Lab 8 Homework. 4-bit by 3-bit multiplier.

testbench.sv

```
// AUTHOR : Gianna Rose
// Lab8 Hw 4-bit x 3-bit multiplier
// https://www.edaplayground.com/x/6r2J

module test;

  reg [2:0] A;
  reg [3:0] B;

  wire [6:0] C;

  // Instantiate the Unit Under Test (UUT)
  multiplier uut(C,B,A);

           initial
          begin
$dumpfile("dump.vcd");  $dumpvars(1, test);

                // display the inputs and outputs
            $monitor( "%d * %d  = %d", A, B, C);

       // Initialize Inputs
//              for(int i = 0; i < 9; i = i + 1) begin
//                for(int j = 0; j < 9; j = j + 1) begin
//                  {B} = j;
//                  #10;
//                end
//                {A} = i;
//                #10;
//              end

             // for loop not working gotta brute :///
                    A=0; B=0;
                #10 A=1; B=1;
                #10 A=1; B=2;
                #10 A=2; B=2;
                #10 A=3; B=2;
                #10 A=4; B=2;
                #10 A=5; B=2;
                #10 A=6; B=3;
                #10 A=7; B=4;
                #10 A=2; B=7;



                  #10 $finish;
```

```
    end
  endmodule
```

```
design.sv
// Code your design here
module fulladder (S, C, x, y, cin);
   input x, y, cin;
   output S, C;

   //internal signals
   wire S1, D1, D2;

   //Instantiate the half adders
    halfadder    HA1 (S1, D1, x, y);
    halfadder    HA2 (S, D2, S1, cin);
    or U3(C, D2, D1);

endmodule

module halfadder (S, C, x, y);
   input x, y;
   output S, C;

//Instantiate primitive gates
   xor U1(S, x, y);
   and U2(C, x, y);

endmodule

module four_bit_adder (S, C4, A, B, Cin);
   input [3:0] A,B;
   input Cin;
   output [3:0] S;
   output C4;


//Declare intermediate carries
 wire C1, C2, C3;

//Instantiate the fulladder
  fulladder FA0(S[0], C1, A[0], B[0], Cin);
  fulladder FA1(S[1], C2, A[1], B[1], C1);
  fulladder FA2(S[2], C3, A[2], B[2], C2);
  fulladder FA3(S[3], C4, A[3], B[3], C3);

endmodule
```

```
module multiplier(C,A,B); // BASED OFF DIAGRAM 4.16 FROM TEXTBOOK
  input[2:0] A;
  input[3:0] B;
  output[6:0] C;

  wire[3:0] S;
  wire[3:0] W0,W1,W2;
  wire Cout;

  // B[0] * A[0] = C[0]
  // Multiply B from pos 3 to pos 1 [3:1]
  assign W0 = B[3:1] & {A[0],A[0],A[0]};
  assign W1 = B & {A[1],A[1],A[1],A[1]};

  // S -> output
  // Cout -> carry output
  // W1 and W1 products being added
  // 0 since not subbing **dont use cin**
  four_bit_adder F0(S, Cout, W1, W0, 0);

  // get last product
  assign W2 = B & {A[2],A[2],A[2],A[2]};

  // assign proper position bits for final output C
  assign C[0] = A[0] & B[0];
  assign C[1] = S[0];

  // C[5:2] -> sum
  // C[6] -> highest bit of C
  // {Cout, S[3:1]} ->  S[3:1] add by the product of highest bit of A mult with all of B bits due to
left shift
  // Cout is the final carry from the first fourbit adder addition
  four_bit_adder F1(C[5:2], C[6], W2, {Cout, S[3:1]}, 0);

endmodule
```


```
0 *  0  =   0
1 *  1  =   1
1 *  2  =   2
2 *  2  =   4
3 *  2  =   6
4 *  2  =   8
5 *  2  =  10
6 *  3  =  18
7 *  4  =  28
2 *  7  =  14
Finding VCD file...
./dump.vcd
```