课程设计任务书

课程名称： EDA 技术

设计题目： EDA 数字系统综合设计与实践

完成期限： 2017 年 12 月 11 日～ 2017 年 12 月 21 日 共 2 周

<table>
<tr><td>内容及任务</td><td colspan="2">数字系统设计，就是要求学生利用 EDA 技术进行综合性的数字电子系统设计，培养学生综合应用能力、独立设计与安装调试能力，并树立工程设计观念。具体任务如下：
（1）综合计时系统的设计：包括系统设计原理，VHDL 等程序设计，时序仿真与分析，逻辑综合与分析、硬件验证等；
（2）SOBEL 图像边缘检测器的设计：包括系统设计思想原理，VHDL 等程序设计，时序仿真与分析，逻辑综合与分析等；各子课题的设计具体要求详见附件一。</td></tr>
<tr><td rowspan="10">进度安排</td><td>起止日期</td><td>工作内容</td></tr>
<tr><td>第 15 周周一上午/下午（5 学时）</td><td>综合计时系统的设计（电 203）</td></tr>
<tr><td>第 15 周周二上午/下午（5 学时）</td><td>综合计时系统的设计（电 203）</td></tr>
<tr><td>第 15 周周三上午/下午（5 学时）</td><td>综合计时系统的设计（电 203）</td></tr>
<tr><td>第 15 周周四上午/下午（5 学时）</td><td>综合计时系统的设计（电 203）</td></tr>
<tr><td>第 16 周周一上午/下午（5 学时）</td><td>SOBEL 图像边缘检测器的设计（电 203）</td></tr>
<tr><td>第 16 周周二上午/下午（5 学时）</td><td>SOBEL 图像边缘检测器的设计（电 203）</td></tr>
<tr><td>第 16 周周三上午/下午（5 学时）</td><td>SOBEL 图像边缘检测器的设计（电 203）</td></tr>
<tr><td>第 16 周周四上午/下午（5 学时）</td><td>SOBEL 图像边缘检测器的设计（电 203）</td></tr>
<tr><td>其它时间</td><td>学生自行查找资料，自行进行实践，撰写报告</td></tr>
<tr><td>参考资料</td><td colspan="2">[1] 谭会生，张昌凡．EDA 技术及应用（第四版）[M]．西安：西安电子科技大学出版社，2016
[2] 谭会生，瞿遂春．EDA 技术综合应用实例与分析 [M]．西安：西安电子科技大学出版社，2004</td></tr>
</table>

附件一：各设计子课题的具体设计要求

1．综合计时系统的设计：

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图 1 所示。

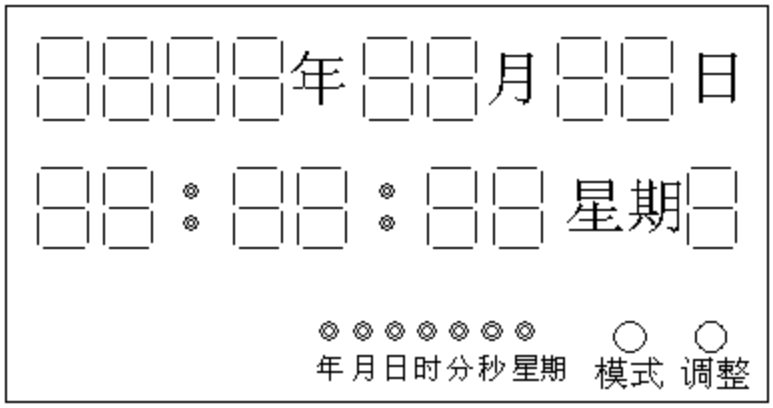


图 1　系统功能面板

2．SOBEL 图像边缘检测器的设计

边缘可定义为图像中灰度发生急剧变化的区域边界,它是图像最基本的特征，是图像分析识别前必不可少的环节,是一种重要的图像预处理技术。边缘检测主要就是(图像的)灰度变化的度量、检测和定位,它是图像分析和模式识别的主要特征提取手段，它在计算机视觉、图像分析等应用中起着重要的作用，是图像分析与处理中研究的热点问题。在过去的 20 年里产生了许多边缘检测器，如 Rorberts 算子，Sobel 算子，Prewitt 算子，Laplacian 算子等。由于 Sobel 算法只涉及加法操作，但却可以得到很好的划分效果，因而是图像处理系统中最常用的边缘检测算法。

Sobel 算法包括带 4 个 3×3 掩码的输入图像数据，即 Sobel 算子，它设置权重来检测水平、垂直、左对角、右对角各个不同方向上密度幅度的不同。这个过程通常被称为过滤。我们来看像素窗口（3×3），如图 2 所示。水平、垂直、左对角、右对角各图像方向上密度幅度的变化可以用如下算子进行计算：

H=(Q0+2Q3+Q6)-(Q2+2Q5+Q8);V=(Q0+2Q1+Q2)-(Q6+2Q7+Q8);

DR=(Q1+2Q0+Q3)-(Q5+2Q8+Q7);DL=(Q1+2Q2+Q5)-(Q3+2Q6+Q7);

H,V,DL,DR 这四个参数用于计算梯度大小和方向。

对梯度大小的一个普遍估计值为：Magnitude＝Max(H,V,DR,DL)。

Q0	Q3	Q6
Q1	[i,j]	Q7
Q2	Q5	Q8

图 2　图像窗口像素的排列

我们通过对图像灰度作直方图分析后，便可以给出区分度阀值Threshold，区分度阀值往往要借助一定的经验并需要反复调整。如果 Magnitude 大于 Threshold，则该像素被声明为边界像素，否则为一般像素。

本课题就是要求使用 LPM 兆功能块设计和 VHDL 程序设计相结合的方式或全部采用 VHDL 程序设计方式，用 FPGA/CPLD 实现 Sobel 算法。

EDA 技术

课 程 设 计 说 明 书

EDA 数字系统综合设计与实践

起止日期：2017 年 12 月 11 日～2017 年 12 月 21 日 共 2 周

学生姓名 _____

班级 电子科学与技术

学号 _____

成绩 _____

指导教师（签字） _____

交通工程学院

2017 年 12 月 21 日

一、综合计时系统的设计

1. 系统设计原理

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。如图所示，该计时系统的设计共分为三个主要模块：综合计时电路（ITC）、显示控制电路(DUC)及调整控制电路(ATCC)。其具体模块功能在此不再详细描述。

综合计时电路（ITC）可分为计秒电路、计分电路、计时电路、计星期电路、计日电路、计月电路和计年电路，共七个子模块。

显示控制电路（DUC）使用的是十五个七段显示数码管。它可分为两个子模块：

（1）.显示控制电路：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，对于选择的数据进行 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。

调整控制电路（ATCC）是通过模式和调整两个外部键完成。其中模式键负责切换正常时间计数模式和时间调整模式，调整键负责在时间调整模式下，对当前模式计时结果进行调整。在模式选择过程中，被选择到的调整模式所对应的发光二极管会被点亮。而正常模式中，7 个发光二极管都不会被点亮。模式调整的 VHDL 程序主要是通过一个状态机来实现。、

调整模式切换顺序（图）

2. VHDL 程序设计

在这次课程设计中，我们运用学习过的 VHDL 语言编译本次课程设计相关功能的模块。下面就简单的罗列一下本次课程设计相关的 VHDL 程序文件。经编译及仿真后下面所描述的文件都是正确的。且为了节约纸张，接下来只罗列一些书上并没有直接给出的一些 VHDL 源程序：

CNT7.VHD、CNT12.VHD、CNT24.VHD、CNT100.VHD、YMQ3_8.VHD、YMQ4_7.VHD，总计六个源程序并简单说明在该系统中相应的功能。

（1）--YMQ47.VHD

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
ENTITY YMQ47 IS
PORT(AIN4:IN STD_LOGIC_VECTOR(3 DOWNTO 0);
DOUT7 :OUT STD_LOGIC_VECTOR(6 DOWNTO 0));
END ENTITY YMQ47;
ARCHITECTURE ART OF YMQ47 IS
BEGIN
PROCESS(AIN4)
BEGIN
CASE AIN4 IS
WHEN"0000"=>DOUT7<="0111111";
WHEN"0001"=>DOUT7<="0000110";
WHEN"0010"=>DOUT7<="1011011";
WHEN"0011"=>DOUT7<="1001111";
WHEN"0100"=>DOUT7<="1100110";
WHEN"0101"=>DOUT7<="1101101";
WHEN"0110"=>DOUT7<="1111101";
WHEN"0111"=>DOUT7<="0000111";
WHEN"1000"=>DOUT7<="1111111";
WHEN"1001"=>DOUT7<="1101111";
```

```
WHEN"1010"=>DOUT7<="1011110";
WHEN"1011"=>DOUT7<="1011100";
WHEN"1100"=>DOUT7<="1010100";
WHEN"1101"=>DOUT7<="1111001";
WHEN OTHERS=>DOUT7<="0000000";
END CASE;
END PROCESS;
END ARCHITECTURE ART;
```

将显示控制器（DUC）中输出的BCD码进行译码后作为七段共阴极发光二极管的输入信号，实现了最终的阿拉伯数字显示在系统功能面板上。（注：该显示部分由十五个共阴极发光二极管构成。）

（2）--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;
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(2 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(2 DOWNTO 0);
BEGIN
PROCESS(CLK_SCAN) IS
BEGIN
IF CLK_SCAN'EVENT AND CLK_SCAN='1' THEN
IF CNT="111" THEN
CNT<="000" ;
ELSE
CNT<=CNT+'1';
END IF;
END IF;
END PROCESS;
SELOUT<=CNT;
PROCESS(CNT) IS
BEGIN
CASE CNT IS
WHEN"000"=>TEMP1<=CONV_INTEGER(SEC)-CONV_INTEGER(SEC)/10*10;
TEMP2<=(CONV_INTEGER(SEC))/10;
WHEN"001"=>TEMP1 <= CONV_INTEGER(MIN)-CONV_INTEGER(MIN)/10*10;
TEMP2<=(CONV_INTEGER(MIN))/10;
WHEN"010"=>TEMP1<=CONV_INTEGER(HOUR)-CONV_INTEGER(HOUR)/10*10;
TEMP2<=CONV_INTEGER(HOUR)/10;
WHEN"011"=>TEMP1<=CONV_INTEGER(DAY)-CONV_INTEGER(DAY)/10*10;
TEMP2<=CONV_INTEGER(DAY)/10;
WHEN"100"=>TEMP1<=CONV_INTEGER(MON)-CONV_INTEGER(MON)/10*10;
TEMP2<=CONV_INTEGER(MON)/10;
WHEN"101"=>TEMP1<=CONV_INTEGER(YEAR)-CONV_INTEGER(YEAR)/10*10;
TEMP2<=CONV_INTEGER(YEAR)/10;
WHEN"110"=>TEMP1<=2; TEMP2<=0;
WHEN"111"=>TEMP1<=CONV_INTEGER(WEEK)-CONV_INTEGER(WEEK)/10*10;
```

```
TEMP2<=CONV_INTEGER(WEEK)/10;
WHEN OTHERS=>NULL;
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;
```

（3）--TZKZQ

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY TZKZQ IS
PORT(KEY: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
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;
BEGIN
PROCESS (KEY,CLK_KEY )
```

```
BEGIN
IF CLK_KEY'EVENT AND CLK_KEY='1'THEN
IF KEY="01" THEN
SEC_EN<='1';MIN_EN<='1';HOUR_EN<='1';
DAY_EN<='1';MON_EN<= '1';YEAR_EN<='1';
WEEK_EN<='1';
CASE MODE IS
WHEN NORMAL=> MODE<=SEC_SET;SEC<=SEC_CUR;SEC_EN<='0';
--切换到秒设置模式，读入当前秒，秒异步并行置位使能有效
WHEN SEC_SET => MODE<=MIN_SET;MIN<=MIN_CUR; SEC_EN<='1';
MIN_EN<='0';
WHEN MIN_SET => MODE<=HOUR_SET;HOUR<=HOUR_CUR;MIN_EN<='1';
HOUR_EN<='0';
WHEN HOUR_SET=>MODE<=DAY_SET;DAY<=DAY_CUR; HOUR_EN<='1';
DAY_EN<='0';
WHEN DAY_SET=>MODE<=MON_SET;MON<=MON_CUR;
DAY_EN<='1';MON_EN<='0';
WHEN MON_SET=> MODE<=YEAR_SET;YEAR<=YEAR_CUR;
MON_EN<='1';YEAR_EN<='0';
WHEN YEAR_SET =>MODE<=WEEK_SET;WEEK<=WEEK_CUR;
YEAR_EN<='1';WEEK_EN<='0';
WHEN WEEK_SET =>MODE<=NORMAL;
END CASE;
ELSIF KEY="10" THEN
CASE MODE IS
WHEN SEC_SET=> SEC_EN<='0';
IF SEC="111011"THEN SEC<="000000";
ELSE  SEC<=SEC+1;
END IF;
WHEN  MIN_SET=>MIN_EN<='0';
```

```
IF MIN="111011"THEN MIN<="000000";
ELSE MIN<=MIN+ 1;
END IF;
WHEN HOUR_SET=>HOUR_EN<='0';
IF HOUR="11000" THEN HOUR<="00000";
ELSE HOUR<=HOUR+1;
END IF;
WHEN DAY_SET=>DAY_EN<='0';
IF DAY<=MAX_DAYS THEN DAY<="00001";
ELSE DAY<=DAY+ 1;
END IF;
WHEN MON_SET=>MON_EN<='0';
IF MON="1100" THEN MON<="0001";
ELSE MON<=MON+1;
END IF;
WHEN YEAR_SET=> YEAR_EN<='0';
IF YEAR="1100011"THEN YEAR<="0000000";
ELSE YEAR<=YEAR+1;
END IF;

WHEN WEEK_SET=>WEEK_EN<='0';
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;
```

(4)--CNT7

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT7 IS
  PORT(CLK:IN STD_LOGIC;
       LD:IN STD_LOGIC;
         DATA:IN STD_LOGIC_VECTOR(2 DOWNTO 0);
        NUM:BUFFER STD_LOGIC_VECTOR(2 DOWNTO 0));
END ENTITY CNT7;
ARCHITECTURE ART OF CNT7 IS
  BEGIN
  PROCESS(CLK,LD)IS
    BEGIN
     IF (LD='0')THEN NUM<=DATA;
     ELSIF CLK'EVENT AND CLK='1' THEN
           IF NUM="111" THEN
           NUM<="000";
           ELSE NUM<=NUM+1;
         END IF;
     END IF;
  END PROCESS;
 END ARCHITECTURE ART;
```

CNT7.VHD 在程序中是实现‘计星期’功能。由下级进位信号作为该级输入信号。并有可输入输出端口实现了在调整模式中的算法‘人机交互’。实现调整功能。

由于 CNT12.VHD、CNT24.VHD 及 CNT100.VHD 在形式上与上述内容类似，分别以‘计月、计时、计年’的功能，且都能实现调整模式中的算法‘人机交互’。因此不再赘谈。

(5)--CNT60.VHD

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

```
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT60 IS
  PORT(CLK:IN STD_LOGIC;
       LD:IN STD_LOGIC;
         DATA:IN STD_LOGIC_VECTOR(5 DOWNTO 0);
        NUM:BUFFER STD_LOGIC_VECTOR(5 DOWNTO 0);
         CO:OUT STD_LOGIC);
END ENTITY CNT60;
ARCHITECTURE ART OF CNT60 IS
  BEGIN
  PROCESS(CLK,LD)IS
    BEGIN
     IF (LD='0')THEN NUM<=DATA;
     ELSIF CLK'EVENT AND CLK='1' THEN
           IF NUM="111011" THEN
           NUM<="000000";CO<='1';
           ELSE NUM<=NUM+1;CO<='0';
         END IF;
     END IF;
  END PROCESS;
 END ARCHITECTURE ART;
```

(6)--CNT30

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT30 IS
  PORT(LD:IN STD_LOGIC;
      CLK:IN STD_LOGIC;
         NIAN:IN STD_LOGIC_VECTOR(6 DOWNTO 0);
         YUE:IN STD_LOGIC_VECTOR(3 DOWNTO 0);
```

```
        DATA:IN STD_LOGIC_VECTOR(4 DOWNTO 0);
        NUM:BUFFER STD_LOGIC_VECTOR(4 DOWNTO 0);
            MAX_DAYS:OUT STD_LOGIC_VECTOR(4 DOWNTO 0);
        CO:OUT STD_LOGIC);
END ENTITY CNT30;
ARCHITECTURE ART OF CNT30 IS
  SIGNAL TOTAL_DAYS:STD_LOGIC_VECTOR(4 DOWNTO 0);
BEGIN
PROCESS(CLK,LD)IS
VARIABLE IS_RUNNIAN:STD_LOGIC;
BEGIN
CASE NIAN IS

WHEN"0000100"=>IS_RUNNIAN:='1';
WHEN"0001000"=>IS_RUNNIAN:='1';
WHEN"0001100"=>IS_RUNNIAN:='1';
WHEN"0010000"=>IS_RUNNIAN:='1';
WHEN"0010100"=>IS_RUNNIAN:='1';
WHEN"0011000"=>IS_RUNNIAN:='1';
WHEN"0011100"=>IS_RUNNIAN:='1';
WHEN"0100000"=>IS_RUNNIAN:='1';
WHEN"0100100"=>IS_RUNNIAN:='1';
WHEN"0101000"=>IS_RUNNIAN:='1';
WHEN"0101100"=>IS_RUNNIAN:='1';
WHEN"0110000"=>IS_RUNNIAN:='1';
WHEN"0110100"=>IS_RUNNIAN:='1';
WHEN"0111000"=>IS_RUNNIAN:='1';
WHEN"0111100"=>IS_RUNNIAN:='1';
WHEN"1000000"=>IS_RUNNIAN:='1';
WHEN"1000100"=>IS_RUNNIAN:='1';
```

```
WHEN"1001000"=>IS_RUNNIAN:='1';
WHEN"1001100"=>IS_RUNNIAN:='1';
WHEN"1010000"=>IS_RUNNIAN:='1';
WHEN"1010100"=>IS_RUNNIAN:='1';
WHEN"1011000"=>IS_RUNNIAN:='1';
WHEN"1100000"=>IS_RUNNIAN:='1';
WHEN OTHERS=>IS_RUNNIAN:='0';

END CASE;
CASE YUE IS
WHEN"0001"=>TOTAL_DAYS<="11111";
WHEN"0011"=>TOTAL_DAYS<="11111";
WHEN"0101"=>TOTAL_DAYS<="11111";
WHEN"0111"=>TOTAL_DAYS<="11111";
WHEN"1000"=>TOTAL_DAYS<="11111";
WHEN"1010"=>TOTAL_DAYS<="11111";
WHEN"1100"=>TOTAL_DAYS<="11111";
WHEN"0100"=>TOTAL_DAYS<="11111";
WHEN"0110"=>TOTAL_DAYS<="11111";
WHEN"1001"=>TOTAL_DAYS<="11111";
 WHEN"1011"=>TOTAL_DAYS<="11111";
 WHEN"0010"=>
              IF(IS_RUNNIAN='1') THEN
                  TOTAL_DAYS<="11101";
              ELSE
                TOTAL_DAYS<="11100";
                END IF;
                WHEN OTHERS=>NULL;
               END CASE;
              IF(LD='0')THEN
```

```
    NUM<=DATA;
ELSIF CLK'EVENT AND CLK='1' THEN
MAX_DAYS<=TOTAL_DAYS;
IF NUM=TOTAL_DAYS THEN
NUM<=NUM+1;CO<='0';
END IF;
END IF;
    END PROCESS;
END ARCHITECTURE;
```

(7)--CNT24

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT24 IS
  PORT(CLK:IN STD_LOGIC;
       LD:IN STD_LOGIC;
         DATA:IN STD_LOGIC_VECTOR(4 DOWNTO 0);
        NUM:BUFFER STD_LOGIC_VECTOR(4 DOWNTO 0);
         CO:OUT STD_LOGIC);
END ENTITY CNT24;
ARCHITECTURE ART OF CNT24 IS
  BEGIN
  PROCESS(CLK,LD)IS
    BEGIN
     IF (LD='0')THEN NUM<=DATA;
     ELSIF CLK'EVENT AND CLK='1' THEN
           IF NUM="10111" THEN
           NUM<="00000";CO<='1';
           ELSE NUM<=NUM+1;CO<='0';
        END IF;
```

```
    END IF;
  END PROCESS;
 END ARCHITECTURE ART;
```

(8)--CNT12

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT12 IS
  PORT(CLK:IN STD_LOGIC;
       LD:IN STD_LOGIC;
         DATA:IN STD_LOGIC_VECTOR(3 DOWNTO 0);
        NUM:BUFFER STD_LOGIC_VECTOR(3 DOWNTO 0);
         CO:OUT STD_LOGIC);
END ENTITY CNT12;
ARCHITECTURE ART OF CNT12 IS
  BEGIN
  PROCESS(CLK,LD)IS
    BEGIN
     IF (LD='0')THEN NUM<=DATA;
     ELSIF CLK'EVENT AND CLK='1' THEN
           IF NUM="1011" THEN
           NUM<="0000";CO<='1';
           ELSE NUM<=NUM+1;CO<='0';
         END IF;
     END IF;
  END PROCESS;
 END ARCHITECTURE ART;
```

(9) --CNT100

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

```
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CNT100 IS
  PORT(CLK:IN STD_LOGIC;
       LD:IN STD_LOGIC;
         DATA:IN STD_LOGIC_VECTOR(6 DOWNTO 0);
        NUM:BUFFER STD_LOGIC_VECTOR(6 DOWNTO 0));
END ENTITY CNT100;
ARCHITECTURE ART OF CNT100 IS
  BEGIN
  PROCESS(CLK,LD)IS
    BEGIN
     IF (LD='0')THEN NUM<=DATA;
     ELSIF CLK'EVENT AND CLK='1' THEN
          IF NUM="1100100" THEN
          NUM<="0000000";
          ELSE NUM<=NUM+1;
        END IF;
     END IF;
  END PROCESS;
 END ARCHITECTURE ART;
```

(10)--YMQ38.VHD

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;
ENTITY YMQ38 IS
PORT(DIN:IN STD_LOGIC_VECTOR(2 DOWNTO 0);
DOUT :OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END ENTITY YMQ38;
ARCHITECTURE ART OF YMQ38 IS
BEGIN
```

```
PROCESS(DIN)
BEGIN
CASE DIN IS
WHEN"000"=>DOUT<="11111110";
WHEN"001"=>DOUT<="11111101";
WHEN"010"=>DOUT<="11111011";
WHEN"011"=>DOUT<="11110111";
WHEN"100"=>DOUT<="11101111";
WHEN"101"=>DOUT<="11011111";
WHEN"110"=>DOUT<="10111111";
WHEN"111"=>DOUT<="01111111";
WHEN OTHERS=>DOUT<="00000000";
END CASE;
END PROCESS;
END ARCHITECTURE ART;
```

YMQ38 实现的是显示调整模式的发光二极管亮灭逻辑，通过该源程序，我们能更为直观的选择、调整计时系统中相应的模式。

3. 时序仿真与分析

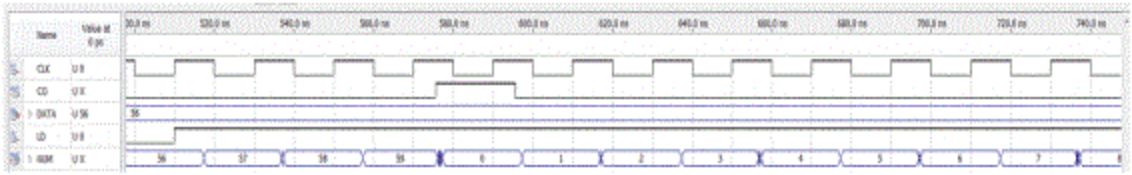


图 1-1

从图 1 可以看出,CNT60 实现了 0 到 59 的循环计数，每实现一次 59 到 0 的计数动作，计数器模块输出一个进位信号。当 LD 端有低电平输入时，说明置数信号（LD）有效，模块将预置数（DATA）56 送入计数结果中去，计数模块从 56 开始重新计数。

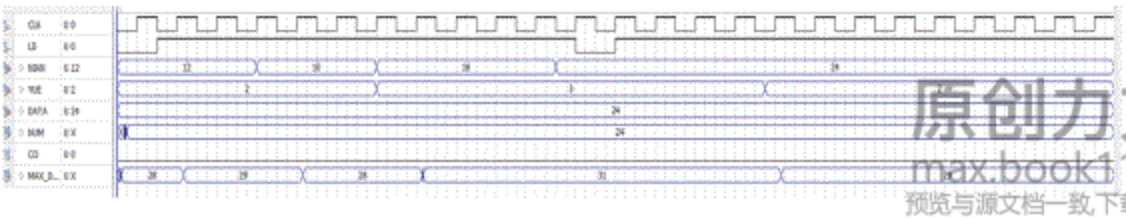


图 1-2

从图 2 可以看出，该模块首先要读取当前年月，在对该月的最大天数进行判断并将结果向外输出。在正常的计数过程中，模块实现了从 0 到最大天数的循环计数

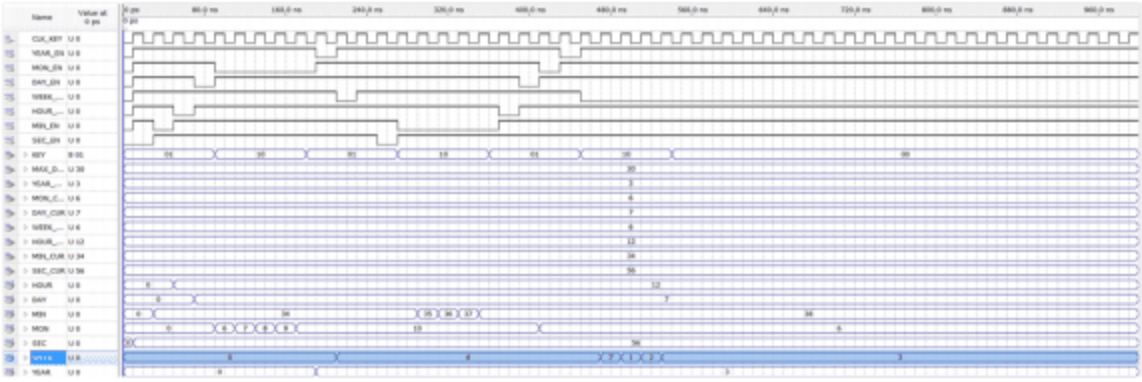


图 1-3

从图 3 可以看出，调整键被按下了 3 次(键盘产生“01”信号即为调整键按下一次)，调分、调时、调日、调月、调年、调星期再回到正常这调整模式依次经过了正常、调秒、3 种模式的循环，即 MODE 依次从 0 到 2,再从 2 回到 0 的循环。在按键过程中，每按下次按钮，相应的被调整模式的异步并行置位使能置“0”，同时通过一个非门点亮该调整模式所对应的发光二极管，作为该调整模式的指示信号。图中，调整前时间为 2003 年 6 月 7 日 12:34: 56 星期 6,在按动了两次模式键(01)后，调整模式切换到调分模式，然后又按动了两次调整键(10),实现了在调分模式下对当前分的值进行调整(从 34 调到 38),调整后时间为 2003 年 6 月 7 日 12:38:56 星期 3。

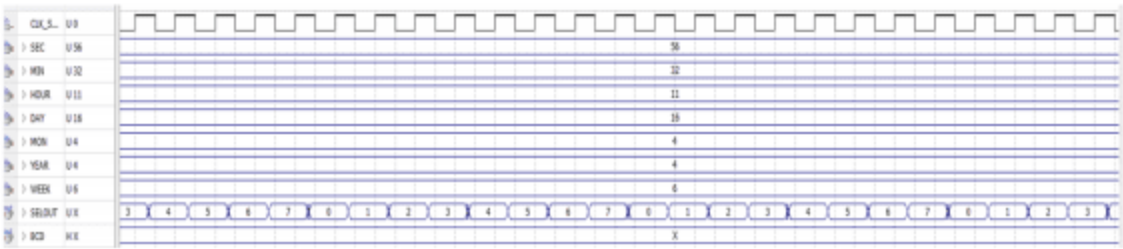


图 1-4

从图 4 可以看出，当 SELOUT 分别等于 0,1,2,3,4,5,6,7 时，分别选择对应的输入数据输出，达到了设计要求。

4. 逻辑综合与分析

综合逻辑组合图如图：（由于总图太大，无法一次截屏，因此分割截图。）

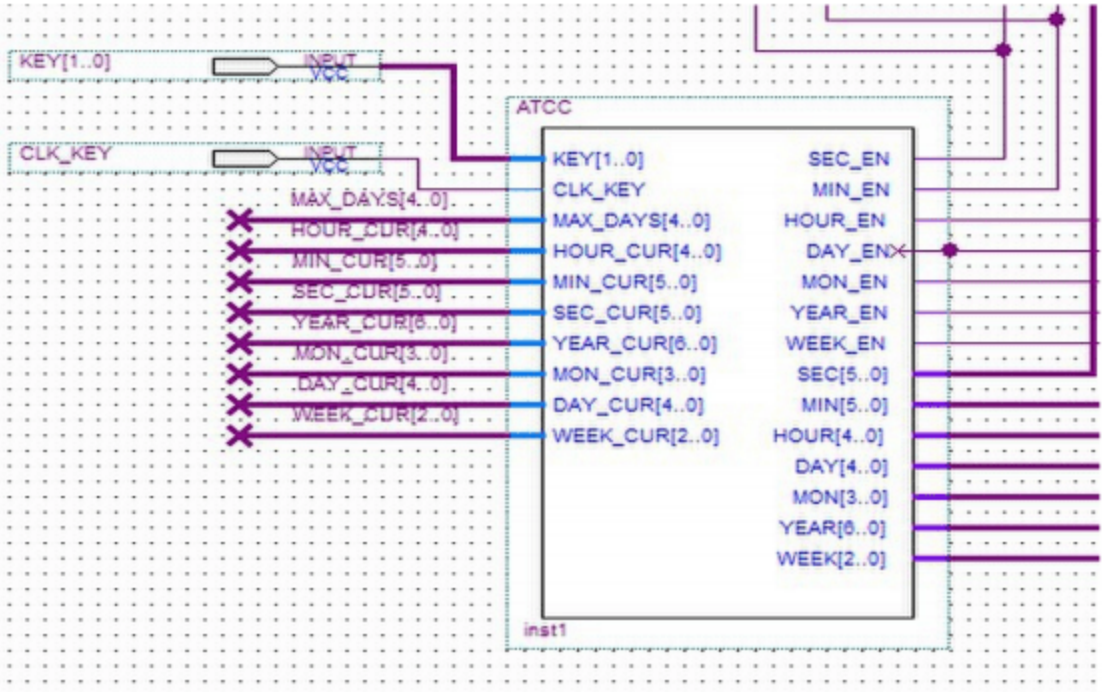


图 1-5

该图为调整控制电路(ATCC)，通过 KEY 进行调整模式的选择；CLK_KEY 进行具体调节相应的秒、分、时、日、月、年等模式。

下图为整个计时系统的运算部分。整体由上一级进位信号作为下一级的是输入信号。实现相关的关联。由于界面过小，因此无法到完整的计数部分的原理图。

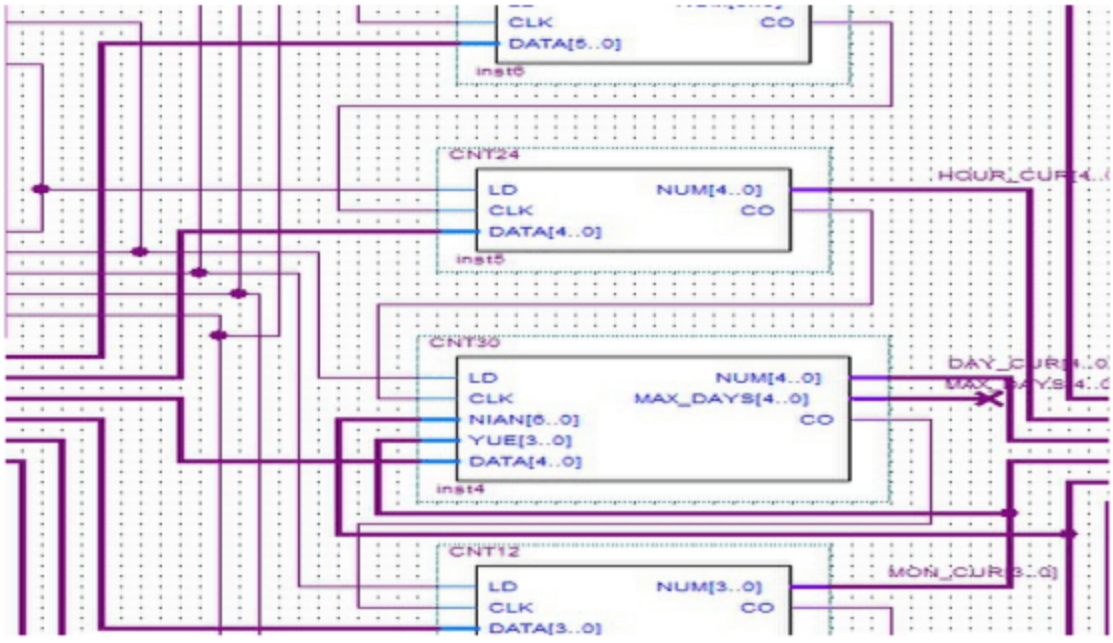


图 1-6

下图是整体电路图中的显示控制部分及译码器部分。该部分实现的是相关的显示在系统功能面板上的一系列 LED 显示管。

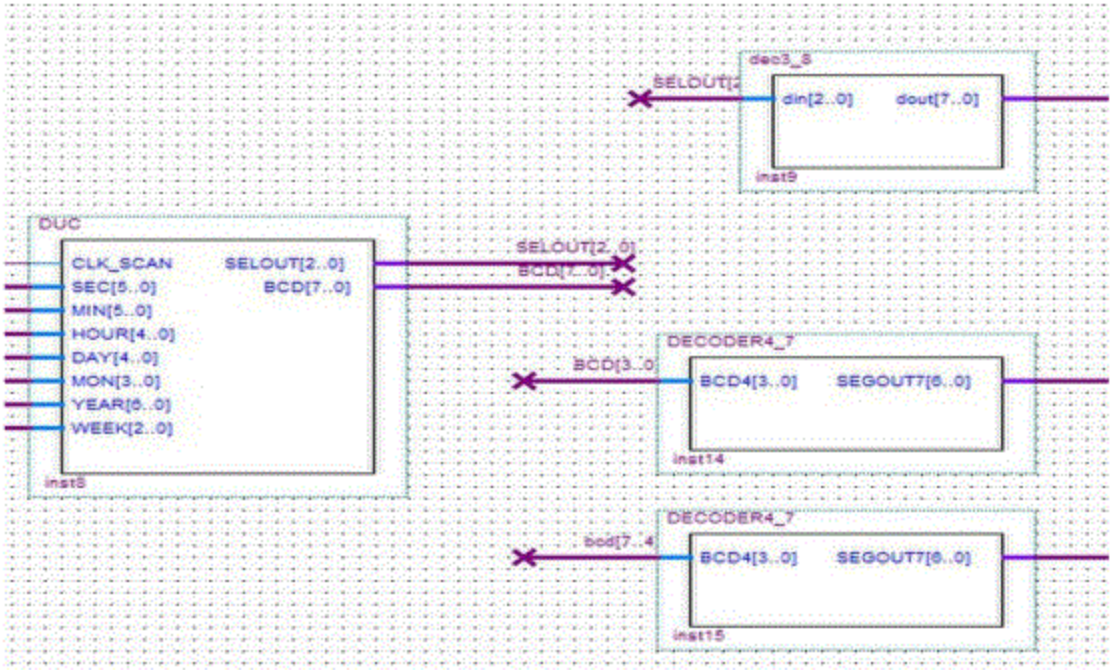


图 1-7

最后就是控制显示面板上的模式选择发光二极管部分的连线。该部分的输入线连接的是调整控制电路的相应 SEC_EN、MIN_IN 等端口。表明控制调整模式的选择。

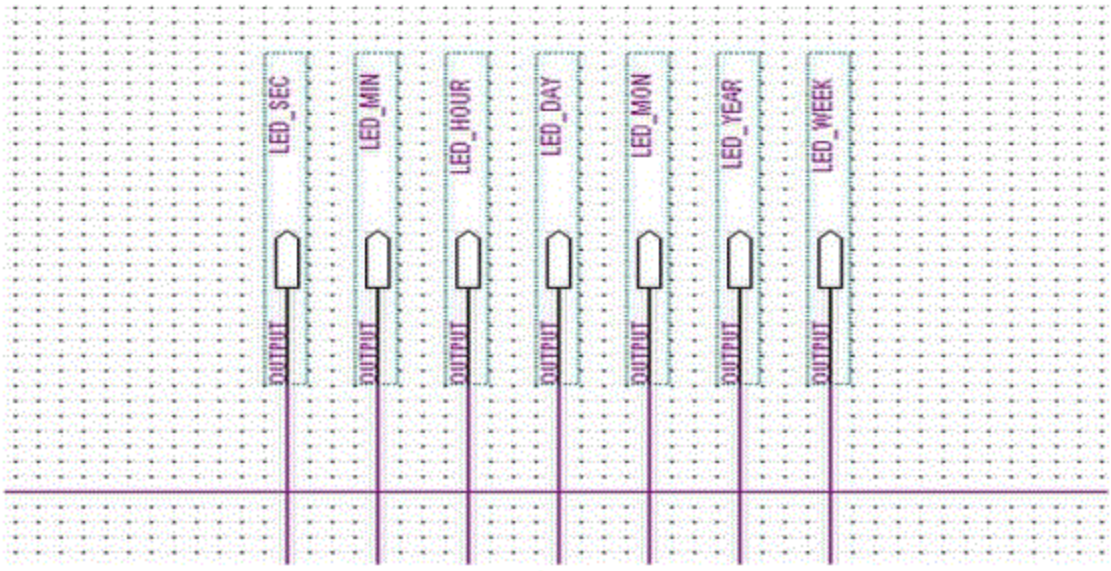


图 1-8

最后是整体波形的仿真图：

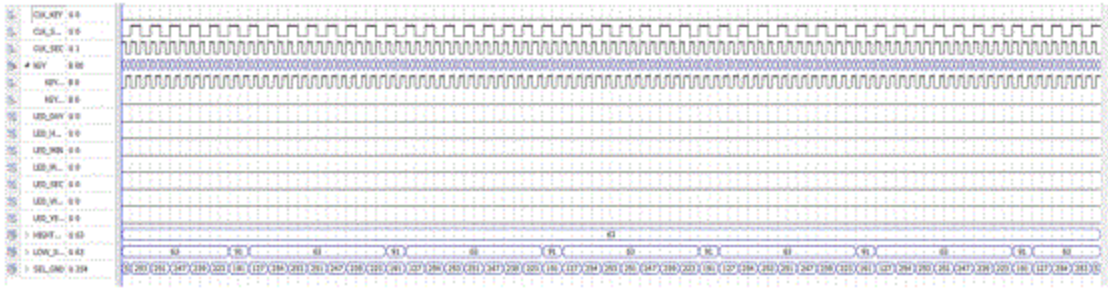


图 1-9

由于相关的波形的设置较为复杂，就简略的设置了一下相应的波形，确保了输出显示。然而在模式选择的输出端却有所欠缺，由于时间较为紧迫，因此没有具体去优化该输入信号的给与。

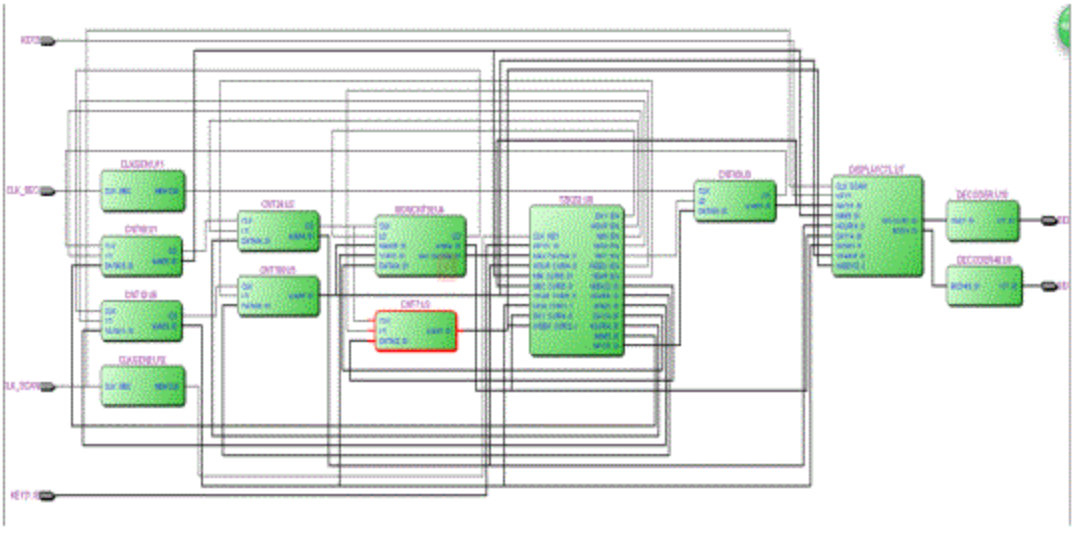


图 1-10　CLOCK 综合后的 RTL 视

Flow Summary	
Flow Status	Successful - Sat Dec 23 15:46:06 2017
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	CLOCK
Top-level Entity Name	CLOCK
Family	Cyclone III
Total logic elements	535 / 5,136 (10 %)
Total combinational functions	523 / 5,136 (10 %)
Dedicated logic registers	122 / 5,136 (2 %)
Total registers	122
Total pins	21 / 183 (11 %)
Total virtual pins	0
Total memory bits	0 / 423,936 (0 %)
Embedded Multiplier 9-bit elements	0 / 46 (0 %)
Total PLLs	0 / 2 (0 %)
Device	EP3C5F256C6
Timing Models	Final

图 1-11 CLOCK 逻辑综合后的资源使用情况

5. 硬件验证及结果

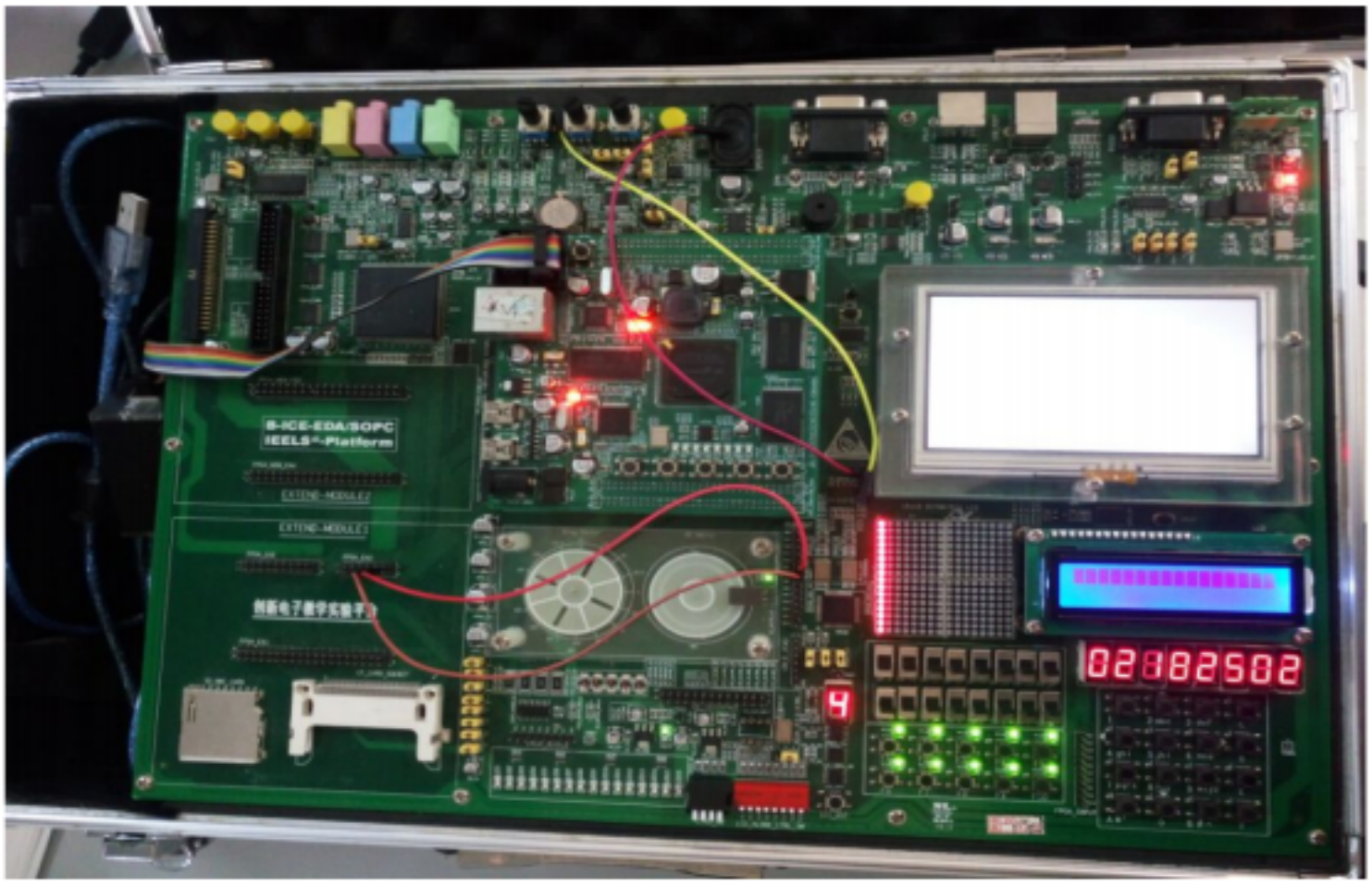


图 1-12　显示周时分秒

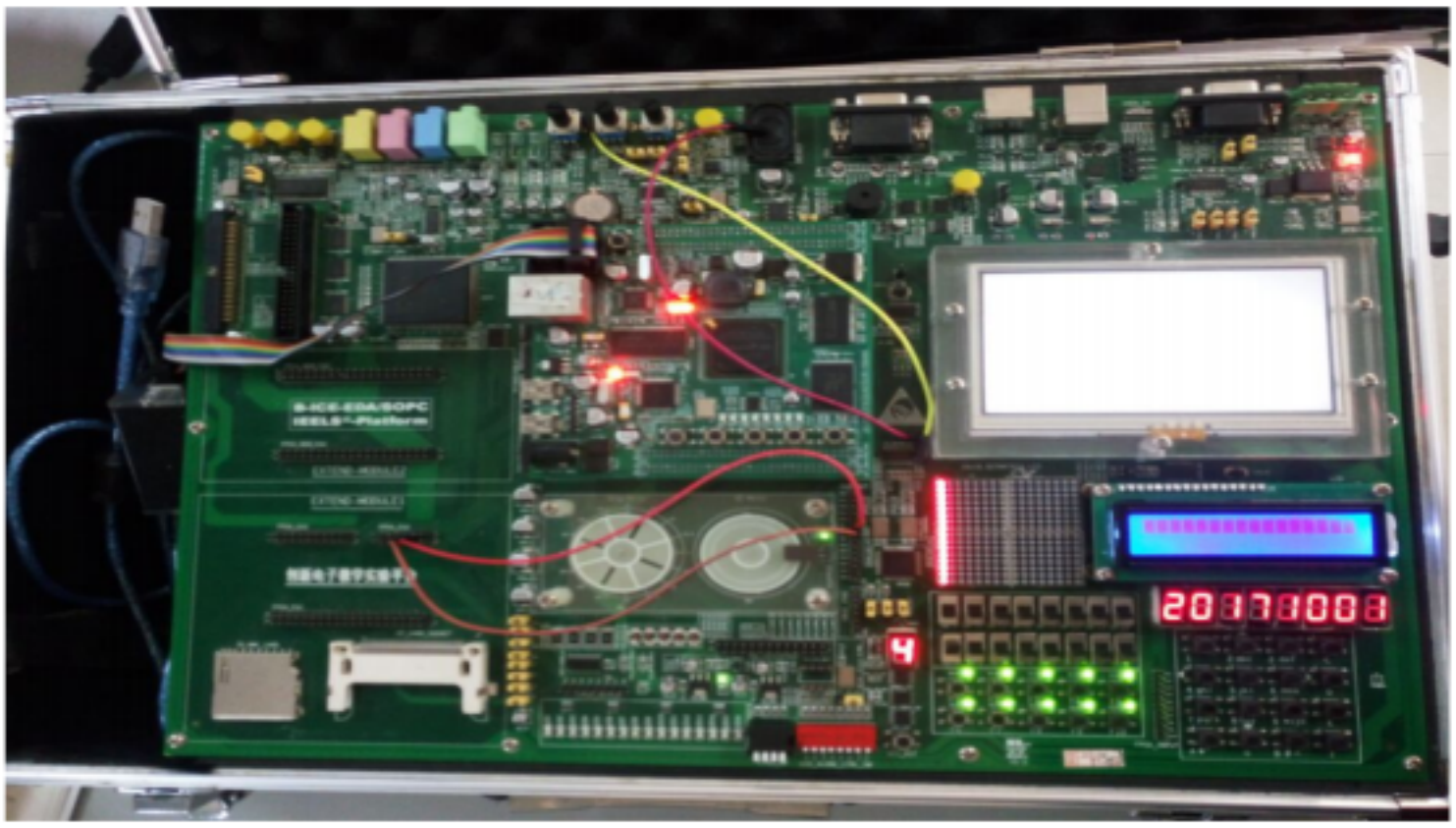


图 1-13　显示年月天

6. 设计收获与体会

对于这次的综合计时器的设计，加深了我对 eda 两种设计的体会，之前一直用 VHDL 语言去编写，没用过图表设计，这一次通过画图的方法设计了顶端程序。

二、SOBEL 图像边缘检测器的设计

1. 系统设计原理

该系统的主要设计原理是采用现代电子设计的最新技术——EDA 技术，使用高速可编程逻辑器件 PFGA/CPLD 自行开发有关处理芯片成了一种全新的解决方案。如下组成框图所示：

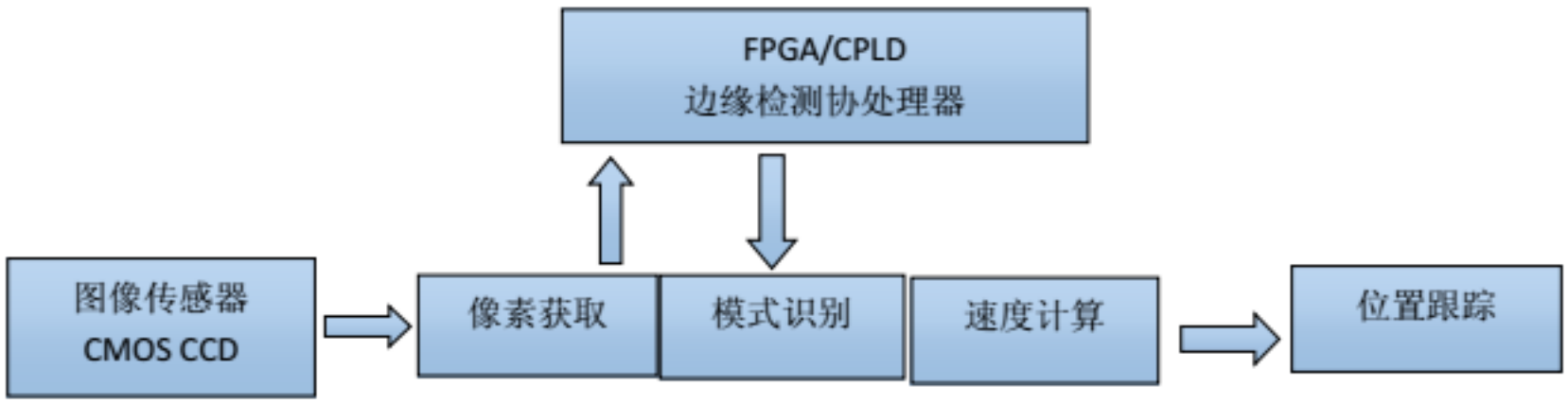


图 2-1 DSP+FPGA/CPLD 图像主处理器

其中图像传感器 CCD 的功能是获取外界图像的各个像素点灰度值；图像主处理器采用数字信号处理器 DSP，主要负责对图像传感器传送的灰度信息进行存储，并负责调用协处理器进行边界像素判别，找出我们感兴趣的目标对象，从而得到该对象的运动信息，以便控制执行装置进行位置跟踪；边缘检测协处理器为 FPGA/CPLD，主要是完成主处理器传送过来的像素的边界判别，并把处理结果返回到主处理器中。

在本系统中，各项设计指标为：数据吞吐量>10Mb/s;动态响应时间<100ms/frame。主处理器初步选用德州公司的 DSP 芯片 TMS320C5402，协处理器拟采用 ALTERA 公司的 FLEX10K20。图像处理系统接口关系如下图所示：

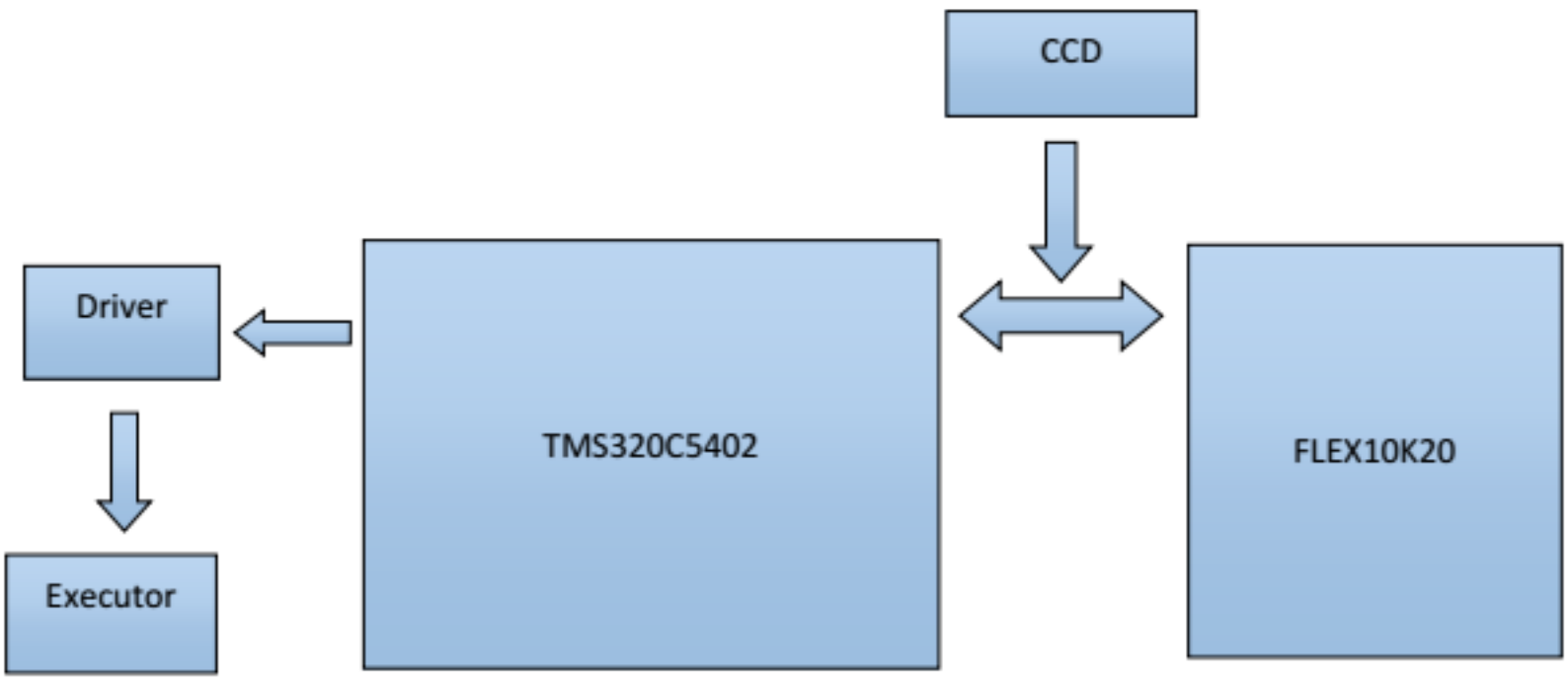


图 2-2

边缘可定义为图像中灰度发生急剧变化的区域边界,它是图像最基本的特征，是图像分析识别前必不可少的环节,是一种重要的图像预处理技术。边缘检测主要就是(图像的)灰度

变化的度量、检测和定位,它是图像分析和模式识别的主要特征提取手段，它在计算机视觉、图像分析等应用中起着重要的作用，是图像分析与处理中研究的热点问题。在过去的 20 年里产生了许多边缘检测器，如 Rorberts 算子，Sobel 算子，Prewitt 算子，Laplacian 算子等。由于 Sobel 算法只涉及加法操作，但却可以得到很好的划分效果，因而是图像处理系统中最常用的边缘检测算法。

Sobel 算法包括带 4 个 3×3 掩码的输入图像数据，即 Sobel 算子，它设置权重来检测水平、垂直、左对角、右对角各个不同方向上密度幅度的不同。这个过程通常被称为过滤。我们来看像素窗口（3×3），如图 1 所示。水平、垂直、左对角、右对角各图像方向上密度幅度的变化可以用如下算子进行计算：

H=(Q0+2Q3+Q6)-(Q2+2Q5+Q8);V=(Q0+2Q1+Q2)-(Q6+2Q7+Q8)；

DR=(Q1+2Q0+Q3)-(Q5+2Q8+Q7);DL=(Q1+2Q2+Q5)-(Q3+2Q6+Q7)；

H,V,DL,DR 这四个参数用于计算梯度大小和方向。

Q0	Q3	Q6
Q1	[i,j]	Q7
Q2	Q5	Q8

图 2-3 像素框

我们通过对图像灰度作直方图分析后，便可以给出区分度阀值 Threshold，区分度阀值往往要借助一定的经验并需要反复调整。如果 Magnitude 大于 Threshold，则该像素被声明为边界像素，否则为一般像素。

本课题就是要求使用 LPM 兆功能块设计和 VHDL 程序设计相结合的方式或全部采用 VHDL 程序设计方式，用 FPGA/CPLD 实现 Sobel 算法。

运用 CPLD/FPGA 设计有关图像处理模块，对象系统速度改善是非常明显：处理一帧图像的时间为 800*600*80ns=38.4ms，结果处理速度比 DSP 高了约两个数量级。

2. VHDL 程序或原理图设计

（1）CSFIFO.VHD

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY CSFIFO IS
GENERIC(WIDTH:INTEGER:=8;
        DEPTH:INTEGER:=8;
          ADDR:INTEGER:=3);
```

```
PORT(DATAIN:IN STD_LOGIC_VECTOR(WIDTH-1 DOWNTO 0);
DATAOUT:OUT STD_LOGIC_VECTOR(WIDTH-1 DOWNTO 0);
CLOCK:IN STD_LOGIC;
WE,RE:IN STD_LOGIC;
FULL,EMPTY:OUT STD_LOGIC);
END ENTITY CSFIFO;
ARCHITECTURE ART OF CSFIFO IS
TYPE MEM IS ARRAY(DEPTH-1 DOWNTO 0) OF STD_LOGIC_VECTOR(WIDTH-1 DOWNTO 0);
SIGNAL RAMTMP: MEM;
SIGNAL WADD:STD_LOGIC_VECTOR(ADDR-1 DOWNTO 0);
SIGNAL RADD:STD_LOGIC_VECTOR(ADDR-1 DOWNTO 0);
SIGNAL W,W1,R,R1:INTEGER RANGE 0 TO 8;
BEGIN
WRITE_POINTER:PROCESS(CLOCK) IS
BEGIN
IF(RISING_EDGE(CLOCK)) THEN
IF(WE='1') THEN
IF(WADD=7)THEN WADD<=(OTHERS=>'0');
ELSE WADD<=WADD+'1';END IF;
END IF;
END IF;
W<=CONV_INTEGER(WADD);
W1<=W-1;
END PROCESS WRITE_POINTER;
WRITE_RAM:PROCESS(CLOCK) IS
BEGIN
IF(RISING_EDGE(CLOCK)) THEN
IF(WE='1')THEN
RAMTMP(CONV_INTEGER(WADD))<=DATAIN;
END IF;
```

```
END IF;
END PROCESS WRITE_RAM;
READ_POINTER:PROCESS(CLOCK) IS
BEGIN
IF(RISING_EDGE(CLOCK)) THEN
IF(RE='1')THEN
IF(RADD=7)THEN RADD<=(OTHERS=>'0');
ELSE RADD<=RADD+'1';END IF;
END IF;
END IF;
R<=CONV_INTEGER(RADD);
R1<=R-1;
END PROCESS READ_POINTER;
READ_RAM:PROCESS(CLOCK) IS
BEGIN
IF(RISING_EDGE(CLOCK)) THEN
IF(RE='1') THEN
DATAOUT<=RAMTMP(CONV_INTEGER(RADD));
END IF;END IF;
END PROCESS READ_RAM;
FULLFLAG:PROCESS(CLOCK) IS
BEGIN
IF(RISING_EDGE(CLOCK)) THEN
IF(WE='1'AND RE='0')THEN
IF(W=R1)OR((WADD=CONV_STD_LOGIC_VECTOR(DEPTH-1,3))
AND(RADD="000")) THEN
FULL<='1';END IF;
ELSE FULL<='0';END IF;
END IF;
END PROCESS FULLFLAG;
```

```
EMPTYFLAG:PROCESS(CLOCK) IS
BEGIN
IF(RISING_EDGE(CLOCK)) THEN
IF(RE='1'AND WE='0') THEN
IF(R=W1)OR((RADD=CONV_STD_LOGIC_VECTOR(DEPTH-1,3))
AND(WADD="000"))THEN
EMPTY<='1';END IF;
ELSE EMPTY<='0';END IF;
END IF;
END PROCESS EMPTYFLAG;
END ARCHITECTURE ART;
```

（2）--COUNTER

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY COUNTER IS
PORT(CLK:IN STD_LOGIC;
CO:OUT STD_LOGIC);
END ENTITY COUNTER;
ARCHITECTURE ART OF COUNTER IS
SIGNAL CNT:NATURAL RANGE 0 TO 2;
BEGIN
PROCESS (CLK) IS
BEGIN
IF CLK'EVENT AND CLK='1' THEN
IF CNT=1 THEN
CNT<=0;
CO<='1';
ELSE
  CNT<=CNT+1;
  CO<='0';
```

```
  END IF;
  END IF;
  END PROCESS;
  END ARCHITECTURE ART;
```

(3)-COMPARE

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY COMPARE IS
GENERIC(SIZE:INTEGER:=10);
PORT(DAT1,DAT2:IN STD_LOGIC_VECTOR(SIZE DOWNTO 0);
MAX:OUT STD_LOGIC_VECTOR(SIZE DOWNTO 0));
END ENTITY COMPARE;
ARCHITECTURE ART OF COMPARE IS
BEGIN
PROCESS(DAT1,DAT2)
BEGIN
IF(DAT1(SIZE-1 DOWNTO 0)>DAT2(SIZE-1 DOWNTO 0))THEN
MAX<=DAT1;
ELSE MAX<=DAT2;
END IF;
END PROCESS;
END ARCHITECTURE ART;
```

(4) --RESULT

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY RESULT IS
GENERIC(SIZE:INTEGER:=10);
PORT(CLK:IN STD_LOGIC;
H,V,DR,DL:IN STD_LOGIC_VECTOR(SIZE DOWNTO 0);
```

```
THRESHOLD:IN STD_LOGIC_VECTOR(7 DOWNTO 0);
MAX:IN STD_LOGIC_VECTOR(SIZE DOWNTO 0);
MAGOUT:OUT STD_LOGIC);
END ENTITY RESULT;
ARCHITECTURE ART OF RESULT IS
SIGNAL DIR: STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN
PROCESS(CLK,THRESHOLD)
BEGIN
IF(CLK'EVENT AND CLK='1')THEN
IF(MAX(SIZE-1 DOWNTO 0)>"00"&THRESHOLD)THEN MAGOUT<='1';
IF(MAX=H AND H(SIZE)='0')THEN DIR<="000";
ELSIF(MAX=H AND H(SIZE)='1')THEN DIR<="100";
ELSIF(MAX=V AND V(SIZE)='0')THEN DIR<="010";
ELSIF(MAX=V AND V(SIZE)='1')THEN DIR<="110";
ELSIF(MAX=DR AND DR(SIZE)='0')THEN DIR<="001";
ELSIF(MAX=DR AND DR(SIZE)='1')THEN DIR<="101";
ELSIF(MAX=DL AND DL(SIZE)='0')THEN DIR<="011";
ELSIF(MAX=DL AND DL(SIZE)='1')THEN DIR<="111";
END IF;
ELSE MAGOUT<='0';DIR<="000";
END IF;
END IF;
END PROCESS;
END ARCHITECTURE ART;
```

(5) --REG

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY REG IS
PORT(CLK:IN STD_LOGIC;
```

```
D:IN STD_LOGIC_VECTOR(7 DOWNTO 0);
Q:OUT STD_LOGIC_VECTOR(7 DOWNTO 0));
END ENTITY REG;
ARCHITECTURE ART OF REG IS
BEGIN
PROCESS(CLK,D)IS
BEGIN
IF(CLK'EVENT AND CLK='1')THEN
Q<=D;
END IF;
END PROCESS;
END ARCHITECTURE;
(6)--CNT3
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY CNT3 IS
PORT(CLK:IN STD_LOGIC;
 CO:OUT STD_LOGIC);
 END ENTITY CNT3;
 ARCHITECTURE ART OF CNT3 IS
 SIGNAL CP:NATURAL RANGE 0 TO 3;
 BEGIN
 PROCESS(CLK) IS
 BEGIN
    IF(CLK'EVENT AND CLK='1') THEN
     IF CP=2 THEN CP<=0; CO<='1';
     ELSE CP<=CP+1; CO<='0';
     END IF;
     END IF;
     END PROCESS;
```

END ARCHITECTURE ART;

（7）FIFO 模块的原理图：

FIFO 模块是由定制的 LMP 模块 CSFIFO、LPM_FF，标准元件 AND2、NOT 以及用 VHDL 编程编译后生成的二进制计数器 COUNTER2 构成。如下图所示：

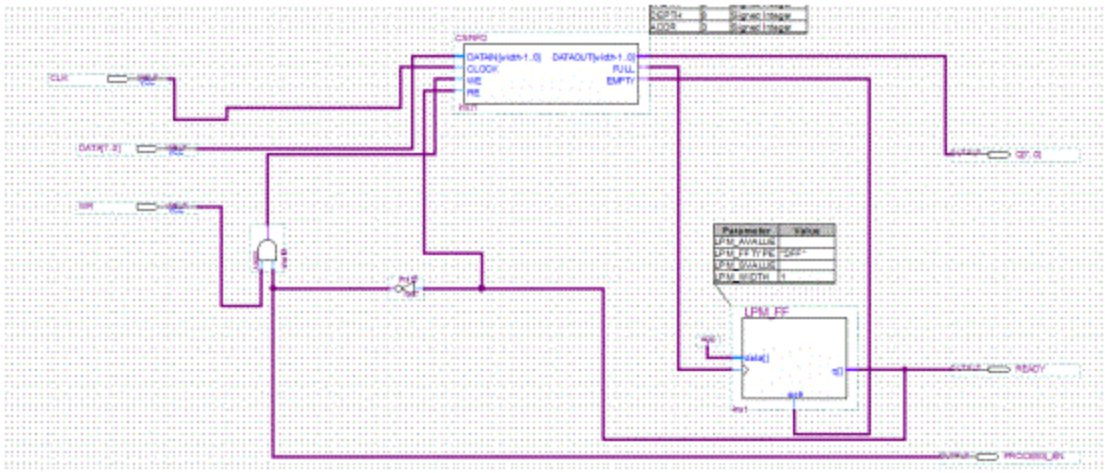


图 2-4

（8）SIPO 的 LPM 原理图：

如图 2-5

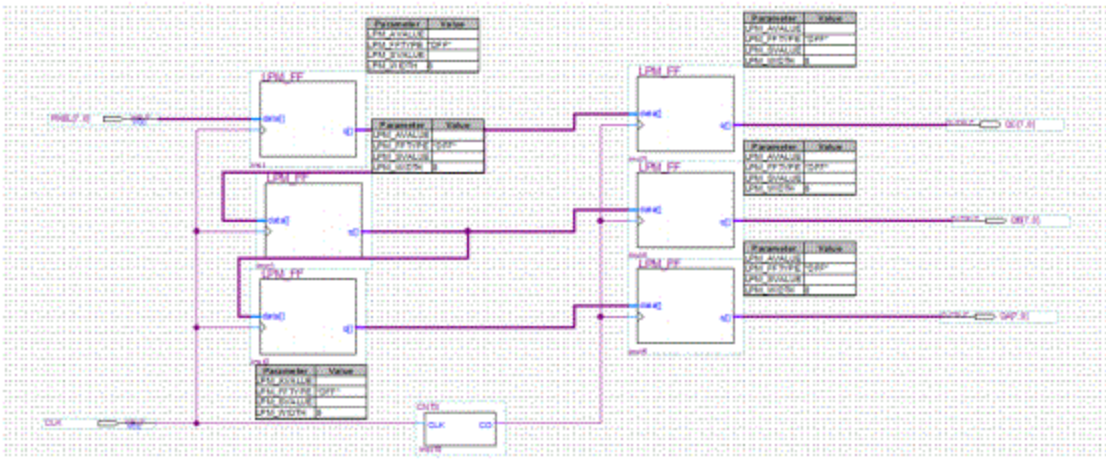
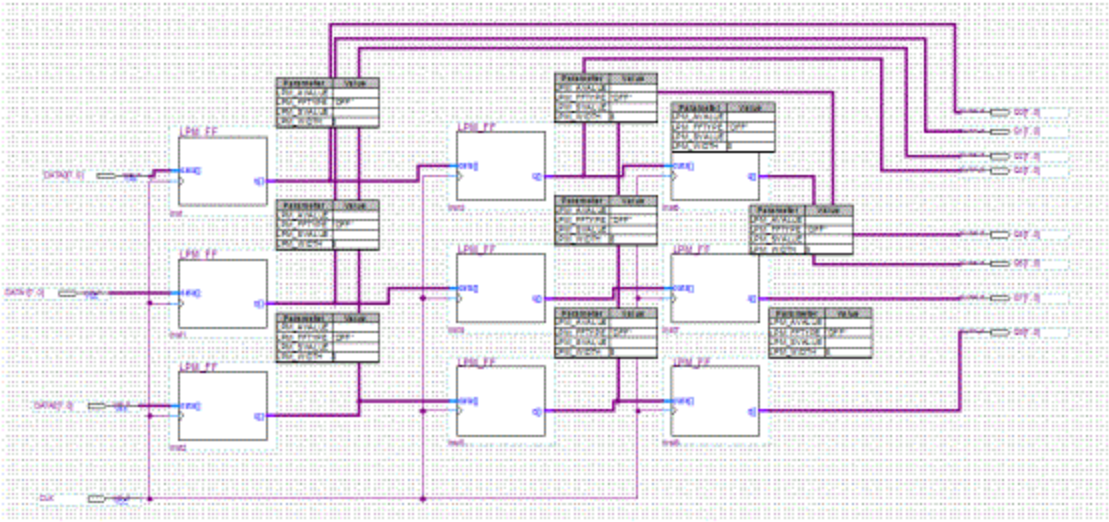


图 2-5

（9） REFRESH 的 LPM 原理图：

如图 2-6



如图 2-6

（10）PROCESS 的 LPM 原理图：

如图 2-7

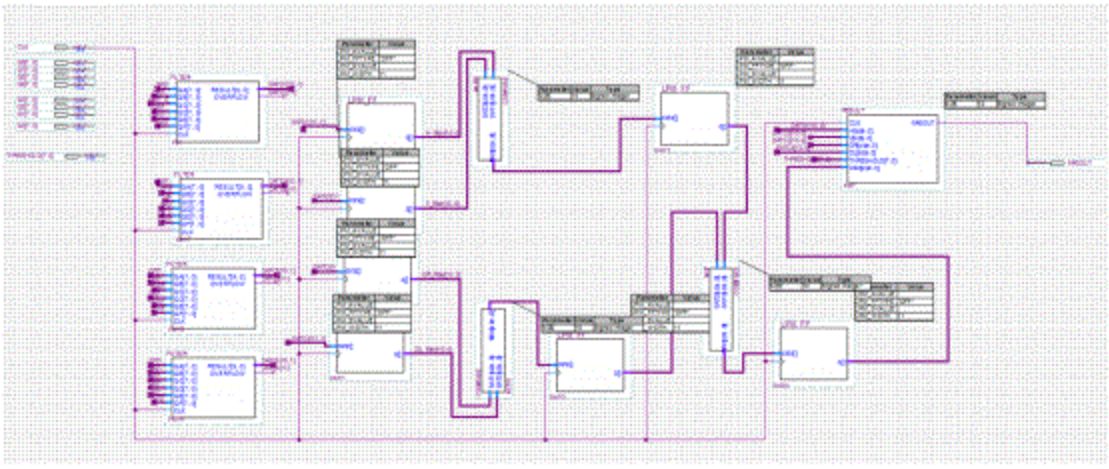


图 2-7

（11）FILTER 的 LPM 原理图：

如图 2-8

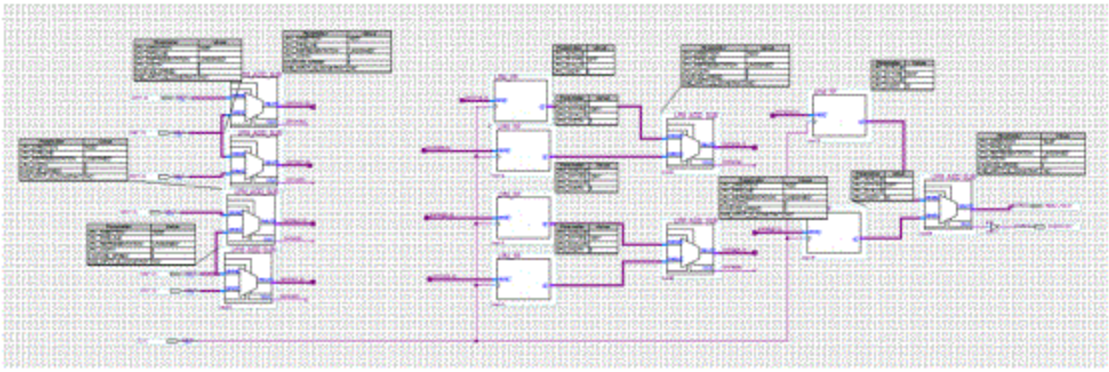


图 2-8

3. 时序仿真与分析

（1）COMPARE

图 2-9

COMPARE 是 PROCESSOR 里的一个简单的数据比较器。通过进行数据比较后选择较大的数据输出。

（2）FIFO

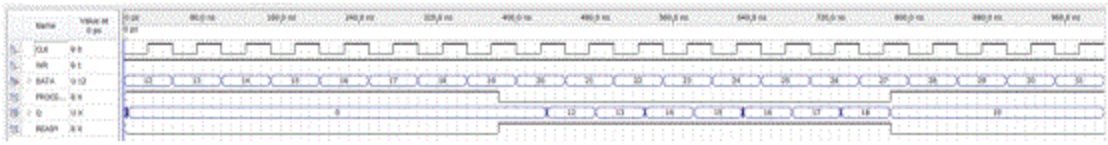


图 2-10

这是帧窗口接收模块 FIFO 的仿真图形。它选择的是帧同步方式。堆栈空时向主机发出准备好信号，主机检测到它的数据传输请求时，传送一帧数据。当接收完毕后，堆栈满，sobel 处理器启动边缘检测进程，处理完一帧数据后，堆栈重新变为空，为下一帧数据处理做准备。

（3）FILTER

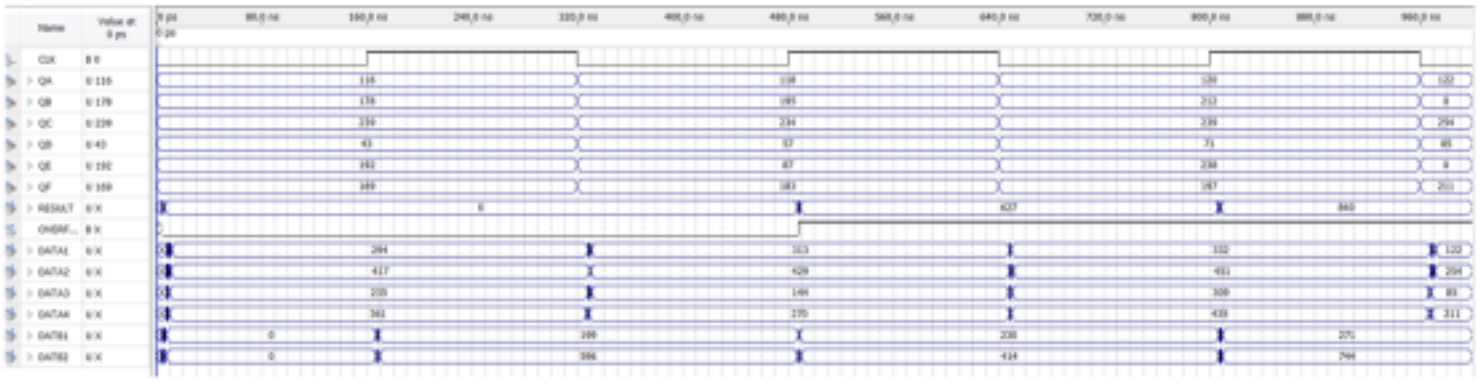


图 2-11

从这个仿真图，结合原有的方案，我们可以知道这是个滤波模块。在本次设计中的加法器采用的是超前进位加法器，提高了加法运算速度，运用了流水线技术，虽然滞后了三个时钟频率，但增加了数据的吞吐量，同时也提高了时钟频率。

(4) PROCESS

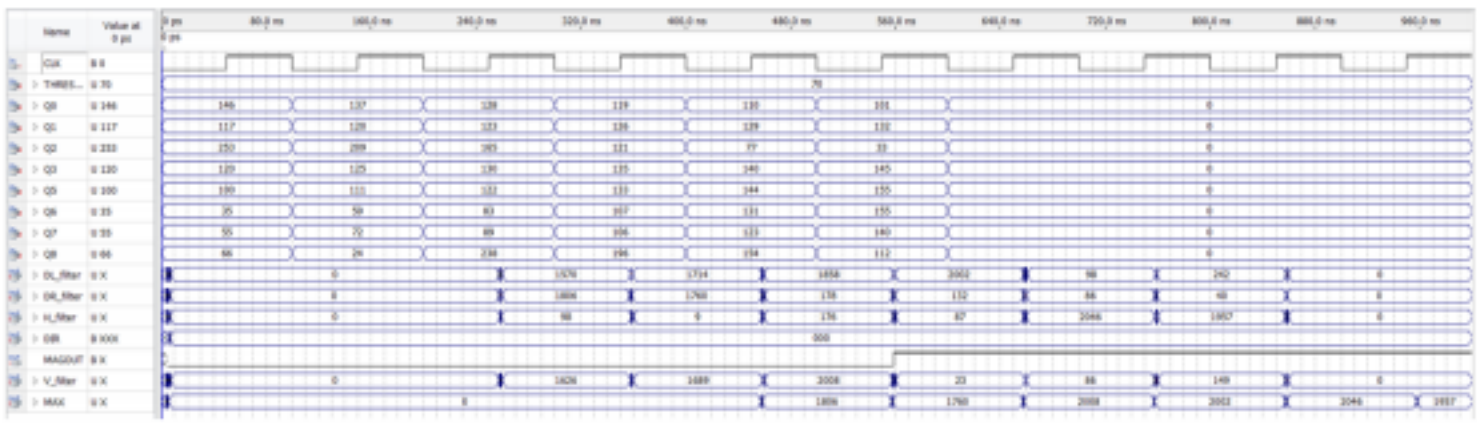


图 2-12

这是数据处理模块的仿真图，是整个边缘处理器的核心部分，实现 Sobel 算法。有五级串行结构，判别速度快。

(5) REFRESH

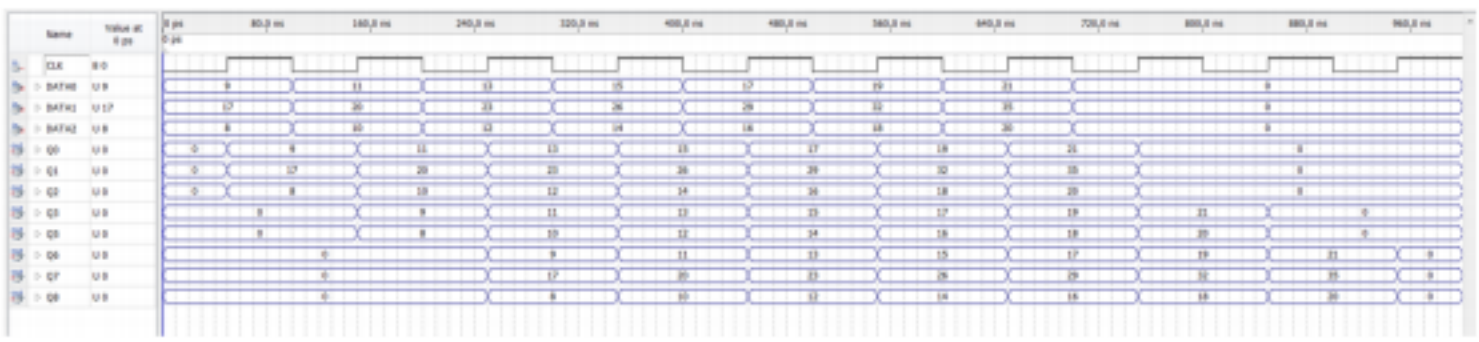


图 2-13

REFRESH 的时序仿真可以看出这是个像素刷新窗口，在接收串入并出模块的三个并行像素，把窗口中原有的第二列像素推入第三列，第一列推入第二列，新到的并行像素填入第一列。其本质是一个位移寄存器。

(6) RESULT

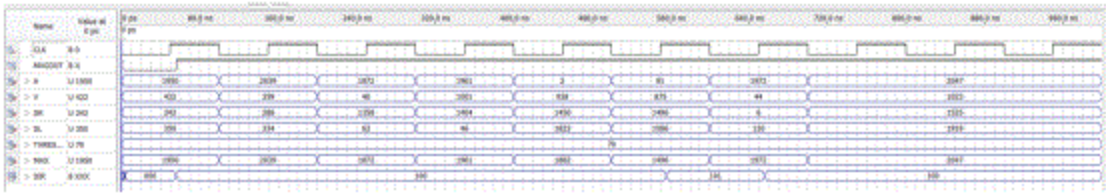


图 2-14

-边界判断器 RESULT 模块是数据处理模块 PROCESS 内部的一个子模块，其功能是根据区分度阀值、四个滤波器的输出及其最大值进行边界判断。

(7) SIPO

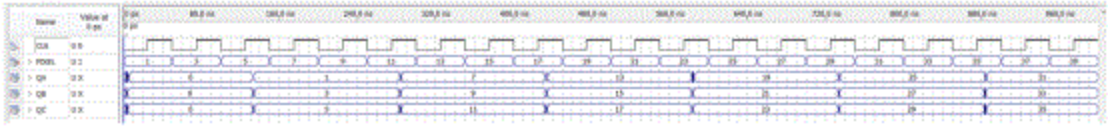


图 2-15

这是传入并出模块的波形仿真图，主要是负责把主处理器传送过来的像素值换成 3*3 像素窗口的一列。该模块主要是由六个 D 触发器和一个三进制计数器组成。

4. 逻辑综合与分析

最后的图像边缘检测器总体结构图如图所示：

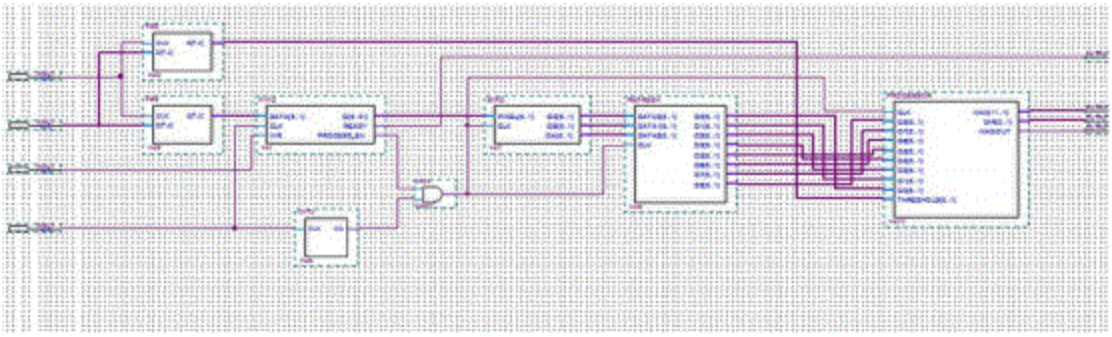


图 2-16

如图所示:整个系统的实现划分为四个大的模块，其总体结构如图所示：其中：帧窗口接收模块（FIFO)负责接收 DSP 传送过来的一个帧窗口的数据，其本质为一个双端口先入先出的出栈 FIFO，其数据宽度为 8，深度等于一个帧窗口内的像素点个数（600*3=1800)。串入并出模块(SIPO)负责把 FIFO 内的数据转换成为像素处理窗口的列像素向量，便于像素处理窗口的数据刷新处理。像素窗口刷新模块（REFRESH）实现对需要处理的像素数据的刷新。数

据处理模块（PROCESSOR)是本图形边缘处理器的核心部分，主要是实现 Sobel 算法，其性能的好坏对整个设计的成败有着关键的作用。本模块拟采用全硬件并行算法实现，因只有五级串行结构，所有相当于 5 个时钟周期内就能完成一个像素点的边界判别。

5. 设计收获与体会

这次课程设计历时两个星期，在这个两个星期里学到了许多，进一步加深了对 EDA 技术的学习。虽然图像边缘检测器，我只做了 EDA 部分，但是这部分却是最节省系统时间的部分，

有点遗憾没有完成硬件部分。最后，对给过我帮助的所有同学和各位指导老师再次表示忠心的感谢！