**计算机组成与设计实验报告**

**实验3·加减法器的实现**

学 院： 网络空间安全学院（研究院）

组 员：孟庆丹、牛中原

***实验三.加减法器的实现***

*Adder-Subtractor*

成员分工：

孟庆丹：编写实验代码，完成实验报告

牛中原：测试仿真结果，完成实验报告

全加器是用门电路实现两个二进制数相加并求出和的组合线路，称为一位全加器。一位全加器可以处理低位进位，并输出本位加法进位。多个一位全加器进行级联可以得到多位全加器。

本次实验的目的是使用 verilog 实现加减法器，熟练使用 modelsim 工具进行仿真和调试，学习掌握实例化模块的方式

1. 一位全加器的实现

根据相加的两位数及进位算出向下一位的进位和输出。

代码实现：

`timescale 1ns / 1ps

module adder(

    input ia,

    input ib,

    input cin,

    output cout,

    output sum

);

    assign cout = (ia&ib) | (ia&cin) | (ib&cin);

    assign sum = ia^ib^cin;

endmodule

Testbench:

`timescale 1ns / 1ps

module adder\_test(

   );

   reg a,b,cin;

   wire cout,s;

   adder add(a, b, cin, cout, s);

   initial begin

        a = 1;

        b = 1;

        cin = 1;

        #100

        a = 1;

        b = 0;

        cin = 1;

        #100

        a = 1;

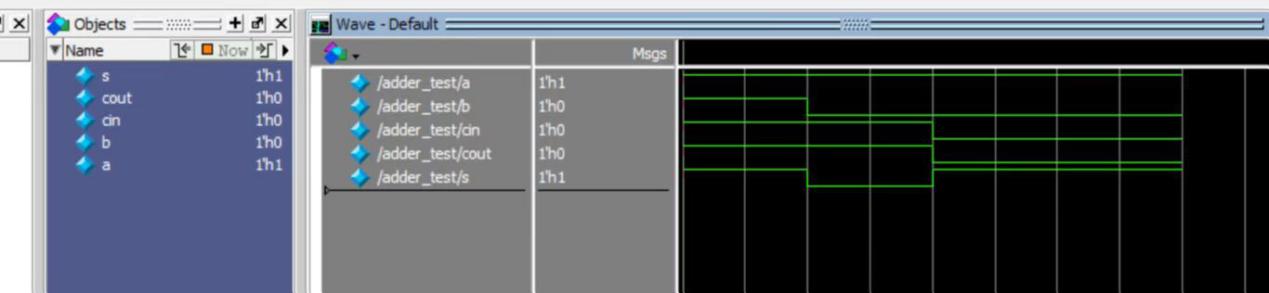
        b = 0;

        cin = 0;

   end

Endmodule

仿真图象：

****

1. 四位串行加法器实现

例化四个一位加法器实现一个四位串行加法器

`timescale 1ns / 1ps

module adder\_4(

    input [3:0] a,

    input [3:0] b,

    input cin,

    output [3:0] s,

    output cf

);

    wire cout0,cout1,cout2;

    adder add1(a[0], b[0], cin, cout0, s[0]);

    adder add2(a[1], b[1], cout0, cout1, s[1]);

    adder add3(a[2], b[2], cout1, cout2, s[2]);

    adder add4(a[3], b[3], cout2, cf, s[3]);

endmodule

Testbench：

`timescale 1ns / 1ps

module adder\_4\_test(

   );

   reg [3:0] a;

   reg [3:0] b;

   reg cin;

   wire cf;

   wire [3:0] s;

   adder\_4 adder4(a, b, cin, s, cf);

   initial begin

        a = 4'b0001;

        b = 4'b0011;

        cin = 1;

        #100

        a = 4'b1000;

        b = 4'b1010;

        cin = 0;

        #100

        a = 4'b0011;

        b = 4'b0110;

        cin = 0;

        #100

        a = 4'b0111;

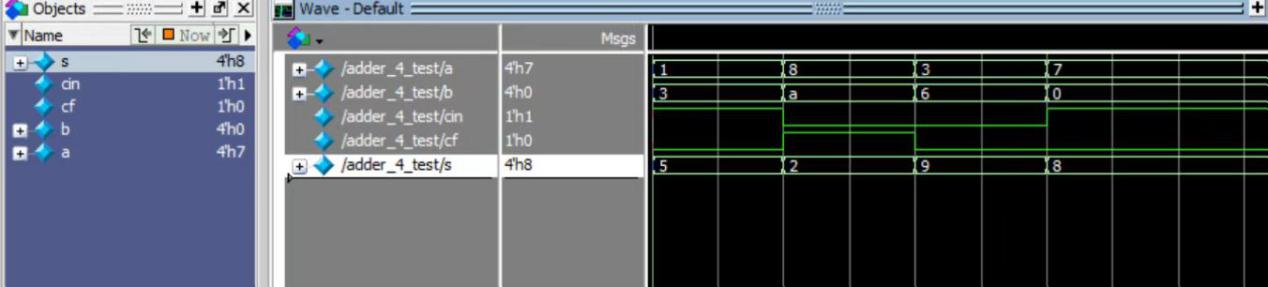
        b = 4'b0000;

        cin = 1;

   end

endmodule

仿真图象：

****

1. 四位有符号加法器实现

补码可以直接参与有符号加减运算，并且通过双符号位判断溢出，当高两位不同时，溢出判断为真。

代码实现：

`timescale 1ns / 1ps

module adder\_4\_s(

    input [3:0] a,

    input [3:0] b,

    input cin,

    output [3:0] s,

    output overflow

);

    wire cout0,cout1,cout2,cout3;

    adder add1(a[0], b[0], cin, cout0, s[0]);

    adder add2(a[1], b[1], cout0, cout1, s[1]);

    adder add3(a[2], b[2], cout1, cout2, s[2]);

    adder add4(a[3], b[3], cout2, cout3, s[3]);

    assign overflow = cout2^cout3;

endmodule

Testbench：

`timescale 1ns / 1ps

module adder\_4\_s\_test(

   );

   reg [3:0] a;

   reg [3:0] b;

   reg cin;

   wire overflow;

   wire [3:0] s;

   adder\_4\_s adder4(a, b, cin, s, overflow);

   initial begin

        a = 4'b0001;

        b = 4'b0011;

        cin = 1;

        #100

        a = 4'b1000;

        b = 4'b1010;

        cin = 0;

        #100

        a = 4'b0011;

        b = 4'b0110;

        cin = 0;

        #100

        a = 4'b0111;

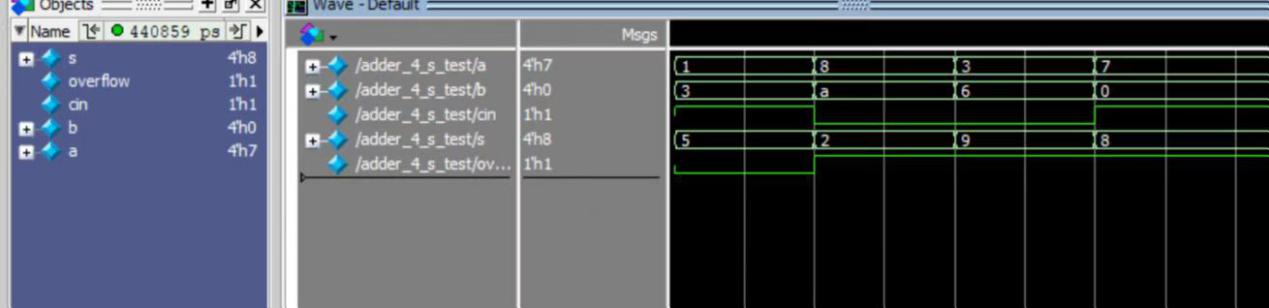
        b = 4'b0000;

        cin = 1;

   end

endmodule

仿真图像：

****4.四位有符号加减法器实现

相对于有符号加法运算器，只需要将减法中的被减数改为原数的相反数。

代码实现：

`timescale 1ns / 1ps

module add\_sub(

    input [3:0] a,

    input [3:0] b,

    input cin,

    input operator,

    output [3:0] result,

    output overflow

);

    wire [3:0] b1;

    wire [3:0] b2;

    assign b1[0] = b[0]^operator;

    assign b1[1] = b[1]^operator;

    assign b1[2] = b[2]^operator;

    assign b1[3] = b[3]^operator;

    assign b2 = b1 + operator;

    adder\_4\_s oper(a,b2,cin,result,overflow);

endmodule

Testbench：

`timescale 1ns / 1ps

module add\_sub\_test(

   );

   reg [3:0] a;

   reg [3:0] b;

   reg cin;

   reg op;

   wire overflow;

   wire [3:0] s;

   add\_sub add\_sub1(a, b, cin, op, s, overflow);

   initial begin

        a = 4'b0001;

        b = 4'b0011;

        op = 0;

        cin = 1;

        #100

        a = 4'b1000;

        b = 4'b1010;

        op = 0;

        cin = 0;

        #100

        a = 4'b0111;

        b = 4'b1000;

        op = 0;

        cin = 1;

        #100

        a = 4'b0001;

        b = 4'b0011;

        op = 1;

        cin = 0;

        #100

        a = 4'b1000;

        b = 4'b1010;

        op = 1;

        cin = 0;

        #100

        a = 4'b0111;

        b = 4'b1000;

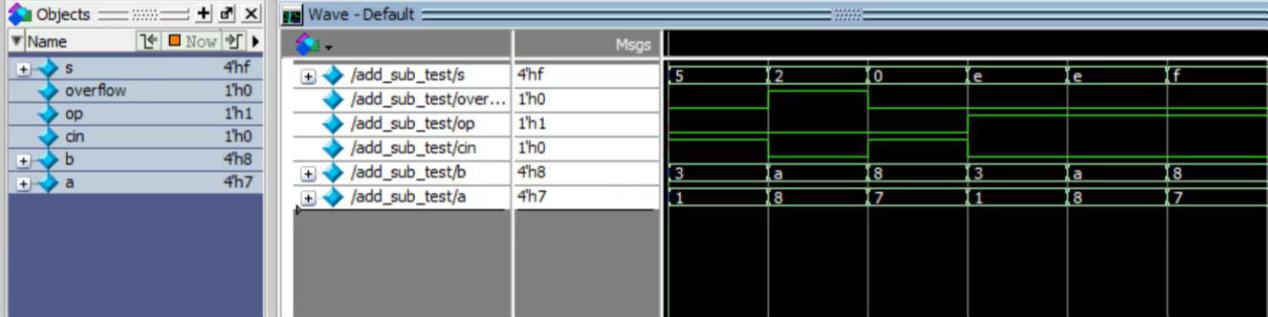
        op = 1;

        cin = 0;

   end

endmodule

仿真结果：

****

5.串行加减法器的开发板实现

通过仿真验证代码功能正确后，本作业要求通过八个开关 SW作为两个四位二进制输入，再将一个开关作 为 cin 输入，一个开关作为 operator 输入，将运算后的四位二进制对应的**无符号十六 进制**结果在七段数码管上显示出来，同时将 overflow 标志绑定

板子上的任意一个 led 灯（在数码管和开关之间），指示是否发生了溢出。

约束文件：

set\_pin\_assignment { a[0] } { LOCATION = T6; }

set\_pin\_assignment { a[1] } { LOCATION = T5; }

set\_pin\_assignment { a[2] } { LOCATION = R5; }

set\_pin\_assignment { a[3] } { LOCATION = T4; }

set\_pin\_assignment { b[0] } { LOCATION = P8; }

set\_pin\_assignment { b[1] } { LOCATION = N6; }

set\_pin\_assignment { b[2] } { LOCATION = P6; }

set\_pin\_assignment { b[3] } { LOCATION = M6; }

set\_pin\_assignment { cin } { LOCATION = N8; }

set\_pin\_assignment { op } { LOCATION = R9; }

set\_pin\_assignment { overflow } { LOCATION = M3; }

set\_pin\_assignment { result[0] } { LOCATION = A2; }

set\_pin\_assignment { result[1] } { LOCATION =B2; }

set\_pin\_assignment { result[2] } { LOCATION = D3; }

set\_pin\_assignment { result[3] } { LOCATION =F5; }

实验总结：学习了modelsim仿真硬件设计中的使用并尝试通过verilog语言完成了四位有符号加减法器的实现，为之后的实验打下了坚实的基础。