**信息科学与工程学院**

**2019－2020学年第二学期**

实 验 报 告

课程名称： 电子设计自动化

实验名称： 实验四 有限状态机设计

专 业 班 级 通信工程 二班

学 生 学 号 201800121050

学 生 姓 名 孟麟芝

实 验 时 间 2020年5月

实验报告

【实验目的】

1.掌握使用VHDL语言进行硬件设计的基本方法

2.掌握基本的Quartus Ⅱ使用方法

3.学会testbench的编写，掌握波形仿真的基本方法

4.掌握有限状态机设计方法

【实验要求】

1.设计一个交通红绿灯控制器模块，实现主干道和支路之间红绿黄灯的信号转换。

2.设计一个10层楼的电梯控制器模块，要求：(1) 时间先后优先级；(2)位置先后优先级。

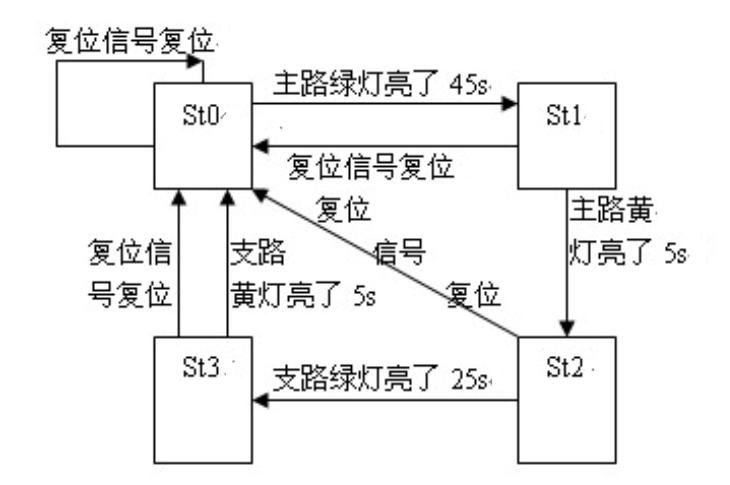
3.设计一个10位计算器(+,-,\*,/)，要有BCD码转换，共阴极LED笔画显示部分实现。

【实验具体内容】

### 【第一个实验】

1.实验原理

状态转换图



2.设计模块代码（Design Block）

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY TRAFFIC IS

PORT(CLK,RST : IN STD\_LOGIC;

YM,GM,RM,YF,GF,RF : OUT STD\_LOGIC

);

END TRAFFIC;

ARCHITECTURE BHV OF TRAFFIC IS

TYPE STATES IS(ST0,ST1,ST2,ST3);

SIGNAL PST: STATES;

BEGIN

PROCESS(CLK,RST)

VARIABLE CNT: INTEGER RANGE 0 TO 45;

BEGIN

IF RST='1' THEN

PST<= ST0;CNT:=0;

ELSIF RISING\_EDGE(CLK) THEN

CASE PST IS

WHEN ST0=>

YM<='0';GM<='1';RM<='0';

YF<='0';GF<='0';RF<='1';

CNT:=CNT+1;

IF(CNT=45) THEN

PST<= ST1;

CNT:=0;

END IF;

WHEN ST1=>

YM<='1';GM<='0';RM<='0';

YF<='0';GF<='0';RF<='1';

CNT:=CNT+1;

IF(CNT=5) THEN

PST<= ST2;

CNT:=0;

END IF;

WHEN ST2=>

YM<='0';GM<='0';RM<='1';

YF<='0';GF<='1';RF<='0';

CNT:=CNT+1;

IF(CNT=25) THEN

PST<= ST3;

CNT:=0;

END IF;

WHEN ST3=>

YM<='0';GM<='0';RM<='1';

YF<='1';GF<='0';RF<='0';

CNT:=CNT+1;

IF(CNT=5) THEN

PST<= ST0;

CNT:=0;

END IF;

WHEN OTHERS=>

PST<=ST0;CNT:=0;

END CASE;

END IF;

END PROCESS;

END BHV;

3.激励模块代码（Test Bench）

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

ENTITY TRAFFIC\_vhd\_tst IS

END TRAFFIC\_vhd\_tst;

ARCHITECTURE TRAFFIC\_arch OF TRAFFIC\_vhd\_tst IS

SIGNAL CLK1 : STD\_LOGIC;

SIGNAL GF1 : STD\_LOGIC;

SIGNAL GM1 : STD\_LOGIC;

SIGNAL RF1 : STD\_LOGIC;

SIGNAL RM1 : STD\_LOGIC;

SIGNAL RST1 : STD\_LOGIC;

SIGNAL YF1 : STD\_LOGIC;

SIGNAL YM1 : STD\_LOGIC;

COMPONENT TRAFFIC

PORT (

CLK : IN STD\_LOGIC;

GF : OUT STD\_LOGIC;

GM : OUT STD\_LOGIC;

RF : OUT STD\_LOGIC;

RM : OUT STD\_LOGIC;

RST : IN STD\_LOGIC;

YF : OUT STD\_LOGIC;

YM : OUT STD\_LOGIC

);

END COMPONENT;

CONSTANT CLK\_P: TIME:=500 MS;

BEGIN

i1 : TRAFFIC

PORT MAP (

CLK => CLK1,

GF => GF1,

GM => GM1,

RF => RF1,

RM => RM1,

RST => RST1,

YF => YF1,

YM => YM1

);

PROCESS BEGIN

CLK1<='0'; WAIT FOR CLK\_P;

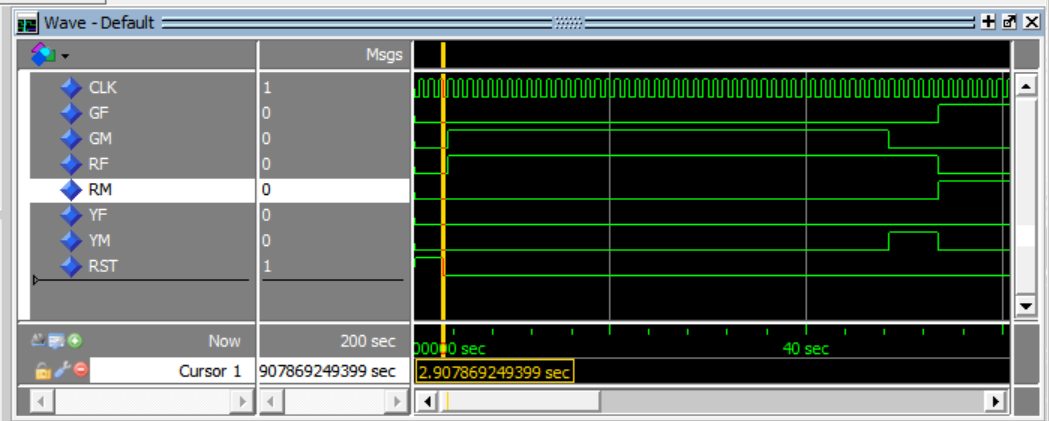
CLK1<='1'; WAIT FOR CLK\_P;

END PROCESS;

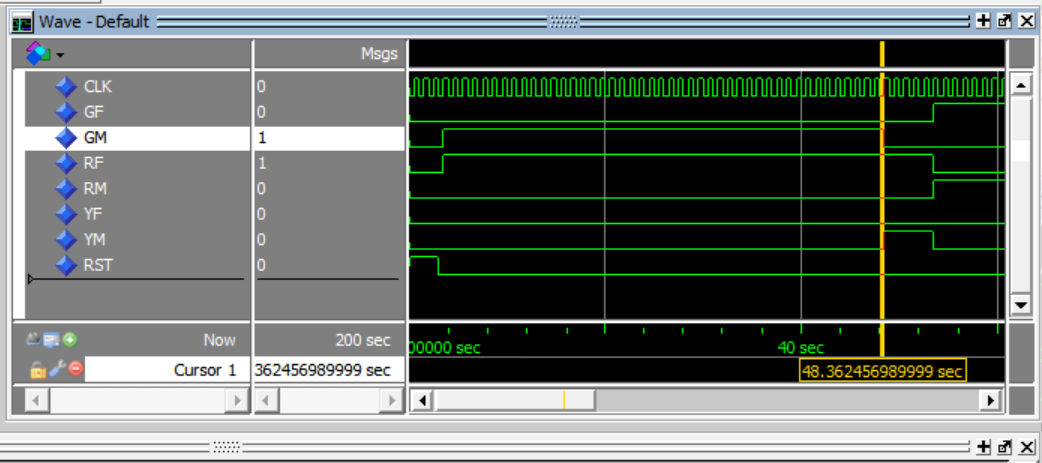
RST1<='1', '0' AFTER 3000 MS;

END TRAFFIC\_arch;

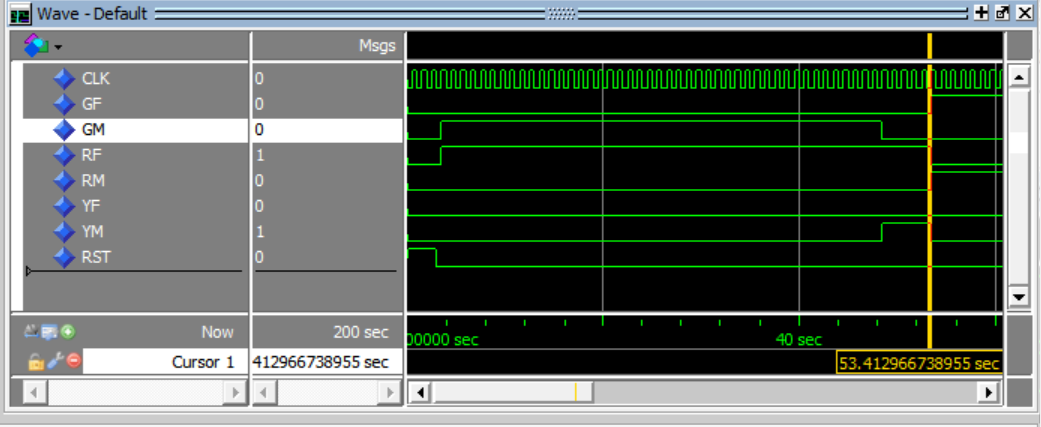
4.仿真波形图



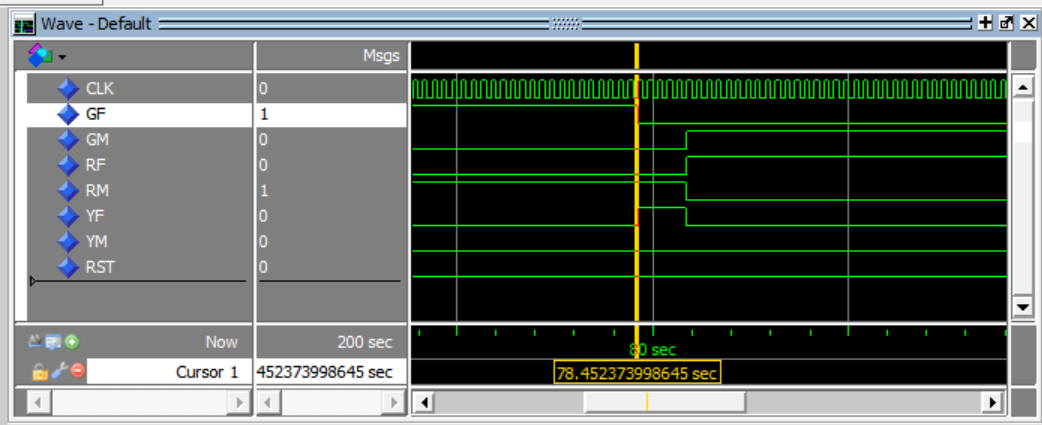
最初三秒令RST为‘1’使之回到零状态



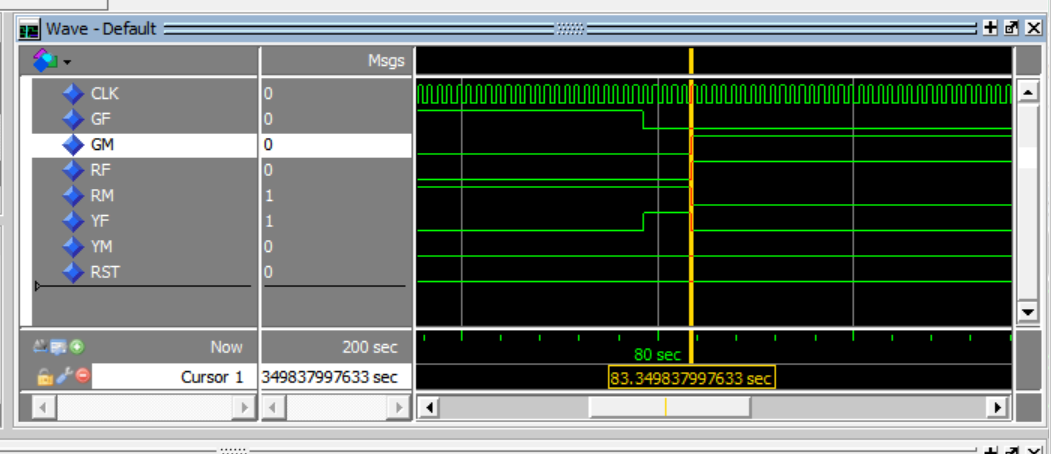
三秒后开始ST0，主干道绿灯亮，支路红灯亮，持续45秒



而后进入ST1，主干道黄灯亮，支路红灯亮，持续5秒



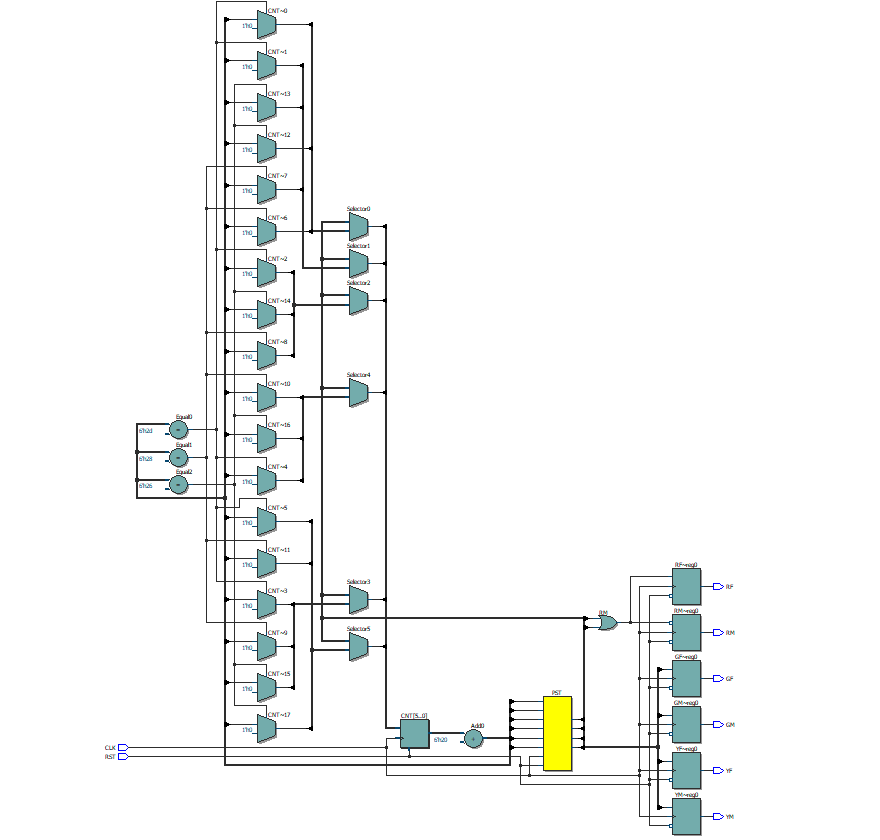
之后进入ST2，支路绿灯亮，主干道红灯亮，持续25秒



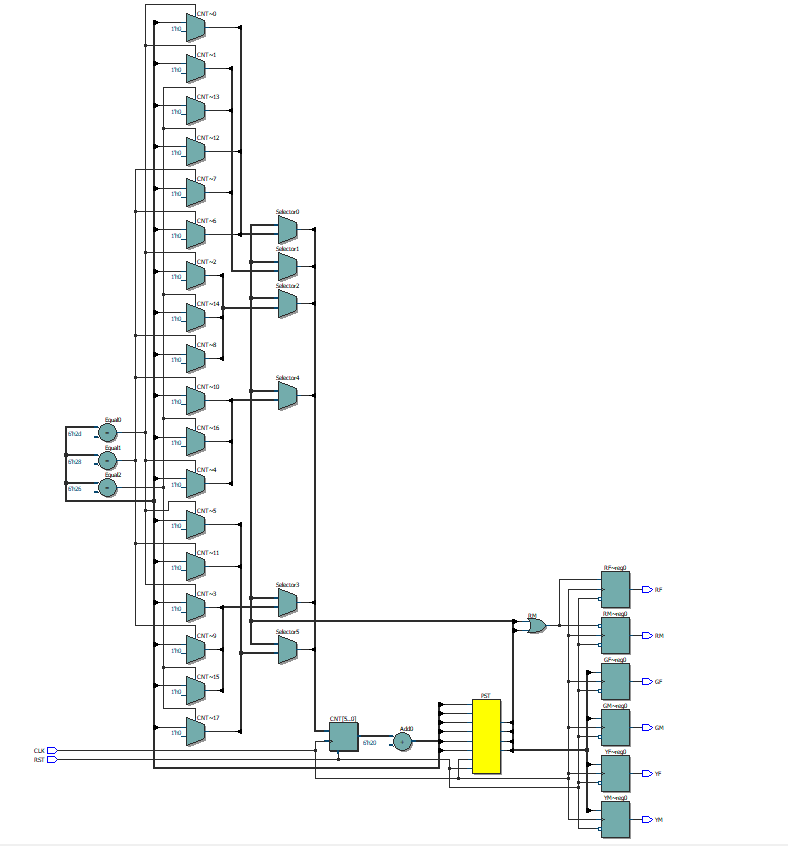
之后进入ST3，支路黄灯亮，主干道红灯亮，持续5秒，之后即回到ST0

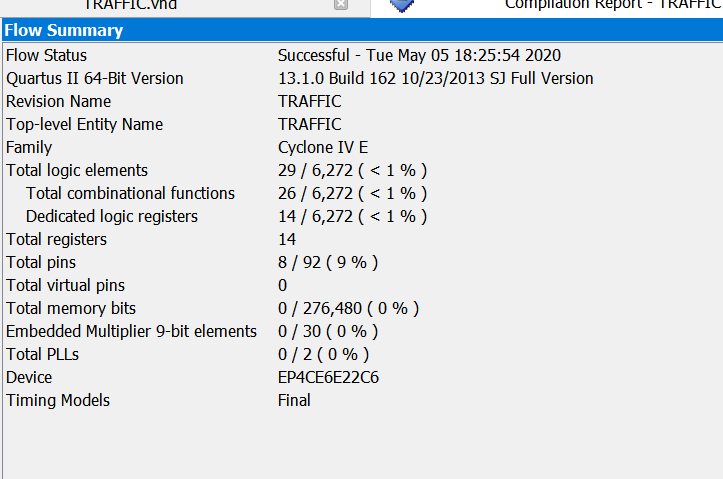
5.门级电路图

（1）使用GRAY编码

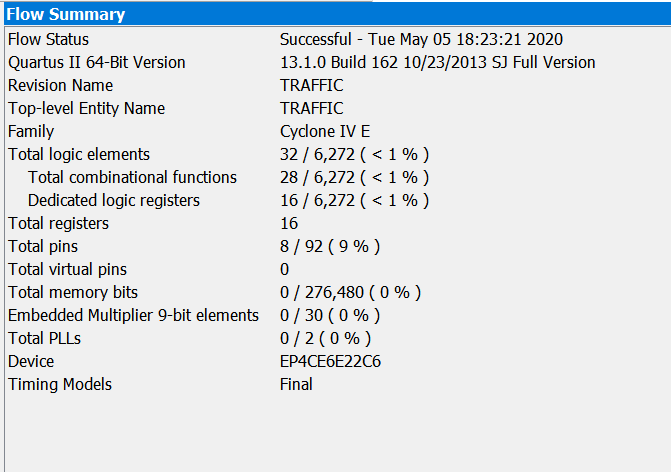


（2）使用ONE-HOT





如上图为使用格雷码资源占用情况



如上图为使用onehot码的资源占用，可见格雷码对资源的占用相对稍小，对于两种编码方式，EP4CE6E22C6这一款芯片都可满足需求

6.结果分析与思考

该设计实现了红绿灯的功能，缺点在于红绿灯时间无法灵活控制，另外其准确性依赖于时钟信号的准确性，若加入分频器和高频稳定的震荡，则时间控制会更准确

### 【第二个实验】

1.实验原理

该电梯选择的是方向优先控制方案。电梯应当满足下面一些基本的要求：

1)．检测上方是否存在请求，存在电梯则运行，不存在则保持停靠

2)．检测下方是否存在请求，存在电梯则运行，不存在则保持停靠

3). 检测当前层是否存在请求，存在则开门，不存在则保持停靠

4). 经过请求楼层时要有开门停靠时间，并清除该层的请求（由于缺乏键盘、显示管等，清除请求这一点暂不予考虑）

5). 电梯要能显示当前所在的楼层

2.设计模块代码（Design Block）

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

ENTITY ELEVATOR IS

PORT(CLK,RST : IN STD\_LOGIC;

UPP : IN STD\_LOGIC\_VECTOR(10 DOWNTO 1);

DOWNN : IN STD\_LOGIC\_VECTOR(10 DOWNTO 1);

INSIDE : IN STD\_LOGIC\_VECTOR(10 DOWNTO 1);

POSITION : OUT INTEGER RANGE 1 TO 10;

DOOR\_LIGHT: OUT STD\_LOGIC

);

END ELEVATOR;

ARCHITECTURE BHV OF ELEVATOR IS

TYPE STATES IS(UPING,DOWNING,STOPPING,OPENDOOR);

SIGNAL UPREC1 : STD\_LOGIC\_VECTOR(10 DOWNTO 1):="0000000000";

SIGNAL DOWNREC1 : STD\_LOGIC\_VECTOR(10 DOWNTO 1):="0000000000";

SIGNAL PST : STATES;

SIGNAL WAITT : STATES;

BEGIN

PROCESS (UPP,INSIDE)

BEGIN

UPREC1 <= UPP OR INSIDE;

END PROCESS;

PROCESS (DOWNN,INSIDE)

BEGIN

DOWNREC1 <= DOWNN OR INSIDE;

END PROCESS;

PROCESS(CLK,RST)

VARIABLE UPREC2 : STD\_LOGIC\_VECTOR(10 DOWNTO 1):="0000000000";

VARIABLE DOWNREC2 : STD\_LOGIC\_VECTOR(10 DOWNTO 1):="0000000000";

VARIABLE CMP : STD\_LOGIC\_VECTOR(10 DOWNTO 1):="0000000000";

VARIABLE LEVEL : INTEGER RANGE 1 TO 10:=1;

VARIABLE CNT,JUDGE1 : INTEGER RANGE 0 TO 10:=0;

VARIABLE JUDGE2 : INTEGER RANGE 0 TO 11:=0;

BEGIN

IF RST='1' THEN

PST<= STOPPING;POSITION<=1;DOOR\_LIGHT<='0'; --RST高电平为复位操作

ELSIF RISING\_EDGE(CLK) THEN

POSITION<=LEVEL;

CASE PST IS

WHEN STOPPING=>

WAITT<=STOPPING;

CMP:=UPREC1 OR DOWNREC1 ;

JUDGE1:=0;

L0:FOR I IN 10 DOWNTO 1 LOOP

IF(CMP(I)='1') THEN

JUDGE1:=I;

EXIT;

ELSE

NEXT L0;

END IF;

END LOOP;

IF(JUDGE1=0) THEN

PST<= STOPPING;

ELSIF(JUDGE1=LEVEL) THEN

PST<= OPENDOOR;

ELSIF(JUDGE1<LEVEL) THEN

PST<= DOWNING;

ELSE

PST<= UPING;

END IF;--判断应当向上/向下

WHEN UPING=>

WAITT<=UPING;

UPREC2:= UPREC1;

L11:FOR I IN 10 DOWNTO 1 LOOP

IF(UPREC1(I)='1' OR DOWNREC1(I)='1') THEN

JUDGE1:=I;

EXIT;

ELSE

NEXT L11;

END IF;

END LOOP;--L11是为了判断有请求的最高楼层

L12:FOR I IN 10 DOWNTO 1 LOOP

IF(UPREC1(I)='1' AND I=10) THEN

JUDGE2:=I;

EXIT;

ELSIF(UPREC1(I)='1') THEN

JUDGE2:=I+1;

UPREC2(10 DOWNTO (JUDGE2+1)):=DOWNREC1(10 DOWNTO (JUDGE2+1));

EXIT;

ELSE

NEXT L12;

END IF;

END LOOP;--L12是为了判断有向上请求的最高楼层的上一层

LEVEL:=LEVEL+1;

IF(UPREC2(LEVEL)='1') THEN--在上升过程中也可以响应新的请求

PST<= OPENDOOR;

ELSE

IF(LEVEL>=JUDGE1) THEN

PST<= STOPPING;

LEVEL:=LEVEL-1;--这里是为了消除上面加一的影响

ELSE

PST<= UPING;

END IF;END IF;

WHEN DOWNING=>

WAITT<=DOWNING;

DOWNREC2:=DOWNREC1 OR UPREC1;

L21:FOR I IN 1 TO 10 LOOP

IF(DOWNREC2(I)='1') THEN

JUDGE1:=I;

EXIT;

ELSE

NEXT L21;

END IF;

END LOOP;--L21的作用与L11恰好相反

L22:FOR I IN 1 TO 10 LOOP

IF(DOWNREC1(I)='1' AND I=1) THEN

JUDGE2:=I;

EXIT;

ELSIF(DOWNREC1(I)='1') THEN

JUDGE2:=I;

DOWNREC2((JUDGE2-1) DOWNTO 1):=UPREC1((JUDGE2-1) DOWNTO 1);

EXIT;

ELSE

NEXT L22;

END IF;

END LOOP;--L22是为了判断有向下请求的最低楼层的下一层

LEVEL:=LEVEL-1;

IF(DOWNREC2(LEVEL)='1') THEN--与向上时相似

PST<= OPENDOOR;

ELSE

IF(LEVEL<=JUDGE1) THEN

PST<= STOPPING;

LEVEL:=LEVEL+1;--这里是为了消除上面减一的影响

ELSE

PST<=DOWNING;

END IF;END IF;

WHEN OPENDOOR=>

DOOR\_LIGHT<='1';

CNT:=CNT+1;

IF(CNT=5) THEN

CASE WAITT IS

WHEN STOPPING=> PST<= STOPPING;

WHEN UPING => PST<= UPING;

WHEN DOWNING => PST<= DOWNING;

WHEN OTHERS => PST<= STOPPING;

END CASE;

CNT:=0;

DOOR\_LIGHT<='0';

END IF;--这一状态设置了一个下降停站过程中5个时钟的开门时间

END CASE;

END IF;

END PROCESS;

END BHV;

3.激励模块代码（Test Bench）

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY ELEVATOR\_vhd\_tst IS

END ELEVATOR\_vhd\_tst;

ARCHITECTURE ELEVATOR\_arch OF ELEVATOR\_vhd\_tst IS

SIGNAL CLK1 : STD\_LOGIC;

SIGNAL DOOR\_LIGHT1 : STD\_LOGIC;

SIGNAL DOWNN1 : STD\_LOGIC\_VECTOR(10 DOWNTO 1);

SIGNAL INSIDE1 : STD\_LOGIC\_VECTOR(10 DOWNTO 1);

SIGNAL POSITION1 : STD\_LOGIC\_VECTOR(3 DOWNTO 0);

SIGNAL RST1 : STD\_LOGIC;

SIGNAL UPP1 : STD\_LOGIC\_VECTOR(10 DOWNTO 1);

COMPONENT ELEVATOR

PORT (

CLK : IN STD\_LOGIC;

DOOR\_LIGHT : OUT STD\_LOGIC;

DOWNN : IN STD\_LOGIC\_VECTOR(10 DOWNTO 1);

INSIDE : IN STD\_LOGIC\_VECTOR(10 DOWNTO 1);

POSITION : OUT STD\_LOGIC\_VECTOR(3 DOWNTO 0);

RST : IN STD\_LOGIC;

UPP : IN STD\_LOGIC\_VECTOR(10 DOWNTO 1)

);

END COMPONENT;

CONSTANT CLK\_P: TIME:=500 MS;

BEGIN

i1 : ELEVATOR

PORT MAP (

CLK => CLK1,

DOOR\_LIGHT => DOOR\_LIGHT1,

DOWNN => DOWNN1,

INSIDE => INSIDE1,

POSITION => POSITION1,

RST => RST1,

UPP => UPP1

);

PROCESS BEGIN

CLK1<='0'; WAIT FOR CLK\_P;

CLK1<='1'; WAIT FOR CLK\_P;

END PROCESS;

RST1<='1','0' AFTER 200 MS;

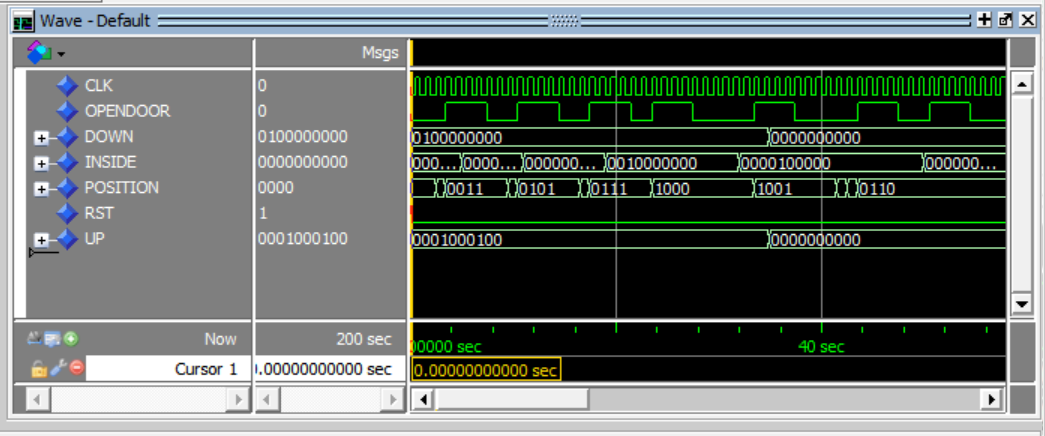
INSIDE1<="0000000000","0000010000" AFTER 5000 MS,"0000000000" AFTER 11000 MS,"0010000000" AFTER 19000 MS,"0000100000" AFTER 32000 MS,"0000000000" AFTER 50000 MS;

UPP1<="0001000100","0000000000" AFTER 35000 MS;

DOWNN1<="0100000000","0000000000" AFTER 35000 MS;

END ELEVATOR\_arch;

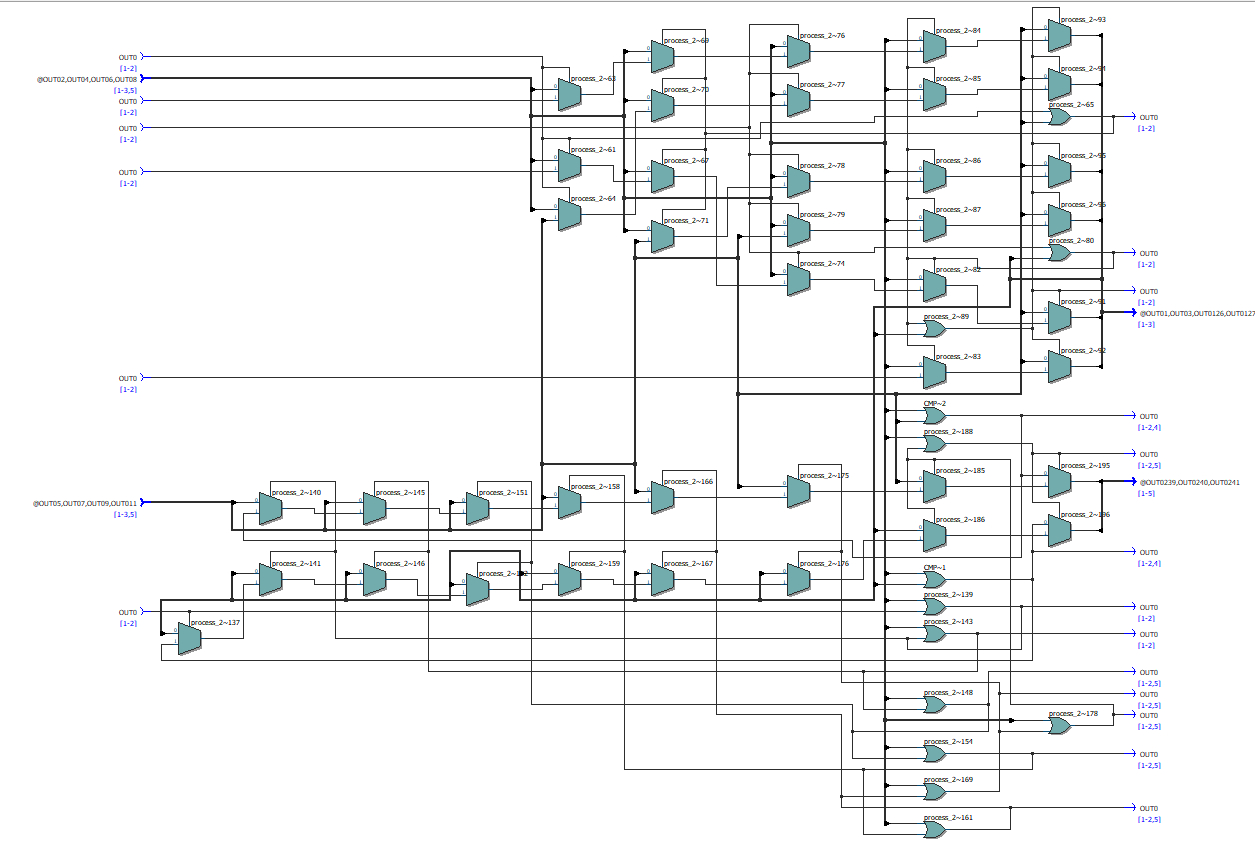
4.仿真波形图



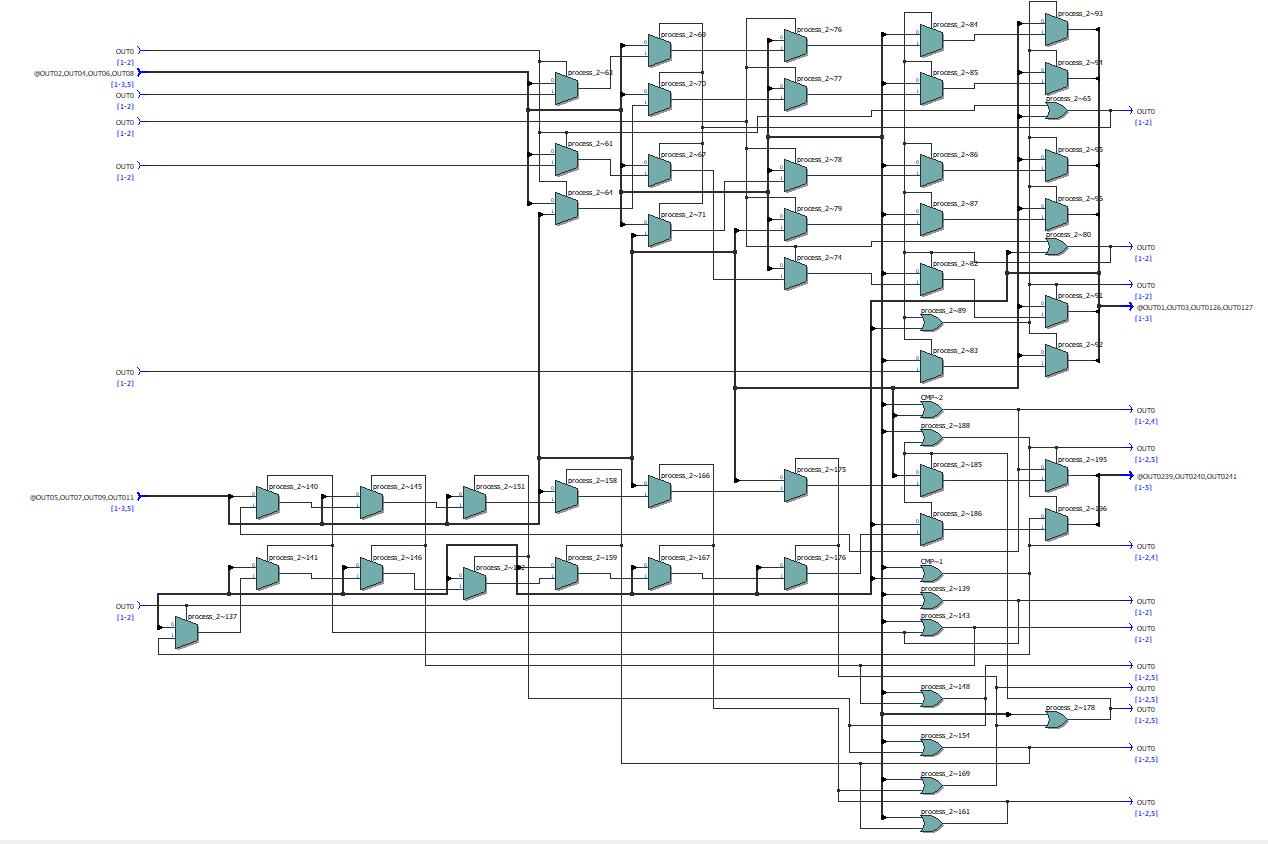
结合testbench，该仿真模拟的情况是3层和7层分别发出了想上楼的请求，第9层发出了想下楼的请求，电梯开始上行，到达3层，该人在电梯内按下5层，而后电梯将会在五层停靠，经停5层后继续上行，到达7层，该人在电梯内按下8层，故电梯在3，5，8层分别停靠了五个时钟周期，由于9层有人想要下楼，故继续上行，到达第9层以后，该人要去第6层，即开始下行，到达第六层以后完成了全部请求，保持停靠，之所以在停在6层后还会开一次门，是因为TESTBENCH里并未在停靠后将去6层的申请及时清除，而这也恰好验证了另一种情况，即停靠时恰好接到了该层的请求，电梯也应当开门

5.门级电路图

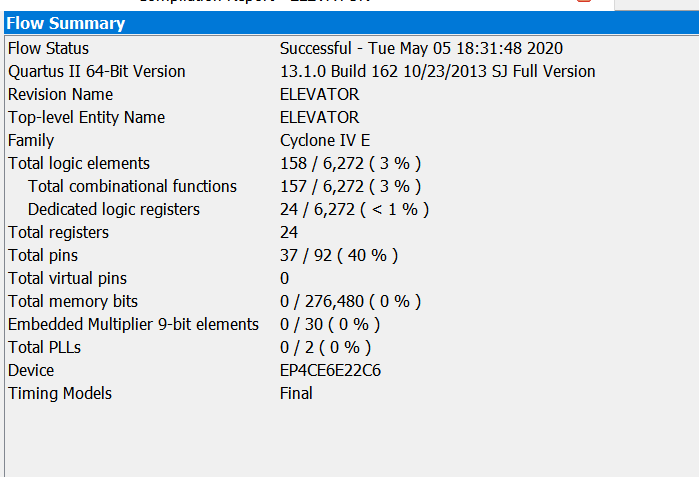
（1）使用one-hot码



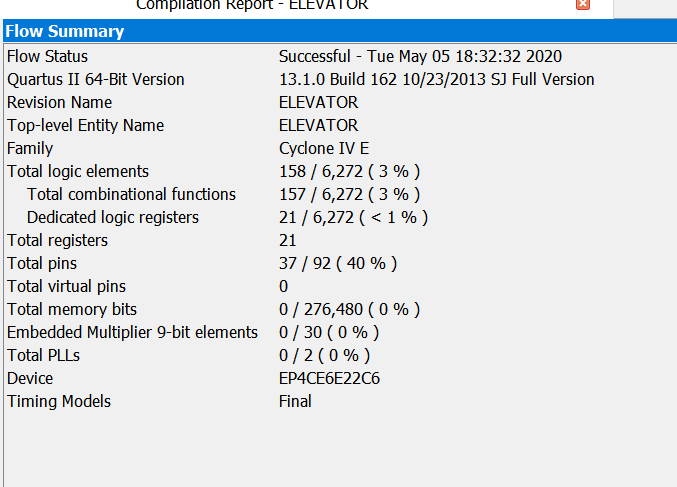
（2）使用GRAY码



如下图为使用onehot码资源占用情况



如下图为使用格雷码资源占用



可见格雷码的资源稍小，对于两种编码，EP4VE6E22C6这一款芯片都可以满足要求

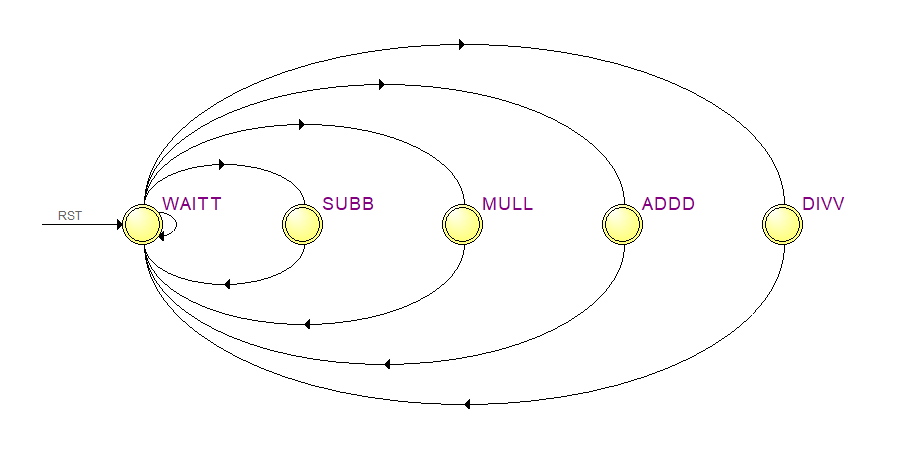
6.结果分析与思考

该电梯能实现方向优先的方案（与学校、商城等的大部分电梯相同），缺点在于电梯无法到达请求楼层后自行消除该层的申请，只能靠仿真时控制输入波形的变化表现“清除”这一过程，限于缺乏其他硬件（如键盘、显示器），现不考虑这一情况

### 【第三个实验】

1.实验原理

核心计算模块实现：



1. 在等待时，检验输入数据是否变化，若有变化则进行下一步，若无变化继续等待
2. 检验算符输入有无变为1的输入，有则跳转到相应运算过程：设运算符为(X)，输入数据为A,上一步完成后输出数据为Dn-1，则Dn= Dn-1 (X) A
3. 二进制结果送给输出端口，将结果转换后送给BCD码输出端口，回到等待状态

7448译码器真值表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| LI | RBIN | D C B A | BIN/RBO | Ya Yb Yc Yd Ye Yf Yg | 显示 |
| 1 | 1 | 0 0 0 0 | 1 | 1 1 1 1 1 1 0 | 0 |
| 1 | X | 0 0 0 1 | 1 | 0 1 1 0 0 0 0 | 1 |
| 1 | X | 0 0 1 0 | 1 | 1 1 0 1 1 0 1 | 2 |
| 1 | X | 0 0 1 1 | 1 | 1 1 1 1 0 0 1 | 3 |
| 1 | X | 0 1 0 0 | 1 | 0 1 1 0 0 1 1 | 4 |
| 1 | X | 0 1 0 1 | 1 | 1 0 1 1 0 1 1 | 5 |
| 1 | X | 0 1 1 0 | 1 | 0 0 1 1 1 1 1 | 6 |
| 1 | X | 0 1 1 1 | 1 | 1 1 1 0 0 0 0 | 7 |
| 1 | X | 1 0 0 0 | 1 | 1 1 1 1 1 1 1 | 8 |
| 1 | X | 1 0 0 1 | 1 | 0 0 0 1 1 0 1 | 9 |

2.设计模块代码（Design Block）

（1）计算输出模块

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL;

USE IEEE.STD\_LOGIC\_ARITH.ALL;

ENTITY CALCULATOR IS

PORT(RST,ADD,SUB,MUL,DIV,CLK : IN STD\_LOGIC;

DIN : IN INTEGER RANGE 0 TO 2000:=0;

DOUT : BUFFER INTEGER RANGE 0 TO 10000000:=0;

BCDOUT : BUFFER STD\_LOGIC\_VECTOR(11 DOWNTO 0)

);

END CALCULATOR;

ARCHITECTURE BHV OF CALCULATOR IS

TYPE STATES IS(ADDD,SUBB,MULL,DIVV,WAITT);

SIGNAL PST: STATES;

BEGIN

PROCESS(CLK,RST)

VARIABLE JUDGE : STD\_LOGIC\_VECTOR(3 DOWNTO 0):="0000";

VARIABLE JUDGE1: STD\_LOGIC\_VECTOR(3 DOWNTO 0):="0000";

VARIABLE JUDGE2: INTEGER RANGE 0 TO 2000:=0;

BEGIN

IF RST='1' THEN

PST<= WAITT;

ELSIF RISING\_EDGE(CLK) THEN

CASE PST IS

WHEN WAITT=>

JUDGE:=ADD&SUB&MUL&DIV;

IF (JUDGE=JUDGE1 AND JUDGE2=DIN) THEN

PST<=WAITT;

ELSE

CASE JUDGE IS

WHEN "1000" => PST<=ADDD;

WHEN "0100" => PST<=SUBB;

WHEN "0010" => PST<=MULL;

WHEN "0001" => PST<=DIVV;

WHEN OTHERS => PST<=WAITT;

END CASE;

END IF;

WHEN ADDD=>

DOUT<=DOUT+DIN;

PST <= WAITT;

JUDGE1:=ADD&SUB&MUL&DIV;

JUDGE2:=DIN;

WHEN SUBB=>

DOUT<=DOUT-DIN;

PST <=WAITT;

JUDGE1:=ADD&SUB&MUL&DIV;

JUDGE2:=DIN;

WHEN MULL=>

DOUT<=DOUT\*DIN;

PST <=WAITT;

JUDGE1:=ADD&SUB&MUL&DIV;

JUDGE2:=DIN;

WHEN DIVV=>

DOUT<=DOUT/DIN;

PST <=WAITT;

JUDGE1:=ADD&SUB&MUL&DIV;

JUDGE2:=DIN;

END CASE;

END IF;

END PROCESS;

PROCESS(CLK)

VARIABLE TEMP:INTEGER;

VARIABLE BIN\_TEMP:INTEGER;

VARIABLE BCD\_TEMP:STD\_LOGIC\_VECTOR(11 DOWNTO 0);

BEGIN

BCD\_TEMP:=X"000";

BIN\_TEMP:=CONV\_INTEGER(DOUT);

FOR K IN 0 TO 2 LOOP

TEMP:=BIN\_TEMP REM 10;

BCD\_TEMP(3+4\*K DOWNTO 4\*K):=CONV\_STD\_LOGIC\_VECTOR(TEMP,4);--将TEMP的值转化为4位二进制数

BIN\_TEMP:=(BIN\_TEMP-TEMP)/10;

IF BIN\_TEMP=0 THEN EXIT;

END IF;

END LOOP;

BCDOUT<=BCD\_TEMP;

END PROCESS;

END BHV;

（2）7448译码器（共阴显示模块）

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY C7448 IS

PORT

(

RBIN : IN STD\_LOGIC;

LTN : IN STD\_LOGIC;

BIN : IN STD\_LOGIC;

D : IN STD\_LOGIC;

C : IN STD\_LOGIC;

B : IN STD\_LOGIC;

A : IN STD\_LOGIC;

YA : OUT STD\_LOGIC;

YB : OUT STD\_LOGIC;

YC : OUT STD\_LOGIC;

YD : OUT STD\_LOGIC;

RBON : OUT STD\_LOGIC;

YE : OUT STD\_LOGIC;

YF : OUT STD\_LOGIC;

YG : OUT STD\_LOGIC

);

END C7448;

ARCHITECTURE bdf\_type OF C7448 IS

SIGNAL SYNTHESIZED\_WIRE\_77 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_78 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_79 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_80 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_81 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_82 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_83 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_7 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_84 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_85 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_36 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_53 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_54 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_55 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_56 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_57 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_58 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_59 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_60 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_61 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_62 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_63 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_64 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_68 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_69 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_71 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_72 : STD\_LOGIC;

SIGNAL SYNTHESIZED\_WIRE\_73 : STD\_LOGIC;

BEGIN

RBON <= SYNTHESIZED\_WIRE\_36;

SYNTHESIZED\_WIRE\_72 <= SYNTHESIZED\_WIRE\_77 AND SYNTHESIZED\_WIRE\_78 AND SYNTHESIZED\_WIRE\_79;

SYNTHESIZED\_WIRE\_36 <= NOT(SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_81 AND SYNTHESIZED\_WIRE\_82 AND SYNTHESIZED\_WIRE\_83 AND SYNTHESIZED\_WIRE\_7 AND LTN);

SYNTHESIZED\_WIRE\_82 <= NOT(D);

SYNTHESIZED\_WIRE\_7 <= NOT(RBIN);

SYNTHESIZED\_WIRE\_54 <= SYNTHESIZED\_WIRE\_82 AND SYNTHESIZED\_WIRE\_81 AND SYNTHESIZED\_WIRE\_79;

SYNTHESIZED\_WIRE\_85 <= NOT(SYNTHESIZED\_WIRE\_82 AND SYNTHESIZED\_WIRE\_84);

SYNTHESIZED\_WIRE\_77 <= NOT(SYNTHESIZED\_WIRE\_81 AND SYNTHESIZED\_WIRE\_84);

SYNTHESIZED\_WIRE\_81 <= NOT(C AND LTN);

SYNTHESIZED\_WIRE\_78 <= NOT(SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_84);

SYNTHESIZED\_WIRE\_80 <= NOT(B AND LTN);

SYNTHESIZED\_WIRE\_79 <= NOT(SYNTHESIZED\_WIRE\_83 AND SYNTHESIZED\_WIRE\_84);

SYNTHESIZED\_WIRE\_83 <= NOT(A AND LTN);

SYNTHESIZED\_WIRE\_57 <= SYNTHESIZED\_WIRE\_77 AND SYNTHESIZED\_WIRE\_78 AND SYNTHESIZED\_WIRE\_79;

SYNTHESIZED\_WIRE\_55 <= SYNTHESIZED\_WIRE\_78 AND SYNTHESIZED\_WIRE\_81;

SYNTHESIZED\_WIRE\_53 <= SYNTHESIZED\_WIRE\_79 AND SYNTHESIZED\_WIRE\_78;

SYNTHESIZED\_WIRE\_71 <= SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_77;

SYNTHESIZED\_WIRE\_68 <= SYNTHESIZED\_WIRE\_77 AND SYNTHESIZED\_WIRE\_85;

SYNTHESIZED\_WIRE\_59 <= SYNTHESIZED\_WIRE\_78 AND SYNTHESIZED\_WIRE\_85;

SYNTHESIZED\_WIRE\_64 <= SYNTHESIZED\_WIRE\_83 AND SYNTHESIZED\_WIRE\_77;

SYNTHESIZED\_WIRE\_62 <= SYNTHESIZED\_WIRE\_78 AND SYNTHESIZED\_WIRE\_85;

SYNTHESIZED\_WIRE\_84 <= BIN AND SYNTHESIZED\_WIRE\_36;

SYNTHESIZED\_WIRE\_73 <= SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_81 AND SYNTHESIZED\_WIRE\_82 AND LTN;

SYNTHESIZED\_WIRE\_63 <= SYNTHESIZED\_WIRE\_79 AND SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_81 AND SYNTHESIZED\_WIRE\_82;

SYNTHESIZED\_WIRE\_58 <= SYNTHESIZED\_WIRE\_77 AND SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_83;

SYNTHESIZED\_WIRE\_56 <= SYNTHESIZED\_WIRE\_81 AND SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_79;

SYNTHESIZED\_WIRE\_69 <= SYNTHESIZED\_WIRE\_81 AND SYNTHESIZED\_WIRE\_78 AND SYNTHESIZED\_WIRE\_83;

YF <= NOT(SYNTHESIZED\_WIRE\_53 OR SYNTHESIZED\_WIRE\_54 OR SYNTHESIZED\_WIRE\_55);

YD <= NOT(SYNTHESIZED\_WIRE\_56 OR SYNTHESIZED\_WIRE\_57 OR SYNTHESIZED\_WIRE\_58);

YB <= NOT(SYNTHESIZED\_WIRE\_59 OR SYNTHESIZED\_WIRE\_60 OR SYNTHESIZED\_WIRE\_61);

YA <= NOT(SYNTHESIZED\_WIRE\_62 OR SYNTHESIZED\_WIRE\_63 OR SYNTHESIZED\_WIRE\_64);

SYNTHESIZED\_WIRE\_60 <= SYNTHESIZED\_WIRE\_77 AND SYNTHESIZED\_WIRE\_78 AND SYNTHESIZED\_WIRE\_83;

YC <= NOT(SYNTHESIZED\_WIRE\_68 OR SYNTHESIZED\_WIRE\_69);

YE <= NOT(SYNTHESIZED\_WIRE\_79 OR SYNTHESIZED\_WIRE\_71);

YG <= NOT(SYNTHESIZED\_WIRE\_72 OR SYNTHESIZED\_WIRE\_73);

SYNTHESIZED\_WIRE\_61 <= SYNTHESIZED\_WIRE\_77 AND SYNTHESIZED\_WIRE\_80 AND SYNTHESIZED\_WIRE\_79;

END bdf\_type;

3.激励模块代码（Test Bench）

下面的代码是针对计算输出模块的

LIBRARY ieee;

USE ieee.std\_logic\_1164.all;

ENTITY CALCULATOR\_vhd\_tst IS

END CALCULATOR\_vhd\_tst;

ARCHITECTURE CALCULATOR\_arch OF CALCULATOR\_vhd\_tst IS

SIGNAL ADD1 : STD\_LOGIC;

SIGNAL BCDOUT1 : STD\_LOGIC\_VECTOR(11 DOWNTO 0);

SIGNAL CLK1 : STD\_LOGIC;

SIGNAL DIN1 : STD\_LOGIC\_VECTOR(10 DOWNTO 0);

SIGNAL DIV1 : STD\_LOGIC;

SIGNAL DOUT1 : STD\_LOGIC\_VECTOR(23 DOWNTO 0);

SIGNAL MUL1 : STD\_LOGIC;

SIGNAL RST1 : STD\_LOGIC;

SIGNAL SUB1 : STD\_LOGIC;

COMPONENT CALCULATOR

PORT (

ADD : IN STD\_LOGIC;

BCDOUT : BUFFER STD\_LOGIC\_VECTOR(11 DOWNTO 0);

CLK : IN STD\_LOGIC;

DIN : IN STD\_LOGIC\_VECTOR(10 DOWNTO 0);

DIV : IN STD\_LOGIC;

DOUT : BUFFER STD\_LOGIC\_VECTOR(23 DOWNTO 0);

MUL : IN STD\_LOGIC;

RST : IN STD\_LOGIC;

SUB : IN STD\_LOGIC

);

END COMPONENT;

CONSTANT CLK\_P: TIME:=5 US;--这里如果是NS得话将会抖动巨大

BEGIN

i1 : CALCULATOR

PORT MAP (

ADD => ADD1,

BCDOUT => BCDOUT1,

CLK => CLK1,

DIN => DIN1,

DIV => DIV1,

DOUT => DOUT1,

MUL => MUL1,

RST => RST1,

SUB => SUB1

);

PROCESS BEGIN

CLK1<='0'; WAIT FOR CLK\_P;

CLK1<='1'; WAIT FOR CLK\_P;

END PROCESS;

RST1<='1','0' AFTER 300 US;

DIN1<="00000010111", "00000010000" AFTER 5000 US ,"00000001000" AFTER 6000 US,"00000001100" AFTER 6600 US,"00000000100" AFTER 6850 US;

ADD1<='0','1' AFTER 400 US,'0' AFTER 600 US,'1' AFTER 5500 US,'0' AFTER 5800 US;

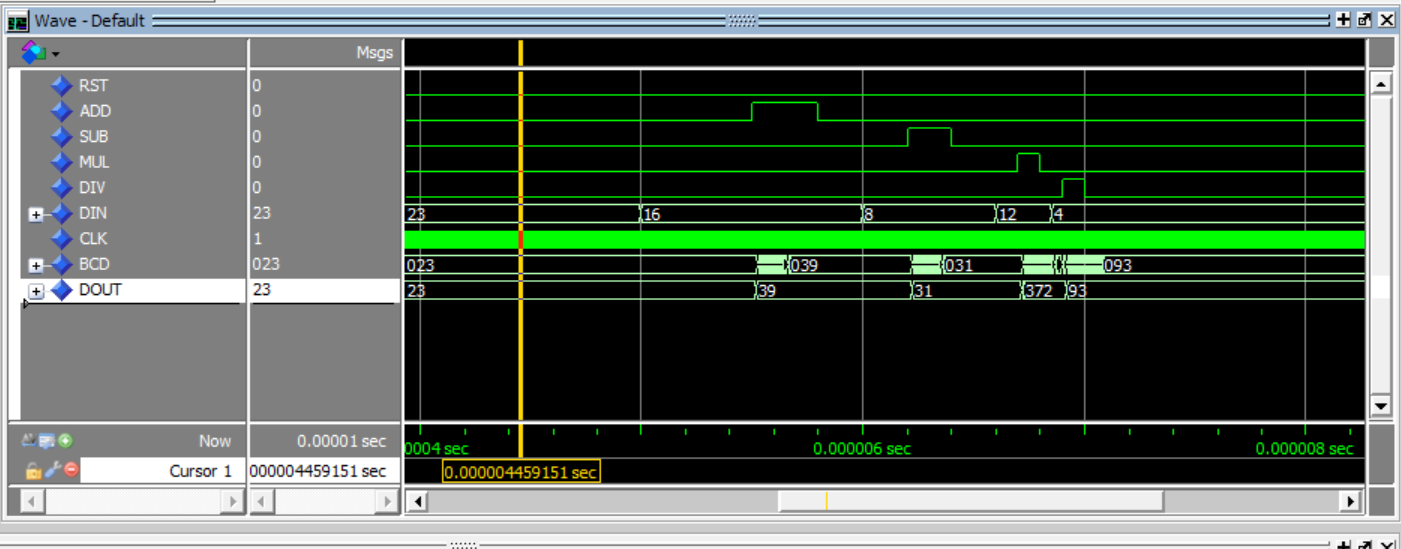
MUL1<='0','1' AFTER 6700 US,'0' AFTER 6800 US;

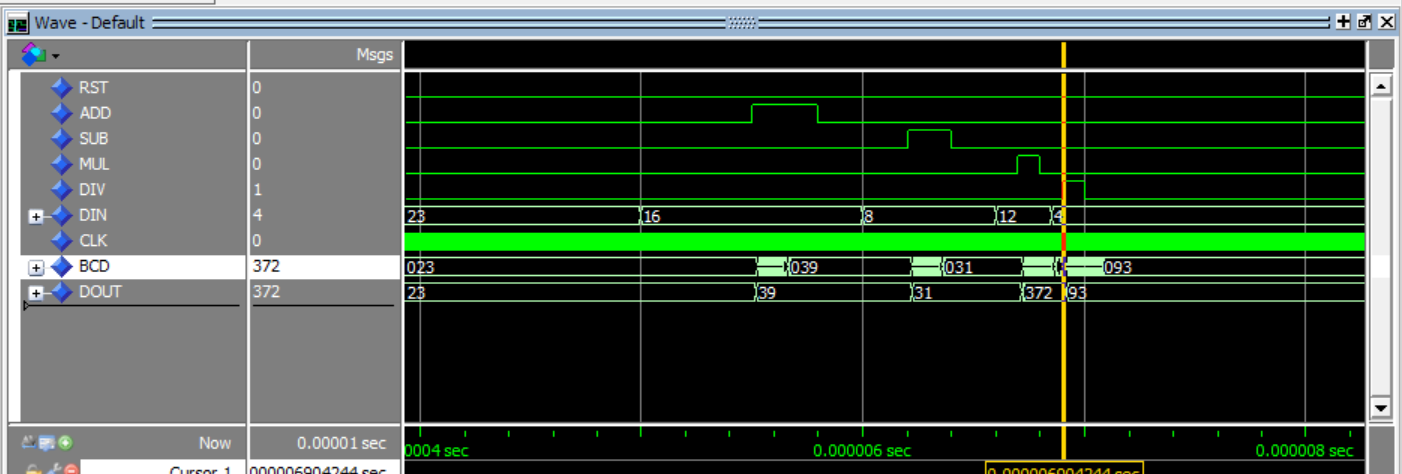
SUB1<='0','1' AFTER 6200 US,'0' AFTER 6400 US;

DIV1<='0','1' AFTER 6900 US,'0' AFTER 7000 US;

END CALCULATOR\_arch;

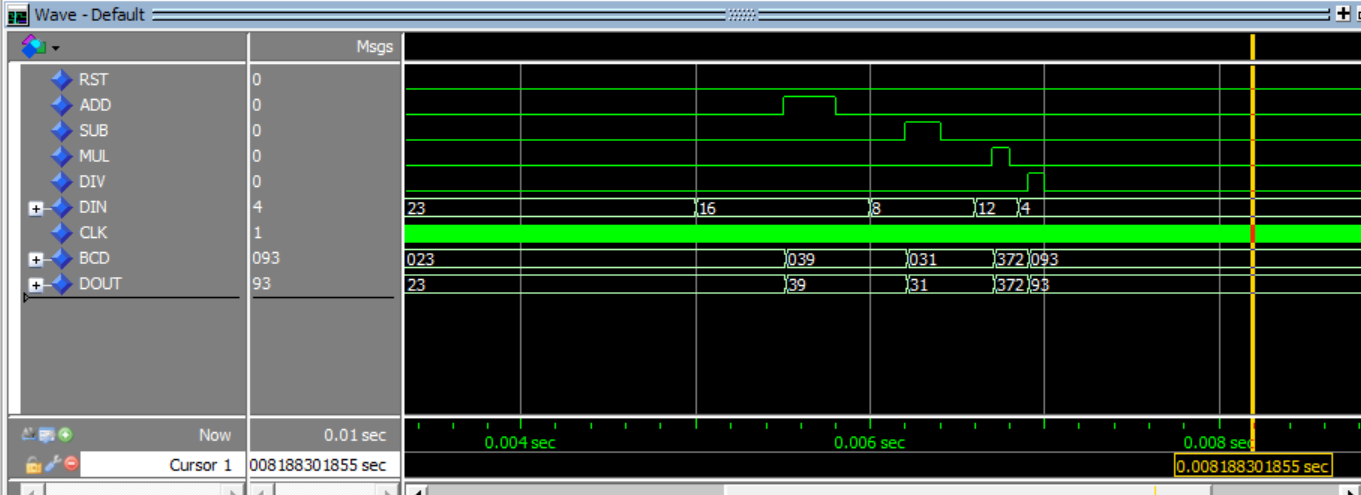
4.仿真波形图



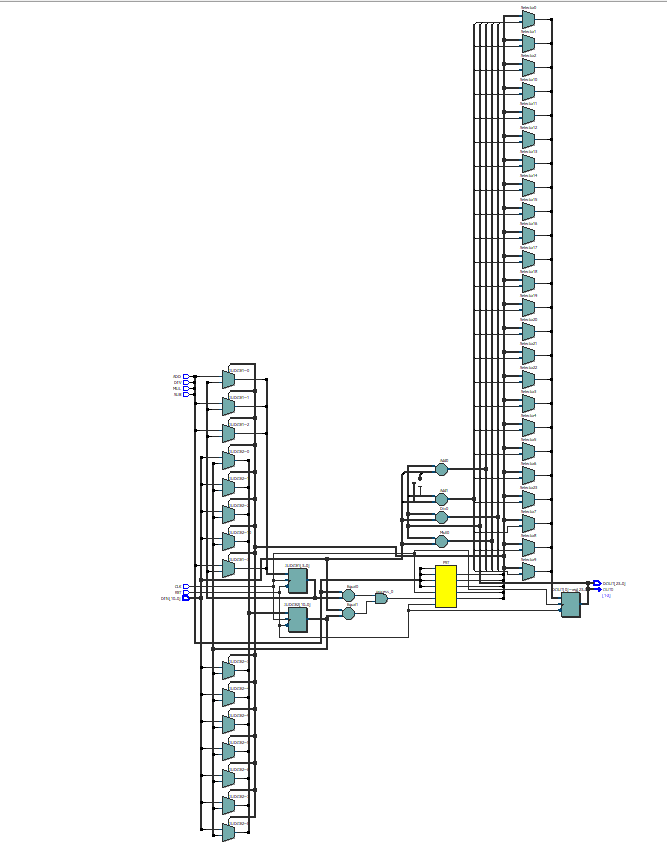
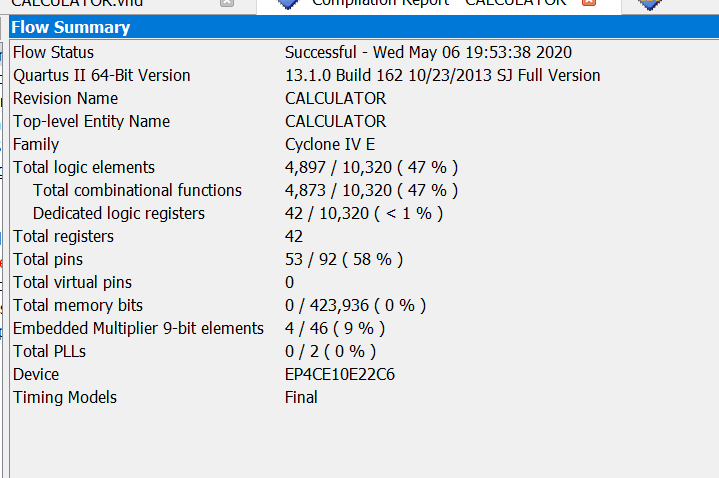


模拟的情况是（23+16-8）\*12/4，中间过程的结果也如图所示，连续的输入数据及算符，可实现连续的计算，图中BCD一行是以16进制显示的，可见符合BCD码要求，但是由于仿真波形为NS级别，加之占用资源较大，计算较慢，其抖动很大，372这个数字停留时间较短，甚至只显示了短暂的时间，若现实中的仿真并不会出现这种情况，DOUT以十进制形式输出，也满足输出要求

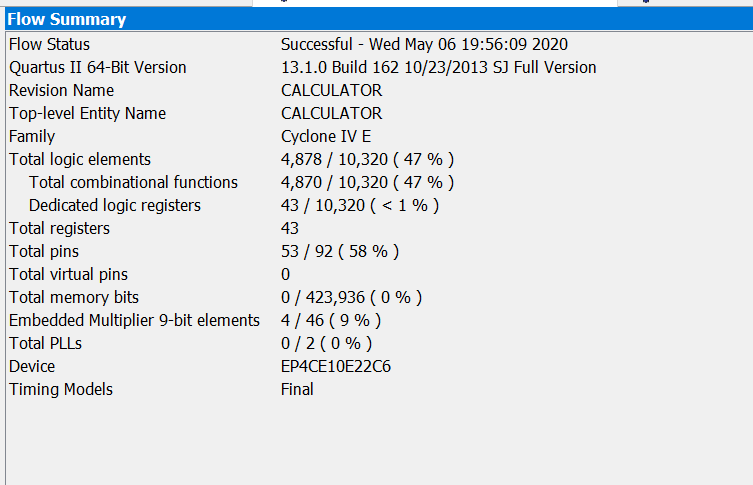
改为US级别以后，抖动即已消失

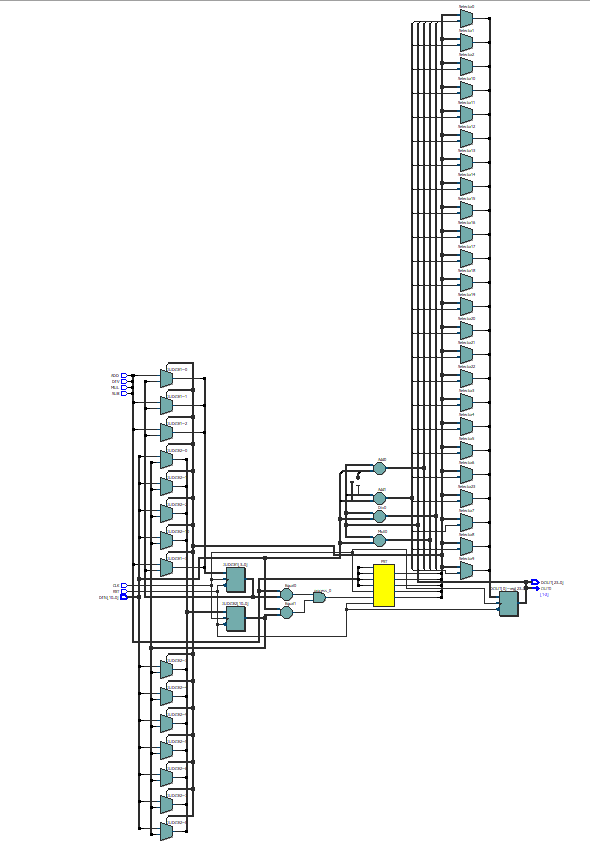


5.门级电路图

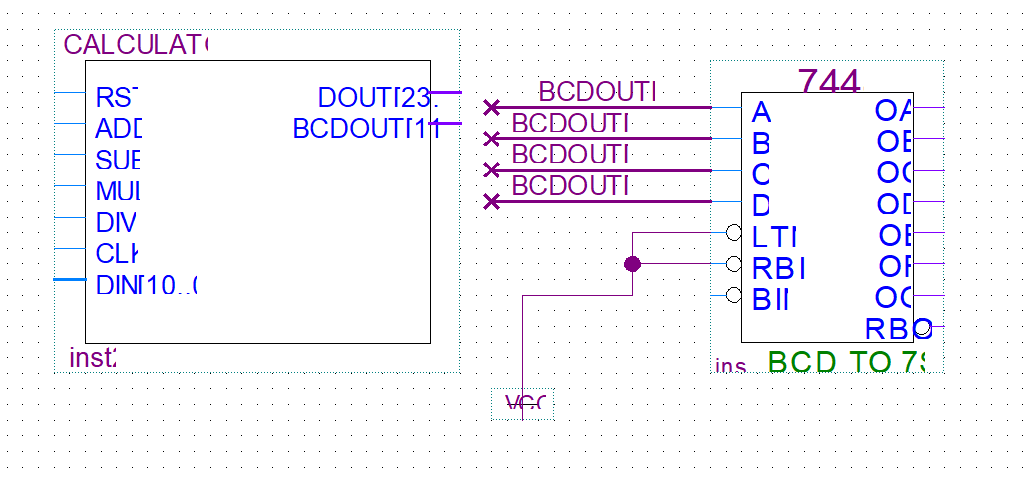
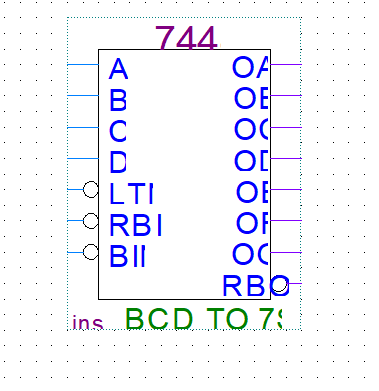


上两图为格雷码资源占用及电路图





上两图为one-hot码占用及电路图，可见格雷码资源占用要稍小一些



如图为7448译码器，及连接方式示意（图中只连接了最低4位） ，再将OA~OG（对应YA~YG）连接到七段共阴数码管即可实现LED显示

6.结果分析与思考

（1）该电路能够实现连续输入数字连续做运算的功能，但占用资源巨大，且速度较为缓慢（在NS级别的仿真抖动很大），适用于US级别以上的运算速度。限于计算机性能，并便于仿真，贴出的程序里将转BCD模块的位数降低了（为12位），如果需要可以扩展到更大位数

（2）该电路的另一个缺点在于连续输入两个相同数的时候不能响应运算，这时只需要在中间插入一个“加0”操作即可，如果进行实际硬件设计，只需要加一个定时功能，响应输入算符后一定时间内的数字输入即可

（3）该计算器所使用的计算单元都是由quartus内部运算符号综合而来的，理论上讲效率不高，而且除法会舍去余数

（4）该设计为了便于仿真，没有加入“完成输入”这一个标志端口，是通过判断前后运算中数据和算符是否相同实现的，所以有一定的局限性，若加以外部设备，如键盘，再增加一个端口，则这一判断会变得简单许多也灵活许多