点亮数字人生.实验报告

2018011324

计82

尤艺霖

1.实验目的&要求

1.1.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。

1.2.实验要求

设计一个数码管显示实验，要求七段数码管有规律地显示自然数列、奇数列、偶数列，要求实验中至少使用一个未经译码的数码管，通过CLK控制数列变化，并且有RST复位功能。

2.思路

由于想要使用JieLabs进行模拟，而JieLabs上的引脚有限，所以选择使用一个未经译码的数码管和两个经过译码的数码管，其中未经译码的数码管负责显示自然数序列。同时由于经过译码的数码管只能正常显示0到9的数码，所以需要对奇数序列和偶数序列进行特判，防止出现9以上的数码无法显示。为了在JieLabs上正常显示，还需要设置为每一百万个时钟周期改一次显示，所以需要添加计数器。

TestBench选择使用教程中的网站生成框架，测试信号为周期为10纳秒的时钟信号，为测试reset功能，在两段0.8秒之间插入一小段rst的高电平，同时注意避免rst的上升沿和clk上升沿重合导致无法区分。

3.实验代码

以下为主要代码 LightDigitalLife.vhd

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

entity LightDigitalLife is
    port(
        clk: in std_logic;                                  --Clock时钟输入
        rst: in std_logic;                                  --Reset输入
        display_natural: out std_logic_vector(6 downto 0); --自然数序列输出端口，使用未经译码的数
码管
        display_odd: out std_logic_vector(3 downto 0);     --奇数序列输出端口，使用经过译码的数码
管
        display_even: out std_logic_vector(3 downto 0)     --偶数序列输出端口，使用经过译码的数码
管
    );
```

```
end LightDigitalLife;

architecture Light of LightDigitalLife is
    signal cnt: integer:=0;                                          --计数器
    signal odd_series: std_logic_vector(3 downto 0):="0001";         --奇数序列
    signal even_series: std_logic_vector(3 downto 0):="0000";        --偶数序列
    signal natural_series: std_logic_vector(3 downto 0):="0000"; --自然数序列
begin
    process(clk,rst)
    begin
        if(rst='1') then                            --重置所有序列和显示
            natural_series<="0000";
            even_series<="0000";
            odd_series<="0001";
            display_even<=even_series;
            display_odd<=odd_series;
            elsif(rising_edge(clk)) then             --遇到clk的上升沿则处理计数器+1后的情况
            if(cnt<1000000) then                     --以10^6个clk的上升沿为周期改变显示的值
                cnt<=cnt+1;
            else
                natural_series<=natural_series+1;
                if(even_series="1000") then          --判断偶数序列是否到10
                    even_series<="0000";
                else
                    even_series<=even_series+2;
                end if;
                if(odd_series="1001") then           --判断奇数序列是否到11
                    odd_series<="0001";
                else
                    odd_series<=odd_series+2;
                end if;
                display_even<=even_series;
                display_odd<=odd_series;
                cnt<=0;
            end if;
        end if;
    end process;
    process(natural_series)
    begin
        case natural_series is                      --处理需要传达给未经译码的数码管的信号
            when "0000"=>display_natural<="1111110";
            when "0001"=>display_natural<="0110000";
            when "0010"=>display_natural<="1101101";
            when "0011"=>display_natural<="1111001";
            when "0100"=>display_natural<="0110011";
            when "0101"=>display_natural<="1011011";
            when "0110"=>display_natural<="1011111";
            when "0111"=>display_natural<="1110000";
            when "1000"=>display_natural<="1111111";
            when "1001"=>display_natural<="1110011";
            when "1010"=>display_natural<="1110111";
            when "1011"=>display_natural<="0011111";
            when "1100"=>display_natural<="1001110";
```

```
            when "1101"=>display_natural<="0111101";
            when "1110"=>display_natural<="1001111";
            when "1111"=>display_natural<="1000111";
            when others=>display_natural<="0000000";
        end case;
    end process;
end Light;
```

以下为TestBench代码 LightDigitalLife_tb.vhd

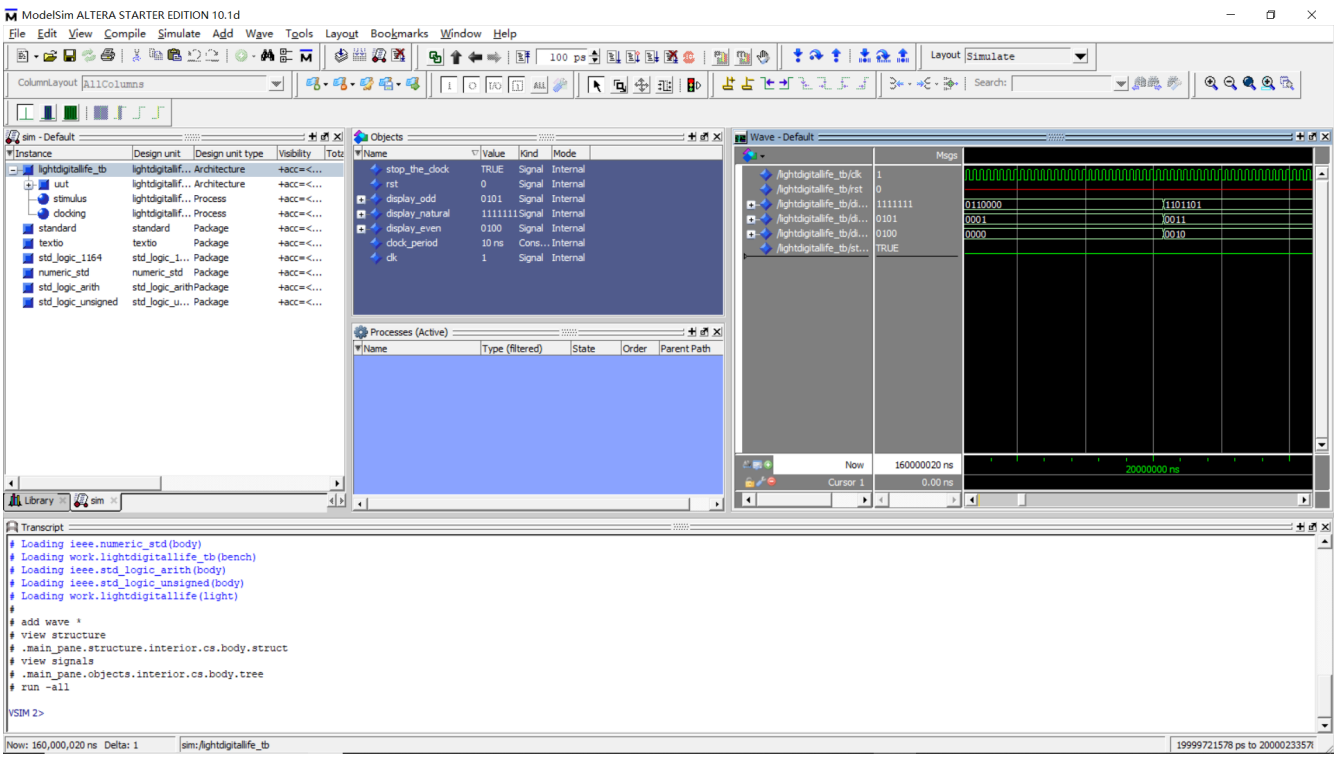
```
-- Testbench created online at:
--   www.doulos.com/knowhow/perl/testbench_creation/
-- Copyright Doulos Ltd
-- SD, 03 November 2002

library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Std.all;

entity LightDigitalLife_tb is
end;

architecture bench of LightDigitalLife_tb is

  component LightDigitalLife
    port(
        clk: in std_logic;
        rst: in std_logic;
        display_natural: out std_logic_vector(6 downto 0);
        display_odd: out std_logic_vector(3 downto 0);
        display_even: out std_logic_vector(3 downto 0)
    );
  end component;

  signal clk: std_logic;
  signal rst: std_logic;
  signal display_natural: std_logic_vector(6 downto 0);
  signal display_odd: std_logic_vector(3 downto 0);
  signal display_even: std_logic_vector(3 downto 0) ;

  constant clock_period: time := 10 ns;--时钟周期为10ns
  signal stop_the_clock: boolean;

begin

  uut: LightDigitalLife port map ( clk             => clk,
                                   rst             => rst,
                                   display_natural => display_natural,
                                   display_odd     => display_odd,
                                   display_even    => display_even );

  stimulus: process
  begin
```

```
    -- Put initialisation code here

    wait for 80000000ns;
    wait for 3ns;          --测试Reset功能，为了在仿真工具中区别出来，将rst和clk的上升沿分开
    rst<='1';
    wait for 4ns;
    rst<='0';
    wait for 3ns;
    wait for 80000000ns;
    -- Put test bench stimulus code here

    stop_the_clock <= true;
    wait;
  end process;

  clocking: process
  begin
    while not stop_the_clock loop
      clk <= '0', '1' after clock_period / 2;
      wait for clock_period;
    end loop;
    wait;
  end process;

end;
```

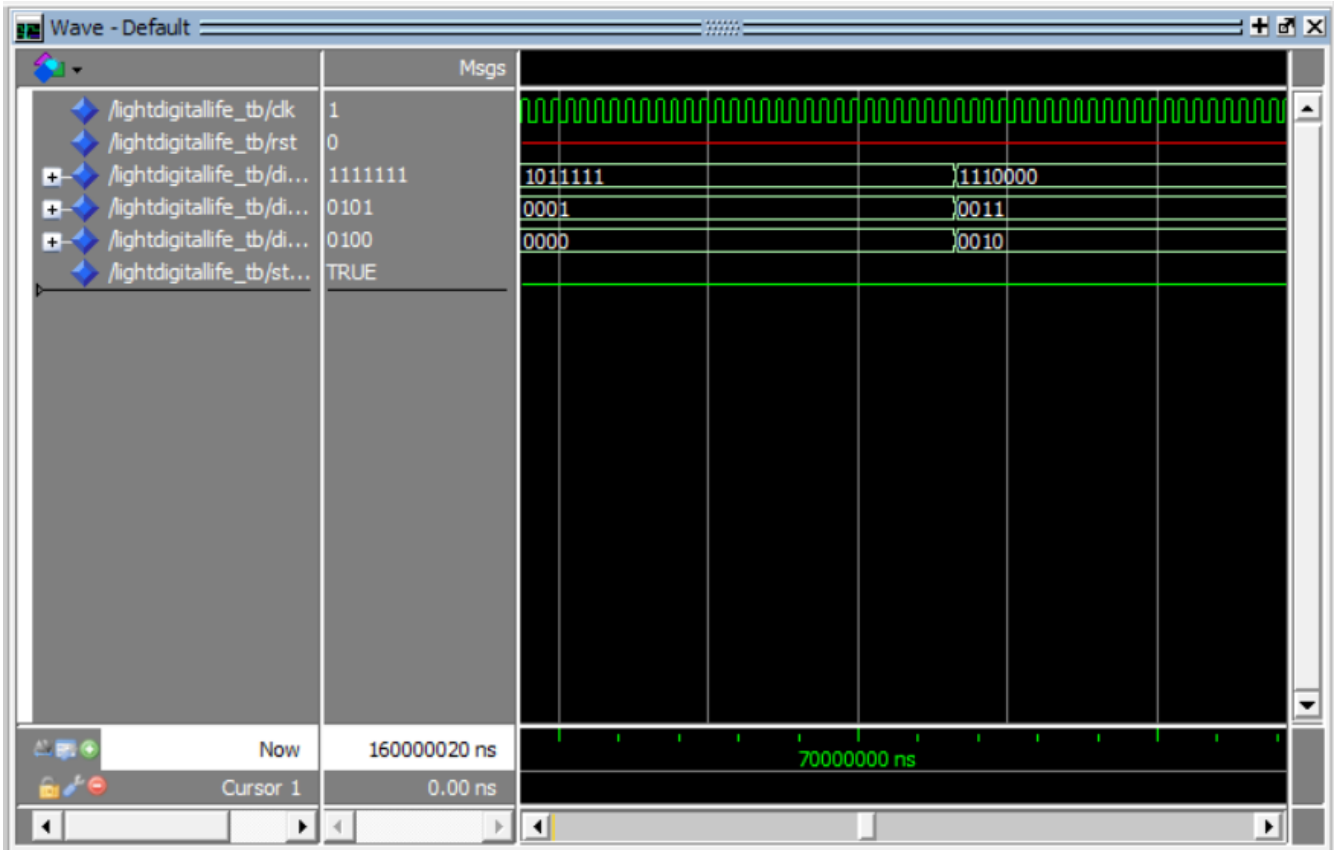
4.实验结果

4.1.Multisim仿真结果

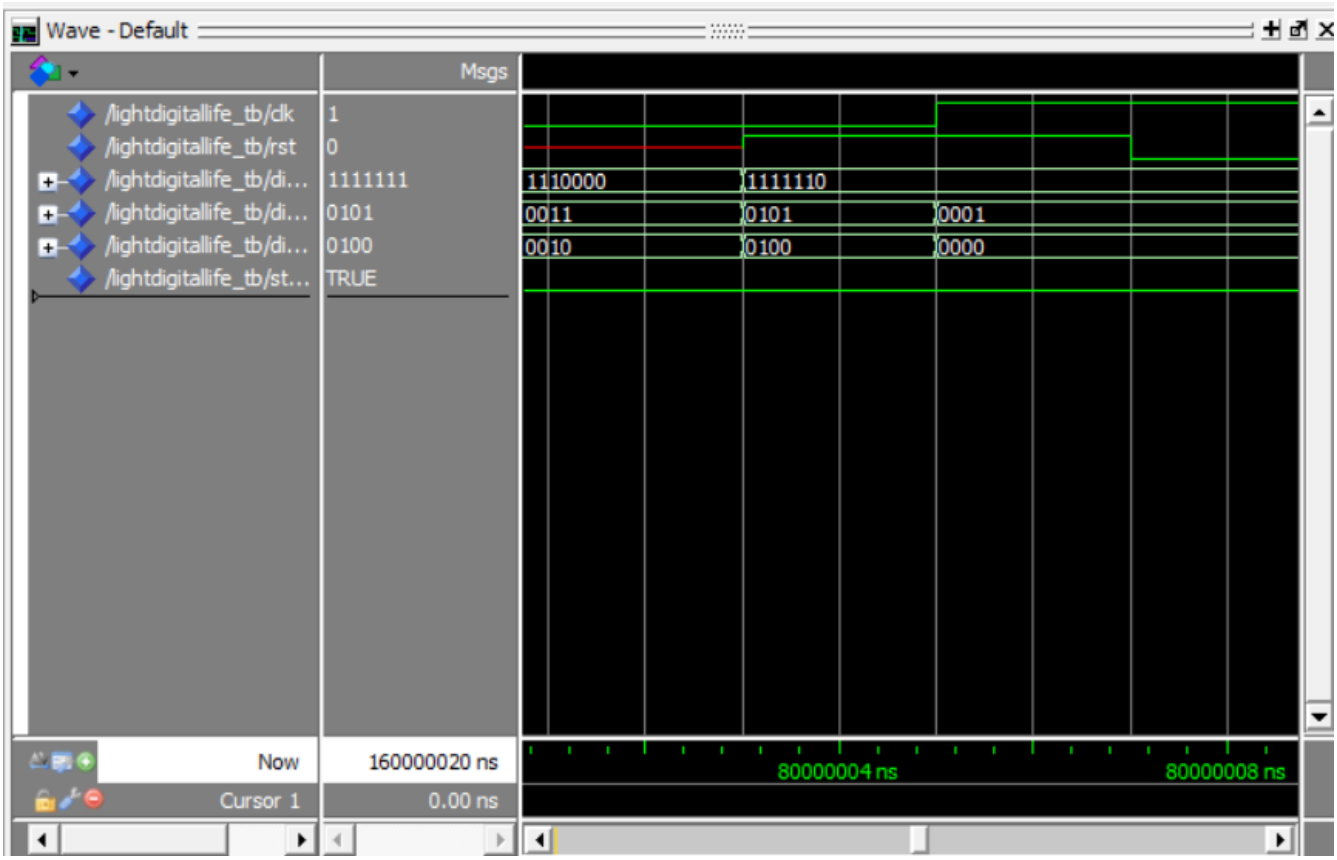
运行Multisim后可以正确产生波形：



另一个输出信号发生改变的例子：

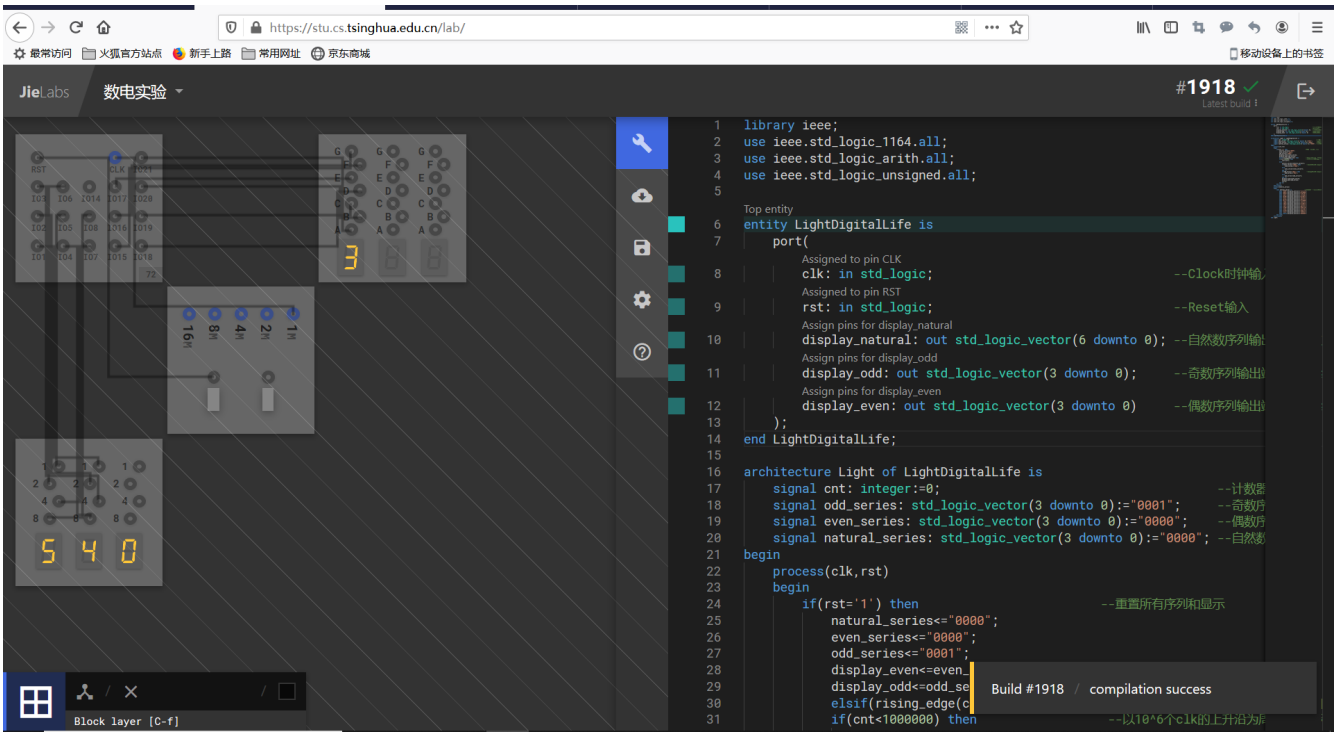


Reset功能的测试，结果正确：



4.2.JieLabs模拟结果

在JieLabs上模拟时设置引脚如下：rst对应RST，clk对应CLK，自然数序列对应IO15到IO21引脚，奇数序列和偶数序列分别对应IO1到IO4和IO5到IO8引脚，IO14引脚不使用。CLK接在1M的时钟信号上，RST接到一个开关上，其他引脚接到对应的数码管上。最终模拟结果是可以正常显示三个序列，Reset也可以正常将三个序列回到起始。



5.遇到的问题&总结

在实验过程中遇到的问题主要有以下几点：一些基础的语法错误，会把一些C++的语法带进来；在同一个process里对两个敏感信号的处理常常导致编译错误，这是对硬件程序理解不够导致的；使用教程的网站初始化TestBench时无法识别Reset，需要我手动设置Reset的测试。

这次实验是我第一次接触到硬件程序，相比于之前的C++、Python等语言有很大区别，特别是其中process看似和其他语言的函数类似，其实有很多限制，还需要更深入的理解。最后也算是成功完成了这一实验，对VHDL有了初步认识。