计算机系统结构试验 Lab02: 4-bit Adder

姓名：N/A

摘要

在 Lab02 中，我使用 Verilog 语言成功实现了 4-bit Adder 功能。通过本次实验，我学会了如何在项目中编写多个 module 的组合逻辑，并进一步理解了 Vivado 的语法、项目流程、仿真方法和调试技巧。这次实验给我带来了很多收获。

目录

1. 实验目的

（1）掌握 Xilinx 逻辑设计工具 Vivado 的基本操作；
（2）掌握 Verilog HDL 进行简单的逻辑设计；
（3）使用功能仿真；
（4）约束文件的使用和直接写法；
（5）生成 bitstream 文件；
（6）熟悉系统硬件开发的基本实验流程。

2. 原理分析

2.1 Vivado 工程的基本组成

（1）adder_1bit.v 文件
（2）adder_4bits.v 文件
（3）Top.v 文件
（4）adder_4bits_tb.v 激励文件
（5）lab02_xdc.xdc 管脚约束文件

2.2 adder_1bit 的原理

adder_1bit 是一个一位全加器，输入为两个一位操作数 a、b 以及一位进位输入 ci；输出为一位加法结果 s 与一位进位输出 co。根据全加器逻辑，当 a、b、ci 中至少两个为 1 时，co 为 1；当 a、b、ci 有奇数个 1 时，s 为 1。代码如下：

```
3   module adder_1bit (
4       input  a,
5       input  b,
6       input  ci,
7       output s,
8       output co
9   );
10
11    wire s1, c1, c2, c3;
12    and (c1, a, b), (c2, b, ci), (c3, a, ci);
13    xor (s1, a, b), (s, s1, ci);
14    or (co, c1, c2, c3);
15
16  endmodule
17
```

2.3 adder_4bits 的原理

adder_4bits 由 4 个 adder_1bit 组合而成。整个模块输入为两个 4 位操作数 a、b 以及一位进位输入 ci；输出为 4 位加法结果与一位进位输出 co。模块中添加了 ct 用来连接一位全加器之间的进位。低位全加器的进位输出是高位全加器的进位输入，最低位全加器的进位输入为 ci，最高位全加器的进位输出为 co。其余输入输出与 a、b、s 对应连接。代码如下：

```
3   module adder_4bits (
4       input [3:0] a,
5       input [3:0] b,
6       input ci,
7       output [3:0] s,
8       output co
9   );
10
11    wire [2:0] ct;
12
13    adder_1bit
14        a1 (
15            .a (a[0]),
16            .b (b[0]),
17            .ci(ci),
18            .s (s[0]),
19            .co(ct[0])
20        ),
21        a2 (
22            .a (a[1]),
23            .b (b[1]),
24            .ci(ct[0]),
25            .s (s[1]),
26            .co(ct[1])
27        ),
28        a3 (
29            .a (a[2]),
30            .b (b[2]),
31            .ci(ct[1]),
32            .s (s[2]),
33            .co(ct[2])
34        ),
35        a4 (
36            .a (a[3]),
37            .b (b[3]),
38            .ci(ct[2]),
39            .s (s[3]),
40            .co(co)
41        );
42
43  endmodule
```

3. 功能实现

为了使用已实现的 4 位全加器，拟用实验板上的 8 个 Switch 对应二组 4 位二进制输入，用 4 个 LED 发光二极管对应输出，并用 2 个七段数码管显示运行结果。故本实验需要用到 display.v 这个七段数码管 SEGMENT 和 LED 发光二极管显示模块（实验室提供 display 核，以网表文件形式给出）。编写 Top 模块代码如下：

```
3   module Top (
4       input clk_p,
5       input clk_n,
6       input [3:0] a,
7       input [3:0] b,
8       input reset,
9
10      output led_clk,
11      output led_do,
12      output led_en,
13
14      output wire seg_clk,
15      output wire seg_en,
16      output wire seg_do
17  );
18
19    wire CLK_i;
20    wire Clk_25M;
21
22    IBUFGDS IBUFGDS_inst (
23        .O (CLK_i),
24        .I (clk_p),
25        .IB(clk_n)
26    );
```

```
28    wire [3:0] s;
29    wire co;
30    wire [4:0] sum;
31    assign sum = {co, s};
32
33    adder_4bits U1 (
34        .a (a),
35        .b (b),
36        .ci(1'b0),
37        .s (s),
38        .co(co)
39    );
40
41    reg [1:0] clkdiv;
42    always @(posedge CLK_i) clkdiv <= clkdiv + 1;
43    assign Clk_25M = clkdiv[1];
44
45    display DISPLAY (
46        .clk(Clk_25M),
47        .rst(1'b0),
48        .en(8'b00000011),
49        .data({27'b0, sum}),
50        .dot(8'b00000000),
51        .led(~{11'b0, sum}),
52        .led_clk(led_clk),
53        .led_en(led_en),
54        .led_do(led_do),
55        .seg_clk(seg_clk),
56        .seg_en(seg_en),
57        .seg_do(seg_do)
58    );
59
60  endmodule
```

4. 结果验证

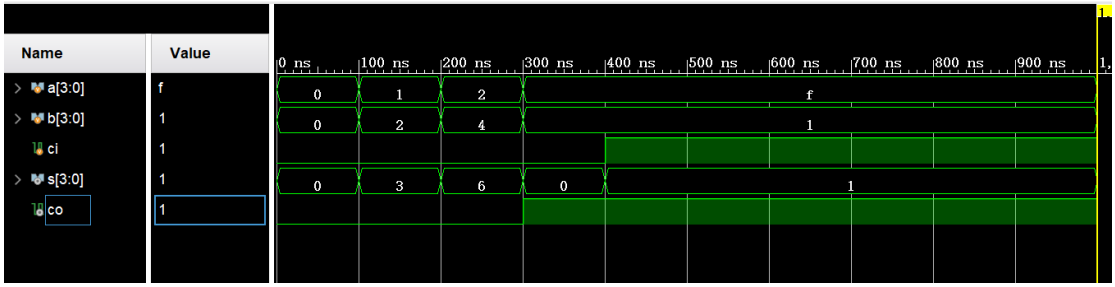
4.1 测试用激励文件

首先，按照实验导书上的要求，编写激励文件。设置各输入初值。代码如下：

```
module adder_4bits_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
```

4.2 加法逻辑的测试

接下来进行仿真，结果如下所示：



上图中可以看到，{co，s}= a + b + ci，加法功能正常。

5. 管脚约束

根据实验指导书编写管脚约束文件如下：

```
set_property PACKAGE_PIN AC18 [get_ports clk_p]
set_property IOSTANDARD  LVDS [get_ports clk_p]

set_property PACKAGE_PIN AA12 [get_ports {a[3]}]
set_property PACKAGE_PIN AA13 [get_ports {a[2]}]
set_property PACKAGE_PIN AB10 [get_ports {a[1]}]
set_property PACKAGE_PIN AA10 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[0]}]

set_property PACKAGE_PIN AD10 [get_ports {b[3]}]
set_property PACKAGE_PIN AD11 [get_ports {b[2]}]
set_property PACKAGE_PIN Y12  [get_ports {b[1]}]
set_property PACKAGE_PIN Y13  [get_ports {b[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[3]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[0]}]

set_property PACKAGE_PIN N26 [get_ports led_clk]
set_property PACKAGE_PIN M26 [get_ports led_do]
set_property PACKAGE_PIN P18 [get_ports led_en]
set_property IOSTANDARD LVCMOS33 [get_ports led_clk]
set_property IOSTANDARD LVCMOS33 [get_ports led_do]
set_property IOSTANDARD LVCMOS33 [get_ports led_en]

set_property PACKAGE_PIN M24 [get_ports seg_clk]
set_property PACKAGE_PIN L24 [get_ports seg_do]
set_property PACKAGE_PIN R18 [get_ports seg_en]
set_property IOSTANDARD LVCMOS33 [get_ports seg_clk]
set_property IOSTANDARD LVCMOS33 [get_ports seg_do]
set_property IOSTANDARD LVCMOS33 [get_ports seg_en]
```

6. 总结与反思

在 Lab02 中，我再次熟悉了 Vivado 的开发环境，并且对 Verilog HDL 的基本语法有了进一步的复习。此外，我还学习了如何使用 module 模块化编写组合逻辑，如何使用 begin-end 块编写时序激励文件。

我要感谢课程组为我们准备的详细指导书。在接下来的学习中，我计划进一步学习 Verilog HDL 的知识，尝试设计通用的模块，提高代码复用性，并学习管脚约束文件的写法。