2016 年《综合课程设计 2-1》课程实验报告（第三类实验）

实验题四：设计一个能显示时、分、秒的实时时钟

姓名：王科

学号：1310583

专业：计算机科学与技术

完成日期：2016/4/10

目录

一、 实验内容

本次实验是使用图形设计方法与 VHDL 编程相结合在 FPGA 芯片内构建独立逻辑实验，要求设计一个能显示时、分、秒的实时时钟，通过 FPGA 板上的数码管显示。初始时间为 0 时 0 分 0 秒。

二、 实验原理

1.设计概述

按照实验内容要求，设计一个实时显示时间的时钟，其初始时间为 0 时 0 分 0 秒，设计一个 reset 和 stop 按钮，当按下 reset 时，时间自动清零，当按下 stop 时，时钟暂停跳动，再次按下时恢复跳动，其输入输出特性如下图：

1.1 输入输出特性表

INPUT			OUTPUT					
clk	reset	stop	hourH	hourL	minH	minL	secH	secL
↑	1	x	0	0	0	0	0	0
↑	0	1	不变	不变	不变	不变	不变	不变
↑	0	0	计时	计时	计时	计时	计时	计时

1.2 设计外部端口

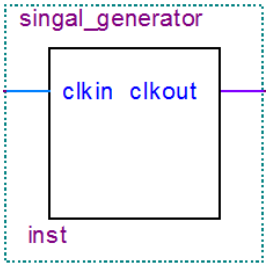
按照设计要求，即需要 3 个输入端口和 24 个输出端口，其中输入端口包括时钟信号（clk）、重置信号（reset）、暂停/恢复信号（stop），输出信号包括显示小时高位和低位、分钟高位和低位、秒数高位和低位，每个位数需要 4 位共 24 个端口。

2.设计内部实现逻辑

即按照自底向上的设计方法，划分为三个模块，第一个模块是信号产生器，即产生 1 秒的脉冲信号，第二个模块是接受 1 秒的 clk 信号进行计数并输出正确的小时、分钟、秒数的值，第三个模块是顶层文件，将信号输出至 6 个七段数码管上。

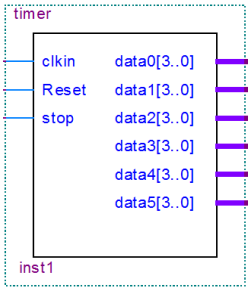
2.1 信号产生器“singal_generator.vhd”

此部分有一个输入端是 184.32KHZ 的时钟信号 clk，在内部进行分频，输出一个周期是 1 秒的脉冲信号 clkout。其整体的外部模块图如下：



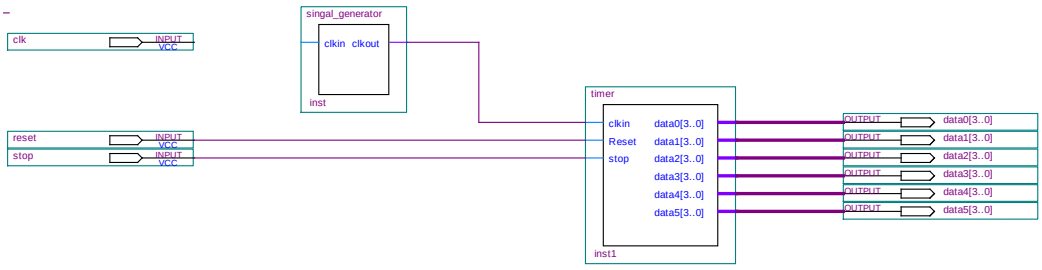
2.2 显示正确的小时、分钟、秒数的值“timer.vhd”

此部分的输入端是一个来自上一模块产生的时钟信号 clk，还有两个分别是重置信号（reset）和暂停/恢复信号（stop）。输出是要显示的小时、分钟、秒数的值，其整体的外部模块图如下：



2.3 顶层文件“clock.bdf”

此部分是顶层文件，将上面的两个模块连接起来，形成整体的时钟电路。其逻辑电路图如下：



3. 设计描述

使用自底向上设计方法，先使用 VHDL 语言设计一个产生周期是 1 秒的脉冲发生器，为其生成元部件；再使用 VHDL 语言设计一个根据时钟信号和其他控制信号输出正确的小时、分钟和秒的读书值，为其生成元部件；然后设计的主体框架是一个使用 quartus 图形化设计方法设计的一个时钟电路，并为其设置引脚并编译生成工程文件。

三、 实验步骤

1. 设计信号产生器“singal_generator.vhd”的端口、内部 VHDL 逻辑程序

按照实验要求设计一个产生周期是 1s 的脉冲信号的元部件，vhdl 语言设计端口如下：

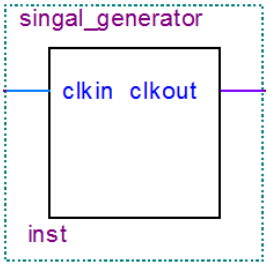
```
PORT(
      clkin: IN STD_LOGIC;          -- 184.32KHz
      clkout: BUFFER STD_LOGIC              --1s
  ):
```

在其内部实现逻辑功能代码如下：

```
ARCHITECTURE content OF singal_generator IS
      SIGNAL counter: integer range 0 to 92160;
      SIGNAL Clk:Std_Logic;
BEGIN
      PROCESS(clkin)
        BEGIN
            IF( counter=184320) THEN
                  counter<= 0;
            ELSIF rising_edge(clkin) THEN
                  counter <= counter+1;
            END IF;
      END PROCESS;
      PROCESS(counter)
      BEGIN
            IF( counter < 10 and counter > 8) then
              clkout <= '0';
            else
              clkout<='1';
```

```
        end if;
    END PROCESS;
END content;
```

编译完成后为其生成.bdf 元器件如下：



2.设计显示小时、分钟、秒数的值“timer.vhd”的端口、内部 VHDL 逻辑程序

按照实验要求设计一个产生正确的小时、分钟、秒数的值的元部件，vhdl 语言设计端口如下：

```
PORT(
    clkin : IN STD_LOGIC;
    Reset : IN STD_LOGIC;
    stop  : IN STD_LOGIC;
    data0 : OUT STD_LOGIC_VECTOR(3 downto 0);
    data1 : OUT STD_LOGIC_VECTOR(3 downto 0);
    data2 : OUT STD_LOGIC_VECTOR(3 downto 0);
    data3 : OUT STD_LOGIC_VECTOR(3 downto 0);
    data4 : OUT STD_LOGIC_VECTOR(3 downto 0);
    data5 : OUT STD_LOGIC_VECTOR(3 downto 0)
);
```

其根据 clk 信号进行计数，并更新小时、分钟、秒数的值的进程代码如下：

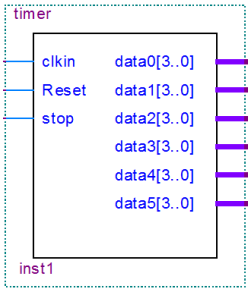
```
timer:process(clkin,Reset,stop)
BEGIN
    if(Reset = '1') then hour <= 0;min <= 0; sec <= 0;
    elsif (stop = '0') then
    if(clkin'event and clkin='1') then
        sec<=sec+1;
        if (sec>=59)then
        sec<=0;min<=min+1;
        if (min>=59)then
```

```
                min<=0;hour<=hour+1;
                    if (hour>=23)then
                        hour<=0;
                    end if;
                end if;
            end if;
           end if;
          end if;
        end if;
end process;
```

随后是 6 个分别显示小时高低位、分钟高低位、秒数高低位的进程，这里以显示秒数低位代码如下（产生七段数码管的对应值）：

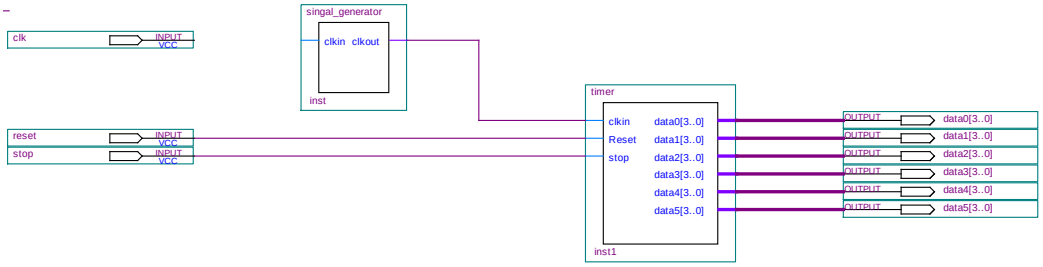
```
secH<=sec / 10;
secL<=sec-secH*10;
minH<=min / 10;
minL<=min-minH*10;
hourH<=hour / 10;
hourL<=hour-hourH*10;

sec_low:process(secL)
BEGIN
     case secL is
       when 0 =>data0<="0000";
       when 1 =>data0<="0001";
       when 2 =>data0<="0010";
       when 3 =>data0<="0011";
       when 4 =>data0<="0100";
       when 5 =>data0<="0101";
       when 6 =>data0<="0110";
       when 7 =>data0<="0111";
       when 8 =>data0<="1000";
       when 9 =>data0<="1001";
       when others =>data0<="1111";
   end case;
end process;
```

编译完成后为其生成.bdf 元器件如下：



3. 使用图形化方法设计原理图

使用 quartus 包含上述设计的元件，设计原理图如下：



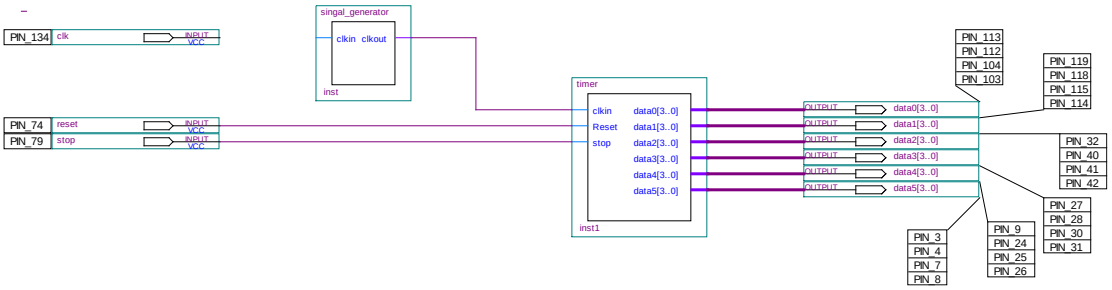
4. 观察 FPGA 电路板，将输入输出端口进行引脚绑定

根据 FPGA 电路板上空闲的剩余引脚，在这次实验中个输入输出端对应的引脚号如下表:

	Node Name	Direction	Location	I/O Bank	Vref Group	I/O Standard	Reserved	Group	Current Strength
1	clk	Input	PIN_134	2	B2_N1	3.3-V LVTTL (default)			24mA (default)
2	data0[3]	Output	PIN_113	2	B2_N0	3.3-V LVTTL (default)		data0[3..0]	24mA (default)
3	data0[2]	Output	PIN_112	2	B2_N0	3.3-V LVTTL (default)		data0[3..0]	24mA (default)
4	data0[1]	Output	PIN_104	3	B3_N0	3.3-V LVTTL (default)		data0[3..0]	24mA (default)
5	data0[0]	Output	PIN_103	3	B3_N0	3.3-V LVTTL (default)		data0[3..0]	24mA (default)
6	data1[3]	Output	PIN_119	2	B2_N0	3.3-V LVTTL (default)		data1[3..0]	24mA (default)
7	data1[2]	Output	PIN_118	2	B2_N0	3.3-V LVTTL (default)		data1[3..0]	24mA (default)
8	data1[1]	Output	PIN_115	2	B2_N0	3.3-V LVTTL (default)		data1[3..0]	24mA (default)
9	data1[0]	Output	PIN_114	2	B2_N0	3.3-V LVTTL (default)		data1[3..0]	24mA (default)
10	data2[3]	Output	PIN_32	1	B1_N1	3.3-V LVTTL (default)		data2[3..0]	24mA (default)
11	data2[2]	Output	PIN_40	4	B4_N1	3.3-V LVTTL (default)		data2[3..0]	24mA (default)
12	data2[1]	Output	PIN_41	4	B4_N1	3.3-V LVTTL (default)		data2[3..0]	24mA (default)
13	data2[0]	Output	PIN_42	4	B4_N1	3.3-V LVTTL (default)		data2[3..0]	24mA (default)
14	data3[3]	Output	PIN_27	1	B1_N1	3.3-V LVTTL (default)		data3[3..0]	24mA (default)
15	data3[2]	Output	PIN_28	1	B1_N1	3.3-V LVTTL (default)		data3[3..0]	24mA (default)
16	data3[1]	Output	PIN_30	1	B1_N1	3.3-V LVTTL (default)		data3[3..0]	24mA (default)
17	data3[0]	Output	PIN_31	1	B1_N1	3.3-V LVTTL (default)		data3[3..0]	24mA (default)
18	data4[3]	Output	PIN_9	1	B1_N0	3.3-V LVTTL (default)		data4[3..0]	24mA (default)
19	data4[2]	Output	PIN_24	1	B1_N1	3.3-V LVTTL (default)		data4[3..0]	24mA (default)
20	data4[1]	Output	PIN_25	1	B1_N1	3.3-V LVTTL (default)		data4[3..0]	24mA (default)
21	data4[0]	Output	PIN_26	1	B1_N1	3.3-V LVTTL (default)		data4[3..0]	24mA (default)
22	data5[3]	Output	PIN_3	1	B1_N0	3.3-V LVTTL (default)		data5[3..0]	24mA (default)
23	data5[2]	Output	PIN_4	1	B1_N0	3.3-V LVTTL (default)		data5[3..0]	24mA (default)
24	data5[1]	Output	PIN_7	1	B1_N0	3.3-V LVTTL (default)		data5[3..0]	24mA (default)
25	data5[0]	Output	PIN_8	1	B1_N0	3.3-V LVTTL (default)		data5[3..0]	24mA (default)
26	reset	Input	PIN_74	3	B3_N1	3.3-V LVTTL (default)			24mA (default)
27	stop	Input	PIN_79	3	B3_N1	3.3-V LVTTL (default)			24mA (default)

编译下载到 FPGA 电路板上，然后连接实验箱的电路。

编译后如下：



5. 电路连接示意图

连接 FPGA 电路板上电路：

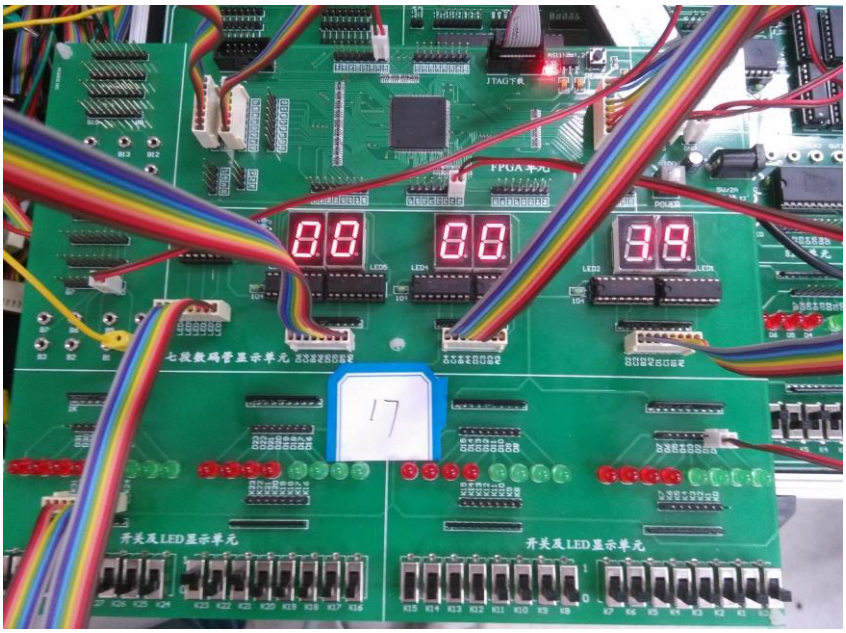
- 将 FPGA 板上的 PIN_67~PIN_96(Y15~Y0)接到 FPGA 电路板的 LED 灯泡上；
- 将 FPGA 板上的 PIN_97、PIN_99、PIN_100、PIN_101(A[3]、A[2]、A[1]、A[0])接到 FPGA 电路板的控制开关上；
- 将 FPGA 板上的 PIN_91（EN）接到 FPGA 电路板的控制开关上；

对应端口连接表			
说明	引脚	说明	引脚
FPGA 板的时钟输入信号（clk）	PIN_134	实验箱的时钟产生信号	184.32KHz
FPGA 板的重置信号（reset）	PIN_74	FPGA 电路板控制开关	K0
FPGA 板的暂停/恢复信号(stop)	PIN_79	FPGA 电路板控制开关	K1
FPGA 板的小时信号高位 data5	PIN_3~PIN_8	FPGA 板上的数码显示单元	A6、B6、C6、D6
FPGA 板的小时信号低位 data4	PIN_9~PIN_26	FPGA 板上的数码显示单元	A5、B5、C5、D5
FPGA 板的分钟信号高位 data3	PIN_27~PIN_31	FPGA 板上的数码显示单元	A4、B4、C4、D4
FPGA 板的分钟信号低位 data2	PIN_32~PIN_42	FPGA 板上的数码显示单元	A3、B3、C3、D3
FPGA 板的秒数信	PIN_119~PIN_114	FPGA 板上的数码	A2、B2、C2、D2

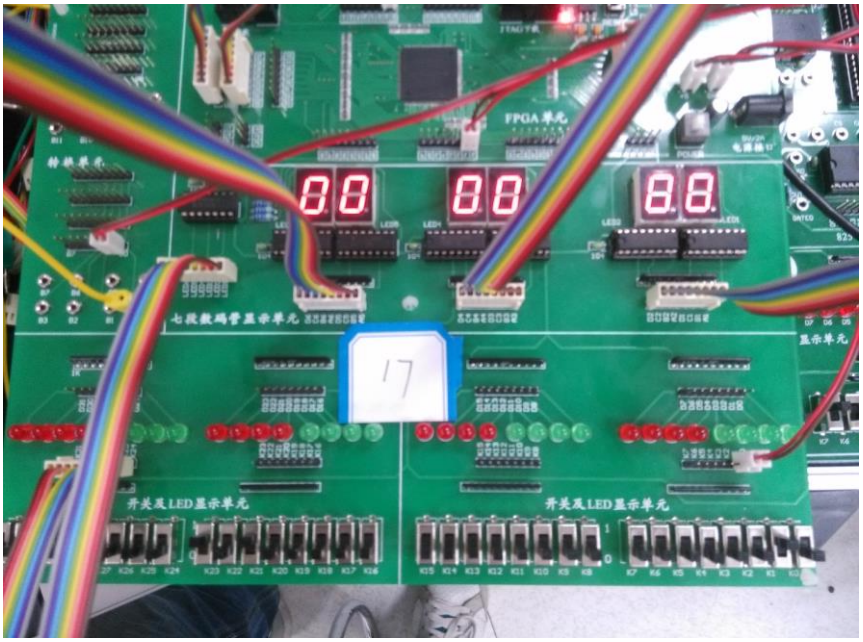
号高位 data1		显示单元	
FPGA 板的秒数信号低位 data0	PIN_113~PIN_103	FPGA 板上的数码显示单元	A1、B1、C1、D1

6. 实验结果

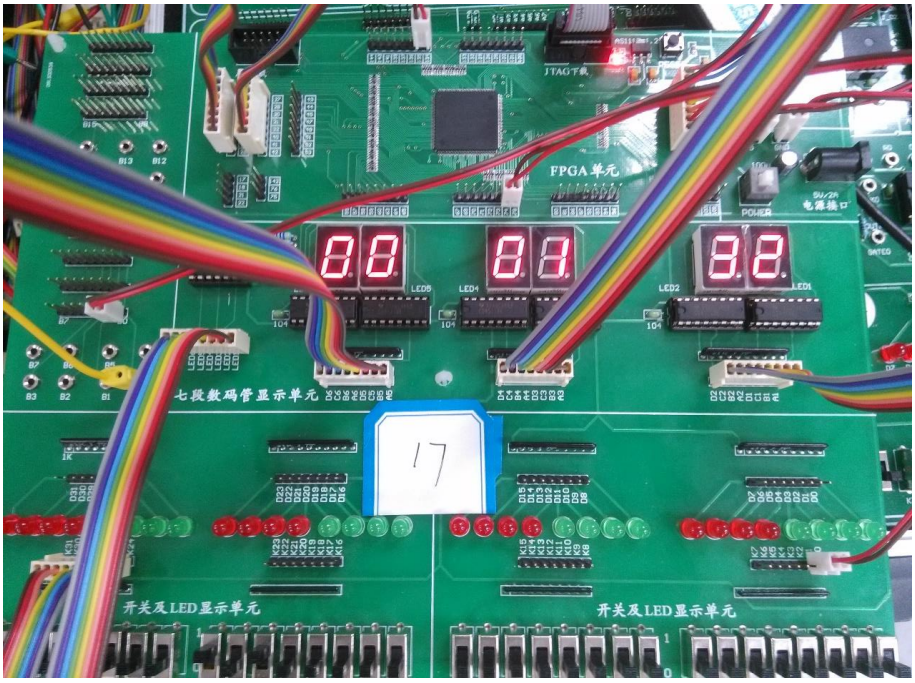
实验时，一开始从 00：00：00 开始记时，实验结果如下：



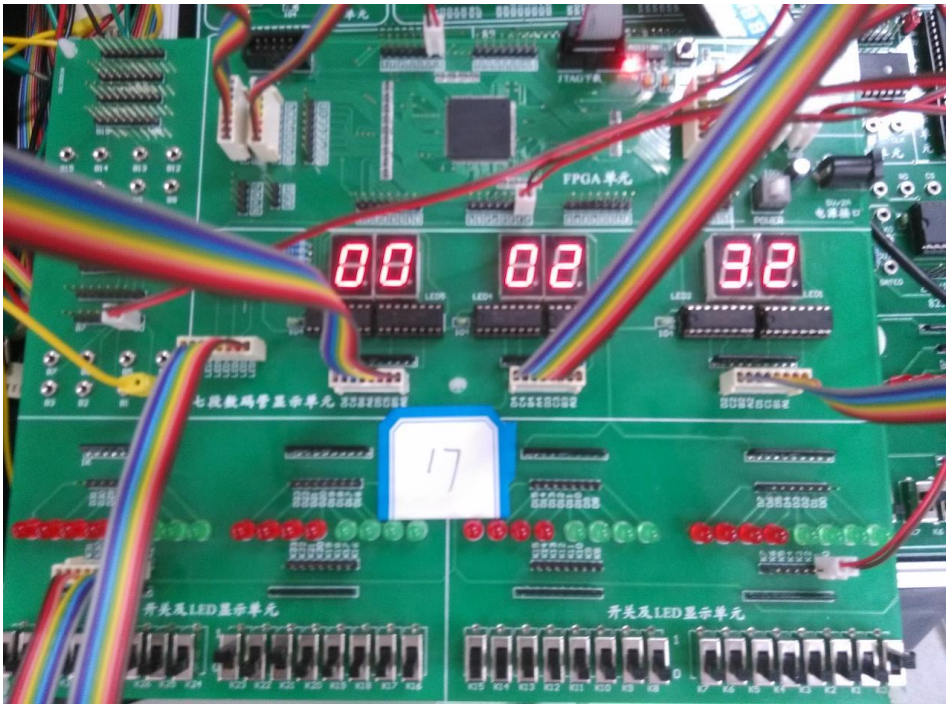
当拨动开关，使得 reset 信号有效时，时间归零如下：



再次拨动开关，使得 reset 信号无效时，时间从头开始恢复记时如下：

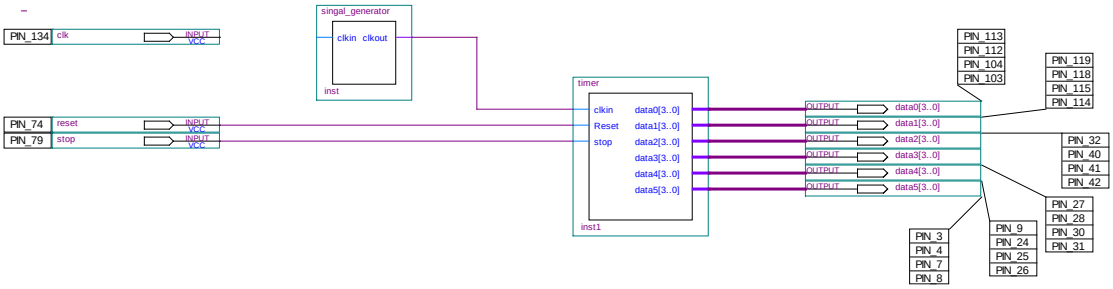


拨动开关，使得 stop 信号有效时，时间暂停，但并不清零，实验结果如下：



四、 实验原理图和 vhdl 程序

1. 整体原理图（“clock.bdf”）



2. 元部件 1（“singal_generator.vhd”）

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;

ENTITY    singal_generator IS
  PORT(
       clkin: IN STD_LOGIC;          -- 184.32KHz
       clkout: BUFFER STD_LOGIC              --1s
  );
END singal_generator;

  ARCHITECTURE content OF singal_generator IS

       SIGNAL counter: integer range 0 to 92160;
       SIGNAL Clk:Std_Logic;

BEGIN

       PROCESS(clkin)
         BEGIN
              IF( counter=184320) THEN
                    counter<= 0;
              ELSIF rising_edge(clkin) THEN
                    counter <= counter+1;
              END IF;
       END PROCESS;

       PROCESS(counter)
       BEGIN
              IF( counter < 10 and counter > 8) then
```

```
            clkout <= '0';
          else
            clkout<='1';
          end if;
    END PROCESS;

END content;
```

3.元部件 2（“timer.vhd”）

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Use ieee.std_logic_arith.all;

ENTITY timer IS
  PORT(
      clkin : IN STD_LOGIC;
      Reset : IN STD_LOGIC;
      stop    : IN STD_LOGIC;
      data0 : OUT STD_LOGIC_VECTOR(3 downto 0);
      data1 : OUT STD_LOGIC_VECTOR(3 downto 0);
      data2 : OUT STD_LOGIC_VECTOR(3 downto 0);
      data3 : OUT STD_LOGIC_VECTOR(3 downto 0);
      data4 : OUT STD_LOGIC_VECTOR(3 downto 0);
      data5 : OUT STD_LOGIC_VECTOR(3 downto 0)
  );
END timer;



ARCHITECTURE timer_architecture OF timer IS

signal sec: integer range 0 to 60 ;
signal secL: integer range 0 to 9;
signal secH: integer range 0 to 9;

signal min: integer range 0 to 60 ;
signal minL: integer range 0 to 9;
signal minH: integer range 0 to 9;

signal hour: integer range 0 to 24 ;
```

```
signal hourL: integer range 0 to 9;
signal hourH: integer range 0 to 9;

BEGIN

timer:process(clkin,Reset,stop)
BEGIN
      if(Reset = '1') then hour <= 0;min <= 0; sec <= 0;
      elsif (stop = '0') then
      if(clkin'event and clkin='1') then
         sec<=sec+1;
         if (sec>=59)then
         sec<=0;min<=min+1;
         if (min>=59)then
              min<=0;hour<=hour+1;
                  if (hour>=23)then
                       hour<=0;
                  end if;
           end if;
           end if;
         end if;
       end if;
end process;


secH<=sec / 10;
secL<=sec-secH*10;
minH<=min / 10;
minL<=min-minH*10;
hourH<=hour / 10;
hourL<=hour-hourH*10;


sec_low:process(secL)
BEGIN
     case secL is
      when 0 =>data0<="0000";
      when 1 =>data0<="0001";
      when 2 =>data0<="0010";
      when 3 =>data0<="0011";
      when 4 =>data0<="0100";
      when 5 =>data0<="0101";
      when 6 =>data0<="0110";
      when 7 =>data0<="0111";
```

```
        when 8 =>data0<="1000";
        when 9 =>data0<="1001";
        when others =>data0<="1111";
    end case;
end process;

sec_high:process(secH)
BEGIN
      case secH is
        when 0 =>data1<="0000";
        when 1 =>data1<="0001";
        when 2 =>data1<="0010";
        when 3 =>data1<="0011";
        when 4 =>data1<="0100";
        when 5 =>data1<="0101";
        when 6 =>data1<="0110";
        when 7 =>data1<="0111";
        when 8 =>data1<="1000";
        when 9 =>data1<="1001";
        when others =>data1<="1111";
    end case;
end process;

min_low:process(minL)
BEGIN
      case minL is
        when 0 =>data2<="0000";
        when 1 =>data2<="0001";
        when 2 =>data2<="0010";
        when 3 =>data2<="0011";
        when 4 =>data2<="0100";
        when 5 =>data2<="0101";
        when 6 =>data2<="0110";
        when 7 =>data2<="0111";
        when 8 =>data2<="1000";
        when 9 =>data2<="1001";
        when others =>data2<="1111";
    end case;
end process;

min_high:process(minH)
BEGIN
      case minH is
        when 0 =>data3<="0000";
```

```
        when 1 =>data3<="0001";
        when 2 =>data3<="0010";
        when 3 =>data3<="0011";
        when 4 =>data3<="0100";
        when 5 =>data3<="0101";
        when 6 =>data3<="0110";
        when 7 =>data3<="0111";
        when 8 =>data3<="1000";
        when 9 =>data3<="1001";
        when others =>data3<="1111";
    end case;
end process;

hour_low:process(hourL)
BEGIN
      case hourL is
        when 0 =>data4<="0000";
        when 1 =>data4<="0001";
        when 2 =>data4<="0010";
        when 3 =>data4<="0011";
        when 4 =>data4<="0100";
        when 5 =>data4<="0101";
        when 6 =>data4<="0110";
        when 7 =>data4<="0111";
        when 8 =>data4<="1000";
        when 9 =>data4<="1001";
        when others =>data4<="1111";
    end case;
end process;

hour_high:process(hourH)
BEGIN
      case hourH is
        when 0 =>data5<="0000";
        when 1 =>data5<="0001";
        when 2 =>data5<="0010";
        when 3 =>data5<="0011";
        when 4 =>data5<="0100";
        when 5 =>data5<="0101";
        when 6 =>data5<="0110";
        when 7 =>data5<="0111";
        when 8 =>data5<="1000";
        when 9 =>data5<="1001";
        when others =>data5<="1111";
```

```
    end case;
end process;

END timer_architecture;
```

五、 实验总结

1.实验遇到的问题

这次的显示时、分、秒的实时时钟设计实验比较复杂，我花费了两节课的时间才把它做完，从一开始的设计分频器，实现输出 1 秒的脉冲信号，到最终的调试成功，数值准确无误的跳动着。这其中的乐趣实在是太大了。我在这其中也遇到了很多的问题，最后都顺利的解决了。现将主要问题记录如下：

- 在实现分频器输出一秒的脉冲信号时，我一开始是直接套用之前的程序，即在一秒时间内，一半时间是高电平，一半时间是低电平。但是后来与 timer 元件连接之后发现再一秒内有两个上升沿（通过在一秒内秒数数值增加 2 发现的），我将之前的 vhdl 程序改写成了在一秒内，只有极短的时间内是高电平，其余仍是低电平，而不是之前的一半时间是高电平，一半时间是低电平；我发现这样就没有问题了，我猜测可能是我们实验箱的时钟信号不是很稳定，虽然电平不变，但仍然有微小的跳变导致了 vhdl 程序检测到了多个上升沿的存在。
- 第二个算是自己比较粗心的问题了，我在一开始将程序下载到 FPGA 电路板上之后，迫不及待的就连好了电路，我以为只要将七段数码显示单元的数据线连好就行了，却发现数码管什么数字也不显示，我开始还以为自己的程序出问题了，找了半天的错。最后老师告诉我需要将数码显示单元的使能端 LED1~LED6 接上低电平。这也反映了我当时没有冷静的观察，心里有点浮躁，粗心大意。

2.实验感悟

这次的使用硬件描述语言 VHDL 设计电路实验是最后一个老师布置的设计实验，对我而言也是有较大难度的，我通过在发现问题、分析问题、查阅资料、解决问题的过程中，一步一步的完成了这次的实验，熟悉了自底向上的方法创建一个工程，明白了做实验要小心细致，仔细检查问题并就解决问题，不可急功近利，心焦气躁。这次的实验不仅将课本上学的理论知识运用到了实际中，更提高了我的实验动手能力，是我对于硬件类实验产生了极高的兴趣，虽然课程内容快结束

了，但是对于我今后运用课上所学的知识进行硬件编程却只是刚刚开始！

最后感谢老师们不厌其烦的为我解答疑惑并帮助我调试 bug，我才能顺利的完成了这次实验！