太原科技大學

电子课程设计

——简易洗衣机控制器设计

学院：_____

班级：_____

姓名：_____

学号：_____

指导老师：_____

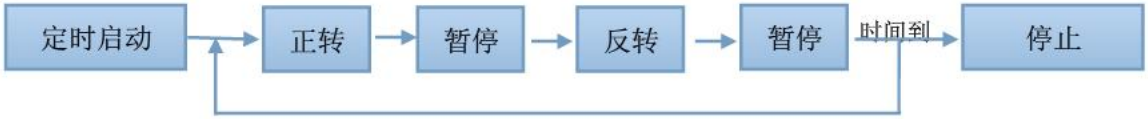
2013 年 12 月

目　录

简易洗衣机控制器设计

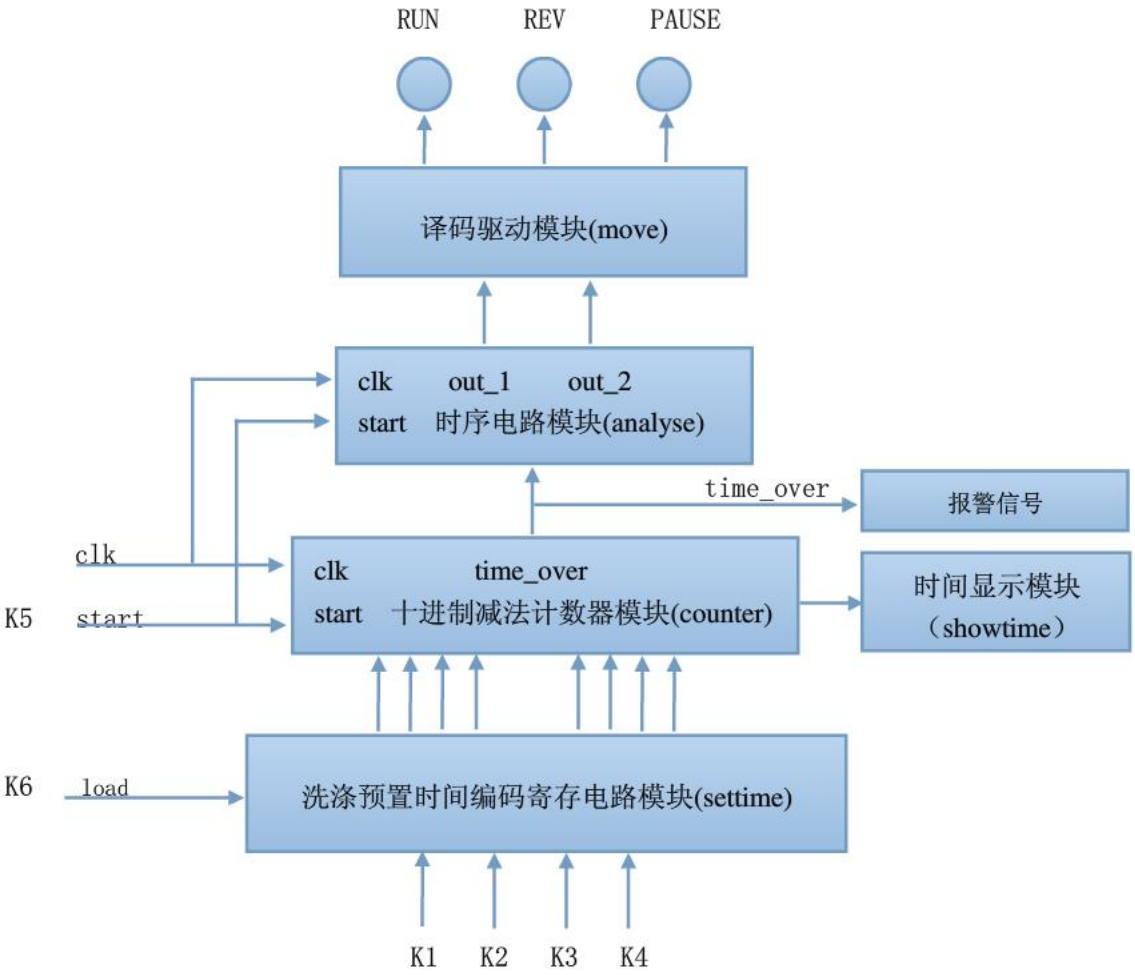
一、设计任务与要求

设计一个洗衣机洗涤程序控制器，控制洗衣机的电动机按下图所示的规律运转：



用两位数码管预置洗涤时间(分钟数)，洗涤过程在送入预置时间后开始运转，洗涤中按倒计时方式对洗涤过程作计时显示，用 LED 表示电动机的正、反转，如果定时时间到，则停机并发出音响信号。

二、总体框图



各个部分的具体功能描述如下：

（一）预设时间和编码电路（**settime**）：接受用户通过按钮预置的时间信息，编码

成八位之后转给减法计数器。

（二）减法计数器电路（**counter**）：接收编码之后的预置时间信息，向电机运转控制电路传递运行信号，并将预置时间信息和剩余时间信息发给数码管显示电路进行实时显示。

（三）数码管显示电路（**showtime**）：接收减法计数器电路传来的时间信息，进行实时译码显示。

（四）电机运转时序控制电路（**analyse**）：接收运行起止信号，安排电机运行状态并编码输出。

（五）译码器（**move**）：接收电机运行状态信号，译码后实时控制电机的正传、反转和暂停。

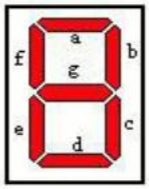
三、选择器件

1、pc 机一台。

2、CPLD/FPGA 适配器板：标准配置 EPF10K10LC84-4 接口板，下载接口是数字芯片的下载接口（DIGITAL JTAG），主要用于 CPLD/FPGA 芯片的数据下载。

3、实验箱：装有七段数码管及蜂鸣器等，七段数码管字形及真值表如下

七段数码管字形如下：



七段数码管真值表如下：

输入				输出							字形
D	C	B	A	F_a	F_b	F_c	F_d	F_e	F_f	F_g	
0	0	0	0	1	1	1	1	1	1	0	0
0	0	0	1	0	1	1	0	0	0	0	1
0	0	1	0	1	1	0	1	1	0	1	2
0	0	1	1	1	1	1	1	0	0	1	3
0	1	0	0	0	1	1	0	0	1	1	4
0	1	0	1	1	0	1	1	0	1	1	5
0	1	1	0	1	0	1	1	1	1	1	6
0	1	1	1	1	1	1	0	0	0	0	7
1	0	0	0	1	1	1	1	1	1	1	8
1	0	0	1	1	1	1	1	0	1	1	9

四、功能模块

4.1 时间预置编码寄存模块(settime)

1、时间预置编码寄存模块 settime 如图 1 所示，time_input 为通过开发板上拨码开关 K1、K2、K3、K4 输入的信号，load 为输入确认信号。本模块将输入的四位时间信号编码成八位二进制数输出到减法计数器电路。

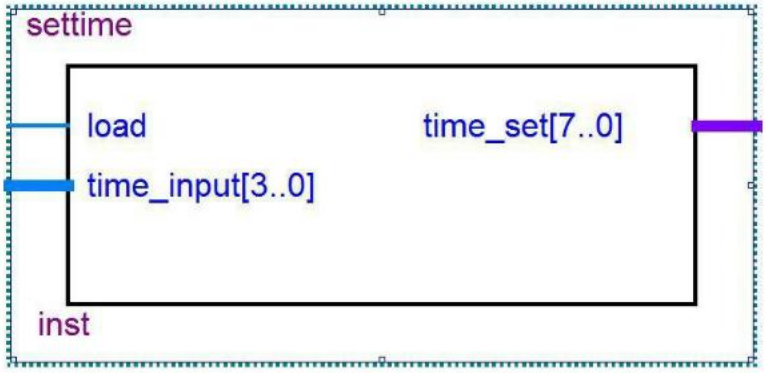


图 1　时间预置编码寄存模块 settime

2、仿真图

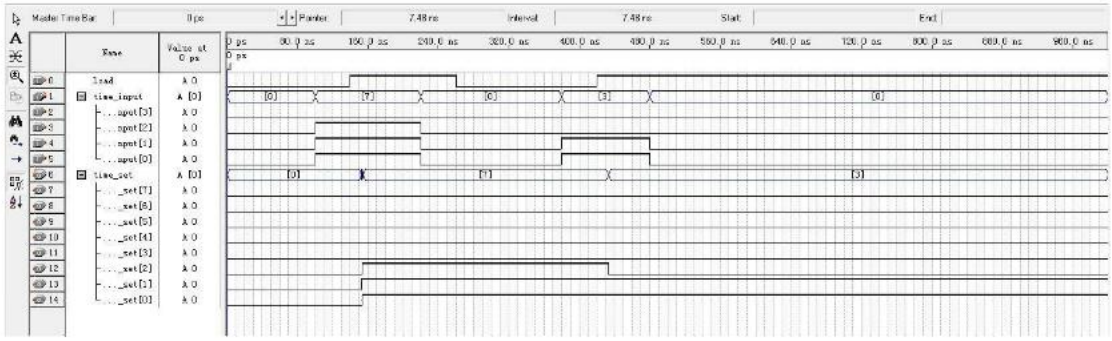


图 2　时间预置编码寄存模块仿真图

用 K1、K2、K3、K4 给 time_input 输入一个二进制数 0111，让 load 有效，输出 time_set 为 00000111。

3、时间预置编码寄存模块源代码

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

entity settime is
   port
   (
       load:in std_logic;
```

```
        time_input:in std_logic_vector(3 downto 0);
        time_set:out std_logic_vector(7 downto 0)
    );
end settime;

architecture settime of settime is
    signal p1:std_logic_vector(7 downto 0);
    begin
        process(load)
            begin
               if(load'event and load='1')
               then
                   case time_input is
                   when "0000"=>p1<="00000000";
                   when "0001"=>p1<="00000001";
                   when "0010"=>p1<="00000010";
                   when "0011"=>p1<="00000011";
                   when "0100"=>p1<="00000100";
                   when "0101"=>p1<="00000101";
                   when "0110"=>p1<="00000110";
                   when "0111"=>p1<="00000111";
                   when "1000"=>p1<="00001000";
                   when "1001"=>p1<="00001001";
                   when others=>p1<="00000000";
                   end case;
               end if;
        end process;
        time_set<=p1;
end settime;
```

4.2 减法计数器模块(counter)

1、减法计数模块 counter 如图 3 所示，本模块中 clk 为系统时序脉冲信号，start 为系统开始运行的信号，time_set 接收编码之后的预置时间信息，向电机运转控制电路传递运行信号，并将预置时间信息和剩余时间信息发给数码管显示电路进行实时显示。time_remain 为输出到数码管显示电路的时间信号，time_over 为系统运行结束信号，可以用来控制蜂鸣器的通断。

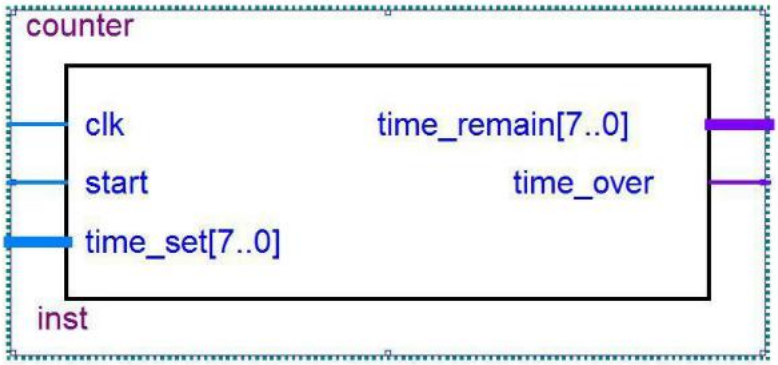


图 3　减法计数模块

2、仿真图

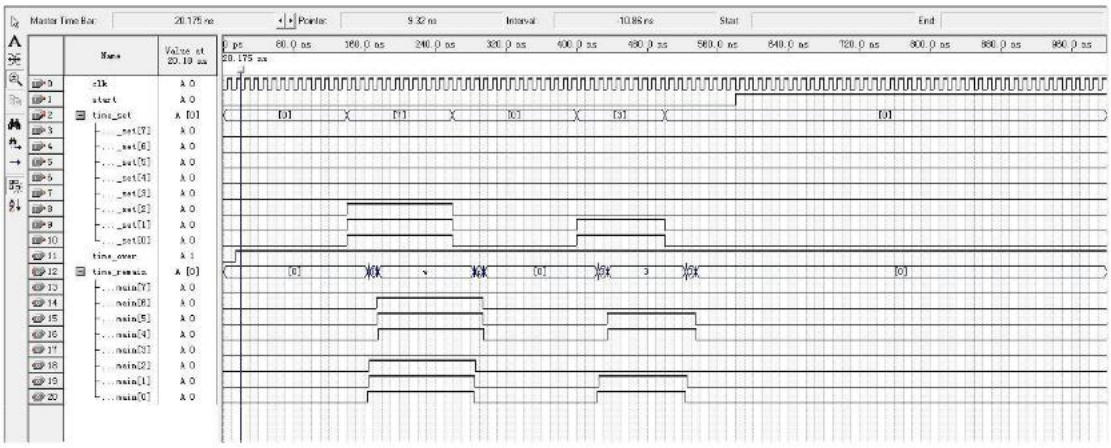


图 4　减法计数模块仿真图

3、减法计数模块源程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity counter is
    port
    (
       clk,start:in std_logic;
       time_set:in std_logic_vector(7 downto 0);
       time_remain:buffer std_logic_vector(7 downto 0);
       time_over:buffer std_logic
    );
end counter;
architecture counter of counter is
    begin
        process(clk)
```

```
variable time_second:integer range 0 to 59 :=59;
    begin
    if(clk'event and clk='1')
        then
        if(start='0')
           then
           if(time_remain(7 downto 0)=0)
              then
              time_remain<=time_set;
           else
              time_remain(7 downto 4)<=time_remain(3 downto 0);
              time_remain(3 downto 0)<=time_set(3 downto 0);
           end if;
           time_second:=59;
           time_over<='1';
        else
           if(time_over='1')
              then
              if(time_second=0 and time_remain(7 downto 0)=0)
                    then
                    time_over<='0';
                else
                    if(time_second=0)
                      then
                      if(time_remain(3 downto 0)=0)
                        then
             time_remain(7 downto 4)<=time_remain(7 downto 4)-1;
             time_remain(3 downto 0)<="1001";
                        time_second:=59;
                        else
             time_remain(7 downto 4)<=time_remain(7 downto 4);
             time_remain(3 downto 0)<=time_remain(3 downto 0)-1;
                             time_second:=59;
                        end if;
                    else
                        time_second:=time_second-1;
                    end if;
                end if;
```

```
                end if;
            end if;
        end if;
    end process;
end counter;
```

4.3 数码管显示模块（showtime）

1、数码管显示模块 showtime 如图 5 所示，本模块 clk 为系统时序脉冲信号,time_remain 接收减法计数器电路传来的时间信息，进行实时译码显示，a,b,c,d,e,f,g 分别对应数码管的七段，minute 和 second 分别位选两个数码管，显示十位和个位。

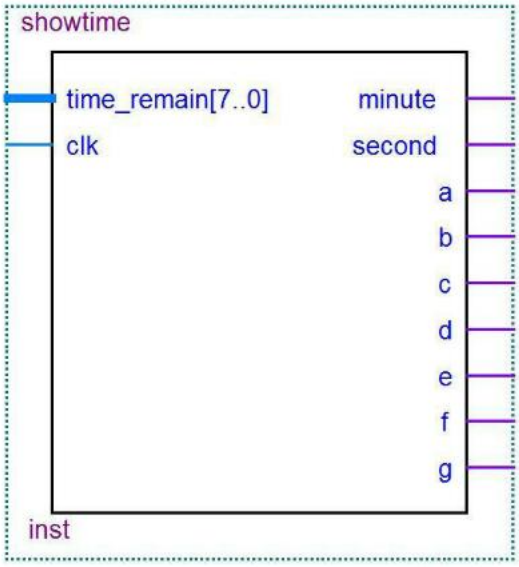


图 5　　数码管显示模块

2、仿真图

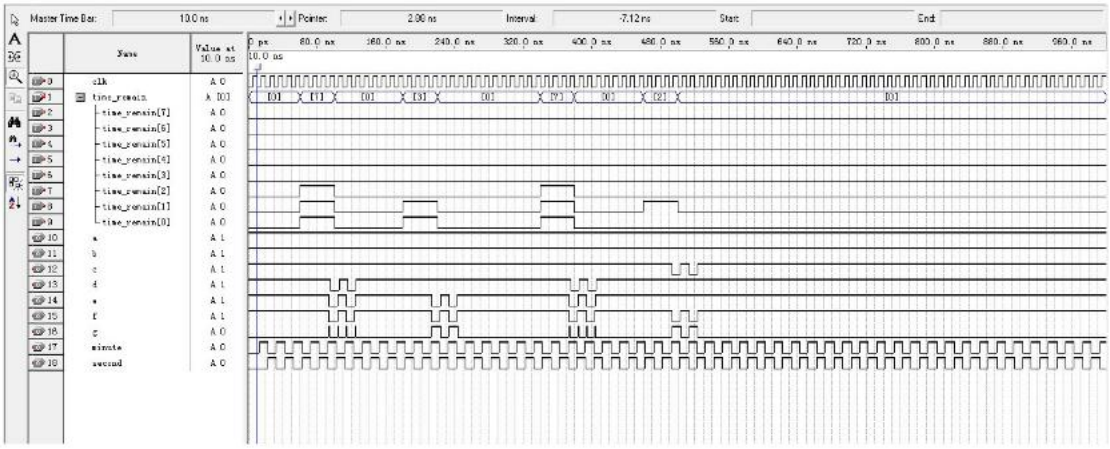


图 6　　数码管显示模块仿真图

3、数码管显示模块源程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity showtime is
    port
    (
      time_remain:in std_logic_vector(7 downto 0);
      clk:in std_logic;
      minute,second:out std_logic;
      a,b,c,d,e,f,g:out std_logic
    );
end showtime;
architecture showtime of showtime is
    signal temp:std_logic_vector(6 downto 0);
    signal bcd:std_logic_vector(3 downto 0);
    signal choose:std_logic;
    begin
        process(clk)
        begin
            if(clk'event and clk='1')
                then
                choose<=not choose;
                if(choose='1')
                    then
                    minute<='0';second<='1';
                    bcd<= time_remain(7 downto 4);
                else
                    minute<='1';second<='0';
                    bcd<= time_remain(3 downto 0);
                end if;
            end if;
        end process;
        process(bcd)
            begin
                case bcd is
                when  "0000"=>temp<="1111110";
                when  "0001"=>temp<="0110000";
```

```
                when  "0010"=>temp<="1101101";
                when  "0011"=>temp<="1111001";
                when  "0100"=>temp<="0110011";
                when  "0101"=>temp<="1011011";
                when  "0110"=>temp<="1011111";
                when  "0111"=>temp<="1110000";
                when  "1000"=>temp<="1111111";
                when  "1001"=>temp<="1111011";
                when  others=>temp<="1111011";
              end case;
a<=temp(6); b<=temp(5); c<=temp(4); d<=temp(3); e<=temp(2); f<=temp(1);
g<=temp(0);
        end process;
end showtime;
```

4.4 时序电路模块(analyse)

1、时序电路模块 analyse 如图 7 所示，本模块由 start 控制使能控制，通过时钟的输入进行计算当前系统所处的状态，并进行编码输出电机的运转状态，out_1 为高位时表示电机正转，out_2 为高位时表示电机反转。由于在显示以及输入的时候只有分钟，故在模块内部设计了一个秒的计时变量。

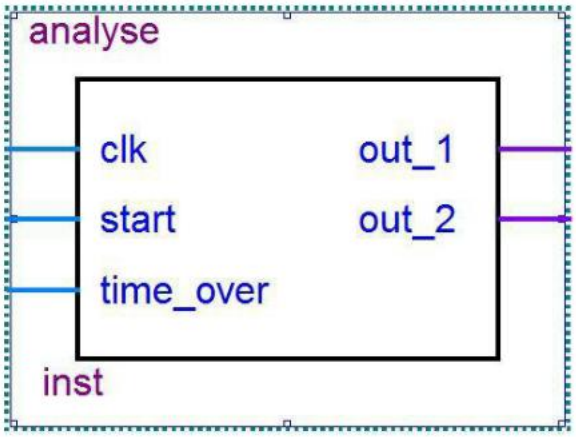
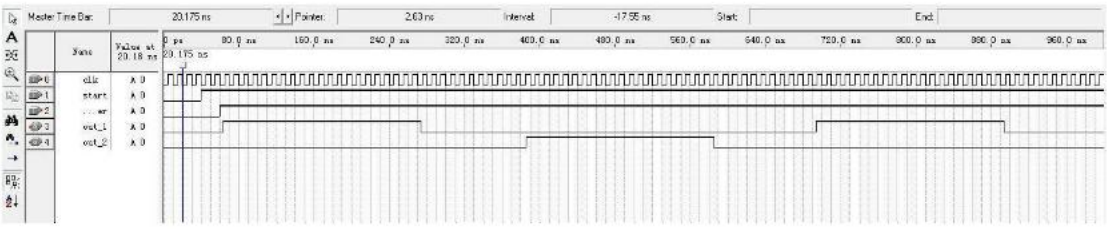


图 7　　时序电路模块

2 、仿真图



3、时序电路模块 analyse 源程序

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity analyse is
    port
    (
      clk,start,time_over:in std_logic;
      out_1,out_2:out std_logic
    );
end analyse;
architecture analyse of analyse is
    begin
        process(clk)
        variable  state:std_logic;
        variable  wash_time:integer:=0;
        variable  wait_time:integer:=0;
        begin
            if(clk'event and clk='1')
                 then
                 if(start='0')
                     then
                     wash_time:=0;
                     wait_time:=0;
                     state:='0';
                     out_1<='0';out_2<='0';
                 else
                     if(time_over='1')
                         then
                         if(wash_time=20)
                             then
                             if(wait_time=10)
                                 then
                                 wash_time:=0;
                                 state:=not state;
                             else
                                 wait_time:=wait_time+1;
                             end if;
```

```
                    else
                        wash_time:=wash_time+1;
                        wait_time:=0;
                    end if;
                end if;
                if (wash_time=20)
                    then
                    out_1<='0';out_2<='0';
                else
                    if(state='0')
                        then
                        out_1<='1';out_2<='0';
                    else
                        out_1<='0';out_2<='1';
                    end if;
                end if;
            end if;
        end if;
    end process;
end analyse;
```

4.5 译码器模块（move）

1、译码器模块 move 如图 9 所示，out_1 和 out_2 接收时序电路模块的信号对信号进行译码，安排电机运行状态即正转（RUN）、反转（REV）、暂停（PAUSE），并进行输出。此模块较为简单，设计基本没什么难度。

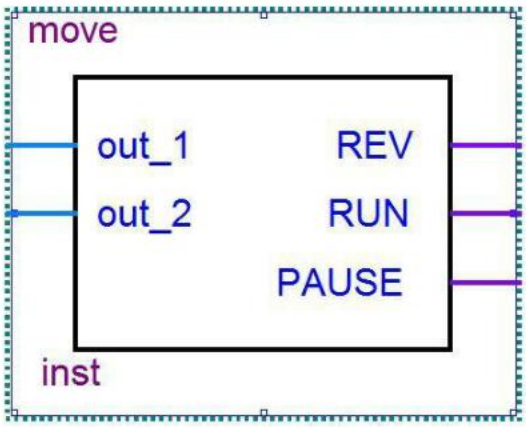


图 9　译码器模块

2、仿真图

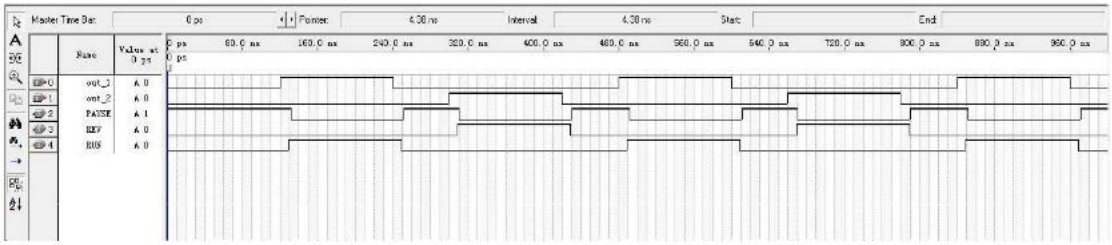


图 10　译码器模块仿真图

3、译码器模块 move 源程序

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

entity move is
    port
    (
      out_1,out_2:in std_logic;
      REV,RUN,PAUSE:buffer std_logic
    );
end move;

architecture move of move is
   signal choose:std_logic_vector(1 downto 0);
   begin
   choose(1)<=out_1;choose(0)<=out_2;
        process(choose)
         begin
         case choose is
         when "00"=>REV<='0';RUN<='0';PAUSE<='1';
         when "10"=>REV<='0';RUN<='1';PAUSE<='0';
         when "01"=>REV<='1';RUN<='0';PAUSE<='0';
         when others=>REV<='0';RUN<='0';PAUSE<='0';
         end case;
         REV<=out_2;RUN<=out_1;PAUSE<=not(out_1 or out_2);
        end process;
end move;
```

五、总体设计电路图

5.1 总体（顶层）设计电路图

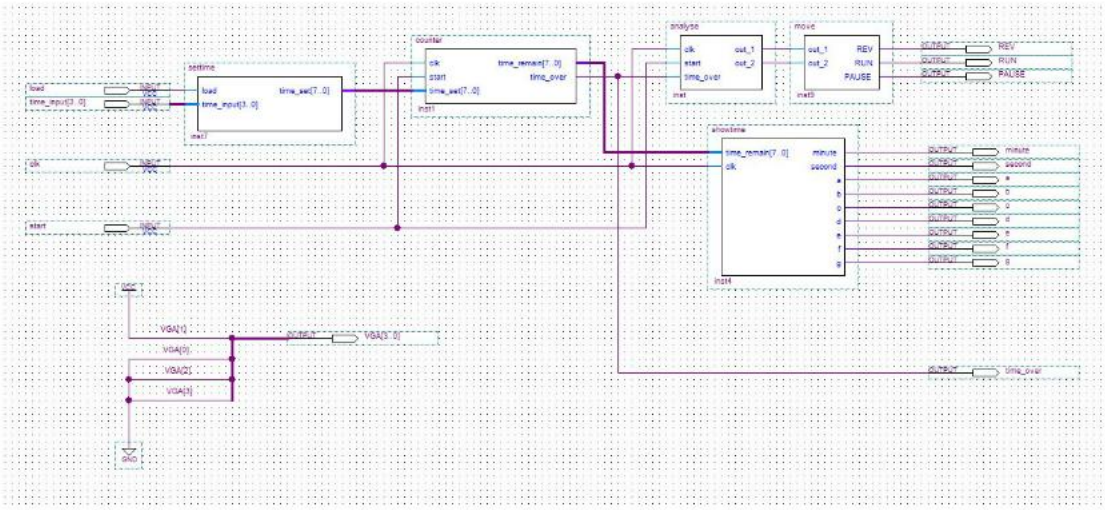
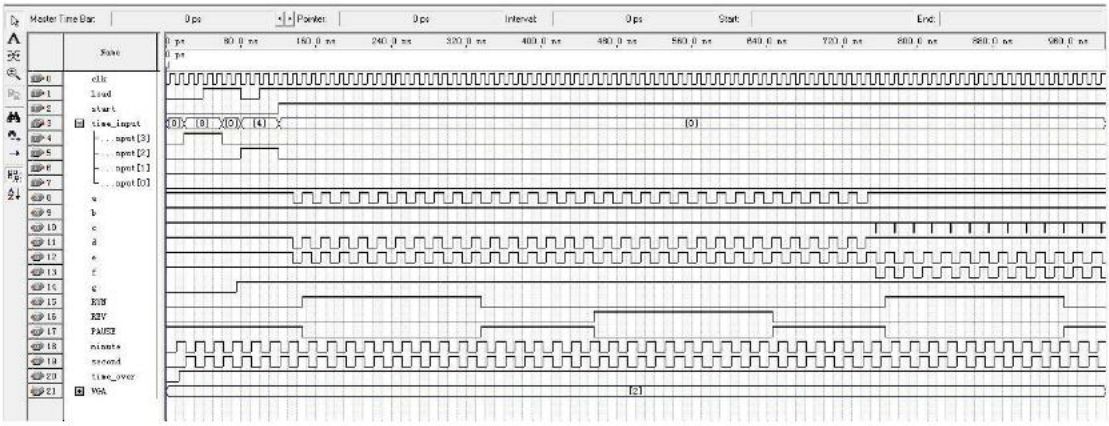


图 11　各模块连结后的电路图

系统运行过程如下：在系统进行运行之前，使用 K 按钮预置洗衣机运转时间，此时用户设定的时间通过数码管时时显示出来，计时设备选取的精度是分钟级，也就是说用户可以设定洗衣时间是多少分钟，范围为 00-99。然后用户可以给出开始信号，系统开始运转并开始从预设时间倒计时，重复“正传->暂停->反转->暂停”的循环过程直至剩余时间变为零，剩余时间为零时，time_over 指示报警信号。数码管在系统的整个运行过程中时时显示剩余运转时间。

本设计在电路中加入了扫描信号，输入到减法模块，时序电路模块，实时显示模块。由于扫描信号非常高，在我们看来，输出在数码管上的数字都是连续的两位数字，由预置时间开始以一分钟减一的速度递减。当数码管显示为零时，洗衣停止。

5.2 顶层文件仿真

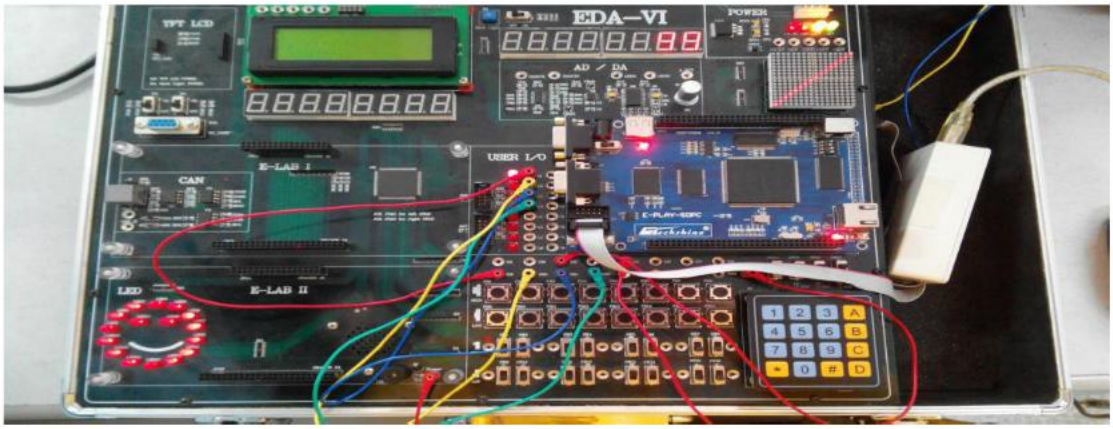


由上图可以看出：当预置号时间，启动 start，数码管显示预置时间，电机开始以正转=>暂停=>反转=>暂停为周期进行循环，一个周期正好费时一分钟，一个周期结束，数码管显示减一，依次循环，直至数码管显示时间为零，洗衣结束。

5.3 管脚分配图

	Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
1	a	Output	PIN_105	4	B4_N0	3.3-V LVTTL (default)	
2	b	Output	PIN_104	4	B4_N0	3.3-V LVTTL (default)	
3	c	Output	PIN_101	4	B4_N1	3.3-V LVTTL (default)	
4	clk	Input	PIN_28	1	B1_N1	3.3-V LVTTL (default)	
5	d	Output	PIN_100	4	B4_N1	3.3-V LVTTL (default)	
6	e	Output	PIN_85	4	B4_N2	3.3-V LVTTL (default)	
7	f	Output	PIN_84	4	B4_N2	3.3-V LVTTL (default)	
8	g	Output	PIN_83	4	B4_N2	3.3-V LVTTL (default)	
9	load	Input	PIN_88	4	B4_N1	3.3-V LVTTL (default)	
10	minute	Output	PIN_86	4	B4_N1	3.3-V LVTTL (default)	
11	PAUSE	Output	PIN_134	3	B3_N2	3.3-V LVTTL (default)	
12	REV	Output	PIN_132	3	B3_N2	3.3-V LVTTL (default)	
13	RUN	Output	PIN_133	3	B3_N2	3.3-V LVTTL (default)	
14	second	Output	PIN_87	4	B4_N1	3.3-V LVTTL (default)	
15	start	Input	PIN_93	4	B4_N1	3.3-V LVTTL (default)	
16	time_input[3]	Input	PIN_94	4	B4_N1	3.3-V LVTTL (default)	
17	time_input[2]	Input	PIN_95	4	B4_N1	3.3-V LVTTL (default)	
18	time_input[1]	Input	PIN_98	4	B4_N1	3.3-V LVTTL (default)	
19	time_input[0]	Input	PIN_99	4	B4_N1	3.3-V LVTTL (default)	
20	time_over	Output	PIN_135	3	B3_N2	3.3-V LVTTL (default)	
21	VGA[3]	Output	PIN_163	3	B3_N0	3.3-V LVTTL (default)	
22	VGA[2]	Output	PIN_164	3	B3_N0	3.3-V LVTTL (default)	
23	VGA[1]	Output	PIN_161	3	B3_N0	3.3-V LVTTL (default)	
24	VGA[0]	Output	PIN_162	3	B3_N0	3.3-V LVTTL (default)	
25	<<new node>>						

5.4 硬件实验效果图



六、课程设计心得体会

二周的课程设计终于做完了，在这二周的课设中我感觉我学到了蛮多东西。

首先，我学会了如何对一个大的课题进行分析——将大的整体划分为许多小的部分，直到各个部分容易设计出来。关于这个洗衣机控制器，就是用模块化层次化的设计方法进行系统层的设计，这样分解下来，设计会更容易点，思路也比较简单。洗衣机控制器主要就只有三个状态，要实现几种状态的多次循环的改变，其他的还有计时和数码显示的功能，通过每个模块的设计最后组装即可完成系统级的设计。在设计的时候，如果特别要注意各个模块之间接口的设计，要是接口不对，模块之间就没法实现组装。

其次，这次课程设计让我感受到了我对所学习的内容是多么的不熟练，在编程的时候还要老是去翻书。我记忆在深刻的是在编写程序时，看了一遍又一遍书但在编写的过程中还是出错了，最后只好对着书编写。但我觉的出现问题并不是很要紧，这些问题能提醒我那些地方没有学好，只要我重视这些地方将其巩固我想我将能学到许多的知识。通过这次设计，对于 VHDL 的设计方法大致有了一些技巧性的了解，位以后的硬件设计打下了基础，对 FPGA 的编程、定时器和计数器的设计都熟悉起来，加深了对时序组合电路的印象。

最后，我感觉我对 Quartus ＩＩ 软件的使用熟练了许多。我虽然以前在试验的时候使用过 Quartus ＩＩ 这个软件，但用的时间毕竟不长，对其不太熟练，经过这次做课设我对这个软件运用熟练很多，这对以后的学习一定有很大的帮助。我想在面对一个问题时不能存在侥幸心理，只要我们认真对待它，我们就能学到东西。通过在网上进行各种资料的查询，也发现了其实 FPGA 的设计具有较好的前景，其功能的强大和设计方法的简单可靠。具有较强的适应能力和可移植性。