Modified P2

(CMPS430 /AK / Due October 10, 2006)

Designing Datapath Elements with Verilog: Assignment 2 (P2)

1. Write a Verilog model for designing an 8 bit adder. The inputs to the adder are two 8-bit numbers and a 1-bit carry-in, while the outputs are 8-bit sum and a 1-bit carryout.

[**Hint: Code for 4-bit adder is provided on the next two pages. Put the given two Verilog modules in two separate files, named "`OneBitFullAdder`.vl" and "Top.vl", respectively.**

To compile: → iverilog –o mycompiledcode Top.vl OneBitFullAdder.vl
To run: → vvp mycompiledcode

]

2. We want to design our first simple ALU in Verilog. The inputs to the module are two 8-bit numbers, say A, and B; and also a 3-bit number, say *ControlCode*. Based on the ControlCode, the module performs operations, as follows:

Value of ControlCode (3 bit value)	Operation Performed
000	A+B (logical sum, not arithmetic sum)
001	A! + B!
010	A.B ('and' operation)
011	A exor B
Anyother value	No operation; display invalid message

The output, clearly, is an 8-bit number.

[Hint: The code can be written with if-then or also case statements].

```
// CMPS 430 (UG)
// My first 1 bit full-adder in Verilog !!!!!!!!!!!!!!
// The adder takes 3 1-bit numbers a, b and cin and produces Sum and
CarryOut

// PLEASE NAME THIS FILE as "OneBitFullAdder.vl"


module OneBitFullAdder(a, b, cin, sum, cout);
input   a, b, cin;
output  sum, cout;


// use the logic equations for 1-bit full adder to compute sum and
carryout

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


endmodule
```

```
// 4-bit Full Adder CMPS 430 (UG) Fall 2006 – P2 (Problem 1)
// The name of the file containing this code is Top.vl

module top();

reg [3:0] Ain;
reg [3:0] Bin;
reg  CarryIn;

wire [3:0] SumOut;
wire  c1, c2, c3, CarryOut;

OneBitFullAdder FBA0(Ain[0], Bin[0], CarryIn, SumOut[0], c1);
OneBitFullAdder FBA1(Ain[1], Bin[1], c1, SumOut[1], c2);
OneBitFullAdder FBA2(Ain[2], Bin[2], c2, SumOut[2], c3);
OneBitFullAdder FBA3(Ain[3], Bin[3], c3, SumOut[3], CarryOut);


initial
      begin
            Ain = 4'b1000;
            Bin = 4'b0001;
            CarryIn = 1'b0;

            #10;
            Ain = 4'b1000;
            Bin = 4'b0001;
            CarryIn = 1'b0;

            #10;
            Ain = 4'b1111;
            Bin = 4'b1111;
            CarryIn = 1'b1;

            #10;
            Ain = 4'b1001;
            Bin = 4'b1011;
            CarryIn = 1'b1;

 // you must do more assignments to test your adder better !!

            #10;
            $dumpflush;
      end


initial
      begin
      $monitor("Ain=%4b, Bin=%4b, Carryin=%b, SumOut=%4b, CarryOut=%b,
time=%t\n", Ain, Bin, CarryIn, SumOut, CarryOut, $time);
      $dumpfile("top.dump");
      $dumpvars(5, top);
end



endmodule
```