计算机系统结构实验报告 实验二

惠宇龙 518030910059

1 实验概述

1.1 实验描述

先实现一个1-bit 的加法器，而后再以此为基础，实现一个4-bit 的加法器，并进行验证实验，检测结果。

1.2 实验目的

- 掌握Xilinx逻辑设计工具 Vivado的基本操作
- 掌握 VerilogHDL进行简单的逻辑设计
- 使用功能仿真
- 约束文件的使用和直接写法
- 添加时序约束
- 生成 Bitstream 文件

2 基础加法器

2.1 基础加法器模块的设计

设置三个输入端口，其中两个为加数，一个表示接受进位的状态。

设置两个输出端口，一个为求和结果，一个表示向前进位的状态。

再利用几种简单的逻辑计算，即可以从输入信号得到输出结果

2.2 基础加法器模块的实现

该部分代码较为简单，已经由参考书给出。分析其逻辑可知：首先用`xor`语句计算加和之后该bit显示的结果。随后，用`and`和`or`计算向前进位的结果。

```
module adder_1bit(
    input a,
    input b,
    input ci,
```

```
    output s,
    output co
    );

    wire s1,c1,c2,c3;

    xor (s1,a,b),
        (s,s1,ci);

    and (c1,a,b),
        (c2,b,ci),
        (c3,a,ci);
    or  (co,c1,c2,c3);
endmodule
```

3 四位加法器

3.1 四位加法器模块的设计原理

输入信号为两个加数 a[3:0], b[3:0] 和需要接受的进位状态 c_i

输出信号为求和结果 s[3:0] 以及需要向前进位的状态 co.

利用已经实现的1-bit 加法器，从低到高，依次处理4个bit 的每一位bit，即可得到最终结果。

3.2 四位加法器模块的实现

该部分代码较为简单，已经由参考书给出。在代码中，先后实例化了四个基本加法器，这与上述的设计原理一致。

```
module adder_4bits(
    input [3:0]a,
    input [3:0]b,
    input ci,
    output [3:0] s,
    output co
    );

    wire [2:0] ct;
    adder_1bit
       a1(.a(a[0]), .b(b[0]), .ci(ci), .s(s[0]), .co(ct[0])),
       a2(.a(a[1]), .b(b[1]), .ci(ct[0]), .s(s[1]), .co(ct[1])),
       a3(.a(a[2]), .b(b[2]), .ci(ct[1]), .s(s[2]), .co(ct[2])),
       a4(.a(a[3]), .b(b[3]), .ci(ct[2]), .s(s[3]), .co(co));

endmodule
```

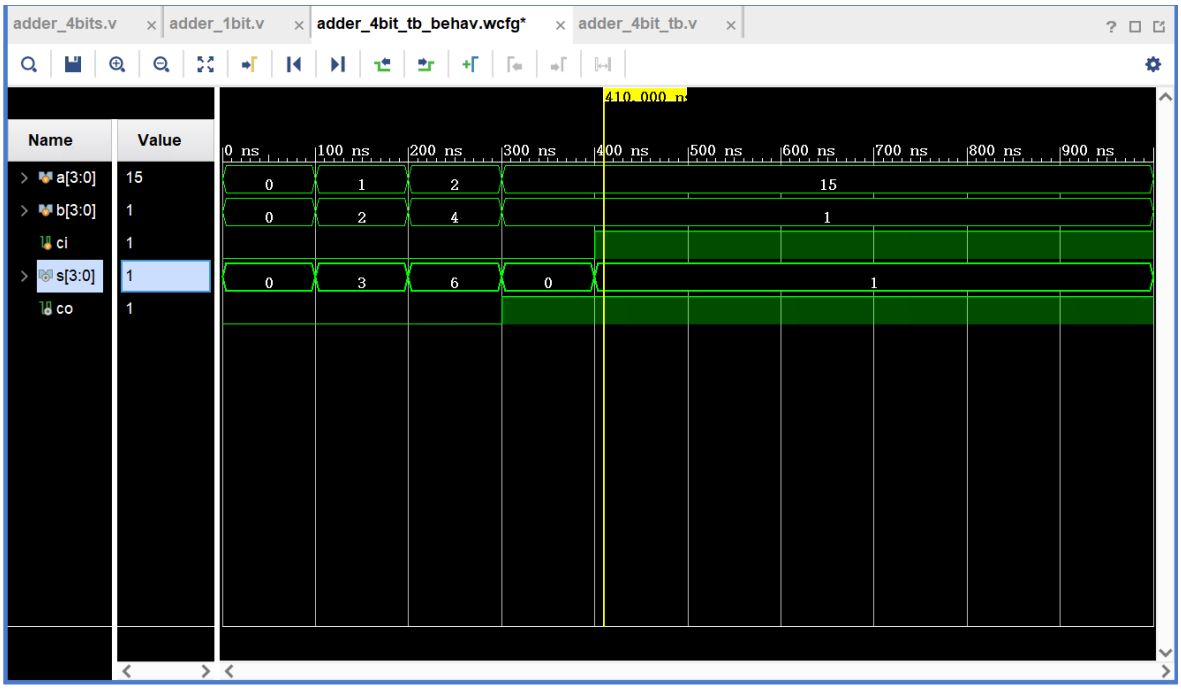
4 最终验证

激励文件已经由参考书给出。在实例化四位加法器之后，多次给两个加数赋值即可。

```
module adder_4bit_tb(
    );
    reg[3:0] a;
    reg[3:0] b;
    reg ci;
```

```
    wire[3:0]s;
    wire co;
    adder_4bits u0(
    .a(a),
    .b(b),
    .ci(ci),
    .s(s),
    .co(co)
    );

    initial begin
    a=0;b=0;ci=0;

    #100;
    a=4'b0001;
    b=4'b0010;
    #100;
    a=4'b0010;
    b=4'b0100;
    #100;
    a=4'b1111;
    b=4'b0001;
    #100;
    ci=1'b1;

    end

endmodule
```

运行测试，我们可以得到如下的波形：



可以看到在0-300ns期间，s都是a和b的运算结果。在300-400ns,因为15+1=16，此时进位状态为1，而s的结果为0。在400ns之后，需要接受更低位的进位状态, 15+1+1=17，因此，此时的进位状态为1，s的结果为1.

综上，该四位加法器可以计算出正确的结果，实现成功。

5 总结反思

本实验依然是按照参考书按部就班地进行操作，并没有太多的难点，但也有不少收获：

本实验对加法器的设计原理，很好地应用了用底层硬件实现上层运算的方法：通过逻辑电路和逻辑关系对数据进行更新，并且通过进位实现较大数字的运算。虽然本实验并不复杂，但其中蕴含的底层表示方法是普遍的，这对我有很大的启发。

本实验另一个关键的思想是将大问题模块化，即先完成一个基础加法器的小模块，在解决大问题时多次利用这一小模块。这样的思想，可以使解决问题的思路和条理更为清晰，同时也提高了工作的效率。无论是在硬件语言还是高级语言中，这都是非常重要的一种思维方式。