum;
logic cout;

shiftRegister sRA( clk, reset, shift, load, a, aOut);
shiftRegister sRB( clk, reset, shift, load, b, bOut);

dFlipFlop dff( cout, clk, reset, cin);
fulladder fa(a[0], b[0], cin, sum, cout);



sumShiftRegister sumShifter( clk, reset, shift, load, sum, registerSum);

endmodule


```

```
`timescale 1ns / 1ps

module testbench2();
logic clk, reset, shift, load;
logic [7:0] a, b, aOut, bOut, sumOut;
serialAdder dut(clk, reset, shift, load, a, b, aOut, bOut ,sumOut);
initial begin
    reset <= 1; #300;
    a[0] <= 0; #5;
    a[1] <= 1; #5;
    a[2] <= 0; #5;
    a[3] <= 0; #5;
    a[4] <= 0; #5;
    a[5] <= 1; #5;
    a[6] <= 0; #5;
    a[7] <= 0; #5;
    b[0] <= 1; #5;
    b[1] <= 1; #5;
    b[2] <= 0; #5;
    b[3] <= 0; #5;
    b[4] <= 1; #5;
    b[5] <= 0; #5;
    b[6] <= 1; #5;
    b[7] <= 0; #5;
    reset <= 0; #400;
    shift <= 0;
    load <= 1;
    a[0] <= 0; #5;
    a[1] <= 1; #5;
    a[2] <= 0; #5;
    a[3] <= 1; #5;
    a[4] <= 0; #5;
    a[5] <= 1; #5;
    a[6] <= 0; #5;
    a[7] <= 0; #5;
    b[0] <= 1; #5;
    b[1] <= 0; #5;
    b[2] <= 0; #5;
    b[3] <= 1; #5;
    b[4] <= 0; #5;
    b[5] <= 1; #5;
    b[6] <= 0; #5;
    b[7] <= 1; #5;
    load <= 1; #5;
    shift <= 1;
end
always
begin
    clk <= 1; #1;
    clk <= 0; #1;
end


endmodule
```