**EDA大作业**

****

学生姓名： 吴程锴

学 号： 18029100040

班 级： 1802015

授课教师： 宗 汝

提交日期：2020年11月12日

目录

[一、 实验目的 1](#_Toc59352929)

[二、 设计任务 1](#_Toc59352930)

[三、 程序设计 1](#_Toc59352931)

[3.1 分频器 1](#_Toc59352932)

[3.2 45秒倒计时 2](#_Toc59352933)

[3.3 数码管译码器 3](#_Toc59352934)

[3.4 总控制器 4](#_Toc59352935)

[3.5 顶层设计 5](#_Toc59352936)

[3.6 引脚分配 6](#_Toc59352937)

[四、 波形仿真 7](#_Toc59352938)

[4.1 引脚说明 8](#_Toc59352939)

[4.2 无紧急状况 8](#_Toc59352940)

[4.3 出现紧急状况 9](#_Toc59352941)

[五、 分析与总结 9](#_Toc59352942)

[六、 附录 10](#_Toc59352943)

# 实验目的

利用所学的EDA知识，制作一个交通控制器 。

# 设计任务

设计一个十字路口交通控制系统，其东西、南北两个方向除了有红、黄、绿灯指示是否允许通行外，还设有时钟，以倒计时方式显示每一路允许通行的时间，绿灯，黄灯，红灯的持续时间分别是40、5和45秒，且在绿灯的最后5秒让绿灯闪烁，提醒司机将要不能通行，红灯的最后5秒让黄灯也亮，提醒司机准备通行。当东西或南北两路中任一道上出现紧急情况，此时交通控制系统应可进入紧急状态，即两条道上的所有车辆皆停止通行，红灯全亮，时钟停止计时，且其数字在闪烁。当紧急状态结束后，控制系统恢复原来的状态，继续正常运行。

# 程序设计

## 分频器

**输入**：时钟信号，重置信号。

**输出**：分频时钟信号。

**功能**：50MHz分频。

由于板子的时钟频率为50MHz，为便于观察，需要设计一个50M的分频器，输出一个1Hz的时钟信号。

分频器其实就是计数器，只要计数器计数到某一模值时，把计数器清零，并且对输出进行反转即可。

分频器模值、系统时钟和期望输出时钟频率关系为



所以，把50MHz时钟分频，输出1Hz的时钟，分频器的模值为



为了保证分频器正常工作，计数器寄存器所能表示的最大值必须大于分频器的模值。这里，设置把计数器寄存器的位数设定为32位，计数器寄存器可表示的最大数值为。具体代码见附录一。流程图如图 1所示。

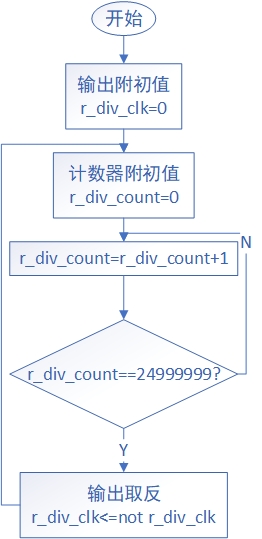


图 1分频器

## 45秒倒计时

**输入**：时钟信号，重置信号，紧急信号

**输出**：十位8421BCD码，个位8421BCD码，交换通行信号

**功能**：44计数到0，可通过紧急信号暂停计数或通过重置信号使计数清零。

一个方向的红绿灯红灯45秒，绿灯40秒，黄灯5秒，一直重复。我设计了一个45秒计时器。当紧急信号为0时，计时器从45倒计时到0，当紧急信号为1时，计时器停止计时。倒计时过程中输出倒计时的十位和个位的8421BCD码。当计数器计数到0时，输出一个周期高电平，代表需要交换通行方向。具体代码见附录二。流程图如图 2所示。

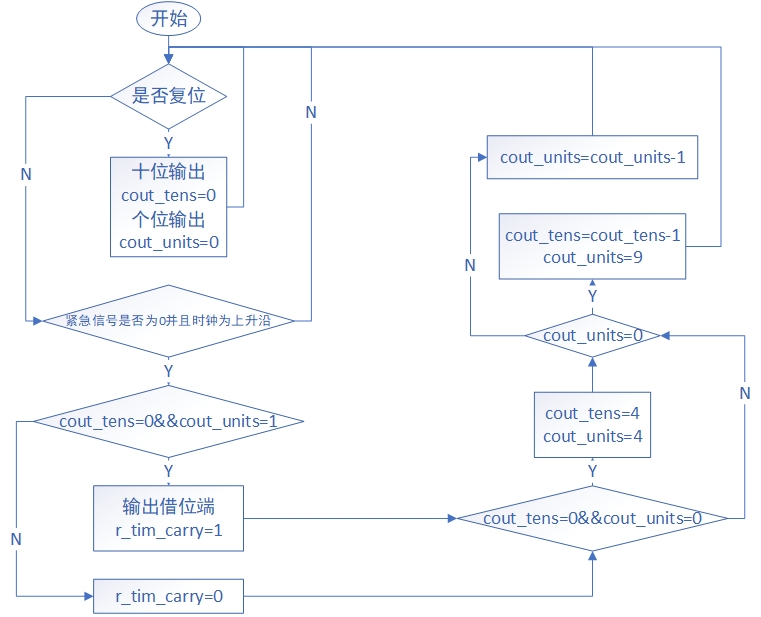


图 2 45秒倒计时

## 数码管译码器

**输入**：数字的8421BCD码，重置信号，使能信号。

**输出**：数码管7段编码。

**功能**：把8421BCD码转化为数码管编码输出。

为了实现输入8421BCD码，时数码管显示相应的数字，需要对8421BCD码进行译码，使用switch-case语句，当输入一个8421BCD码时，输出相应的数码管的编码。输入还有一个使能信号，当使能为1时，输出对应的数码管编码，当是能为0时，输出全1，时数码管熄灭。具体代码见附录三。流程图如图 3所示。

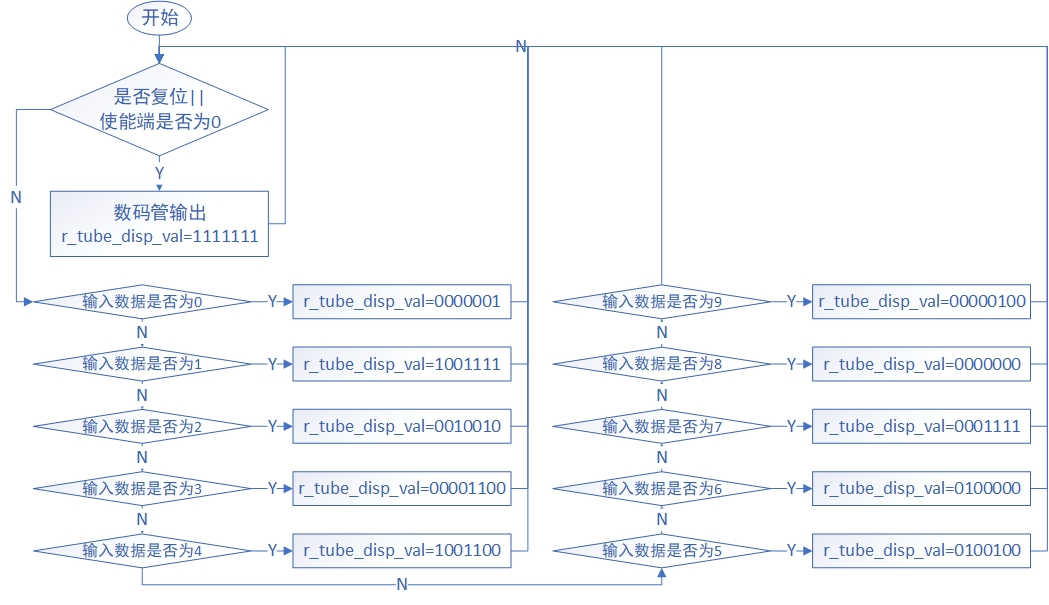


图 3数码管显示

## 总控制器

**输入**：重置信号，时钟信号，倒计时十位8421BCD码，倒计时个位8421BCD码，交换通行信号，紧急信号。

**输出**：东西向十位8421BCD码，东西向个位8421BCD码，南北向十位8421BCD码，南北向个位8421BCD码，数码管使能信号，两个方向的红黄绿信号。

**功能**：根据从倒计时模块输入的数值和交换通行信号和规则以及紧急信号，按照规则控制6个灯和4个数码管的亮灭和闪烁。

主要使用if语句实现。

设一个允许通行方向标志位，当标志位为1时，表示东西方向允许通行，当标志位为0时，表示南北方向允许通行，标志位的切换由45秒计时器输出的交换通行信号的下降沿来决定，当45秒计时器输出一个周期高电平，下降沿到达时标志位取反。

当紧急信号为0时，在允许通行的方向的45秒内，前40秒绿灯亮，数码管使能端为1，输出的显示从39倒计时到0的十位和个位的8421BCD码；在最后5秒，只有黄灯亮，数码管使能端为时钟信号，输出从4倒计时到0的十位和个位的8421BCD码。在禁止通行的方向的45秒内，输出44倒计时到0的十位和个位的8421BCD码；在前40秒，红灯常亮，数码管使能端为1；在最后5秒，红灯和黄灯都亮，数码管使能端为时钟信号，让数码管闪烁。

当紧急信号为1时，数码管使能端和两个方向的红灯均为时钟信号，让他们闪烁，其他灯熄灭。

具体代码见附录四。流程图如图 4所示。

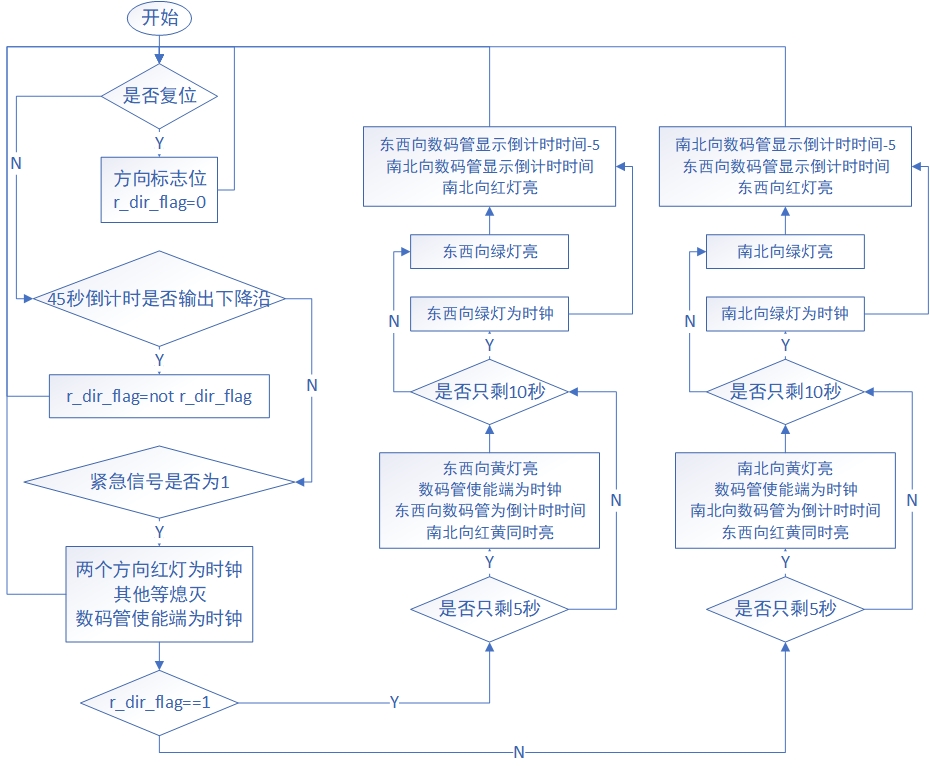


图 4总控制器

## 顶层设计

把以上四个代码生成模块，并按照逻辑连接，连接如图 5所示

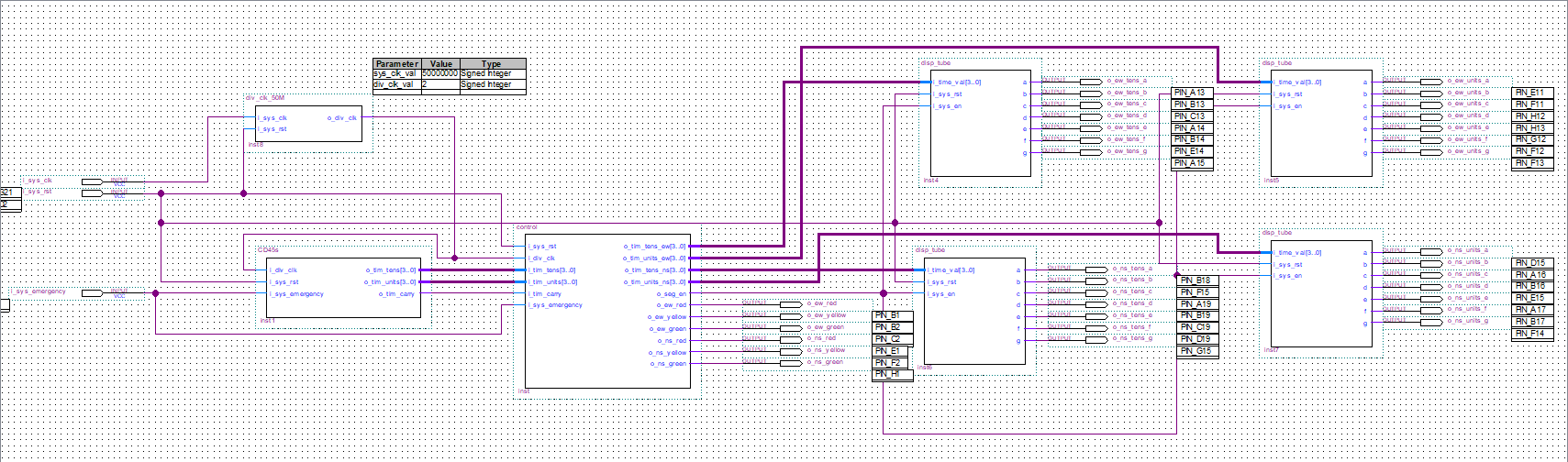


图 5顶层设计

由于太长，把图从中间截为两张，如图 6和图 7所示

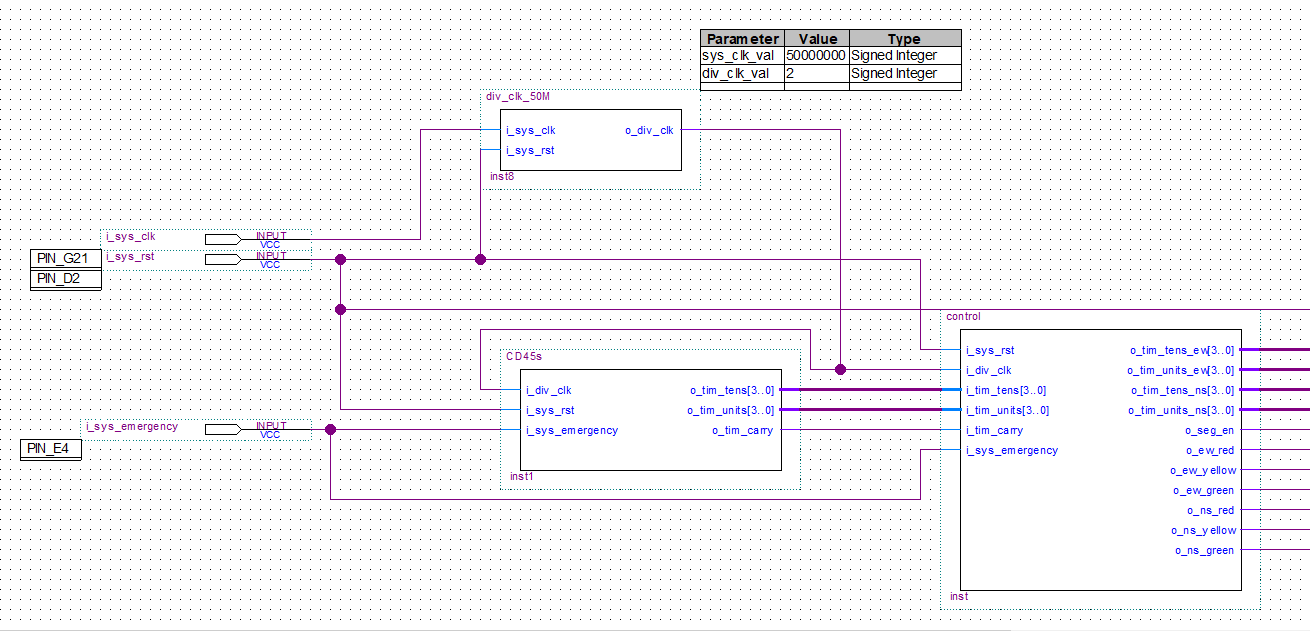


图 6分频器、倒计时、总控制器

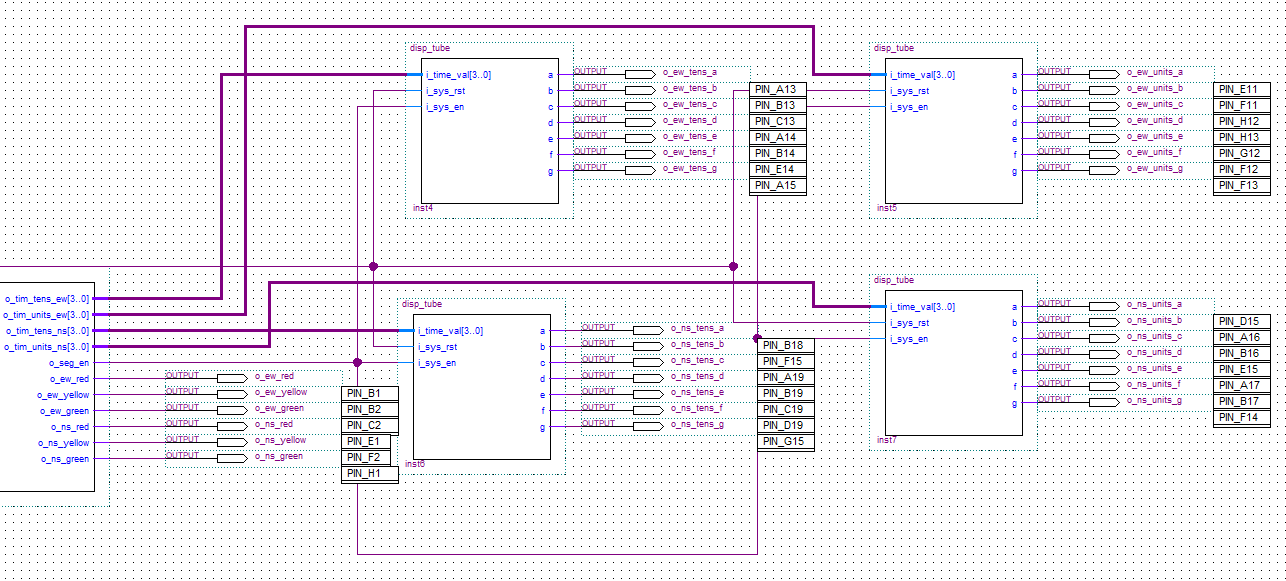


图 7数码管显示

## 引脚分配

根据开发手册配置相关引脚如图 8所示。

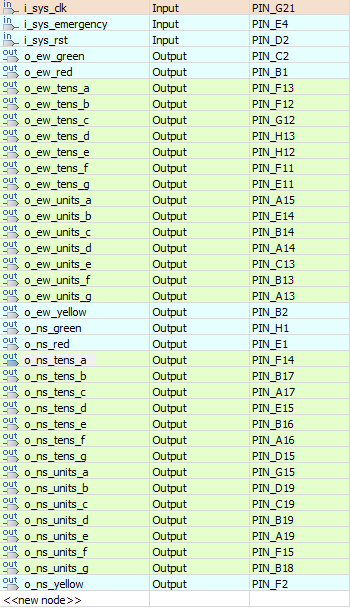


图 8配置引脚

# 波形仿真

由于仿真的原因，需要把50M分频器去除。

在命令行输入代码如下

1. force i\_sys\_clk 0 0,1 10 -r 20
2. force i\_sys\_rst 1 1,0 10 -r 200000000
3. force i\_sys\_emergency 1 1,0 500 -r 20000
4. run 30ns

## 引脚说明

i\_sys\_rst：重置信号

i\_sys\_clk：时钟信号

i\_sys\_emergency：紧急信号

o\_xx\_yy：红绿灯，其中xx=ew,ns表示东西或南北方向，yy=red,yellow,green表示红黄绿灯

o\_xx\_yy\_zz：数码倒计时，其中xx=ew,ns表示东西或南北方向，yy=tens,units表示十位和个位，zz=a,b,c,d,e,f,g表示数码管输出的7个段。

## 无紧急状况

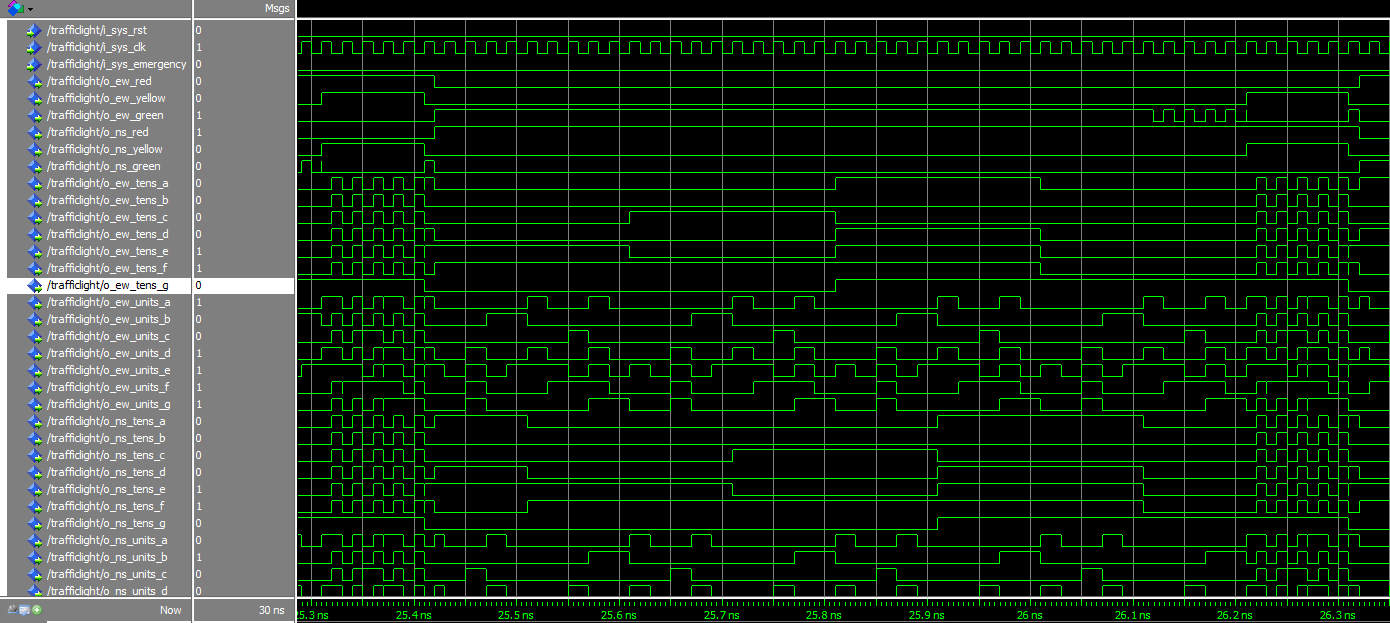


图 9无紧急状况波形仿真

如图 9可以看出从南北向变为红灯后，东西向由红灯变为绿灯，经过35秒，绿灯开始闪烁，到达40秒后绿灯熄灭黄灯亮起，这期间南北向红灯一直亮且在40秒后黄灯也亮起。再过5秒后，东西向由黄灯变为红灯，南北向由红灯和黄灯变为绿灯。这一期间东西向的数码由39减小到00，又从04码减小到00；南北向的数码从44减小到00。在最后5秒4个数码管都开始闪烁。

## 出现紧急状况

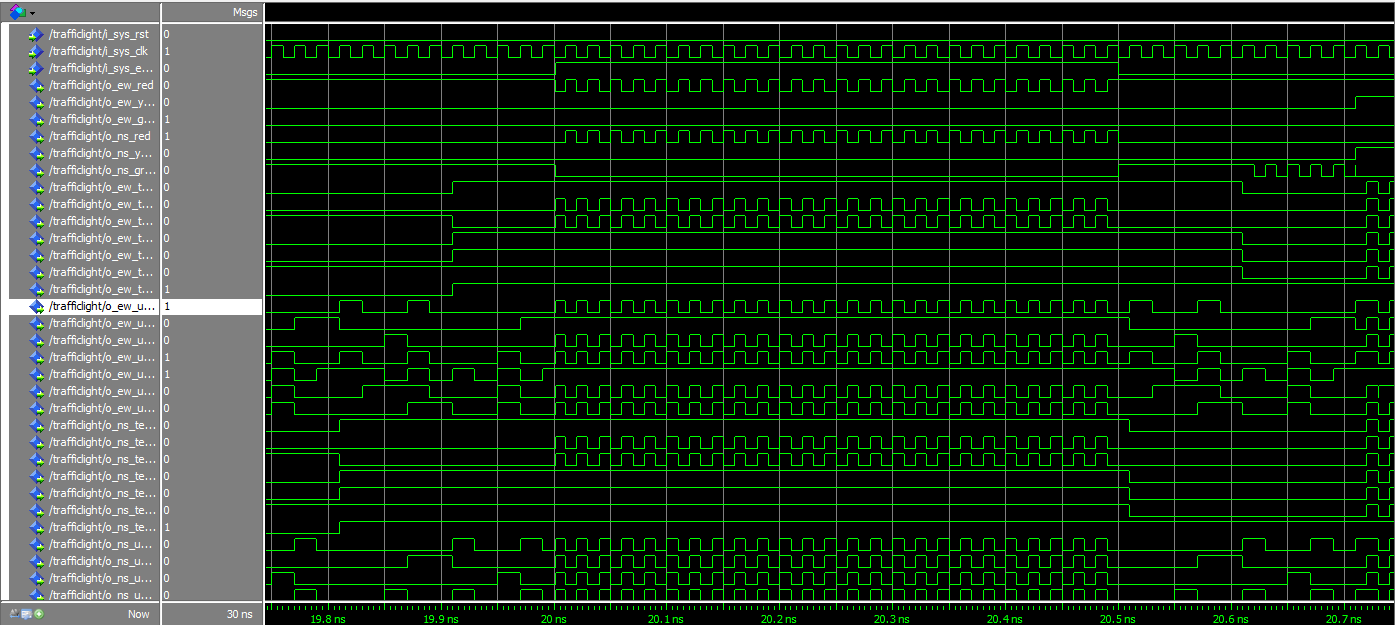


图 10紧急状况波形仿真

如图 10可以看出，当i\_sys\_emergency从0变为1时，东西和南北方向都变为红灯且跟随时钟信号开始闪烁，数码停止改变并闪烁。当i\_sys\_emergency变为0后，恢复紧急前的状况。

# 分析与总结

这次实验完成了设计任务，且在实验板上顺利运行，没有任何bug。

通过这次设计，我对VHDL编程更加得熟练，掌握了基本的语法，学会通过波形仿真来修改bug。

# 附录

**附录一**

1. LIBRARY IEEE;
2. USE IEEE.std\_logic\_1164.all;
3. USE IEEE.std\_logic\_arith.all;
4. USE IEEE.std\_logic\_unsigned.all;
6. --分频器
7. entity div\_clk\_50M **is**
8. generic(
9. sys\_clk\_val:INTEGER:=50000000;
10. div\_clk\_val:INTEGER:=1
11. );
12. port(
13. i\_sys\_clk:**in** std\_logic;
14. i\_sys\_rst:**in** std\_logic;
15. --i\_sys\_emergency:**in** std\_logic;
16. o\_div\_clk:out std\_logic
17. );
18. end entity div\_clk\_50M;
20. architecture behavior of div\_clk\_50M **is**
21. signal r\_div\_count:std\_logic\_vector(31 downto 0);
22. signal r\_div\_clk:std\_logic;
23. begin
24. process(i\_sys\_rst,i\_sys\_clk)
25. begin
26. **if**(i\_sys\_rst='1')then--复位键
27. r\_div\_count<=x"00000000";
28. r\_div\_clk<='0';
29. elsif(i\_sys\_clk'event AND i\_sys\_clk='1')then--上升沿计数
30. **if**(r\_div\_count=sys\_clk\_val/div\_clk\_val/2-1)then--到达24999999清零，输出反相
31. r\_div\_count<=x"00000000";
32. r\_div\_clk<=NOT r\_div\_clk;
33. **else**
34. r\_div\_count<=r\_div\_count+1;
35. end **if**;
36. end **if**;
37. end process;
38. o\_div\_clk<=r\_div\_clk;
39. end architecture behavior;

**附录二**

1. LIBRARY IEEE;
2. USE IEEE.std\_logic\_1164.all;
3. USE IEEE.std\_logic\_arith.all;
4. USE IEEE.std\_logic\_unsigned.all;
6. --倒计时45秒
7. entity CD45s **is**
8. port(
9. i\_div\_clk:**in** std\_logic;                             --1hz时钟输入
10. i\_sys\_rst:**in** std\_logic;                             --复位
11. i\_sys\_emergency:**in** std\_logic;                       --紧急状态
12. o\_tim\_tens:out std\_logic\_vector(3 downto 0);    --输出倒计时十位BCD
13. o\_tim\_units:out std\_logic\_vector(3 downto 0);--输出倒计时个位BCD
14. o\_tim\_carry:out std\_logic                         --输出进位
15. );
16. end entity CD45s;
18. architecture behavior of CD45s **is**
19. signal cout\_tens,cout\_units:std\_logic\_vector(3 downto 0);
20. signal r\_tim\_carry:std\_logic;
21. begin
22. process(i\_sys\_rst,i\_div\_clk,i\_sys\_emergency)
23. begin
24. **if** i\_sys\_rst='1'then
25. cout\_tens<="0000";
26. cout\_units<="0000";
27. elsif(i\_div\_clk'event AND i\_div\_clk='1')then
28. **if** i\_sys\_emergency='0'then
30. **if**(cout\_tens=0 AND cout\_units=1)then
31. r\_tim\_carry<='1';
32. **else**
33. r\_tim\_carry<='0';
34. end **if**;
36. **if**(cout\_tens=0 AND cout\_units=0)then
37. cout\_tens<="0100";
38. cout\_units<="0100";
39. --r\_tim\_carry<='1';
40. elsif(cout\_units=0)then
41. cout\_tens<=cout\_tens-1;
42. cout\_units<="1001";
43. --r\_tim\_carry<='0';
44. **else**
45. cout\_units<=cout\_units-1;
46. --r\_tim\_carry<='0';
47. end **if**;
48. end **if**;
49. end **if**;
50. end process;
51. o\_tim\_tens<=cout\_tens;
52. o\_tim\_units<=cout\_units;
53. o\_tim\_carry<=r\_tim\_carry;
54. end architecture behavior;

**附录三**

1. LIBRARY IEEE;
2. USE IEEE.std\_logic\_1164.all;
3. USE IEEE.std\_logic\_arith.all;
4. USE IEEE.std\_logic\_unsigned.all;
6. --七段数码管显示
7. entity disp\_tube **is**
8. port(
9. i\_time\_val:**in** std\_logic\_vector(3 downto 0);
10. i\_sys\_rst:**in** std\_logic;
11. i\_sys\_en:**in** std\_logic;
12. a,b,c,d,e,f,g:out std\_logic
13. --o\_tube\_disp\_val:out std\_logic\_vector(6 downto 0)
14. );
15. end entity disp\_tube;
17. architecture behavior of disp\_tube **is**
18. signal r\_tube\_disp\_val:std\_logic\_vector(6 downto 0);
19. begin
20. process(i\_sys\_rst,i\_time\_val)
21. begin
22. **if**(i\_sys\_en='1')then
23. **if**(i\_sys\_rst='1')then
24. r\_tube\_disp\_val<="1111111";
25. **else**
26. case i\_time\_val **is**
27. when "0000"=>r\_tube\_disp\_val<="1000000";
28. when "0001"=>r\_tube\_disp\_val<="1111001";
29. when "0010"=>r\_tube\_disp\_val<="0100100";
30. when "0011"=>r\_tube\_disp\_val<="0110000";
31. when "0100"=>r\_tube\_disp\_val<="0011001";
32. when "0101"=>r\_tube\_disp\_val<="0010010";
33. when "0110"=>r\_tube\_disp\_val<="0000010";
34. when "0111"=>r\_tube\_disp\_val<="1111000";
35. when "1000"=>r\_tube\_disp\_val<="0000000";
36. when "1001"=>r\_tube\_disp\_val<="0010000";
37. when others=>r\_tube\_disp\_val<="1111111";
38. end case;
39. end **if**;
40. **else**
41. r\_tube\_disp\_val<="1111111";
42. end **if**;
43. end process;
44. a<=r\_tube\_disp\_val(6);
45. b<=r\_tube\_disp\_val(5);
46. c<=r\_tube\_disp\_val(4);
47. d<=r\_tube\_disp\_val(3);
48. e<=r\_tube\_disp\_val(2);
49. f<=r\_tube\_disp\_val(1);
50. g<=r\_tube\_disp\_val(0);
51. --o\_tube\_disp\_val<=r\_tube\_disp\_val;
52. end architecture behavior;

**附录四**

1. LIBRARY IEEE;
2. USE IEEE.std\_logic\_1164.all;
3. USE IEEE.std\_logic\_arith.all;
4. USE IEEE.std\_logic\_unsigned.all;
6. entity control **is**
7. port(
8. i\_sys\_rst:**in** std\_logic;
9. i\_div\_clk:**in** std\_logic;                                 --1hz时钟输入
10. i\_tim\_tens:**in** std\_logic\_vector(3 downto 0);     --输入倒计时十位BCD
11. i\_tim\_units:**in** std\_logic\_vector(3 downto 0);        --输入倒计时个位BCD
12. i\_tim\_carry:**in** std\_logic;                             --输入进位
13. i\_sys\_emergency:**in** std\_logic;                           --紧急状态
14. o\_tim\_tens\_ew:out std\_logic\_vector(3 downto 0); --输出东西倒计时十位BCD
15. o\_tim\_units\_ew:out std\_logic\_vector(3 downto 0);--输出东西倒计时个位BCD
16. o\_tim\_tens\_ns:out std\_logic\_vector(3 downto 0); --输出南北倒计时十位BCD
17. o\_tim\_units\_ns:out std\_logic\_vector(3 downto 0);--输出南北倒计时个位BCD
18. o\_seg\_en:out std\_logic;
19. o\_ew\_red,o\_ew\_yellow,o\_ew\_green,o\_ns\_red,o\_ns\_yellow,o\_ns\_green:out std\_logic--输出东西南北红绿灯
21. );
22. end entity control;
24. architecture behavior of control **is**
25. signal r\_tim\_tens\_ew:std\_logic\_vector(3 downto 0);
26. signal r\_tim\_units\_ew:std\_logic\_vector(3 downto 0);
27. signal r\_tim\_tens\_ns:std\_logic\_vector(3 downto 0);
28. signal r\_tim\_units\_ns:std\_logic\_vector(3 downto 0);
29. signal r\_dir\_flag:std\_logic;                                --方向标志位
30. signal r\_ew\_red,r\_ew\_yellow,r\_ew\_green,r\_ns\_red,r\_ns\_yellow,r\_ns\_green:std\_logic;--输出东西南北红绿灯
31. signal r\_seg\_en:std\_logic;
32. begin
33. process(i\_sys\_rst,i\_div\_clk,i\_tim\_tens,i\_tim\_units,i\_tim\_carry,i\_sys\_emergency)
34. begin
35. **if**(i\_sys\_rst='1')then
36. r\_dir\_flag<='0';
37. **else**
38. **if**(i\_tim\_carry'event AND i\_tim\_carry='0')then--交换通行
39. r\_dir\_flag<=NOT r\_dir\_flag;
40. end **if**;
41. **if**(i\_sys\_emergency='1')then
42. r\_seg\_en<=i\_div\_clk;
43. --r\_ew\_red<='1';
44. r\_ew\_red<=i\_div\_clk;
45. r\_ew\_yellow<='0';
46. r\_ew\_green<='0';
47. --r\_ns\_red<='1';
48. r\_ns\_red<=i\_div\_clk;
49. r\_ns\_yellow<='0';
50. r\_ns\_green<='0';
51. **else**
52. r\_seg\_en<='1';
53. **if**(r\_dir\_flag='1')then--东西通行
54. **if**(i\_tim\_tens=0 AND i\_tim\_units<5)then
55. r\_ew\_red<='0';
56. r\_ew\_yellow<='1';
57. r\_ew\_green<='0';
58. r\_tim\_tens\_ew<=i\_tim\_tens;
59. r\_tim\_units\_ew<=i\_tim\_units;
60. r\_seg\_en<=i\_div\_clk;
62. r\_ns\_red<='1';
63. r\_ns\_yellow<='1';
64. r\_ns\_green<='0';
65. **else**
66. r\_ew\_red<='0';
67. r\_ew\_yellow<='0';
68. **if**(i\_tim\_tens=0 AND i\_tim\_units<=9)then--最后5秒绿灯闪烁
69. r\_ew\_green<=i\_div\_clk;
70. **else**
71. r\_ew\_green<='1';
72. end **if**;
74. **if**(i\_tim\_units<5)then
75. r\_tim\_tens\_ew<=i\_tim\_tens-1;
76. r\_tim\_units\_ew<=i\_tim\_units+5;
77. **else**
78. r\_tim\_tens\_ew<=i\_tim\_tens;
79. r\_tim\_units\_ew<=i\_tim\_units-5;
80. end **if**;
81. r\_ns\_red<='1';
82. r\_ns\_yellow<='0';
83. r\_ns\_green<='0';
84. end **if**;
85. r\_tim\_tens\_ns<=i\_tim\_tens;
86. r\_tim\_units\_ns<=i\_tim\_units;
88. **else**--南北通行
89. **if**(i\_tim\_tens=0 AND i\_tim\_units<5)then
90. r\_ns\_red<='0';
91. r\_ns\_yellow<='1';
92. r\_ns\_green<='0';
93. r\_tim\_tens\_ns<=i\_tim\_tens;
94. r\_tim\_units\_ns<=i\_tim\_units;
95. r\_seg\_en<=i\_div\_clk;
97. r\_ew\_red<='1';
98. r\_ew\_yellow<='1';
99. r\_ew\_green<='0';
100. **else**
101. r\_ns\_red<='0';
102. r\_ns\_yellow<='0';
103. **if**(i\_tim\_tens=0 AND i\_tim\_units<=9)then--最后5秒绿灯闪烁
104. r\_ns\_green<=i\_div\_clk;
105. **else**
106. r\_ns\_green<='1';
107. end **if**;
108. **if**(i\_tim\_units<5)then
109. r\_tim\_tens\_ns<=i\_tim\_tens-1;
110. r\_tim\_units\_ns<=i\_tim\_units+5;
111. **else**
112. r\_tim\_tens\_ns<=i\_tim\_tens;
113. r\_tim\_units\_ns<=i\_tim\_units-5;
114. end **if**;
115. r\_ew\_red<='1';
116. r\_ew\_yellow<='0';
117. r\_ew\_green<='0';
118. end **if**;
119. r\_tim\_tens\_ew<=i\_tim\_tens;
120. r\_tim\_units\_ew<=i\_tim\_units;
121. end **if**;
122. end **if**;
123. end **if**;
124. end process;
125. o\_tim\_tens\_ew<=r\_tim\_tens\_ew;
126. o\_tim\_units\_ew<=r\_tim\_units\_ew;
127. o\_tim\_tens\_ns<=r\_tim\_tens\_ns;
128. o\_tim\_units\_ns<=r\_tim\_units\_ns;
129. o\_seg\_en<=r\_seg\_en;
130. o\_ew\_red<=r\_ew\_red;
131. o\_ew\_yellow<=r\_ew\_yellow;
132. o\_ew\_green<=r\_ew\_green;
133. o\_ns\_red<=r\_ns\_red;
134. o\_ns\_yellow<=r\_ns\_yellow;
135. o\_ns\_green<=r\_ns\_green;
136. end architecture behavior;