自主性学习与研究项目一

课程名称	EDA 技术			
项目名称	基于 FPGA 的综合计时系统的设计与实现			
实验地点	电气楼 203	实验时间	2023 年 5 月 7 日	
考核项目	目标 1	目标 2	目标 3	目标 4
考核成绩				
指导教师	谭会生			
学院名称	轨道交通学院			
学生班级	电子科学与技术 2102			
学生姓名	付允松			
学生学号	21419000503			

湖南工业大学轨道交通学院　制

项目一　基于 FPGA 的综合计时系统的设计与实现

一、 研究项目概述

现欲设计一个综合性的计时系统，要求能实现年、月、日、时、分、秒及星期的计数等综合计时功能，同时将计时结果通过 15 个七段数码管实现显示，并且可通过两个设置键，在计时过程中，对计时系统的有关参数进行调整。具体系统功能面板如下所示

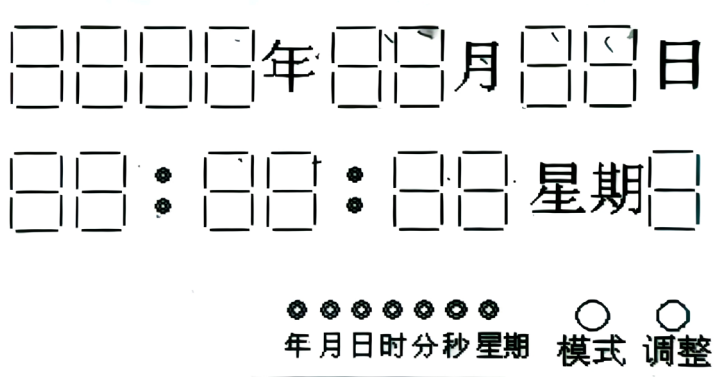


图 1

二、 研究项目设计方案比较

1） 设计方案一：

计数结果直接电路中，通过硬件译码模块在 15 个数码管中分别显示年、月、日、时、分、秒及星期

2） 设计方案二：

计数结果经过 VHDL 内部译码模块后，通过控制信号在 8 个数码管上分批显示年、月、日、时、分、秒及星期，首先显示秒、分、时

由于实验硬件环境中没有译码电路，同时只有 8 个数码管，因此采用方案二

三、 研究项目系统结构设计

1.综合计时电路的设计

根据系统的设计要求，综合计时电路可分为计秒电路、计分电路、计时电路、计星期电路、计日电路、计月电路、计年电路等 7 个子模块，这了个子模块都必须具有预置、计数和进位功能，其设计思想如下：

(1)计秒电路：直接输入或由分频器产生的秒脉冲作为计秒电路的计数时钟信号，待计数至 60 瞬间，进位至计分电路加 1，而计秒电路则清零并重新计秒。

(2)计分电路、计时电路：其设计思想与计秒电路类似。

(3)计星期电路：将计时电路产生的进位脉冲信号作为计星期电路的计数时钟信号，待计数至 7 瞬间，计星期电路返回 1 重新开始计数。

(4)计日电路：将计时电路产生的进位脉冲信号作为计日电路的计数时钟信号通过系统辨认，确定本月总天数 X(包括 28、29、30、31 四种情况)，待计数至 X+1 瞬间，进位至计月电路加 1，而计日电路返回 1 重新开始计数。

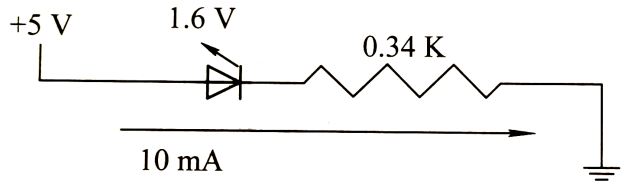
(5)计月电路：将计日电路产生的进位脉冲信号作为计月电路的计数时钟信号，待计数至 12 瞬间，进位至计年电路加 1，而计月电路返回 1 重新开始计数。

(6)计年电路：将计月电路产生的进位脉冲信号作为计年电路的计数时钟信号，待计数至 100 瞬间，计年电路返回 0 重新开始计数。(由于本系统的计年范围仅为 2000～2099 年，所以计年模块只对年份的后两位进行计数，年份的前两位始终保持为“20”。)

2.显示控制电路的设计

本设计显示需要使用的是 15 个七段显示数码管。在计时结果显示电路中，七段数码管显示部分是一个不容忽视的环节，如若处理不得当，可能引起系统功率过大，产生散热问题，严重时甚至会导致系统的烧毁。为了解决好以上问题，下面对七段数码管显示电路做简要的分析和介绍。

通常点亮一个 LED 所需的电流是 5～50mA，通电的电流愈大 1ED 的亮度愈高，相处的也会使其左命缩短。一般以 10mA 的导通电流来估算它所必须申联的阻值，其计算方式如下图



$$(5-1.6)/10\ \text{mA} = 0.34\ \text{k}\Omega$$

图 2

而七段显示器可分为共阳极、共阴极型两种，它们都可以等效成 8 个 LED 的连接电路，其中图 6.38 就是共阴极型七段显示器的等效电路和每节 LED 的定义位置图。

因此，若是要点亮七段显示器并实现一个 3 的数字符号并不点亮 P 点 LED，则输入七段显示码是“01001111”，而且这个码字的每个位所对应位置和图 6.38 相同，顺序是“pgfedcba”。依此类推可得到 0～F 的显示码。

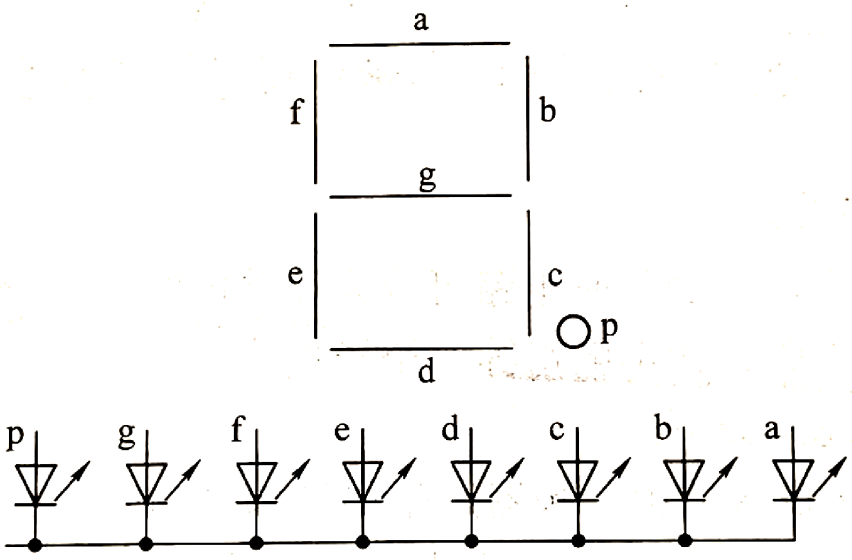


图 3

由于本设计的目标是设计一个综合的计时系统，要求同时显示年(在这里年份的前两位固定为 20)、月、日、时、分、秒及星期共 15 个数字。依照图 6.37 的计算方式，同时点亮一个七段显示器的 8 节 LED，需电流：10mA×8=80mA。若再进一步同时点亮 15 个七段显示器，这时所需电流为 80 mA×15=1200 mA=1.2A。这对于一般的电子电路来说，是一个不小的电流，不但 CPLD 和 FPGA 无法负荷这样的电流驱动，而 H.这个功率也太大，散热是个问题，容易造成电路被烧毁。因此显示电路部分不能直接实现各个计时结果同时显示，只能另外通过一个扫描电路对计时输出进行逐不扫描，使七段数码管以两个为一组，逐个进行显示。只要每个扫描频率超过人的眼睛视觉暂留频率 24Hz，就可以达到点亮两个七段数码管，却能享有所有七段数码管同时显示的视觉效果，而且显示也不致闪烁抖动，从而间接实现计时结果同时显示。

根据以上设计思想，本系统的数据显示电路可分为两个子模块：

(1)显示控制电路 XSKZQ：负责完成数据选择扫描及数码管位选择信号的产生、数据扫描选择输出、对选择的数据进行 BCD 码转换等功能。

(2)显示译码电路：将用于显示的 BCD 码数据进行译码。

XSKZQ 的输入，输出端口定义如下：输入信号 CLK_SCAN 为用于产生数据选择扫描等控制信号的时钟信号；输入信号 SEC、MIN、HOUR、DAY、YUE、NIAN、WEEK 分别来自计秒电路、计分电路、计时电路、计日电路、计月电路、计年电路、计星期电路等计时电路的计时结果输出端；输出信号 HBCD 和 LBCD 为被选择进行显示的计秒/计分/计时计日/计月/计年/计星期电路等计时电路的计时结果的高性能 BCD 码和低性能 BCD 码，输

第 6 章 VHDL 设计应用实例

出端 SELOUT 经外部的 3-8 译码电路译码后用于选择对应计时结果显示数码 COM。下图为 15 个共阴极型七段数码管驱动接线图。

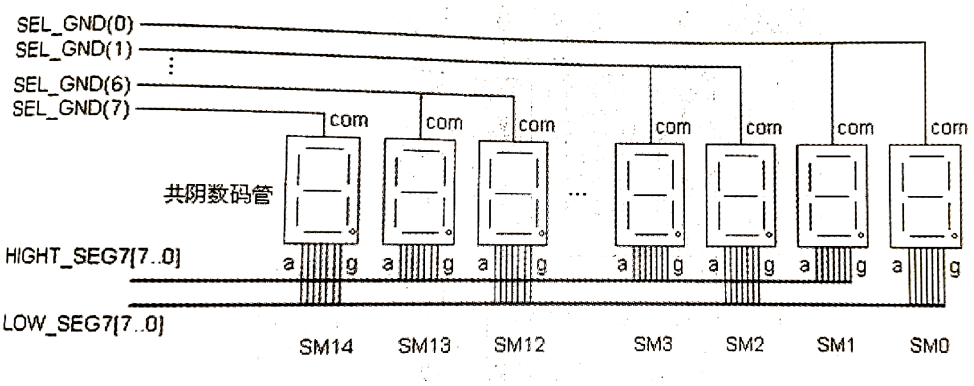


图 4

3.调整控制电路的设计

系统中的时间调整电路，拟通过模式和调整两个外部按键对系统的计时进程进行模式键：负责切换正常时间计数模式和时间调整模式，调整模式切换顺序，如下图调整键：在时间调整模式之下，对当前模式的计时结果进行调整。

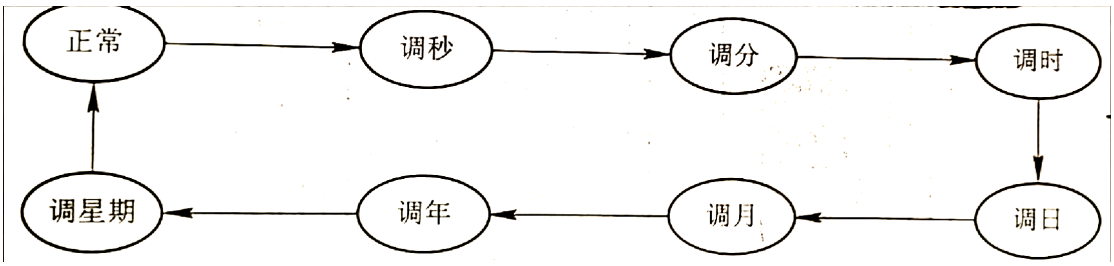


图 5

在模式选择过程中，被选择到的调整模式，其所对应的发光二极管会被点亮。例如：按动模式键，选定“2003 年 6 月 5 日 12:34:56 星期 4”的小时数“12”，其对应的调时式发光二极管将会被点亮，剩下的六个调整模式发光一极管不被点亮。当处于正常模式时，七个发光二极管均不被点亮。被调整的计时结果之间相互独立。调整过程中，只有被选择到的计数结果才会接受调整，例如上述的“12”小时数部分，它可能会有的调整范围

是 0～23，其余的数字将固定显示不动。

TZKZQ 模块负责各个模式之间的相互切换以及对被选中模式进行时间调整。TZKZQ 模块的输入、输出端口定义如下：输入信号 KEY1.0]为键盘信号，当 KEY=01 时，表示这下了设置键，系统切换到下一状态。当 KEY=10°时，表示按下了调整键，系统进行自加：输入信号 CLK__KEY 为按键扫描时钟信号；输入信号 YEAR_CUR、MON_COR、DAY_CUR、HOUR_CUR、MIN_CUR、SEC_CUR、WEEK_CUR、MAX_DAYS 均为来自各进时电联输出的当前计时结果的反馈值)输出信号 SEC_EN、MIN_EN、HOUR EN、DAY_EN, MON_EN、YEAR_EN、WEEK_EN 均为对应的计时电路的异步并行置数使能信是：输出信号 SEC、MIN、HOUR、 DAY、MON, YEAR、WEEK 则为调整后的对应时间预置的数。该模块的 VHDL 程序主要通过一个状态机来实现，对应的状态输出及状态转换条件请参看后述的 TZKZQ.VHD。

4.系统总体电路的设计

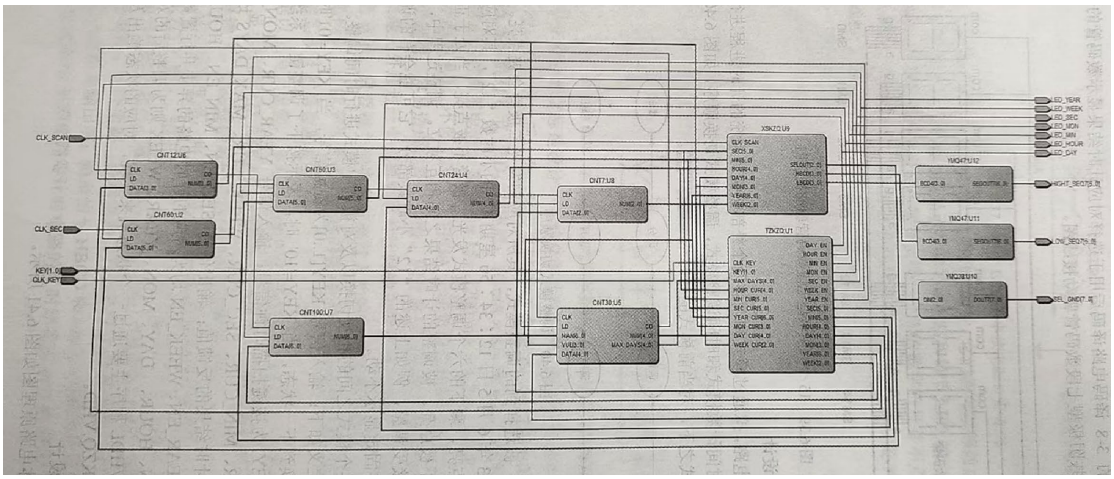


图 6

四、 系统主要 VHDL 源程序设计

TZKZQ.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity TZKZQ is
    port(key3: in std_logic;
         key4: in std_logic;
         key5: in std_logic;
         key6: in std_logic;
         clk_key:in std_logic;
         max_days:in std_logic_vector(4 downto 0);
         sec_en,min_en,hour_en,day_en,
         mon_en,year_en,week_en:out std_logic;
         hour_cur:in std_logic_vector(4 downto 0);
         min_cur,sec_cur:in std_logic_vector(5 downto 0);
         year_cur:in std_logic_vector(6 downto 0);
         mon_cur:in std_logic_vector(3 downto 0);
         day_cur:in std_logic_vector(4 downto 0);
         week_cur:in std_logic_vector(2 downto 0);
         sec,min:buffer std_logic_vector(5 downto 0);
         hour:buffer std_logic_vector(4 downto 0);
         day:buffer std_logic_vector(4 downto 0);
         mon:buffer std_logic_vector(3 downto 0);
         year:buffer std_logic_vector(6 downto 0);
         week:buffer std_logic_vector(2 downto 0));
end entity TZKZQ;
architecture art of tzkzq is
type statetype is (normal,sec_set,min_set,hour_set,
                   day_set,mon_set,year_set,week_set);
    signal mode :statetype;
    signal key :std_logic_vector(2 downto 0);
    begin
    key<=key3&key4&key5;

    process(key,clk_key) is
    begin
    -- sec_en<="1";
    -- min_en<="1";
    -- hour_en<="1";
    -- day_en<="1";
    -- mon_en<="1";
    -- year_en<="1";
```

```
    -- week_en<="1";
--assigning initial values in VHDL synthesis is meaningless

    if clk_key'event and clk_key='1' then

    if (key="000") then
        sec_en<='1';
        min_en<='1';
        hour_en<='1';
        day_en<='1';
        mon_en<='1';
        year_en<='1';
        week_en<='1';
    end if;
    --give an initial state based on the initial hardware in sequential statements

    if (key="001") then mode <=sec_set;sec_en<='0';sec<=sec_cur;
    else sec_en<='1';
    end if;

    if (key="010") then mode <=min_set;min_en<='0';min<=min_cur;
    else min_en<='1';
    end if;

    if (key="011") then mode <=hour_set;hour_en<='0';hour<=hour_cur;
    else hour_en<='1';
    end if;

    if (key="100") then mode <=day_set;day_en<='0';day<=day_cur;
    else day_en<='1';
    end if;

    if (key="101") then mode <=mon_set;mon_en<='0';mon<=mon_cur;
    else mon_en<='1';
    end if;

    if (key="110") then mode <=year_set;year_en<='0';year<=year_cur;
    else year_en<='1';
    end if;

    if (key="111") then mode <=week_set;week_en<='0';week<=week_cur;
    else week_en<='1';
    end if;
```

```
--when the set signal is active, there will be no carry
--when the set signal is invalid, counting will continue

    if(key6='1') then
    case mode is
    when sec_set=>
        if sec="111011" then
                        sec<="000000";
                else
                    sec<=sec+1;
                end if;
    when min_set=>
        if min="111011" then
                        min<="000000";
                else
                    min<=min+1;
                end if;
    when hour_set=>
        if hour="11000" then
                        hour<="00000";
                else
                    hour<=hour+1;
                end if;
    when day_set=>
        if day=max_days then
                        day<="00001";
                else
                    day<=day+1;
                end if;
    when mon_set=>--mon<=mon_cur;
        if mon="1100" then
                        mon<="0001";
                else
                    mon<=mon+1;
                end if;
    when year_set=>
        if year="1100011" then
                        year<="0000001";
                else
                    year<=year+1;
                end if;
    when week_set=>
        if week="111" then
                        week<="001";
```

```
                    else
                        week<=week+1;
                    end if;
    when others =>null;
    end case;
    end if;

    end if;
    end process;
end architecture art;
```

XSKZQ.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
entity XSKZQ is
    port(clk_scan:in std_logic;
         --Scanning clock requires a fast speed
         sec,min:in std_logic_vector(5 downto 0);
         hour:in std_logic_vector(4 downto 0);
         day:in std_logic_vector(4 downto 0);
         mon:in std_logic_vector(3 downto 0);
         year:in std_logic_vector(6 downto 0);
         week:in std_logic_vector(2 downto 0);
         selout:out std_logic_vector(3 downto 0);
         hbcd,lbcd:out std_logic_vector(3 downto 0));
end entity XSKZQ;

architecture art of XSKZQ is
    signal temp1,temp2:integer range 0 to 9;
    signal cnt:std_logic_vector(3 downto 0);
    begin
    process(clk_scan) is
    begin
        if clk_scan'event and clk_scan='1' then
            if cnt="1111" then
                cnt<="0000";
            else
                cnt<=cnt + '1';
            end if;
        end if;
    end process;
```

```
selout<=cnt;

process(cnt) is
  begin
  case cnt is
      when"0000" =>temp1<=conv_integer(sec)-conv_integer(sec)/10*10;
      when"0001" =>temp2<=(conv_integer(sec))/10;
      --sec
      when"0010" =>temp1<=conv_integer(min)-conv_integer(min)/10*10;
      when"0011" =>temp2<=(conv_integer(min))/10;
      --min
      when"0100" =>temp1<=conv_integer(hour)-conv_integer(hour)/10*10;
      when"0101" =>temp2<=conv_integer(hour)/10;
      --hour
      when"0110" =>temp1<=conv_integer(day)-conv_integer(day)/10*10;
      when"0111" =>temp2<=conv_integer(day)/10;
      --day
      when"1000" =>temp1<=conv_integer(mon)-conv_integer(mon)/10*10;
      when"1001" =>temp2<=conv_integer(mon)/10;
      --mon
      when"1010" =>temp1<=conv_integer(year)-conv_integer(year)/10*10;
      when"1011" =>temp2<=conv_integer(year)/10;
      --year
      when"1100" =>temp1<=0;
      when"1101" =>temp2<=2;
      --year
      when"1110" =>temp1<=conv_integer(week)-conv_integer(week)/10*10;
      when"1111" =>temp2<=conv_integer(week)/10;
      --weak
      when others=>temp1<=0;temp2<=0;
  end case;

case temp1 is
    when 0 =>lbcd<="0000";
    when 1 =>lbcd<="0001";
    when 2 =>lbcd<="0010";
    when 3 =>lbcd<="0011";
    when 4 =>lbcd<="0100";
    when 5 =>lbcd<="0101";
    when 6 =>lbcd<="0110";
    when 7 =>lbcd<="0111";
    when 8 =>lbcd<="1000";
    when 9 =>lbcd<="1001";
    when others =>lbcd<="0000";
```

```
    end case;

    case temp2 is
        when 0 =>hbcd<="0000";
        when 1 =>hbcd<="0001";
        when 2 =>hbcd<="0010";
        when 3 =>hbcd<="0011";
        when 4 =>hbcd<="0100";
        when 5 =>hbcd<="0101";
        when 6 =>hbcd<="0110";
        when 7 =>hbcd<="0111";
        when 8 =>hbcd<="1000";
        when 9 =>hbcd<="1001";
        when others =>hbcd<="0000";
    end case;

    end process;

end architecture art;
```

LED_XS.vhd

```
library ieee;
use ieee.std_logic_1164.all;
entity LED_XS is
  port(sec_en ,min_en,hour_en,day_en,
       mon_en,year_en,week_en:in std_logic;
       led1: out std_logic;
       led2: out std_logic;
       led3: out std_logic;
       led4: out std_logic;
       led5: out std_logic;
       led6: out std_logic;
       led7: out std_logic;
       led8: out std_logic);
end entity led_xs;
architecture art of led_xs is
  begin
  process(sec_en ,min_en,hour_en,day_en,mon_en,year_en,week_en) is
  begin

    if(sec_en='1' and min_en='1' and hour_en='1' and day_en='1'
       and mon_en='1'and year_en='1'and week_en='1')     then
    led1<='1';led2<='1';led3<='1';led4<='1';
    led5<='1';led6<='1';led7<='1';led1<='1';
```

```
    end if;--give an initial state

    if(sec_en = '0') then
      led1<='0';
    else led1<='1';
    end if;

    if(min_en = '0') then
      led2<='0';
    else led2<='1';
    end if;

    if(hour_en = '0') then
      led3<='0';
    else led3<='1';
    end if;

    if(day_en = '0') then
      led4<='0';
    else led4<='1';
    end if;

    if(mon_en = '0') then
      led5<='0';
    else led5<='1';
    end if;

    if(year_en = '0') then
      led6<='0';
    else led6<='1';
    end if;

    if(week_en = '0') then
      led7<='0';
    else led7<='1';
    end if;

    led8<='1';

    end process;
end architecture;
```

TOP.vhd

```
library ieee;
use ieee.std_logic_1164.all;
entity TOP is
  port(clk:in std_logic;--clock frequency division
   --s_sec_en:buffer std_logic;
   --s_min_en:buffer std_logic;
   --s_hour_en:buffer std_logic;
   --s_day_en:buffer std_logic;
   --s_mon_en:buffer std_logic;
   --s_year_en:buffer std_logic;
   --s_week_en:buffer std_logic;
   --s_selout:buffer std_logic_vector(3 downto 0);
   --s_newclk:buffer std_logic;
       key1:in std_logic;--key
       key2:in std_logic;
       key3:in std_logic;
       key4:in std_logic;
       key5:in std_logic;
       key6:in std_logic;
       led1:out std_logic;--led
       led2:out std_logic;
       led3:out std_logic;
       led4:out std_logic;
       led5:out std_logic;
       led6:out std_logic;
       led7:out std_logic;
       led8:out std_logic;
       com:out std_logic_vector(7 downto 0);--common select
       seg:out std_logic_vector(7 downto 0);--segment select
       clr:in std_logic);--Clear signal (active high)
end entity;
architecture art of TOP is

  component CNT7 is
    port(ld:in std_logic;--Set signal (active low)
         clk:in std_logic;--Clock pulse (rising edge triggered))
         data:in std_logic_vector(2 downto 0);
         key6:in std_logic;
         num:buffer std_logic_vector(2 downto 0));
  end component CNT7;

  component CNT12 is
    port(ld:in std_logic;--Set signal (active low)
         clk:in std_logic;--Clock pulse (rising edge triggered))
```

```
        data:in std_logic_vector(3 downto 0);
        num:buffer std_logic_vector(3 downto 0);
        key6:in std_logic;
        co:out std_logic);--carry signal
  end component CNT12;

  component CNT24 is
    port(ld:in std_logic;--Set signal (active low)
         clk:in std_logic;--Clock pulse (rising edge triggered))
         data:in std_logic_vector(4 downto 0);
         num:buffer std_logic_vector(4 downto 0);
         key6:in std_logic;
         co:out std_logic);--carry signal
  end component CNT24;

  component CNT30 is
   port(ld: in std_logic;--Set signal (active low)
         clk:in std_logic;--Clock pulse (rising edge triggered)
           datain: in std_logic_vector(4 downto 0);--preset value
           year :in std_logic_vector(6 downto 0);--current year
           yue: in std_logic_vector(3 downto 0);--current month
           num: buffer std_logic_vector( 4 downto 0);--counting result
           maxday: out std_logic_vector(4 downto 0);--total number of days in this
month
       key6:in std_logic;
           co : out std_logic);--carry signal
  end component CNT30;

  component CNT60 is
    port(ld:in std_logic;--Set signal (active low)
        clr:in std_logic;--Clear signal (active high)
        clk:in std_logic;--Clock pulse (rising edge triggered))
        data:in std_logic_vector(5 downto 0);
        num:buffer std_logic_vector(5 downto 0);
        key6:in std_logic;
        co:out std_logic);--carry signal
  end component CNT60;

  component CNT100 is
    port(ld:in std_logic;--Set signal (active low)
         clk:in std_logic;--Clock pulse (rising edge triggered))
         data:in std_logic_vector(6 downto 0);
         num:buffer std_logic_vector(6 downto 0);
         key6:in std_logic;
```

```
       co : out std_logic);--carry signal
end component CNT100;

component DISPLAY is
  port(hbcd:in std_logic_vector(3 downto 0);
       lbcd:in std_logic_vector(3 downto 0);
       selout:in std_logic_vector(3 downto 0);
       key1: in std_logic;
       key2: in std_logic;
       --Clock display for date, with a period of 1 second
       com: out std_logic_vector(7 downto 0);
      --  bcd: buffer std_logic_vector(3 downto 0);
      --  div2clk: buffer std_logic;
       seg: out std_logic_vector(7 downto 0));
end component;

component TZKZQ is
  port(key3: in std_logic;
         key4: in std_logic;
       key5: in std_logic;
       key6: in std_logic;
       clk_key:in std_logic;
       max_days:in std_logic_vector(4 downto 0);
       sec_en ,min_en,hour_en,day_en,
       mon_en,year_en,week_en:out std_logic;
       hour_cur:in std_logic_vector(4 downto 0);
       min_cur,sec_cur:in std_logic_vector(5 downto 0);
       year_cur:in std_logic_vector(6 downto 0);
       mon_cur:in std_logic_vector(3 downto 0);
       day_cur:in std_logic_vector(4 downto 0);
       week_cur:in std_logic_vector(2 downto 0);
       sec,min:buffer std_logic_vector(5 downto 0);
       hour:buffer std_logic_vector(4 downto 0);
       day:buffer std_logic_vector(4 downto 0);
       mon:buffer std_logic_vector(3 downto 0);
       year:buffer std_logic_vector(6 downto 0);
       week:buffer std_logic_vector(2 downto 0));
end component TZKZQ;

component XSKZQ is
  port(clk_scan:in std_logic;
      --Scanning clock requires a fast speed
      sec,min:in std_logic_vector(5 downto 0);
      hour:in std_logic_vector(4 downto 0);
```

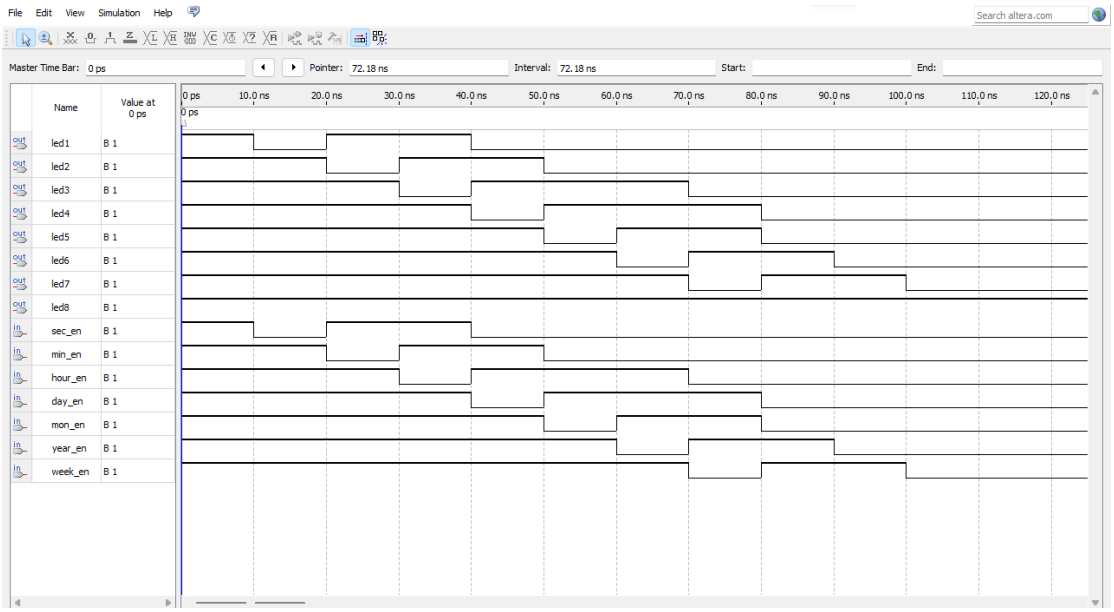
```
      day:in std_logic_vector(4 downto 0);
      mon:in std_logic_vector(3 downto 0);
      year:in std_logic_vector(6 downto 0);
      week:in std_logic_vector(2 downto 0);
      selout:out std_logic_vector(3 downto 0);
      hbcd,lbcd:out std_logic_vector(3 downto 0));
end component XSKZQ;

component LED_XS is
          port(sec_en ,min_en,hour_en,day_en,
       mon_en,year_en,week_en:in std_logic;
       led1: out std_logic;
       led2: out std_logic;
       led3: out std_logic;
       led4: out std_logic;
       led5: out std_logic;
       led6: out std_logic;
       led7: out std_logic;
       led8: out std_logic);
end component LED_XS;

component CLKGEN is
  port(clk: in std_logic;
       newclk: out std_logic);
end component CLKGEN;

signal s_sec,s_min:std_logic_vector(5 downto 0);
signal s_hour:std_logic_vector(4 downto 0);
signal s_day:std_logic_vector(4 downto 0);
signal s_mon:std_logic_vector(3 downto 0);
signal s_year:std_logic_vector(6 downto 0);
signal s_week:std_logic_vector(2 downto 0);

signal s_sec_en:std_logic;
signal s_min_en:std_logic;
signal s_hour_en:std_logic;
signal s_day_en:std_logic;
signal s_mon_en:std_logic;
signal s_year_en:std_logic;
signal s_week_en:std_logic;

signal s_sec_cur,s_min_cur:std_logic_vector(5 downto 0);
signal s_hour_cur:std_logic_vector(4 downto 0);
signal s_day_cur:std_logic_vector(4 downto 0);
```

```
  signal s_mon_cur:std_logic_vector(3 downto 0);
  signal s_year_cur:std_logic_vector(6 downto 0);
  signal s_week_cur:std_logic_vector(2 downto 0);

  signal s_sec_co60,s_min_co60,s_co24,s_co30,s_co12,s_co100:std_logic;

  -- signal s_newclk:std_logic;
  signal s_lbcd,s_hbcd:std_logic_vector(3 downto 0);
  -- signal s_selout:std_logic_vector(3 downto 0);
  signal s_maxday:std_logic_vector(4 downto 0);
  begin
  u1:CLKGEN
    port map(clk,s_newclk);
  u2:CNT7
    port map(s_week_en,s_co24,s_week,key6,s_week_cur);
  u3:CNT12
    port map(s_mon_en,s_co30,s_mon,s_mon_cur,key6,s_co12);
  u4:CNT24
    port map(s_hour_en,s_min_co60,s_hour,s_hour_cur,key6,s_co24);
  u5:CNT30
    port
map(s_day_en,s_co24,s_day,s_year,s_mon,s_day_cur,s_maxday,key6,s_co30);
  u6:CNT60
    port map(s_sec_en,clr,s_newclk,s_sec,s_sec_cur,key6,s_sec_co60);
    u7:CNT60
    port map(s_min_en,clr,s_sec_co60,s_min,s_min_cur,key6,s_min_co60);
  u8:CNT100
    port map(s_year_en,s_co12,s_year,s_year_cur,key6,s_co100);
  u9:DISPLAY
    port map(s_hbcd,s_lbcd,s_selout,key1,key2,com,seg);
  u10:TZKZQ
    port map(key3,key4,key5,key6,s_newclk,s_maxday,s_sec_en,s_min_en,
             s_hour_en,s_day_en,s_mon_en,s_year_en,s_week_en,s_hour_cur,
             s_min_cur,s_sec_cur,s_year_cur,s_mon_cur,s_day_cur,
             s_week_cur,s_sec,s_min,s_hour,s_day,s_mon,s_year,
             s_week);
  u11:XSKZQ
    port
map(clk,s_sec_cur,s_min_cur,s_hour_cur,s_day_cur,s_mon_cur,s_year_cur,s_week_cu
r,
             s_selout,s_hbcd,s_lbcd);
  u12:LED_XS
    port map(s_sec_en,s_min_en,s_hour_en,s_day_en,s_mon_en,s_year_en,
             s_week_en,led1,led2,led3,led4,led5,led6,
```

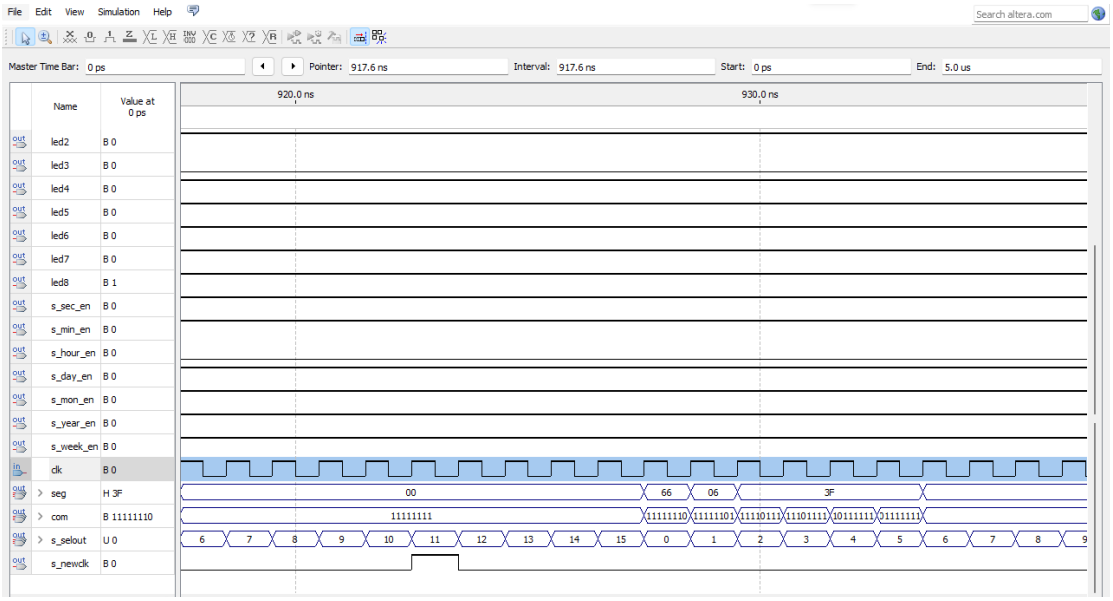
```
                led7,led8);
end architecture art;
```

五、 主要 VDHL 源程序仿真结果

LED_XS



TOP



六、 硬件验证方案及验证结果

硬件验证方案：

为了验证基于 FPGA 的综合计时系统的设计与实现项目的正确性，我采用了基于仿真波形观察的验证方案。具体来说，我使用 quartus 工具进行仿真，观察各个模块的输出波形，以确保综合计时系统的各个模块的功能正常。

首先，我对时钟模块进行了仿真验证。我观察时钟信号的波形，以确保时钟信号的产生和分频功能正常。通过观察时钟信号的频率和占空比等参数，我验证了时钟信号的稳定性和精度。

接着，我对计数模块进行了仿真验证。我观察计数器的输出波形，以确保计数器的计数和显示功能正常。通过观察计数器的计数范围和步进值，我验证了计数器的准确性和可靠性。

然后，我对显示控制模块进行了仿真验证。我观察该模块的输出波形，以确保显示控制器能够正确地控制七段数码管的显示。通过观察七段数码管的显示结果，我验证了显示控制器的功能和正确性。

最后，我对调整控制模块进行了仿真验证。我观察该模块的输出波形，以确保调整控制器能够正确地对时钟信号进行调整。通过观察时钟信号的频率和占空比等参数，我验证了调整控制器的功能和正确性。

验证结果：

经过仿真验证，我成功地验证了基于 FPGA 的综合计时系统的设计与实现项目的正确性。时钟模块能够产生稳定的时钟信号，并能够对时钟信号进行分频，以实现不同的计数功能。计数模块能够正确地计数和显示计数结果，并能够在达到计数范围时停止计数。显示控制模块能够正确地控制七段数码管的显示。调整控制模块能够正确地对显示信号进行调整，以确保显示数据的准确性。

总体来说，基于 FPGA 的综合计时系统的设计与实现项目在仿真验证方面表现出了良好的性能和可靠性。

以下是相关步骤：

芯片管脚的锁定

工程编译和有关仿真都通过后，就可以将配置数据下载到应用系统进行验证。下载之前首先要对系统顶层模块进行引脚锁定保证锁定的引脚与实际的应用系统相吻合。

1)目标芯片的确认及闲置引脚的设定

管脚锁定前，先进行芯片的确定或修改，如图 4.43 所示。单击图 4.43 中的【Device and Pin Options...】按钮，在弹出的【Unused Pins】设置框中进行闲置引脚的设定，详见图 4.25 所示。对设计中未用到的器件引脚，有三种处理方式：输入引脚(呈高组态)、输出引脚(呈低电平)或输出引脚(输出不定状态)。通常情况下选择第一项，避免未用到的引脚对应用系统产生影响。

2)引脚锁定

本设计系统的顶层模块 TOP 拟选用 EP3C55484C8 芯片，根据需使用的 FDA 资验开发系统(板)的有关输入和输出的资源情况进行引脚锁定,一般应事先列出一个管脚领定表，并将闲置引脚设定为三态门状态。

引脚的锁定方法有三种：一是使用引脚锁定窗口进行锁定；二是使用记事本或其他文木

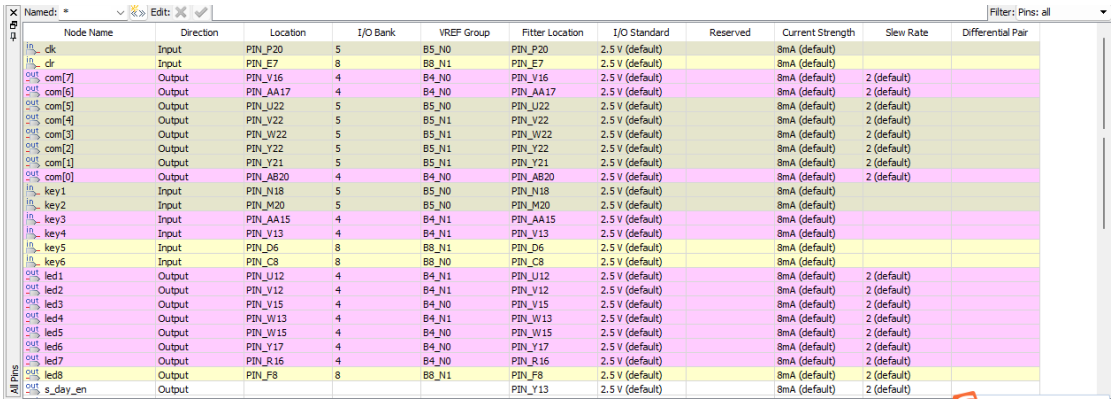
编辑工具直接编辑.qsf 文件进行引脚锁定：三是通过输入 TCL 脚本语言文件进行,本次实验使用第一种：

使用引脚锁定窗口进行锁定

打开【Assignments】菜单下的【Pins】命令，打开引脚锁定窗口，如图 4.44 所示。先用鼠标指到要锁定的端口信号名与【Location】栏交汇的地方，这时此处呈蓝色，然后双击对应的交汇处，在出现的下拉栏中选择对应端口信号名的器件引脚号(例如对应 ENA，选择 PIN_99)，引脚锁定后将下拉菜单复原，则系统自动保存该锁定。在如图 4.44 所示的窗口中，还能对引脚作进一步的设定，如在 Reserved 栏，可对某些空闲的 1/O 引脚的电气特性进行设置。

按前面提到的引脚信息添加锁定引脚，直到全部输入完毕。锁定引脚后必须再编译次本能将引腿锁定信息应用到最终的下载文件中，此后就可以将编译好的 SOF 文件下载到实验系统的 FPGA 中了。

管脚锁定图



Named: * Edit: Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair
clk	Input	PIN_P20	5	B5_N0	PIN_P20	2.5 V (default)		8mA (default)		
clr	Input	PIN_E7	8	B8_N1	PIN_E7	2.5 V (default)		8mA (default)		
com[7]	Output	PIN_V16	4	B4_N0	PIN_V16	2.5 V (default)		8mA (default)	2 (default)	
com[6]	Output	PIN_AA17	4	B4_N0	PIN_AA17	2.5 V (default)		8mA (default)	2 (default)	
com[5]	Output	PIN_U22	5	B5_N0	PIN_U22	2.5 V (default)		8mA (default)	2 (default)	
com[4]	Output	PIN_V22	5	B5_N1	PIN_V22	2.5 V (default)		8mA (default)	2 (default)	
com[3]	Output	PIN_W22	5	B5_N1	PIN_W22	2.5 V (default)		8mA (default)	2 (default)	
com[2]	Output	PIN_Y22	5	B5_N1	PIN_Y22	2.5 V (default)		8mA (default)	2 (default)	
com[1]	Output	PIN_Y21	5	B5_N1	PIN_Y21	2.5 V (default)		8mA (default)	2 (default)	
com[0]	Output	PIN_AB20	4	B4_N0	PIN_AB20	2.5 V (default)		8mA (default)	2 (default)	
key1	Input	PIN_N18	5	B5_N0	PIN_N18	2.5 V (default)		8mA (default)		
key2	Input	PIN_M20	5	B5_N0	PIN_M20	2.5 V (default)		8mA (default)		
key3	Input	PIN_AA15	4	B4_N1	PIN_AA15	2.5 V (default)		8mA (default)		
key4	Input	PIN_V13	4	B4_N1	PIN_V13	2.5 V (default)		8mA (default)		
key5	Input	PIN_D6	8	B8_N1	PIN_D6	2.5 V (default)		8mA (default)		
key6	Input	PIN_C8	8	B8_N0	PIN_C8	2.5 V (default)		8mA (default)		
led1	Output	PIN_U12	4	B4_N1	PIN_U12	2.5 V (default)		8mA (default)	2 (default)	
led2	Output	PIN_V12	4	B4_N1	PIN_V12	2.5 V (default)		8mA (default)	2 (default)	
led3	Output	PIN_V15	4	B4_N0	PIN_V15	2.5 V (default)		8mA (default)	2 (default)	
led4	Output	PIN_W13	4	B4_N1	PIN_W13	2.5 V (default)		8mA (default)	2 (default)	
led5	Output	PIN_W15	4	B4_N0	PIN_W15	2.5 V (default)		8mA (default)	2 (default)	
led6	Output	PIN_Y17	4	B4_N0	PIN_Y17	2.5 V (default)		8mA (default)	2 (default)	
led7	Output	PIN_R16	4	B4_N0	PIN_R16	2.5 V (default)		8mA (default)	2 (default)	
led8	Output	PIN_F8	8	B8_N1	PIN_F8	2.5 V (default)		8mA (default)	2 (default)	
s_day_en	Output				PIN_Y13	2.5 V (default)		8mA (default)	2 (default)	

All Pins

图 7

以下是实验系统管脚图

发光二极管 LED8-LED1	CON1.9-CON1.2	T17、R16、Y17、W15、 W13、V15、V12、U12
数码管驱动端 8XSEG LH-LA	CON1.32-CON1.25	M19、M21、N20、N21、 P21、R21、W20、AA20
数码管公共端 8XSEG DS8-DS1	CON1.51-CON1.44	V16、AA17、U22、V22、 W22、Y22、Y21、AB20
按键 SW4-按键 SW1	CON1.61、 CON1.54、 CON1.34、 CON1.33、	V13、AA15、M20、N18
按键 SW8-按键 SW5	CON2.59、 CON2.60、 CON2.61、CON2.62、	F8、E7、C8、D6

图 8

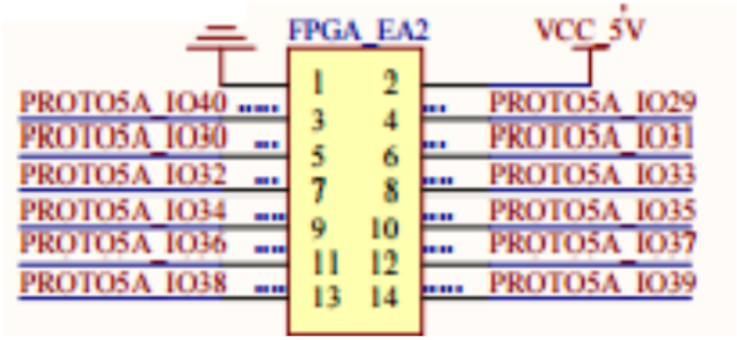
扩展端口 FPGA_EA2		
APROTO5A IO29、IO30	CON3.30、 CON3.35、	R17、N19
PROTO5A IO31、IO32	CON3.36、 CON1.59、	P20、W14
APROTO5A IO33、IO34	CON1.58、 CON1.60、	AB16、Y13
PROTO5A IO35、IO36	CON3.61、 CON3.38、	AB14、R19
PROTO5A IO37、IO38	CON3.39、 CON2.57、	T31、6B
PROTO5A IO39、IO40	CON2.56、 CON2.55、	B4、B5

图 9

编程下载及验证

1) 编程下载硬件准备

先阅读有关 EDA 实验开发系统(板)手册了解 EDA 实验开发系统(板方式。在断电的情况下将有关硬件设备进行正确的物理连接，经检查无误开发系统(板)的电源开关。

2) FPGA 的编程下载

连接好下载电缆，打开电源。在菜单【Tool】中选择【Programmer】，栏上的快捷键，可以打开编程下载窗口

在下载编程前需要选择下载接口方式。口中单击【Hardware Setup】，在打开的设置窗口根据实际情况进行设置【USB3.0】，双击鼠标后，关闭该窗口。

在【Mode】栏中有四种编程模式可以选择：JTAG、PassSerial 和 In-Socket。为了直接对 FPGA 进行配置，在编程窗口的编程模式【JTAG】，并选中下载文件右侧的第一个小方框 Program/Configure。核对名，如果此文件没有出错或者有错，单击左侧的【Add File】按钮，找 TOP.sof。单击【Start】按钮，即进入对目标器件 FPGA 的配置下载操显示为 100%时，编译成功，可以观察实验面板，进行硬件测试验证。

下载完成后观察实验现象

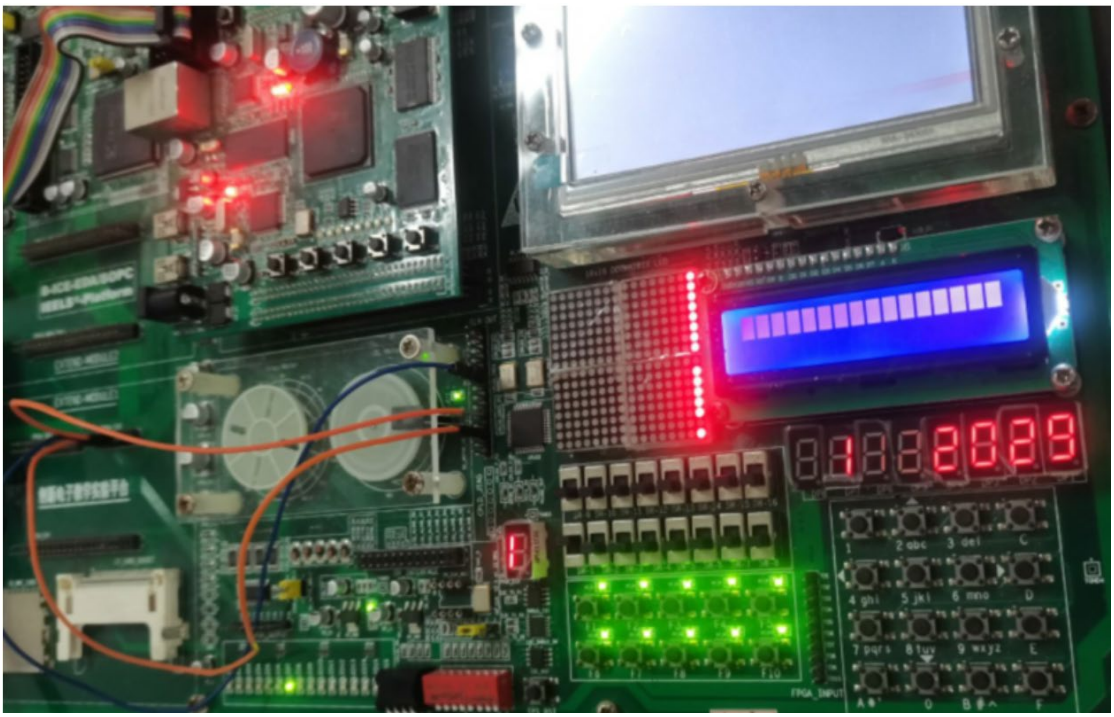


图 10

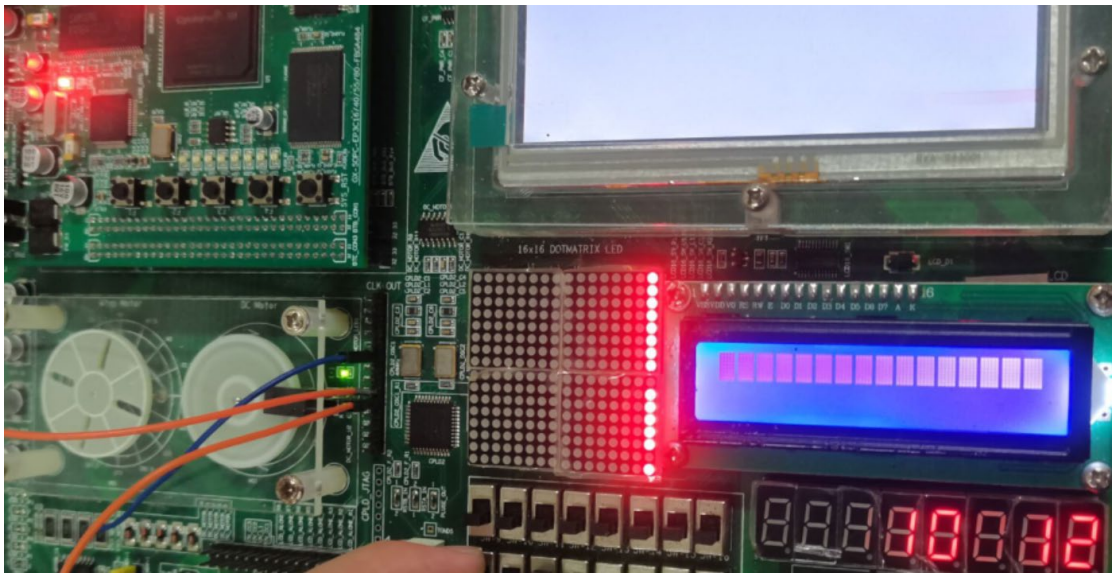


图 11

可以发现硬件现象和仿真基本一致

顶层 RTL 图：

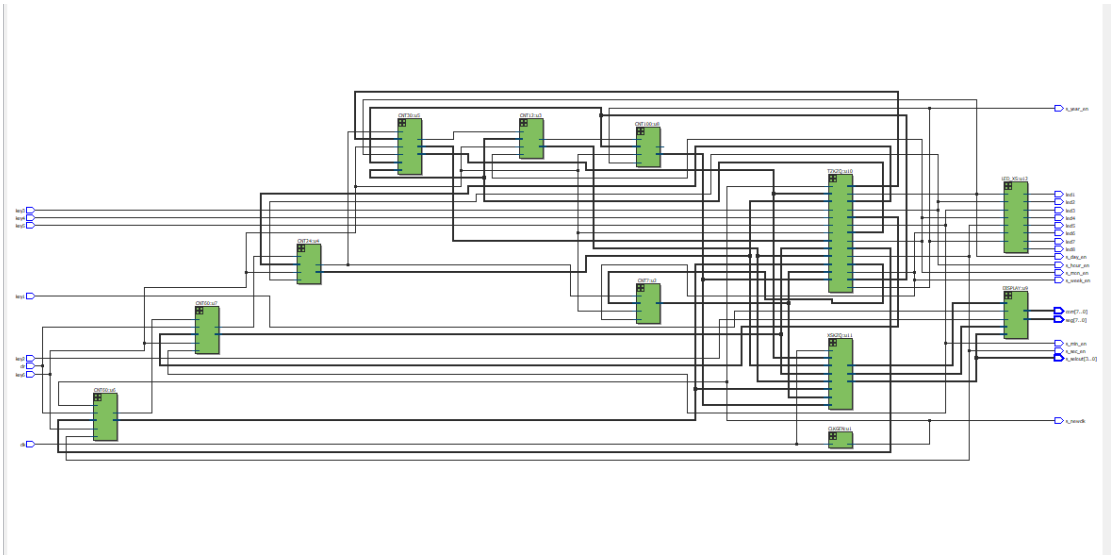


图 12

可以看到对应的模块都已显示

七、 研究性学习收获体会与研究结论

在本次基于 FPGA 的综合计时系统的设计与实现项目中，我深入学习了 FPGA 的综合计时系统的设计与实现。我首先通过学习 FPGA 的基础知识，包括 FPGA 的结构和原理、时序分析和时序约束等内容，并掌握了 VHDL 语言的基本语法和 FPGA 的设计流程。在此基础上，我开始着手设计综合计时系统。

在设计过程中，我使用了 quartus 工具对设计进行综合、布局和路由，并通过仿真验证了设计的正确性。具体来说，我先设计了一个时钟模块，实现了时钟信号的产生和分频，以及时钟的计数和显示功能。我还设计了一个控制信号，用于计数状态。最后，我将这些模块进行组合，完成了综合计时系统的设计与实现。

通过这个项目，我不仅提高了自己的编程能力，也对数字硬件设计有了更深入的理解。我深入了解了 FPGA 的可编程性和灵活性，以及其在数字电路设计中的优点和应用。我还了解了 FPGA 的设计流程和工具链，掌握了使用 quartus 工具进行综合、布局和适配的技能。

研究结论：

本次项目中，我成功地设计并实现了一个基于 FPGA 的综合计时系统。该系统通过使用 FPGA 实现了时钟的分频、计数和显示等功能。具体来说，它能够产生稳定的时钟信号，并能够对时钟信号进行分频，以实现不同的计数功能。此外，它还能够将计数结果以 7 段显示的形式输出。我使用 VHDL 语言实现了设计，通过 quartus 工具进行综合、布局和路由，并通过仿真验证了设计的正确性。在实际使用中，该系统稳定可靠，具有较高的计时精度和灵活性。

通过本次项目，我发现 FPGA 在数字电路设计方面有着广泛的应用。FPGA 具有可编程性强、功能灵活、时序性能优异等优点，适用于高速计算、图像处理、通信等领域。在未来的学习和实践中，我将继续深入学习 FPGA 的相关知识，并探索其在更多领域中的应用。我相信这些经验和技能将对我的未来学习和职业发展有所帮助。