BILKENT UNIVERSITY

*Department of Computer Engineering*

*Bilkent 06800 Ankara Turkey*



**CS223 – Digital Design**

Melih Obut

21502659

Section: 01

Lab: 02

20.10.2019

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

/ one bit full adder

module full\_add\_1bit(a , b, cin, sum, cout);

// by dataflow modelling

   input a,b,cin;

   output sum, cout;

//specify the function of a full adder

assign sum = (a ^ b) ^ cin;

assign cout = (a & b) | ((a ^ b) & cin);

endmodule

// one bit full subtractor

module full\_subtractor ( a ,b ,bin ,diff ,cout );  
  
  
assign diff = (a ^ b) ^ cin;  
assign cout = ((-a) & b) | (-(a ^b) &bin);  
  
endmodule

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

module full\_adder(x,y,cin,s,cout);

input x,y,cin;  
output s,cout;

wire s1,c1,c2,c3;

xor(s1,x,y);

xor(s,cin,s1);

and(c1,x,y);

and(c2,y,cin);

and(c3,x,cin);

or(cout,c1,c2,c3);

endmodule

module full\_subtractor ( a ,b ,cin ,diff ,borrow );  
  
output diff ;  
output borrow ;  
  
input a ;  
input b ;  
input cin ;  
  
assign diff = a ^ b ^ cin;  
assign borrow = ((-a) \* b) + (b & cin) + (cin a (-a));  
  
endmodule

module testbench    // testbench to test the full adder

// declaration of input

reg A,B,Cin;

// declaration of output

wire Sum, carry;

// instiation of fullbit adder

full\_add\_1bit f1( A,B,cin,sum,carry[0]);

initial begin

     A=0; B=0; Cin=0;

   end

initial begin

   #100;

   // excersie all input combination and check the output

    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;

#100

   $finish;

    end

endmodule

6- Structural Systemverilog module for the 2-bit full adder & subtractor, and a testbench for them. Use the 1-bit adder & subtractor module you wrote inside.

Module full\_add\_ 2(A, B , CI, S CO);

input A ,B, CI;

output S, CO;

wire S1, C1, C2;

//build full adder from 1 bit full-adders

full\_adder PARTSUM(A,B,S1,C1),

SUM(S1,C1,S,C2);

//add an OR gate for he carry

or CARRY (CO, C2, C1);

endmodule;

Module full\_subractor\_ 2(A, B , CI, S CO);

input A ,B, CI;

output diff, borrow;

wire S1, C1, C2;

//build full subractor from 1 bit full-subractor

full\_subractor borrow(A,B,S1,C1),

diff(S1,C1,S,C2);

endmodule;