计算机系统结构实验 Lab2 实验报告

- 姓名：夏鸿驰
- 学号：520021910965
- 日期：2022年4月26日.

目录

1. 实验概述

1.1 实验名称

- FPGA 基础实验：4-bit Adder

1.2 实验目的

1. 掌握 Xilinx 逻辑设计工具 Vivado 的基本操作
2. 掌握 VerilogHDL 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

2. 实验步骤

2.1 新建工程

- 此系列步骤与Lab1中的创建步骤几乎一致（除命名之外）

2.1.1 启动桌面 Vivado 2018.3 开发工具

2.1.2 点击 Create Project

2.1.3 弹出 New Project，建立一个新工程，点击 Next

2.1.4 输入工程名称 lab02，选择工程位置，确认勾选中 Create project subdirectory 后点击 Next

2.1.5 选择 RTL Project 工程类型，勾选 Do not specify sources at this time，点击Next

2.1.6 选择实验板的 FPGA 参数：Family 选 Kintex-7，Package 选 ffg676，Speed grade 选-2，在具体型号中选 xc7k325tffg676-2，点击 Next

2.1.7 点击 Finish 结束工程的创建

2.2 添加文件

- 此系列前五个步骤与Lab1中的添加文件步骤一致（除文件名之外）

2.2.1 点击左侧区 Flow Navigator 下的 Project Manager->Add Sources 或中间区 Sources 的“+”号，打开 Add Sources 对话框

2.2.2 选择第二项 Add or Create Design Sources，用来添加或新建 Verilog HDL 源文件，点击 Next

2.2.3 若已有源文件或内核文件，可选 Add Files 项以添加文件。这里是要新建 1 位全加器模块文件，选择 Create File

2.2.4 弹框 Create Source File 中输入 adder_1bit 文件名，点击 OK

2.2.5 点击 Finish

2.2.6 在弹出的 Define Module 中输入设计模块所需的端口，并设置端口方向；完成后点击 OK
其中：

- a, b 代表两个位的操作数
- ci 代表输入的进位（上次加法得到的）
- s 代表此位加法得到的结果
- co 代表此位加法得到的进位

2.2.7 向创建的文件中的源代码区输入以下代码：

```
`timescale 1ns / 1ps

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

```

即完成了一位加法器的实现

2.2.8 同样的步骤，我们创建一个新的源文件，命名为adder_4bits， 代表要实现的4 位加法器，输入模块所需的端口，并设置端口方向，若端口属总线型，勾选 Bus，并由 MSB 和 LSB 确定宽度点击 OK

此时：

- a[3:0], b[3:0] 代表两个4位的操作数
- ci 代表一位加法时输入的进位（上次加法得到的）
- s[3:0] 代表a,b两个4位的数加法得到的结果
- co 代表一位加法时得到的进位

2.2.9 打开刚刚创建的源文件adder_4bits，输入以下的代码：

```
`timescale 1ns / 1ps

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

```

2.3 功能仿真

- 同样的，创建激励文件的方式和Lab1中一致，但文件名不一致，代码不一致。

2.3.1 创建激励测试文件。在中间区 Sources 栏点击“+”号或于左
侧区 PROJEU MANAGER 下选择 Add Source

2.3.2 在 Add Sources 选择 Add or Create Simulation Source，点击 Next

2.3.3 选择 Create File 创建一个仿真激励文件

2.3.4 激励文件测试名可输入 adder_4bits_tb，点击 OK

2.3.5 点击 Finish，创建激励文件不需要对外端口，点击 OK

2.3.6 在弹出的对话框点击 YES

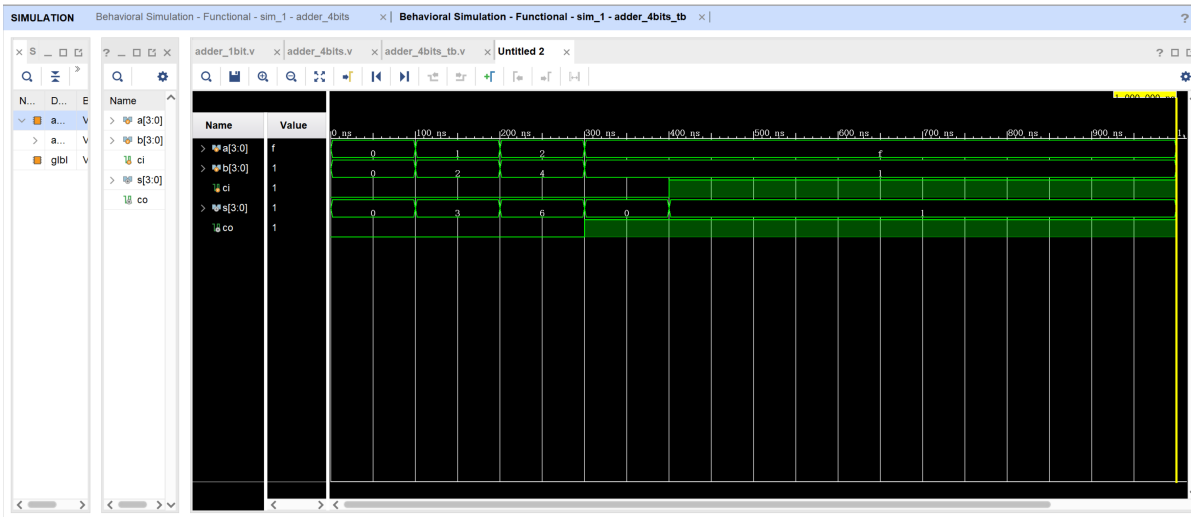
2.3.7 在Source 区 Simulation Sources 下，打开测试文件 adder_4bit_tb，在其中对要仿真的模块进行实例化和激励代码的编写（并点击保存），如下：

```
`timescale 1ns / 1ps

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

```

2.3.8 开始仿真：点击左侧区的 Run Simulation 并选择 Run Behavioral Simulation。下图为仿真运行后得到的仿真波形图样例：

Figure:



2.4 工程实现

- 在本实验中要用实验板上的 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

2.4.1 将4位加法器的输出赋予LED和七段数码管显示，需要创建一个顶层源文件，可命名Top

- 首先先创建源文件Top
- 进入Define Module，这里我们略过端口定义，后可在源程序中自行添加。

- 双击Top模块，添加以下端口变量和代码语句：

```
`timescale 1ns / 1ps

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_clk(led_clk),
    .led_en(led_en),
    .led_do(led_do),
    .seg_clk(seg_clk),
    .seg_en(seg_en),
```

```
    .seg_do(seg_do)

    );
endmodule

```

2.4.2 Top.v中用到了一个display IP核（已转换为网表），接下来需要添加该核。

- 在Add Sources 中，点击Add Sources。
- 在Add Source Files中同时选择一对与该核同前缀名的端口和网表文件
- 点击Finish，则在Sources区的顶层源文件下关联了这一对源文件

2.4.3 用直接写法添加约束文件：

- 首先先打开Add Sources对话框, 选择第一项创建新的约束文件。
- 再接着打开这新建的空白的约束文件
- 添加如下约束代码：

```
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[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {a[3]}]

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[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS15 [get_ports {b[3]}]

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]
```

3. 实验心得

3.1 实验重难点

- 本实验的重难点在于通过一位加法器的设计从而得到四位加法器的实现，通过分时的步骤进行Verilog HDL 代码的调试，通过直接编写法创建约束文件及通过加入display IP核来工程实现。

3.2 实验感想

- 在本次实验过程中，从一位加法器的设计从而得到四位加法器的实现这个新颖的编写Verilog HDL代码的方式让我感到十分的新奇，也给了我对于创建更加复杂的Verilog HDL文件的一些思路和想法。此外，在实验中遇到的困难都会在微信群里得到老师及时的反馈，所以要在这里感谢老师的付出。

4. 参考资料

[1] 2022计算机系统结构实验指导书lab2