全自动电梯控制电路

一、实验任务及要求：

设计一个四层楼房全自动电梯控制电路，功能如下：

1）每层电梯入口设有上下请示开关各一个（最低层只有向上请示，最高层只有向下请示开关），电梯内设有乘客到达层次的停站要求开关；

2）电梯所处位置指示装置和电梯上下行状态指示装置；

3）电梯每秒升（降）一层，到达某一层时，数码管显示该层层数，并一直保持到电梯到达新一层为止；

4）电梯到达有停站要求的梯层后，经过 0.5 秒，电梯门自动打开（开门指示灯亮），经过 5 秒后，电梯门自动关闭（开门指示灯灭），电梯继续运行；

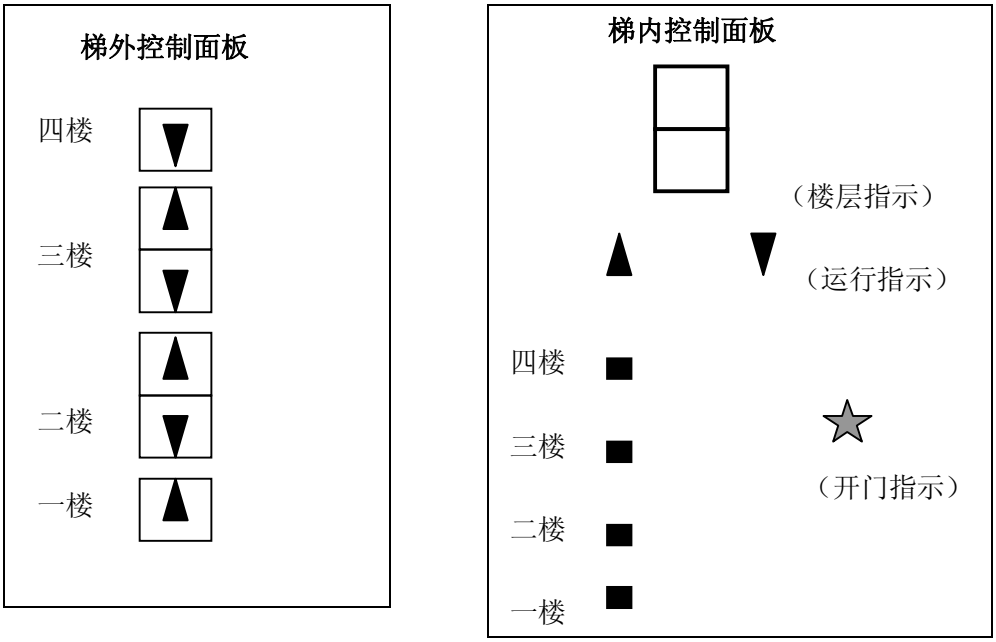
5）能保证响应电梯内外的所有请求信号，，并按照电梯运行规则次第响应，每个请求信号保留至执行后撤除；

6）电梯运行规则：电梯处于上升模式时，只响应比所在位置高的梯层的上楼请求信号，由上而下逐个执行直到最后一个请求执行完毕。如有更高层有下梯请求，则直接升到有下梯请求的最高层接客，然后转入下降模式。电梯处于下降模式时与之相反，仅响应比电梯所在位置低的下楼请求，由上到下逐个解决，直到最后一个请求被处理完毕。如果最低层有上楼请求时，则降至该层楼，并转入上升模式，电梯执行完所有的请求后，应停在最后所在位置不变，等待新的请求；

7）开机时，电梯应停在一楼，而各种上下请求均被清除。

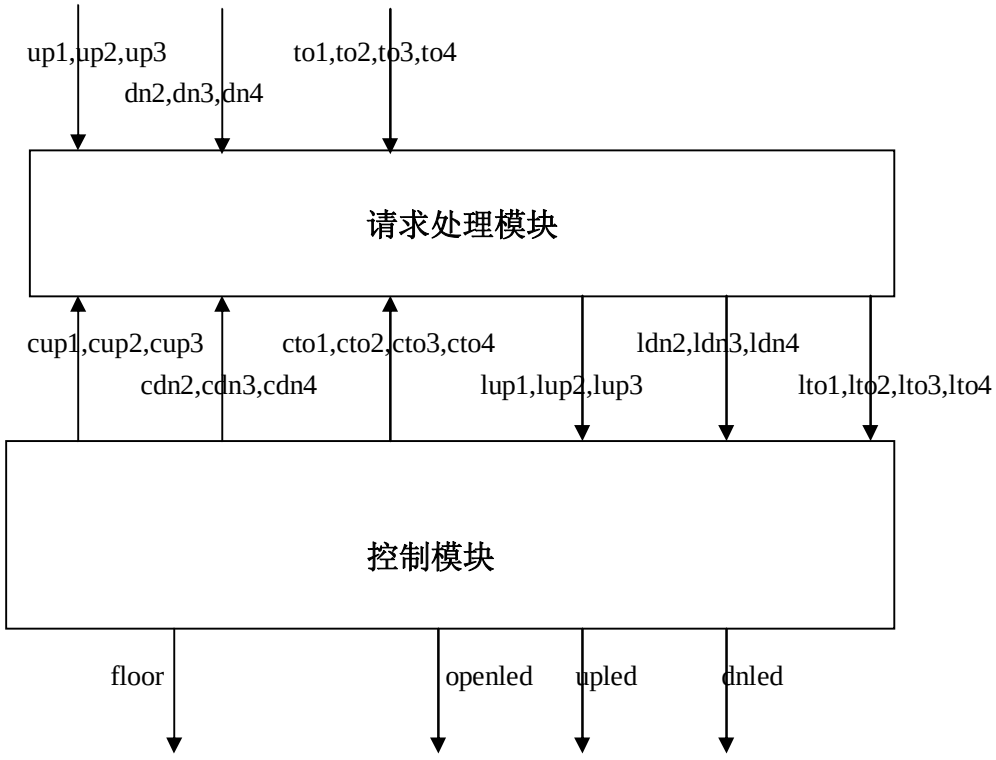
二、设计说明与提示

电路面板如下所示：



说明：以上面板除了表明“指示”的器件（即只是用于显示的器件，如开门指示）之外，其余的均为显示和输入器件，即比如“一楼 ■”表明其可作输入按键（作为到一楼的请求输入），同时亦作为显示器件（指示请求是否被处理）。

设计框图如下显示：



1．信号说明：

up1-up3：分别为 1，2，3 楼用户上楼请求信号；

dn2-dn4：分别为 2，3，4 楼用户下楼请求信号；

to1-to4：分别为电梯内用户到 1，2，3，4 楼的请求信号；

lup1-lup3：分别为 1，2，3 楼用户上楼请求指示；

ldn2-ldn4：分别为 2，3，4 楼用户下楼请求指示；

lto1-lto4：分别为电梯内用户到 1，2，3，4 楼的请求指示；

cup1-cup3：分别用于清除为 1，2，3 楼用户上楼请求；

cdn2-cdn4：分别用于清除为 2，3，4 楼用户下楼请求；

cto1-cto4：分别用于清除电梯内用户到 1，2，3，4 楼的请求；

floor：楼层显示；

openled：开门指示；

upled：上升指示；

dnled：下降指示；

2．模块说明：

请求处理模块：

存储用户的请求以及当请求被处理后请求指示的清除。

控制模块：

1） 电梯到达有停站要求的梯层后，经过 0.5 秒，电梯门自动打开（开门指示灯亮），经过 5 秒后，电梯门自动关闭（开门指示灯灭），电梯继续运行；

2） 能保证响应电梯内外的所有请求信号，，并按照电梯运行规则次第响应，每个请求信号保留至执行后撤除；

3） 开机时，电梯应停在一楼，而各种上下请求均被清除。

程序设计说明：

请求处理模块：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY key IS
    PORT(
        reset                : IN  STD_LOGIC;
        up1,up2,up3              : IN  STD_LOGIC;
        dn2,dn3,dn4              : IN  STD_LOGIC;
        to1,to2,to3,to4      : IN  STD_LOGIC;
        cup1,cup2,cup3           : IN  STD_LOGIC;
        cdn2,cdn3,cdn4           : IN  STD_LOGIC;
        cto1,cto2,cto3,cto4  : IN  STD_LOGIC;
        lup1,lup2,lup3       : OUT     STD_LOGIC;
        ldn2,ldn3,ldn4       : OUT     STD_LOGIC;
        lto1,lto2,lto3,lto4  : OUT     STD_LOGIC);
END key;
ARCHITECTURE a OF key IS
BEGIN
up1label:
PROCESS (reset,up1,cup1)
```

```
BEGIN
IF cup1='1' or reset='1' THEN      ——清零或复位
      lup1<='0';
ELSIF up1'event AND up1='1' THEN——保存请求信号
      lup1<='1';
END IF;
END PROCESS up1label;
                                        ——以下类似处理
up2label:
PROCESS (reset,up2,cup2)
BEGIN
IF cup2='1' or reset='1' THEN
      lup2<='0';
ELSIF up2'event AND up2='1' THEN
      lup2<='1';
END IF;
END PROCESS up2label;
up3label:
PROCESS (reset,up3,cup3)
BEGIN
IF cup3='1' or reset='1' THEN
      lup3<='0';
ELSIF up3'event AND up3='1' THEN
      lup3<='1';
END IF;
END PROCESS up3label;
dn2label:
PROCESS (reset,dn2,cdn2)
BEGIN
IF cdn2='1' or reset='1' THEN
      ldn2<='0';
ELSIF dn2'event AND dn2='1' THEN
      ldn2<='1';
END IF;
END PROCESS dn2label;
dn3label:
PROCESS (reset,dn3,cdn3)
BEGIN
IF cdn3='1' or reset='1' THEN
      ldn3<='0';
ELSIF dn3'event AND dn3='1' THEN
      ldn3<='1';
END IF;
END PROCESS dn3label;
```

```
dn4label:
PROCESS (reset,dn4,cdn4)
BEGIN
IF cdn4='1'    or reset='1' THEN
      ldn4<='0';
ELSIF dn4'event AND dn4='1' THEN
      ldn4<='1';
END IF;
END PROCESS dn4label;
to1label:
PROCESS (reset,to1,cto1)
BEGIN
IF cto1='1' or reset='1' THEN
      lto1<='0';
ELSIF to1'event AND to1='1' THEN
      lto1<='1';
END IF;
END PROCESS to1label;
to2label:
PROCESS (reset,to2,cto2)
BEGIN
IF cto2='1' or reset='1' THEN
      lto2<='0';
ELSIF to2'event AND to2='1' THEN
      lto2<='1';
END IF;
END PROCESS to2label;
to3label:
PROCESS (reset,to3,cto3)
BEGIN
IF cto3='1' or reset='1' THEN
      lto3<='0';
ELSIF to3'event AND to3='1' THEN
      lto3<='1';
END IF;
END PROCESS to3label;
to4label:
PROCESS (reset,to4,cto4)
BEGIN
IF cto4='1' or reset='1' THEN
      lto4<='0';
ELSIF to4'event AND to4='1' THEN
      lto4<='1';
END IF;
```

```
END PROCESS to4label;

END a;
```

控制模块：

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY fcontrol IS
    PORT(
        clk16,reset                    : IN  STD_LOGIC;
        lup1,lup2,lup3                 : IN  STD_LOGIC;
        ldn2,ldn3,ldn4                 : IN  STD_LOGIC;
        lto1,lto2,lto3,lto4            : IN  STD_LOGIC;
        openled,upled,dnled            : OUT     STD_LOGIC;
        floor                          : BUFFER      STD_LOGIC_VECTOR(3
DOWNTO 0);
        cup1,cup2,cup3                 : OUT     STD_LOGIC;
        cdn2,cdn3,cdn4                 : OUT     STD_LOGIC;
        cto1,cto2,cto3,cto4            : OUT     STD_LOGIC);
END fcontrol;
ARCHITECTURE a OF fcontrol IS
SIGNAL state,openjudge,directclose : STD_LOGIC;
SIGNAL    keyupfloor1,keyupfloor2,keyupfloor3,keyupfloor4    :    STD_LOGIC_VECTOR(3
          DOWNTO 0);
SIGNAL    keydnfloor1,keydnfloor2,keydnfloor3,keydnfloor4    :    STD_LOGIC_VECTOR(3
          DOWNTO 0);
SIGNAL up,dn,sport,upask,dnask:STD_LOGIC;
SIGNAL upfloor,dnfloor,dntofloor,uptofloor,directfloor:STD_LOGIC_VECTOR(3 DOWNTO 0);
SIGNAL countopen : STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL countsport : STD_LOGIC_VECTOR(5 DOWNTO 0);
SIGNAL temp : STD_LOGIC_VECTOR(4 DOWNTO 0);
SIGNAL statesport : STD_LOGIC_VECTOR(1 DOWNTO 0);
BEGIN
keyupfloor1<="0001" WHEN   lup1='1' OR lto1='1' ELSE "0000";
keyupfloor2<="0010" WHEN   lup2='1' OR lto2='1' ELSE "0000";
keyupfloor3<="0011" WHEN   lup3='1' OR lto3='1' ELSE "0000";
keyupfloor4<="0100" WHEN   lto4='1' ELSE "0000";
keydnfloor1<="0001" WHEN   lto1='1' ELSE "0000";
keydnfloor2<="0010" WHEN   ldn2='1' OR lto2='1' ELSE "0000";
keydnfloor3<="0011" WHEN   ldn3='1' OR lto3='1' ELSE "0000";
keydnfloor4<="0100" WHEN   ldn4='1' OR lto4='1' ELSE "0000";
——上升或下降模式中各层请求信号的信息存储
openjudge<='1' WHEN (state='1' AND (floor=keyupfloor1 OR floor=keyupfloor2 OR
floor=keyupfloor3 OR floor=keyupfloor4)) OR
```

```
                    (state='0' AND (floor=keydnfloor1 OR floor=keydnfloor2 OR
floor=keydnfloor3 OR floor=keydnfloor4)) ELSE '0';
    ——开门信号的判断，符合开门条件时输出为高电平
    upfloor<="0100" WHEN lto4='1' ELSE
            "0011" WHEN lup3='1' OR lto3='1' ELSE
            "0010" WHEN lup2='1' OR lto2='1' ELSE
            "0000";
    ——用于计载应响应的最高层的信息
    up<='1' WHEN floor<upfloor ELSE '0'; ——用于上升模式的判别
    dnfloor<="0001" WHEN lto1='1' ELSE
            "0010" WHEN ldn2='1' OR lto2='1' ELSE
            "0011" WHEN ldn3='1' OR lto3='1' ELSE
            "0100"  ;
    ——用于计载应响应的最低层的信息
    dn<='1' WHEN floor>dnfloor ELSE '0';——用于下降模式的判别
    uptofloor<="0100" WHEN ldn4='1' AND dn='0' AND state='0' AND floor<4 ELSE
          "0011" WHEN ldn3='1' AND dn='0' AND state='0' AND floor<3 ELSE
          "0010" WHEN ldn2='1' AND dn='0' AND state='0' AND floor<2 ELSE
          "0000";——state=0，下降模式；1，上升模式
    ——切换到上升模式时应响应的楼层（优先权由高到低）
    dntofloor<="0001" WHEN lup1='1' AND up='0' AND state='1' AND floor>1 ELSE
          "0010" WHEN lup2='1' AND up='0' AND state='1' AND floor>2 ELSE
          "0011" WHEN lup3='1' AND up='0' AND state='1' AND floor>3 ELSE
          "0000";
    ——切换到下降模式时应响应的楼层（优先权由低到高）
    sport<='1' WHEN (state='1' AND upask='1') OR (state='0' AND dnask='1' )
              ELSE '0';
    ——运动状态判别：处于有效的响应运动中
    upask<='1' WHEN   up='1' OR (NOT (uptofloor="0000")) ELSE '0';
     ——上升请求：继续向上运动或者由下降模式切换到上升模式
    dnask<='1' WHEN   dn='1' OR (NOT (dntofloor="0000" )) ELSE '0';   ——下降请求
    ——下降请求：继续向下运动或者由上升模式切换到下降模式
    openled<='1' WHEN statesport="01" OR (statesport="10" AND countsport=0) ELSE '0';
    ——开门指示灯
    upled<='1' WHEN state='1' ELSE '0';
    ——上升指示
    dnled<='1' WHEN state='0' ELSE '0';
    ——下降指示
    temp<=floor & state;
    updnlabel:
    PROCESS (reset,clk16)
    BEGIN
    IF reset='1' THEN            ——设置初始状态
        floor<="0001";
```

statesport<="00"; ——statesport=00 开门前 0.5 秒的等待状态；01 开门持续 5 秒的开门状态；11 运行状态。

countopen<="00000000";—用于对等待状态和开门状态的计数，以控制状态的变换

state<='1'; ——state=1 上升状态,0 下降状态。

directfloor<="0000"; ——直达楼层信号

directclose<='0'; ——直达运行状态信号

ELSIF clk16'event and clk16='1' THEN ——以 1/16 的频率判别状态的转换

IF statesport(1)='0' then countopen<=countopen+1; ELSE countopen<="00000000"; END IF;——等待和开门状态的计数

IF (NOT (uptofloor="0000")) then directfloor<=uptofloor-floor;directclose<='1';

ELSIF (NOT (dntofloor="0000")) THEN directfloor<=floor-dntofloor;directclose<='1';

ELSIF countsport=15 AND directclose='1' THEN directfloor<=directfloor-1; END IF;

——直达楼层信号的锁定，及每一秒变化一层

IF (statesport(1)='1' AND sport='1') or (NOT(directfloor="0000")) then countsport<=countsport+1; ELSE countsport<="000000"; END IF;

——运行模式时运行状态的计数

IF countopen=8 then statesport<="01"; ELSIF countopen=88 THEN statesport<="10"; END IF;

——由等待模式切换到开门模式，以及由开门模式切换到运行模式

IF countsport=1 AND state='1' THEN floor<=floor+1; ELSIF countsport=1 AND state='0' THEN floor<=floor-1; END IF;

——将要到达楼层的指示

IF (countsport=16 AND openjudge='1' AND directclose='0')OR(directclose='1' AND directfloor="0000") THEN statesport<="00";directclose<='0';

——到达某层且要开门时，转入等待状态；以及直达某层后直达信号的关闭

ELSIF (countsport=16 AND (NOT(directfloor="0000"))) OR (countsport=16 AND openjudge='0' AND directclose='0') THEN countsport<="000000"; END IF;

——不开门继续运行时，转入另一个运行状态

IF state='1' AND sport='0' AND dnask='1' THEN state<='0'; ELSIF state='0' AND sport='0' AND upask='1' THEN state<='1'; END IF;

——运行状态的切换

IF countopen=8 THEN ——到达请求层时，清除响应的请求信号（开门状态时发出清零脉冲）

```
CASE temp IS
    WHEN "00010" =>
        cto1<='1';
    WHEN "00100" =>  ——下降到达第 2 层
        cdn2<='1';        ——清除第二层的下降请求
        cto2<='1';         ——清除到第二层的请求
    WHEN "00110" =>   ——其它类似处理
        cdn3<='1';
        cto3<='1';
    WHEN "01000" =>
        cdn4<='1';
```

```
                cto4<='1';
            WHEN "00011" =>
                cup1<='1';
                cto1<='1';
            WHEN "00101" =>
                cup2<='1';
                cto2<='1';
            WHEN "00111" =>
                cup3<='1';
                cto3<='1';
            WHEN "01001" =>
                cto4<='1';
            WHEN OTHERS =>
                NULL;
        END CASE;
        ELSE
                cup1<='0';
                cup2<='0';
                cup3<='0';
                cdn4<='0';
                cdn3<='0';
                cdn2<='0';
                cto4<='0';
                cto3<='0';
                cto2<='0';
                cto1<='0';
        END IF;
    END IF;
    END PROCESS updnlabel;
    END a;
```

三、实验报告要求

1.理解原理，并画出状态切换流图。

2.对照实验要求分析电路工作原理。

3.写出各功能模块的 VHDL 语言源文件。

4.叙述各模块的工作原理。

5. 祥述控制器的工作原理，绘出完整的电路或写出 VHDL 源文件。

6. 书写实验报告时应结构合理，层次分明，在分析时注意语言的流畅。