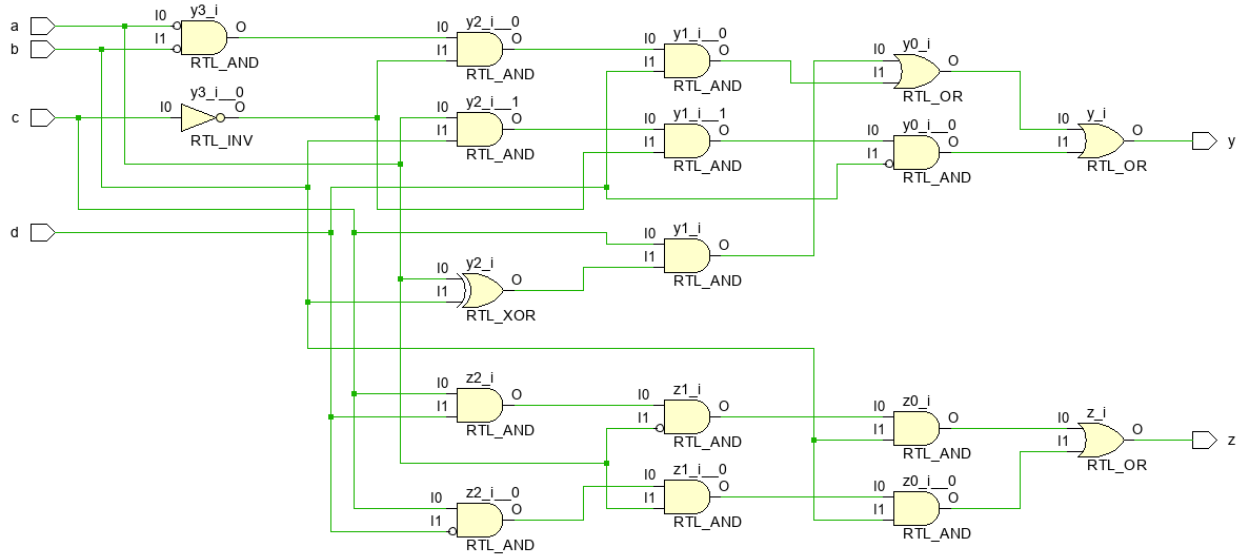
2021–2022 FALL SEMESTER

CS223 – LAB 02

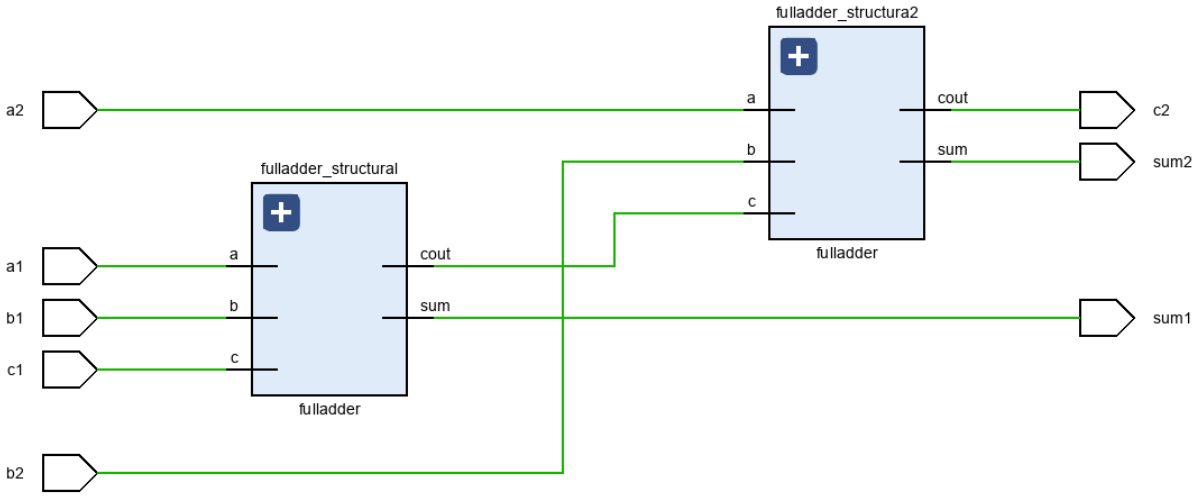
Full adder, 2-bit adder, lab calculator on FPGA

NAME: ABDULLAH RIAZ
ID : 22001296
COURSE:CS223 – DIGTAL DESGIN
SECTION: 01
DATE: 17/10/2021

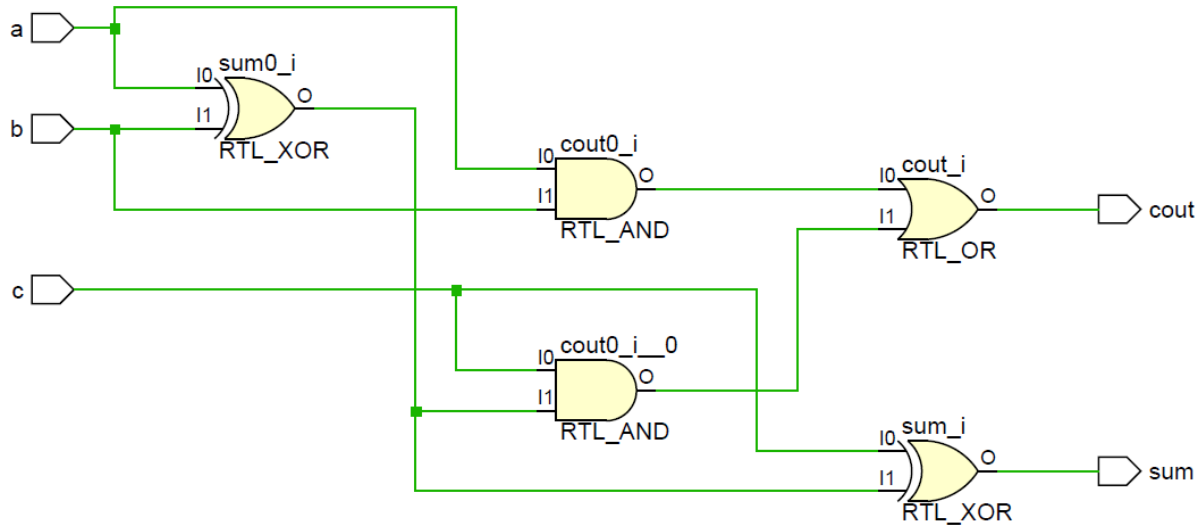
b) **Circuit schematic for full adder using 2-input XOR, AND, and OR gates**



c) **Circuit schematic for a 2-bit adder**



d) **Circuit schematic for lab calculator**



e) **Full Adder**

- **Behavioral SystemVerilog module**

```
module fulladder(

   input logic a, b, c,
   output logic sum, cout
   );

   assign sum = c ^ (a^b);
   assign cout = a & b | c & (a ^ b);
endmodule
```

- **TestBench module**

```
module fulladder_tb();
logic a, b, c , sum, cout;

fulladder dut(a, b, c, sum, cout);

   initial begin
      a = 0; b = 0; c=0; #10;
      a = 0; b = 0; c=1; #10;
      a = 0; b = 1; c=0; #10;
```

```
    a = 0; b = 1; c=1; #10;
    a = 1; b = 0; c=0; #10;
    a = 1; b = 0; c=1; #10;
    a = 1; b = 1; c=0; #10;
    a = 1; b = 1; c=1; #10;
    end
endmodule
```

f) **Full Adder**

- **Structural SystemVerilog module**

```
module xor2(input logic a,b, output logic c);
assign c = a ^ b;
endmodule

module and2(input logic a,b, output logic c);
assign c = a & b;
endmodule

module or2(input logic a,b, output logic c);
assign c = a | b;
endmodule


module fulladder_structural(
input logic a, b, c,
output logic sum, cout
);

logic o1,o2,o3;
xor2 i1(a,b,o1);
xor2 i2(c,o1,sum);
and2 i3(a,b,o2);
and2 i4(c,o1,o3);
or2 i5(o2,o3,cout);


endmodule
```

- **Testbench**

```
module fulladder_structural_tb();
logic a, b, c , sum, cout;

fulladder_structural dut(a, b, c, sum, cout);

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

g) **2-bit adder**

- **Structural SystemVerilog module**

```
module two_bit_fulladder(
input logic a1,b1,c1,a2,b2,
output logic sum1, sum2, c2
);

logic cout;

//1 bit adder
fulladder fulladder_structural(a1,b1,c1,sum1,cout);
fulladder fulladder_structura2(a2,b2,cout, sum2, c2);
endmodule
```

- **Testbench**

```
module two_bit_fb_tb();
 logic a1,b1,c1,a2,b2,sum1,sum2,c2;

 two_bit_fulladder dut(a1,b1,c1,a2,b2, sum1, sum2, c2);

//Partial Truth Table
  initial begin
      a1 = 0; a2 = 0; b1=0; b2=0; c1=0; #10;
      a1 = 0; a2 = 0; b1=0; b2=0; c1=1; #10;
      a1 = 0; a2 = 0; b1=0; b2=1; c1=0; #10;
      a1 = 0; a2 = 0; b1=0; b2=1; c1=1; #10;
      a1 = 0; a2 = 0; b1=1; b2=0; c1=0; #10;
      a1 = 0; a2 = 0; b1=1; b2=0; c1=1; #10;
      a1 = 0; a2 = 0; b1=1; b2=1; c1=0; #10;
      a1 = 0; a2 = 0; b1=1; b2=0; c1=1; #10;
      a1 = 1; a2 = 0; b1=0; b2=0; c1=0; #10;
      a1 = 1; a2 = 0; b1=0; b2=0; c1=1; #10;
      a1 = 1; a2 = 0; b1=0; b2=1; c1=0; #10;
      a1 = 1; a2 = 0; b1=0; b2=1; c1=1; #10;
      a1 = 1; a2 = 0; b1=1; b2=0; c1=0; #10;
  end
endmodule
```

h) **Lab Calculator**

- **Structural SystemVerilog module**

```
module labcalc(input logic a,b,c,d, output logic y,z);

assign z =  ( c & d & ~a & b ) | ( c & ~d & a & b );
assign y = c & ( a ^ b ) | ~a & ~b & ~c & d | a & b & ~c & ~d;
endmodule
```

- **testbench**

```
module labcalc_tb();
  logic a,b,c,d,y,z;
  labcalc calc(a,b,c,d,y,z);

 initial begin
     c = 0; d = 0; a=0; b=0; #10;
     assert (y === 0) else $error("0000 failed.");
     c = 0; d = 0; a=0; b=1; #10;
     assert (y === 0) else $error("0001 failed.");
     c = 0; d = 0; a=1; b=0; #10;
     assert (y === 0) else $error("0010 failed.");
     c = 0; d = 0; a=1; b=1; #10;
     assert (y === 1) else $error("0011 failed.");
     c = 0; d = 1; a=0; b=0; #10;
     assert (y === 1) else $error("0100 failed.");
     c = 0; d = 1; a=0; b=1; #10
     assert (y === 0) else $error("0101 failed.");
     c = 0; d = 1; a=1; b=0; #10;
     assert (y === 0) else $error("0110 failed.");
     c = 0; d = 1; a=1; b=1; #10;
```

```
    assert (y === 0) else $error("0111 failed.");
    c = 1; d = 0; a=0; b=0; #10;
    assert (z === 0 & y === 0) else $error("1000 failed.");
    c = 1; d = 0; a=0; b=1; #10;
    assert (z === 0 & y === 1 ) else $error("1001 failed.");
    c = 1; d = 0; a=1; b=0; #10;
    assert (z === 0 & y === 1) else $error("1010 failed.");
    c = 1; d = 0; a=1; b=1; #10;
    assert (z === 1 & y === 0) else $error("1011 failed.");
    c = 1; d = 1; a=0; b=0; #10;
    assert (z === 0 & y === 0) else $error("1100 failed.");
    c = 1; d = 1; a=0; b=1; #10;
    assert (z === 1 & y === 1) else $error("1101 failed.");
    c = 1; d = 1; a=1; b=0; #10;
    assert (z === 0 & y === 1) else $error("1110 failed.");
    c = 1; d = 1; a=1; b=1; #10;
    assert (z === 0 & y === 0) else $error("1111 failed.");
  end

endmodule
```