实验一、四位二进制加法器

实验源程序：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity zq is

port(clk:in std\_logic;

rst:in std\_logic;

q:out std\_logic\_vector(3 downto 0));

end entity zq;

architecture bhv of zq is

signal q1:std\_logic\_vector(3 downto 0);

begin

process(rst,clk)

begin

if(clk'event and clk='1')then

q1<=q1-1;

end if;

if(rst='0')then

q1<="1111";

end if;

end process;

q<=q1;

end architecture bhv;

实验步骤：代码编译后进行引脚绑定，clk时钟引脚绑定引脚P20，rst为复位端，绑定SW-1的N18引脚，其余引脚绑定：q3---🡪U12 q2--🡪V12 q1--🡪V15 q0---🡪W13

实验原理：代码表示在内部对4位二进制数q进行减1计数，由于led是反逻辑，为0时点亮，q=1111时全灭。

实验现象：4个led灯按照0000-🡪1111的顺序依次点亮（1为亮），时钟频率为1HZ，rst=0时，led全灭。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验二、半加器程序设计

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity zq is

port(a:in std\_logic;

b:in std\_logic;

sum:out std\_logic;

carry:out std\_logic);

end entity zq;

architecture fh1 of zq is

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

begin

state<=a&b;

process(state)

begin

case state is

when "00"=>sum<='1';carry<='1';

when "01"=>sum<='0';carry<='1';

when "10"=>sum<='0';carry<='1';

when "11"=>sum<='1';carry<='0';

when others=>null;

end case;

end process;

end architecture fh1;

实验步骤：代码编译后进行引脚绑定，a🡪N18(SW-1),b🡪M20(SW-2),sum🡪U12（LED1）,carry🡪V12（LED2）

实验原理：结构体内部算法对ab进行判断再对sum、carry赋值，从而实现半加器的逻辑运算。

实验现象：用SW逻辑电平开关作为输入信号a，b的值，当ab=00时，sum=carry=0；ab=01 or 10时，sum=1，carry=0；ab=11时，sum=1，carry=1.实现一位半加器的功能。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验三、数码管计数显示

实验源程序：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity zq is

port(clk:in std\_logic;

rst:in std\_logic;

sel:out std\_logic;//weixuan AB20 DP1

q:out std\_logic\_vector(3 downto 0);

d:out std\_logic\_vector(7 downto 0)

);

end entity zq;

architecture fh1 of zq is

signal q1,q2:std\_logic\_vector(3 downto 0);

begin

process(clk,rst)

begin

if(clk'event and clk='1')then

q1<=q1-1;

end if;

if(rst='0')then

q1<="1111";

end if;

end process;

q<=q1;

process(q1)

begin

if(rst='1')then sel<='0';

case q1 is

when "1111"=>d<="11111100";//a - h jian shu biao

when "1110"=>d<="01100000";

when "1101"=>d<="11011010";

when "1100"=>d<="11110010";

when "1011"=>d<="01100110";

when "1010"=>d<="10110110";

when "1001"=>d<="10111110";

when "1000"=>d<="11100000";

when "0111"=>d<="11111110";

when "0110"=>d<="11110110";

when "0101"=>d<="11101110";

when "0100"=>d<="00111110";

when "0011"=>d<="10011100";

when "0010"=>d<="01111010";

when "0001"=>d<="10011110";

when "0000"=>d<="10001110";

end case;

end if;

end process;

end architecture fh1;

实验步骤：代码编译成功后引脚绑定，绑定引脚：

clk：P20；rst：N18(SW-1)；sel：AB20 --位选地址；d7：AA20----A段选地址；d6：W20----B段选地址；d5：R21----C段选地址；d4：P21----D段选地址 d5：N21------E段选地址 d6：N20------F段选地址 d7：M21-----G段选地址 d8：M19----H段选地址

q3---🡪U12（LED1） q2--🡪V12(LED2) q1--🡪V15(LED3) q0---🡪W13(LED4)

实验原理：代码一方面执行led的点亮操作，另一方面同时利用case语句对8位二进制数d进制编码操作，数码管为共阳接法，通过三极管驱动，依照各段点亮可以推出8个三极管的基极电流触发，三极管作为开关电路的时刻，进而得到d的编码值。

实验现象：rst为1时，4个led 依次计数点亮，同时数码管最低位亮，显示1-F 16个数字计数，rst=0时清零。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验四：16x16点阵显示

实验源程序：

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity cn4 is

generic(n:integer:=72);

port(clk,mode,rst:in std\_logic;

row:out std\_logic\_vector(15 downto 0);

col:out std\_logic\_vector(3 downto 0));

end cn4;

architecture behave of cn4 is

type code is array(0 to n-1)of std\_logic\_vector(15 downto 0);

constant

code\_0:code:=(x"0003",x"000c",x"0030",x"0090",x"0090",x"0030",x"000c",

x"0003",x"0000",x"00ff",x"0091",x"0091",x"0091",x"00aa",x"0044",x"0000",x"0000",x"0000",

x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",

x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",

x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",

x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",x"0000",

x"0000",x"0000");

signal cntscan,frame:std\_logic\_vector(3 downto 0);

signal i,j,f:integer range 0 to n-1;

signal cnt,cnt1:integer range 0 to 20;

begin

process(clk,frame,rst)

begin

if rst='0'then

cntscan<="0000";frame<="0000";i<=0;j<=0;cnt<=0;cnt1<=0;

row<=x"0000";

elsif clk'event and clk='1'then

if mode='0'then

if f=4 then

f<=0;

j<=0;

else

row<=code\_0(conv\_integer(cntscan)+j);

col<=cntscan;

if cntscan="1111" then

cntscan<="0000";

cnt1<=cnt1+1;

else

cntscan<=cntscan+1;

end if;

if cnt1=30 then

j<=j+18;

f<=f+1;

cnt1<=0;

end if;

end if;

else

col<=frame;

case frame is

when "0000"=>

row<=code\_0((i)mod n);

when "0001"=>

row<=code\_0((i+1)mod n);

when "0010"=>

row<=code\_0((i+2)mod n);

when "0011"=>

row<=code\_0((i+3)mod n);

when "0100"=>

row<=code\_0((i+4)mod n);

when "0101"=>

row<=code\_0((i+5)mod n);

when "0110"=>

row<=code\_0((i+6)mod n);

when "0111"=>

row<=code\_0((i+7)mod n);

when "1000"=>

row<=code\_0((i+8)mod n);

when "1001"=>

row<=code\_0((i+9)mod n);

when "1010"=>

row<=code\_0((i+10)mod n);

when "1011"=>

row<=code\_0((i+11)mod n);

when "1100"=>

row<=code\_0((i+12)mod n);

when "1101"=>

row<=code\_0((i+13)mod n);

when "1110"=>

row<=code\_0((i+14)mod n);

when "1111"=>

row<=code\_0((i+15)mod n);

i<=i+1;

cnt<=cnt+1;

frame<="0000";

when others=>

null;

end case;

if i=n-1 then

i<=0;

else

if cnt=10 then

i<=i+1;

cnt<=0;

end if;

end if;

frame<=frame+1;

end if;

end if;

end process;

end behave;

实验步骤：代码编译成功后引脚绑定，绑定引脚：

clk绑定P20引脚输入时钟信号、Mode绑定M20(SW-2) rst 绑定N18（SW-1）

row0：A4 row1:A5 row2: A6 row3:B6 row4:E11 row5:C13 row6:F11 row7:C15

row8:E14 row9:B7 row10:B8 row11:B9 row12:B10 row13:D13 row14:F9

row15:A13 col0:A14 col1:A15 col2:A16 col3:C4

实验原理：array为所用字库，程序中定义列扫描模式，在循环语句中，时钟脉冲下依次显示字库中的数据点亮对应的LED，led进而实现在16x16点阵上的字符显示。

实验现象：点阵上显示AB两个字母。mode可以控制显示模式，模式有从右向左动态运动的AB字，还有静态显示的AB字。通过调节clk接入的时钟频率，可以调节点阵显示的亮度和扫描显示的时间。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验五、4D触发器设计

library ieee;

use ieee.Std\_logic\_1164.all;

entity HCT175 is

port(D : in std\_logic\_vector(3 downto 0);

Q : out std\_logic\_vector(3 downto 0);

CLRBAR, CLK : in std\_logic);

end HCT175;

architecture VER1 of HCT175 is

begin

Q <= (others => '0') when (CLRBAR = '0')

else D when rising\_edge(CLK)

else unaffected;

end VER1;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验六、4x4矩阵键盘

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.std\_logic\_arith.all;

entity test is

port (clk: in std\_logic; --系统内或外时钟信号

start: in std\_logic; ---外控信号

KBCol: in std\_logic\_vector (3 downto 0);--列扫描信号

KBRow: out std\_logic\_vector (3 downto 0);--行扫描信号

seg7: out std\_logic\_vector (6 downto 0);--数码管7段显示信号

scan: out std\_logic\_vector (7 downto 0));--数码管位控信号

end;

architecture bev of test is

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

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

begin

scan <="11111110"; --用一个数码管

a:

process(clk) ----循环扫描计数器

begin

if (clk'event and clk='1') then

count<=count+1;

end if;

end process a;

b:

process(clk) -----循环列扫描进程

begin

if (clk'event and clk='1') then

case count (1 downto 0) is

when "00"=>KBRow<="0111";

sta<="00";

when "01"=>KBRow<="1011";

sta<="01";

when "10"=>KBRow<="1101";

sta<="10";

when "11"=>KBRow<="1110";

sta<="11";

when others=>KBRow<="1111";

end case;

end if;

end process b;

c:

process(clk, start) -----行扫描译码进程

begin

if(start='0')then

seg7<="0000000";

elsif (clk'event and clk='1') then

case sta is

when"00"=> --cdef列

case KBCol is

when "1110"=> seg7<="1001110";--"1001110";--c c

when "1101"=> seg7<="0111101";--"1111111";--8 d

when "1011"=> seg7<="1001111";--"0110011";--4 e

when "0111"=> seg7<="1000111";--"1111110";--0 f

when others=> seg7<="0000000";--1取消这四行结果会有什么变化？

end case;

when"01"=> --89ab列

case KBCol is

when "1110"=> seg7<="1111111";--"0111101";--d 8

when "1101"=> seg7<="1110011";--"1110011";--9 9

when "1011"=> seg7<="1110111";--"1011011";--5 a

when "0111"=> seg7<="0011111";--"0110000";--1 b

when others=> seg7<="0000000";--1取消这四行结果会有什么变化？

end case;

when"10"=> --4567列

case KBCol is

when "1110"=> seg7<="0110011";--"1001111";--e 4

when "1101"=> seg7<="1011011";--"1110111";--a 5

when "1011"=> seg7<="1011111";--"1011111";--6 6

when "0111"=> seg7<="1110000";--"1101101";--2 7

when others=> seg7<="0000000";--1取消这四行结果会有什么变化？

end case;

when "11"=> --0123

case KBCol is

when "1110"=> seg7<="1111110";--"1000111";--f 0

when "1101"=> seg7<="0110000";--"0011111";--b 1

when "1011"=> seg7<="1101101";--"1110000";--7 2

when "0111"=> seg7<="1111001";--"1111001";--3 3

when others=> seg7<="0000000";--1取消这四行结果会有什么变化？

end case;

when others=> seg7<="0000000";

end case;

end if;

end process c;

end bev;

实验原理：

实验步骤：

实验现象：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验七、1602 LCD显示任意字符实验

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_arith.all;

use ieee.std\_logic\_unsigned.all;

entity test is

generic (N:integer:=200;

delay:integer:=100);

port (clk:in std\_logic;--系统时钟输入

reset: in std\_logic;--复位信号，在下完程序之后需要对液晶进行复位清零

oe: out std\_logic;--接LCD使能端

rs: out std\_logic;--接LCD\_da信号输入端

rw: out std\_logic;--接LCD读写信号输入端

data: out std\_logic\_vector(7 downto 0));--接LCD数据输入位

end;

architecture behavioral of test is

type state is

(clear\_lcd, entry\_set, display\_set, funtion\_set, position\_set1, write\_data1, position\_set2, write\_data2,stop);

signal current\_state: state:=clear\_lcd;---写指令，写数据状态

type ram is array(0 to 23) of std\_logic\_vector(7 downto 0);

signal dataram: ram:=(("00110000"), ("00110000"), ("00111010"), ("00110000"),

("00110000"), ("00111010"), ("00110000"), ("00110000"),---时钟数据存储

x"80", x"80", x"5A", x"51", x"2D" x"31", x"34", x"30", x"33", x"31", x"32", x"30", x"35", x"80", x"80", x"80"); --显示学号：ZQ-14031205

--若为www.BUAA.edu.cn（x"77", x"77", x"77", x"2E", x"42", x"55", x"41", x"41", x"2E", x"65", x"64", x"75", x"2E", x"63", x"6e", x"80"）

signal clk\_250Khz,clk\_1Hz:std\_logic;

signal cnt1, cnt2: integer range 0 to 200000;

signal

hour\_h\_tmp, hour\_l\_tmp, min\_h\_tmp, min\_l\_tmp, sec\_h\_tmp, sec\_l\_tmp: std\_logic\_vector(3 downto 0):="0000";

begin

--（液晶）数据交换频率

lcd\_clk:

process(clk, reset)

variable c1: integer range 0 to 100;

variable c2: integer range 0 to 50000000;

variable clk0, clk1:std\_logic;

begin

if(reset='0')then

c1:=0; c2:=0;

elsif (clk'event and clk='1') then

If (c1=N/2-1) then------250K时钟分频

c1:=0;

clk0:=not clk0;

else

c1:=c1+1;

end if;

if c2=50000000/2-1 then--1Hz时钟分频，用于时钟计数

c2:=0;

clk1:=not clk1;

else

c2:=c2+1;

end if;

end if;

clk\_250Khz<=clk0;

clk\_1hz<=clk1;

end process;

write:

process(clk\_250Khz, reset)

begin

if (clk\_250Khz'event and clk\_250Khz='1') then----将时钟计数结果存于dataram中，前面加“0011”是

dataram(0)<="0011" & hour\_h\_tmp; --根据0-9对应地址和数字的关系得到的

dataram(1)<="0011" & hour\_l\_tmp; --此处的时钟用ms级就可以，因为时钟每隔1秒才刷新一次

dataram(3)<="0011" & min\_h\_tmp;

dataram(4)<="0011" & min\_l\_tmp;

dataram(6)<="0011" & sec\_h\_tmp;

dataram(7)<="0011" & sec\_l\_tmp;

end if;

end process;

--液晶驱动部分

control:

process(clk\_250Khz, reset)

--variable cnt3: std\_logic\_vector (3 downto 0);

begin

if (reset='0') then

current\_state<=clear\_lcd;

cnt1<=0; cnt2<=0;

elsif rising\_edge(clk\_250Khz) then

case current\_state is

when clear\_lcd=>--清屏，文档中要求至少延时1.64ms

oe<='1';

rs<='0';

rw<='0';

data<=x"01";

cnt1<=cnt1+1;

if cnt1>delay\*1 and cnt1<=delay\*6 then--延时操作

oe<='0'; --保证液晶有足够的使能时间

else

oe<='1';

end if;

if cnt1=delay\*7 then

current\_state<=entry\_set;

cnt1<=0;

end if;

when entry\_set=>--写入新数据后光标右移，写入新数据后显示屏不移动

oe<='1';

rs<='0';

rw<='0';

data<=x"06";

cnt1<=cnt1+1;

if (cnt1>delay\*1 and cnt1<=delay\*2 )then--延时操作

oe<='0'; --保证液晶有足够的使能时间

else

oe<='1';

end if;

if (cnt1=delay\*3 )then

current\_state<=display\_set;

cnt1<=0;

end if;

when display\_set=>--显示方式：控制显示器开/关、光标显示/关闭以及光标是否闪烁

oe<='1';

rs<='0';

rw<='0';

data<=x"0C";

cnt1<=cnt1+1;

if (cnt1>delay and cnt1<=delay\*2 )then--延时操作

oe<='0'; --保证液晶有足够的使能时间

else

oe<='1';

end if;

if (cnt1=delay\*3) then

current\_state<=funtion\_set;

cnt1<=0;

end if;

when funtion\_set=>--功能设置：设定数据总线位数、显示的行数及字型

oe<='1';--设置为8位并行，显示2行，5\*7点阵显示

rs<='0';

rw<='0';

data<=x"38";

cnt1<=cnt1+1;

if (cnt1>delay and cnt1<=delay\*2 )then--延时操作

oe<='0'; --保证液晶有足够的使能时间

else

oe<='1';

end if;

if (cnt1=delay\*3) then

current\_state<=position\_set1;

cnt1<=0;

end if;

when position\_set1=> --设置显示数码的初始位置

oe<='1';

rs<='0';

rw<='0';

data<=x"38";

cnt1<=cnt1+1;

if (cnt1>delay and cnt1<=delay\*2 )then--延时操作

oe<='0'; --保证液晶有足够的使能时间

else

oe<='1';

end if;

if (cnt1=delay\*3 )then

current\_state<=write\_data1;

cnt1<=0;

end if;

when write\_data1=> --将数据写入液晶

oe<='1';

rs<='1';

rw<='0';

if cnt2<=7 then

data<=dataram(cnt2);

cnt1<=cnt1+1;

if (cnt1>delay and cnt1<=delay\*2) then

oe<='0';

else

oe<='1';

end if;

if (cnt1=delay\*3) then

current\_state<=write\_data1;

cnt1<=0;

cnt2<=cnt2+1;

end if;

else

cnt2<=0;

current\_state<=position\_set2;

end if;

when position\_set2=>--设置显示数据的初始位置，第二行的地址设置

oe<='1';

rs<='0';

rw<='0';

data<=x"c0";

cnt1<=cnt1+1;

if (cnt1>delay and cnt1<=delay\*2) then--延时操作

oe<='0'; --保证液晶有足够的使能时间

else

oe<='1';

end if;

if (cnt1=delay\*3) then

current\_state<=write\_data2;

cnt1<=0;

end if;

when write\_data2=> --将数据写入液晶

oe<='1';

rs<='1';

rw<='0';

if cnt2<=15 then

data<=dataram (cnt2+8);

cnt1<=cnt1+1;

if( cnt1>delay and cnt1<=delay\*2 )then

oe<='0';

else

oe<='1';

end if;

if (cnt1=delay\*3 )then

current\_state<=write\_data2;

cnt1<=0;

cnt2<=cnt2+1;

end if;

else

cnt2<=0;

current\_state<=position\_set1;

end if;

When stop=>

null;

end case;

end if;

end process;

clock:

process (clk\_1hz, reset)-----------时钟单元

begin

if reset='0' then

hour\_h\_tmp<="0000";

hour\_l\_tmp<="0000";

min\_h\_tmp<="0000";

min\_l\_tmp<="0000";

sec\_h\_tmp<="0000";

sec\_l\_tmp<="0000";

elsif (clk\_1hz'event and clk\_1hz='1')then

if sec\_l\_tmp="1001" then

sec\_l\_tmp<="0000";

if sec\_h\_tmp="0101" then

sec\_h\_tmp<="0000";

if min\_l\_tmp="1001" then

min\_l\_tmp<="0000";

if min\_h\_tmp="0101" then

min\_h\_tmp<="0000";

if hour\_h\_tmp="0010" then

if hour\_l\_tmp="0011" then

hour\_l\_tmp<="0000";

hour\_h\_tmp<="0000";

else

hour\_l\_tmp<=hour\_l\_tmp+'1';

end if;

else

if hour\_l\_tmp="1001"then

hour\_l\_tmp<="0000";

hour\_h\_tmp<=hour\_h\_tmp+'1';

else

hour\_l\_tmp<=hour\_l\_tmp+'1';

end if;

end if;

else

min\_h\_tmp<=min\_h\_tmp+'1';

end if;

else

min\_l\_tmp<=min\_l\_tmp+'1';

end if;

else

sec\_h\_tmp<=sec\_h\_tmp+'1';

end if;

else

sec\_l\_tmp<=sec\_l\_tmp+'1';

end if;

end if;

end process;

end behavioral;

实验步骤：

实验原理：

实验现象：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验八、4位十进制数字频率计

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity plj is

port ( start:in std\_logic; --复位信号

clk :in std\_logic; --系统时钟

clk1:in std\_logic; --被测信号

yy1:out std\_logic\_vector(7 downto 0); --八段码

w1 :out std\_logic\_vector(3 downto 0)); --数码管位选信号

end plj;

architecture behav of PLj is

signal b1,b2,b3,b4,b5,b6,b7:std\_logic\_vector(3 downto 0); --十进制计数器

signal bcd:std\_logic\_vector(3 downto 0); --BCD码寄存器

signal q :integer range 0 to 49999999; --秒分频系数

signal qq : integer range 0 to 499999; --动态扫描分频系数

signal en,bclk:std\_logic; --使能信号，有效被测信号

signal sss : std\_logic\_vector(3 downto 0); --小数点

signal bcd0,bcd1,bcd2,bcd3 : std\_logic\_vector(3 downto 0);

--寄存7位十位计数器中有效的高4位数据

begin

second:process(clk) --此进程产生一个持续时间为一秒的的闸门信号

begin

if start='1' then q<=0;

elsif clk'event and clk='1' then

if q<49999999 then q<=q+1;

else q<=49999999;

end if;

end if;

if q<49999999 and start='0' then en<='1';

else en<='0';

end if;

end process;

and2:process(en,clk1) --此进程得到7位十进制计数器的计数脉冲

begin

bclk<=clk1 and en;

end process;

com:process(start,bclk) --此进程完成对被测信号计脉冲数

begin

if start='1' then --复位

b1<="0000";b2<="0000";b3<="0000";b4<="0000";b5<="0000";b6<="0000";b7<="0000";

elsif bclk'event and bclk='1' then

if b1="1001" then b1<="0000"; --此IF语句完成个位十进制计数

if b2="1001" then b2<="0000"; --此IF语句完成百位十进制计数

if b3="1001" then b3<="0000"; --此IF语句完成千位十进制计数

if b4="1001" then b4<="0000"; --此IF语句完成万位十进制计数

if b5="1001" THEN b5<="0000"; --此IF语句完成十万位十进制计数

if b6="1001" then b6<="0000"; --此IF语句完成百万位十进制计数

if b7="1001" then b7<="0000"; --此IF语句完成千万位十进制计数

else b7<=b7+1;

end if;

else b6<=b6+1;

end if;

else b5<=b5+1;

end if;

else b4<=b4+1;

end if;

else b3<=b3+1;

end if;

else b2<=b2+1;

end if;

else b1<=b1+1;

end if;

end if;

end process;

process(clk) --此进程把7位十进制计数器有效的高4位数据送如bcd0~3；并得到小数点信息

begin

if rising\_edge(clk) then

if en='0' then

if b7>"0000" then bcd3<=b7; bcd2<=b6; bcd1<=b5; bcd0<=b4; sss<="1110";

elsif b6>"0000" then bcd3<=b6; bcd2<=b5; bcd1<=b4; bcd0<=b3; sss<="1101";

elsif b5>"0000" then bcd3<=b5; bcd2<=b4; bcd1<=b3; bcd0<=b2; sss<="1011";

else bcd3<=b4; bcd2<=b3; bcd1<=b2; bcd0<=b1; sss<="1111";

end if;

end if;

end if;

end process;

weixuan:process(clk) --此进程完成数据的动态显示

begin

if clk'event and clk='1' then

if qq< 99999 then qq<=qq+1;bcd<=bcd3; w1<="0111";

if sss="0111" then yy1(0)<='0';

else yy1(0)<='1';

end if;

elsif qq<199999 then qq<=qq+1;bcd<=bcd2; w1<="1011";

if sss="1011" then yy1(0)<='0';

else yy1(0)<='1';

end if;

elsif qq<299999 then qq<=qq+1;bcd<=bcd1; w1<="1101";

if sss="1101" then yy1(0)<='0';

else yy1(0)<='1';

end if;

elsif qq<399999 then qq<=qq+1;bcd<=bcd0; w1<="1110";

if sss="1110" then yy1(0)<='0';

else yy1(0)<='1';

end if;

else qq<=0;

end if;

end if;

end process;

m0: process (bcd) --译码

begin

case bcd is

when "0000"=>yy1(7 downto 1)<="0000001";

when "0001"=>yy1(7 downto 1)<="1001111";

when "0010"=>yy1(7 downto 1)<="0010010";

when "0011"=>yy1(7 downto 1)<="0000110";

when "0100"=>yy1(7 downto 1)<="1001100";

when "0101"=>yy1(7 downto 1)<="0100100";

when "0110"=>yy1(7 downto 1)<="1100000";

when "0111"=>yy1(7 downto 1)<="0001111";

when "1000"=>yy1(7 downto 1)<="0000000";

when "1001"=>yy1(7 downto 1)<="0001100";

when others=>yy1(7 downto 1)<="1111111";

end case;

end process;

end behav;

实验步骤：

实验原理：

实验现象：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

实验九、六层电梯模拟实验

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.std\_logic\_arith.all;

entity dianti is

port ( clk : in std\_logic; --时钟信号（频率为2Hz）

full,deng,quick,clr : in std\_logic; --超载、关门中断、提前关门清除报警信号

c\_u1,c\_u2,c\_u3,c\_u4,c\_u5: in std\_logic; --电梯外人的上升请求信号

c\_d2,c\_d3,c\_d4,c\_d5,c\_d6 : in std\_logic; --电梯外人的下降请求信号

d1,d2,d3,d4,d5,d6 : in std\_logic; --电梯内人的请求信号

g1,g2,g3,g4,g5,g6 : in std\_logic; --到达楼层信号

door : out std\_logic\_vector(1 downto 0); --电梯门控制信号

led : out std\_logic\_vector(6 downto 0); --电梯所在楼层显示

led\_c\_u:out std\_logic\_vector(5 downto 0); --电梯外人上升请求信号显示

led\_c\_d:out std\_logic\_vector(5 downto 0); --电梯外人下降请求信号显示

led\_d : out std\_logic\_vector(5 downto 0); --电梯内请求信号显示

wahaha : out std\_logic; --看门狗报警信号

ud,alarm : out std\_logic; --电梯运动方向显示，超载警告信号

up,down : out std\_logic ); --电机控制信号和电梯运动

end dianti;

architecture behav of dianti is

signal d11,d22,d33,d44,d55,d66:std\_logic; --电梯内人请求信号寄存信号

signal c\_u11,c\_u22,c\_u33,c\_u44,c\_u55:std\_logic; --电梯外人上升请求信号寄存信号

signal c\_d22,c\_d33,c\_d44,c\_d55,c\_d66:std\_logic; --电梯外人下降请求信号寄存信号

signal q:integer range 0 to 1; --分频信号

signal q1:integer range 0 to 6; --关门延时计数器

signal q2:integer range 0 to 9; --看门狗计数器

signal dd,cc\_u,cc\_d,dd\_cc:std\_logic\_vector(5 downto 0); --电梯内外请求信号寄存器

signal opendoor:std\_logic; --开门使能信号

signal updown:std\_logic; --电梯运动方向信号寄存器

signal en\_up,en\_dw:std\_logic; --预备上升、预备下降预操作使能信号

begin

com:process(clk)

begin

if clk'event and clk='1' then

if clr='1' then q1<=0;q2<=0;wahaha<='0'; --清除故障报警

elsif full='1' then alarm<='1'; q1<=0; --超载报警

if q1>=3 then door<="10";

else door<="00";

end if;

elsif q=1 then q<=0;alarm<='0';

if q2=3 then wahaha<='1'; --故障报警

else

if opendoor='1' then door<="10";q1<=0;q2<=0;up<='0';down<='0'; --开门操作

elsif en\_up='1' then --上升预操作

if deng='1' then door<="10";q1<=0;q2<=q2+1; --关门中断

elsif quick='1' then q1<=3; --提前关门

elsif q1=6 then door<="00";updown<='1';up<='1'; --关门完毕，电梯进入上升状态

elsif q1>=3 then door<="01";q1<=q1+1; --电梯进入关门状态

else q1<=q1+1;door<="00"; --电梯进入等待状态

end if;

elsif en\_dw='1' then --下降预操作

if deng='1' then door<="10";q1<=0;q2<=q2+1;

elsif quick='1' then q1<=3;

elsif q1=6 then door<="00";updown<='0';down<='1';

elsif q1>=3 then door<="01";q1<=q1+1;

else q1<=q1+1;door<="00";

end if;

end if;

if g1='1' then led<="1001111"; --电梯到达1楼，数码管显示1

if d11='1' or c\_u11='1' then d11<='0'; c\_u11<='0';opendoor<='1';

--有当前层的请求，则电梯进入开门状态

elsif dd\_cc>"000001" then en\_up<='1'; opendoor<='0';

--有上升请求，则电梯进入预备上升状态

elsif dd\_cc="000000" then opendoor<='0'; --无请求时，电梯停在1楼待机

end if;

elsif g2='1' then led<="0010010"; --电梯到达2楼，数码管显示2

if updown='1' then --电梯前一运动状态位上升

if d22='1' or c\_u22='1' then d22<='0'; c\_u22<='0'; opendoor<='1';

--有当前层的请求，则电梯进入开门状态

elsif dd\_cc>"000011" then en\_up<='1'; opendoor<='0';

--有上升请求，则电梯进入预备上升状态

elsif dd\_cc<"000010" then en\_dw<='1'; opendoor<='0';

--有下降请求，则电梯进入预备下降状态

end if;

--电梯前一运动状态为下降

elsif d22='1' or c\_d22='1' then d22<='0'; c\_d22<='0';opendoor<='1';

--有当前层的请求，则电梯进入开门状态

elsif dd\_cc<"000010" then en\_dw<='1'; opendoor<='0';

--有下降请求，则电梯进入预备下降状态

elsif dd\_cc>"000011" then en\_up<='1'; opendoor<='0';

--有上升请求，则电梯进入预备上升状态

end if;

elsif g3='1' then led<="0000110"; --电梯到达3楼，数码管显示3

if updown='1' then

if d33='1' or c\_u33='1' then d33<='0'; c\_u33<='0';opendoor<='1';

elsif dd\_cc>"000111" then en\_up<='1'; opendoor<='0';

elsif dd\_cc<"000100" then en\_dw<='1'; opendoor<='0';

end if;

elsif d33='1' or c\_d33='1' then d33<='0'; c\_d33<='0'; opendoor<='1';

elsif dd\_cc<"000100" then en\_dw<='1'; opendoor<='0';

elsif dd\_cc>"000111" then en\_up<='1'; opendoor<='0';

end if;

elsif g4='1' then led<="1001100"; --电梯到达4楼，数码管显示4

if updown='1' then

if d44='1' or c\_u44='1' then d44<='0'; c\_u44<='0'; opendoor<='1';

elsif dd\_cc>"001111" then en\_up<='1'; opendoor<='0';

elsif dd\_cc<"001000" then en\_dw<='1'; opendoor<='0';

end if;

elsif d44='1' or c\_d44='1' then d44<='0'; c\_d44<='0'; opendoor<='1';

elsif dd\_cc<"001000" then en\_dw<='1'; opendoor<='0';

elsif dd\_cc>"001111" then en\_up<='1'; opendoor<='0';

end if;

elsif g5='1' then led<="0100100"; --电梯到达5楼，数码管显示5

if updown='1' then

if d55='1' or c\_u55='1' then d55<='0'; c\_u55<='0';opendoor<='1';

elsif dd\_cc>"011111" then en\_up<='1'; opendoor<='0';

elsif dd\_cc<"010000" then en\_dw<='1'; opendoor<='0';

end if;

elsif d55='1' or c\_d55='1' then d55<='0'; c\_d55<='0';opendoor<='1';

elsif dd\_cc<"010000" then en\_dw<='1'; opendoor<='0';

elsif dd\_cc>"011111" then en\_up<='1'; opendoor<='0';

end if;

elsif g6='1' then led<="0100000"; --电梯到达6楼，数码管显示6

if d66='1' or c\_d66='1' then d66<='0'; c\_d66<='0';opendoor<='1';

elsif dd\_cc<"100000" then en\_dw<='1'; opendoor<='0';

end if;

else en\_up<='0';en\_dw<='0'; --电梯进入上升或下降状态

end if;

end if;

else q<=1;alarm<='0'; --清除超载报警

if d1='1' then d11<=d1; --对电梯内人请求信号进行检测和寄存

elsif d2='1' then d22<=d2;

elsif d3='1' then d33<=d3;

elsif d4='1' then d44<=d4;

elsif d5='1' then d55<=d5;

elsif d6='1' then d66<=d6;

end if;

if c\_u1='1' then c\_u11<=c\_u1; --对电梯外人上升请求信号进行检测和寄存

elsif c\_u2='1' then c\_u22<=c\_u2;

elsif c\_u3='1' then c\_u33<=c\_u3;

elsif c\_u4='1' then c\_u44<=c\_u4;

elsif c\_u5='1' then c\_u55<=c\_u5;

end if;

if c\_d2='1' then c\_d22<=c\_d2; --对电梯外人下降请求信号进行检测和寄存

elsif c\_d3='1' then c\_d33<=c\_d3;

elsif c\_d4='1' then c\_d44<=c\_d4;

elsif c\_d5='1' then c\_d55<=c\_d5;

elsif c\_d6='1' then c\_d66<=c\_d6;

end if;

dd<=d66&d55&d44&d33&d22&d11; --电梯内人请求信号并置

cc\_u<='0'&c\_u55&c\_u44&c\_u33&c\_u22&c\_u11; --电梯外人上升请求信号并置

cc\_d<=c\_d66&c\_d55&c\_d44&c\_d33&c\_d22&'0'; --电梯外人下降请求信号并置

dd\_cc<=dd or cc\_u or cc\_d; --电梯内、外人请求信号进行综合

end if;

ud<=updown; --电梯运动状态显示

led\_d<=dd; --电梯内人请求信号显示

led\_c\_u<=cc\_u; --电梯外人上升请求信号显示

led\_c\_d<=cc\_d; --电梯外人下降请求信号显示

end if;

end process;

end behav;