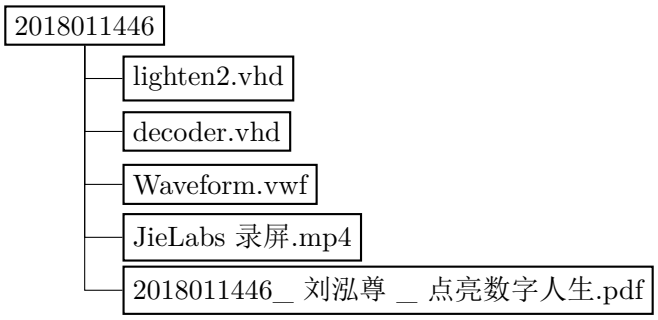
点亮数字人生：实验报告

刘泓尊　2018011446　计 84

2020 年 3 月 26 日

目录

1 File Structure



2 实验目的

(1) 通过数码管点亮程序，熟悉 VHDL 语言，了解掌握硬件程序的编写规范

(2) 掌握 EDA 软件的使用方法和工作流程

(3) 进一步理解可编程芯片的工作原理

3 实验任务

设计一个数码管显示实验，要求七段数码管有规律地显示数列 (奇数列，偶数列和自然数列)，尽可能多地点亮数码管。要求实验中至少使用一个不带译码的数码管，充分利用实验平台上的接口和按键 (CLK,RST)。

本实验中，我使用 1 个不带译码的数码管显示自然数列 (0 F)，2 个带译码的数码管显示奇数列 (1, 3, 5, 7, 9) 和偶数列 (0, 2, 4, 6, 8)。并进行了 modelsim 仿真和 jielab 实验平台仿真.

4 代码解释及注释

4.1 4 位 2 进制译码器

基于模块化设计的思想，我首先编写了编码器 (decoder) 元件，将 4 位二进制数根据数码管的规则译码为 7 位二进制数，用于 7 段数码管的显示。之后在顶层实体中进行元件例化。代码如下:

decoder.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity decoder is
    port(
        bit_4_vec: in std_logic_vector(3 downto 0);
        bit_7_vec: out std_logic_vector(6 downto 0)
    );
end decoder;

architecture bhv of decoder is begin
    process(bit_4_vec) begin
        case bit_4_vec is --译码处理
            when "0000" => bit_7_vec <= "1111110";
            when "0001" => bit_7_vec <= "0110000";
            when "0010" => bit_7_vec <= "1101101";
            when "0011" => bit_7_vec <= "1111001";
            when "0100" => bit_7_vec <= "0110011";
            when "0101" => bit_7_vec <= "1011011";
            when "0110" => bit_7_vec <= "1011111";
```

```
            when "0111" => bit_7_vec <= "1110000";
            when "1000" => bit_7_vec <= "1111111";
            when "1001" => bit_7_vec <= "1110011";
            when "1010" => bit_7_vec <= "1110111";
            when "1011" => bit_7_vec <= "0011111";
            when "1100" => bit_7_vec <= "1001110";
            when "1101" => bit_7_vec <= "0111101";
            when "1110" => bit_7_vec <= "1001111";
            when "1111" => bit_7_vec <= "1000111";
            when others => bit_7_vec <= "0000000"; --其他情况全灭
        end case;
    end process;
end bhv;
```

4.2 三种数列的更新

在顶层实体文件中, 我设置了计数器 count, 每遇到一次时钟上升沿便 +1,count 每超过 1,000,000 时，重置 count 为 0，之后将奇数列、偶数列、自然数列对应的 std_logic_vector 加 1，实现数列的更新。这样处理，可以在 1MHz 的时钟周期下实现每 1s 更新一次数字。具体代码如下:

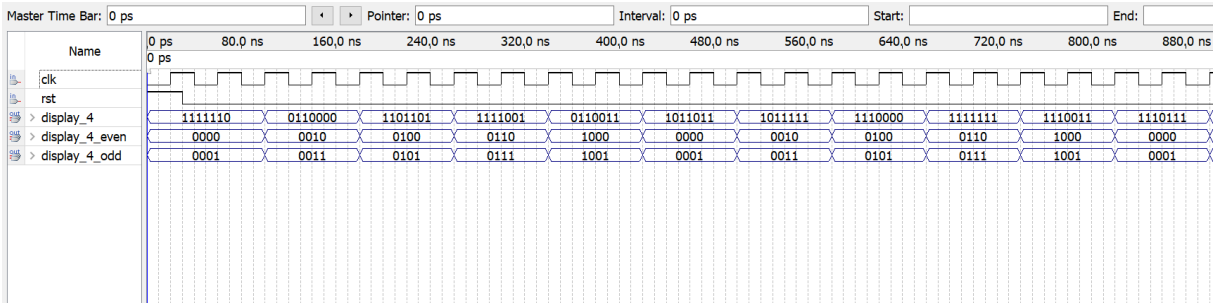
lighten2.vhd

```
--点亮数字人生, 使用1个不带译码,2个带译码的数码管--
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity lighten2 is
    port(
        clk: in std_logic; --控制时序变化
        rst: in std_logic; --重置
        display_4: out std_logic_vector(6 downto 0); --Without Decoder
        display_4_odd: out std_logic_vector(3 downto 0); --With Decoder
        display_4_even: out std_logic_vector(3 downto 0) --With Decoder
    );
end lighten2;

architecture bhv of lighten2 is
    component decoder --译码器例化
        port(
            bit_4_vec: in std_logic_vector(3 downto 0);
            bit_7_vec: out std_logic_vector(6 downto 0)
        );
    end component;
    signal display_4_odd_buf: std_logic_vector(3 downto 0) := "0001";--奇数
```

```
25      signal display_4_even_buf: std_logic_vector(3 downto 0) := "0000";--偶数
26      signal display_4_buf: std_logic_vector(3 downto 0) := "0000";--自然数
27      signal count: integer := 0; --计数器
28  begin
29      u1: decoder port map(bit_4_vec=>display_4_buf, bit_7_vec=>display_4);--
            译码
30      display_4_even <= display_4_even_buf;--带译码器的直接输出
31      display_4_odd <= display_4_odd_buf; --带译码器的直接输出
32      process(clk, rst) begin--时钟控制周期信号
33          if rst='1' then --异步复位
34              count <= 0;
35              display_4_buf <= "0000";--reset
36              display_4_even_buf <= "0000";
37              display_4_odd_buf <= "0001";
38          elsif clk'event and clk='1' then --处在上升沿看，也可以用rising_edge(
                clk)
39              if count < 1_000_000 then --每1000000次上升，也就是当1MHz(1000ns)
                    时钟频率时，1s更新1次(仿真的时候设为1，即一次上升沿更新一次)
40                  count <= count + 1;
41              else
42                  count <= 0;
43                  if display_4_buf = "1111" then
44                      display_4_buf <= "0000";--重置:F->0
45                  else
46                      display_4_buf <= display_4_buf + 1;--更新自然数
47                  end if;
48                  if display_4_even_buf = "1000" then
49                      display_4_even_buf <= "0000";--重置:8->0
50                  else
51                      display_4_even_buf <= display_4_even_buf + 2;--更新偶数
52                  end if;
53                  if display_4_odd_buf = "1001" then
54                      display_4_odd_buf <= "0001";--重置9->1
55                  else
56                      display_4_odd_buf <= display_4_odd_buf + 2;--更新奇数
57                  end if;
58              end if;
59          end if;
60      end process;
61  end bhv;
```

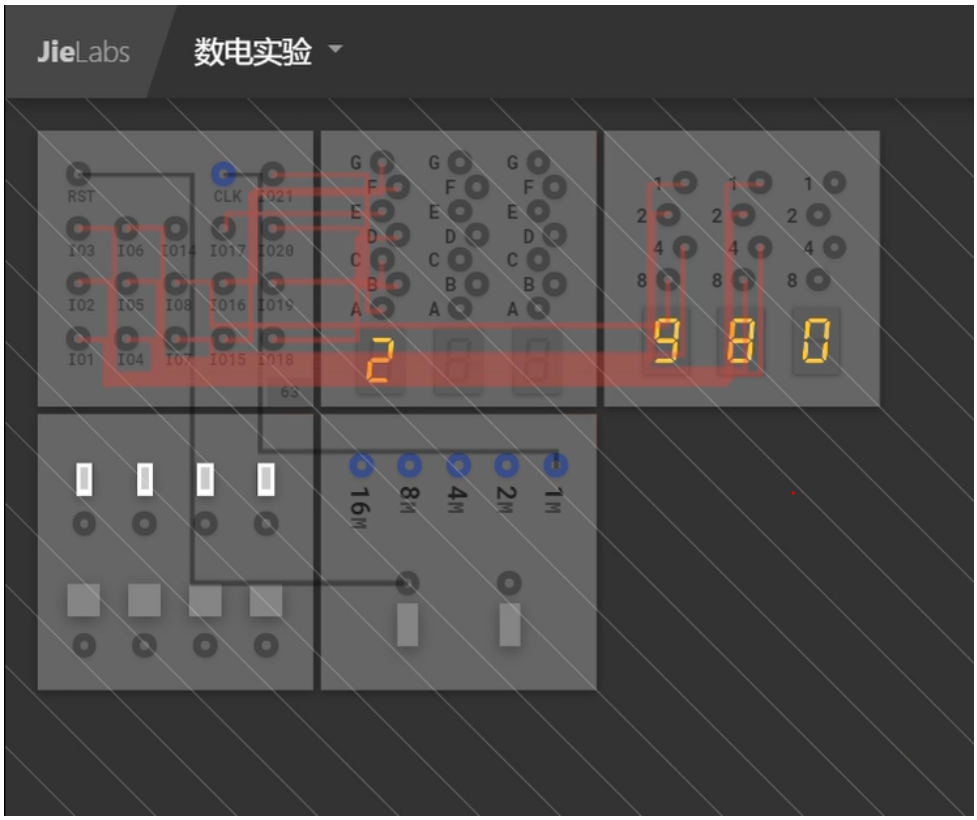
5 仿真结果

使用 Quartus 的 ModelSim 进行仿真 (根目录附"**Waveform.vwf**" 文件)，结果如下 (仿真时代码的计数器更新间隔做了调整，以保证展示更多的数字). 可以看到，仿真结果与预期一致，每隔一定时间，三个数列分别以自然数列、奇数列、偶数列更新。



6 JieLab 运行结果 (附录屏)

我将代码放在了 JieLab 实验平台上进行了验证，下面是运行时的截图，在根目录下有"**JieLabs 录屏.mp4**"，您可以更直观地检查我的实现效果，视频时长 50s，详细展示了三种数列 [自然数列 (左)、奇数列 (中)、偶数列 (右)] 的更新与复位 (rst) 操作.



7 实验总结

这是我第一次进行 VHDL 的编写，自己完成了 QuartusII 的安装、配置与 VHDL 的训练、仿真。我初步了解了 VHDL 的编写规范和语言特点，算是推开了" 数字人生" 的大门，也燃起了自己设计一些有趣的硬件的兴趣，。

在此感谢老师、助教悉心准备的安装指南，这些视频节省了我很多时间！我还要感谢 JieLab 的推出，得益于此实验平台，我可以将我的代码进行直观的验证，免去了只进行仿真的” 心虚” 之感，引脚的分配与连接更是让我身临其境，收获良多。感谢实验平台开发者与老师、助教的悉心指导！