

四川大学计算机学院、软件学院

实验报告

学号：2023141460321 姓名：孙谦昊 专业：计算机科学与技术班级：行政七班 第 11 周

课程名称	计算机组成原理实验	实验课时	1-4 节
实验项目	4 位加法器的设计与实现	实验时间	2024. 11. 14
实验目的	1.掌握ISE软件设计的方法 2.掌握一位全加器的工作原理和逻辑功能 3.掌握简单的Verilog代码编写		
实验环境	SWORD 4.0 套件、ISE Design Suite 14.7 、Vivado Design Suite 2014.3 及以上版本		
实 验 内 容 (算法、程序、步骤和方法)	<div>1. 建立新的工程</div> <div>双击桌面上“Xilinx ISE 14.7”图标，启动 ISE 软件，选择 File>New Project 选项，弹出新建工程引导界面。在对话框中输入工程名称 adder_4bit，并指定工程路径。点击 Next 按钮进入下一页选 Kintex7 XC7K325T 芯片，采用 FFG676 封装。另外，选择 Verilog 作为默认的硬件描述语言。再点击 Next 按钮进入下一页确认新建工程的信息。若无误，点击 Finish 。</div> <div>2. 硬件代码或原理图设计与输入</div> <div>在工程管辖区（左上角）任意位置单击鼠标右键，在弹出的菜单中选择 New Source 命令，弹出新建源代码对话框：Select Source Type。选择 Verilog Module，命名为 adder_1bit。单击 Next 进入下一步，忽略模块的端口定义，点 next 继续，点击 Finish 完成创建。完成对一个 Verilog 模块模板创建，并且在源代码编辑区打开。在源代码编辑中编写：</div> <pre>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);</pre>		

```
endmodule
```

保存。在过程管理区（Synthesize-XST）双击 Check Syntax 检查是否有错误。

随后以 Verilog 代码输入的形式实现 4 位加法器。首先新建源文件，选择 Verilog module，命名为 adder_4bit。单击 Next 进入下一步，忽略模块的端口定义，点 next 继续，点击 Finish 完成创建。在代码编辑器中输入代码：

```
module adder_4bits(  
    input [size-1 : 0] a,  
    input [size-1 : 0] b,  
    input ci, // carry input  
    output [size-1 : 0] s, // sum  
    output co  
);  
parameter size = 4;  
wire [2:0] ct; // carry temp  
adder_1bit al(.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
```

到目前为止，用 Verilog 已经完成一个 4 位全加器。保存并检查语法正确后就开始进行综合。

3. 代码综合（ Synthesize-XST）

在工程管理区的 View 中选择 Implementation，并选中要综合的模块 adder_4bits, 然后在过程管理区中双击 Synthesize-XST，开始综合过程。

4. 软件仿真（Simulation）

综合通过后，进入仿真测试阶段。在工程管理区将 View 设置为 Simulation, 在任意位置单击鼠标右键，并在弹出的菜单中选择 New Source，弹出新建源代码对话框，在类型中选择 Verilog Test Fixture，输入测试文件名：adder_4bit_tb，单击下一步。

选择要进行测试的模块：adder_4bits 模块。点击 Next。单击 Finish 按钮，ISE 会在源代码编辑区自动生成测试模块的代码。在 initial...end 块中的 “//Add stimulus here” 后面添加测试激励代码：

```
// Add stimulus here  
a = 4'b0001;  
b = 4'b0010;  
#100;  
a = 4'b1111;
```

```

b = 4'b0001;
#100;
ci = 1'b1;

```

完成测试文件编辑并保存后，确认工程管理区中 View 选项设置为 Simulation，并选中 adder_4bit_tb 模块。

右键单击其中的 Simulate Behavioral Model 项，选择弹出菜单中的 Process Properties 项，会弹出如图 1.13 所示的属性设置对话框，其中 Simulation Run Time 就是仿真时间的设置，可将其修改为任意时长（默认为 1000ns）。在工程管理区选中测试代码（顶层代码），然后在过程管理区双击 Simulate Behavioral Model。配置正确情况下可看到仿真结果与逻辑结果是一致的。

5. LED

新建源文件，选择 Verilog module，命名为 LED_ctrl。单击 Next 进入下一步，忽略模块的端口定义，点 next 继续，点击 Finish 完成创建。LED 控制代码如下：

```

module LED_ctrl(
    input [15:0] sw,
    input clk200P,
    input clk200N,
    input reset,
    output reg led_do,
    output led_pen,
    output led_clk,
    output led_clr
);
wire Clk_100M;
reg [15:0] sw_d1;
parameter piso_shift = 16;
reg [piso_shift-2:0] sw_shift;
reg [16:0] counter = 17'h0;
wire [15:0] shift_load;

SWORD_LED_CLK ClkGen100M
(// Clock in ports
.CLK_IN1_P(clk200P), // IN
.CLK_IN1_N(clk200N),
// Clock out ports
.CLK_OUT1(Clk_100M), // OUT
// Status and control signals
.RESET(RESET), // IN

```

```
.LOCKED(LOCKED));
```

```
always@(posedge Clk_100M)
    if(!reset) sw_d1 <= 16'h0;
    else sw_d1 <= sw;
assign shift_load = sw^sw_d1;
always @(posedge Clk_100M)
    if (shift_load) begin
        sw_shift <= sw[piso_shift-2:0];
        led_do <= ~sw[15];
        counter <= 17'h1ffff;
    end
    else begin
        sw_shift <= {sw_shift[piso_shift-3:0], 1'b0};
        led_do <= ~sw_shift[14];
        counter <= {1'b0, counter[16:1]};
    end
end
assign led_clk = Clk_100M & counter[0];
assign led_clr = reset;
assign led_pen = 1'b1;
endmodule
```

添加 CLK IP 核，在工程管理区（左上角）任意位置单击鼠标右键，在弹出的菜单中选择 New Source 命令，选择 IP(CORE Generator & Architecture Wizard)，输入文件名 SWORD_LED_CLK。点 next 继续，等待界面加载完成，在 Search IP Catalog 输入 clock，并选择 Clocking Wizard。点击 finish，稍等片刻弹出配置界面。注意在配置界面 Source 处选择 Differential clock capable pin。点击 Next，在配置界面将输出频率设置成 100MHz，并点击 Generate，完成添加。

新建源文件，选择 Verilog module，命名 top。单击 Next 按钮进入端口定义对话框：Define Module。略过此处，点击 Next 后可以看到配置信息，点击 Finish 完成添加。输入如下代码：

```
module top(
    input [3:0] a,
    input [3:0] b,
    input clk200P,
    input clk200N,
    input RSTN,
    output LEDCLK,
    output LEDDT,
    output LEDCLR
```

```

    );

    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) );
    LED_ctrl U2 (
        .sw({11'b0, sum}),
        .clk200P(clk200P),
        .clk200N(clk200N),
        .reset(RSTN),
        .led_do(LEDĐT),
        // .led_pen(led_pen),
        .led_clk(LEDCLK),
        .led_clr(LEDCLR)
    );
    Endmodule

```

6. 引脚约束

添加引脚约束文件，在工程管理区单击鼠标右键，点击 New Source，在类型中选择 Implementation Constraints File，输入文件名：myucf，系统会生成一个空白的约束文件并打开。

添加如下代码：


```

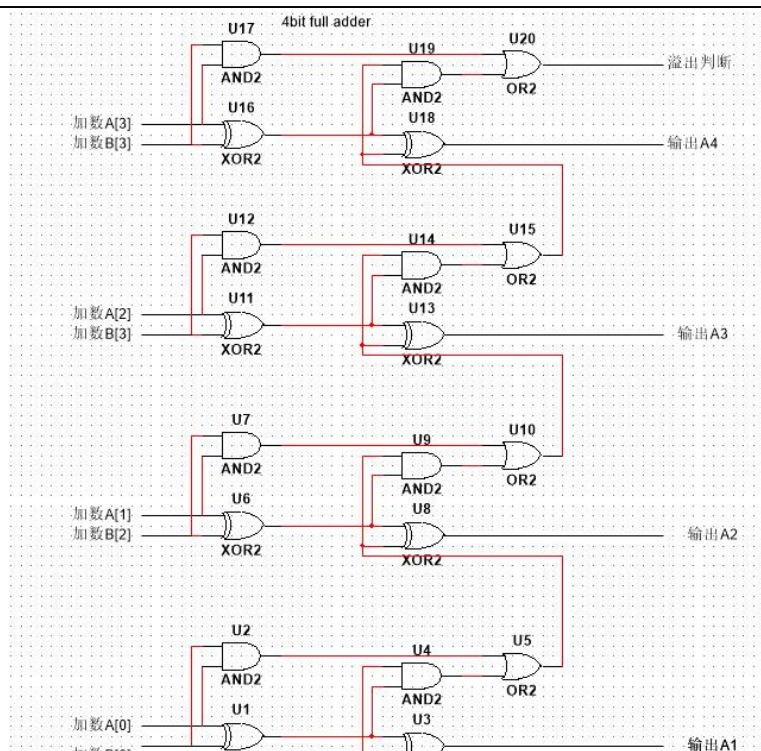
NET "clk200P"    LOC="AC18" | IOSTANDARD = LVDS;
NET "clk200N"    LOC="AD18" | IOSTANDARD = LVDS;

NET "RSTN"       LOC = W13 | IOSTANDARD = LVCMOS18 ;

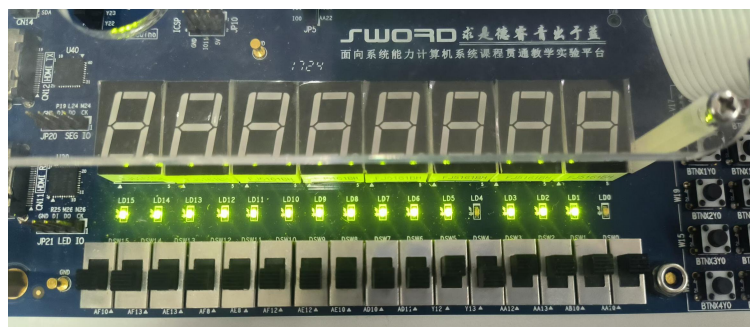
NET "a[0]"       LOC = AA10 | IOSTANDARD = LVCMOS15 ;
NET "a[1]"       LOC = AB10 | IOSTANDARD = LVCMOS15 ;
NET "a[2]"       LOC = AA13 | IOSTANDARD = LVCMOS15 ;
NET "a[3]"       LOC = AA12 | IOSTANDARD = LVCMOS15 ;
NET "b[0]"       LOC = Y13  | IOSTANDARD = LVCMOS15 ;
NET "b[1]"       LOC = Y12  | IOSTANDARD = LVCMOS15 ;
NET "b[2]"       LOC = AD11 | IOSTANDARD = LVCMOS15 ;
NET "b[3]"       LOC = AD10 | IOSTANDARD = LVCMOS15 ;

```

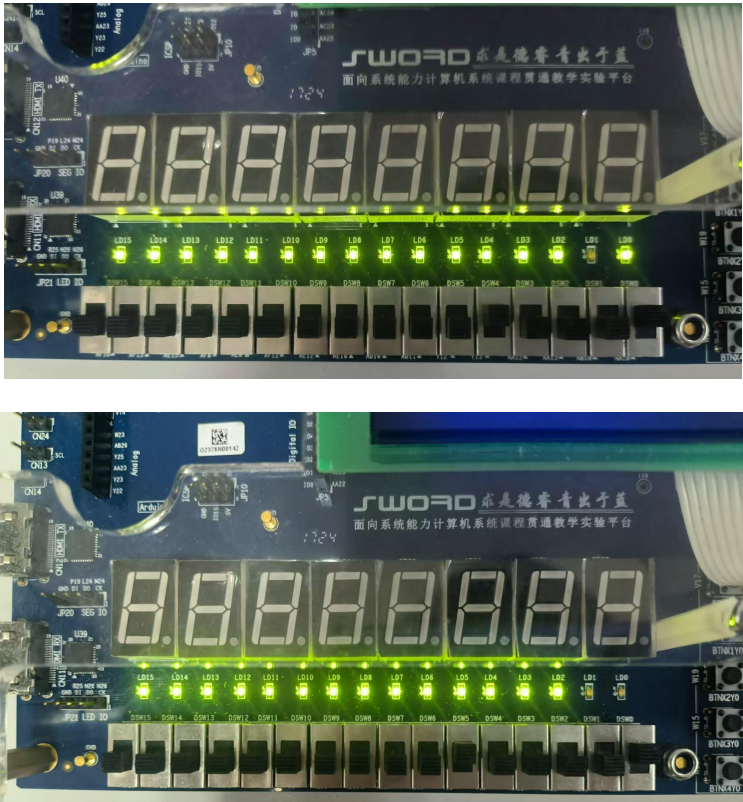
	<div>NET "LEDCLK" LOC = N26 IOSTANDARD = LVCMOS33 ; NET "LEDCLR" LOC = N24 IOSTANDARD = LVCMOS33 ; NET "LEDDT" LOC = M26 IOSTANDARD = LVCMOS33 ;</div> <div>7. 实现 (Implement Design) 及生成流代码 首先是 Synthesize-XST, 然后是实现设计 (Implement Design), 最后是生成可下载到硬件中的比特流文件 (Generate Programming File)。</div> <div>8. 下载与测试 等待生成比特流文件后, 将板卡正确连接到 PC。连接电源线并打开电源, 将比特文件下载到板卡上观察实验结果。在工程管理区中选中 top, 然后在下方的 process: top 中双击 Configure Target Device, 打开下载工具 IMPACT。遇到警告直接略过。然后系统会自动启动 IMPACT。先双击 IMPACT Flows 中的 Boundary Scan, 然后在空白处单击右键, 然后单击 Initialize Chain 初始化板卡。对弹出的提示单击 NO。在接下来弹出的窗口选择 cancel。选中芯片, 单击右键, 选择 Launch File Assignment Wizard。在工程目录下找到生成的 bit 文件, 选中然后单击 open, 下载 bit 文件。如遇提示 Attach SPI BPI PROM (如图 1. 36), 选择 NO。在接下来弹出的窗口选择 cancel。回到软件主界面, 选中芯片单击右键, 选择 program。</div> <div></div>
数据记录 和计算	<div>1. 四位加法器的逻辑电路图 观察 adder_1bit 文件中代码, 注意到存在如下代码: and (c1, a, b), (c2, b, ci), (c3, a, ci); xor (s1, a, b), (s, s1, ci); or (co, c1, c2, c3);</div> <div>结合 adder_4bit 文件中对于 adder_1bit 的函数调用流程, 可知本实验中用代码实现的四位加法器的逻辑电路图应为:</div>



2. 该实验涉及的拨动开关为 DSW0、DSW1、DSW2、DSW3、DSW4、DSW5、DSW6、DSW7, 分别代表 a0, a1, a2, a3, b0, b1, b2, b3。涉及的 LED 为 LD0、LD1、LD02、LD3、LD4、LD5、LD6、LD7。其中, 在输入阶段 LD0、LD1、LD02、LD3 表示加数 a 的值。LD4、LD5、LD6、LD7 表示加数 b 的值。在结果展示阶段, LD0、LD1、LD02、LD3、LD4、LD5 表示 a+b 的值。
3. 设置 a 的值为: 1111, b 的值为: 0010。理论上 a+b 的计算结果应为: 10001, 在 LD0、LD1、LD02、LD3、LD4、LD5 上表现为“灭亮亮亮灭”。实际测试结果为:



实际测试结果与理论结果相同。多次进行试验, 观察到实际试验结果与理论结果均相同。表明借助试验设备成功实现了四位全加法器的功能。

	<div data-bbox="417 211 1164 1009"></div>
结 论 (结 果)	实验结果与理论结果保持一致。
小 结	1. 本实验中了解了如何借助 SWORD 4.0 套件、ISE Design Suite 14.7 、 Vivado Design Suite 2014.3 及以上版本完成项目的创建。 2. 复习了四位全加器的逻辑电路图和计算原理。
指导老师 评 议	<div data-bbox="372 1691 495 1720">成绩评定：</div> <div data-bbox="852 1691 1030 1720">指导教师签名：</div>