点亮数字人生

计 74　傅舟涛 2017010682

1 实验目的

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，掌握 EDA 软件的使用方法和工作流程，进一步理解可编程芯片的工作原理。

2 实验内容及要求

使用至多两个带译码的数码管和至少一个不带译码的数码管，有规律地显示奇数列、偶数列、自然数列。

3 实验代码及简要原理分析

1） 实验代码内容

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

entity light is
    port(
        light_even_4: out std_logic_vector(3 downto 0);--偶数灯（带译码），对应PIN4-PIN1接口
        light_odd_4: out std_logic_vector(3 downto 0);--奇数灯（带译码），对应PIN8-PIN5接口
        light_natural_6: out std_logic_vector(6 downto 0);--自然数灯（不带译码），对应PIN20-PIN14接口，高位到低位为A到G
        test: out std_logic;--rst输出检测，对应PIN21接口，与电压表相连，rst起作用时，test为高电平，否则为低电平，用于指示
        clk, rst: in std_logic--clk对应PIN12接口，即CLK接口，负责计数器；rst对应PIN44接口，即RST接口，负责复原
    );
end light;

architecture lighting of light is
    signal signal_natural_4: std_logic_vector(3 downto 0):="0000";--自然数计数器
    signal signal_odd_4: std_logic_vector(3 downto 0):="0001";--奇数计数器
    signal signal_even_4: std_logic_vector(3 downto 0):="0000";--偶数计数器
begin
    process(clk, rst)--当按下clk或者rst变化时触发此process，此process只改变计数器，不改变对外输出
    begin
        if (clk'event and clk='1') then--若处于clk上升沿，则各计数器+1
            if (signal_natural_4="1001") then
                signal_natural_4<="0000";
            else
                signal_natural_4<=signal_natural_4+'1';
            end if;
            if (signal_even_4="1000") then
                signal_even_4<="0000";
            else
                signal_even_4<=signal_even_4+2;
            end if;
            if (signal_odd_4="1001") then
                signal_odd_4<="0001";
            else
                signal_odd_4<=signal_odd_4+2;
            end if;
        end if;
        if (rst='1') then--若被触发是rst处于"按下"的状态，则将各计数器复原
            signal_natural_4<="0000";
            signal_odd_4<="0001";
            signal_even_4<="0000";
            test<='1';
        else
            test<='0';
        end if;
    end process;

    process(signal_even_4)--负责改变偶数灯的输出
    begin
        light_even_4<=signal_even_4;
    end process;

    process(signal_odd_4)--负责改变奇数灯的输出
    begin
        light_odd_4<=signal_odd_4;
    end process;

```

```
59
60      process(signal_natural_4)--负责改变自然数灯的输出，0-15对应0-F，本实验中仅用到0-9
61      begin
62          case signal_natural_4 is
63              when "0000"=>light_natural_6<="1111110";
64              when "0001"=>light_natural_6<="0110000";
65              when "0010"=>light_natural_6<="1101101";
66              when "0011"=>light_natural_6<="1111001";
67              when "0100"=>light_natural_6<="0110011";
68              when "0101"=>light_natural_6<="1011011";
69              when "0110"=>light_natural_6<="0011111";
70              when "0111"=>light_natural_6<="1110000";
71              when "1000"=>light_natural_6<="1111111";
72              when "1001"=>light_natural_6<="1110011";
73              when "1010"=>light_natural_6<="1110111";
74              when "1011"=>light_natural_6<="0011111";
75              when "1100"=>light_natural_6<="1001110";
76              when "1101"=>light_natural_6<="0111101";
77              when "1110"=>light_natural_6<="1001111";
78              when "1111"=>light_natural_6<="1000111";
79              when others=>light_natural_6<="0000000";
80          end case;
81      end process;
82  end lighting;
```

2）原理及功能简要说明

输出接口包括 light_even_4、light_odd_4、light_natural_6、test、clk 和 rst。功能如代码中所示。

```
light_even_4: out std_logic_vector(3 downto 0);--偶数灯（带译码），对应PIN4-PIN1接口
light_odd_4: out std_logic_vector(3 downto 0);--奇数灯（带译码），对应PIN8-PIN5接口
light_natural_6: out std_logic_vector(6 downto 0);--自然数灯（不带译码），对应PIN20-PIN14接口，高位到低位为A到G
test: out std_logic;--rst输出检测，对应PIN21接口，与电压表相连，rst起作用时，test为高电平，否则为低电平，用于指示
clk, rst: in std_logic--clk对应PIN12接口，即CLK接口，负责计数器；rst对应PIN44接口，即RST接口，负责复原
```

Light 组与数码管相连，高位对应数码管的高位、低位对低位，即 PIN4 对应数码管的 8，PIN20 对应数码管的 A 等。

clk 作为计数器，每按下一次 clk，三个数码管各自显示下一数字；rst 作为复原器，按下后所有数码管回到初始值（偶数、奇数、自然数列分别回到 0、1、0）。在本程序中，约定当 rst 处于按下的状态时，无论 clk 的状态如何，数码管均显示初始值（原因见 3.3.1）。

test 作为 rst 的输出检测，即当 rst='1'时，test 输出'1'，rst='0'时，test 输出'0'。将 test 与电压表相连，使用时，如果按 rst 来复原，约定应当在 rst 弹起并且电压表示数归 0 后再按 clk，否则 clk 无法工作。

```
architecture lighting of light is
    signal signal_natural_4: std_logic_vector(3 downto 0):="0000";--自然数计数器
    signal signal_odd_4: std_logic_vector(3 downto 0):="0001";--奇数计数器
    signal signal_even_4: std_logic_vector(3 downto 0):="0000";--偶数计数器
```

3 个 signal 作为传递信号，用于记忆上一状态和方便转换。

```
process(signal_even_4)--负责改变偶数灯的输出
begin
    light_even_4<=signal_even_4;
end process;

process(signal_odd_4)--负责改变奇数灯的输出
begin
    light_odd_4<=signal_odd_4;
end process;
```

这两个 process 负责在计数器变化时改变奇偶数列的输出（分离计数器与数码管输出的 process 原因见 3.3.2）

```
process(clk, rst)--当按下clk或者rst变化时触发此process，此process只改变计数器，不改变对外输出
begin
    if (clk'event and clk='1') then--若处于clk上升沿，则各计数器+1
        if (signal_natural_4="1001") then
            signal_natural_4<="0000";
        else
            signal_natural_4<=signal_natural_4+'1';
        end if;
        if (signal_even_4="1000") then
            signal_even_4<="0000";
        else
            signal_even_4<=signal_even_4+2;
        end if;
        if (signal_odd_4="1001") then
            signal_odd_4<="0001";
        else
            signal_odd_4<=signal_odd_4+2;
        end if;
    end if;
    if (rst='1') then--若被触发是rst处于"按下"的状态，则将各计数器复原
        signal_natural_4<="0000";
        signal_odd_4<="0001";
        signal_even_4<="0000";
        test<='1';
    else
        test<='0';
    end if;
end process;
```

此 process 负责改变三个计数器（signal），当 clk 按下时，自然数列+1，奇数列和偶数列+2，并进行溢出判断和处理。如果此时 rst 处于高电平，则会将计数器复原。

```
process(signal_natural_4)--负责改变自然数灯的输出，0-15对应0-F，
begin
    case signal_natural_4 is
        when "0000"=>light_natural_6<="1111110";
        when "0001"=>light_natural_6<="0110000";
        when "0010"=>light_natural_6<="1101101";
        when "0011"=>light_natural_6<="1111001";
        when "0100"=>light_natural_6<="0110011";
        when "0101"=>light_natural_6<="1011011";
        when "0110"=>light_natural_6<="0011111";
        when "0111"=>light_natural_6<="1110000";
        when "1000"=>light_natural_6<="1111111";
        when "1001"=>light_natural_6<="1110011";
        when "1010"=>light_natural_6<="1110111";
        when "1011"=>light_natural_6<="0011111";
        when "1100"=>light_natural_6<="1001110";
        when "1101"=>light_natural_6<="0111101";
        when "1110"=>light_natural_6<="1001111";
        when "1111"=>light_natural_6<="1000111";
        when others=>light_natural_6<="0000000";
    end case;
end process;
```

这个 process 负责 8421 码向数码管码的转码。

3）process 的原理（重点）

3.3.1 计数器的设计

理论上实现异步控制是更好的，即 clk 与 rst 独立控制，采用两个 process，但由于不同 process 内不能均改变同一个 signal/port（见 4.1），因此只能将它们放在同一

个 process 中。由于同一个 process 内不能检测两个 event（见 4.2）、event 不能放在 process 中较后的位置（见 4.2），因此只能用 if(rst=1)来作为判断。signal 在进程结束时被赋值（见 4.3），因此在进程中多次赋值以最后一次（rst）为准，因此 rst 为 1 时会隔断 clk 的效果。

3.3.2 计数器改变与输出改变的分离

由于 signal 不能被即时赋值的特性（见 4.3），如果 clk 改变后计数器改变和输出改变在同一个 process 中的话，会使得输出变成计数器上一时刻的状态，造成的结果是开始时或按 rst 后必须按两次 cst 才能启动。举例如下（sgn 初始为 0，op 初始为 Z）：

sgn<=1;op<=sgn;

上面为一个 process 的两段代码，执行这两句后，结果为：sgn=1，op=0，即 op 变成了 sgn 在进程开始时的样子。为防止此情况出现，需将两者的改变放在不同的 process 中。

4 VHDL 语言探究、bug 的发现与修复（重点）

4.1 process 与赋值

经过测试，VHDL 中，不同 process 不能给同一信号/变量赋值。我认为这是因为，process 设计时是并行进行的，而同一时间（或在很近的时间内先后）给一个内容赋值可能发生危险，因此 VHDL 禁止了这样。

4.2 event 的检测

经过测试，VHDL 中，一个 process 里最多只能检测一个 event，且 event 的检测不能放在代码很后面。猜测前者是因为由于 event 时间很短，几乎不可能有两个 event 同时发生的情况，因此禁用了这种做法。我认为后者是因为，一个上升/下降沿时间很短，如果在 process 进行一段时间后检测 event 会检测不到，因此 VHDL 也禁止这种做法。

4.3 signal 和 variable 的赋值

首先指出，port（接口）也是一种 signal（信号）。在前两次课上，老师讲 signal 设计出来不能被立即复制，而 variable 可以，我当时不理解是什么意思。后来经过尝试知道：signal 使用<=赋值，有延迟，**在进程结束时被赋值**；variable 使用:=赋值，无延迟，在语句执行时立刻被赋值。因此如果执行在同一进程中 b<=a;c<=b;实际的结果等价于 c<=b;b<=a;，而非我想象中的 b<=a;c<=a;。再举一例说明此问题：

s1、s2、s3 分别为三个 integer 的信号，初值为 1、2、3，v1、v2、v3 分别为 3 个变量，初值为 1、2、3，在一个 process 中有如下代码：

s1<=s2;s2<=s3;s3<=s1;m1:=m2;m2:=m3;m3:=m1;

得到的结果为：s1=2,s2=3,s3=1,m1=2,m2=3,m3=2.在 verilog 中，这两种性质的赋值分别称为非阻塞赋值和阻塞赋值，VHDL 中似乎没有这种说法，不过两种赋值方式的性质与 verilog 中相同。

因此如果不将 signal 和 port 的改变分开，很容易出现开始时必须按两次 cst 数码管才变换的情况。

4.4 一些其他的发现

1）如果一个 if 太长，那么它就不能有 else，不知道为什么这么设计；

2）该语言完全不区分大小写。

5 关于环境配置、软件使用和硬件出现的问题与解决

1）破解的时候长时间未找到破解版的 sys_cpt.dll，最后依赖北邮人的资源解决；

2）选错了芯片型号，在群内同学的提醒下解决了；

3）不知道要分配引脚和如何分配引脚，在室友的提示下解决了；

4）不知道如何仿真，看书解决了；

5）为了使得积木整齐美观，我将线缕了一下使其变得整齐，并用一根绝缘线捆在一起，然后发现按 clk 时数码管经常会跳到复原状态（未按下 rst，且电压表无示数），解开线后此状况出现次数降低，将 rst 线与其他线分离后状况消失。猜测是许多线在靠的很近的情况下，由于互感现象，rst 内出现了脉冲信号使得数码管跳到复原状态，由于脉冲时间很短，电压表未检测出电压。这告诉我除了接线也是实验中的重要一部分，并且对于敏感信号而言，最好不要把线离得太近产生干扰。或许在未来的学习中，我学习了防抖措施的书写，给 rst 和 clk 增加防抖措施，应该可以排除这个干扰。

6 实验心得

在最开始时下载并破解软件、配置环境等方面耗费了大量时间，不过它们大多时候可以避免的：5.1 可以找同学拷贝一下 sys_cpt.dll 文件并问一下如何配置环境，5.2-5.4 在实验书上都有提到，非常详尽，只不过由于我过于心急而忽略了教程的重要性，应引以为戒。

VHDL 实验确实非常有趣，它的特性与以前学的 C++、python 等顺序性程序完全不同，不同 process 是并行触发的，赋值也是在进程结束时同时赋值，思维需要发生转化。

在操作和添加新功能的过程中，也发现了 VHDL 的一些有趣的特性（见 4），虽然当时遇到的时候感觉很奇怪，不过从硬件的角度思考之后，大部分特性还是可以理解的。不过仍然不太理解为什么一个 if 太长时就不允许有 else，可能我会在第八周交流答疑课上向老师询问。