CS223

Digital Design

Section: 1

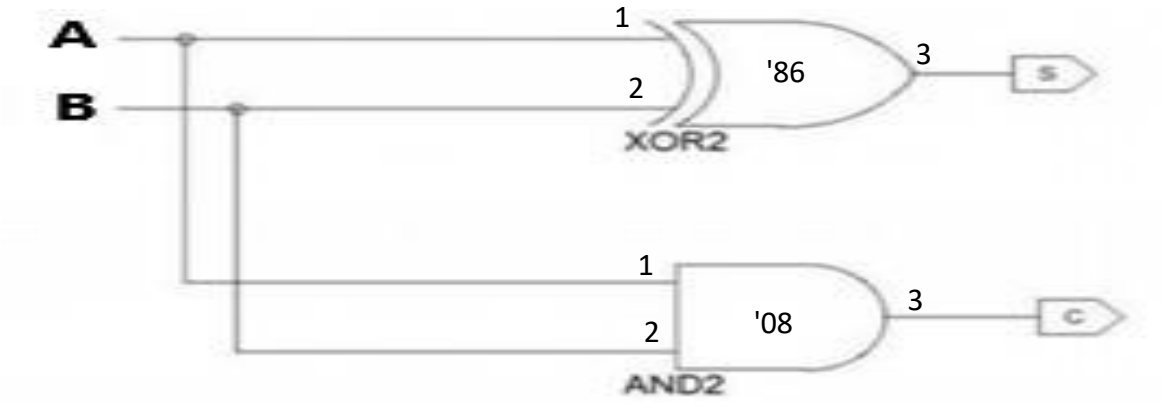
Lab-02

Name: Tolga Han Arslan

ID: 22003061

17.10.2022

b)Circuit schematic for half adder:

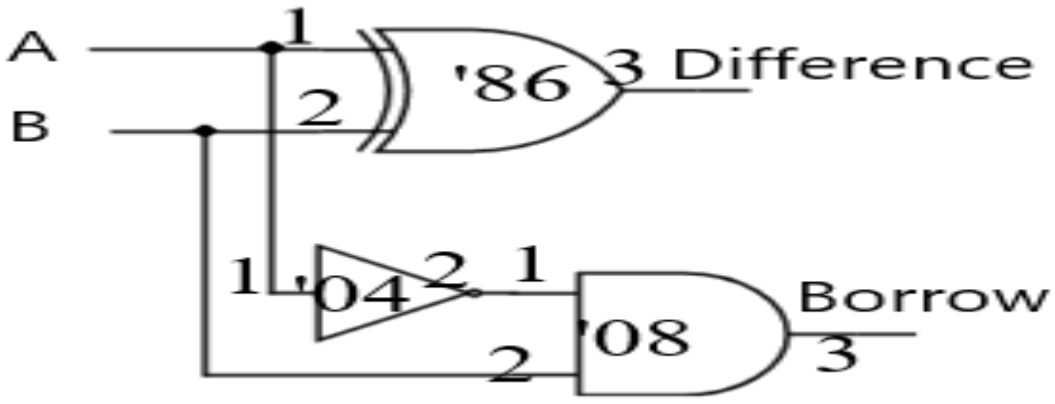


IC List:

1- One 74LS86 quad 2-input XOR gate
2- One 74LS08 quad 2-input AND gate

*74LS86	*74LS08
GND-7	GND-7
+5V-14	+5V-14

c)Circuit schematic for half subtractor:

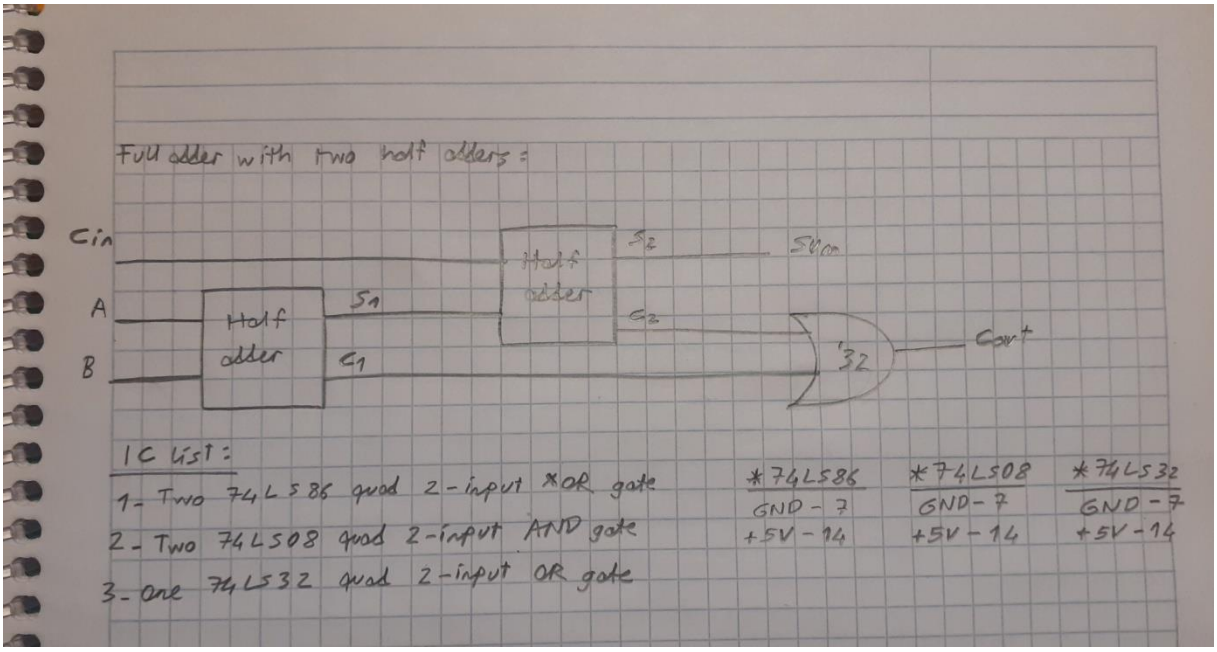


IC List: *74LS86 *74LS08 *74LS04

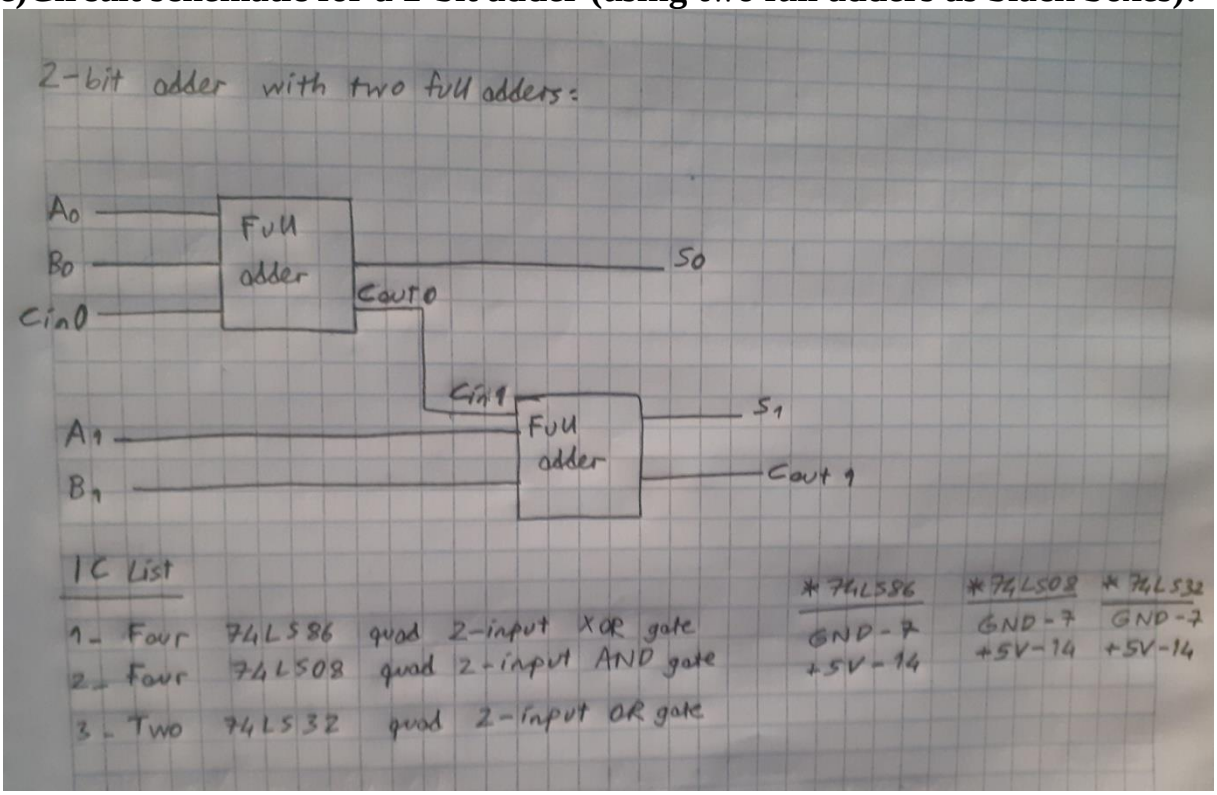
1-One 74LS86 quad 2-input XOR gate	GND-7	GND-7	GND-7
2-One 74LS32 quad 2-input OR gate	+5V-14	+5V-14	+5V-14

3-One 74LS04 HEXInverting gate

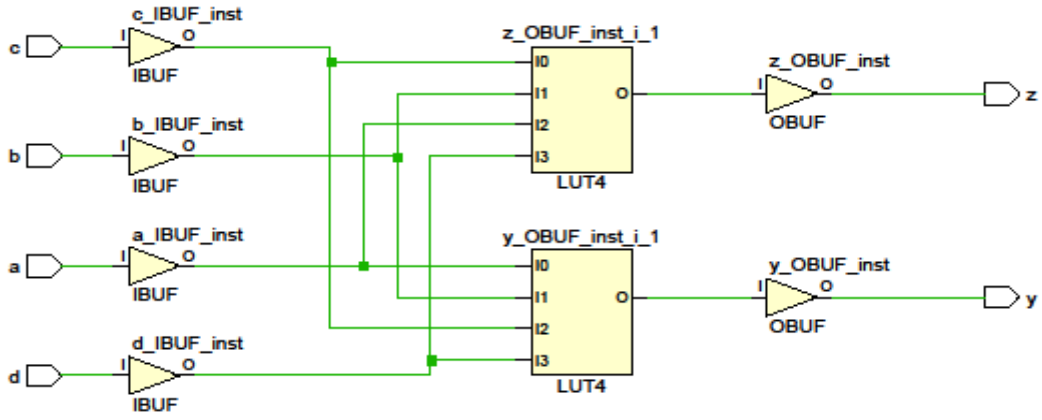
d)Circuit schematic for a full adder (using two half adders as black boxes):



e)Circuit schematic for a 2-bit adder (using two full adders as black boxes):



f)Circuit schematic for the lab calculator (refer Figure 1-2):



g)Behavioral SystemVerilog module for the half adder and a testbench for it:

```
module half_adder (input logic a, b, output logic s, c);
   assign s = a ^ b;
   assign c = a & b;
endmodule
```

```
module half_adder_testbench();
   logic a, b, s, c;
   half_adder dut(a, b, s, c);
   initial begin
      a = 0; b = 0; #10;
      b = 1;  #10;
      a = 1; b = 0; #10;
      b = 1; #10;
   end
endmodule
```

h)**Behavioral SystemVerilog module for the half subtractor and a testbench for it:**

```
module half_subtractor( input logic a, b, output logic d, bout);
   assign d = a^b;
   assign bout = ~a & b;
endmodule


//testbench module
module half_subtractor_testbench();
   logic a, b, d, bout;
   half_subtractor dut(a,b,d,bout);
```

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

i)Structural SystemVerilog module for the full adder and a testbench for it. Use the half adder module you wrote in part (g):

```
module full_adder(input logic a, b, cin, output logic sum, cout);
    logic s1, c1, c2;
    half_adder adder1(a,b,s1,c1);
    half_adder adder2(cin, s1,sum ,c2);
    assign cout = c2 | c1;
endmodule

    //testbench
module full_adder_testbench();
    logic a,b,cin, sum, cout;
    full_adder dut(a,b,cin,sum,cout);
    initial begin
        a=0; b=0; cin=0; #10;
        cin=1; #10
        b=1; cin=0; #10;
        cin=1; #10;
        a=1; b=0; cin=0; #10;
        cin = 1; #10;
        b=1; cin=0; #10;
        cin=1; #10;
    end
endmodule
```

j)Structural SystemVerilog module for the 2-bit adder and a testbench for it. Use the full adder module you wrote in part (i): // partial testbench

```
module two_bit_adder(input logic a[1:0], b[1:0], cin, output logic s[1:0], cout);
   logic cin1;
   full_adder adder1(a[0], b[0], cin, s[0], cin1);
   full_adder adder2(a[1], b[1], cin1, s[1], cout);
endmodule

//testbench
module two_bit_adder_testbench();
   logic a[1:0], b[1:0], cin, s[1:0], cout;
   two_bit_adder dut(a[1:0],b[1:0],cin,s[1:0],cout);
   initial begin
      a[0]=0; a[1]=0; b[0]=0; b[1]=0; cin=0; #10;
      a[0] = 1; #10;
      a[0] = 0; a[1] = 1; #10;
      a[0] = 1; a[1] = 0; b[0] = 1; #10;
      a[0] = 0; a[1] =1; b[0] = 0; b[1] = 1; #10;
      b[1] = 0; cin = 1;   #10;
      a[0] = 1; b[0] = 1; b[1] = 0; #10;
      a[1] = 1; b[1] = 1; #10;
   end
endmodule
```

k)Structural SystemVerilog module for the lab calculator and a testbench for it. (Use your half-adder, and half subtractor as the building blocks for your highest level module):

```
module lab_calculator(input logic a,b,c,d, output logic y,z);
   assign y = (a^b) & ~c&~d | ~(a&b) & ~c&d | (a^b) & c&~d | (a^b) & c&d
   assign z = (a&b) & c & ~d | (~a&b) &c&d
endmodule

module lab_calculator_testbench();
   logic a,b,c,d,y,z;
   lab_calculator dut(a,b,c,d,y,z);
   initial begin
      a=0; b=0; c=0; d=0; #10;
      b=1; #10;
      a=1; b=0; #10;
      b=1; #10;
      a=0; b=0; d=1; #10;
      b=1; #10;
      b=0; a=1; #10;
      b=1; #10;
      a=0; b=0; d=0; c=1; #10;
      b=1; #10;
      a=1; b=0; #10;
      b=1; #10;
      d=1; a=0; b=0; #10;
      b=1; #10;
      a=1; b=0; #10;
      b=1; #10;
```