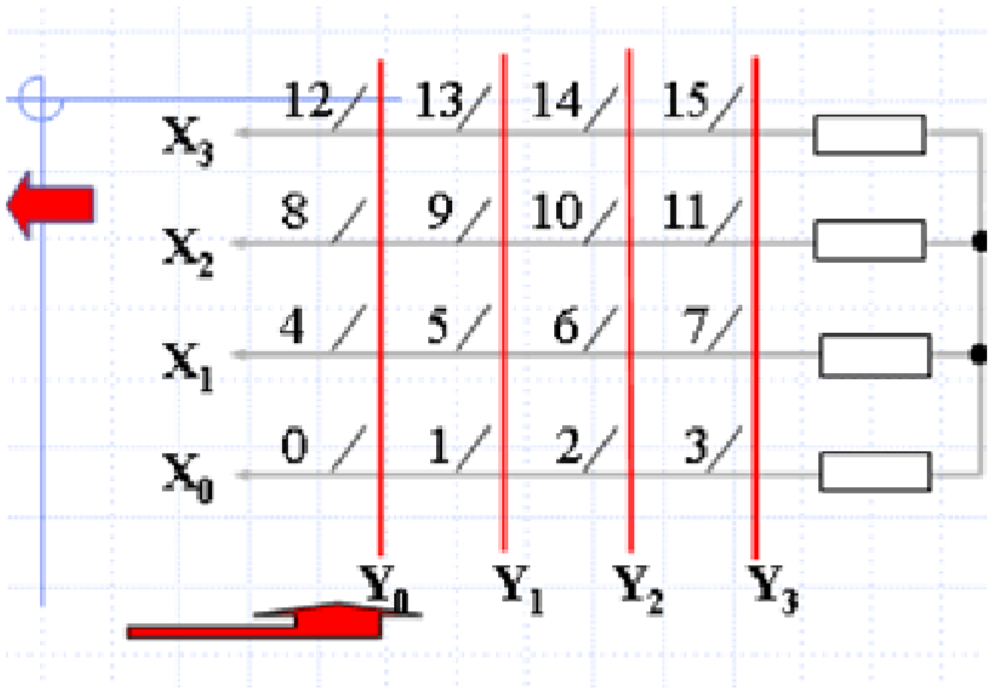
第十章综合设计实例

1 键盘扫描与显示

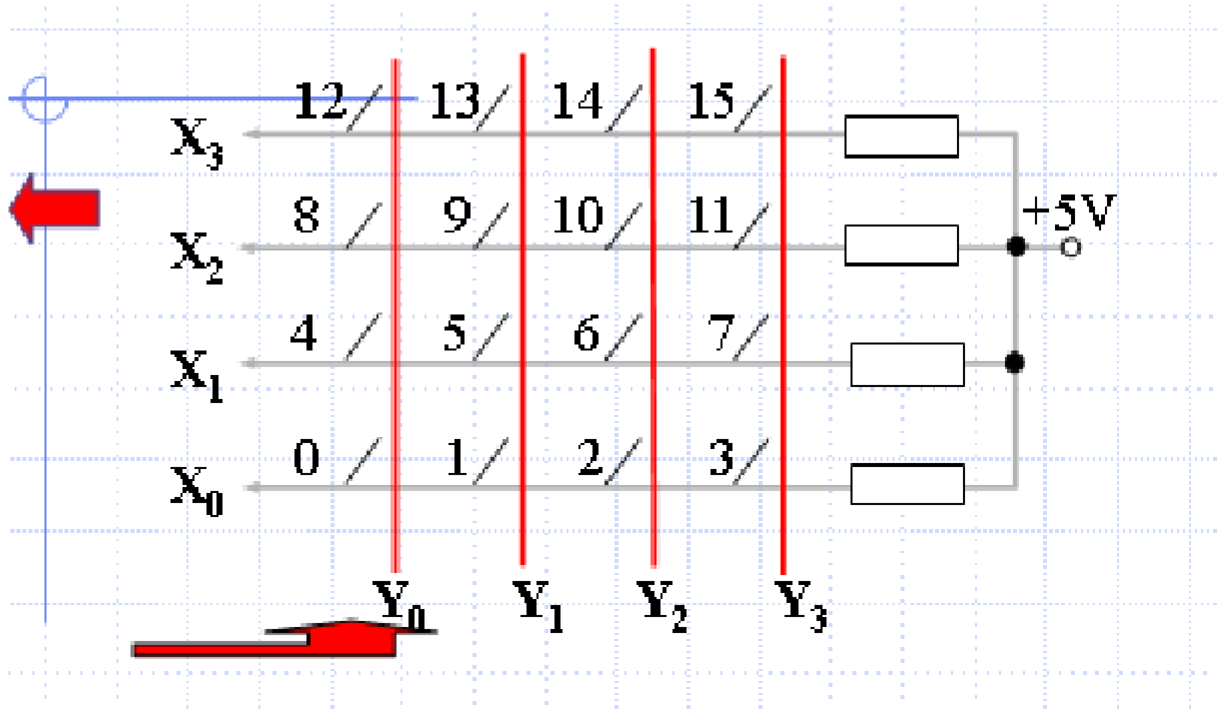
- 矩阵式键盘:行,列



矩阵是键盘以行列形式排列,键盘上每个按键其实是一个开关电路，当某键被按下时，该按键对应的位置就呈现逻辑0状态.

行扫描方式:逐行送0电平,读取列的状态,以判断按下的键号.

列扫描方式:逐列送0电平,读取行的状态,以判断按下的键号.

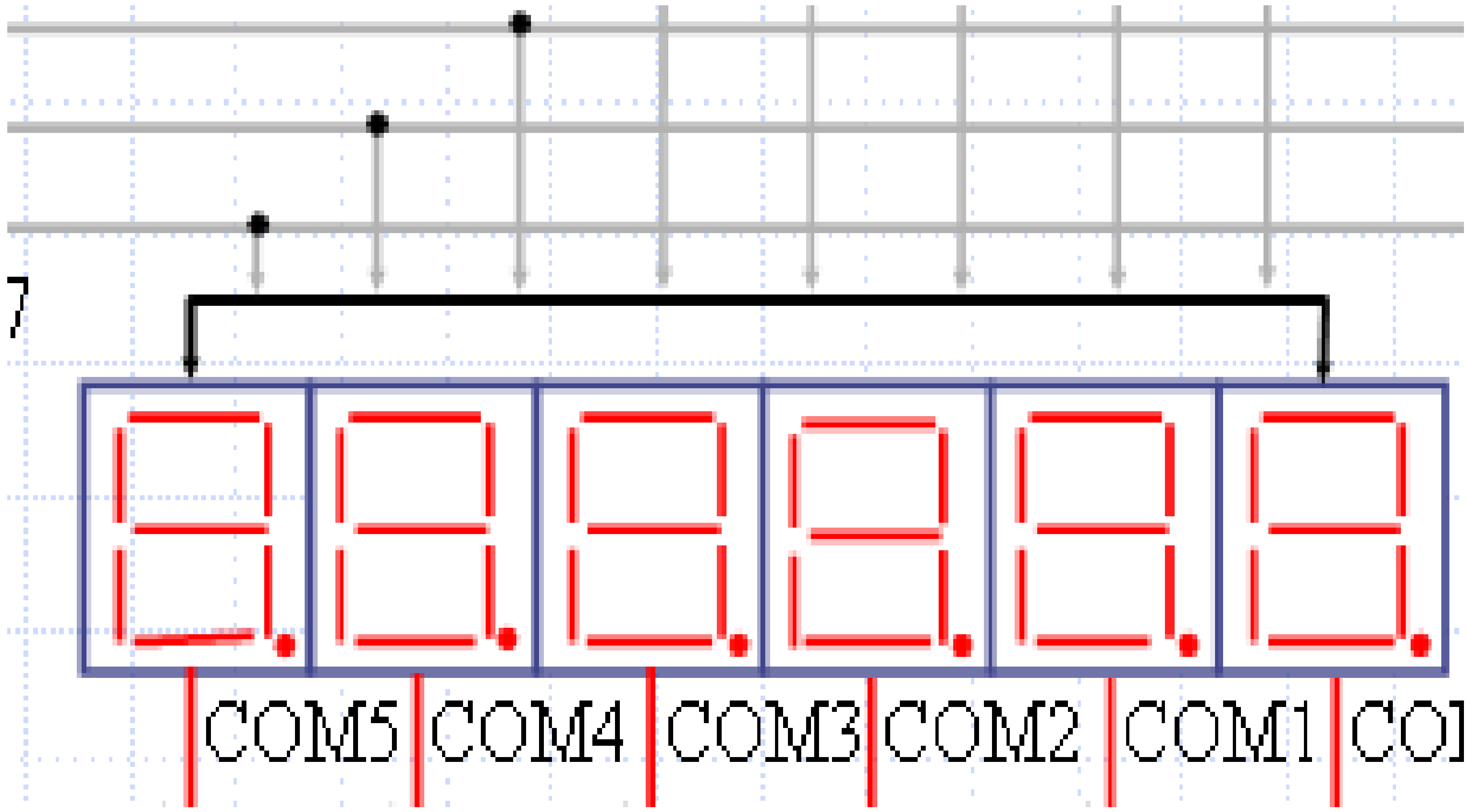


以行扫描为例:

1给行依次送0111,1011,1101,1110 信号;

2读取列电平状态;

数码管显示



按键扫描控制程序

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

•   entity key_scan is
•     port(column:in std_logic_vector(3 downto 0);  --      列状态
•           scan_cnt:in std_logic_vector(3 downto 0);---      扫描字
•           row:out std_logic_vector(3 downto 0);---     行状态
•           key_pressed:out std_logic);-----    按键有效与否,后续判断为零则为有键按下
•   end ;

•   architecture rtl of key_scan is
•   begin
•     row<="1110" when scan_cnt(3 downto 2)="00" else
•          "1101" when scan_cnt(3 downto 2)="01" else
•           "1011" when scan_cnt(3 downto 2)="10" else
•          "0111";
•      key_pressed<=column(0) when scan_cnt(1 downto 0)="00" else
•                   column(1) when scan_cnt(1 downto 0)="01" else
•                   column(2) when scan_cnt(1 downto 0)=“10" else
•                   column(3);
•      end rtl ;
```

按键处理控制模块

```
•   library ieee;
•   use ieee.std_logic_1164.all;
•   use ieee.std_logic_unsigned.all;
•   use ieee.std_logic_arith.all;
•   entity scan_count is
•   port(clk:in std_logic;--clock
•   scan_clk:in std_logic;--1khz clk
•   key_pressed:in std_logic;--检测按键有效与否,停止计数.
•       scan_cnt:out std_logic_vector(3 downto 0));--计数
•   end;

•   architecture behav of scan_count is
•   signal qscan:std_logic_vector(3 downto 0);
•   begin
•    scan_1:process(clk,scan_clk,key_pressed)
•           begin
•               if(clk'event and clk='1')then
•                  if(scan_clk='1' and key_pressed='1')then
•                         qscan<=qscan+1;          end if;   end if;
•            end process;
•           scan_cnt<=qscan;      end;
```

按键消抖控制模块

- **library ieee;**
- **use ieee.std_logic_1164.all;**
- **use ieee.std_logic_unsigned.all;**
- **use ieee.std_logic_arith.all;**
- **entity debounce is**
- **port(key_pressed:in std_logic;**
-     **clk:in std_logic;--**　同步时钟
-     **scan_clk:in std_logic;--1khz clock**
-     **key_valid:out std_logic);**
- **end;**
- **architecture behav of debounce is**
- **begin**
-

```
•   debounce:process(clk,scan_clk,key_pressed)
•         variable dbnq:std_logic_vector(5 downto 0);
•        begin
•           if(key_pressed='1')then
•               dbnq:="111111";--unkey_pressed,count reset at 63
•                 elsif(clk'event and clk='1')then
•                    if scan_clk='1' then
•                      if dbnq/=1 then
•                            dbnq:=dbnq-1;--key_pressed not enough long time
•                       end if;  end if;             end if;
•                     if dbnq=2 then
•                         key_valid<='1';--key_valid after key_pressed 1/63k second
•                        else
•                          key_valid<='0';
•                       end if;
•                    end process;
•            end;
```

键盘译码及按键存储模块

```
• library ieee;
• use ieee.std_logic_1164.all;
• use ieee.std_logic_unsigned.all;
• use ieee.std_logic_arith.all;
• entity code_tran is
• port(clk:in std_logic;--clock for synchrony
•      scan_cnt:in std_logic_vector(3 downto 0);--1khz clock
•      key_valid:in std_logic;
•      butt_code:out std_logic_vector(3 downto 0));
• end;

• architecture bb of code_tran is
• begin
•    process(clk)
•   begin
•         if(clk'event and clk='1')then
•           if key_valid='1' then
•                  case scan_cnt is
•   when"0000"=>butt_code<="0001";--1
•    when"0001"=>butt_code<="0010";--2
•    when"0010"=>butt_code<="0011";--3
•    when"0011"=>butt_code<="1100";--c
•    when"0100"=>butt_code<="0100";--4
•    when"0101"=>butt_code<="0101";--5
•    when"0110"=>butt_code<="0110";--6
•    when"0111"=>butt_code<="1101";--d
•    when"1000"=>butt_code<="0111";--7
•    when"1001"=>butt_code<="1000";--8
•    when"1010"=>butt_code<="1001";--9
•    when"1011"=>butt_code<="1110";--e
•    when"1100"=>butt_code<="1010";--a
•    when"1101"=>butt_code<="0000";--0
•    when"1110"=>butt_code<="1011";--b
•    when others =>butt_code<="1111";--f
•   end case;
•   end if; end if;
•   end process;end ;
```

电锁控制模块

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity ctrl is
 port(data_n:in std_logic_vector(3 downto 0);
      key_valid,clk:in std_logic;
      enlock:out std_logic;
      d,c,b,a:out std_logic_vector(3 downto 0));
end;

architecture aaa of ctrl is
signal acc,reg:std_logic_vector(15 downto 0);
signal nc:std_logic_vector(2 downto 0);
signal qa,qb:std_logic;
begin
keyin:block is
  begin
      process(data_n,key_valid)
```

```
begin
         if data_n="1101" then
             acc<="0000000000000000";
             nc<="000";
         elsif key_valid'event and key_valid='1' then
            if data_n<"1101" then
                 if nc<=4 then
                     acc<=acc(11 downto 0)&data_n;
                     nc<=nc+1;
                    end if;end if;end if;end process;end block;
   lock:block is
     begin
        process(clk,data_n)
          begin
             if(clk'event and clk='1')then
               if nc=4 then
                   if data_n="1110" then
                       reg<=acc;
                        qa<='1';qb<='0';
                 elsif data_n="1111" then
                    if reg=acc then
                       qa<='0';qb<='1';
                    end if;end if;end if;end if;end process;end block;
```

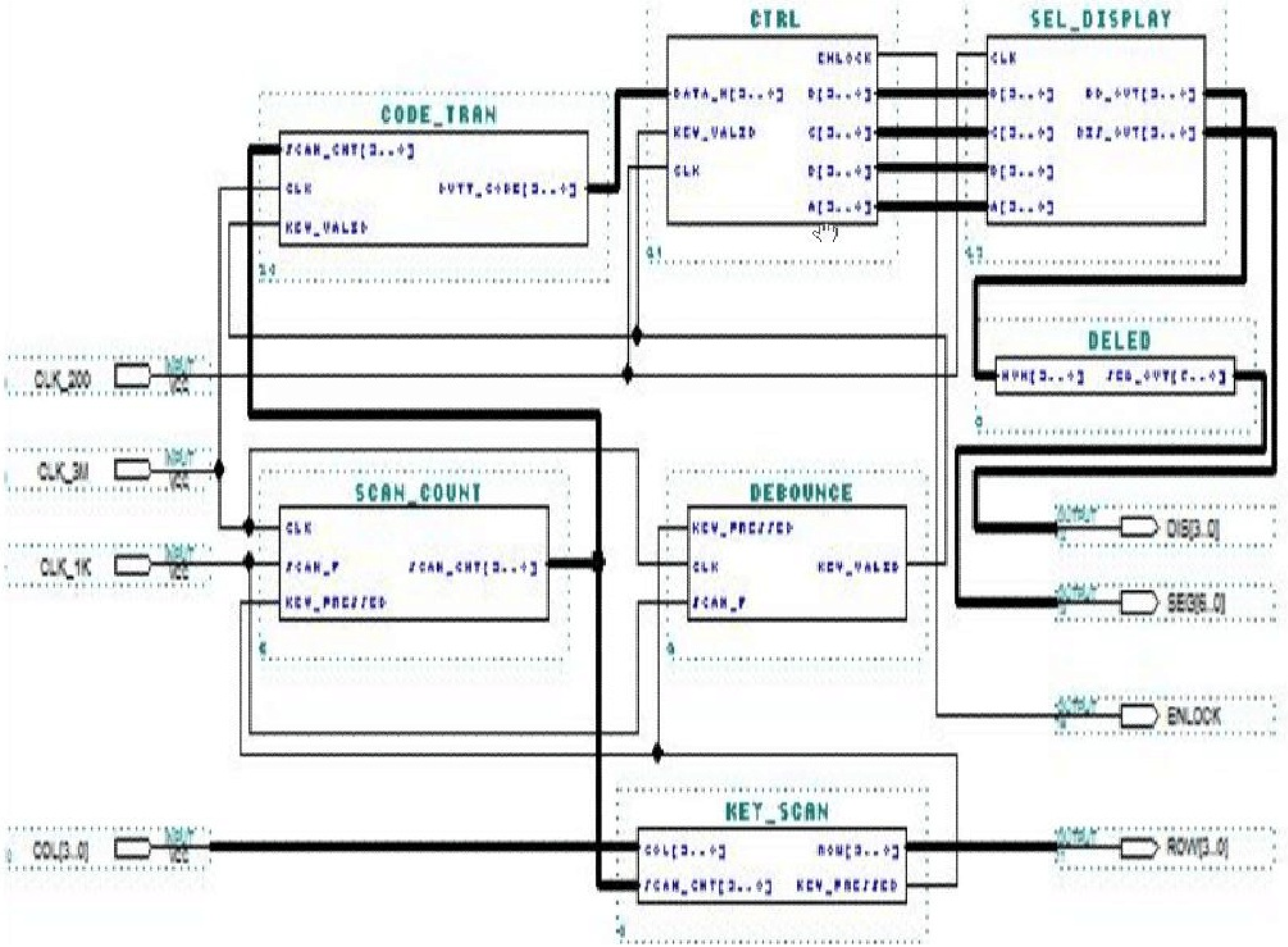
- enlock<=qa and not qb;
- d<=acc(15 downto 12);
- c<=acc(11 downto 8);
- b<=acc(7 downto 4);
- a<=acc(3 downto 0);
- end aaa;

动态扫描显示控制模块

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity sel_display is
port(clk:in std_logic;
     d,c,b,a:in std_logic_vector(3 downto 0);
      db_out:out std_logic_vector(3 downto 0);
      dis_out:out std_logic_vector(3 downto 0));
end entity;

architecture rtl of sel_display is
  signal sel:std_logic_vector(1 downto 0);
  signal dis:std_logic_vector(3 downto 0);
  signal db:std_logic_vector(3 downto  0);
begin
```

```
counter:block is
  signal q:std_logic_vector(6 downto 0);
   begin
      process(clk)
        begin
          if clk'event and clk='1' then
            q<=q+1;
         end if;
       end process;
      sel<=q(1 downto 0);
     end block counter;
```

```
multiplexer:block is
    begin
      process(sel)
        begin
          if sel=0 then
             db<=d;   dis<="0111";
            elsif sel=1 then
             db<=c;  dis<="1011";
            elsif sel=2 then
             db<=b;  dis<="1101";
             elsif sel=3 then
             db<=a; dis<="1110";
           end if;
         end process;
      end block multiplexer;
         db_out<=db;
         dis_out<=dis;
      end rtl;
```


CTRL
SEL_DISPLAY
CODE_TRAN
DELED
SCAN_COUNT
DEBOUNCE
KEY_SCAN
CLK_200
CLK_3M
CLK_1K
COL[3..0]
DIS[3..0]
SEG[6..0]
ENLOCK
ROW[3..0]
ENLOCK
DATA_N[3..0]
KEY_VALID
CLK
D[3..0]
C[3..0]
B[3..0]
A[3..0]
DD_OUT[3..0]
DIS_OUT[3..0]
SCAN_CNT[3..0]
OUTT_CODE[3..0]
NUM[3..0]
SEG_OUT[6..0]
SCAN_F
KEY_PRESSED
COL[3..0]
ROW[3..0]

实例1　数字钟设计

实时显示时、分、秒

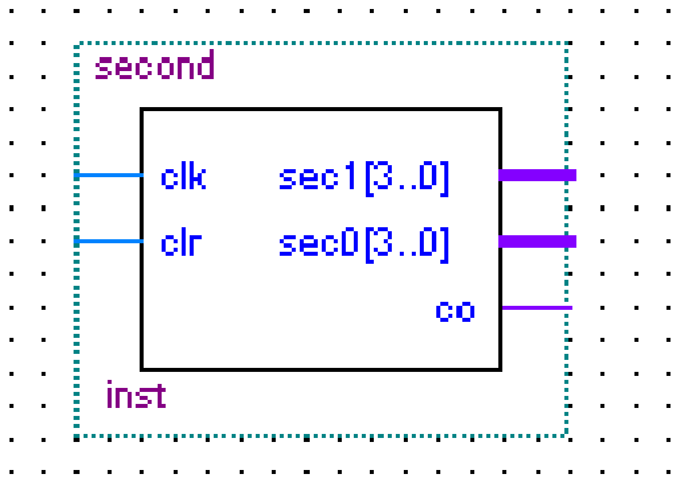
分析：

1、最小计时单位：秒。因此，首先要由时钟产1HZ的信号；

2、对秒进行0-59的计数，并且有进位功能，且显示；

3、对分进行0-59的计数，并且有进位功能，且显示；

4、对时进行0-59的计数，且显示；

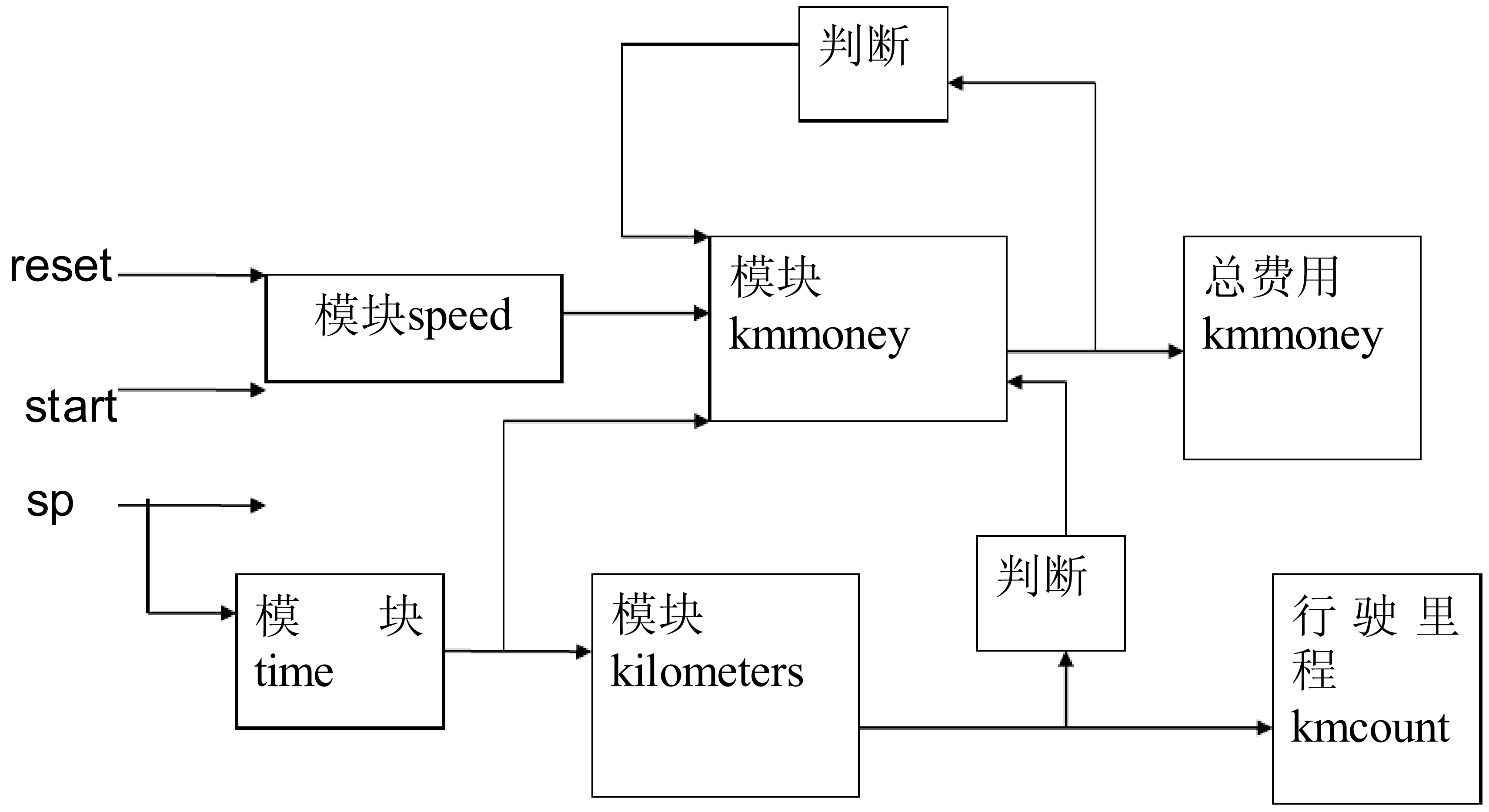

second
clk
clr
sec1[3..0]
sec0[3..0]
co
inst

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all
entity second is
  port(clk,clr:in std_logic;--clk=1Hz
      sec1,sec0:out std_logic_vector(3 downto 0);
          co:out std_logic );
end second;
architecture arch of second is
begin

process(clk,clr)
    variable cnt1,cnt0:std_logic_vector(3 downto 0);
    begin
        if clr='0' then
               cnt1:="0000";
               cnt0:="0000";
        elsif clk'event and clk='1' then
               if cnt1="0101" and cnt0="1000" then co<='1';
                      cnt0:="1001";
               elsif cnt0<"1001" then
                      cnt0:=cnt0+1;
 else
         cnt0:="0000";
if cnt1<"0101" then
cnt1:=cnt1+1;
else
        cnt1:="0000";
              co<='0';
                     end if;
               end if;
          end if;

         sec1<=cnt1;
         sec0<=cnt0;
       end process;
 end arch;
```

实例2　出租车计费器设计

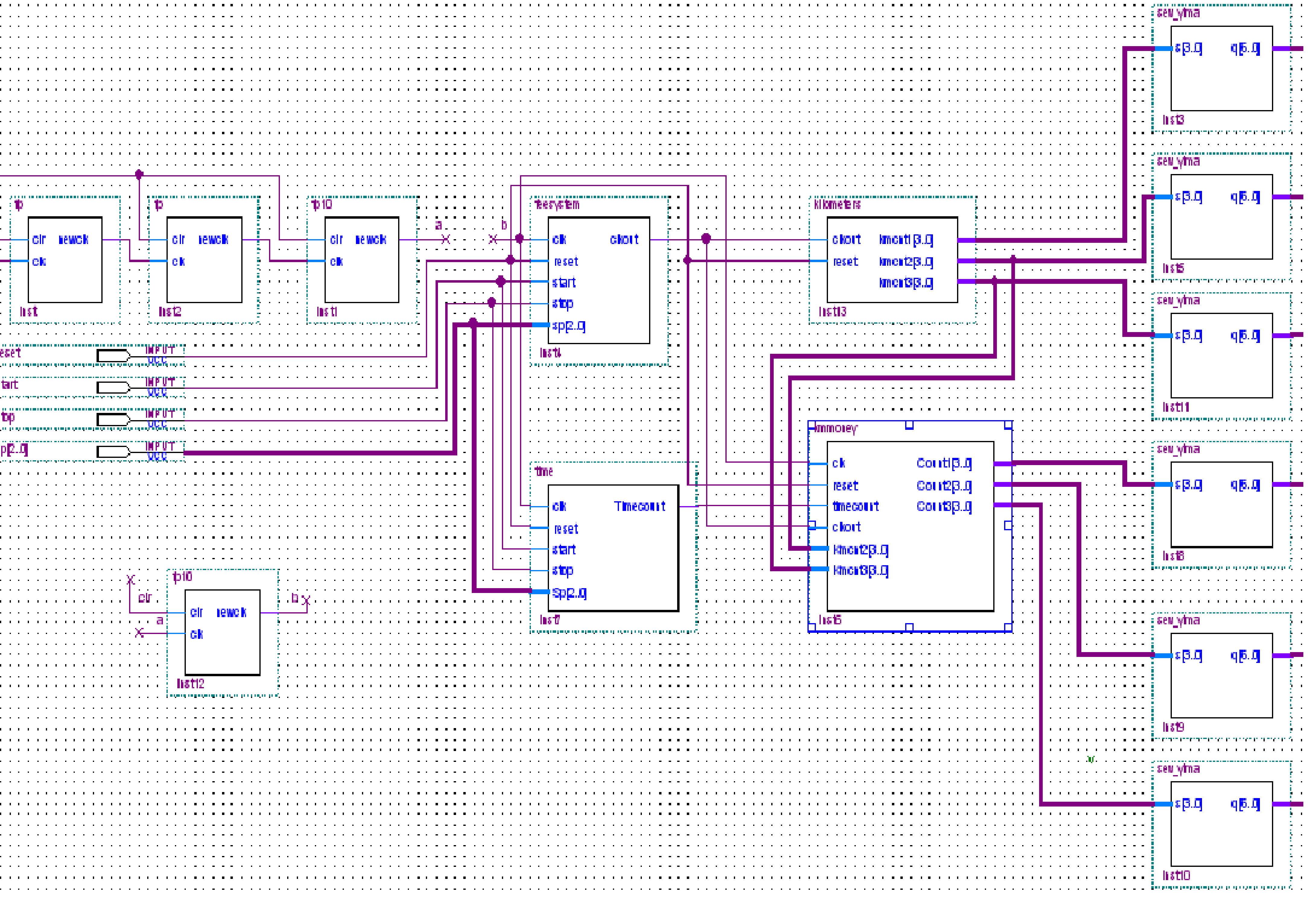
功能:

- （1）实现计费功能。车起步开始计费，首先显示起步价，设起步费为8.00元，车在行驶3 km以内，只收起步价8.00元。车行驶超过3 km后，每公里2元，车费依次累加。当总费用达到或超过40元时，每公里收费4元。当遇到红灯或客户需要停车等待时，则按时间计费，计费单价为每20S收费1元。
- （2）实现预置功能：预置起步费、每公里收费、车行加费里程、计时收费。
- （3）实现模拟功能：模拟汽车行驶、停止、暂停等状态。
- （4）实现显示功能：将路程与车费显示出来，以十进制显示。


判断
reset
模块speed
模块
kmmoney
总费用
kmmoney
start
sp
模　块
time
模块
kilometers
判断
行 驶 里
程
kmcount

系统流程如下：

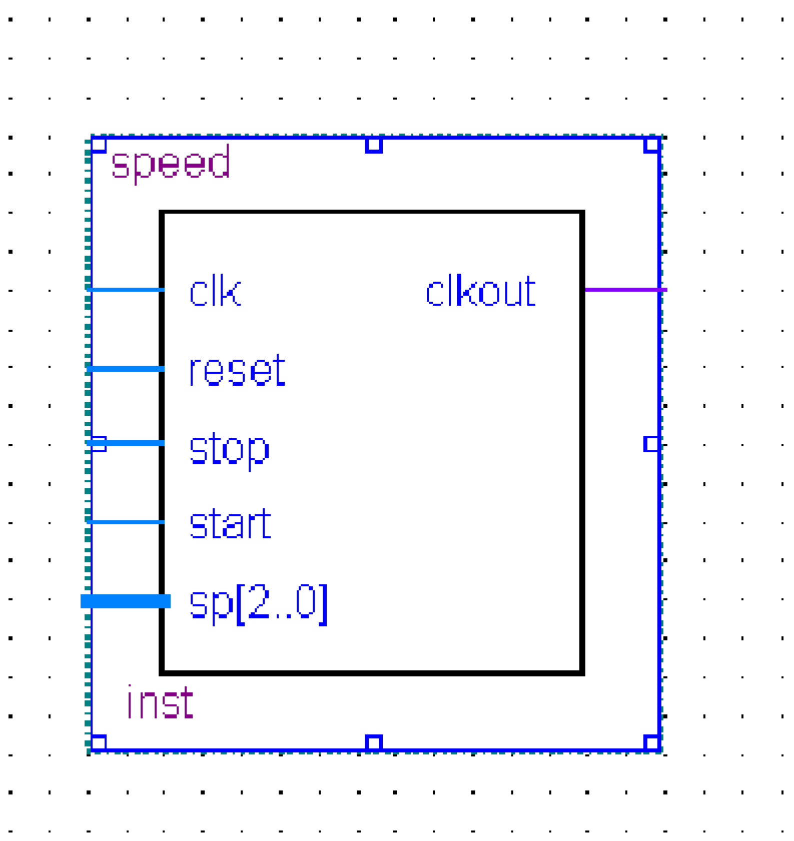
- （1）系统接收到reset信号后，总费用变为8元，同时其他计数器，寄存器等全部清零。
（2）系统接收到start信号后，首先把部分寄存器赋值，总费用不变，单价price寄存器通过对总费用的判断后赋为2元。其他寄存器和计数器等继续保持为0。
（3）speed进程：通过对速度信号sp的判断，决定变量kinside的值。kinside即是行进100m所需要的时钟周期数，然后每行进100m，则产生一个脉冲clkout。
（4）kilometers进程：由于一个clkout信号代表行进100m，故通过对clkout计数，可以获得共行进的距离kmcount。
（5）time进程：在汽车启动后，当遇到顾客等人或红灯时，出租车采用计时收费的方式。通过对速度信号sp的判断决定是否开始记录时间。当sp=0时，开始记录时间。当时间达到足够长时则产生timecount脉冲，并重新计时。一个timecount脉冲相当于等待的时间达到了时间计费的长度。这里选择系统时钟频率为500Hz，20s即计数值为1000。
（6）kmmoney可分为kmmoney1和kmmoney2两个进程。
　kmmoney1进程：根据条件对enable和price赋值。当记录的距离达到3公里后enable变为‘1’，开始进行每公里收费，当总费用大于40元后，则单价price由原来的2元每公里变为4元每公里。
　kmmoney2进程：在每个时钟周期判断timecount和clkout的值。当其为‘1’时，则在总费用上加上相应的费用。



fp
clr newclk
clk
inst
fp
clr newclk
clk
inst2
fp10
clr newclk
clk
inst1
a
b
taxisystem
clk clkout
reset
start
stop
sp[2..0]
inst4
kilometers
clkout kmout1[3..0]
reset kmout2[3..0]
kmout3[3..0]
inst13
eset
INPUT
VCC
tart
INPUT
VCC
top
INPUT
VCC
p[2..0]
INPUT
VCC
time
clk Timecount
reset
start
stop
sp[2..0]
inst7
mmoney
clk Count1[3..0]
reset Count2[3..0]
timecount Count3[3..0]
clkout
kmout2[3..0]
kmout3[3..0]
inst5
fp10
clr
clr newclk
a
clk
b
inst12
sev_yima
s[3..0] q[6..0]
inst3
sev_yima
s[3..0] q[6..0]
inst5
sev_yima
s[3..0] q[6..0]
inst11
sev_yima
s[3..0] q[6..0]
inst8
sev_yima
s[3..0] q[6..0]
inst9
sev_yima
s[3..0] q[6..0]
inst10

speed 模块

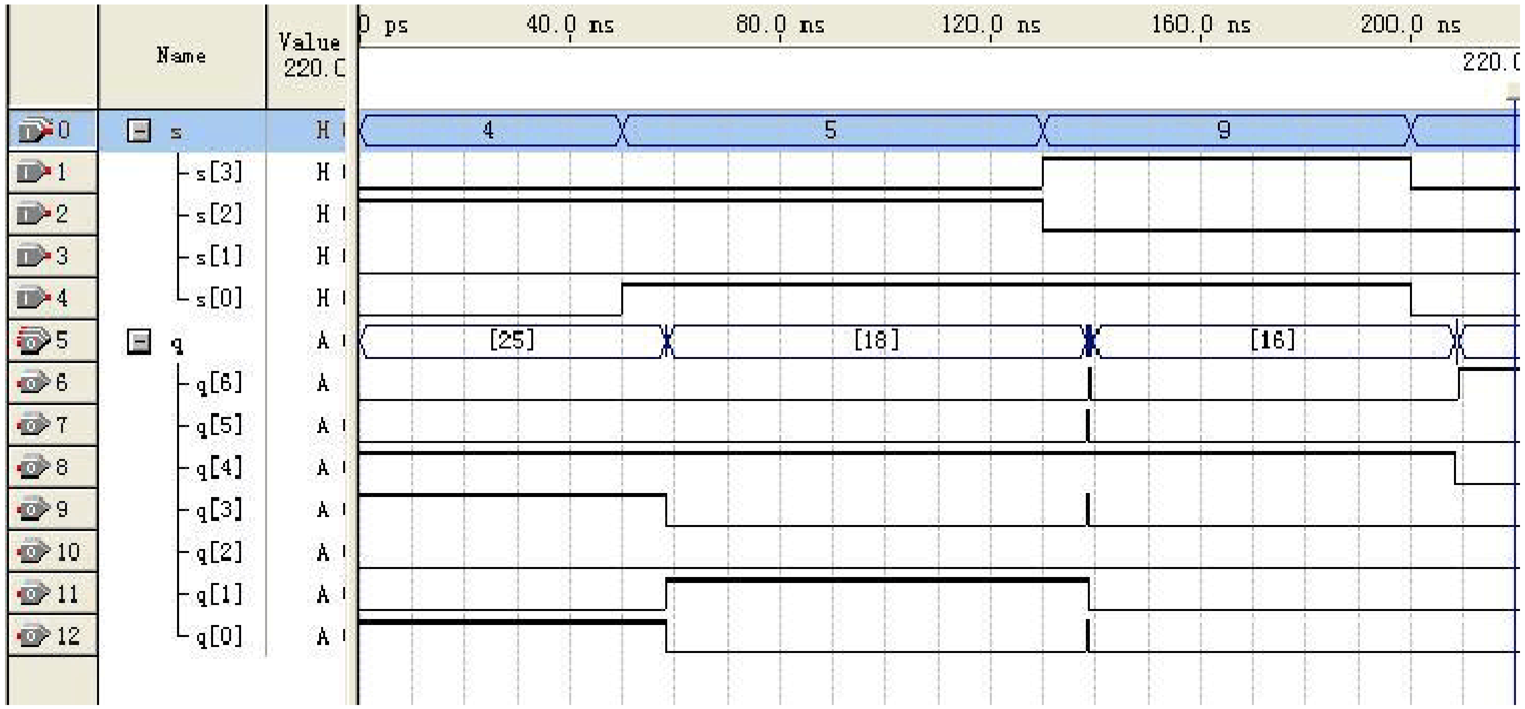
speed 进程首先根据**start**信号判断是否开始计费，然后根据输入的速度档位**sp[2..0]** 的判断，确定行驶**100M**所需要的时钟数，每前进**100M**，输出一个**clkout** 信号。同时由**cnt** 对**clk**进行计数，当**cnt=kinside** 时，把**clkout** 信号置 **‘1“，cnt** 清**0**。



```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Entity feesystem is
  Port(clk:in std_logic;
       reset:in std_logic;
       start: in std_logic;
       stop: in std_logic;
       sp:in std_logic_vector(2 downto 0);
       clkout:out std_logic);
end feesystem;
architecture behav of feesystem is
begin
  process(clk,reset,stop,start,sp)
    type state_type is(s0,s1);
    variable s_state:state_type;
    variable cnt:integer range 0 to 28;
    variable kinside:integer range 0 to 30;
  begin
    case sp is              ---7档速度选择，具体每档kinside的值可根据实际情
况设定
      when "000"=> kinside:=0;    -- 停止状态或空挡
```

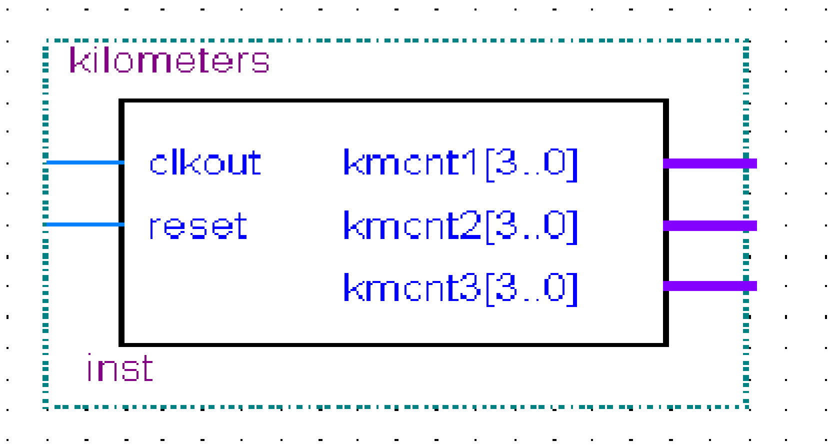
```
 when "001"=> kinside:=28; -- 第一档，慢速行驶状态，行驶100m需要28个时
钟周期
     when "010"=> kinside:=24;  -- 第二档
     when "011"=> kinside:=20;  -- 第三档
     when "100"=> kinside:=16;  -- 第四档
     when "101"=> kinside:=12;  -- 第五档
     when "110"=> kinside:=8;   --第六档
     when "111"=> kinside:=4;   --第七档，也是速度最大的档
   end case;
     if reset='1'then s_state:=s0;
     elsif clk'event and clk='1'then
        case s_state is
        when s0=>
          cnt:=0;clkout<='0';
          if start='1'then s_state:=s1;
          else s_state:=s0;
          end if;
```

```
when s1=>
        clkout<='0';
        if stop='1' then s_state:=s0;  --相当于无客户上车
        elsif sp="000" then s_state:=s1;  ---有客户上车，但车速位0，即客户刚上车还未起步
        elsif cnt=kinside then cnt:=0;clkout<='1'; s_state:=s1;
        else cnt:=cnt+1; s_state:=s1;
        end if;
      end case;
   end if;
 end process;
end behav;
```



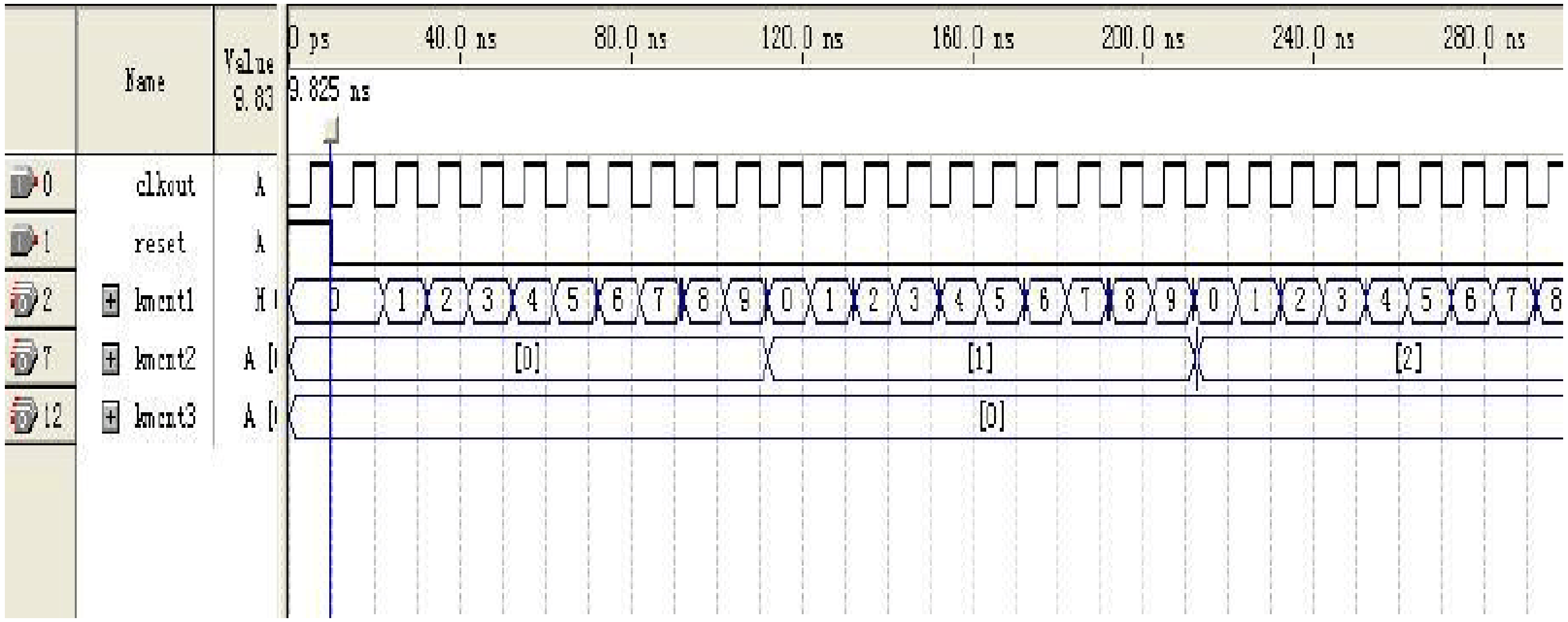
kilometers 模块

此模块主要用于记录行进的距离。通过clkout 信号的计数，可以计算行驶的距离kmcount 。一个clkout 脉冲相当于行进100m，所以只要记录clkout 的脉冲数目即可确定共行进的距离kmcount1 为十分位，kmcount2 为个位,kmcount3 为十位，分别为十进制数。



```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Entity kilometers is
   Port(clkout,reset:in std_logic;
       kmcnt1:out std_logic_vector(3 downto 0);
       kmcnt2:out std_logic_vector(3 downto 0);
       kmcnt3:out std_logic_vector(3 downto 0));
end kilometers;
architecture behav of kilometers is
begin
   process(clkout,reset)
     variable km_reg:std_logic_vector(11 downto 0);
begin
   if reset='1'then km_reg:="000000000000";
   elsif clkout'event and clkout='1'then            --km_reg(3 downto 0)对应里程十分
位
```

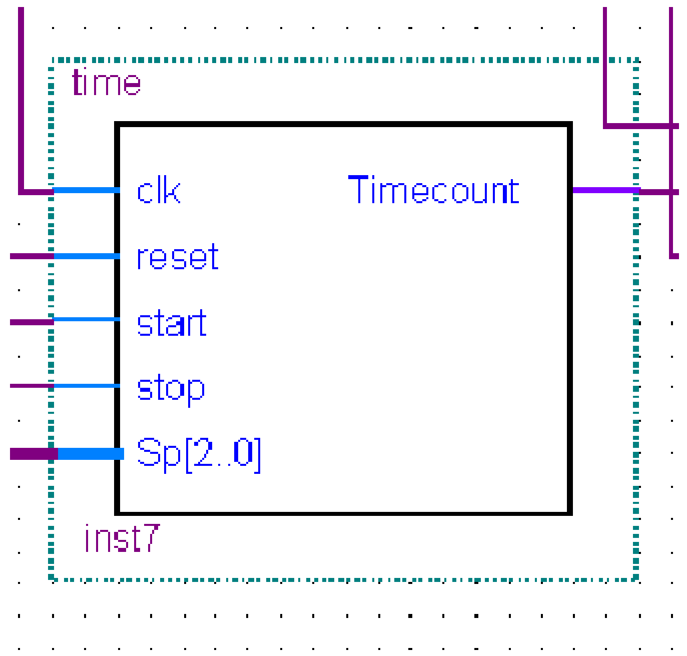
```
 if km_reg(3 downto 0)="1001"then
    km_reg:=km_reg+"0111";                --十分位向个位的进位处理
  else km_reg(3 downto 0):=km_reg(3 downto 0)+"0001";
  end if;
  if km_reg(7 downto 4)="1010"then
    km_reg:=km_reg+"01100000";            --个位向十位的进位处理
  end if;
end if;
kmcnt1<=km_reg(3 downto 0);
kmcnt2<=km_reg(7 downto 4);
kmcnt3<=km_reg(11 downto 8);
end process;
end behav;
```



time模块

time模块主要用于计时收费。记录计程车速度为0的时间（如等待红灯）。通过对sp信号的判断，当sp=0，开始记录时间。当时间达到足够长时，产生timecount脉冲，并重新计时。

time进程，产生时间计费脉冲timecount，单位计费时间可设定，这里选为20s



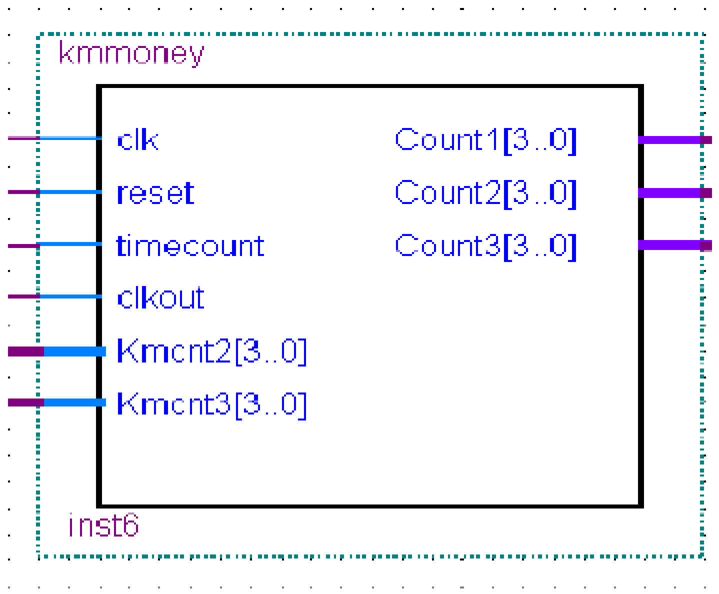
```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Entity time is
   Port(clk,reset,start,stop:in std_logic;
                      Sp:in std_logic_vector(2 downto 0);
                  Timecount:out std_logic);
End time;
architecture behav of time is
begin
   process(reset,clk,sp,stop,start)
     type state_type is(t0,t1,t2);
     variable t_state:state_type;
     variable waittime: integer range 0 to 1000;
   begin
   if reset='1'then t_state:=t0;
   elsif(clk'event and clk='1')then
```

```
 case t_state is
       when t0 =>
         waittime:=0;timecount<='0';
         if start='1'then t_state:=t1;
         else t_state:=t0;
         end if;
       when t1 =>
          if sp="000"then t_state:=t2;
          else waittime:=0;t_state:=t1;
          end if;
       when t2 =>
          waittime:=waittime+1;
          timecount<='0';
          if waittime=1000 then
             timecount<='1';     --20s,即1000个clk，产生一个时间计费脉冲
             waittime:=0;
          elsif stop='1'then t_state:=t0;
          elsif sp="000"then t_state:=t2;
          else timecount<='0';t_state:=t1;
          end if;
      end case;
end if;
  end process;
end behav;
```

kmmoney 模块

kmmoney 可分为**kmmoney1** 和**kmmoney2** 两个模块。

kmmoney1 用于产生**enable** 和**price** 信号。当记录距离达到**3**公里后，**enable** 信号为 '**1**'，开始进行每公里的收费。当总费用大于**40**元后，单价**price** 由原来的**2**元变为**4**元。用作计时收费。通过对**sp**信号的判断，当**sp=0**，开始记录时间。当时间达到足够长时，产生**timecount** 脉冲，并重新计时。**kmmoney2** 用于判断**timecount** 和**clkout** 的值，当其为 '**1**'时，总费用加**1**。最终输出为总费用。

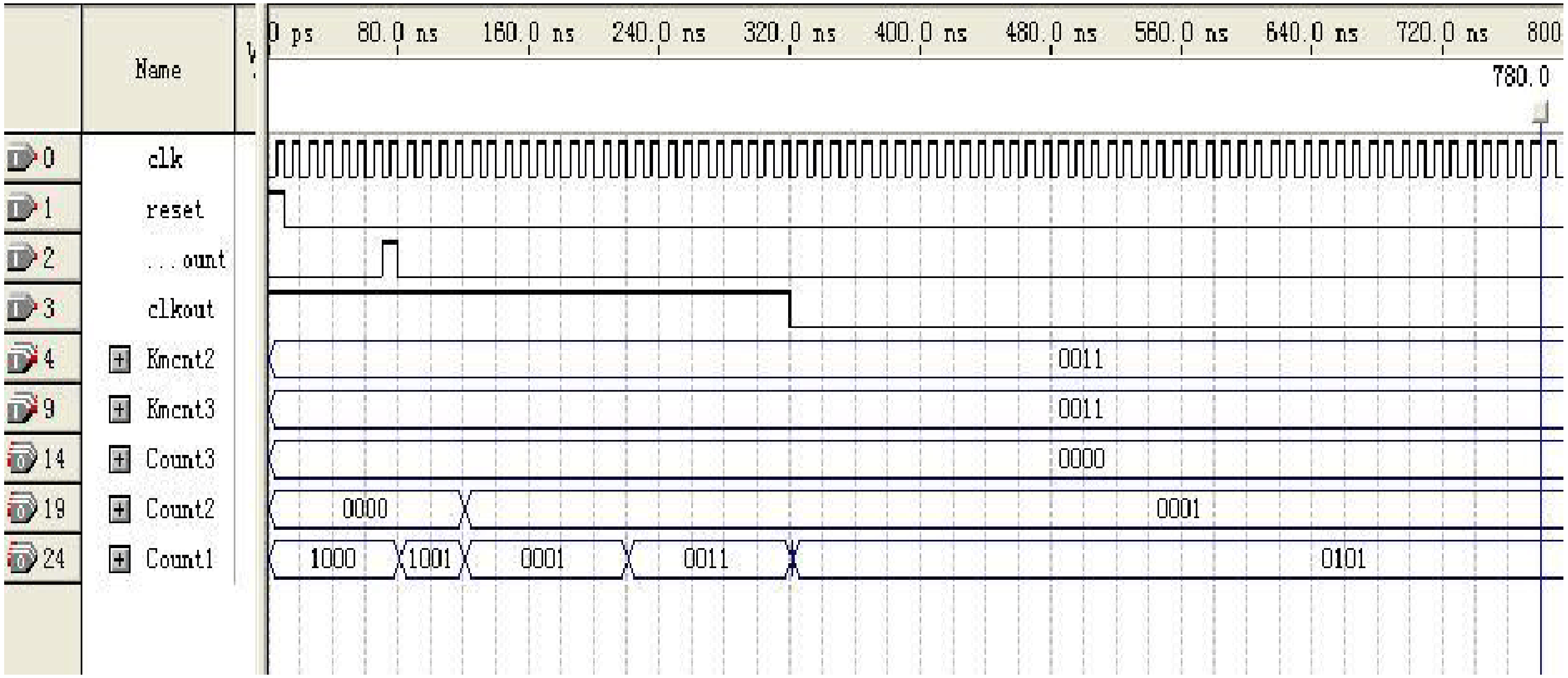


计费采用两个进程**kmmoney1** 和**kmmoney2** 实现;
总费用上限位**999**元，一旦超过上限，显示会出现不正常现象

```
Library ieee;
Use ieee.std_logic_1164.all;
Use ieee.std_logic_unsigned.all;
Entity kmmoney is
   Port(clk,reset,timecount,clkout:in std_logic;
              Kmcnt2:in std_logic_vector(3 downto 0);
              Kmcnt3:in std_logic_vector(3 downto 0);
               Count1:out std_logic_vector(3 downto 0);
               Count2:out std_logic_vector(3 downto 0);
               Count3:out std_logic_vector(3 downto 0));
End kmmoney;
Architecture behav of kmmoney is
Signal cash:std_logic_vector(11 downto 0);
Signal price:std_logic_vector(3 downto 0);
Signal enable: std_logic;
Begin
   Process(cash,kmcnt2)
   Begin
      If cash>="000001000000"then price<="0100";
      Else price<="0010";
      End if;
```

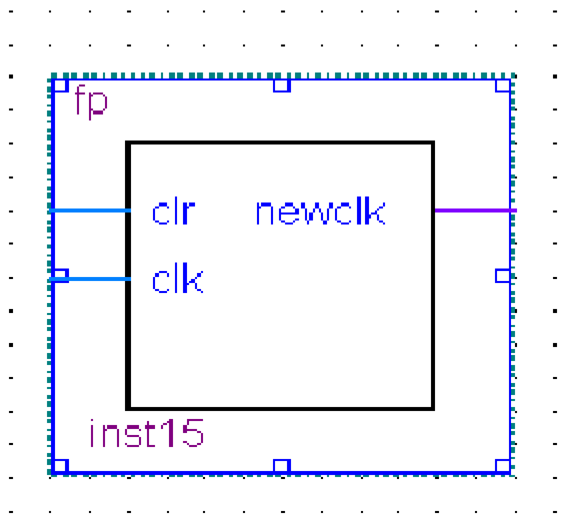
```
 If(kmcnt2>="0011")or(kmcnt3>="0001")then enable<='1';
     Else enable<='0';
     End if;
   End process;
kmmoney2:process(reset,clkout,clk,enable,price,kmcnt2)
variable reg2:std_logic_vector(11 downto 0);
variable clkout_cnt:integer range 0 to 10;
begin
 if reset='1' then cash<="000000001000";    -- 起步费用设为8元
 elsif clk'event and clk='1' then
  if timecount='1' then                    --判断是否需要时间计费，每20s加一元
   reg2:=cash;
   if reg2(3 downto 0)+"0001">"1001" then
     reg2(7 downto 0):=reg2(7 downto 0)+"00000111";
     if reg2(7 downto 4)>"1001" then
       cash <=reg2+"000001100000";
     else cash<=reg2;
     end if;
   else cash<=reg2+"0001";
   end if;
elsif clkout='1' and enable='1' then          --里程计费
```

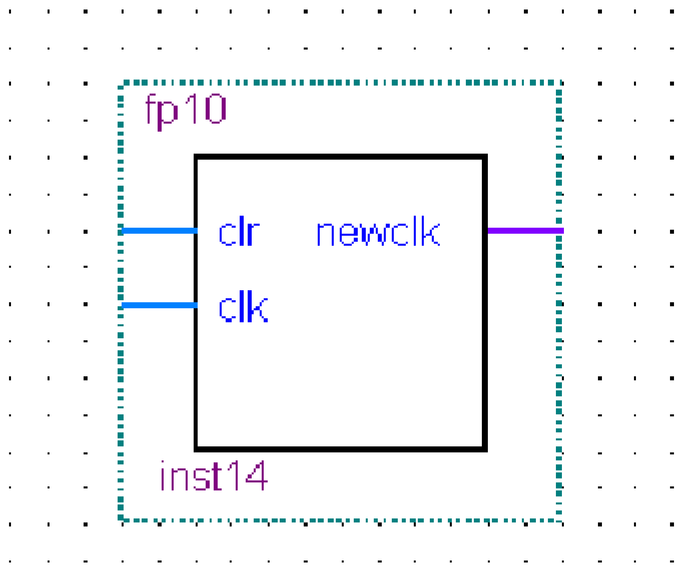
```
 if clkout_cnt=9 then
      clkout_cnt:=0;
      reg2:=cash;
      if"0000"®2(3 downto 0)+price(3 downto 0)>"00001001"then
         reg2(7 downto 0):=reg2(7 downto 0)+"00000110"+price;
         if reg2(7 downto 4)>"1001"then
            cash<=reg2+"000001100000";
         else cash<=reg2;
         end if;
     else cash<=reg2+price;
     end if;
   else clkout_cnt:=clkout_cnt+1;
   end if;
end if;
end if;
end process;
count1<=cash(3 downto 0);    --总费用的个位
count2<=cash(7 downto 4);    --总费用的十位
count3<=cash(11 downto 8);   --总费用的百位
End behav;
```



Name
0 ps
80.0 ns
160.0 ns
240.0 ns
320.0 ns
400.0 ns
480.0 ns
560.0 ns
640.0 ns
720.0 ns
800
780.0
0 clk
1 reset
2 ...ount
3 clkout
4 Kmcnt2
9 Kmcnt3
14 Count3
19 Count2
24 Count1
0011
0011
0000
0000
0001
1000
1001
0001
0011
0101

分频模块
对系统的时钟进行分频，以模拟轮胎的滚动。
（a）100分频　　　(b)10分频

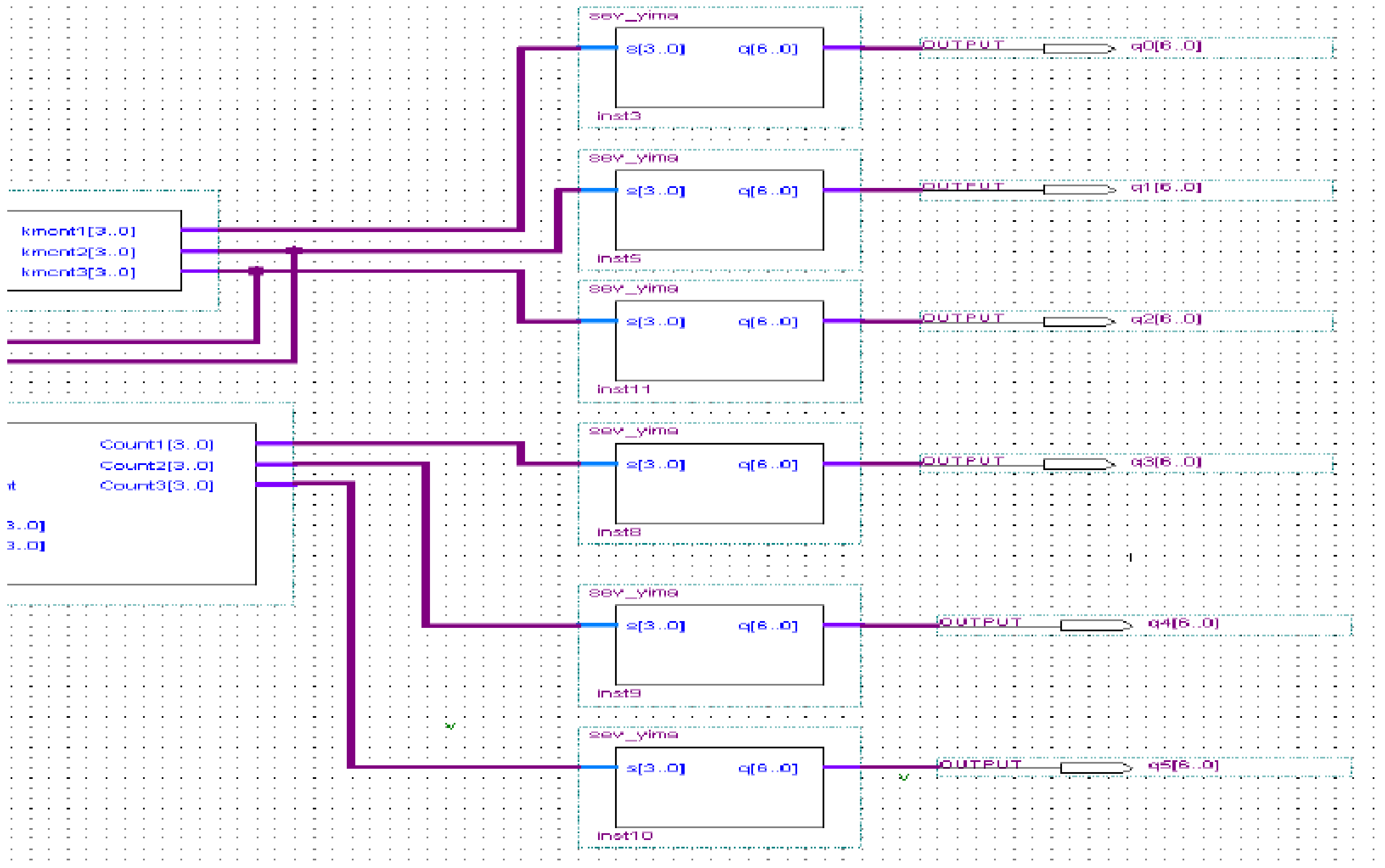




```
Library ieee;
Use ieee.std_logic_1164.all;
entity fp is
  port(clr,clk:in std_logic;
        newclk:out std_logic);
end fp;
architecture behav of fp is
signal tem:integer range 0 to 99;
begin
  process(clk,clr)
  begin
     if(clr='1')then
        tem<=0;
        newclk<='0';
     elsif(clk'event and clk='1')then
     if(tem=99)then
        tem<=0;
        newclk<='1';
      else tem<=tem+1;
        newclk<='0';
      end if;
      end if;
   end process;
end behav;
```

```
Library ieee;
Use ieee.std_logic_1164.all;
entity fp10 is
  port(clr,clk:in std_logic;
        newclk:out std_logic);
end fp10;
architecture behav of fp10 is
signal tem:integer range 0 to 9;
begin
  process(clk,clr)
  begin
     if(clr='1')then
        tem<=0;
        newclk<='0';
     elsif(clk'event and clk='1')then
     if(tem=9)then
        tem<=0;
        newclk<='1';
      else tem<=tem+1;
        newclk<='0';
      end if;
      end if;
   end process;
end behav;
```

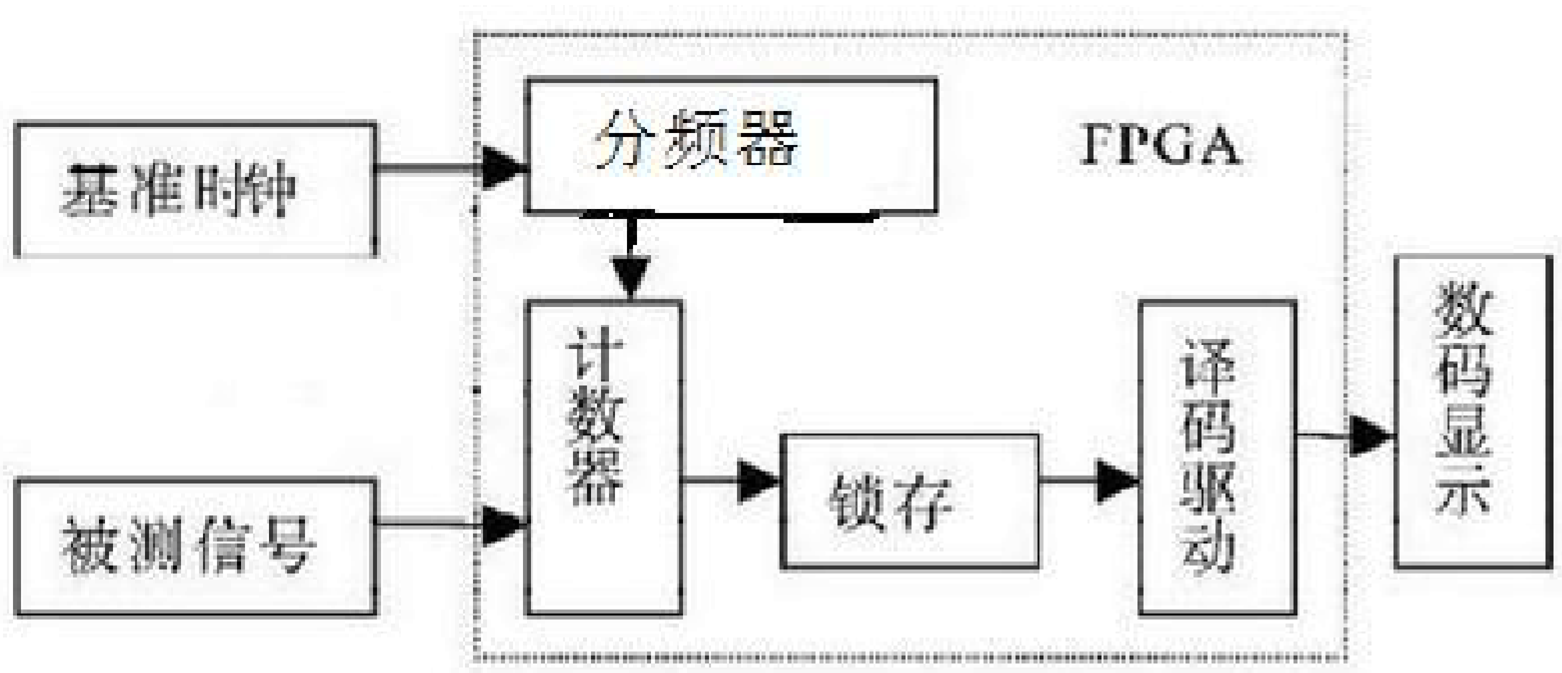
显示模块


sev_yima
s[3..0]
q[6..0]
inst3
inst5
inst11
inst8
inst9
inst10
OUTPUT
q0[6..0]
q1[6..0]
q2[6..0]
q3[6..0]
q4[6..0]
q5[6..0]
kmcnt1[3..0]
kmcnt2[3..0]
kmcnt3[3..0]
Count1[3..0]
Count2[3..0]
Count3[3..0]

```
library ieee;
use ieee.std_logic_1164.all;
entity sev_yima is
port(s:in std_logic_vector(3 downto 0);
   q:out std_logic_vector(6 downto 0));
end sev_yima;
architecture rtl of sev_yima is
begin
   with s select
   q<= "1000000"when "0000",
      "1111001"when "0001",
      "0100100"when "0010",
      "0110000"when "0011",
      "0011001"when "0100",
      "0010010"when "0101",
      "0000010"when "0110",
      "1111000"when "0111",
      "0000000"when "1000",
      "0010000"when "1001",
      "1111111"when others;
end rtl;
```

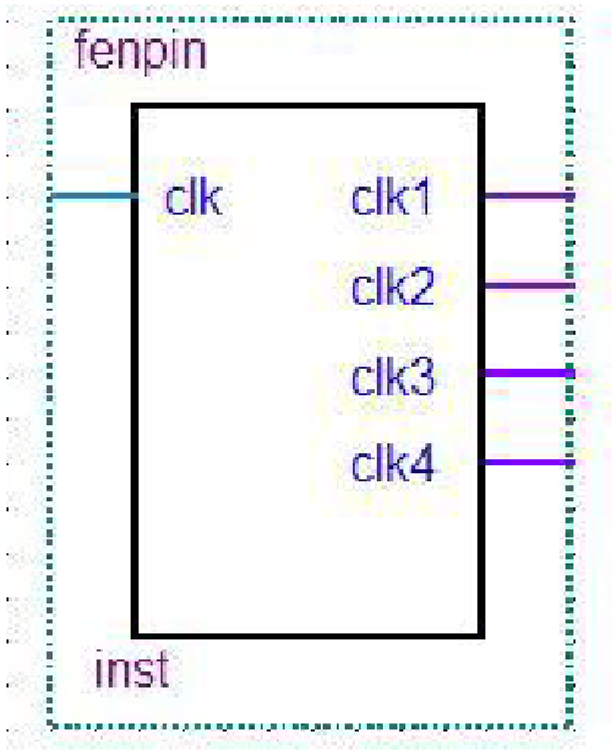
实例3 频率计设计

- 要求：对输入信号进行频率的测量并实时显示。



分频器模块

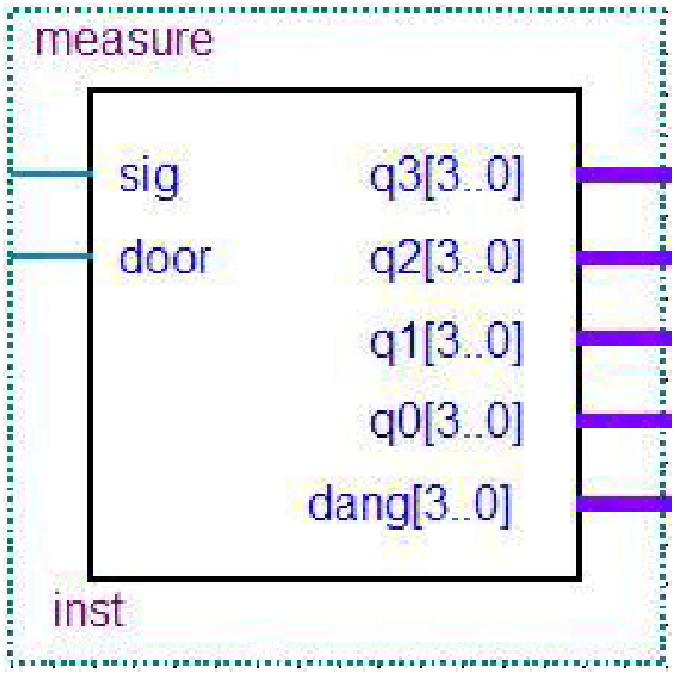
时钟信号源输出的时钟信号频率高达50MHz,经过分频器将其分频为1Hz、4Hz、500Hz和1000Hz时钟信号。这四种信号再经过1/2分频,得到时间基准信号,其中1000Hz的时钟基准信号用于动态扫描译码电路,1Hz的时钟基准信号用于计数器的始能信号。



```
library ieee;
use ieee.std_logic_1164.all;
 entity fenpin is
  port( clk: in  std_logic;--50MHz
       clk1: out std_logic;--1Hz
       clk2: out std_logic;--4Hz
       clk3: out std_logic;--500hz
       clk4: out std_logic --1khz
       );
end fenpin;
architecture arch of fenpin is
begin
process(clk)
   variable  cnt1: integer range 0 to 49999999;
   variable  cnt2: integer range 0 to 12499999;
   variable  cnt3: integer range 0 to 99999;
   variable  cnt4: integer range 0 to 49999;
   variable x1,x2,x3,x4:  std_logic:='0';
begin
   if clk'event and clk='1' then
       if cnt1<49999999 then
          cnt1:=cnt1+1;
       else
          cnt1:=0;   x1:=not x1;
       end if;
       if cnt2<12499999 then
            cnt2:=cnt2+1;
       else
          cnt2:=0;    x2:=not x2;
       end if;
       if cnt2<99999 then
            cnt3:=cnt3+1;
       else
          cnt3:=0; x3:=not x3;
       end if;
       if cnt4<49999 then
            cnt4:=cnt4+1;
       else
          cnt4:=0; x4:=not x4;
       end if;
   end if;
   clk1<=x1;
   clk2<=x2;
   clk3<=x3;
   clk4<=x4;
end process;
end arch;
```

计数器模块

计数器模块始能端door输入分频器模块分频出的0.5Hz 为基准时钟信号频率。计数器sig输入待测信号与基准时钟信号进行比较，并且计数。q0[3..0] 对应的是个位输出，q1[3..0] 对应的是十位输出，q2[3..0] 对应的是百位输出，q3[3..0] 对应的是千位输出。


measure
sig
door
q3[3..0]
q2[3..0]
q1[3..0]
q0[3..0]
dang[3..0]
inst