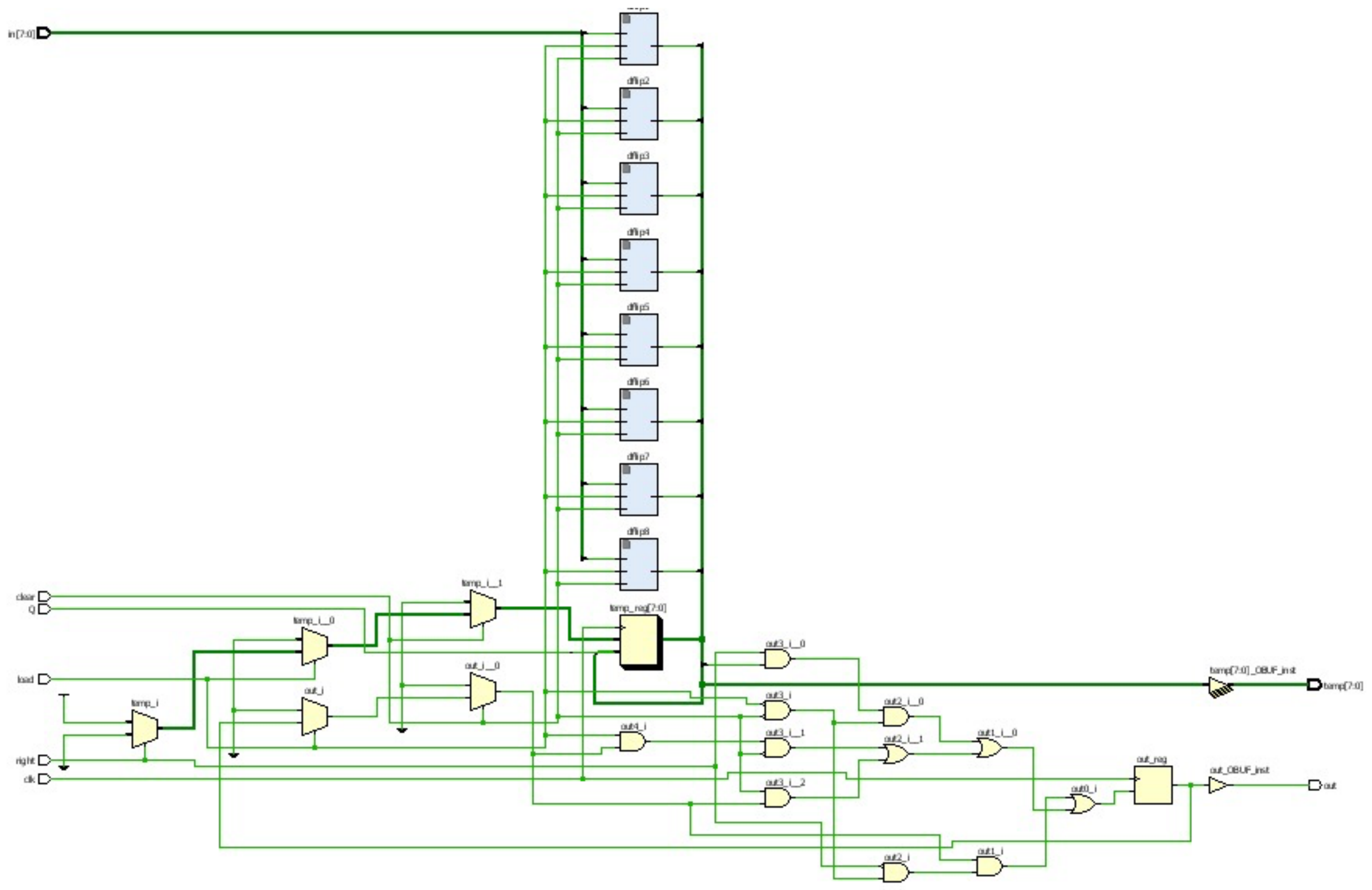
CS 223 DIGITAL DESIGN
SECTION 6
LAB 5

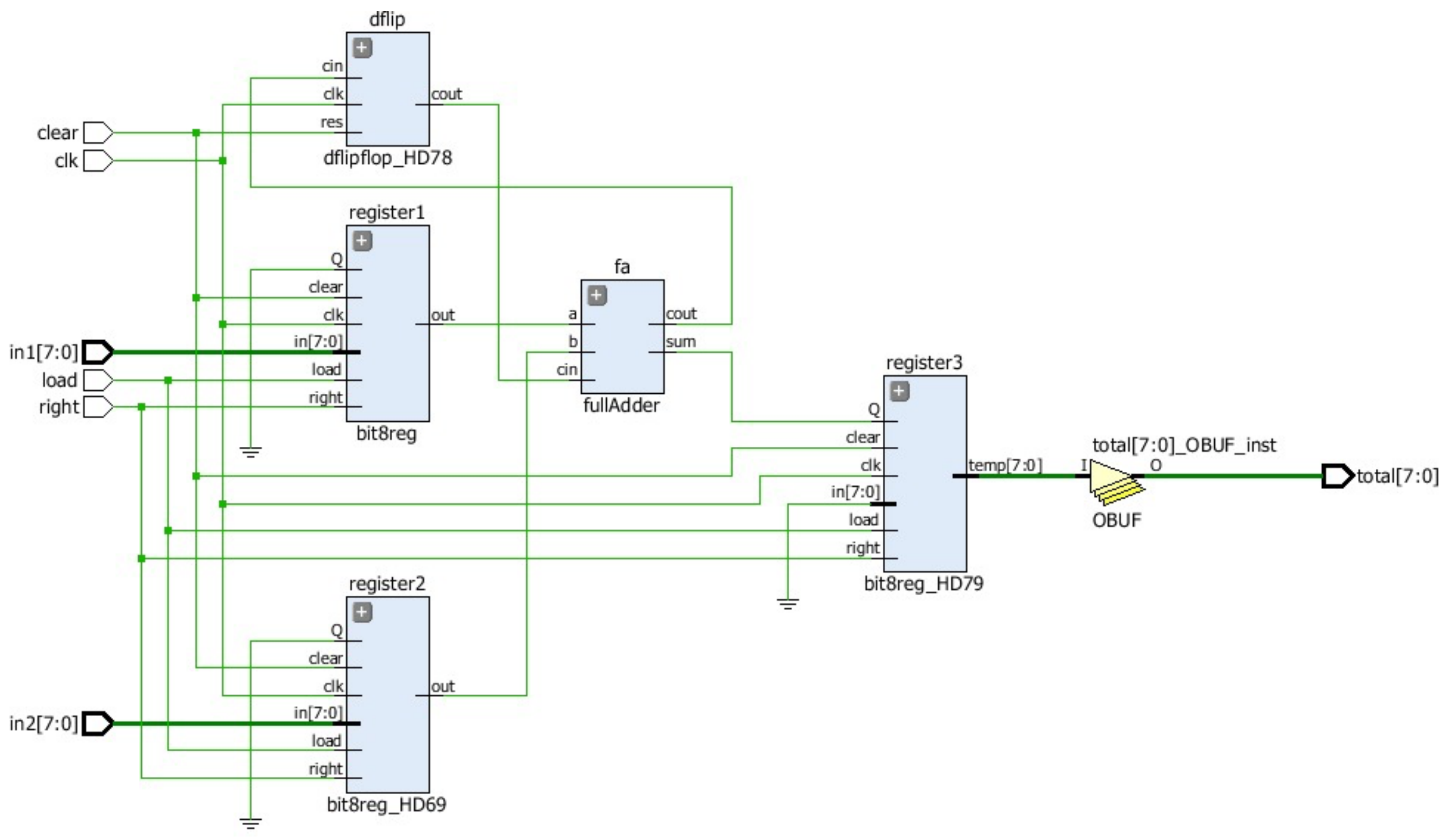
YASEMİN AKIN
22101782

5 DECEMBER 2022

(b) Circuit schematic for shift register design using D flip-flops.



(c) Circuit schematic for 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 resettableDFlipFlop (input logic d, reset, clk, output logic q);

    always_ff @(posedge clk) begin
        if(reset) q <= 1'b0;
        else q <= d;
    end

endmodule

//behavioral halfadder module to use in structural fulladder module
`timescale 1ns/1ps
module halfadder(
    input a,
    input b,
    output reg s,
    output reg c );
        always @(a, b) begin
            s <= a^b;
            c <= a&b;
        end
endmodule

//structural fulladder module to use in serial adder module
`timescale 1ns/1ps
module fulladder(input logic a,
    input logic b,
    input  logic cin,
    output logic sum,
    output logic cout
    );
    logic w1, w2, w3;
    halfadder halfadder0(a, b, w1, w2);
    halfadder halfadder1(w1, cin, sum, w3);
    or or0(cout, w2, w3);
endmodule

//(e) Structural SystemVerilog module for shift register using the D flip-
flop module along
//with the testbench.
`timescale 1ns/1ps
module register8bit(input logic [7:0] in,
                    input logic clk, load, rightshift, reset,
                    output logic [7:0] out);
    logic [7:0] tempOutput;

    resettableDFlipFlop ff0 (in[7], reset, load, tempOutput[7]);
    resettableDFlipFlop ff1 (in[6], reset, load, tempOutput[6]);
    resettableDFlipFlop ff2 (in[5], reset, load, tempOutput[5]);
    resettableDFlipFlop ff3 (in[4], reset, load, tempOutput[4]);
    resettableDFlipFlop ff4 (in[3], reset, load, tempOutput[3]);
    resettableDFlipFlop ff5 (in[2], reset, load, tempOutput[2]);
    resettableDFlipFlop ff6 (in[1], reset, load, tempOutput[1]);
    resettableDFlipFlop ff7 (in[0], reset, load, tempOutput[0]);

    always@(posedge clk)
        begin
```

```
            if(reset)
                out = 0;
            else if(load)
                out = tempOutput;
            else if(rightshift)
                out = out >> 1;
            else
                out = out;
        end
endmodule

//testbench
`timescale 1ns/1ps
module register8bitTest();
    logic [7:0] in;
    logic clk, load, rightshift, reset;
    logic [7:0] out;

    register8bit uut(input, clk, load, rightshift, reset, out);

    initial begin
        clk <= 0;

        reset = 1; #10;
        in = 8'b10110101; load = 1; #10;
        load = 0; rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
    end
    always #2 clk <= ~clk;
endmodule


//(f) Structural SystemVerilog module for serial adder using the shift
register, full adder,
//and D flip-flop modules along with the testbench.
`timescale 1ns/1ps
```

```
module serial8bitadder(input logic shiftSignal, loadSignal, reset, clk,
                        input logic [7:0] in1, in2,
                        output logic [7:0] out);
    logic outa, outb, sum, cin, cout;

    register8bit rega(in1, clk, loadSignal, rightshift, reset, outa);
    register8bit regb(in2, clk, loadSignal, rightshift, reset, outb);

    fulladder adder1(outa, outb, cin, sum, cout);
    resettableDFlipFlop ff1(cout, reset, clk, cin);
    register8bit regc(sum, clk, loadSignal, rightshift, reset, out);

endmodule

//testbench
`timescale 1ns/1ps
module serialAdderTest();
    logic shiftSignal, loadSignal, reset, clk;
    logic [7:0] in1, in2, out;

    serial8bitadder uut(shiftSignal, loadSignal, reset, clk, in1, in2,
out);

    initial begin
        clk <= 0;

        reset = 1; #10;
        in1 = 8'b10110101;
        in2 = 8'b00110011;

        loadSignal = 1; reset = 0; #10;
        loadSignal = 0; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
        rightshift = 1; #10;
    end
    always #2 clk <= ~clk;

endmodule
```