# 汽车尾灯控制电路

### 设计要求

设计一个汽车尾灯的控制电路，要求如下：

1. 假设汽车尾部左右两侧各有3个指示灯(用发光二极管模拟)。
2. 汽车左转弯时,左侧3个指示灯按左循环顺序点亮,汽车右转弯时,右侧3个指示灯按右循环顺序点亮,临时刹车和倒车时所有指示灯同时亮。
3. 设置两个可控制的开关,可产生00、 01、 10、11四种状态：

开关置为00状态时,表示汽车处于正常运行状态。

开关置为01状态时,表示汽车处于右转弯状态。

开关置为10状态时,表示汽车处于左转弯状态。

开关置为11状态时,表示汽车处于刹车状态。

另设一开关表示汽车运行状态, 0表示向前运行, 1表示向后倒车。

(4) 拓展功能：

设计左转、右转、刹车、倒车不同状态下对应数码管的显示图案并展示。

### 工作原理

i、根据设计要求，汽车尾灯控制电路的工作过程可按如下逻辑功能表进行描述：

说明:

左转时，左侧3个指示灯按左循环顺序点亮，右转时，右侧3个指示灯按右循环顺序点亮,临时刹车和倒车时所有指示灯同时亮。

功能表中“1”表示指示灯亮，“0”表示指示灯灭。

tleft、tright为左转、右转和刹车控制键，mark为倒车控制键。

 hex0~hex3 对应的为数码管的编号。

图2.1 汽车尾灯控制电路逻辑功能表

ii、总体原理框图

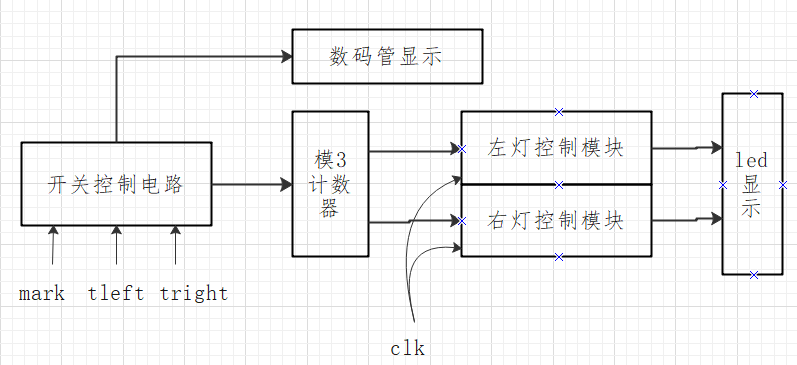


图2.2 原理框图

### 电路设计

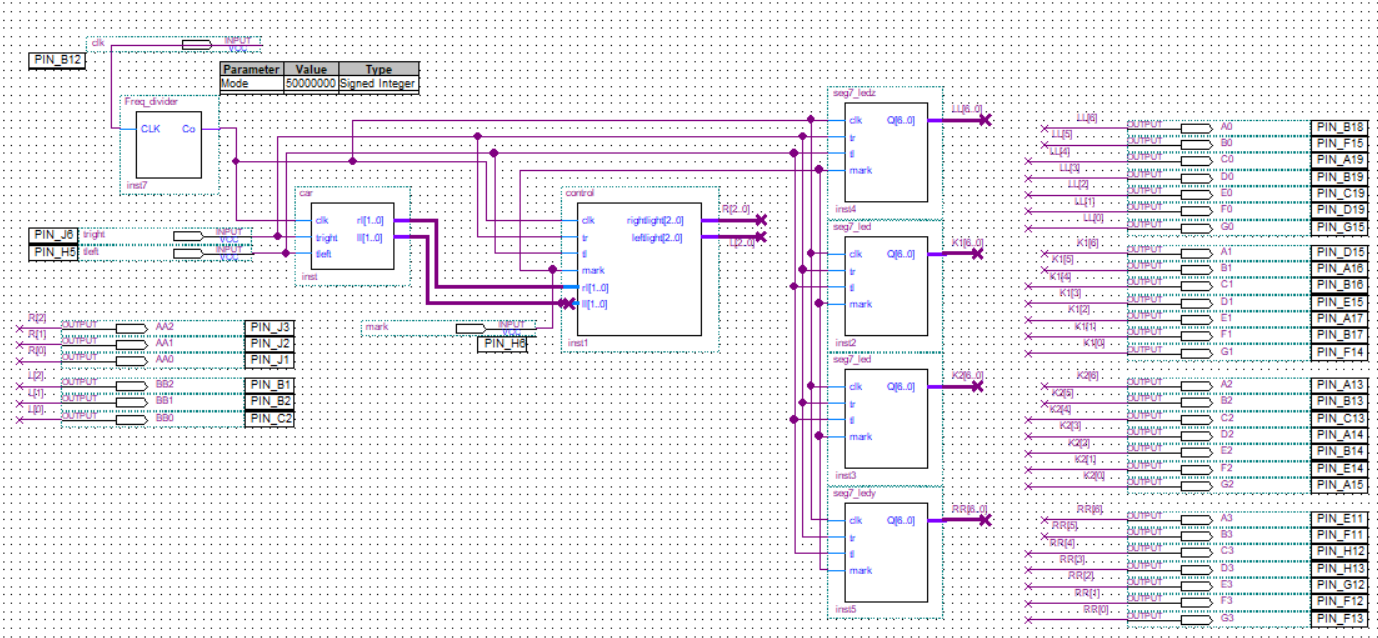
为了实现题目要求，顶层电路设计如图3.1所示：

图3.1 汽车尾灯控制电路顶层电路图

如图3.1所示，整个系统包括四个模块，分别是分频器Freq\_divider、模三计数器car、控制及显示电路control、数码管显示译码电路seg7\_led、seg7\_ledy、seg7\_ledz。

其中，分频器负责将系统的输入时钟CLK变成1Hz；模三计数器负责产生3种状态信号；控制及显示电路实现led灯在不同状态下的显示，数码管显示译码电路实现4位BCD码到7段共阳数码管（共4个）的译码与显示。

**3.1 分频器**

分频器在本质上是一个计数器，只是输出时，需要构造出占空比接近50%的脉冲。在本系统中，开发板板载时钟为50MHz，要生成1Hz的计时时钟，分频系数为：

|  |  |
| --- | --- |
|  | (式1) |

所以，需要构造一个模值为50,000,000的计数器，在0-24,999,999时输出低电平，在25,999,999-49,999,999输出高电平，对应的VHDL代码如下：

|  |
| --- |
| **Freq\_divider.vhd 分频器** |
|  |

**3.2 模三计数器**

当时钟信号clk上升沿有效时，分别控制右信号rlight和左信号llight，使其在01、10、11三种状态间转换，模三计数器的核心代码如下：

|  |
| --- |
| **car.vhd 模三计数器** |
|  |

**3.3 控制及显示电路**

tleft、tright作为控制信号，控制尾灯不同状态下的显示，rl、ll为模三计数器的输出信号，可以控制左转或右转时3个尾灯向左或向右循环亮，当刹车状态或者mark为“1”时，6个尾灯全亮，正常行驶时，6个尾灯全灭，控制及显示电路的核心代码如下：

|  |
| --- |
| **control.vhd 控制及显示电路** |
|  |

**3.4 数码管显示译码电路**

显示译码器实现4位BCD码到7段共阳数码管的译码与显示，核心代码为：

|  |
| --- |
| **seg7\_led.vhd、seg7\_ledy.vhd、seg7\_ledz.vhd 数码管显示译码电路** |
| （由于4个数码管中，中间两个数码管的图案相同，故此处设计三种不同的数码管译码电路） |

### 仿真结果

在Win10 64b的操作系统中，基于Quartus II 9.0 32b软件，对汽车尾灯控制电路，进行了设计与仿真。

**4.1 分频器仿真**

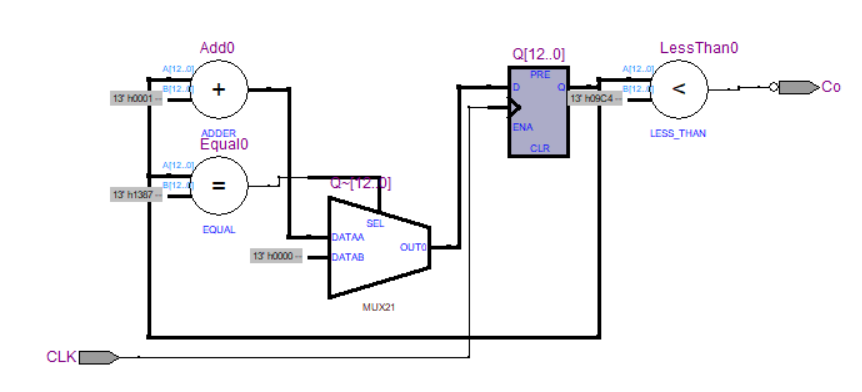
为了缩短仿真耗时，在设计文件**Freq\_divider.vhd**中，将分频系数Mode设为5000，而输入时钟CLK保持50MHz（周期20ns）保持不变。因为至少要观察两个输出周期的波形，所以仿真时长不能少于20ns×5000×2=200us。

新建**Freq\_divider.vwf**文件，在Edit下拉菜单中选择End Time…，将Time设置为1ms。其他仿真参数的设置见表4.1。

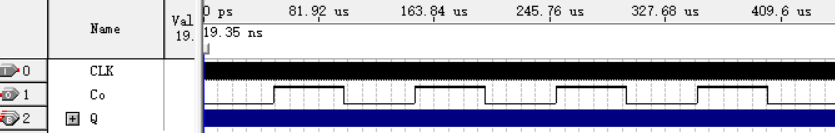
表4.1 仿真参数设置

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **参数设置** | **含义** |
| CLK | INPUT | 20ns（DE0开发板-50MHz） | 系统时钟 |
| Co | OUTPUT |  | 分频输出 |
| Q | REGISTER | Unsigned Decimal | 计数指示 |

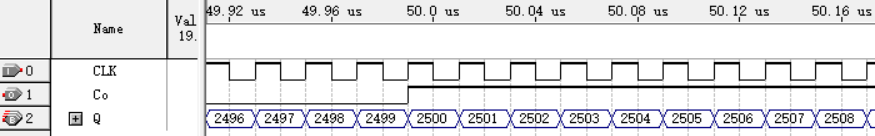
分频器RTL图如下：



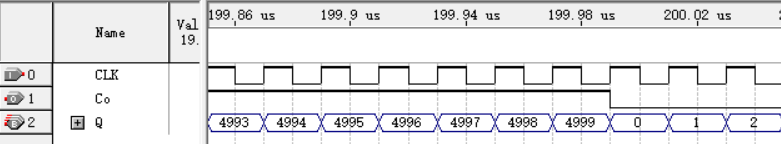
仿真结果如图4.1所示：



(a)



(b)



(c)

图4.1 分频器的功能仿真结果. (a)波形总览 (b)上升沿 (c)下降沿

从图4.1的波形可以看出，当分频系数Mode=5000时，分频后输出的Co为占空比50%的方波，在Q=0-2499时为低电平，在Q=2500-4999时为高电平。

分频器模块符合既定的设计目标。

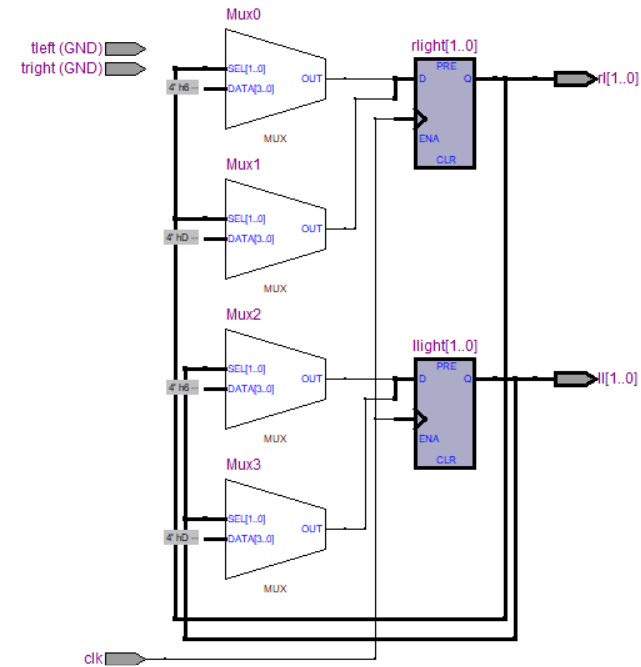
**4.2 模三计数器**

模三计数器的仿真设置和仿真结果如下：

表4.2 仿真参数设置

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **参数设置** | **含义** |
| CLK | INPUT | 20ns | 工作时钟 |
| rlight | OUTPUT |  | 右转计数器 |
| llight | OUTPUT |  | 左转计数器 |

模三计数器RTL图如下：



仿真结果如图4.2所示：

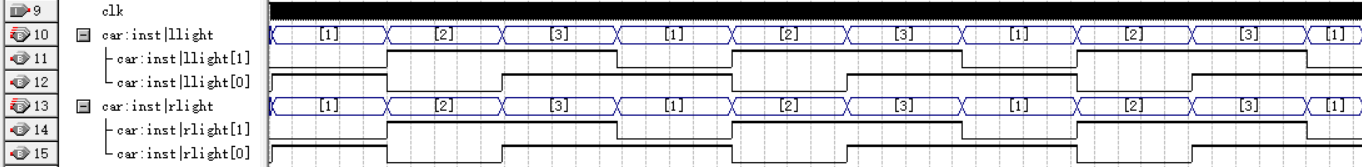


图4.2 模三计数器的功能仿真结果

从图4.2可以看出，在时钟上升沿有效时，分别控制右信号rlight和左信号llight，使其在01、10、11三种状态间转换，所以，此模块符合既定的设计目标。

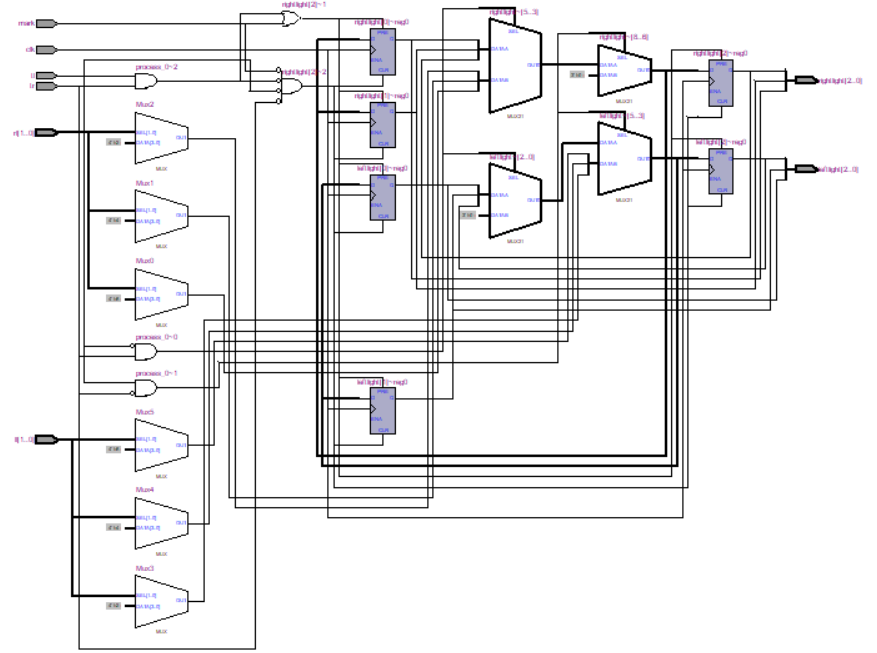
**4.3 控制及显示电路**

控制及显示电路的仿真设置和结果如下：

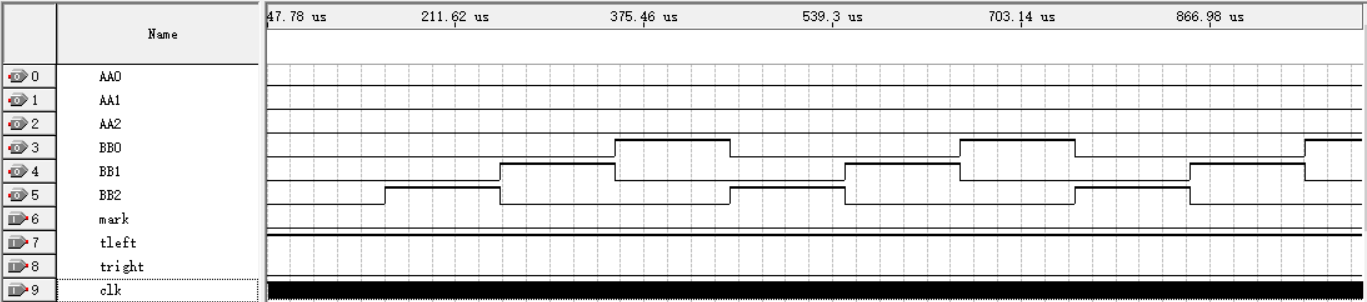
表4.3 仿真参数设置

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **参数设置** | **含义** |
| tleft | INPUT | H/L | H-左转/刹车，L-右转/正常 |
| tright | INPUT | H/L | H-右转/刹车，L-左转/正常 |
| mark | INPUT | H/L | H-倒车，L-正常前进 |
| CLK | INPUT | 20ns | 工作时钟 |
| rl 1/0 | INPUT |  | 3种状态信号 |
| Ll 1/0 | INPUT |  | 3种状态信号 |
| BB0~BB2 | OUTPUT | H/L | H-led灯亮，L-led灯亮（控制左边3个led灯） |
| AA0~AA2 | OUTPUT | H/L | H-led灯亮，L-led灯亮（控制右边3个led灯） |

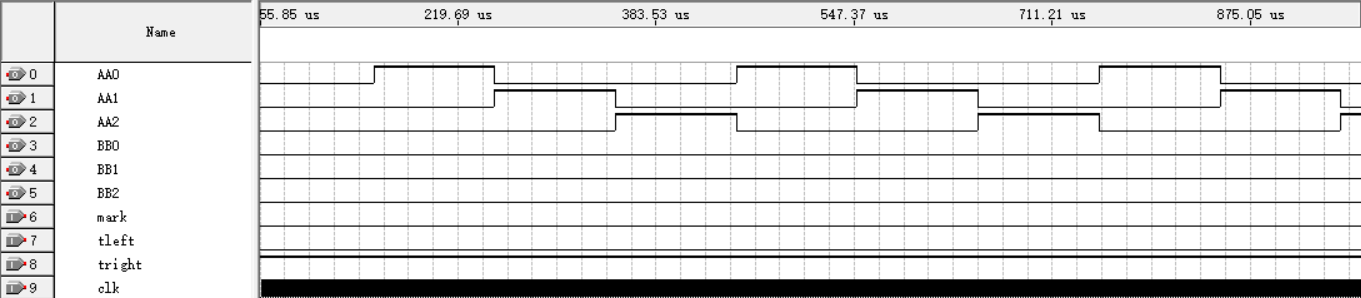
控制及显示电路的RTL图如下：



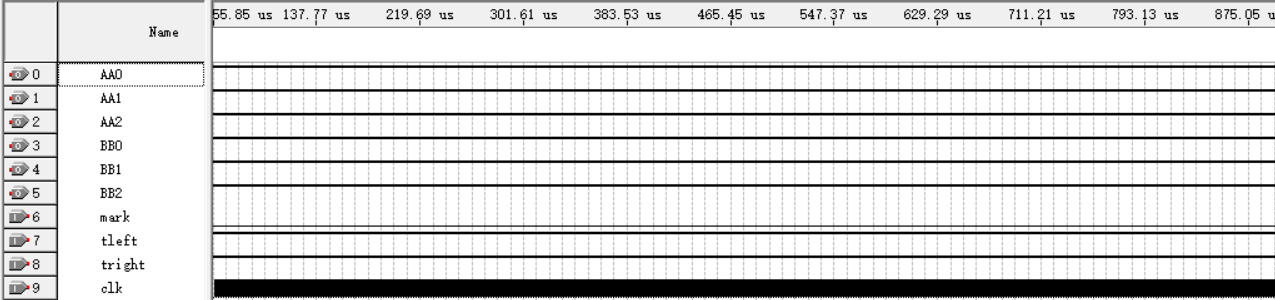
仿真结果如图4.4所示：



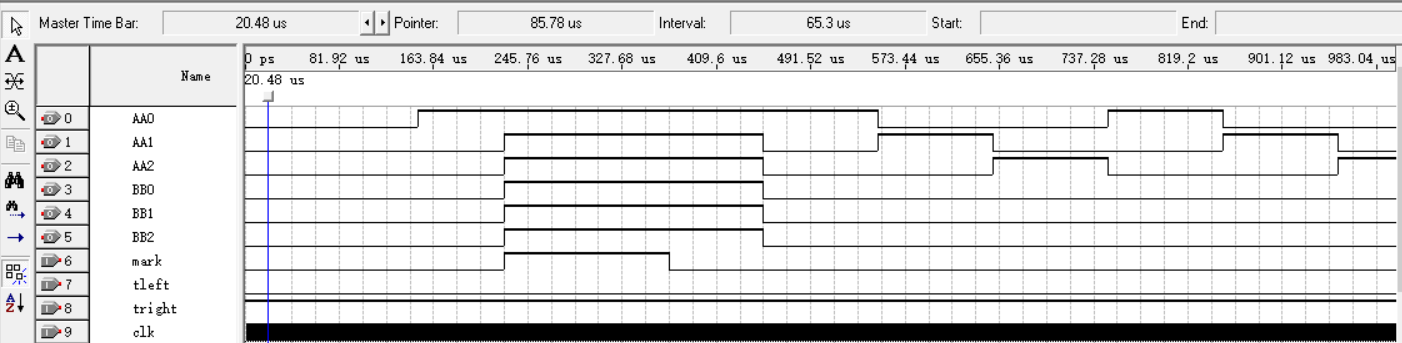
(a) tleft = ‘1’ tright = ‘0’ mark = ‘0’



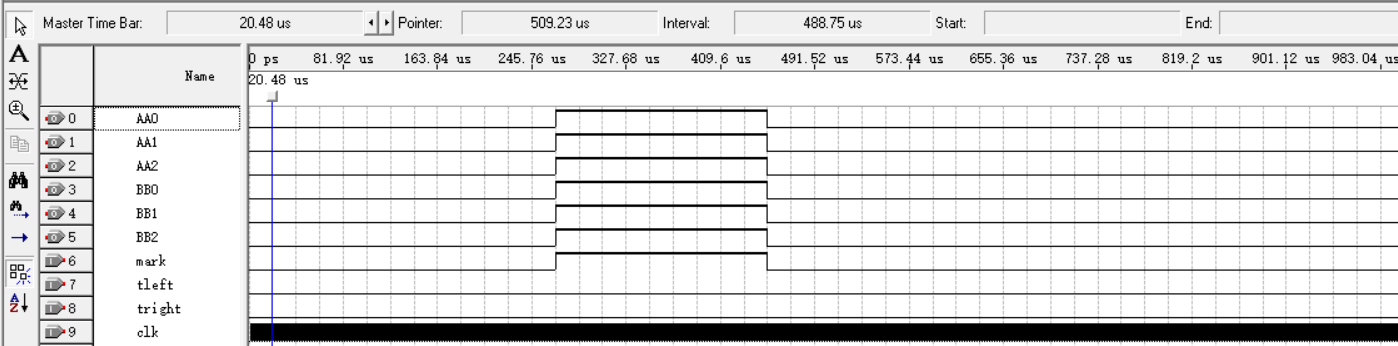
(b) tleft = ‘0’ tright = ‘1’ mark = ‘0’



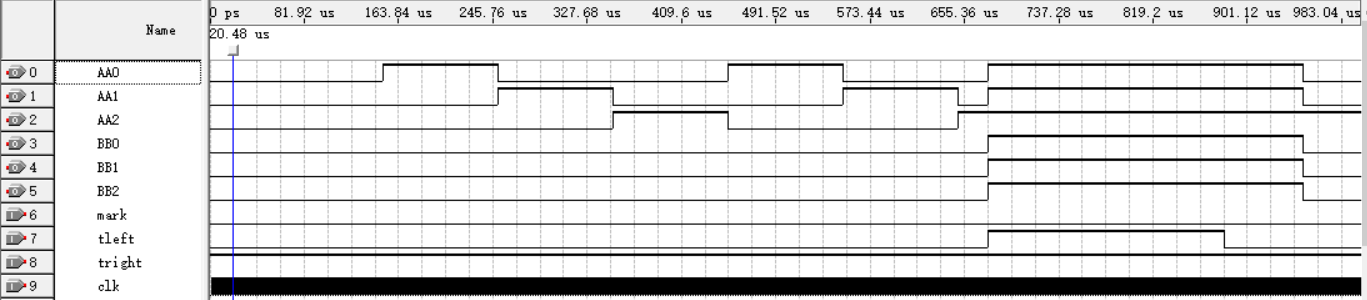
（c）tleft = ‘1’ tright = ‘1’ mark = ‘0’



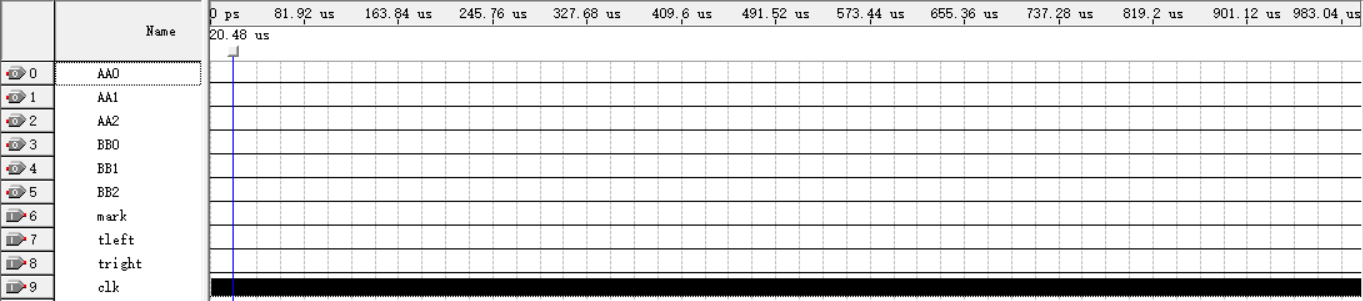
(d) tleft = ‘0’ tright = ‘1’ mark = ‘1’



（e）tleft = ‘0’ tright = ‘0’ mark = ‘1’



(f) tleft = ‘1’ tright = ‘1’ mark = ‘0’



（g）tleft = ‘0’ tright = ‘0’ mark = ‘0’

图4.3 控制及显示电路的功能仿真结果.

(a)左转 (b)右转(c)刹车(d)右转+倒车(e)倒车(f)紧急刹车(g)正常前进

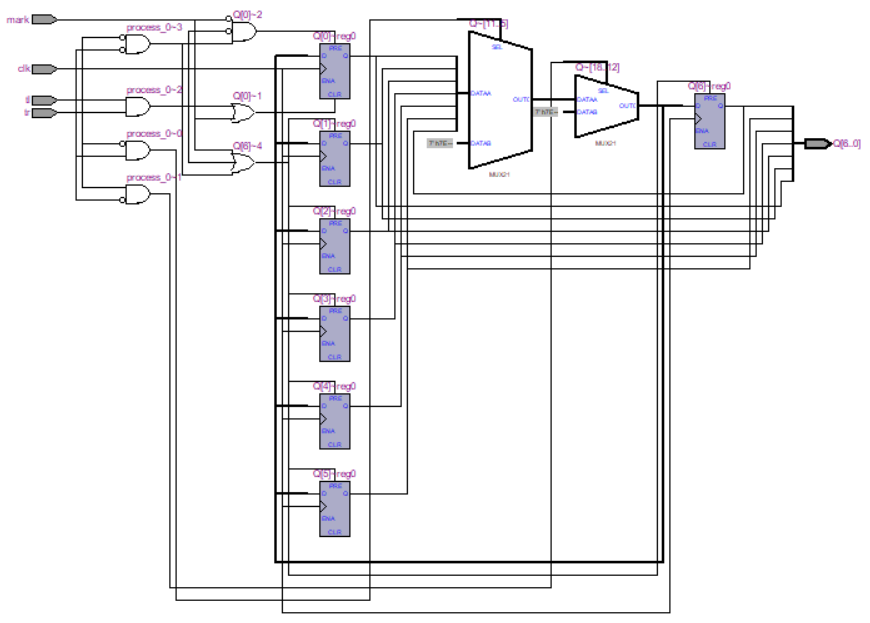
**4.4 显示译码器仿真**

显示译码器的仿真设置和结果如下：

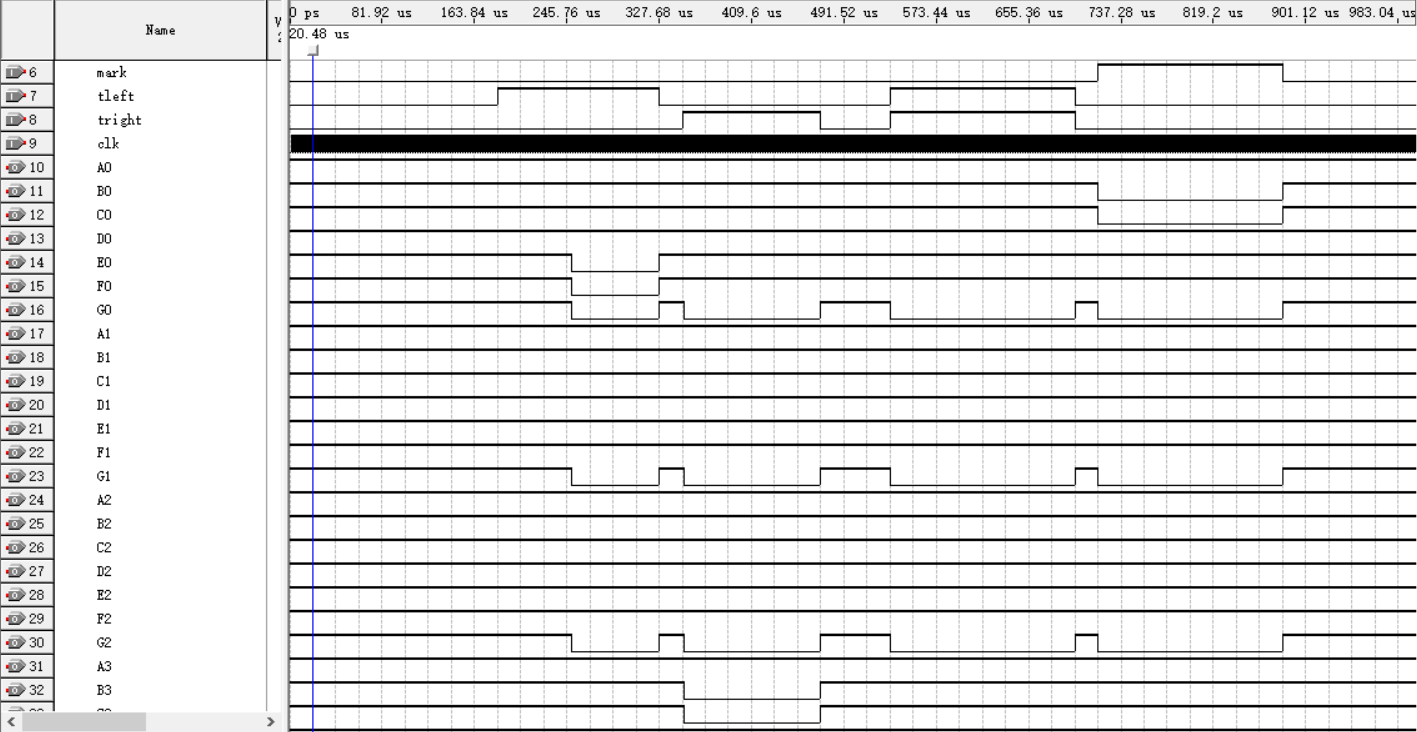
表4.4 仿真参数设置

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **参数设置** | **含义** |
| tleft | INPUT | H/L | H-左转/刹车，L-右转/正常 |
| tright | INPUT | H/L | H-右转/刹车，L-左转/正常 |
| mark | INPUT | H/L | H-倒车，L-正常前进 |
| CLK | INPUT | 20ns | 工作时钟 |
| A0~G0 | OUTPUT |  | 对应7段共阳LED  的a-g(hex3) |
| A1~G1 | OUTPUT |  | 对应7段共阳LED  的a-g(hex2) |
| A2~G2 | OUTPUT |  | 对应7段共阳LED  的a-g(hex1) |
| A3~G3 | OUTPUT |  | 对应7段共阳LED  的a-g(hex0) |

显示译码器的RTL图如下（以seg7\_led为例）：



仿真结果如图4.5所示：



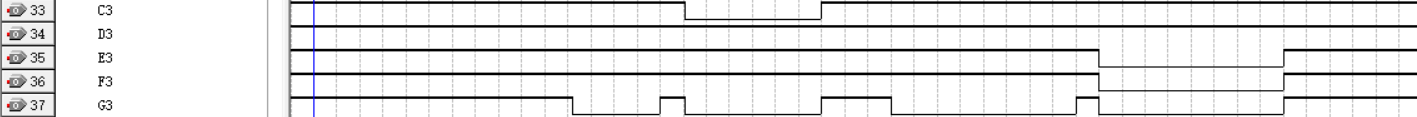


图4.4 显示译码器的功能仿真结果

从图4.4中可以看出，

当正常行驶时数码管均为高电平；

当左转时，E0、F0、G0、G1、G2、G3为低电平；

当右转时，B3、C3、G0、G1、G2、G3为低电平；

当刹车时，G0、G1、G2、G3为低电平；

当倒车时，B0、C0、E3、F3、G0、G1、G2、G3为低电平。

符合既定的设计目标。

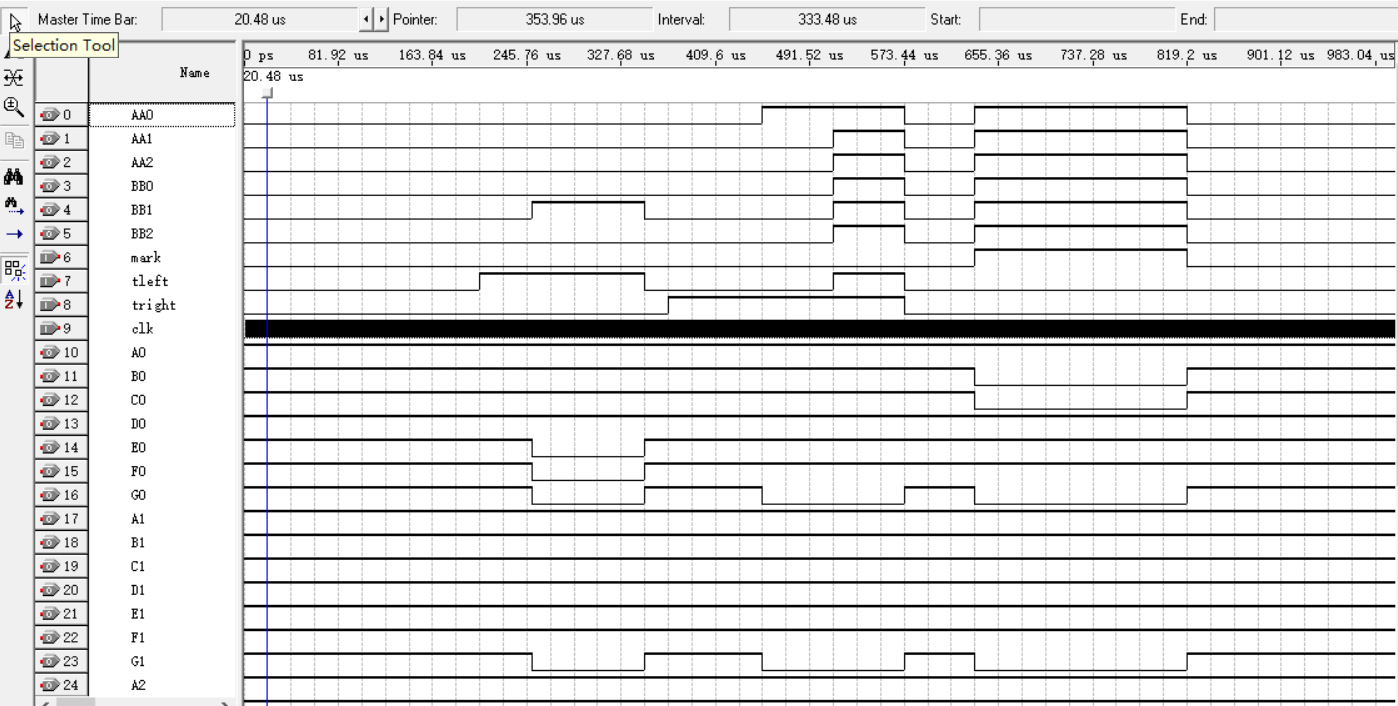
**4.5 系统顶层电路仿真**

汽车尾灯控制电路的仿真设置和结果如下：

表4.5 仿真参数设置

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **类型** | **参数设置** | **含义** |
| CLK | INPUT | 20ns | 系统时钟 |
| tleft | INPUT | H/L | H-左转/刹车，L-右转/正常 |
| tright | INPUT | H/L | H-右转/刹车，L-左转/正常 |
| mark | INPUT | H/L | H-倒车，L-正常前进 |
| BB0~BB2 | OUTPUT | H/L | H-led灯亮，L-led灯亮（控制左边3个led灯） |
| AA0~AA2 | OUTPUT | H/L | H-led灯亮，L-led灯亮（控制右边3个led灯） |
| A-G 3/2/1/0 | OUTPUT |  | 7段共阳LED的a-g |

仿真结果如图4.6所示：



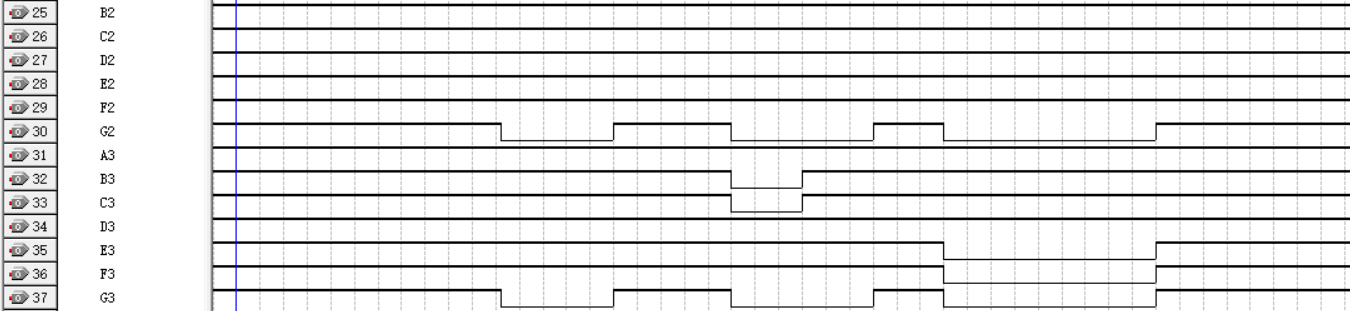


图4.5 汽车尾灯控制电路的功能仿真结果

整体RTL图如下：

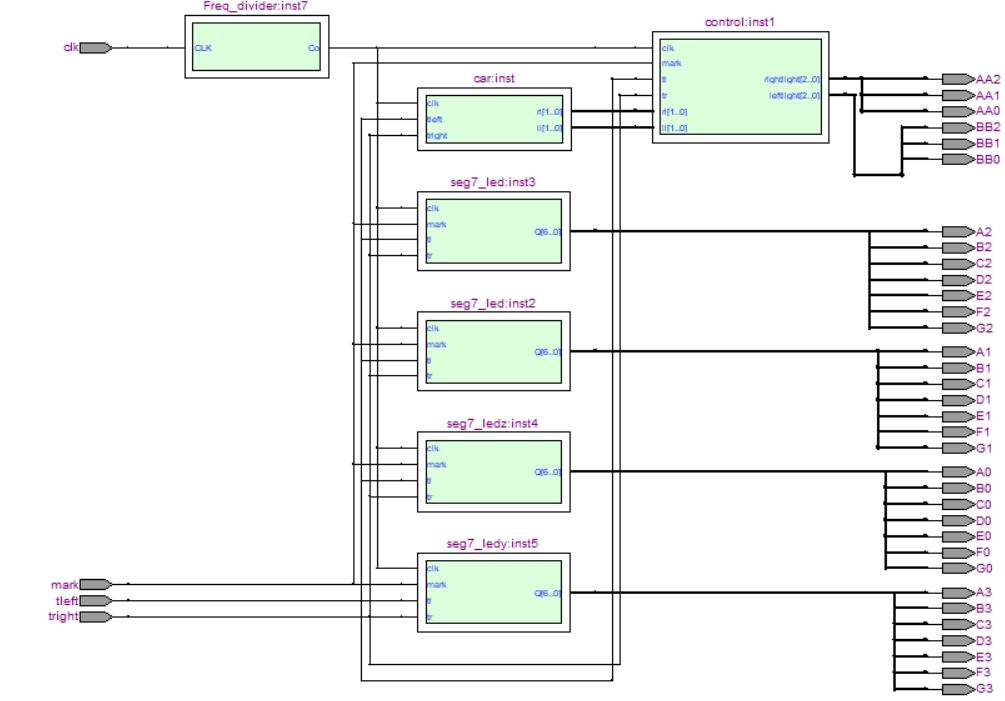


图4.6 RTL图

### 功能扩展

结合生活中的实际情况，为进一步提高汽车尾灯控制电路的实用性，还可以在以下几个方面进行功能扩展：

1. 加入夜间行驶信号控制的led指示灯，在夜间行驶时保持led灯常亮；
2. 可以加入双闪控制信号，控制led灯闪烁式点亮；
3. 为避免刹车和倒车时控制led灯的状态相同，可以在倒车时再加入两个led灯辅助显示，以和刹车状态相区分。

### 学习小结

通过本课程的学习，我深深的体会到它的重要性和目的性。本次设计课不仅仅培养了我的实际操作能力，也培养了我灵活运用课本知识，理论联系实际，独立自主进行设计的能力。它不仅仅是一个学习新知识新方法的好机会，同时也是对我所学知识的一次综合的检验和复习，使我明白了自己的缺陷所在，从而查漏补缺。 并且，在设计、实现、仿真和硬件下载运行汽车尾灯控制电路的过程中，我更加深入地认识到“纸上得来终觉浅，绝知此事要躬行”。理论学习结合实践操作，才能更加深刻地体会到电路方案的从无到有，从有到优的设计思路和实现方法。

### 参考资料

1. 杨颂华等，《数字电子技术基础》(第三版)，西安电子科技大学出版社，2016
2. 侯伯亨等，《VHDL硬件描述语言与数字逻辑电路设计》(第三版)，西安电子科技大学出版社，2009
3. 陈欣波等，《基于FPGA的现代数字电路设计》，北京理工大学出版社，2018
4. 孙万蓉等，中国大学MOOC国家精品课程，“数字电路与系统设计”，[https://www.icourse163.org/learn/XDU-1002203001?tid=1450595469#/learn/announce](https://www.icourse163.org/learn/XDU-1002203001?tid=1450595469" \l "/learn/announce)
5. 学在西电在线课程 <http://mooc1.chaoxing.com/course/214280880.html>

### 附录

**附1 模3计数器代码**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity car is

port(clk,tright,tleft:IN std\_logic;

rl:out std\_logic\_vector(1 downto 0);

ll:out std\_logic\_vector(1 downto 0));

end entity car;

ARCHITECTURE bhv of car IS

signal llight:std\_logic\_vector(1 downto 0);

signal rlight:std\_logic\_vector(1 downto 0);

begin

process(clk,tleft,tright)

begin

if clk'event and clk='1'then

case rlight is

when "01"=>rlight<="10";

when "10"=>rlight<="11";

when "11"=>rlight<="01";

when others=>rlight<="01";

end case;

case llight is

when "01"=>llight<="10";

when "10"=>llight<="11";

when "11"=>llight<="01";

when others=>llight<="01";

end case;

end if;

end process;

rl <=rlight;

ll <=llight;

END bhv;

**附2 分频器代码**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY Freq\_divider IS

GENERIC (Mode: INTEGER :=50000000); -- Normal

PORT(CLK : IN STD\_LOGIC;

Co: OUT STD\_LOGIC);

END Freq\_divider;

ARCHITECTURE bhv OF Freq\_divider IS

SIGNAL Q: INTEGER RANGE 0 TO Mode-1:=0; --Normal

BEGIN

PROCESS(CLK)

BEGIN

IF CLK'EVENT AND CLK = '1' THEN

IF Q = Mode-1 THEN Q <= 0;

ELSE

Q <= Q + 1 ;

END IF;

**附3 控制及显示电路代码**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity control is

port(clk,tr,tl,mark:IN std\_logic;

rl,ll:in std\_logic\_vector(1 downto 0);

rightlight:out std\_logic\_vector(2 downto 0);

leftlight:out std\_logic\_vector(2 downto 0));

end entity control;

architecture behave of control is

begin

process(clk,tr,tl,rl,ll)

begin

if clk'event and clk='1'then

if tl='0'and tr='1'then

leftlight<="000";

case rl is

when"01"=>rightlight<="001";

when"10"=>rightlight<="010";

when"11"=>rightlight<="100";

when others=>rightlight<="000";

end case;

end if;

if tl='1'and tr='0'then

rightlight<="000";

case ll is

when"01"=>leftlight<="100";

when"10"=>leftlight<="010";

when"11"=>leftlight<="001";

when others=>leftlight<="000";

end case;

end if;

end if;

if tl='1'and tr='1'then

leftlight<="111";

rightlight<="111";

elsif tl='0'and tr='0'then

leftlight<="000";

rightlight<="000";

end if;

if mark='1'then

leftlight<="111";

rightlight<="111";

end if;

end process;

end behave;

**附4 显示译码器代码**

**(1)、**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity seg7\_led is

port(clk,tr,tl,mark:IN std\_logic;

Q: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

end entity seg7\_led;

architecture behave of seg7\_led is

begin

process(clk,tr,tl)

begin

if clk'event and clk='1'then

if tl='0'and tr='1'then

Q<="1111110";

end if;

if tl='1'and tr='0'then

Q<="1111110";

end if;

end if;

if tl='1'and tr='1'then --sha che

Q<="1111110";

elsif tl='0'and tr='0'then

Q<="1111111";

end if;

if mark='1'then --dao che

Q<="1111110";

end if;

end process;

end behave;

**（2）、**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity seg7\_ledy is

port(clk,tr,tl,mark:IN std\_logic;

Q: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

end entity seg7\_ledy;

architecture behave of seg7\_ledy is

begin

process(clk,tr,tl)

begin

if clk'event and clk='1'then

if tl='0'and tr='1'then

Q<="1001110";

end if;

if tl='1'and tr='0'then

Q<="1111110";

end if;

end if;

if tl='1'and tr='1'then --sha che

Q<="1111110";

elsif tl='0'and tr='0'then

Q<="1111111";

end if;

if mark='1'then --dao che

Q<="1111000";

end if;

end process;

end behave;

**（3）、**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity seg7\_ledz is

port(clk,tr,tl,mark:IN std\_logic;

Q: OUT STD\_LOGIC\_VECTOR(6 DOWNTO 0));

end entity seg7\_ledz;

architecture behave of seg7\_ledz is

begin

process(clk,tr,tl)

begin

if clk'event and clk='1'then

if tl='0'and tr='1'then

Q<="1111110";

end if;

if tl='1'and tr='0'then

Q<="1111000";

end if;

end if;

if tl='1'and tr='1'then --sha che

Q<="1111110";

elsif tl='0'and tr='0'then

Q<="1111111";

end if;

if mark='1'then --dao che

Q<="1001110";

end if;

end process;

end behave;