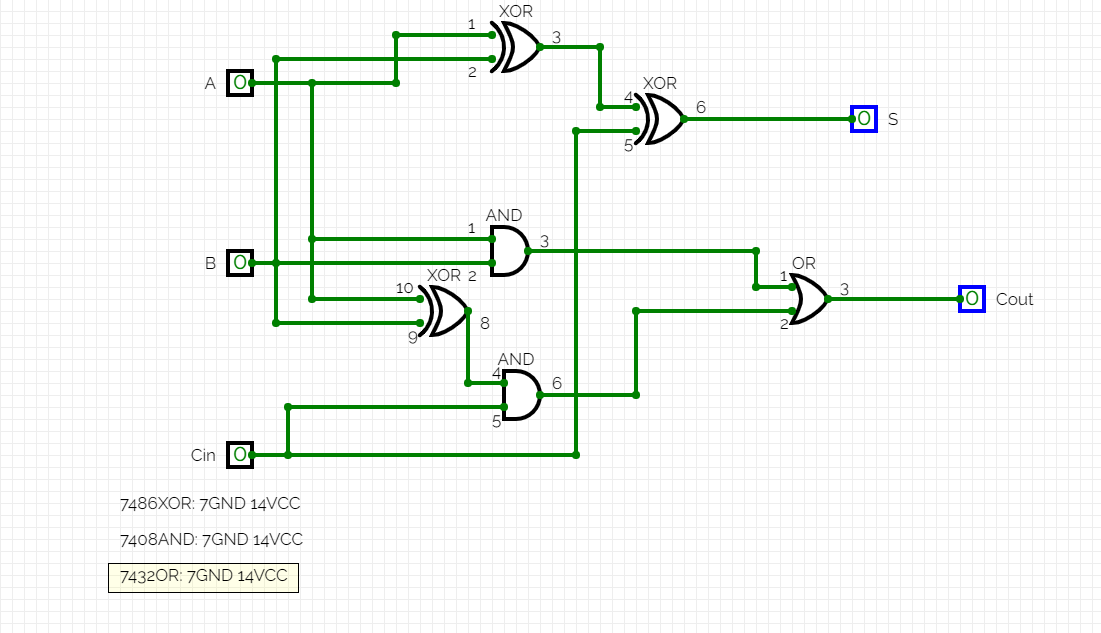
Altay İlker Yiğitel

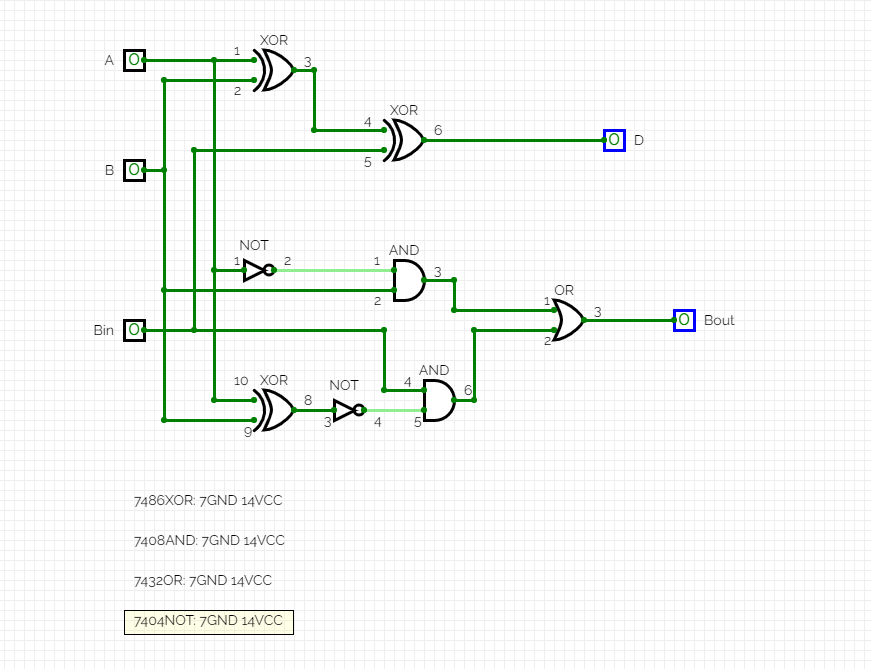
22203024 10.03.2024

CS223 Sec:01 Lab02

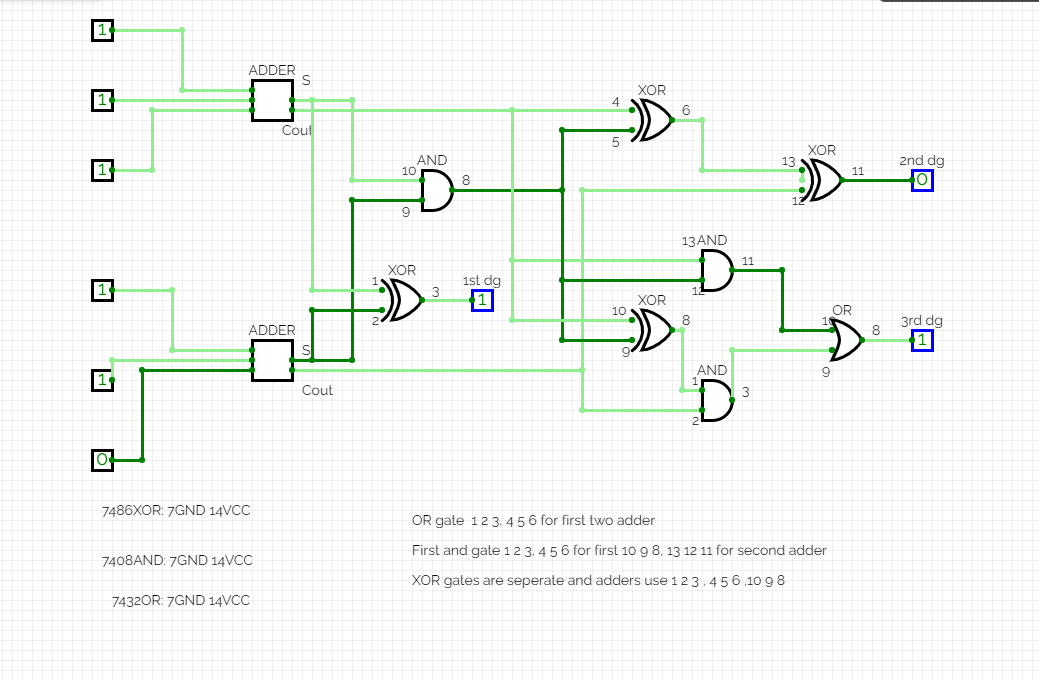
Adder:



Substractor:



2 full adders combined:



Behavioral SystemVerilog module for the full adder:

`timescale 1ns / 1ps module   
full\_adder( A, B, Cin, S, Cout);  
  
 input wire A, B, Cin;  
 output reg S, Cout;  
  
 always @(A or B or Cin)  
 begin   
 S = A ^ B ^ Cin;   
 Cout = A&B | (A^B) & Cin;   
 end  
endmodule

Structural SystemVerilog module for the full adder and a testbench for it:

module full\_adder\_s (  
 input a,b,cin;  
 output sum,carry;  
);  
  
wire w1,w2,w3,w4;   
  
xor(w1,a,b);  
xor(sum,w1,cin);   
  
and(w2,a,b);  
and(w3,b,cin);  
and(w4,cin,a);  
  
or(carry,w2,w3,w4);   
  
endmodule

Testbench:

module full\_adder\_tb;  
reg a,b,cin;  
wire sum,carry;  
  
full\_adder\_s uut(a,b,cin,sum,carry);  
  
initial begin  
a = 0; b = 0; cin = 0;  
#100  
a = 0; b = 0; cin = 1;  
#100  
a = 0; b = 1; cin = 0;  
#100  
a = 0; b = 1; cin = 1;  
#100  
a = 1; b = 0; cin = 0;  
#100  
a = 1; b = 0; cin = 1;  
#100  
a = 1; b = 1; cin = 0;  
#100  
a = 1; b = 1; cin = 1;  
#100  
$finish();  
end  
   
endmodule

Behavioral SystemVerilog module for the full substractor:

`timescale 1ns / 1ps module   
full\_substractor( A, B, C, Borrow, Diff);  
  
 input wire A, B, C;  
 output reg Borrow, Diff;  
  
 always @(A or B or C)  
 assign {Borrow,Diff} = (~A) + B + C;

endmodule

Structural SystemVerilog module for the full subtractor and a testbench for it:

module full\_substractor\_s (  
 input a,b,c;  
 output borrow,diff;  
);  
wire w1,w4,w5,w6;

xor (diff,a,b,c);

not n1(w1,a);

and a1(w4,w1,b);

and a2(w5,w1,c);

and a3(w6,b,c);

or o1(borrow,w4,w5,w6);  
  
endmodule

Testbench:

initial begin

A = 0;

B = 0;

C = 0;  
#100;

#100; A = 0;B = 0;C = 1;

#100; A = 0;B = 1;C = 0;

#100; A = 0;B = 1;C = 1;

#100; A = 1;B = 0;C = 0;

#100;A = 1;B = 0;C = 1;

#100; A = 1;B = 1;C = 0;

#100; A = 1;B = 1;C = 1;

end

Behavioral SystemVerilog module for the 2 full adders:

`timescale 1ns / 1ps module   
full\_adder( A, B, C, D, E, F S, Cout, S2, Cout2, first, second, third);  
  
 input wire A, B, C, D, E, F;  
 output reg first, second, third;  
  
 always @(A or B or C or D or E or F)  
 begin   
 S = A ^ B ^ C;   
 Cout = A&B | (A^B) & C;

S2 = D ^ E ^ F;   
 Cout2 = D&E | (D^E) & F;

first: S ^ S2;

second: (S&S2) ^ Cout ^ Cout2;

third: (S&S2) & Cout | ((S&S2) ^ Cout) & Cout2;

end  
endmodule

Structural SystemVerilog module for the 2 full adders and a testbench for it:

module full\_adders\_s (  
 input a,b,c,d,e,f;  
 output first,second,third;  
);  
  
wire w1,w2,w3,w4,w5,w6,w7,w8,w9,w10,w11,w12,w13,w14,w15;   
  
xor(w1,a,b);  
xor(w9,w1,cin);   
  
and(w2,a,b);  
and(w3,b,cin);  
and(w4,cin,a);  
  
or(w10,w2,w3,w4);

xor(w5,d,e);  
xor(w11,w5,f);   
  
and(w6,d,e);  
and(w7,e,f);  
and(w8,f,d);  
  
or(w12,w6,w7,w8);;

xor(first,w9,w11);

xor(second,first,w10,w12);

and(w13,first,w10);  
and(w14,first,w12);  
and(w15,w10,w12);  
  
or(third,ww13,ww14,w15);

endmodule

Testbench:

module full\_adders\_tb;  
reg a,b,c,d,e,f;  
wire first,second,third;  
  
full\_adder\_s uut(a,b,c,d,e,f,first,second,third);

a = 0; b = 0; c = 0; d = 0; e = 0; f = 0;  
#100  
  
initial begin  
  
  
$finish();  
end  
   
endmodule