CS 223 Digital Design

Bilkent University
Fall 2023-2023

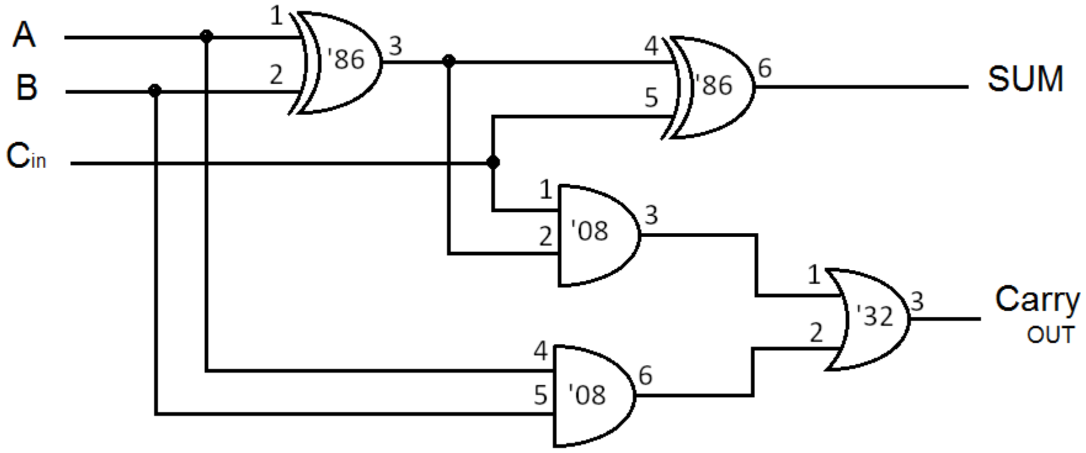
Laboratory Assignment 2
Preliminary Report

Section 2

Mehmet Akif Şahin
22203673

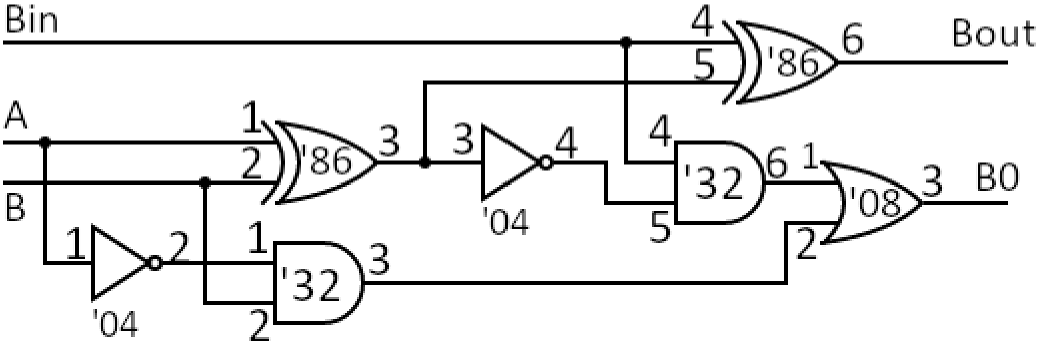
Sun. October 8th, 2023

b) Circuit schematic for full adder



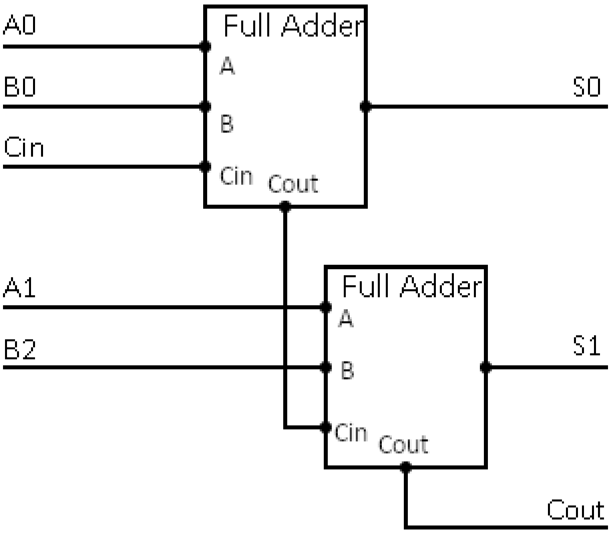
IC List: 1 7408 Quad-2 input AND gate, 1 7432 Quad-2 input OR gate, 1 7486 Quad-2 input XOR gate.
Vcc and GND connections: 7408, 7432, 7486 - 14. pin for Vcc and 7. pin for GND.

c) Circuit schematic for full subtractor



IC List: 1 7404 Hex-Inverting gate, 1 7408 Quad-2 input AND gate, 1 7432 Quad-2 input OR gate, 1 7486 Quad-2 input XOR gate.
Vcc and GND connections: 7404, 7408, 7432, 7486 - 14. pin for Vcc and 7. pin for GND.

d) Circuit schematic for 2-bit adder



(e) Behavioral SystemVerilog module for the full adder and a testbench for it.

Module

```
module full_adder_behavioral(
    input a,
    input b,
    input cin,
    output sum,
    output cout
    );
    assign sum = a ^ b ^ cin;
    assign cout = (a & b) | ((a ^ b) & cin);
endmodule
```

Testbench

```
module tb_full_adder_behavioral();
    logic a, b, cin, sum, cout;
    full_adder_behavioral uut (a, b, cin, sum, cout);

    initial begin
        a=0;
        b=0;
        cin=0;
        #10;
        a=1;
        b=0;
        cin=0;
        #10;
        a=0;
        b=1;
        cin=0;
        #10;
        a=1;
        b=1;
        cin=0;
        #10;
        a=0;
        b=0;
        cin=1;
        #10;
        a=1;
        b=0;
        cin=1;
        #10;
        a=0;
        b=1;
        cin=1;
        #10;
        a=1;
        b=1;
        cin=1;
        #10;
    end
endmodule
```

(f) Structural SystemVerilog module for the full adder and a testbench for it.

Module

```
module full_adder_structural(
    input a,
    input b,
    input cin,
    output sum,
    output cout
    );
    logic sum1, cout1, cout2;
    half_adder first_half_adder (.a(a), .b(b), .sum(sum1), .cout(cout1));
    half_adder second_half_adder (.a(cin), .b(sum1), .sum(sum), .cout(cout2));
    or (cout, cout1, cout2);
endmodule

module half_adder(
    input a,
    input b,
    output sum,
    output cout
    );
    xor (sum, a, b);
    and (cout, a ,b);
endmodule
```

Testbench

```
module tb_full_adder(

    );
     logic a, b, cin, sum, cout;
    full_adder_structural uut (a, b, cin, sum, cout);

    initial begin
        a=0;
        b=0;
        cin=0;
        #10;
        a=1;
        b=0;
        cin=0;
        #10;
        a=0;
        b=1;
        cin=0;
        #10;
        a=1;
        b=1;
        cin=0;
        #10;
        a=0;
        b=0;
        cin=1;
        #10;
        a=1;
        b=0;
        cin=1;
        #10;
        a=0;
        b=1;
        cin=1;
        #10;
        a=1;
        b=1;
        cin=1;
        #10;
    end
endmodule
```

(g) Structural SystemVerilog module for the full subtractor and a testbench for it.

Module

```
module full_subtractor(
    input a,
    input b,
    input bin,
    output diff,
    output bout
    );
    logic diff1, bout1, bout2;
    half_subtractor first_half_subtractor (.a(a), .b(b), .diff(diff1), .bout(bout1));
    half_subtractor second_half_subtractor (.a(diff1), .b(bin), .diff(diff), .bout(bout2));
    or (bout, bout1, bout2);
endmodule

module half_subtractor(
    input a,
    input b,
    output diff,
    output bout
    );
    logic a_compelement;
    xor (diff, a, b);
    not (a_complement, a);
    and (bout, a_complement, b);
endmodule
```

Testbench

```
module tb_full_subtractor(

    );
     logic a, b, bin, diff, bout;
    full_subtractor uut (a, b, bin, diff, bout);

    initial begin
        a=0;
        b=0;
        bin=0;
        #10;
        a=1;
        b=0;
        bin=0;
        #10;
        a=0;
        b=1;
        bin=0;
        #10;
        a=1;
        b=1;
        bin=0;
        #10;
        a=0;
        b=0;
        bin=1;
        #10;
        a=1;
        b=0;
        bin=1;
        #10;
        a=0;
        b=1;
        bin=1;
        #10;
        a=1;
        b=1;
        bin=1;
        #10;
    end
endmodule
```

(h) Structural SystemVerilog module for the 2-bit adder and a testbench for it. Use the full adder module you wrote in part (f).

Module

```
module two_bit_adder(
    input a[1:0],
    input b[1:0],
    input cin,
    output sum[1:0],
    output cout
    );
    logic cout1;
    full_adder_structural first_full_adder(.a(a[0]), .b(b[0]), .cin(cin), .sum(sum[0]),
.cout(cout1));
    full_adder_structural second_full_adder(.a(a[1]), .b(b[1]), .cin(cout1), .sum(sum[1]),
.cout(cout));
endmodule
```

Testbench

```
module tb_two_bit_adder(

    );
    logic a[1:0], b[1:0], cin, sum[1:0], cout;
    two_bit_adder uut (.a(a[1:0]),.b(b[1:0]),.cin(cin),.sum(sum),.cout(cout));
    initial begin
        for (int i = 0; i < 32; i = i + 1) begin
            {a[1], a[0], b[1], b[0], cin} = i;
        end
    end
endmodule
```