

CS-223

Digital Design

Section: 3

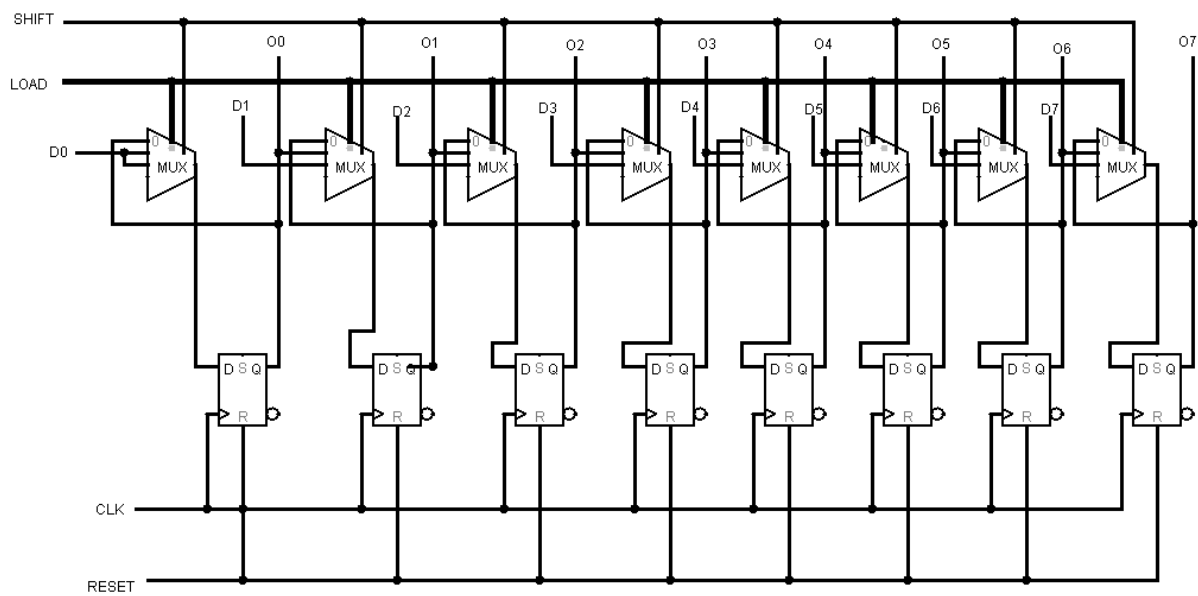
LAB 4

Mehmet Hasat Serinkan

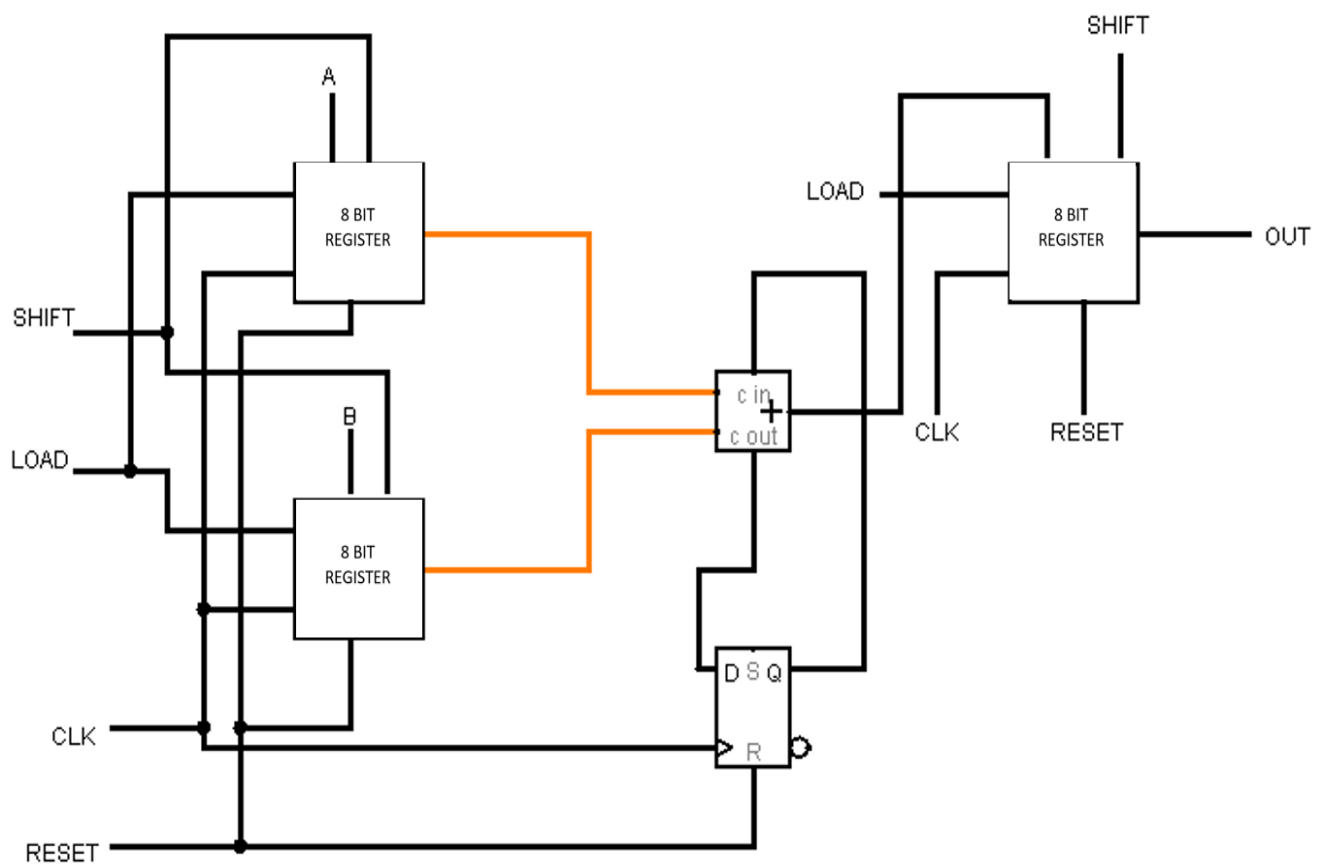
21901649

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.

```
1  `timescale 1ns / 1ps
2
3  module shiftRegister( input logic clk, input logic reset,
4  input logic shift, input logic load,
5  input logic[7:0] in,
6  output logic[7:0] q
7  );
8
9  logic d0, d1, d2, d3, d4, d5, d6, d7;
10
11  dFlipFlop dff0(d0, clk, reset, q[0] );
12  dFlipFlop dff1(d1, clk, reset, q[1] );
13  dFlipFlop dff2(d2, clk, reset, q[2] );
14  dFlipFlop dff3(d3, clk, reset, q[3] );
15  dFlipFlop dff4(d4, clk, reset, q[4] );
16  dFlipFlop dff5(d5, clk, reset, q[5] );
17  dFlipFlop dff6(d6, clk, reset, q[6] );
18  dFlipFlop dff7(d7, clk, reset, q[7] );
19
20  always_ff @(posedge clk)
21  begin
22      if(reset)
23      begin
24          d0 = 0;
25          d1 = 0;
26          d2 = 0;
27          d3 = 0;
28          d4 = 0;
29          d5 = 0;
30          d6 = 0;
31          d7 = 0;
32      end
33
34      else if(shift)
35      begin
36          d0 = 0;
37          d1 = q[0];
38          d2 = q[1];
39          d3 = q[2];
40          d4 = q[3];
41          d5 = q[4];
42          d6 = q[5];
43          d7 = q[6];
44      end
45      else if(load)
46      begin
47          d0 = in[0];
48          d1 = in[1];
49          d2 = in[2];
50          d3 = in[3];
51          d4 = in[4];
52          d5 = in[5];
53          d6 = in[6];
54          d7 = in[7];
55      end
56  end
57
58  endmodule
```

```

1  `timescale 1ns / 1ps
2
3  module testbench1();
4  logic clk, reset, shift, load;
5  logic [7:0] in, q;
6  shiftRegister dut(clk, reset, shift, load, in, q);
7  initial begin
8      reset <= 1;
9      reset <= 0;
10     shift <= 0;
11     load <= 1;
12     in[0] <= 0; #5;
13     in[1] <= 1; #5;
14     in[2] <= 1; #5;
15     in[3] <= 0; #5;
16     in[4] <= 1; #5;
17     in[5] <= 0; #5;
18     in[6] <= 1; #5;
19     in[7] <= 0; #5;
20     load <= 0;
21     shift <= 1;
22 end
23 always
24 begin
25     clk <= 1; #1;
26     clk <= 0; #1;
27 end
28
29
30 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.

```

1  module sumShiftRegister( input logic clk, input logic reset,
2  input logic shift, input logic load,
3  input logic in,
4  output logic[7:0] q
5  );
6
7  logic d0, d1, d2, d3, d4, d5, d6, d7;
8
9  dFlipFlop dff0(d0, clk, reset, q[0] );
10 dFlipFlop dff1(d1, clk, reset, q[1] );
11 dFlipFlop dff2(d2, clk, reset, q[2] );
12 dFlipFlop dff3(d3, clk, reset, q[3] );
13 dFlipFlop dff4(d4, clk, reset, q[4] );
14 dFlipFlop dff5(d5, clk, reset, q[5] );
15 dFlipFlop dff6(d6, clk, reset, q[6] );
16 dFlipFlop dff7(d7, clk, reset, q[7] );
17
18 always_ff @(posedge clk)
19 begin
20     if(reset)
21     begin
22         d0 = 0;
23         d1 = 0;
24         d2 = 0;
25         d3 = 0;
26         d4 = 0;
27         d5 = 0;
28         d6 = 0;
29         d7 = 0;
30     end
31
32     else if(shift)
33     begin
34         d0 = in;
35         d1 = q[0];
36         d2 = q[1];

```

```

36         d2 = q[1];
37         d3 = q[2];
38         d4 = q[3];
39         d5 = q[4];
40         d6 = q[5];
41         d7 = q[6];
42     end
43
44 end
45
46 endmodule
47

```

This is the structural serial adder and its testbench.

```

1  `timescale 1ns / 1ps
2
3  module halfadder( input logic a, b, output logic sum, carry);
4
5      assign sum = a ^ b;
6      assign carry = a & b;
7
8  endmodule
9
10 module fulladder( input logic a, b, cin, output logic sum, cout);
11
12     logic sum1, carry1, carry2;
13
14     //Halfadders
15     halfadder hal(a, b, sum1, carry1);
16     halfadder ha2(sum1, cin, sum, carry2);
17
18     //Or gate
19     or( cout, carry1, carry2);
20
21 endmodule
22
23
24
25 module serialAdder( input logic clk, input logic reset,
26 input logic shift, input logic load,
27 input logic[7:0] a, input logic[7:0] b,
28 output logic[7:0] aOut,
29 output logic[7:0] bOut,
30 output logic[7:0] registerSum
31 );
32
33     logic d;
34     logic cin;
35
36     logic q;
37     logic sum;
38     logic cout;
39
40     shiftRegister sRA( clk, reset, shift, load, a, aOut);
41     shiftRegister sRB( clk, reset, shift, load, b, bOut);
42
43     dFlipFlop dff( cout, clk, reset, cin);
44     fulladder fa(a[0], b[0], cin, sum, cout);
45
46
47
48     sumShiftRegister sumShifter( clk, reset, shift, load, sum, registerSum);
49
50 endmodule
51
52

```

```

1  `timescale 1ns / 1ps
2
3  module testbench2();
4  logic clk, reset, shift, load;
5  logic [7:0] a, b, aOut, bOut, sumOut;
6  serialAdder dut(clk, reset, shift, load, a, b, aOut, bOut, sumOut);
7  initial begin
8      reset <= 1; #300;
9      a[0] <= 0; #5;
10     a[1] <= 1; #5;
11     a[2] <= 0; #5;
12     a[3] <= 0; #5;
13     a[4] <= 0; #5;
14     a[5] <= 1; #5;
15     a[6] <= 0; #5;
16     a[7] <= 0; #5;
17     b[0] <= 1; #5;
18     b[1] <= 1; #5;
19     b[2] <= 0; #5;
20     b[3] <= 0; #5;
21     b[4] <= 1; #5;
22     b[5] <= 0; #5;
23     b[6] <= 1; #5;
24     b[7] <= 0; #5;
25     reset <= 0; #400;
26     shift <= 0;
27     load <= 1;
28     a[0] <= 0; #5;
29     a[1] <= 1; #5;
30     a[2] <= 0; #5;
31     a[3] <= 1; #5;
32     a[4] <= 0; #5;
33     a[5] <= 1; #5;
34     a[6] <= 0; #5;
35     a[7] <= 0; #5;
36     b[0] <= 1; #5;
37     b[1] <= 0; #5;
38     b[2] <= 0; #5;
39     b[3] <= 1; #5;
40     b[4] <= 0; #5;
41     b[5] <= 1; #5;
42     b[6] <= 0; #5;
43     b[7] <= 1; #5;
44     load <= 1; #5;
45     shift <= 1;
46 end
47 always
48 begin
49     clk <= 1; #1;
50     clk <= 0; #1;
51 end
52
53
54 endmodule

```