10/19/2016

Project 2

Design an 8-bit Fast Adder

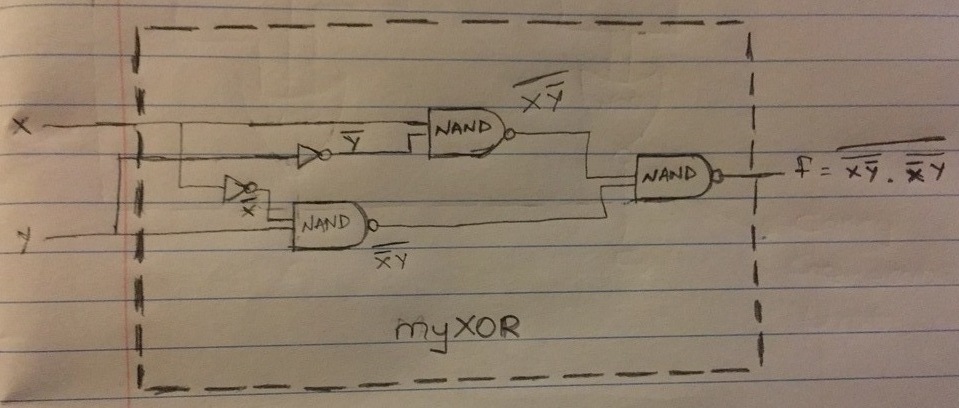
Bibodi, Jay Nikhil

Sec 2: 5:30-6:45

Fall 2016

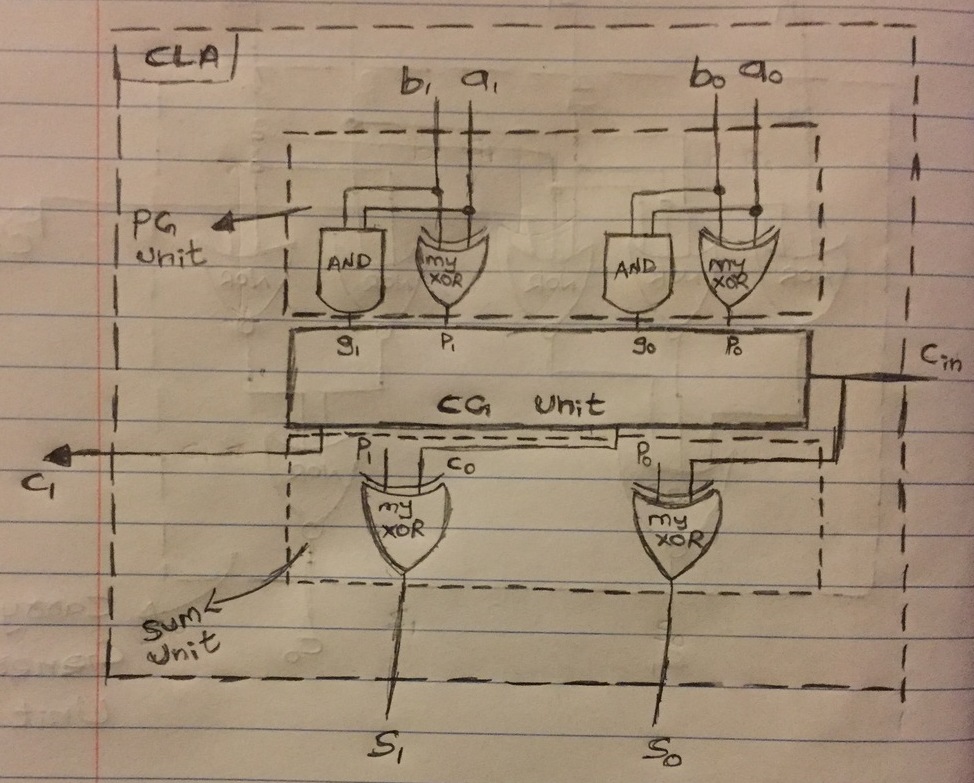
**Block Diagram for myXOR**

**Equation:**

****

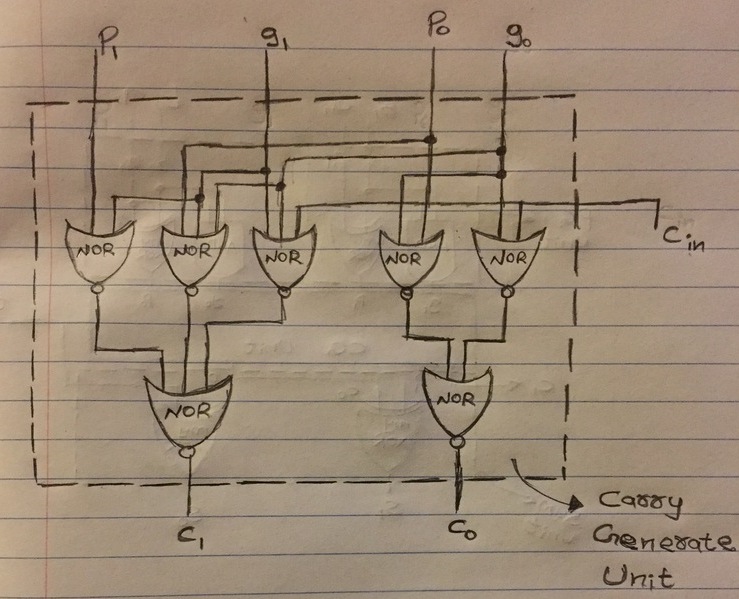
**Block diagram of 2 bit CLA Adder with pgu, cgu and su units**

**Equations:**

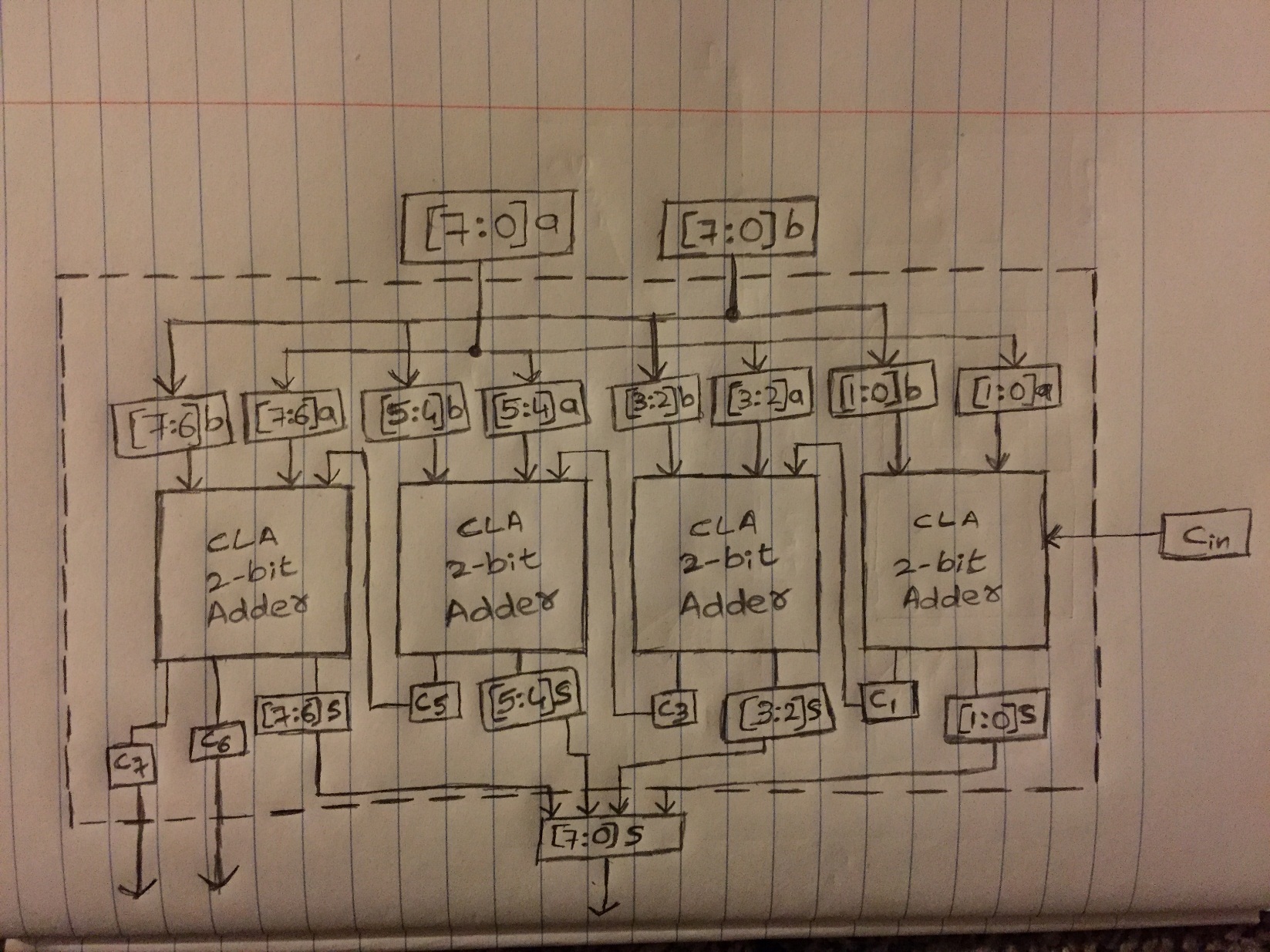
****

**Block diagram for carry generate unit (CGU)**

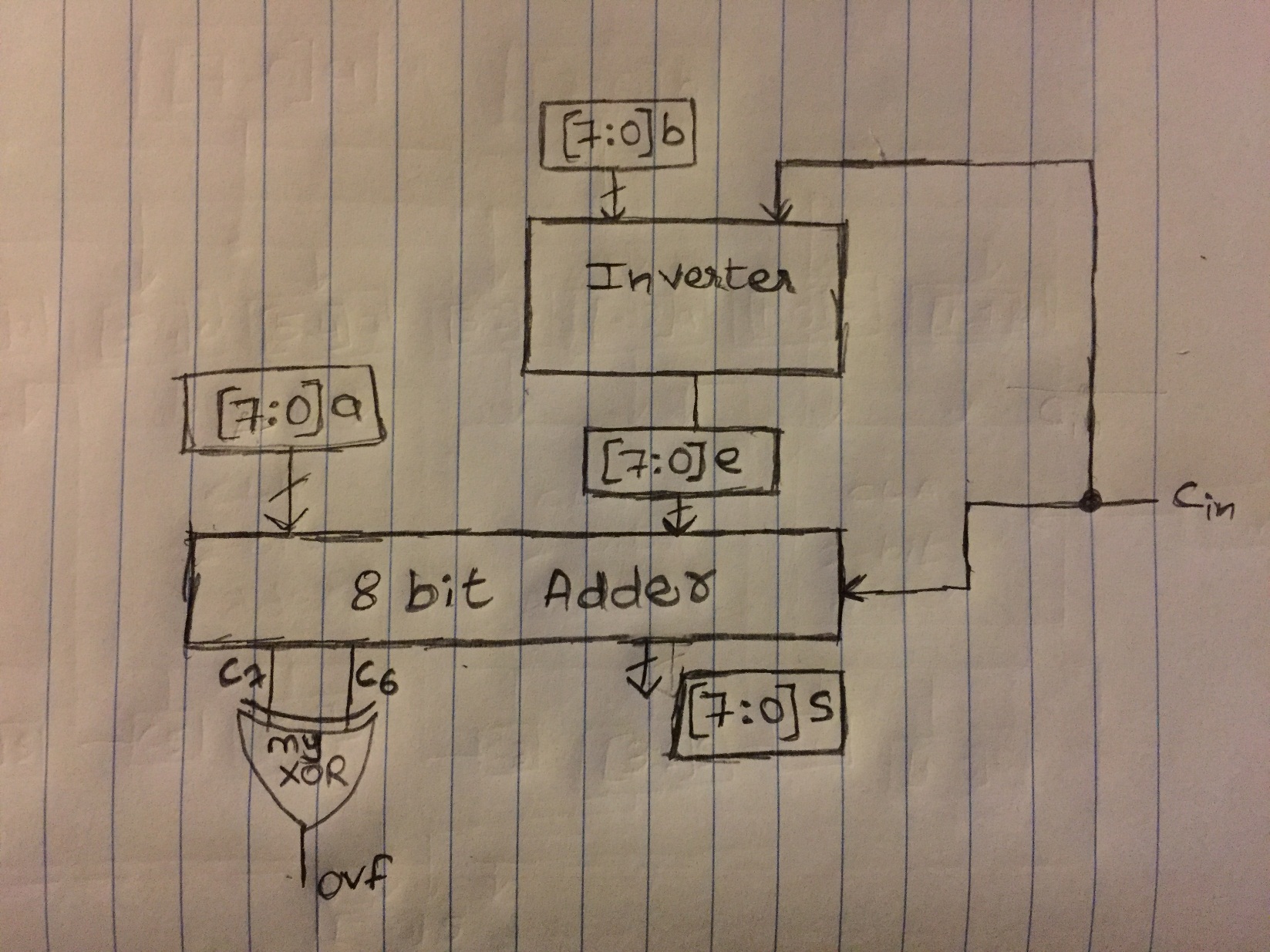
**Equations:**



**Block Diagram for 8bitclaHybridAdder**

****

**Block diagram of 8 bit adder/subtractor**

****

**HDL Designs**

**myXOR.v**

module myXOR(x,y,f); input x,y; output f;

wire xor1, xor2, xor3, xor4;

not xxor1(xor1,x); not yxor2(xor2,y);

nand yxor1(xor3,xor1,y); nand xxor2(xor4,xor2,x); nand xor34(f,xor3,xor4);

endmodule

**pg\_su.v**

`include "cgu.v" module pg\_su(a,b,ci,s,c);

input [1:0] a,b; input ci; output [1:0] s,c;

wire [1:0] pwire, gwire;

myXOR myXOR0(a[0],b[0],pwire[0]); myXOR myXOR1(a[1],b[1],pwire[1]);

and gwire0(gwire[0],a[0],b[0]); and gwire1(gwire[1],a[1],b[1]);

cgu cgu1(pwire,gwire,ci,c);

myXOR myXOR2(pwire[0],ci,s[0]); myXOR myXOR3(pwire[1],c[0],s[1]);

endmodule

**cgu.v**

module cgu(p,g,ci,c);

input [1:0] p,g; input ci; output [1:0] c;

wire nor1,nor2,nor3,nor4,nor5;

nor g0ci(nor4,g[0],ci); nor g0p0(nor5,g[0],p[0]); nor nor45(c[0],nor4,nor5);

nor g1g0ci(nor1,g[1],g[0],ci); nor g1g0p0(nor2,g[1],g[0],p[0]); nor g1p1(nor3,g[1],p[1]);

nor nor123(c[1],nor1,nor2,nor3);

endmodule

**cla2bitAdder.v**

`include "pg\_su.v" module cla2bitAdder (

input [1:0] a,b, input ci, output [1:0] s,c

);

pg\_su pg\_su1(a,b,ci,s,c);

endmodule

**cla8bitAdder.v**

`include "cla2bitAdder.v" module cla8bitAdder (

input [7:0] a,b, input cin, output [7:0] s, c );

cla2bitAdder unit1(a[1:0],b[1:0],cin,s[1:0],c[1:0]); cla2bitAdder unit2(a[3:2],b[3:2],c[1],s[3:2],c[3:2]); cla2bitAdder unit3(a[5:4],b[5:4],c[3],s[5:4],c[5:4]); cla2bitAdder unit4(a[7:6],b[7:6],c[5],s[7:6],c[7:6]);

endmodule

**invertor.v**

module invertor (

input cin, input [7:0] b, output reg [7:0] bout

);

always@(\*) begin if (cin==1'b1) bout = ~b; else bout = b;

end

endmodule

**adderSubtractor.v**

`include "cla8bitAdder.v" `include "invertor.v" `include "myXOR.v"

module adderSubtractor (

input [7:0] a,b, input cin, output [7:0] s,c, output ovf );

wire [7:0] binvertor;

invertor invt1(cin,b[7:0],binvertor); cla8bitAdder add1(a,binvertor,cin,s,c); myXOR xyz(c[7],c[6],ovf);

endmodule

**testBench.v**

`include "adderSubtractor.v" module tester();

reg [7:0] a,b; reg cin; wire [7:0] s,c; wire ovf;

adderSubtractor adsub(a,b,cin,s,c,ovf);

initial begin a=8'hFF;b=8'h01;cin=1'b0; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

a=8'h7F;b=8'h01;cin=1'b0; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

a=8'h01;b=8'hFF;cin=1'b0; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

a=8'h55;b=8'hAA;cin=1'b0; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

a=8'h80;b=8'h01;cin=1'b1; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

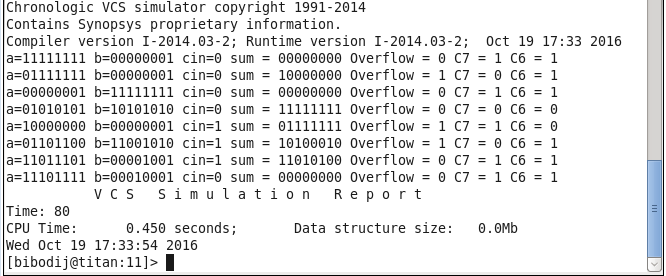
a=8'h6C;b=8'hCA;cin=1'b1; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

a=8'hDD;b=8'h09;cin=1'b1; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

a=8'hEF;b=8'h11;cin=1'b0; #10 $display("a=%8b b=%8b cin=%b sum = %8b Overflow = %b C7 = %b C6 = %b ",a,b,cin,s,ovf,c[7],c[6]);

end endmodule

**Simulation for adder/subtractor**

****