专业：电气工程及其自动化
姓名：严旭铧
学号：3220101731
日期：2023 年 10 月 9 日
地点：东 三 406 教 室

浙江大学实验报告

课程名称：电路与电子技术实验Ⅰ 指导老师：祁才君 成绩：
实验名称：仪器使用练习 实验类型：电学实验 同组学生姓名：褚玘铖

实验 半导体二极管特性测试

一、 实验目的

1. 了解数字电路基础知识
2. 掌握简单数字逻辑电路的设计
3. 学习 FPGA 开发工具 Quartus 和 ModelSim 的使用

二、 实验内容

用 FPGA 开发工具 Quartus 和 ModelSim 以及 DE10 板实现以下功能：

1. 输入是三位二进制数 A,B,C，要求当输入是 2 或 3 的倍数时输出等于逻辑 1，其它情况，输出等于 0。
2. 设计并实现一位二进制全加器。
3. （拓展）某客厅四周有 4 个房间，每个房间门口有一个开关，客厅中间有一盏灯 S。试设计一个逻辑电路，要求每个开关都能控制灯 S 的亮灭，并在 DE10 开发板上模拟此逻辑电路的功能。

三、 实验器材

硬件：
搭载 64 位 Windows11 家庭中文版操作系统的笔记本电脑
Terasic DE10 开发板，芯片型号为 10M50DAF484C7G
软件及版本信息：
Quartus (Quartus Prime 22.1std) Lite Edition
ModelSim - Intel FPGA Starter Edition Model Technology ModelSim - Intel FPGA Edition vsim 2020.1 (Quartus Prime 20.1)

四、 实验方案，实验数据记录、处理与分析

内容一：二进制数 ABC 为 2 或 3 的倍数时输出 1

1. 实验步骤

（1） 根据实验要求列出真值表（如表 1）。

输入				输出
A	B	C	对应十进制数	Z
0	0	0	0	0
0	0	1	1	0
0	1	0	2	1
0	1	1	3	1
1	0	0	4	1
1	0	1	5	0
1	1	0	6	1
1	1	1	7	0

表 1 内容一真值表

（2） 创建文件夹“task 1”，路径为“D:\PLD\task 1”（这是我设定的路径）。

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_arith.all;

entity TEST is
    port (  A,B,C: in std_logic;
            Z: out std_logic);
end TEST;

architecture BEHAV of TEST is
begin

process(A,B,C)
variable tmp:std_logic_vector(2 downto 0);
begin
tmp:=A&B&C;
case tmp is
 when "000" => Z<='0';
 when "001" => Z<='0';
 when "010" => Z<='1';
 when "011" => Z<='1';
 when "100" => Z<='1';
 when "101" => Z<='0';
 when "110" => Z<='1';
 when "111" => Z<='0';
 when others => Z<='X';
end case ;

end process;

end BEHAV;
```

图 1 内容一设计代码

```
entity TEST_tb is
end TEST_tb;

architecture STRU of TEST_tb is

component TEST is
    port (  A,B,C: in std_logic;
            Z: out std_logic);
end component ;

signal A,B,C : std_logic;
signal Z : std_logic;

begin

MAP1: TEST port map(A=>A,B=>B,C=>C,Z=>Z);
tb : process
        begin
         A<='0';B<='0';C<='0';
         wait for 10ns;
         A<='0';B<='0';C<='1';
         wait for 10ns;
         A<='0';B<='1';C<='0';
         wait for 10ns;
         A<='0';B<='1';C<='1';
         wait for 10ns;
         A<='1';B<='0';C<='0';
         wait for 10ns;
         A<='1';B<='0';C<='1';
         wait for 10ns;
         A<='1';B<='1';C<='0';
         wait for 10ns;
         A<='1';B<='1';C<='1';
         wait for 10ns;
```

图 2 内容一仿真代码

（3） 打开 ModelSim 软件，单击菜单栏“File”，选择“Change Directory…”，将路径更改为（2）中设定的路径。

（4） 新建 VHDL 语言文件，“File-New-Source-VHDL”。

（5） 按照课件给的部分源码，依据 VHDL 语言规范要求编写设计代码。保存为“TEST.vhd”。

（6） 编译：“Compile-Compile-选中要编译的文件-Compile”。如有报错则修改代码直至“Transcript”

栏中出现“ERROR：0”为止。

（7） 按照(4)(5)(6)的步骤编写仿真测试代码，并保存在同一目录下，文件名为“TEST_tb.vhd”。

（8） 仿真测试：仿真代码编译无误后，在左侧“Library”栏中找到“work”（一般在最上或最下），单击左侧“+”，双击“TEST_tb”。在“Objects”一栏中选中 A、B、C、Z，右击，选择“Add wave”，在上方将“Run Length”调为 10ns（与仿真代码中设定的“wait for”值对应）。多次单击“Run”按钮，直至出现完整的 8 段波形，检查是否与真值表对应。若对应则说明该设计代码能够在仿真环境下实现要求的功能。

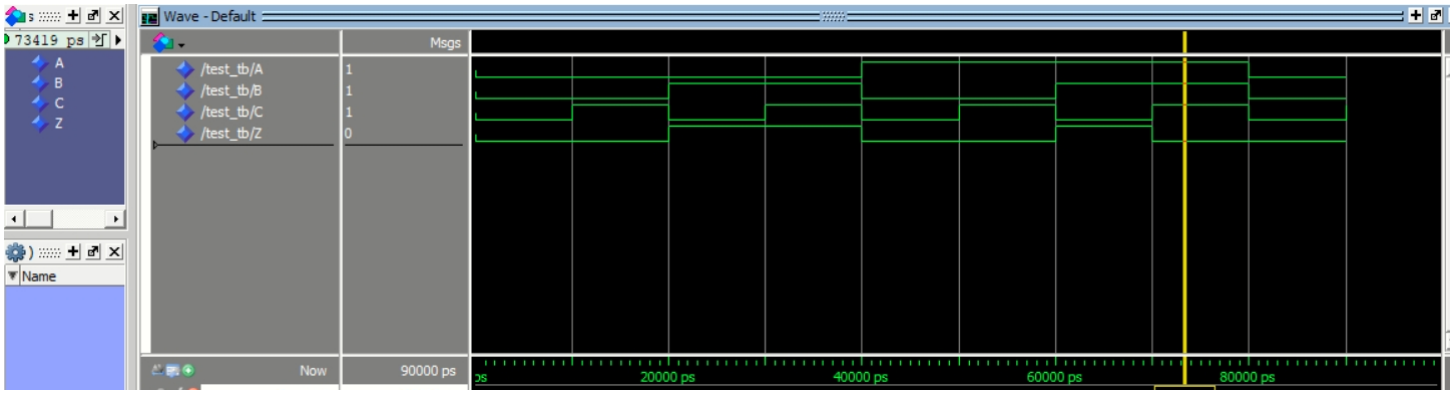


图 3 内容一仿真波形

（9） 打开 Quartus 软件，单击菜单栏选择“File-New Project Wizard”在跳出的窗口中点选“Next”，选择 directory 为“D:\PLD\task 1”（步骤 2 中设定的路径），之后两个 name 都填“TEST”（与设计文件文件名一致），“Next-OK-Next-Next”。

（10） 选择硬件设备：“Device Family”栏，“Family”选择“MAX 10(DA/DD/DF/DC/SA/SC/SL)”。“Available devices:”栏选择“10M50DAF484C7G”。单击“Next-Next”，“Summary”窗口检查无误后单击“Finish”。

（11） 加载文件：Project Navigator – Files – 右击 Files – Add/Remove Files in Project – 添加文件“TEST.vhd”。

（12） 编译分析：Tasks 栏中双击 Compile Design 左侧蓝色三角形，待其分析完成。

（13） 查看 RTL 原理图：Compile Design 菜单展开 – Analysis & Synthesis 菜单展开 – Netlist Viewers – RTL Viewer。

（14） 管脚分配：Assignments – Pin Planner – 为每个输入输出分配 Location，Location 值与电路板元件对应关系可从说明书查到。输入点分配逻辑开关元件，输出点分配

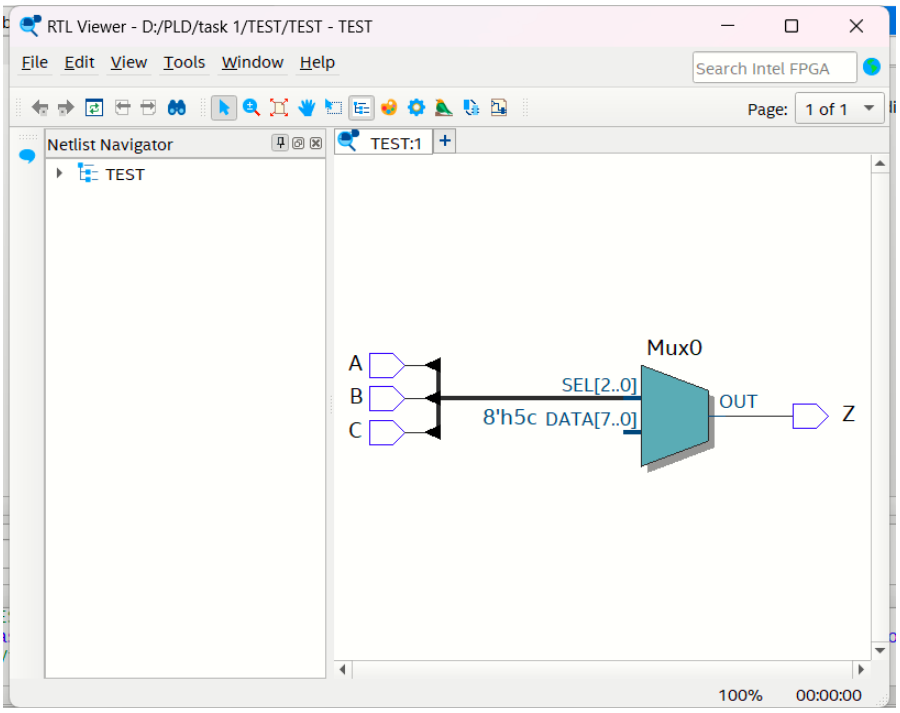


图 4 内容一 RTL 原理图

LED 元件。这里我选用 A – C10; B – C11; C – D12; Z – A8。

（15） 生成项目文件：双击 Assembler (Generate programming files)左侧蓝色小三角形，待加载完成后，双击下方 Program Device (Open Programmer)

（16） 在 DE10 开发板上实现：用 USB 线连接开发板和电脑，在弹出的窗口单击 Hardware Setup，将 No Hardware 改为 USB-0，单击 Close。单击 Start，此时开发板上的灯会有一个闪烁，之后 10 个 LED 灯变暗。操作分配给 ABC 的逻辑开关改变电路逻辑值，记录分配给输出 Z 的 LED 灯的亮暗情况。

2. 实验数据

表 2 内容一实验结果

逻辑开关输入				LED 输出
A PIN_C10	B PIN_C11	C PIN_D12	对应十进制数	Z PIN_A8
关	关	关	0	暗
关	关	开	1	暗
关	开	关	2	亮
关	开	开	3	亮
开	关	关	4	亮
开	关	开	5	暗
开	开	关	6	亮
开	开	开	7	暗

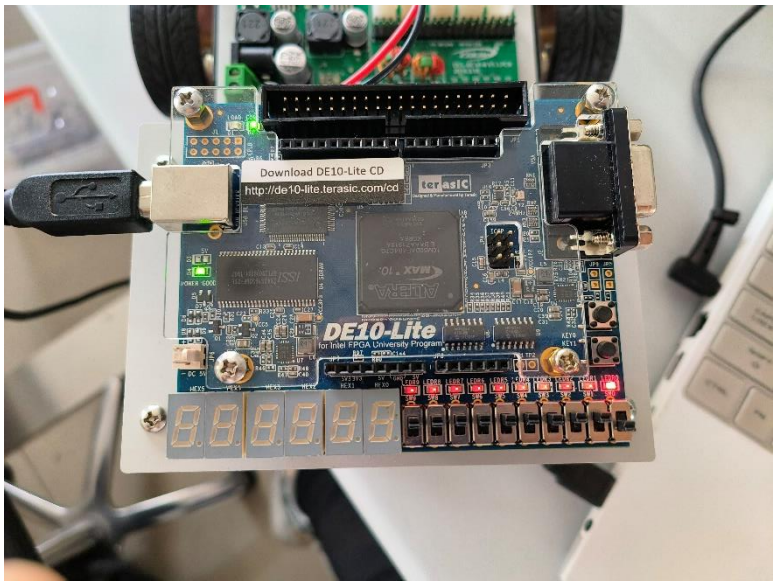


图 5 内容一实物验证（110 状态，亮）

在开发板上测试得到的结果与真值表一一对应，实验成功。

3. 数据处理与分析

内容一中设计代码的编写我采用了顺序 CASE 语句，将真值表直接映射在代码中，相当于枚举法。由于该实验仅涉及 8 种情况，直接用 CASE 语句描述比逻辑语句更为清晰更为简单。当然，这样得到的 RTL 图没有那么好看。

内容二：设计并实现一位二进制全加器

1. 实验步骤

（1） 根据实验要求列出真值表（如表 3）。

表 3 内容二真值表

输入			输出	
A	B	C0（进位）	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

（2） 实验步骤与内容一基本相同，创建“task 2”文件夹，路径改为“D:\PLD\task 2”。设计文件名为“AD.vhd”，仿真文件名为“AD_tb.vhd”。

（3） 设计代码和仿真代码

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_arith.all;

entity AD is
    port (  A,B,C0: in std_logic;
            C,S: out std_logic);
end AD;


architecture BEHAV of AD is
begin

P1:process(A,B,C0)
variable tmp:std_logic_vector(2 downto
begin
tmp:=A&B&C0;
case tmp is
 when "000" => C<='0';S<='0';
 when "001" => C<='0';S<='1';
 when "010" => C<='0';S<='1';
 when "011" => C<='1';S<='0';
 when "100" => C<='0';S<='1';
 when "101" => C<='1';S<='0';
 when "110" => C<='1';S<='0';
 when "111" => C<='1';S<='1';
 when others => C<='X';S<='X';
end case ;

end process;

end BEHAV;
```

图 7 内容二设计代码

```
use IEEE.std_logic_arith.all;

entity AD_tb is
end AD_tb;

architecture STRU of AD_tb is
component AD is
    port (  A,B,C0: in std_logic;
            C,S: out std_logic);
end component;

signal A,B,C0 :std_logic;
signal C,S :std_logic;

begin
map1: AD port map (A=>A,B=>B,C0=>C0,C=>C,S=>S);
process
    begin
        A<='0';B<='0';C0<='0';
        wait for 10ns;
        A<='0';B<='0';C0<='1';
        wait for 10ns;
        A<='0';B<='1';C0<='0';
        wait for 10ns;
        A<='0';B<='1';C0<='1';
        wait for 10ns;
        A<='1';B<='0';C0<='0';
        wait for 10ns;
        A<='1';B<='0';C0<='1';
        wait for 10ns;
```

图 6 内容二仿真代码

（4） 仿真波形图

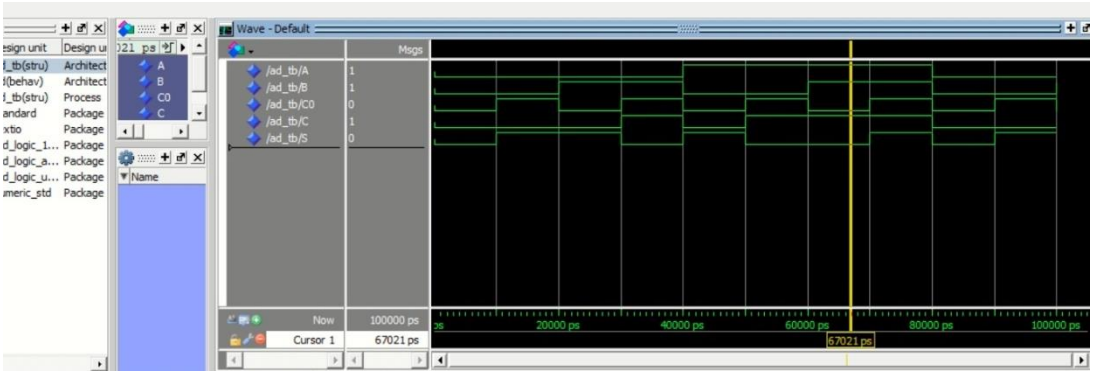


图 8 内容二仿真波形图

（5） 管脚分配

表 4 内容二管脚分配

in A	Input	PIN_C11	7	B7_N0
in B	Input	PIN_C10	7	B7_N0
out C	Output	PIN_A9	7	B7_N0
in C0	Input	PIN_D12	7	B7_N0
out S	Output	PIN_A8	7	B7_N0

（6） RTL 原理图

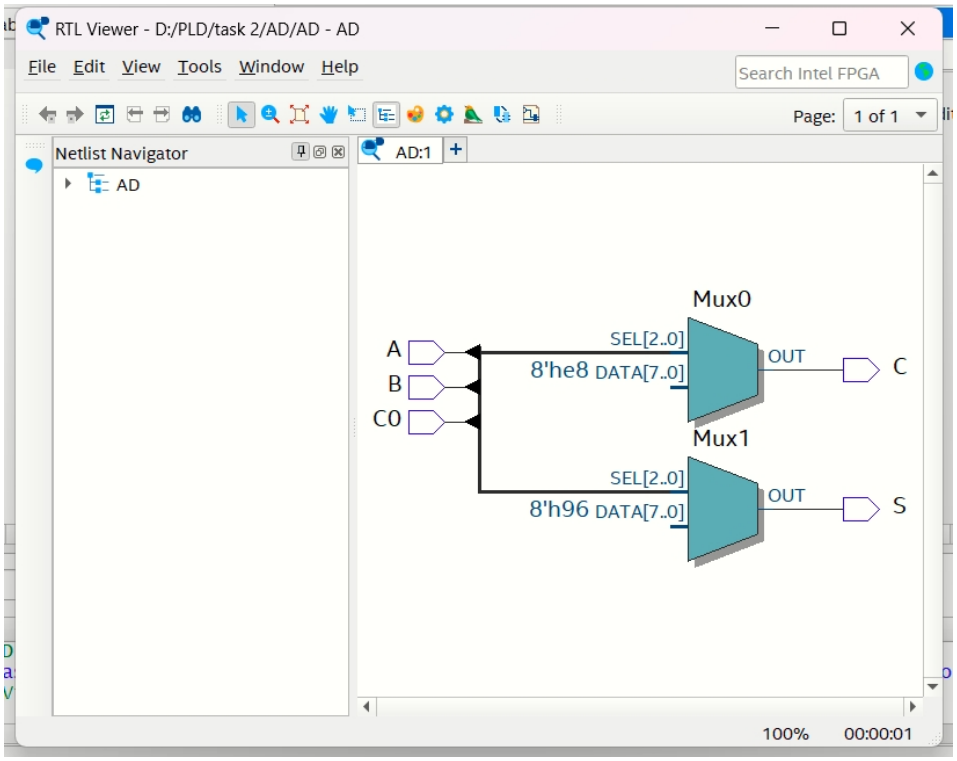


图 9 内容二 RTL 原理图（枚举法）

（7） 实物图

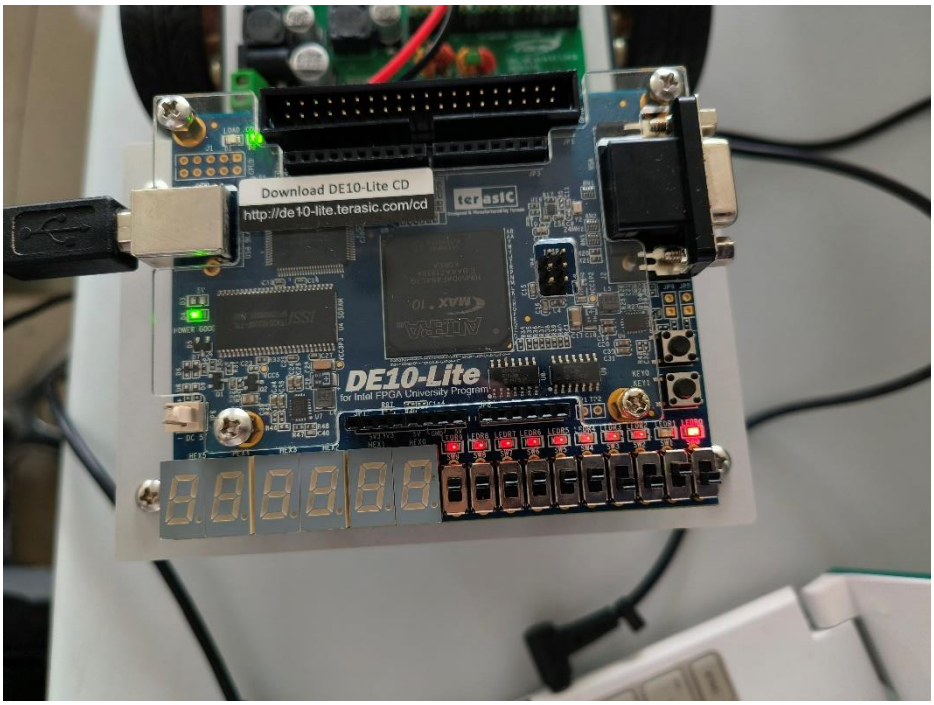


图 10 内容二实物验证（010 状态，C 暗 S 亮）

2. 实验数据

表 5 内容二实验结果

逻辑开关输入			LED 输出	
A PIN_C11	B PIN_C10	C0（进位） PIN_D12	C PIN_A9	S PIN_A8
关	关	关	暗	暗
关	关	开	暗	亮
关	开	关	暗	亮
关	开	开	亮	暗
开	关	关	暗	亮
开	关	开	亮	暗
开	开	关	亮	暗
开	开	开	亮	亮

3. 数据处理与分析

这里直接套用内容一的代码，仍然采用枚举法 CASE 语句写设计代码。当然，一位二进制全加器还可以采用逻辑语句书写。

将其转化为 VHDL 语言即为：

```
architecture STRU of AD is
begin
C<=(A and B) or (A and C0) or (B and C0);
S<=A xor B xor C0;
end STRU;
```

对应的 RTL 图为：

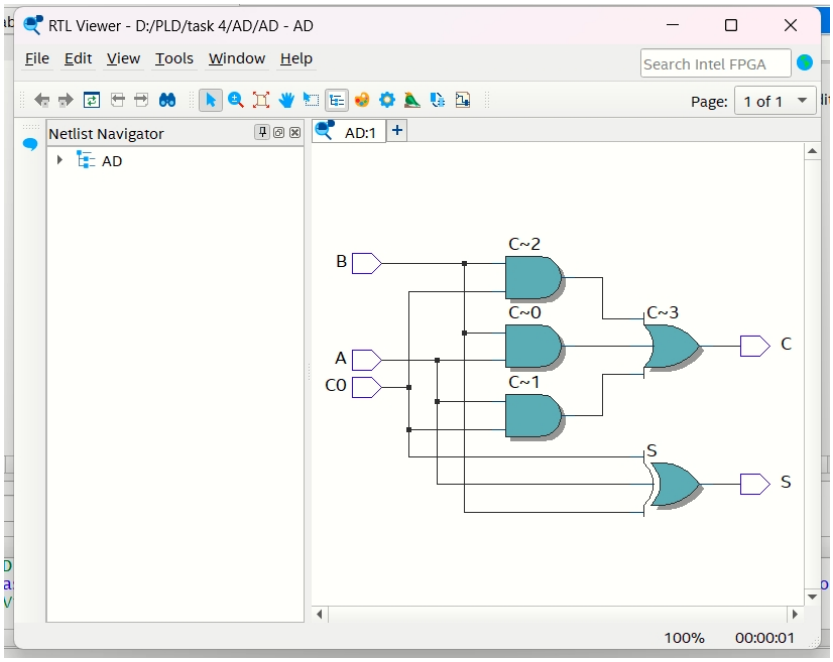


图 11 内容二 RTL 原理图（逻辑语句法）

内容三：（拓展）四个开关一盏灯

1. 实验步骤

（1） 分析题目条件，列出真值表。转化之后其实就是有奇数个开关打开时灯亮，偶数个开关打开时灯灭。

（2） 实验步骤与内容一基本相同，创建“task 3”文件夹，路径改为“D:\PLD\task 3”。设计文件名为“SW.vhd”，仿真文件名为“SW_tb.vhd”。

（3） 设计代码和仿真代码：

表 6 内容三真值表

序号	开关输入状态				输出
	A	B	C	D	S
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	1
5	0	1	0	1	0
6	0	1	1	0	0
7	0	1	1	1	1
8	1	0	0	0	1
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	1

12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	0

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
use IEEE.numeric_std.all;
use IEEE.std_logic_arith.all;

entity SW is
    port (  A,B,C,D: in std_logic;
            S: out std_logic);
end SW;
architecture BEHAV of SW is
begin

P1:process(A,B,C,D)
variable tmp:std_logic_vector( 3 downto 0);
begin
tmp:=A&B&C&D;
case tmp is
 when "0000" => S<='0';
 when "0001" => S<='1';
 when "0010" => S<='1';
 when "0011" => S<='0';
 when "0100" => S<='1';
 when "0101" => S<='0';
 when "0110" => S<='0';
 when "0111" => S<='1';
 when "1000" => S<='0';
 when "1001" => S<='0';
 when "1010" => S<='1';
 when "1011" => S<='1';
 when "1100" => S<='0';
 when "1101" => S<='1';
 when "1110" => S<='1';
 when "1111" => S<='0';
 when others => S<='X';
```

图 13 内容三设计代码

```
entity SW_tb is
end SW_tb;

architecture STRU of SW_tb is
component SW is
    port (  A,B,C,D: in std_logic;
            S: out std_logic);
end component;

signal A,B,C,D :std_logic;
signal S :std_logic;

begin
map1: SW port map (A=>A,B=>B,C=>C,D=>D,S=>S);
process
    begin
        A<='0';B<='0';C<='0';D<='0';
        wait for 10ns;
        A<='0';B<='0';C<='0';D<='1';
        wait for 10ns;
        A<='0';B<='0';C<='1';D<='0';
        wait for 10ns;
        A<='0';B<='0';C<='1';D<='1';
        wait for 10ns;
        A<='0';B<='1';C<='0';D<='0';
        wait for 10ns;
        A<='0';B<='1';C<='0';D<='1';
        wait for 10ns;
        A<='0';B<='1';C<='1';D<='0';
        wait for 10ns;
        A<='0';B<='1';C<='1';D<='1';
        wait for 10ns;
        A<='1';B<='0';C<='0';D<='0';
```

图 12 内容三仿真代码

（4） 仿真波形图

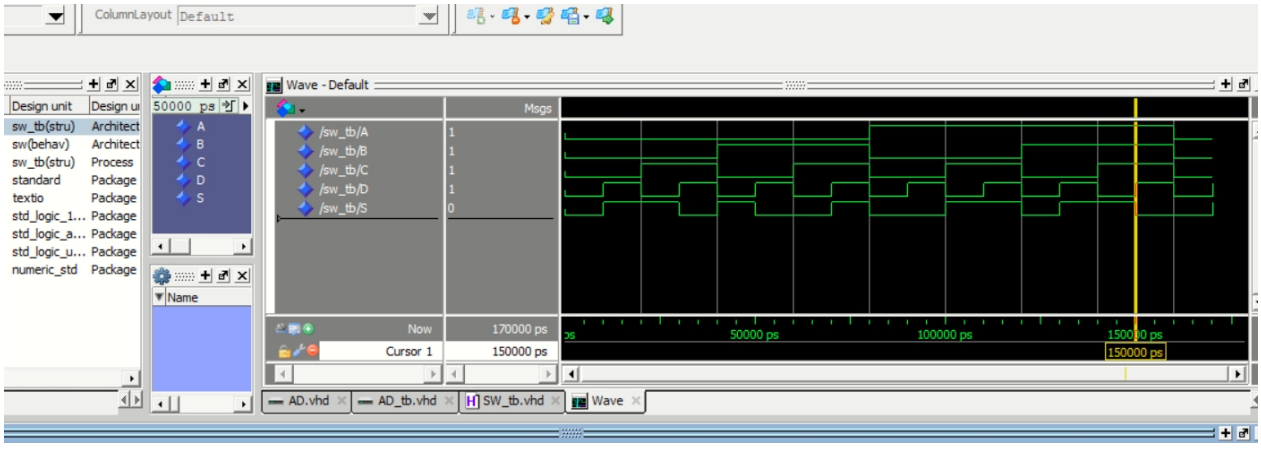


图 14 内容三仿真波形图

（5） 管脚分配

表 7 内容三管脚分配

A	Input	PIN_C12	7	B7_N0
B	Input	PIN_D12	7	B7_N0
C	Input	PIN_C11	7	B7_N0
D	Input	PIN_C10	7	B7_N0
S	Output	PIN_A8	7	B7_N0

（6） RTL 原理图：

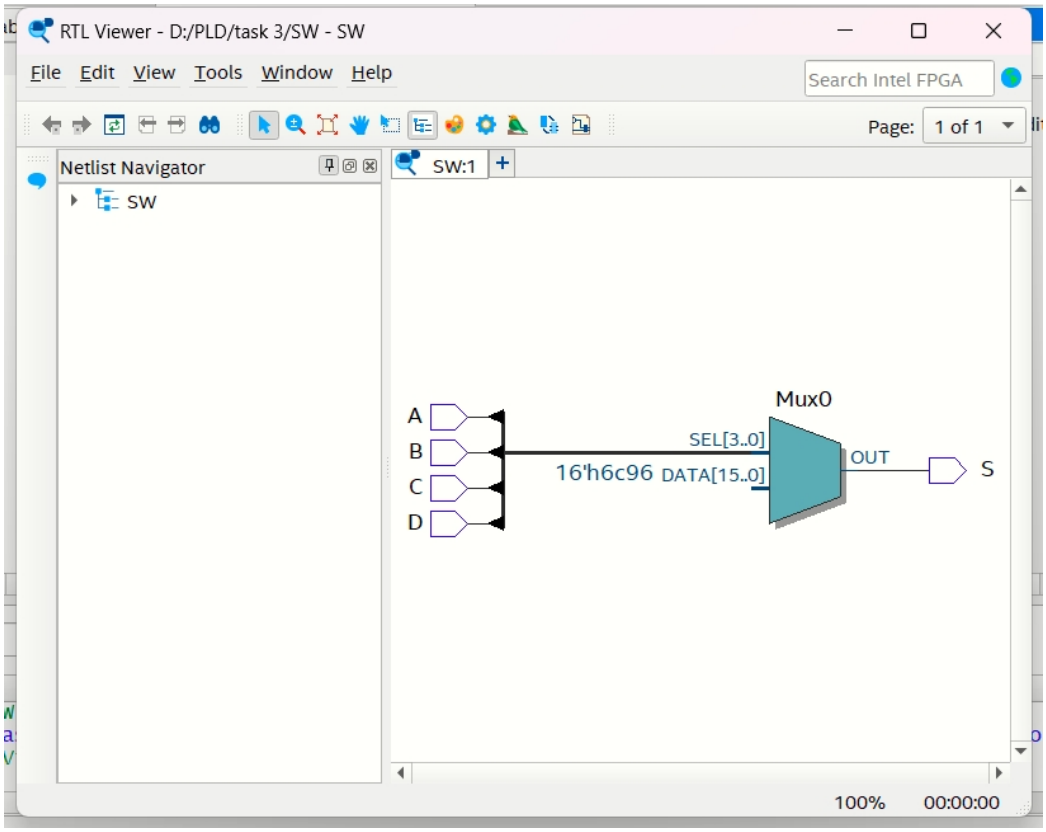


图 15 内容三 RTL 原理图（枚举法）

（7） 实物图

图 16 内容三实物验证（BCD 开关打开，灯亮）

2. 实验数据

表 8 内容三实验结果

序号	开关输入状态				输出
	A	B	C	D	S
0	关	关	关	关	暗
1	关	关	关	开	亮
2	关	关	开	关	亮
3	关	关	开	开	暗
4	关	开	关	关	亮
5	关	开	关	开	暗
6	关	开	开	关	暗
7	关	开	开	开	亮
8	开	关	关	关	亮
9	开	关	关	开	暗
10	开	关	开	关	暗
11	开	关	开	开	亮
12	开	开	关	关	暗
13	开	开	关	开	亮
14	开	开	开	关	亮
15	开	开	开	开	暗

与真值表对应，完成要求。

3. 数据处理与分析

本实验难点主要在条件转化。注意到四个开关控制同一盏灯，实际上每一次开关状态变化就伴随着灯的变化，“一步一动”。对于 n 个开关来说，若有 s 个开关处于开的状态，变动一次，开着的开关数为 s+1 或 s-1，此时 s 由奇数变为偶数（或者偶数变为奇数），$s\pm1 \mod 2=0$（或 1），符合逻辑开关的元件特性。因此，条件实际上就是转换成了，有奇数个开关处于开启状态时，灯亮；偶数个开关开启时，灯灭。

四、 实验心得体会

1. 在使用 ModelSim 和 Quartus 软件时，一定要注意文件路径/目录的设置。同一个工程，其所有文件应该存放在一个文件夹中，且文件命名均采用字母、数字、下划线。这是一个好习惯。而且在 Quartus 中项目名称应该和文件名对应，否则也可能出错。
2. ModelSim 的代码调试功能只能说很烂，代码写好之后需要先保存再编译，若编译出现错误，它能给你提供的信息很少（只有少分号或者 begin/end 时候能显示具体原因），要进行调试也不如 DEV C++ 等编译器那样便捷。为了更好解决调试问题，我的思路是，在代码的每一块之间空出明显间隔，可以采用“注释-编译-逐步取消注释-编译-更正-编译”的方法，先定位是哪一块的代码出错了，再具体到哪一行，有几处 ERRORS。
3. 环境配置和软件安装：

   我从 Quartus 官网上下载的是 22.1 新版，但是安装后发现没有集成 ModelSim，此时需要重新去官网

下载 ModelSim 入门版。这需要在 Quartus20.1 版本及以前的选项中选择 Individual Files，再点选进行下载。

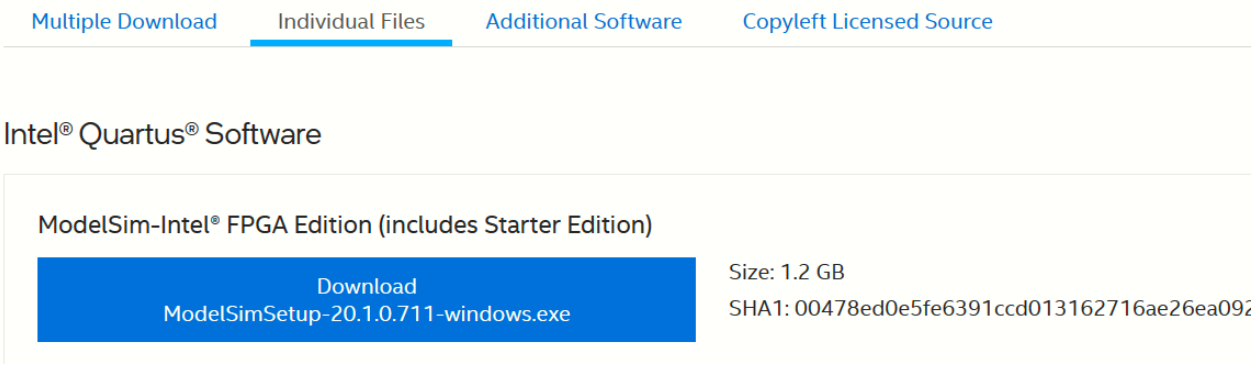
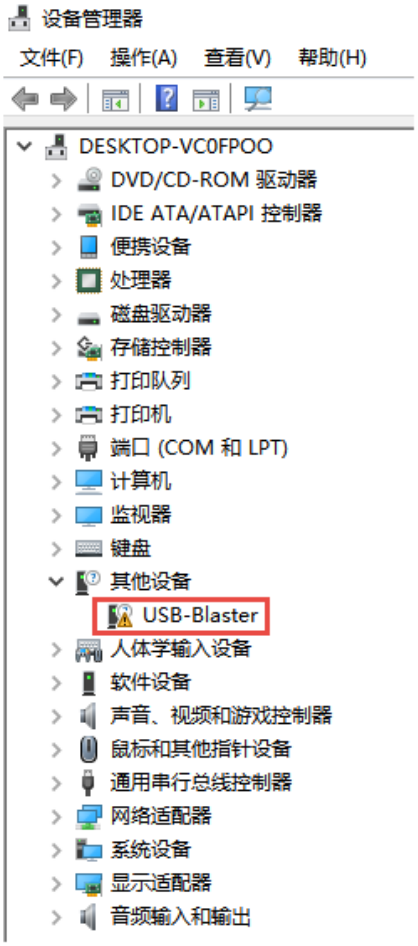


图 17 ModelSim 的下载

在 Quartus 上进行到最后一步，即用 USB 线连接电脑和开发板时，第一次连接可能会出现无法识别 USB 端口的情况。此时是因为 USB Blaster 驱动没有安装或更新。在较新版本的 Quartus 安装包中是有集成这一驱动的，只要在设备管理器安装即可。右击如图红框标注的图表，选择“更新驱动程序”，然后选择手动查找并安装。在 Quartus 安装路径里“D:\Quartus\quartus\drivers\usb-blaster”（quartus 之前的是自己安装的路径），查找并安装驱动完毕后，就能正常识别出硬件了



4. VHDL 语言实际是比较好上手的，在现成的源码上做修改就能得到需要的代码。重要的是将设计要求转化为逻辑原理图和真值表，再选用合适的语句将其表述出来。

5. 一个疑惑：

在使用 ModelSim 进行波形仿真时，有时会丢失在 Objects 栏中的元件，以至于无法选中输入输出元件，无法 Add Wave。我出现过一次这个情况，解决途径是重来一遍，重新建目录再做仿真。但是在实验室遇到一个同学同样出现这种情况，除了重头再来不知道有没有什么其他易行的解决方案？或者出现这样情况的原因是什么？