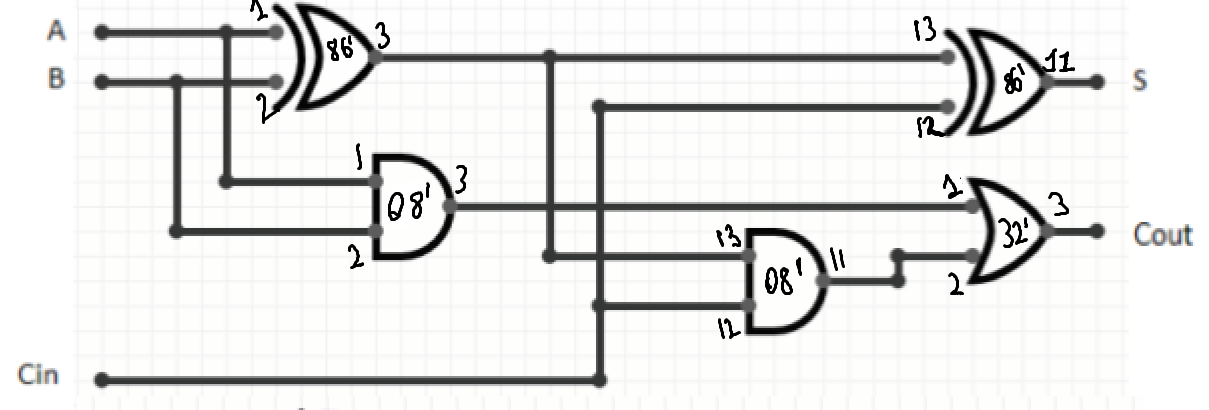
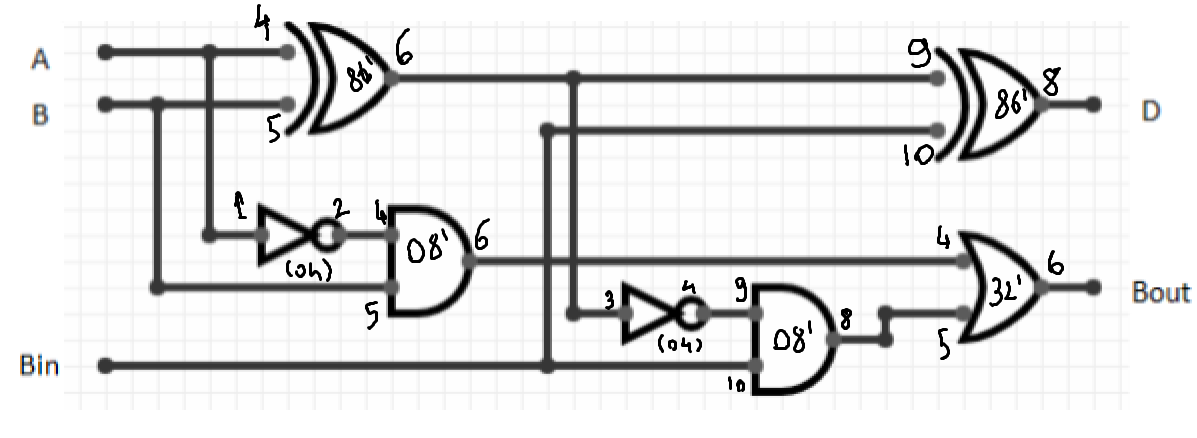
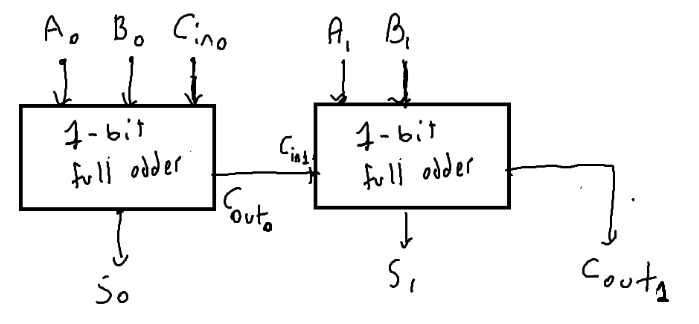
**CS223 Preliminary Report**

**2)** Circuit Schematics for the 1-bit Full Adder (above) and 1-bit Full Substractor (below), IC numbers and pin numbers were specified in the first report:

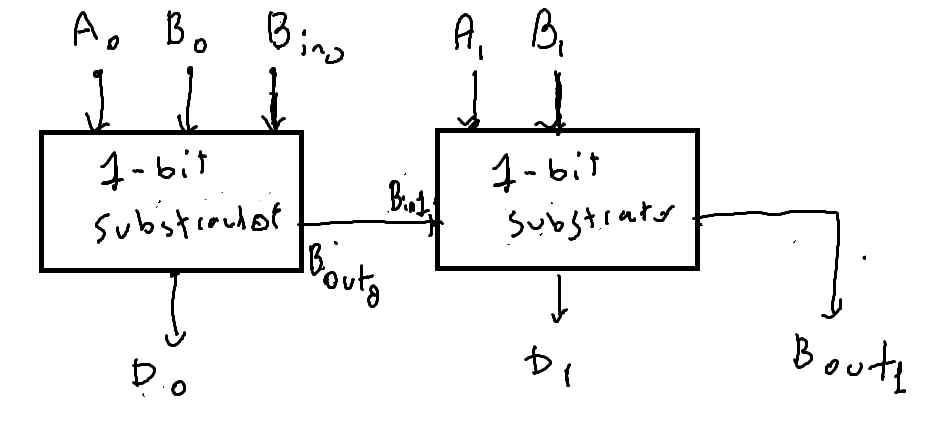




**3)** Schematic for a 2-bit full adder & subtractor made from two 1-bit full adder & subtractors.



(2-bit full adder schematics)



(2 bits substractor schematics)

**4)** Dataflow Systemverilog module for the 1-bit full adder & subtractor.

module b\_full\_adder(

input logic a, b, c,

output logic sum, carry

);

assign sum = a ^ b ^ c;

assign carry = (a & b) | (c & (a ^ b) );

endmodule

module b\_full\_substractor(

input logic a, b, B\_in,

output logic diff, borrow

);

assign diff = a ^ b ^ B\_in;

assign borrow = (~a & B\_in) | (~a & b) | (b & B\_in);

endmodule

**5)** Structural Systemverilog module for the 1-bit full adder & subtractor and a testbench for

them.

**Note:** half\_substractor and half\_adder modules are built in order to write structural modules. In case there is a confusion about how they work, their code is also shared.

\*

BEHAVIOURAL HALF\_ADDER

\*/

module half\_adder(

input logic a, b,

output logic sum, carry

);

assign sum = a ^ b;

assign carry = a & b;

endmodule

/\*

BEHAVIOURAL HALF\_SUBSTRACTOR

\*/

module half\_substractor(

input logic a, b,

output logic diff, borrow

);

assign diff = a ^ b;

assign borrow = b & ~a;

endmodule

/\*

STRUCTURAL FULL SUBSTRACTOR

\*/

module s\_full\_substractor(

input a, b, B\_in,

output diff, borrow

);

logic d1, b1, b2;

half\_substractor Hs1(a, b, d1, b1);

half\_substractor Hs2(d1,B\_in, diff, b2);

or(borrow, b1, b2);

endmodule

/\*

STRUCTURAL FULL ADDER

\*/

module s\_full\_adder(

input a, b, c,

output sum, carry

);

logic s1, c1, c2;

half\_adder H1(a, b, s1, c1);

half\_adder H2(s1, c, sum, c2);

or(carry, c1, c2);

endmodule

/\*

TESTBENCH01

\*/

module TestBench01();

logic a, b, c;

logic sum, carry, diff, borrow;

s\_full\_adder Fa01(a, b, c, sum, carry);

s\_full\_substractor Fs01(a, b, c, diff, borrow);

// apply inputs one at a time

initial begin

a = 0; b = 0; c = 0; #10;

c = 1; #10;

b = 1; c = 0; #10;

c = 1; #10;

a = 1; b = 0; c = 0; #10;

c = 1; #10;

b = 1; c = 0; #10;

c = 1; #10;

end

endmodule

**6)** Structural Systemverilog module for the 2-bit full adder & subtractor, and a testbench for them.

/\*

Structural 2-bit adder

\*/

module full\_adder\_2bit(

input logic a0, a1, b0, b1, c\_in,

output logic r0, r1, carry

);

logic c0;

s\_full\_adder sFa01(a0, b0, c\_in, r0, c0);

s\_full\_adder sFa02(a1, b1, c0, r1, carry);

endmodule

/\*

Structural 2-bit substractor

\*/

module full\_substractor\_2bit(

input logic a0, a1, b0, b1,b\_in,

output logic r0, r1, borrow

);

logic borrow\_0;

s\_full\_substractor Fs01(a0, b0, b\_in, r0, borrow\_0);

s\_full\_substractor Fs02(a1, b1, borrow\_0, r1, borrow);

endmodule

/\*

TestBench for 2bit substractor and adder

\*/

module TestBench02();

logic a0, a1, b0, b1;

logic s0, s1, c, d0, d1, b;

full\_adder\_2bit Fa(a0, a1, b0, b1, 0, s0, s1, c);

full\_substractor\_2bit Fs(a0, a1, b0, b1, 0, d0, d1, b);

// apply inputs one at a time

initial begin

a1 = 0; a0 = 0; b1 = 0; b0 = 0; #10; // 00 - 00

b0 = 1; #10; // 00 - 01

b1 = 1; b0 = 0; #10; // 00 - 10

b0 = 1; #10; // 00 - 11

a0 = 1; b1 = 0; b0 = 0; #10; // 01 - 00

b0 = 1; #10; // 01 - 01

b1 = 1; b0 = 0; #10; // 01 - 10

b0 = 1; #10; // 01 - 11

a1 = 1; a0 = 0; b1 = 0; b0 = 0; // 10 - 00

b0 = 1; #10; // 10 - 01

b1 = 1; b0 = 0; #10; // 10 - 10

b0 = 1; #10; // 10 - 11

a0 = 1; b1 = 0; b0 = 0; #10; // 11 - 00

b0 = 1; #10; // 11 - 01

b1 = 1; b0 = 0; #10; // 11 - 10

b0 = 1; #10; // 11 - 11

end

endmodule