实验八　4×4 矩阵键盘

一、　实验目的

1. 熟悉并掌握 4×4 矩阵键盘驱动电路设计方法；
2. 熟悉并掌握硬件资源分时复用设计方法。

二、　实验内容

本实验系统 4×4 矩阵键盘电路示意图如图 2-8-1 所示。PORT A 端口只有最低的 8 位有效，高 4 位为键盘的 4 位列扫描输出（低电平有效），对应 Y1~Y4，低 4 位为键盘的 4 位行查询输入（低电平有效），对应 X1~X4。PORT A 的 D7~D4 对应第 1 列至第 4 列扫描输出（最左边为第 1 列），D3~D0 对应第 1 行至第 4 行查询输入（最上面为第 1 行）。例如，当 D7~D4 输出 0111 时，扫描到键盘第 1 列，若此时 D3~D0 返回 0111，说明第 1 列的第 1 行按下，即按键“1”按下；若返回 1011，说明第 1 列的第 2 行按下，即按键“4”按下，若返回 1111，说明此列没有任何键按下，其他按键的识别以此类推。

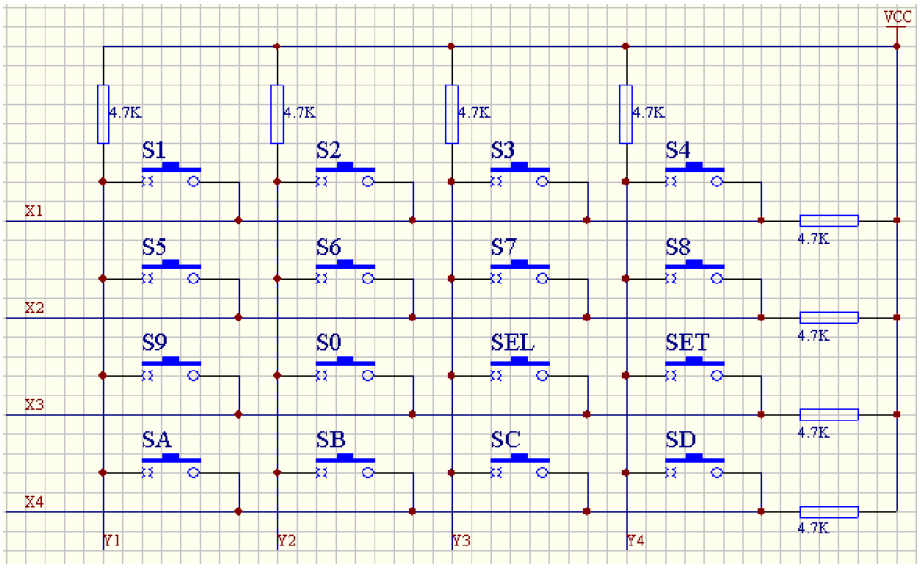


图 2-8-1　4×4 矩阵键盘硬件电路示意图

表 2-8-1 给出了键盘编码的情况。为了区分不同的按键，对每个键进行了二进制编码。16 个键采用了 4 位二进制。其中，0~9 编码为“0000”~“1001”，A~D 编码为“1010”~“1101”，“*”编码为“1110”，“#”编码为“1111”。

表 2-8-1　键盘编码

Y1~Y4	X1~X4	对应的按键	键盘编码输出
1110	**1110**	D	1101
	1101	C	1100
	1011	B	1011
	0111	A	1010

1101	**1110**	#	1111
	1101	9	1001
	1011	6	0110
	0111	3	0011
1011	**1110**	0	0000
	1101	8	1000
	1011	5	0101
	0111	2	0010
0111	**1110**	*	1110
	1101	7	0111
	1011	4	0100
	0111	1	0001

实验箱上的 16 个拨码开关，16 个按键，12 个交通灯和蜂鸣器四周 4 个灯，8 位数码管，4x4 矩阵键盘，16x16 点阵 LED 均是从底板的两片 CPLD 引出的，这些资源有 IO 方式和总线操作两种控制方式。这种资源共享的方式减少了实验过程中连线的麻烦，但也带来一个问题：资源不能同时使用。例如，使用了数码管，如果想同时进行点阵显示就不行，除非改变 M[3..0]的值。

我们提出一个类似数码管动态扫描显示的解决方案。按一定的节拍依次改变 M[3..0]的值，同时将不同资源的数据依次与 PORT A 的 D15~D0、以及 A4~A1 相连。因此，任意时刻其实只有一种资源占用 PORT A 总线资源，但当循环扫描时可以达到不同资源对总线分时复用的效果。

实验要求：

实现 4×4 矩阵键盘的驱动电路设计，要求按下某键时，该键对应的编码（表 2-8-1）在数码管上显示出来。

提示：

- 建议采用原理图与文本相结合的混合层次化设计方法。4×4 矩阵的识别编码程序（keyboard4_4.vhd）如下所示。程序对输入时钟（1KHz）分频后得到键盘扫描时钟，按此节拍实现了一个模 4 计数器，并依次产生列扫描信号，根据列扫描信号和返回的行值判断当前的按键情况，并进行编码。该程序生成原理图符号如图 2-8-2 所示。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity keyboard4_4 is
    port(
        rst      :    in     std_logic;
        clk_scan :    in     std_logic;
        keyin    :    in     std_logic_vector(3 downto 0);
        scan     :    out    std_logic_vector(3 downto 0);
        leds     :    out    std_logic_vector(3 downto 0));
```

```
end;
architecture keyboard4_4_arch of keyboard4_4 is
 signal scnlin       :       std_logic_vector(3 downto 0);
 signal clkfrq       :       std_logic;
 signal cntfrq       :       std_logic_vector(4 downto 0);
 signal lednum       :       std_logic_vector(7 downto 0);
 signal cntscn       :       std_logic_vector(1 downto 0);
begin
 scan <= not scnlin;
 lednum <= scnlin & (not keyin);
 process(rst,clk_scan)    --去抖时钟,50 分频,形成扫描时钟
  begin
       if rst = '0' then
            clkfrq <= '0';
            cntfrq <= (others => '0');
       elsif rising_edge(clk_scan) then
            if cntfrq = "11000" then
                 cntfrq <= (others => '0');
                 clkfrq <= not clkfrq;
            else
                 cntfrq <= cntfrq + 1;
            end if;
       end if;
  end process;
process(rst,clkfrq)       --  根据扫描时钟产生扫描线
      begin
           if rst = '0' then
                cntscn <= "00";
           elsif rising_edge(clkfrq) then
                if cntscn = "11" then
                     cntscn <= "00";
                else
                     cntscn <= cntscn+1;
                end if;
                case cntscn is
                     when "00" => scnlin <= "0001";
                     when "01" => scnlin <= "0010";
                     when "10" => scnlin <= "0100";
                     when "11" => scnlin <= "1000";
                     when others => null;
                end case;
           end if;
      end process;
      process(rst, clkfrq)     --根据按键点亮相应的 leds
```

```
begin
    if(rst = '0' ) then
        leds <= "0000";
    elsif clkfrq'event and clkfrq = '0' then
        case lednum is
            when "10001000" =>leds <= "0001";  --1
            when "01001000" =>leds <= "0010";  --2
            when "00101000" =>leds <= "0011";  --3
            when "00011000" =>leds <= "1010";   --A
            when "10000100" =>leds <= "0100";  --4
            when "01000100" =>leds <= "0101"; --5
            when "00100100" =>leds <= "0110";  --6
            when "00010100" =>leds <= "1011";  --B
            when "10000010" =>leds <= "0111";  --7
            when "01000010" =>leds <= "1000";  --8
            when "00100010" =>leds <= "1001";  --9
            when "00010010" =>leds <= "1100";  --C
            when "10000001" =>leds <= "1110";  --*
            when "01000001" =>leds <= "0000";  --0
            when "00100001" =>leds <= "1111"; --#
            when "00010001" =>leds <= "1101"; --D
            when others =>null;
        end case;
    end if;
end process;
end;
```

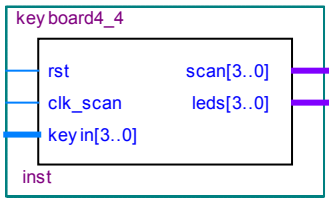


图 2-8-2　4×4 键盘识别编码电路

■ 为了达到更好的按键效果，需要对行返回值 keyin[3..0]进行消抖（如图 2-8-3）。其中，key_dounce4 模块为 4 位按键消抖电路，其底层电路如图 2-8-4 所示。

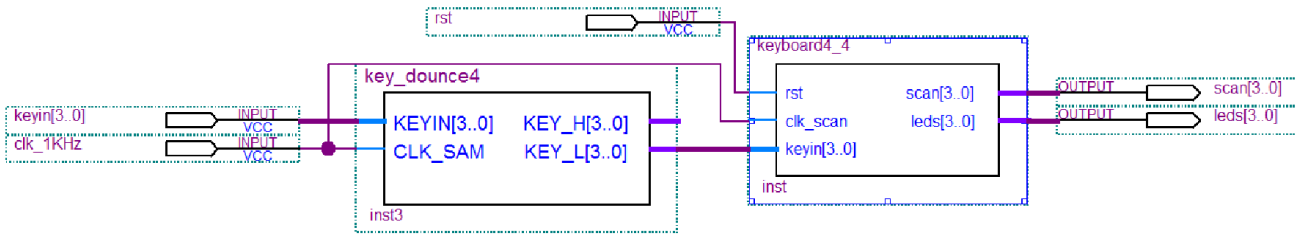
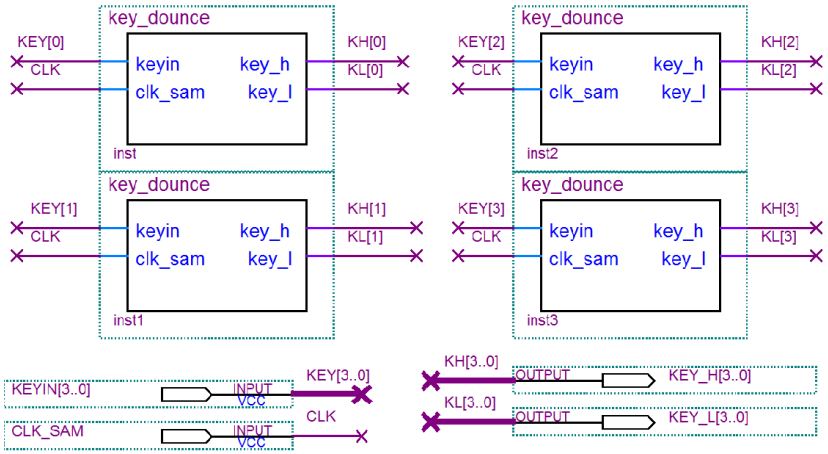


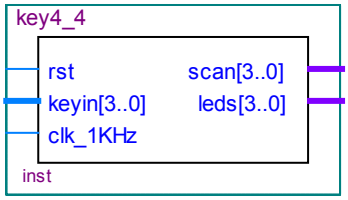
图 2-8-3　带按键消抖的 4×4 矩阵键盘驱动电路

图 2-8-4 中的 key_dounce 模块为 1 位按键消抖，其底层电路（key_dounce.bdf）如图 2-5-3 所示。



2-8-4 4 位按键消抖电路

■ 将图 2-8-3 带按键消抖的 4×4 矩阵键盘驱动电路生成原理图符号（如图 2-8-5）。



2-8-5 顶层原理图

■ 顶层原理图如图 2-8-6 所示。其中，SCAN_LED 模块实现 8 位数码管的显示驱动，将矩阵键盘的二进制编码译成数码管显示的七段数据，其底层 VHDL 语言程序（SCAN_LED.vhd）可参见实验六相关内容。data_bus 模块实现不同资源对 PORT A 端口总线的分时复用，其 VHDL 程序（data_bus.vhd）如下。该程序设计了模 6 计数器，依次产生 6 种 M[3..0]值，并根据相应 M 值，完成不同资源与 PORT A 端口的连接。

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY data_bus IS
PORT (CLK : IN STD_LOGIC;
      SW : OUT STD_LOGIC_VECTOR(16 DOWNTO 1);----M[3..0]=0001
      PB : OUT STD_LOGIC_VECTOR(16 DOWNTO 1);----M[3..0]=0011
      LED : IN STD_LOGIC_VECTOR(16 DOWNTO 1);----M[3..0]=0111
      SEG : IN STD_LOGIC_VECTOR(7 DOWNTO 0);----M[3..0]=0010
      COM : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
      ROW : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);----M[3..0]=0101
      COL : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
```

```
        DOT : IN STD_LOGIC_VECTOR(15 DOWNTO 0);----M[3..0]=0110
        ADDRESS : IN STD_LOGIC_VECTOR(4 DOWNTO 1);
        ---------------------------------------------
        Mout : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);----M[3..0]
        D : INOUT STD_LOGIC_VECTOR(15 DOWNTO 0);----PORT A
        A : OUT STD_LOGIC_VECTOR(4 DOWNTO 1));
END;
ARCHITECTURE behav OF data_bus IS
 SIGNAL Mtemp : STD_LOGIC_VECTOR(3 DOWNTO 0);
 SIGNAL CQI : STD_LOGIC_VECTOR(2 DOWNTO 0);
 SIGNAL SWtemp,PBtemp : STD_LOGIC_VECTOR(16 DOWNTO 1);
 SIGNAL ROWtemp : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
 Mout <= Mtemp;
 PROCESS(CLK)
BEGIN
 IF CLK'EVENT AND CLK='1' THEN
  IF CQI < 5 THEN CQI <= CQI + 1;
  ELSE CQI <= "000";
 END IF;
 END IF;
END PROCESS;
PROCESS(CQI)
BEGIN
 CASE CQI IS
 WHEN "000" => Mtemp<="0001";
 WHEN "001" => Mtemp<="0011";
 WHEN "010" => Mtemp<="0111";
 WHEN "011" => Mtemp<="0010";
 WHEN "100" => Mtemp<="0101";
 WHEN "101" => Mtemp<="0110";
 WHEN OTHERS => NULL;
 END CASE;
END PROCESS;
PROCESS(CLK,Mtemp,LED,SEG,COM,COL,DOT,ADDRESS,D)
 BEGIN
  CASE Mtemp IS
  WHEN "0001" => SWtemp(16 DOWNTO 1)<=D(15 DOWNTO 0);D<=(OTHERS=>'Z');
  WHEN "0011" => PBtemp(16 DOWNTO 1)<=D(15 DOWNTO 0);D<=(OTHERS=>'Z');
  WHEN "0111" => D(15 DOWNTO 0)<=LED(16 DOWNTO 1);
  WHEN "0010" => D(15 DOWNTO 8) <= COM; D(7 DOWNTO 0) <=SEG;
  WHEN "0101" => D(7 DOWNTO 4)<=COL;D(15 DOWNTO 8)<=(OTHERS=>'Z');
                 D(3 DOWNTO 0)<=(OTHERS=>'Z');ROWtemp<=D(3 DOWNTO 0);
  WHEN "0110" => D <= DOT;A<=ADDRESS;
```

```
    WHEN OTHERS => D<=(OTHERS=>'Z');
    END CASE;
    IF CLK'EVENT AND CLK='0' THEN
        SW <=SWtemp;
        PB <=PBtemp;
        ROW <=ROWtemp;
    END IF;
  END PROCESS;
END;
```

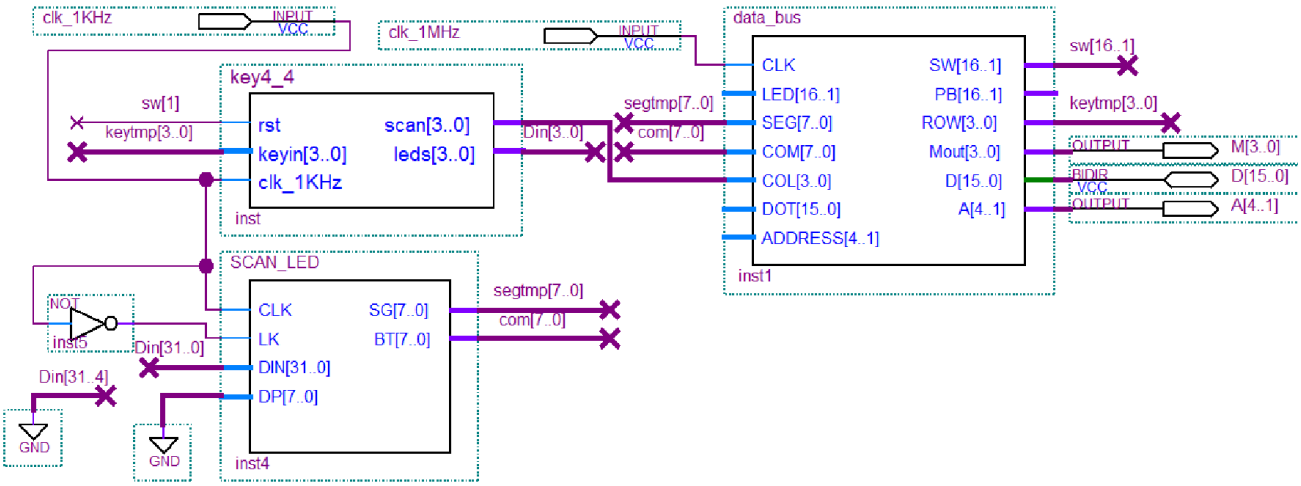


图 2-8-6　顶层原理图

★特别注意：由于在 data_bus 模块生成了 M[3..0]的值，因此无需再特别设置。但还需要正确分配引脚号，对**未分配的管脚设置为三态输入，同时要对双功能引脚进行合理设置（如图 2-8-7）。否则，对硬件有损害，切记！！！**

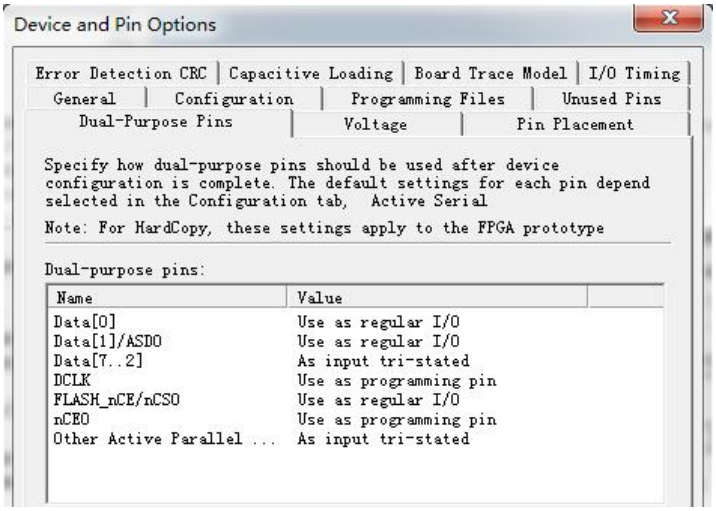


图 2-8-7　双功能引脚设置

引脚分配建议：

- 参照**《EL-SOPC4000 实验系统简介》表 2** 设置 M [3..0]引脚分配如下：
- 将 clk_1KHz 分配到 PORT B 的 GPIO1（参照**《EL-SOPC4000 实验系统简介》表 2**），从实验箱 IO1 引出，通过导线连接到用户时钟单元 ADJ_CLK 孔，根据**《EL-SOPC4000 实验系统简介》表 5** 拨动 SW17~SW20，将时钟输入设置成 1KHz；
- 将 clk_1MHz 分配到 PORT B 的 GPIO2，从实验箱 IO2 引出，通过导线连接到用户时钟单元 CLK2，；
- 将 D [15..0]分配到 PORT A 的 D15~D0，将 A[4..1]分配到 PORT A 的 A4~A1。

PORT B	FPGA 管脚	信号	PORT B	FPGA 管脚	信号
PIN_41	PIN_P11	M[0]	PIN_42	PIN_U11	M[1]
PIN_43	PIN_R11	M[2]	PIN_44	PIN_N11	M[3]

硬件测试时按下矩阵键盘，观察数码管的显示结果，验证矩阵键盘驱动电路的正确性，并记录下来。

三、 实验报告

根据以上的实验内容写出实验报告，包括仿真分析、硬件测试结果和详细实验过程，解释矩阵键盘驱动电路及总线分时复用电路的工作原理。

思考：分时复用电路是否能够实现拨码开关、按键、12 个交通灯和蜂鸣器四周 4 个灯、8 位数码管、4x4 矩阵键盘以及 16×16 点阵 LED 全部资源的合理使用？为什么？