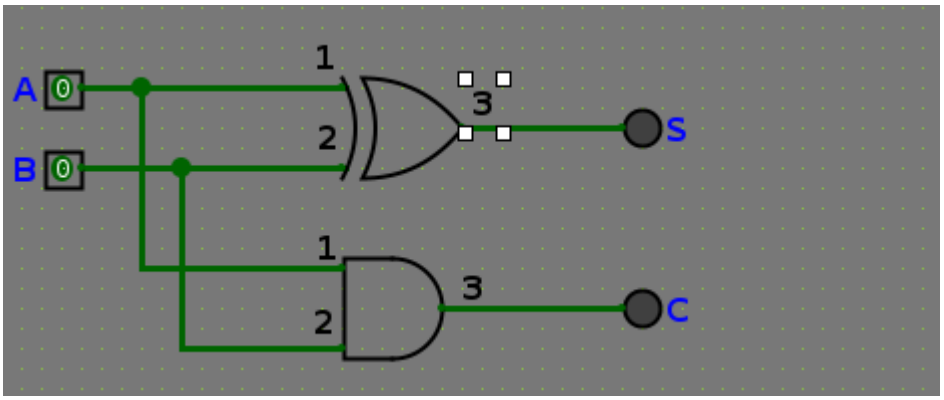
CS223
Digital Design
Section 1
Lab2

Erkin Aydın
ID: 22002956

17/10/2022

Circuit Schematic for a half adder:



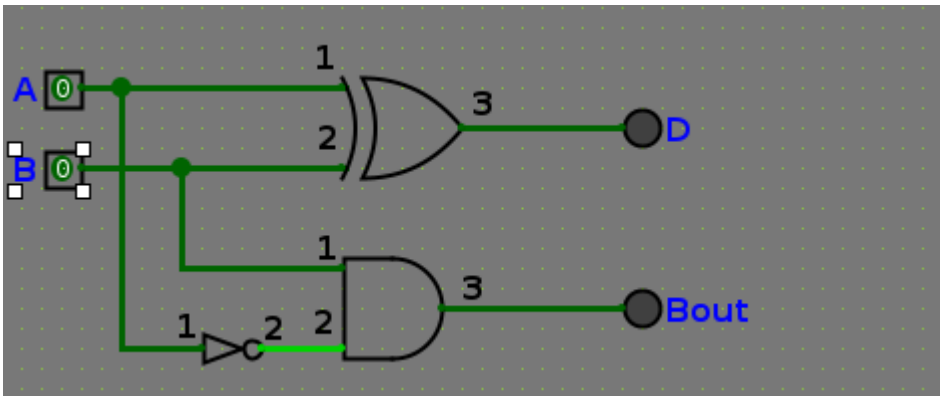
IC List for Circuit Schematic:

1)One 7486 Quad 2-Input XOR Gate
2)One 7408 Quad 2-Input AND Gate

Pin Numbers for GND and +V for Circuit Schematic:

7486 Quad 2-Input XOR Gate: GND-7, +5V-14
7408 Quad 2-Input AND Gate: GND-7, +5V-14

Circuit Schematic for a half subtractor:



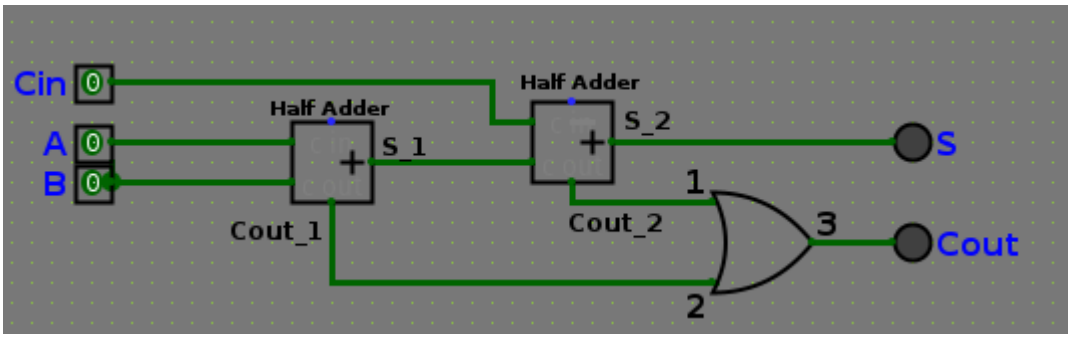
IC List for Circuit Schematic:

1)One 7486 Quad 2-Input XOR Gate
2)One 7408 Quad 2-Input AND Gate
3)7404 INVERTER

Pin Numbers for GND and +V for Circuit Schematic:

7486 Quad 2-Input XOR Gate: GND-7, +5V-14
7408 Quad 2-Input AND Gate: GND-7, +5V-14
7404 INVERTER: GND-7, +5V-14

Circuit Schematic for a full adder:



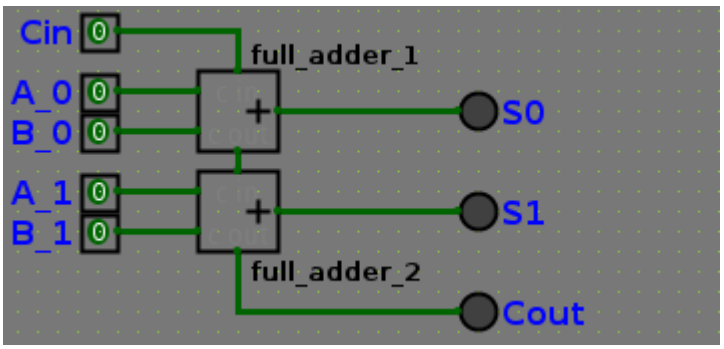
IC List for Circuit Schematic:

1)One 7486 Quad 2-Input XOR Gate
2)One 7408 Quad 2-Input AND Gate
3)One 7432 Quad 2-Input OR Gate

Pin Numbers for GND and +V for Circuit Schematic:

7486 Quad 2-Input XOR Gate: GND-7, +5V-14
7408 Quad 2-Input AND Gate: GND-7, +5V-14
7432 Quad 2-Input OR Gate: GND-7, +5V-14

Circuit Schematic for a 2-bit adder:



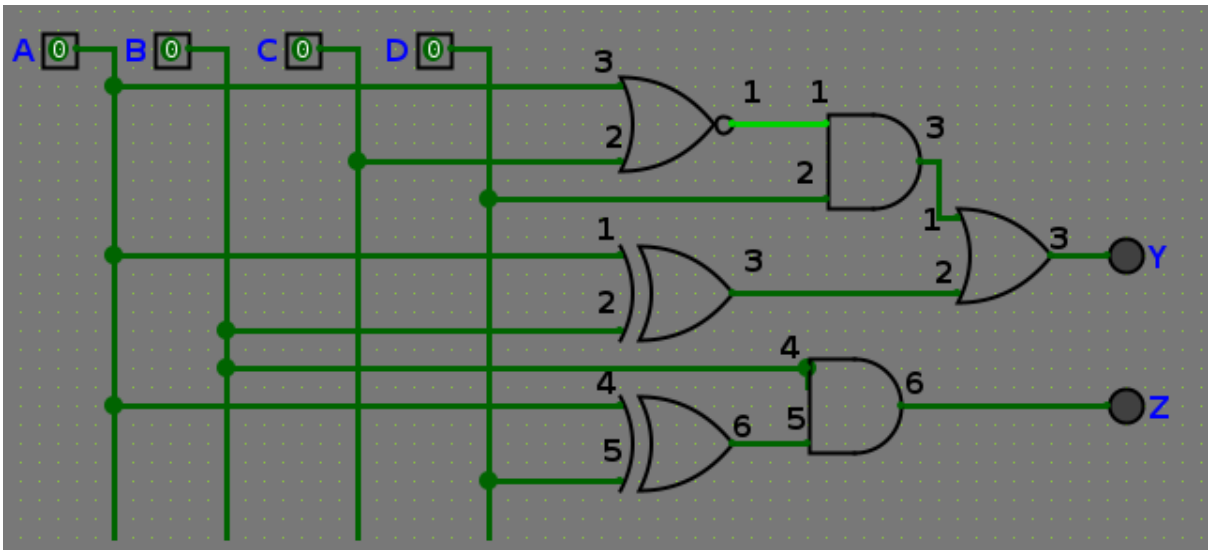
IC List for Circuit Schematic:

1)One 7486 Quad 2-Input XOR Gate
2)One 7408 Quad 2-Input AND Gate
3)One 7432 Quad 2-Input OR Gate

Pin Numbers for GND and +V for Circuit Schematic:

7486 Quad 2-Input XOR Gate: GND-7, +5V-14
7408 Quad 2-Input AND Gate: GND-7, +5V-14
7432 Quad 2-Input OR Gate: GND-7, +5V-14

Circuit Schematic for a lab calculator:



IC List for Circuit Schematic:

1)One 7486 Quad 2-Input XOR Gate
2)One 7408 Quad 2-Input AND Gate
3)One 7432 Quad 2-Input OR Gate
4)One 7428 Quad 2-Input NOR Gate

Pin Numbers for GND and +V for Circuit Schematic:

7486 Quad 2-Input XOR Gate: GND-7, +5V-14
7408 Quad 2-Input AND Gate: GND-7, +5V-14
7432 Quad 2-Input OR Gate: GND-7, +5V-14
7428 Quad 2-Input NOR Gate: GND-7, +5V-14

Half Adder Behavioral Module:

```
//Implementation of half_adder
module half_adder(input a, b, output s, c);

        assign s = a ^ b;
        assign c = a & b;

endmodule
```

Half Adder Testbench:

```
module half_adder_testbench();
        logic a, b, s, c;
        half_adder ha(a,b,s,c);
```

```
        initial begin
        assign a = 0;
        assign b = 0;
        #100;

        assign b= 1; #100;

        assign a = 1;
        assign b = 0; #100;

        assign b = 1; #100;

        end
endmodule
```

Half Subtractor Behavioral Module:

```
//Implementation of half_subtractor
module half_subtractor(input a, b, output d, bout);

        assign d = a ^ b;
        assign bout = (~a) & b;

endmodule
```

Half Subtractor Testbench:

```
module half_subtractor_tb();
        logic a, b, d, bout;
        half_subtractor hs(a,b,d,bout);
        initial begin
        assign a = 0;
        assign b = 0;
        #100;

        assign b= 1; #100;

        assign a = 1;
        assign b = 0; #100;

        assign b = 1; #100;

        end
```

```
endmodule
```

Full Adder Structural Module:

```
//Implementation of full_adder
module full_adder(input a, b, cin, output s, cout);

        logic s_1, cout_1, s_2, cout_2;

        half_adder ha_1(a, b, s_1, cout_1);
        half_adder ha_2(cin, s_1, s_2, cout_2);

        assign s = s_2;
        or o( cout, cout_1, cout_2);

endmodule
```

Full Adder Testbench:

```
module full_adder_tb();

        logic a, b, cin, s, cout;
        full_adder fa(a, b, cin, s, cout);

        initial begin
        assign a = 0;
        assign b = 0;
        assign cin = 0; #100;

        assign cin = 1; #100;

        assign b = 1;
        assign cin = 0; #100;

        assign cin = 1; #100;

        assign a = 1;
        assign b = 0;
        assign cin = 0; #100

        assign cin = 1; #100;

        assign b = 1;
```

```
        assign cin = 0; #100;

        assign cin = 1; #100;

        end
endmodule
```

2-Bit Adder Structural Module:

```
//Implementation of two_bit_adder
module two_bit_adder(input logic [1:0]a,
                     input logic [1:0]b,
                     input logic cin,
                     output logic [1:0]s,
                     output logic cout);

        logic sum_1, sum_2, carry_1, carry_2;

        full_adder fa_1(a[0], b[0], cin, sum_1, carry_1);
        full_adder fa_2(a[1], b[1], carry_1, sum_2, carry_2);

        assign s[0] = sum_1;
        assign s[1] = sum_2;
        assign cout = carry_2;

endmodule
```

2-Bit Adder Testbench:

```
module two_bit_adder_tb();

        reg [1:0]a;
        reg [1:0]b;
        logic cin;
        reg [1:0]s;
        logic cout;

        two_bit_adder two_add(a, b, cin, s, cout);

        initial begin

        assign a = 2'b00;
        assign b = 2'b00;
```

```
assign cin = 0; #20;

assign cin = 1; #20;

assign b = 2'b01;
assign cin = 0; #20;

assign cin = 1; #20;

assign b = 2'b10;
assign cin = 0; #20;

assign cin = 1; #20;

assign b = 2'b11;
assign cin = 0; #20;

assign cin = 1; #20;

//

assign a = 2'b01;
assign b = 2'b00;
assign cin = 0; #20;

assign cin = 1; #20;

assign b = 2'b01;
assign cin = 0; #20;

assign cin = 1; #20;

assign b = 2'b10;
assign cin = 0; #20;

assign cin = 1; #20;

assign b = 2'b11;
assign cin = 0; #20;

assign cin = 1; #20;

//
```

```
        assign a = 2'b10;
        assign b = 2'b00;
        assign cin = 0; #20;

        assign cin = 1; #20;

        assign b = 2'b01;
        assign cin = 0; #20;

        assign cin = 1; #20;

        assign b = 2'b10;
        assign cin = 0; #20;

        assign cin = 1; #20;

        assign b = 2'b11;
        assign cin = 0; #20;

        assign cin = 1; #20;

        //
        assign a = 2'b11;
        assign b = 2'b00;
        assign cin = 0; #20;

        assign cin = 1; #20;

        assign b = 2'b01;
        assign cin = 0; #20;

        assign cin = 1; #20;

        assign b = 2'b10;
        assign cin = 0; #20;

        assign cin = 1; #20;

        assign b = 2'b11;
        assign cin = 0; #20;

        assign cin = 1; #20;
        end
endmodule
```

Lab Calculator Structural Module:

```
//Implementation of lab calculator
module lab_calculator(input logic a,
                      input logic b,
                      input logic c,
                      input logic d,
                      output logic y,
                      output logic z);

        logic out_1, out_2, out_3, out_4;
        logic carry_1, bout_1;

        xor in_1(out_1, a, b);
        nand in_2(out_2, a, b);
        half_adder in_3(a, b, out_3, carry_1);
        half_subtractor in_4(a, b, out_4, bout_1);

        mux4 mux_1(out_4, out_3, out_2, out_1, c, d, y);
        mux2 mux_2(bout_1, carry_1, d, z);

endmodule
```

Lab Calculator Testbench:

```
module lab_calculator_tb();

        logic a, b, c, d, y, z;

        lab_calculator lb(a, b, c, d, y, z);

        initial begin

        assign a = 0;
        assign b = 0;
        assign c = 0;
        assign d = 0; #20

        assign d = 1; #20;

        assign c = 1;
        assign d = 0; #20;
```

```
        assign d = 1; #20;

        //
        assign a = 0;
        assign b = 1;
        assign c = 0;
        assign d = 0; #20

        assign d = 1; #20;

        assign c = 1;
        assign d = 0; #20;

        assign d = 1; #20;

        //
        assign a = 1;
        assign b = 0;
        assign c = 0;
        assign d = 0; #20

        assign d = 1; #20;

        assign c = 1;
        assign d = 0; #20;

        assign d = 1; #20;

        //
        assign a = 1;
        assign b = 1;
        assign c = 0;
        assign d = 0; #20

        assign d = 1; #20;

        assign c = 1;
        assign d = 0; #20;

        assign d = 1; #20;
        end

endmodule
```