

# 计算机系统结构实验Lab02: 4-bit Adder

常烁晨 521021910369

2023.3.23

## 摘要

lab02 的实验目标是在 Vivado 实验环境中用 Verilog 语言实现一个四位的加法器，这是接下来几个 lab 的基础，借助 lab02，掌握CPU最基本的计算功能的实现方式。同时，有了 lab01 的基础，我们可以比较轻松完成 lab02 的工程创建、项目流程、上板验证等环节。通过 lab02 的训练，我进一步掌握了硬件模块化编程的方式，了解了CPU的基本工作方式和模块的输入输出、运算设计。

由于工程创建、仿真创建、上板验证等重复内容在 lab01 的实验报告中已经详细介绍，因此此后的实验报告着重介绍实验本身的设计思路、代码编写以及仿真结果展示，重复性的内容不再赘述。

# 目录

摘要	1
1、实验目的	3
2、源代码实现	3
2.1 adder_1bit	3
2.2 adders_4bits	4
3、仿真实现	5
4、工程实现	7
4.1 Top文件	7
4.2 display IP	9
4.3 约束文件	9
4.4 下载验证	9
5、致谢	10

## 1、实验目的

(1) 在 lab01 的基础上，进一步掌握 Vivado 的基本操作、VerilogHDL 的简单逻辑设计、激励文件与仿真实现、约束文件的应用。具体是根据实验报告指导书，完成四位加法器的实验工程代码的编写、实验仿真。

(2) 确保实验仿真结果正确无误后，进行工程实现。应用课程组提供的文件，连接 FPGA 开发板，在开发板上实现四位加法器功能。

## 2、源代码实现

### 2.1 adder\_1bit

首先完成一位加法器的设计和代码编写。四位加法器就是在一位加法器的基础上完成的。

```
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module adder_4bits(
24     input [3:0] a,
25     input [3:0] b,
26     input ci,
27     output [3:0] s,
28     output co
29 );
30     wire [2:0] ct;
31     adder_1bit a1(.a(a[0]),.b(b[0]),.ci(ci),.s(s[0]),.co(ct[0])),
32     a2(.a(a[1]),.b(b[1]),.ci(ct[0]),.s(s[1]),.co(ct[1])),
33     a3(.a(a[2]),.b(b[2]),.ci(ct[1]),.s(s[2]),.co(ct[2])),
34     a4(.a(a[3]),.b(b[3]),.ci(ct[2]),.s(s[3]),.co(co));
35 endmodule
36
```

这段代码实现了一位加法器的操作，根据加法的流水实现，对每一位进行运算，最后判断进位的情况。

由于计算机中运算方式是二进制，即相加的两位是 0 或者 1，因此对每一位做and 运算或者 xor 运算，就可以判断加法的结果和进位情况。

在上面的模块中，加法器看作一个 module，这个 module 接入了三个输入信号以及两个输出信号，我们可以把这个模块想象成一个具有运算功能的黑盒子，接入三根数据线输入三个比特，经过运算后得到两个输出线比特值，这就是 Vivado 环境中创建模块的方式。

## 2.2 adders\_4bits

四位加法器是在一位加法器的基础上实现的，在 Vivado 中，新建的模块可以调用其他已经创建好的模块，这相当于把几个“黑箱子”组合起来，获得一个功能更强大的运算模块。

```
adder_4bits_tb.v x adder_4bits.v x adder_1bit.v x Untitled 1 x
E:/lab02/lab02.srscs/sources_1/new/adder_1bit.v
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//
//
module adder_1bit(
    input a,
    input b,
    input ci,
    input s,
    output co
);
    wire s1, c1, c2, c3;
    and (c1, a, b),
        (c2, b, ci),
        (c3, a, ci);
    xor (s1, a, b),
        (s, s1, ci);
    or (co, c1, c2, c3);
endmodule
```

在四位加法器中，输入信号是两个4bit的数字信号，得到的输出结果也是4bit。在这样的模块中调用四个一位加法器，分别完成每一位的加法以及进位情况，最后得到对应的四位加法结果并通过输出信号输出。

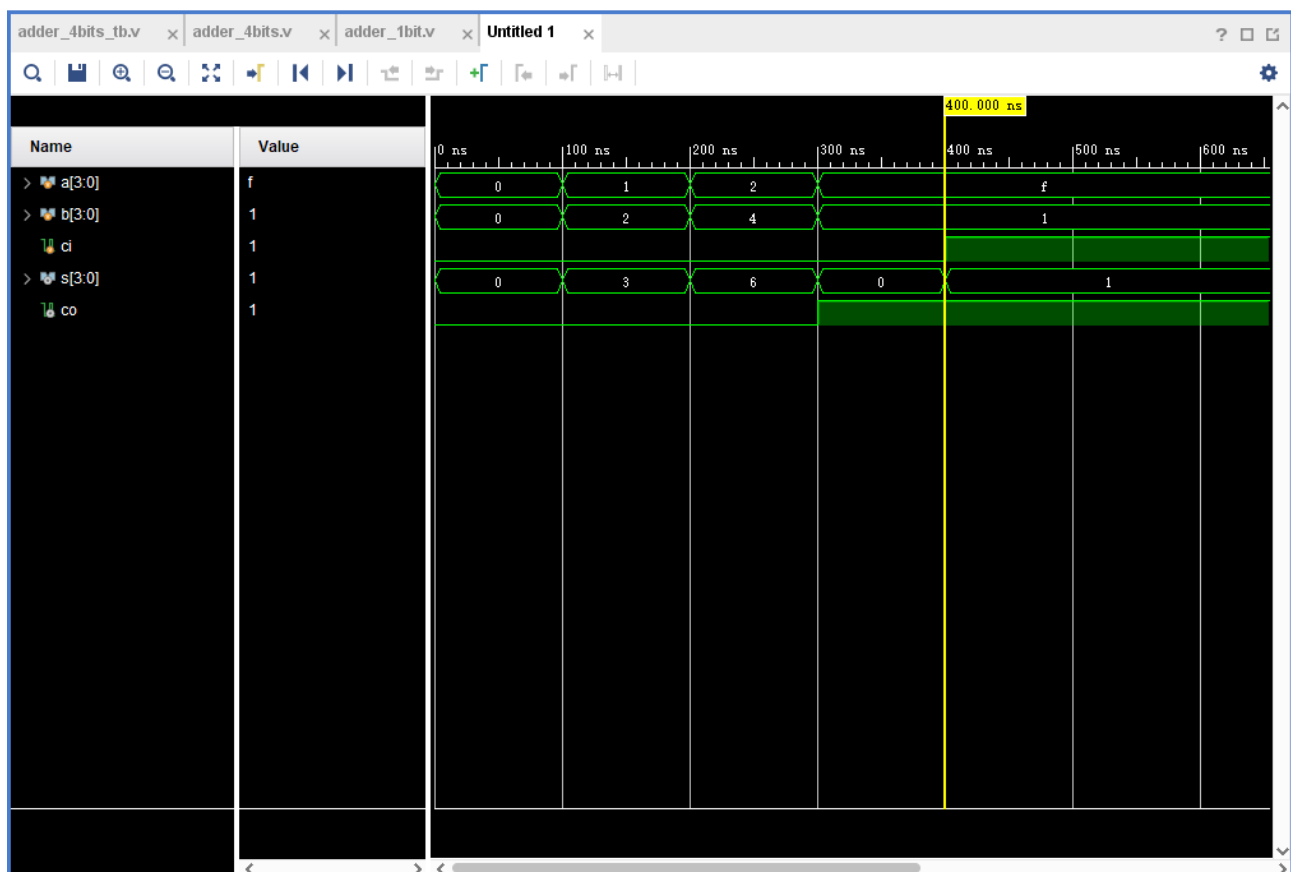
### 3、仿真实现

创建激励仿真文件 adder\_4bits\_tb，激励文件的代码如下所示。

```
Top.v x adder_4bits.v x display.v x display.edif x adder_1bit.v x lab02_xdc.xdc x adder_4bits_tb.v x
C:/Users/cetc01/Desktop/lab02/lab02.srscs/sim_1/new/adder_4bits_tb.v
🔍 📁 ⬅ ➡ ✂ 📄 📁 ❌ // 📖 💡
22
23 module adder_4bits_tb(
24
25 );
26     reg [3:0] a;
27     reg [3:0] b;
28     reg ci;
29
30     wire [3:0] s;
31     wire co;
32
33     adder_4bits u0 (
34         .a(a),
35         .b(b),
36         .ci(ci),
37         .s(s),
38         .co(co)
39     );
40     initial begin
41         a = 0;
42         b = 0;
43         ci = 0;
44         #100;
45         a = 4'b0001;
46         b = 4'b0010;
47         #100;
48         a = 4'b0010;
49         b = 4'b0100;
50         #100;
51         a = 4'b1111;
52         b = 4'b0001;
53         #100;
54         ci = 1'b1;
55
56     end
57 endmodule
```

这段代码每间隔 100 ps 会给出两个不同的输入，如果上面的模块正常工作，那么输出结果应该是两个数的和。仿真结果如下所示。

可以发现仿真的结果与实验指导书一致，这说明上面的四位加法器可以正确工作。事实上，仅从仿真结果就可以确定加法是否正确运行。



## 4、工程实现

### 4.1 Top文件

首先创建 Top 源文件用于控制将加法器的输入和输出与实验板上的数码管进行连接。Top代码内容如下。

```

module Top(
    input clk_p,
    input clk_n,
    input [3:0] a,
    input [3:0] b,
    input reset,
    output led_clk,
    output led_do,
    output led_en,
    output wire seg_clk,
    output wire seg_en,
    output wire seg_do
);

    wire CLK_i;
    wire Clk_25M;

    IBUFGDS IBUFGDS_inst (
        .O(CLK_i),
        .I(clk_p),
        .IB(clk_n)
    );

    wire [3:0] s;
    wire co;
    wire [4:0] sum;
    assign sum = {co, s};

```

```

    adder_4bits U1(
        .a(a),
        .b(b),
        .ci(1'b0),
        .s(s),
        .co(co)
    );

```

```

    reg [1:0] clkdiv;
    always@(posedge CLK_i)
        clkdiv <= clkdiv + 1;
    assign Clk_25M=clkdiv[1];

```

```

    display DISPLAY(
        .clk(Clk_25M),
        .rst(1'b0),
        .en(8'b00000011),
        .data({27'b0, sum}),
        .dot(8'b00000000),
        .led("~{11'b0, sum}),
        .led_en(led_en),
        .led_do(led_do),
        .seg_clk(seg_clk),
        .seg_en(seg_en),
        .seg_do(seg_do)
    );

```

```

endmodule

```

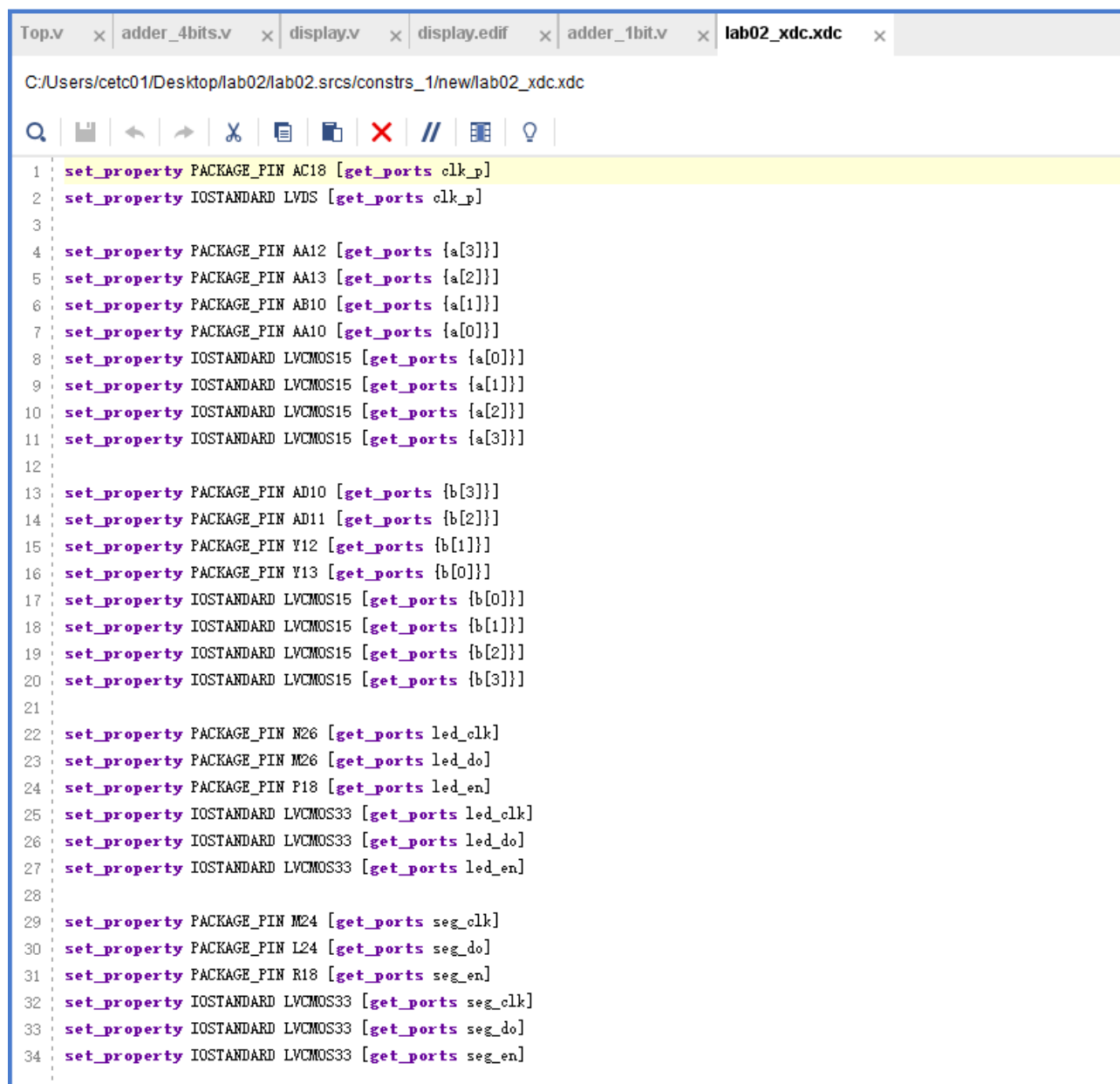


## 4.2 display IP

课程组提供了 display IP 核，在 Source 区关联源文件。

## 4.3 约束文件

新建约束文件 lab02\_xdc.xdc，输入管脚约束代码。



```
Top.v x adder_4bits.v x display.v x display.edif x adder_1bit.v x lab02_xdc.xdc x
C:/Users/cetc01/Desktop/lab02/lab02.srscs/constrs_1/new/lab02_xdc.xdc

1 set_property PACKAGE_PIN AC18 [get_ports clk_p]
2 set_property IOSTANDARD LVDS [get_ports clk_p]
3
4 set_property PACKAGE_PIN AA12 [get_ports {a[3]}]
5 set_property PACKAGE_PIN AA13 [get_ports {a[2]}]
6 set_property PACKAGE_PIN AB10 [get_ports {a[1]}]
7 set_property PACKAGE_PIN AA10 [get_ports {a[0]}]
8 set_property IOSTANDARD LVCMOS15 [get_ports {a[0]}]
9 set_property IOSTANDARD LVCMOS15 [get_ports {a[1]}]
10 set_property IOSTANDARD LVCMOS15 [get_ports {a[2]}]
11 set_property IOSTANDARD LVCMOS15 [get_ports {a[3]}]
12
13 set_property PACKAGE_PIN AD10 [get_ports {b[3]}]
14 set_property PACKAGE_PIN AD11 [get_ports {b[2]}]
15 set_property PACKAGE_PIN Y12 [get_ports {b[1]}]
16 set_property PACKAGE_PIN Y13 [get_ports {b[0]}]
17 set_property IOSTANDARD LVCMOS15 [get_ports {b[0]}]
18 set_property IOSTANDARD LVCMOS15 [get_ports {b[1]}]
19 set_property IOSTANDARD LVCMOS15 [get_ports {b[2]}]
20 set_property IOSTANDARD LVCMOS15 [get_ports {b[3]}]
21
22 set_property PACKAGE_PIN N26 [get_ports led_clk]
23 set_property PACKAGE_PIN M26 [get_ports led_do]
24 set_property PACKAGE_PIN P18 [get_ports led_en]
25 set_property IOSTANDARD LVCMOS33 [get_ports led_clk]
26 set_property IOSTANDARD LVCMOS33 [get_ports led_do]
27 set_property IOSTANDARD LVCMOS33 [get_ports led_en]
28
29 set_property PACKAGE_PIN M24 [get_ports seg_clk]
30 set_property PACKAGE_PIN L24 [get_ports seg_do]
31 set_property PACKAGE_PIN R18 [get_ports seg_en]
32 set_property IOSTANDARD LVCMOS33 [get_ports seg_clk]
33 set_property IOSTANDARD LVCMOS33 [get_ports seg_do]
34 set_property IOSTANDARD LVCMOS33 [get_ports seg_en]
```

## 4.4 下载验证

与 lab01 类似，将生成的比特流文件上传到实验板上，观察实验是否符合预期。实验板上低四位和次高四位代表两个加数，LED的低四位代表两加数的和，第五位是进位；同时两个七段数码管代表数值结果。

## 5、致谢

感谢刘老师以及三位实验助教学长学姐的悉心指导和帮助。

在完成本次实验的过程中，我因为不够细心，导致管脚约束部分输入出错，这导致仿真结果正确的前提下无法在实验板上取得正确的结果，感谢助教学姐专门帮我检查错误，使得我能顺利完成本次实验内容；感谢助教学长为向大家讲解 Verilog 语言的编程逻辑和基本语法，这些让我们能够更轻松的完成接下来的实验内容，谢谢几位的付出。

感谢上海交通大学的计算机系统结构实验室，为我们准备了相关的实验环境以及 FPGA 开发板，学校和学院提供的优秀设备让我们受益匪浅。