内容1

1. 三位数码管下载到实验板 (2019-09-09)

首先在平台 http://10.64.17.119/ 相应课程中下载 SCAN 工程文件,并解压到文件夹

安装 USB-BLASTER 驱动

打开 Quartus 文件,打开已有的工程,找到解压 SCAN 工程的文件夹,选中 qbf 工程文件

待 Quartus 完全打开工程,选择 TOOLS-Programmer,添加 hardware setup, USB-BLASTER 驱动

选中相应 SOF 文件, START

数码管实验现象

下载完成之后,数码管左侧三个数字几乎同时闪烁,并且标有 101 和 100 的 红色 1ed 小灯亮起

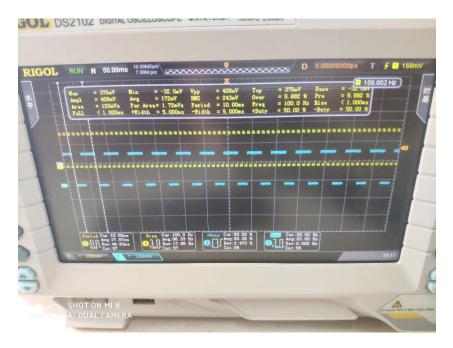
底下一排开关,拨动,相应绿色 1ed 小灯会亮

每4个小开关为一组,对应权值从左到右为8421,拨动小开关,数码管显示值会根据相应权值变化

实验板左上角的 91 号开关,三个小开关,拨动,数码管闪烁频率会变低,并且左侧小开关影响变低的程度>右侧的小开关。

2. 示波器测量截图 (2019-09-13)





位选信号和时钟信号如上图。

时钟信号为黄色波形,从示波器中,可以得到,其频率为100HZ,且波峰波谷时间几乎一样。

位选信号为蓝色波形,从示波器中可以得到,其频率为 100HZ, 符合 5 个为一周期。同时,波峰的长度明显比波谷的长度长一些。

内容 2

实验目的:

了解 Quatrus 的使用方法以及基于 Quartus 的设计流程

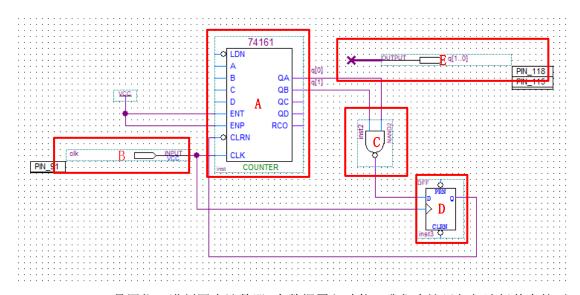
了解 DDA 系列实验板的使用

了解图形设计和 VHDL 语言(硬件描述语言)两种设计方式

了解层次设计的过程

设计:

模四计数器示意图:



A: 74161 是四位二进制同步计数器,有数据置入功能。我们在这里仅仅选择其中的两位,构成两位计数器,5个脉冲信号一个周期;

- B: 时钟信号;
- C: NAND2门,输入到DFF;
- D: DFF 触发器,上升沿激活;
- E: 输出接口。

实验现象:

115 和 118 号红色 LED 小灯交替闪烁,并且将频率调低,可以看到,一共有 4 种闪烁情况,(灭灭,灭亮,亮灭,亮亮)。但是灭的时间,比其他三种情况长一些。

模四计数器,4种现象,5信号1周期,验证如上,正确。

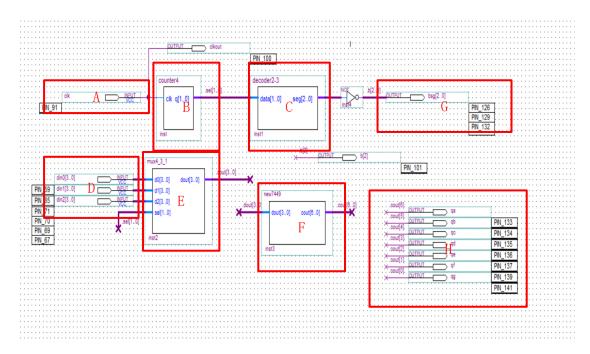
两种验证方式的比较:

仿真验证:分为功能仿真和时序仿真。功能仿真仅仅关心输出和输入的逻辑关系是否正确,不考虑时间延时信息。时序仿真不仅反应出输出和输入的逻辑关系,同时还计算了时间的延时信息,是与实际系统更接近的一种仿真结果。

下载验证: 有现实中信号的延时问题, 是最真实的验证方式。

实现:

顶层电路设计



- A: 时钟信号输入;
- B: 模四计数器接受时钟信号进行计数,模四计数器将值进行累加**并**输出 给 2-3 译码器和三选一选择器;
- C: 2-3 译码器控制位选信号的输出,接受模四计数器传来的信号,输出数码管的位选信号;
 - D: 三路数据,由拨码开关输入;
- E: 三路选择器同时可以输入三路数据与模四计数器传来的信号(,输出给 BCD-7 段译码器其中的一路数据;
- F: BCD-7 段译码器接受三选一选择器传来的信,并将其转换为 7 位数码管的段选信号并输出;
 - G: 输出位选信号;
 - H: 输出段选信号。

下载截图: (输入 0-f)







7449 优化方案

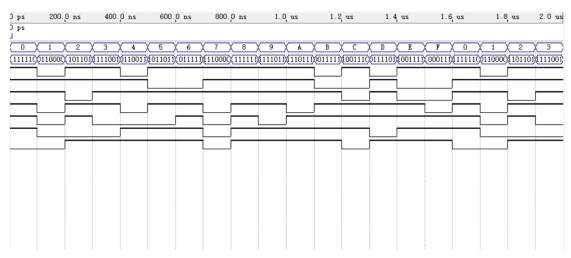
通过 VHDL 语言对 7449 模块进行重写,完善 7449 的功能定义自己的 BCD-7 段译码器。

new7449 硬件描述语言如下:

```
    library ieee;

2. use ieee.std_logic_1164.all;
3. entity new7449 is
4. port(dout: in std_logic_vector(3 downto 0);
          cout: out std_logic_vector(6 downto 0));
6. end new7449;
7. architecture change of new7449 is
8. begin
          cout <= "1111110" when dout =
9.
                                            "0000" else
10.
                        "0110000"
                                                 "0001"
                                  when
                                        dout =
                        "1101101" when
11.
                                        dout
                                                 "0010"
                                                        else
12.
                        "1111001"
                                  when
                                        dout = "0011"
                                                        else
13.
                        "0110011"
                                                 "0100"
                                                        else
                                   when
                                        dout =
14.
                        "1011011"
                                        dout = "0101"
                                  when
                                                        else
                        "1011111"
15.
                                  when
                                        dout = "0110"
                                                        else
                        "1110000"
                                        dout = "0111"
16.
                                  when
                                                        else
17.
                        "111111" when
                                        dout = "1000"
                                                        else
18.
                        "1111011" when
                                        dout = "1001"
                                                        else
19.
                        "1110111" when
                                        dout = "1010"
                                                        else
                        "0011111"
                                        dout = "1011"
20.
                                   when
                                                        else
21.
                        "1001110" when
                                        dout = "1100"
                                                        else
22.
                        "0111101"
                                   when
                                        dout = "1101"
                                                        else
23.
                         "1001111" when
                                        dout = "1110" else
24.
                        "1000111" when
                                        dout = "1111";
25. end architecture change;
```

验证如下



通过仿真结果,看到,新的BCD-7段译码器,将不同的数据译码成了数码管对应的段选信号。

总结

掌握的知识点如下:

Quartus 平台基本使用

BCD 译码规则

两种仿真方式的区别

DDA 实验平台的基本知识

掌握的技能如下:

使用模四计数器、2-3 译码器、三路选择器、BCD-7 段译码器基本的图形设计法

人文:

增强自己的动手能力和实践能力,自己动手做实验,增强了对数字系统 实验的热爱