DIGITAL DESIGN 5th LAB

PRELIMINARY REPORT

Course Code: CS 223

Course Name: Digital Design

Section: 1

Name: Arda

Surname: İynem

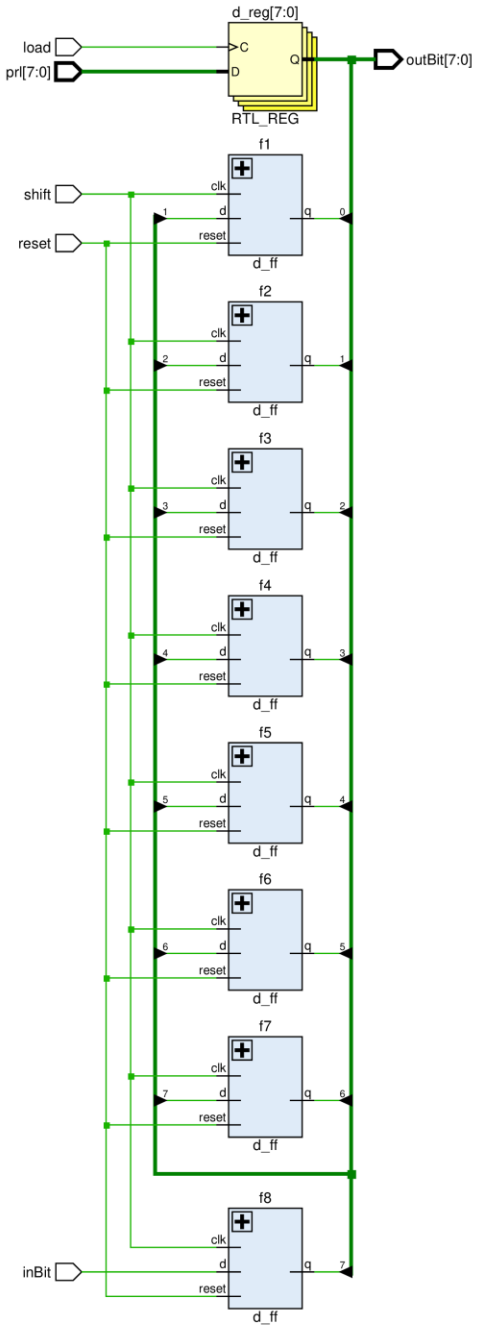
Student ID: 22002717

Date: 10.04.2022

SHIFT REGISTER

CIRCUIT SCHEMATIC

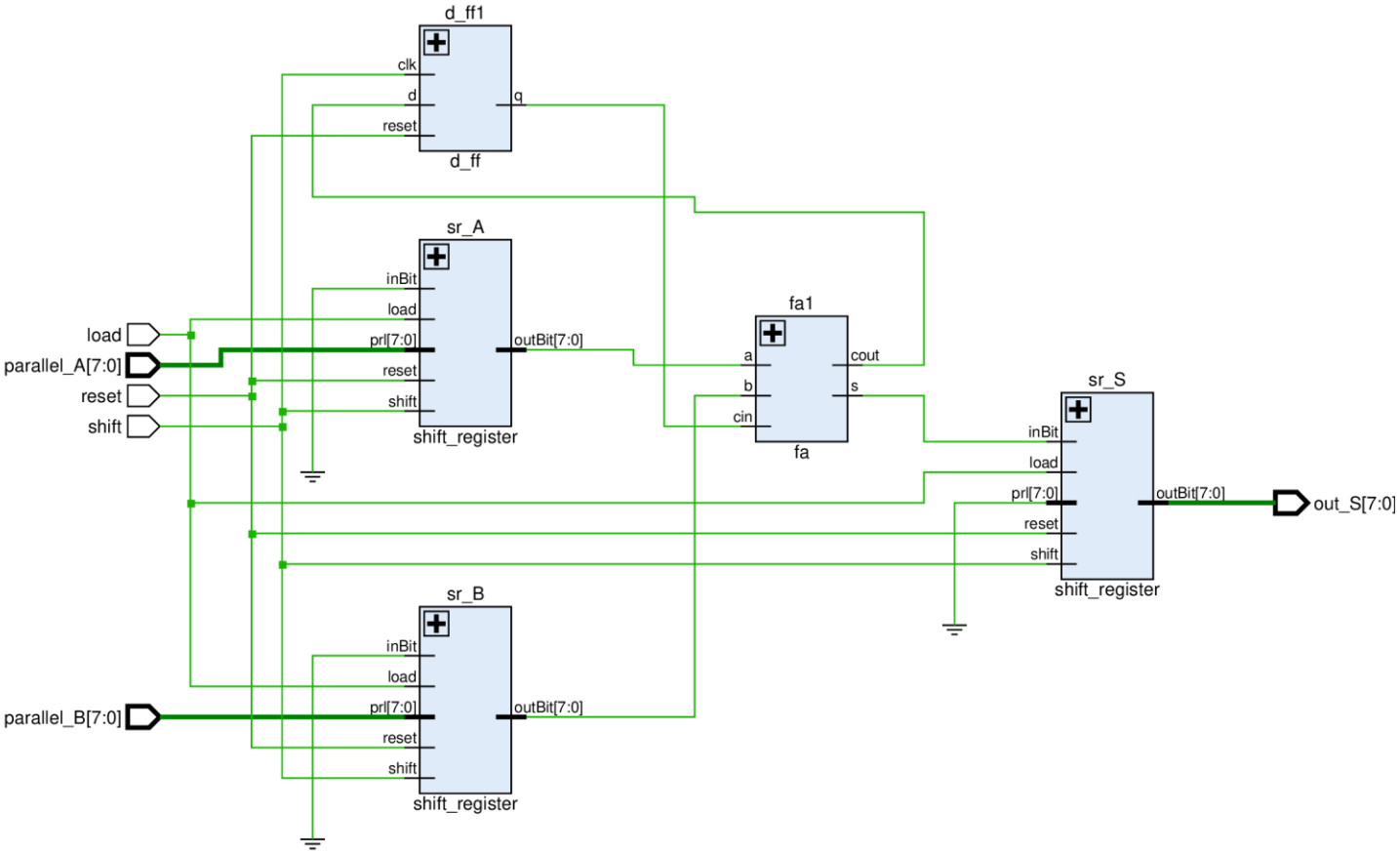
b) Circuit Schematic for Shift Register using D flip-flops.



SERIAL ADDER

CIRCUIT SCHEMATIC

c) Circuit Schematic of Serial Adder using Shift Registers, Full Adder, D flip-flop.



SHIFT REGISTER

D FLIP FLOP MODULE
STRUCTURAL MODULE
TESTBENCH

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

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

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

(e) Structural SystemVerilog module for your shift register using the D flip-flop module along

```
module shift_register(shift, reset, load, prl, inBit,
outBit);
    input logic shift, reset, load, inBit;
    input logic [7:0] prl;
    logic [7:0] d;
    output logic [7:0] outBit;

    always @ (posedge load)
        if (load) d = prl;

    d_ff f8(inBit, d[7], shift, reset);
    d_ff f7(d[7], d[6], shift, reset);
    d_ff f6(d[6], d[5], shift, reset);
    d_ff f5(d[5], d[4], shift, reset);
    d_ff f4(d[4], d[3], shift, reset);
    d_ff f3(d[3], d[2], shift, reset);
    d_ff f2(d[2], d[1], shift, reset);
    d_ff f1(d[1], d[0], shift, reset);
    assign outBit = d;
endmodule
```

(e) Structural SystemVerilog module for your shift register using the D flip-flop module along

```
module tb_sr();
    logic shift, reset, load, inBit;
    logic [7:0] outBit;
    logic [7:0] prl;

    shift_register dut (shift, reset, load, prl, inBit,
outBit);

        initial begin
            prl = 8'b01100110; shift = 0;
            inBit = 0; reset = 0; #10; load = 1; #10;
            load = 0; shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            prl = 8'b10101010; load = 1; #10;
            load = 0; shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            reset = 1; #10; reset= 0;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
        end
endmodule
```

SERIAL ADDER

STRUCTURAL MODULE TESTBENCH

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

```
module serial_adder(shift, reset, load, parallel_A, parallel_B,
out_S);
    input logic shift, reset, load;
    input logic [7:0] parallel_A;
    input logic [7:0] parallel_B;
    logic [7:0] out_A;
    logic [7:0] out_B;
    logic cin, cout, sum;
    output logic [7:0] out_S;

    shift_register sr_A(shift, reset, load, parallel_A, 0, out_A);
    shift_register sr_B(shift, reset, load, parallel_B, 0, out_B);
    shift_register sr_S(shift, reset, load, 0, sum, out_S);
    fa fa1(out_A[0], out_B[0], cin, sum, cout);
    d_ff d_ff1(cout, cin, shift, reset);
endmodule
```

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

```
module tb_sa();
logic shift, reset, load;
    logic [7:0] parallel_A;
    logic [7:0] parallel_B;
    logic [7:0] out_S;

    serial_adder dut (shift, reset, load, parallel_A, parallel_B,
out_S);

        initial begin
            reset = 1; #10;
            parallel_A = 8'b01001010; parallel_B = 8'b00110011;
shift = 0; reset = 0; #10; load = 1; #10;
            load = 0; shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            parallel_A = 8'b10101010; parallel_B = 8'b10101010;
load = 1; #10;
            load = 0; shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            reset = 1; #10; reset = 0;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            shift = 1; #10; shift = 0; #10;
            end
endmodule
```