深 圳 大 学 实 验 报 告

课程名称：硬件描述语言及数字系统设计

实验项目名称：实验四 矩阵键盘扫描

学院：医学院

专业：生物医学工程

指导教师：但果、董磊

报告人：陈焕鑫 学号：**2016222042** 班级：生工 2 班

实验时间：**2018.11.14**

实验报告提交时间：**2018.11.15**

教务部制

一、实验平台：

安装了 ISE 软件的 PC 计算机

硬件描述语言及数字系统设计实验平台

JTAG 下载线

二、实验目的：

当你完成整个项目之后，你将会学会一下内容：

- 掌握矩阵键盘的扫描原理
- 了解键盘防抖的原理
- 将矩阵键盘的值显示在数码管上

三、实验内容：

用 VHDL 代码描述一个矩阵键盘扫描程序，并编写测试激励进行测试

按照按键防抖电路图，分析按键防抖的原理，并编写 VHDL 代码

将矩阵键盘的按键结果（带防抖程序）显示到七段数码管上

编写引脚约束文件，用 ISE 软件生成.bit 文件，下载到硬件描述语言及数字系统设计实验平台进行板级验证

四、实验原理：

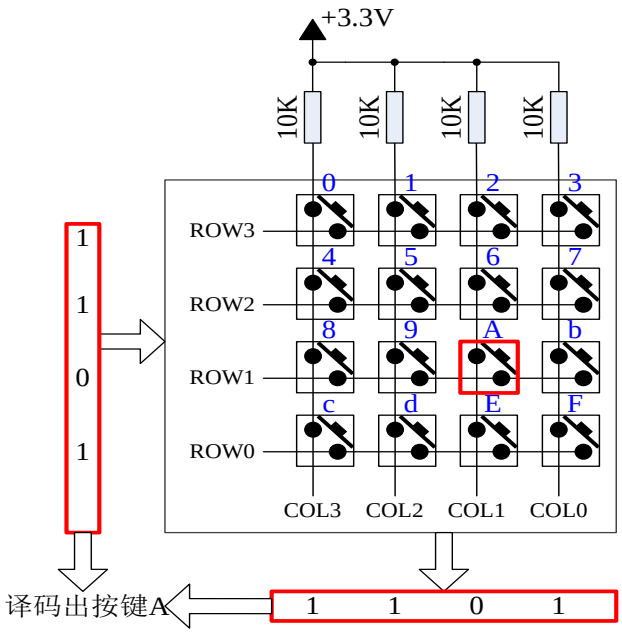

图 4-1 4*4 矩阵键盘电路图

所谓键盘扫描，就是在键盘的行（或列）依次送入扫描码，以便定位被按下的键，键盘编码是对键盘扫描值译码得到按下键的按键值。ROW3-ROW0 依次按照“0111”，“1011”，“1101”到“1110”的顺序进行循环，表示依次扫描第一行、第二行、第三行和第四行。同时，记录扫描回来的值 COL3-COL0，然后将 ROW3-ROW0 和 COL3-COL0

进行组合译码，就可以得出每个键盘的扫描值。比如，按下 0 号键，那么在扫描第一行（也就是 ROW3-ROW0 为“1000”）的时候 COL3 就为“0”，COL1-COL3 为“111”，如下图所示，这样（ROW3-ROW0，COL3-COL0）就是“01110111”，译码后就是“0”。

图 4-2 所示的是实验矩阵键盘扫描的原理图，虚框内是 FPGA 内部电路图，在这里要特别注意 FPGA 内部电路图中，s_row 信号和 s_row_reg 信号之间有个寄存器，这个寄存器是为了和消抖电路产生的延迟保持一致。

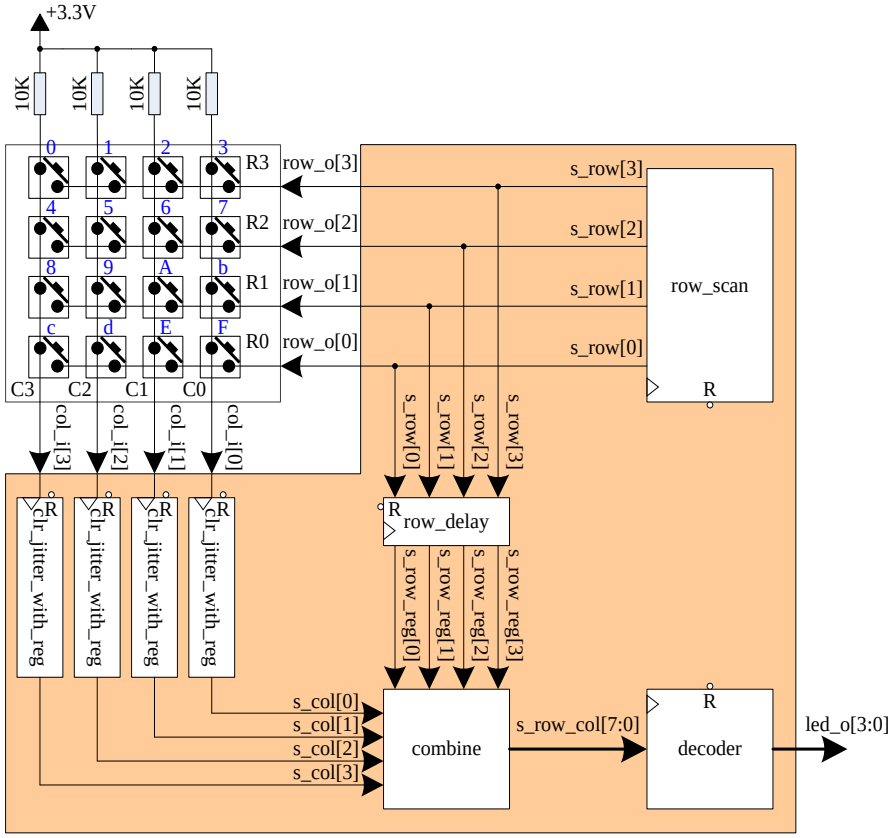

图 4-2 实验矩阵键盘原理图

现在电子元器件市场上有的按键，绝大多数都是机械式的开关结构，而机械式开关的核心部件为弹性金属簧片，因而在开关切换的瞬间会在接触点出现来回弹跳的现象。虽然只是进行了一次按键，结果在按键信号稳定的前后出现了多个脉冲，如图 2-32 所示的原始波形。对于原始波形，画一条中轴线，将中轴线以上的信号视为高电平，将中轴线以下的信号视为低电平，这样就出现了如图 2-32 所示的理想化波形。如果将这样的信号直接送给微处理器扫描采集的话，就可能把按键稳定前后出现的脉冲信号当作按键信号，这就出现人为的一次按键但微处理器以为多次按键现象。为了确保按键识别的准确性，在按键信号抖动的情况下不能进入状态输入，为此就必须对按键进行消抖处理，消除抖动时不稳定、随机的电压信号。机械式按键的抖动次数、抖动时间、抖动波形都是随机的。不同类型的按键其最长抖动时间也有差别，抖动时间的长短和按键的机械特性有关，一般为 5~10ms，而一般人按键的时间大于 100ms。

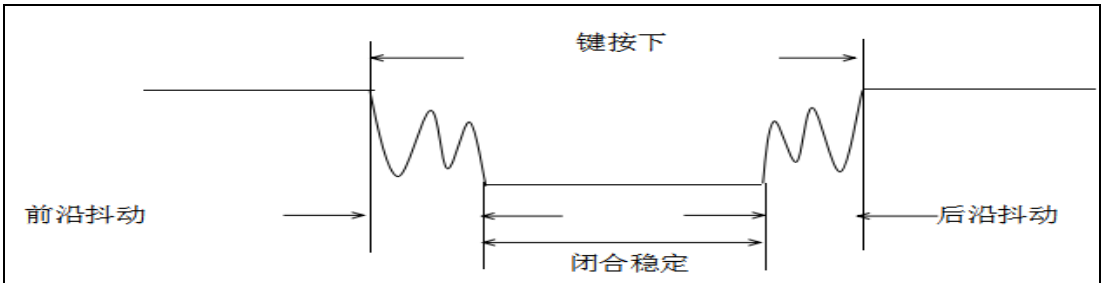

图 4-3 按键按下的波形

消抖原理正是基于这两个时间参数的差异，取一个中间值（例如 50ms）作为界限，周期小于 50ms 的视为抖动脉冲，而周期大于 50ms 的视为按键信号。

五、实验方法步骤及 VHDL 代码：

根据分频原理进行代码设计，设计一个产生 50Hz 的时钟信号。

具体的 VHDL 代码如下所示：

```
--------------------------------------------------------------------------------
--模块名称：clk_gen_50hz
--摘要提示：
--当前版本：1.0.0
--模块作者：
--完成日期：20XX 年 XX 月 XX 日
--内容提要：
--需要注意：
--------------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity clk_gen_50hz is
    generic(
    CNT_MAX : integer := 1000000;
    CNT_HALF : integer := 500000
   );

   Port ( clk_i : in STD_LOGIC;
```

```
          rst_n_i : in  STD_LOGIC;
          clk_o : out  STD_LOGIC
   );
end clk_gen_50hz;

architecture rtl of clk_gen_50hz is
  signal s_cnt : integer range 0 to CNT_MAX := 0;
begin
  process(clk_i, rst_n_i, s_cnt)
  begin
    if(rst_n_i = '0')then
      s_cnt <= 0;
      clk_o <= '0';
    elsif rising_edge(clk_i)then
      if(s_cnt = CNT_MAX)then
        s_cnt <= 0;
        clk_o <= '0';
      elsif(s_cnt = CNT_HALF)then
        s_cnt <= s_cnt + 1;
        clk_o <= '1';
      else
        s_cnt <= s_cnt + 1;
      end if;
    end if;
  end process;

end rtl;
```

行扫描模块，每接收到一次时钟信号就转换一次输出，输出在“1110”到“0111”之间反复循环。

```
--------------------------------------------------------------------------------
--模块名称：row_scan
--摘要提示：
--当前版本：1.0.0
--模块作者：
--完成日期：20XX 年 XX 月 XX 日
--内容提要：
--需要注意：
--------------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
```

```
--修改文件:
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity row_scan is
    Port ( clk_i : in  STD_LOGIC;
           rst_n_i : in  STD_LOGIC;
           row_o : out  STD_LOGIC_VECTOR (3 downto 0)
   );
end row_scan;

architecture rtl of row_scan is

  component clk_gen_50hz is
    port(
      clk_i : in  STD_LOGIC;
         rst_n_i : in  STD_LOGIC;
         clk_o : out  STD_LOGIC
    );
  end component;

  signal s_clk_50hz : std_logic;
  signal s_row      : std_logic_vector(3 downto 0);

begin

  i_clk_gen_50hz : clk_gen_50hz
  port map(
    clk_i   => clk_i,
    rst_n_i => rst_n_i,
    clk_o   => s_clk_50hz
  );

  process(s_clk_50hz, rst_n_i, s_row)
  begin
    if(rst_n_i = '0')then
      s_row <= "1110";
    elsif rising_edge(s_clk_50hz)then
      s_row <= s_row(0) & s_row(3 downto 1);
    end if;
  end process;

  row_o <= s_row;
```

```
end rtl;
```

延迟模块，因为列输入与行扫描之间存在一个周期的延迟，所以需要人为地把行状态延迟一个周期，才能和列的状态匹配。

```
----------------------------------------------------------------------------
--模块名称：row_delay
--摘要提示：
--当前版本：1.0.0
--模块作者：
--完成日期：20XX 年 XX 月 XX 日
--内容提要：
--需要注意：
----------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity row_delay is
    Port ( clk_i : in  STD_LOGIC;
           rst_n_i : in  STD_LOGIC;
           row_i : in  STD_LOGIC_VECTOR (3 downto 0);
           row_reg_o : out  STD_LOGIC_VECTOR (3 downto 0)
   );
end row_delay;

architecture rtl of row_delay is

  component clk_gen_50hz is
    port(
      clk_i : in  STD_LOGIC;
         rst_n_i : in  STD_LOGIC;
         clk_o : out  STD_LOGIC
    );
  end component;

  signal s_clk_50hz : std_logic;
```

```
begin
  i_clk_gen_50hz : clk_gen_50hz
  port map(
    clk_i   => clk_i,
    rst_n_i => rst_n_i,
    clk_o   => s_clk_50hz
  );

  process(s_clk_50hz, rst_n_i, row_i)
  begin
    if(rst_n_i = '0')then
      row_reg_o <= "0000";
    elsif rising_edge(s_clk_50hz)then
      row_reg_o <= row_i;
    end if;
  end process;

end rtl;
```

将行和列采集到的状态拼接成八位，然后输出到译码器中进行译码。

```
----------------------------------------------------------------------------
--模块名称：combine
--摘要提示：
--当前版本：1.0.0
--模块作者：
--完成日期：20XX年XX月XX日
--内容提要：
--需要注意：
----------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
----------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity combine is
    Port ( row_i : in  STD_LOGIC_VECTOR (3 downto 0);
           col_i : in  STD_LOGIC_VECTOR (3 downto 0);
           row_col_o : out  STD_LOGIC_VECTOR (7 downto 0)
   );
```

```
end combine;

architecture rtl of combine is

begin
  row_col_o <= row_i & col_i;

end rtl;
```

按键去抖模块，利用两个寄存器，消除按键的抖动。

```
--------------------------------------------------------------------------------
--模块名称：clr_jitter_with_reg
--摘要提示：
--当前版本：1.0.0
--模块作者：
--完成日期：20XX 年 XX 月 XX 日
--内容提要：
--需要注意：
--------------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity clr_jitter_with_reg is
    Port ( clk_50mhz_i : in  STD_LOGIC;
           rst_n_i : in  STD_LOGIC;
           button_i : in  STD_LOGIC;
           button_o : out  STD_LOGIC
    );
end clr_jitter_with_reg;

architecture rtl of clr_jitter_with_reg is

  component clk_gen_50hz is
    port(
      clk_i : in  STD_LOGIC;
        rst_n_i : in  STD_LOGIC;
        clk_o : out  STD_LOGIC
```

```
    );
  end component;

  signal s_clk_50hz : std_logic;
  signal x          : std_logic;
  signal y          : std_logic;
begin
  i_clk_gen_50hz : clk_gen_50hz
  port map(
    clk_i   => clk_50mhz_i,
    rst_n_i => rst_n_i,
    clk_o   => s_clk_50hz
  );

  process(s_clk_50hz, rst_n_i, button_i)
  begin
    if(rst_n_i = '0')then
      x <= '0';
    elsif rising_edge(s_clk_50hz)then
      x <= button_i;
    end if;
  end process;

  process(s_clk_50hz, rst_n_i, x)
  begin
    if(rst_n_i = '0')then
      y <= '0';
    elsif rising_edge(s_clk_50hz)then
      y <= x;
    end if;
  end process;

  button_o <= not((not x) and y);

end rtl;
```

对各个模块进行整合，组成成矩阵键盘。

```
--------------------------------------------------------------------------------
--模块名称：Key4x4
--摘要提示：
--当前版本：1.0.0
--模块作者：
--完成日期：20XX 年 XX 月 XX 日
```

```
--内容提要:
--需要注意:
--------------------------------------------------------------------------------
--取代版本:
--模块作者:
--完成日期:
--修改内容:
--修改文件:
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity Key4x4 is
    Port ( clk_50mhz_i : in  STD_LOGIC;
           rst_n_i : in  STD_LOGIC;
           col_i   : in STD_LOGIC_VECTOR(3 downto 0);
           row_o   : out STD_LOGIC_VECTOR(3 downto 0);
           led_o : out  STD_LOGIC_VECTOR (3 downto 0)
   );
end Key4x4;

architecture rtl of Key4x4 is

  component clr_jitter_with_reg is
    port(
      clk_50mhz_i : in  STD_LOGIC;
      rst_n_i : in  STD_LOGIC;
      button_i : in  STD_LOGIC;
      button_o : out  STD_LOGIC
    );
  end component;

  component row_scan is
    port(
      clk_i : in  STD_LOGIC;
      rst_n_i : in  STD_LOGIC;
      row_o : out  STD_LOGIC_VECTOR (3 downto 0)
    );
  end component;

  component row_delay is
    port(
```

```
      clk_i : in  STD_LOGIC;
        rst_n_i : in  STD_LOGIC;
        row_i : in  STD_LOGIC_VECTOR (3 downto 0);
        row_reg_o : out  STD_LOGIC_VECTOR (3 downto 0)
    );
  end component;

  component combine is
    port(
      row_i : in  STD_LOGIC_VECTOR (3 downto 0);
        col_i : in  STD_LOGIC_VECTOR (3 downto 0);
        row_col_o : out  STD_LOGIC_VECTOR (7 downto 0)
    );
  end component;

  component decoder is
    port(
      clk_i : in  STD_LOGIC;
        rst_n_i : in  STD_LOGIC;
        row_col_i : in  STD_LOGIC_VECTOR (7 downto 0);
        led_o : out  STD_LOGIC_VECTOR (3 downto 0)
    );
  end component;

  signal s_row : std_logic_vector(3 downto 0);
  signal s_col : std_logic_vector(3 downto 0);
  signal s_row_reg : std_logic_vector(3 downto 0);
  signal s_row_col : std_logic_vector(7 downto 0);

begin
  i_row_scan : row_scan
  port map(
    clk_i => clk_50mhz_i,
    rst_n_i => rst_n_i,
    row_o   => s_row
  );

  row_o <= s_row;

  i_row_delay : row_delay
  port map(
    clk_i => clk_50mhz_i,
    rst_n_i => rst_n_i,
    row_i => s_row,
```

```
    row_reg_o => s_row_reg
  );
  i_clr_jitter_with_reg0 : clr_jitter_with_reg
  port map(
    clk_50mhz_i => clk_50mhz_i,
    rst_n_i => rst_n_i,
    button_i => col_i(0),
    button_o => s_col(0)
  );
  i_clr_jitter_with_reg1 : clr_jitter_with_reg
  port map(
    clk_50mhz_i => clk_50mhz_i,
    rst_n_i => rst_n_i,
    button_i => col_i(1),
    button_o => s_col(1)
  );
  i_clr_jitter_with_reg2 : clr_jitter_with_reg
  port map(
    clk_50mhz_i => clk_50mhz_i,
    rst_n_i => rst_n_i,
    button_i => col_i(2),
    button_o => s_col(2)
  );
  i_clr_jitter_with_reg3 : clr_jitter_with_reg
  port map(
    clk_50mhz_i => clk_50mhz_i,
    rst_n_i => rst_n_i,
    button_i => col_i(3),
    button_o => s_col(3)
  );
  i_combine : combine
  port map(
    row_i => s_row_reg,
    col_i => s_col,
    row_col_o => s_row_col
  );
  i_decoder : decoder
  port map(
    clk_i => clk_50mhz_i,
    rst_n_i => rst_n_i,
    row_col_i => s_row_col,
    led_o => led_o
  );
end rtl;
```

六、综合、仿真结果及分析：

RTL 级电路综合：

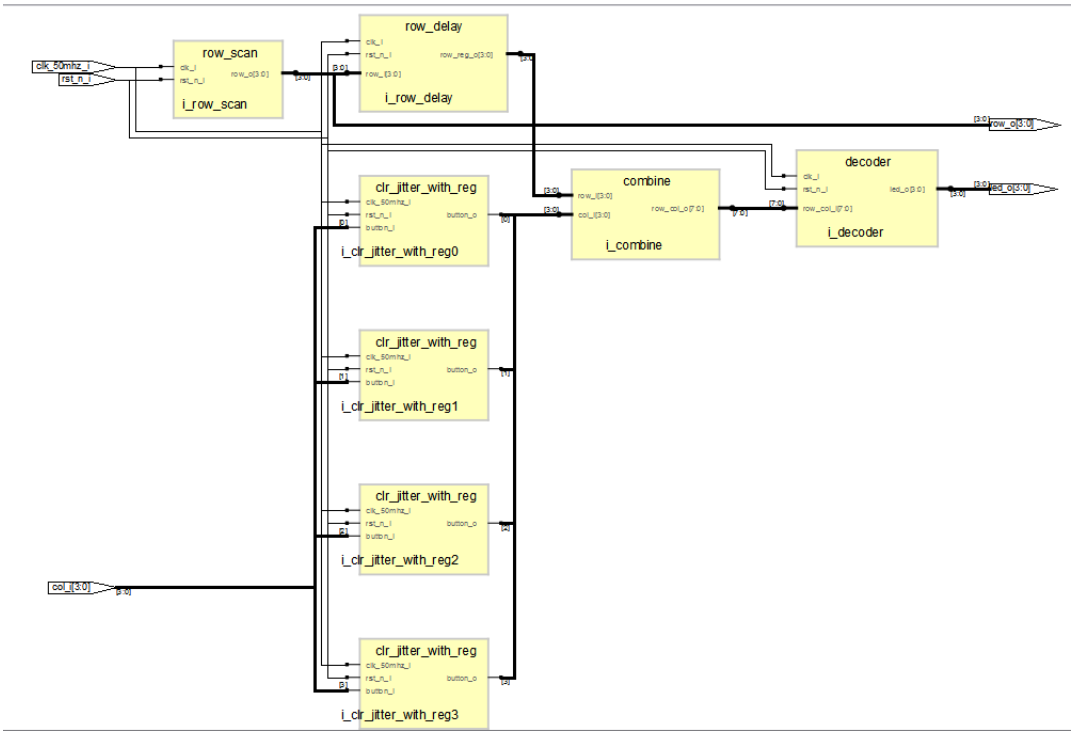

图 6-1 整体 RTL 级电路

50Hz 时钟信号发生器电路图：

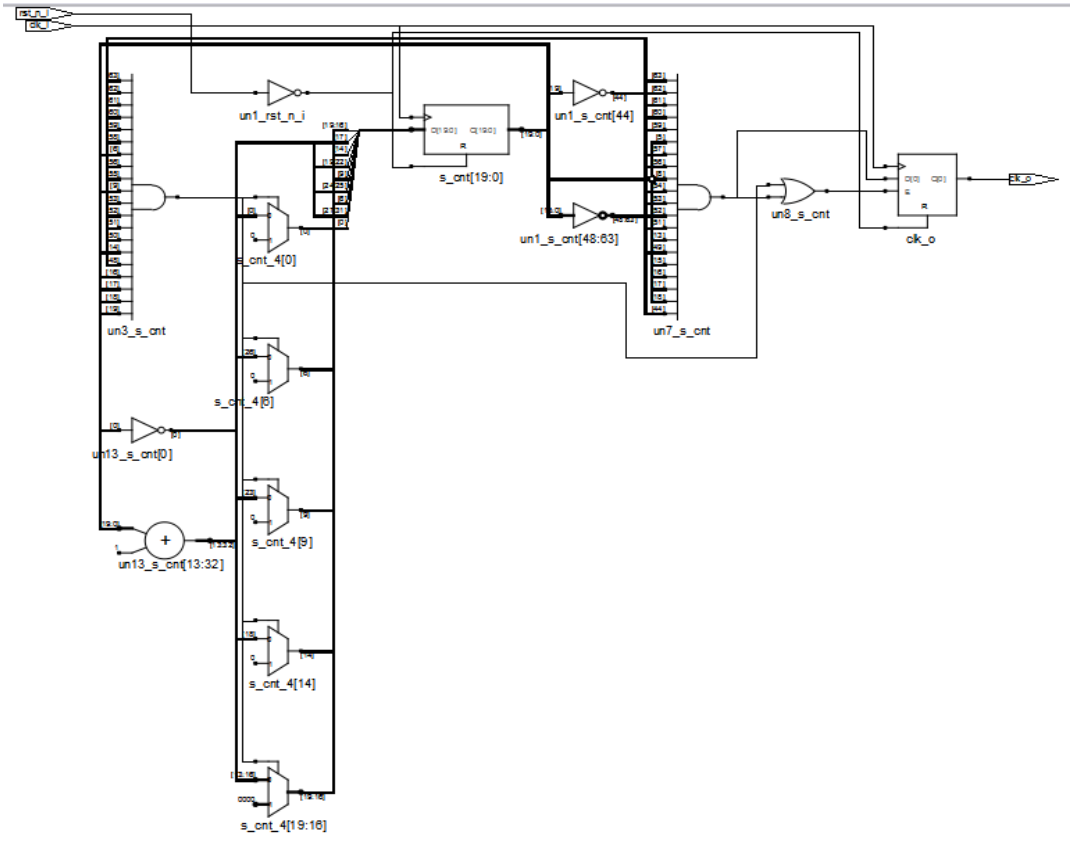

图 6-2 50Hz 时钟信号产生电路

row_scan 行扫描电路图：

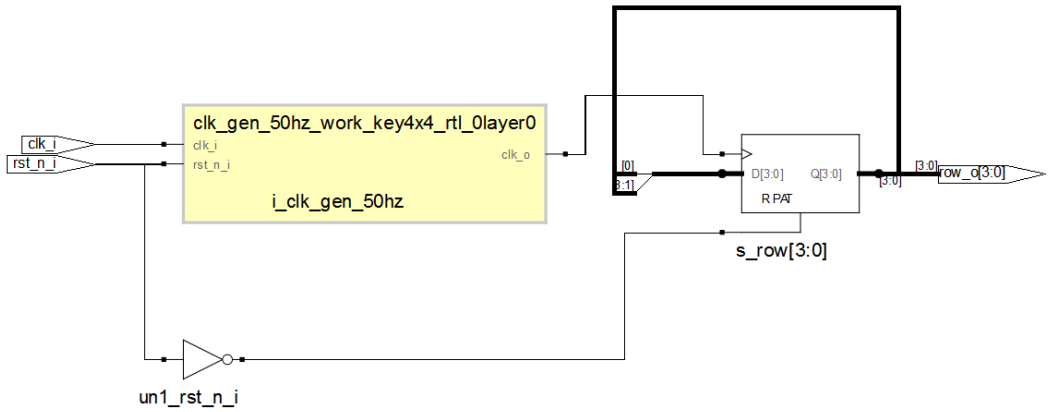

图 6-3 行扫描电路

row_delay 行延迟电路图：

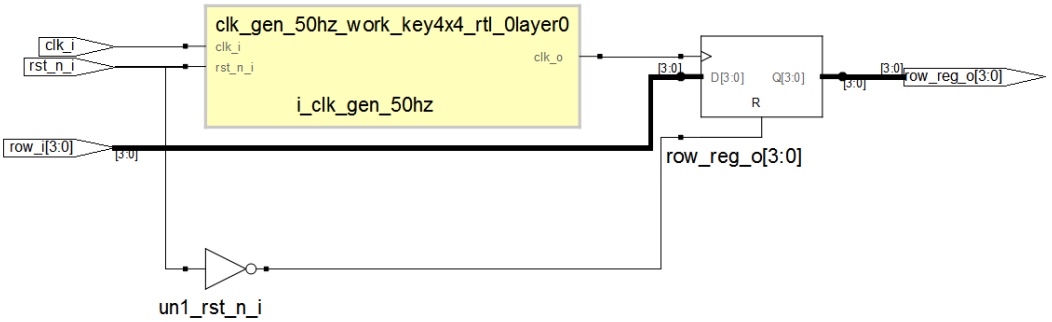

图 6-4 行延迟电路

按键消抖电路图：

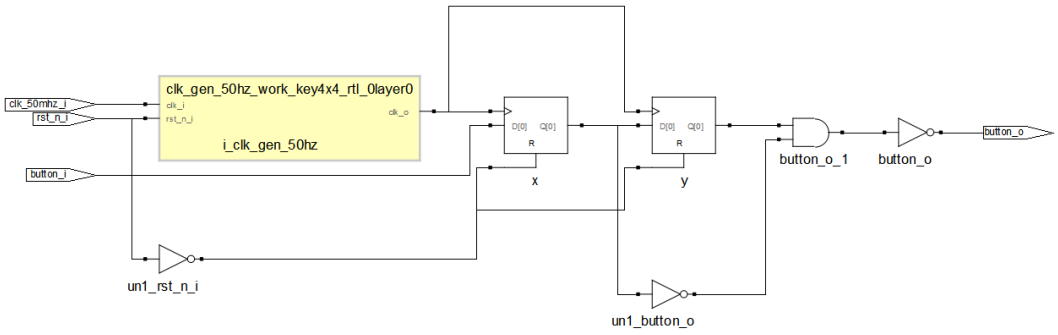

图 6-5 按键消抖电路

译码器电路图：

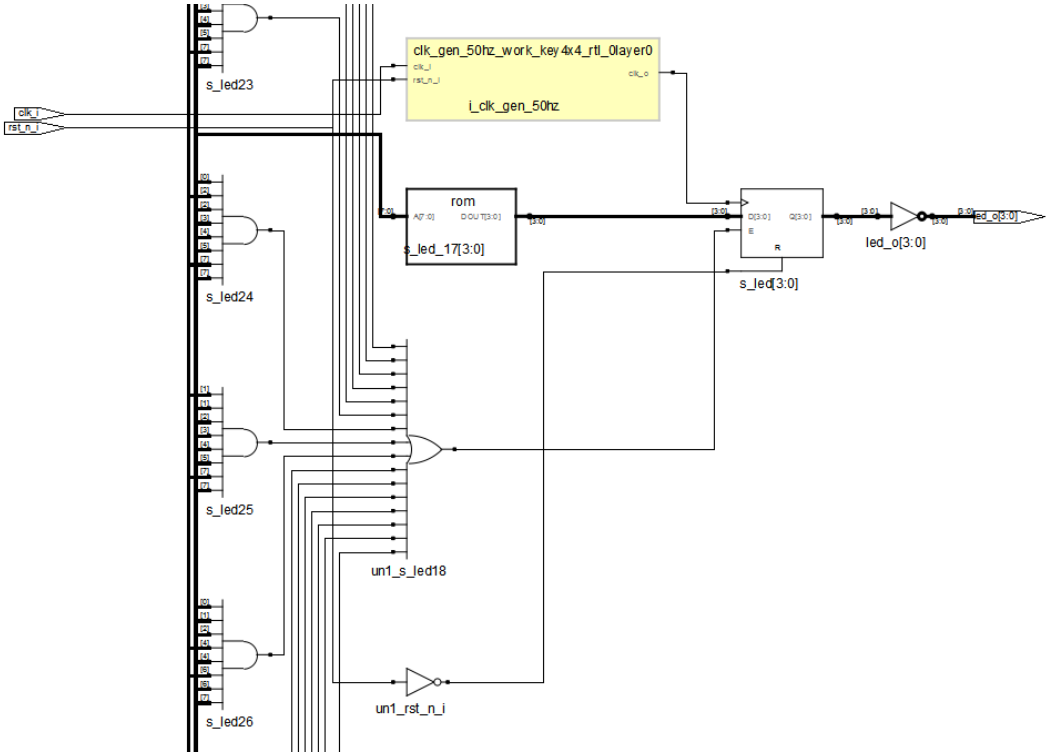

图 6-6 整体 RTL 级电路

2、电路仿真

测试激励 Test Bench 代码如下：

```
--------------------------------------------------------------------------------
--模块名称：combine_tb
--摘要提示：combine 测试激励
--当前版本：1.0.0
--模块作者：
--完成日期：2018 年 9 月 19 日
--内容提要：
--需要注意：
--------------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY combine_tb IS
END combine_tb;

ARCHITECTURE behavior OF combine_tb IS
```

```
    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT combine
    PORT(
         row_i : IN  std_logic_vector(3 downto 0);
         col_i : IN  std_logic_vector(3 downto 0);
         row_col_o : OUT  std_logic_vector(7 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal row_i : std_logic_vector(3 downto 0) := (others => '0');
   signal col_i : std_logic_vector(3 downto 0) := (others => '0');

  --Outputs
   signal row_col_o : std_logic_vector(7 downto 0);
   -- No clocks detected in port list. Replace <clock> below with
   -- appropriate port name

BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: combine PORT MAP (
          row_i => row_i,
          col_i => col_i,
          row_col_o => row_col_o
        );


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

    row_i <= "0111";
    col_i <= "1011";

      -- insert stimulus here

      wait;
   end process;
```

```
END;
```

仿真结果如下图所示：

combine 电路的仿真结果如图所示：

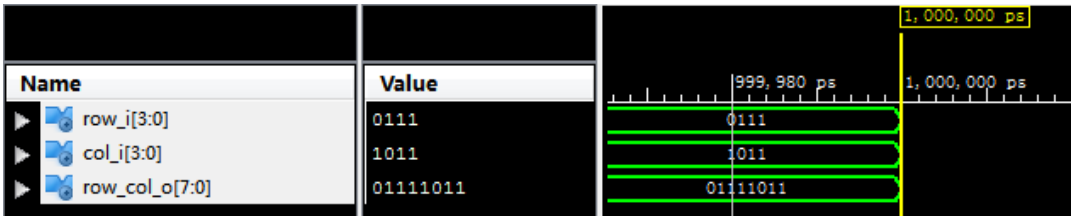

图 6-7 combine 电路仿真

```
--------------------------------------------------------------------------------
--模块名称：row_scan_tb
--摘要提示：行扫描测试激励
--当前版本：1.0.0
--模块作者：
--完成日期：2018 年 9 月 19 日
--内容提要：
--需要注意：
--------------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY row_scan_tb IS
END row_scan_tb;

ARCHITECTURE behavior OF row_scan_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT row_scan
    PORT(
         clk_i : IN  std_logic;
         rst_n_i : IN  std_logic;
         row_o : OUT  std_logic_vector(3 downto 0)
```

```
        );
    END COMPONENT;


   --Inputs
   signal clk_i : std_logic := '0';
   signal rst_n_i : std_logic := '0';

  --Outputs
   signal row_o : std_logic_vector(3 downto 0);

   -- Clock period definitions
   constant clk_i_period : time := 10 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: row_scan PORT MAP (
          clk_i => clk_i,
          rst_n_i => rst_n_i,
          row_o => row_o
        );

   -- Clock process definitions
   clk_i_process :process
   begin
    clk_i <= '0';
    wait for clk_i_period/2;
    clk_i <= '1';
    wait for clk_i_period/2;
   end process;


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
    rst_n_i <= '1';
      wait for clk_i_period*10;


      -- insert stimulus here
```

```
      wait;
   end process;

END;
```

仿真结果如下图所示：

row_scan 电路的仿真结果：

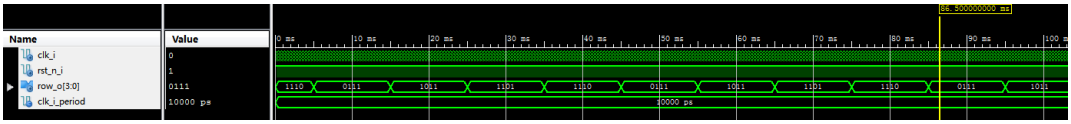

图 6-8　row_scan 电路的仿真

```
--------------------------------------------------------------------------------
--模块名称：row_delay_tb
--摘要提示：行延迟测试激励
--当前版本：1.0.0
--模块作者：
--完成日期：2018 年 9 月 19 日
--内容提要：
--需要注意：
--------------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY row_delay_tb IS
END row_delay_tb;

ARCHITECTURE behavior OF row_delay_tb IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT row_delay
    PORT(
         clk_i : IN  std_logic;
         rst_n_i : IN  std_logic;
         row_i : IN  std_logic_vector(3 downto 0);
         row_reg_o : OUT  std_logic_vector(3 downto 0)
        );
```

```
    END COMPONENT;

   --Inputs
   signal clk_i : std_logic := '0';
   signal rst_n_i : std_logic := '0';
   signal row_i : std_logic_vector(3 downto 0) := (others => '0');

  --Outputs
   signal row_reg_o : std_logic_vector(3 downto 0);

   -- Clock period definitions
   constant clk_i_period : time := 10 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
   uut: row_delay PORT MAP (
         clk_i => clk_i,
         rst_n_i => rst_n_i,
         row_i => row_i,
         row_reg_o => row_reg_o
       );

   -- Clock process definitions
   clk_i_process :process
   begin
    clk_i <= '0';
    wait for clk_i_period/2;
    clk_i <= '1';
    wait for clk_i_period/2;
   end process;

   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;
    rst_n_i <= '1';
      wait for clk_i_period*1000000;
    row_i <= "0111";
      wait for clk_i_period*1000000;
    row_i <= "1011";
      wait for clk_i_period*1000000;
    row_i <= "1101";
```

```
      wait for clk_i_period*1000000;
    row_i <= "1110";
      wait for clk_i_period*1000000;
    row_i <= "0111";
      wait for clk_i_period*1000000;
    row_i <= "1011";

      -- insert stimulus here

      wait;
   end process;
END;
```

仿真结果如下图所示：

row_delay 电路的仿真结果：

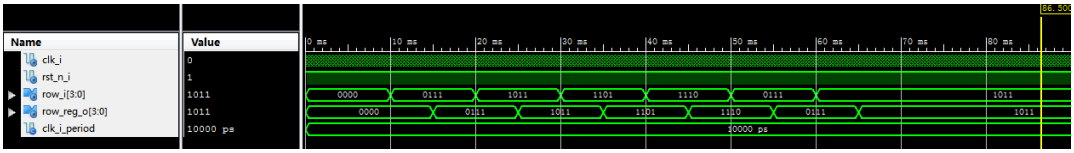

图 6-9　row_delay 电路的仿真

```
----------------------------------------------------------------------------------
--模块名称：decoder_tb
--摘要提示：译码器测试激励
--当前版本：1.0.0
--模块作者：
--完成日期：2018 年 9 月 19 日
--内容提要：
--需要注意：
----------------------------------------------------------------------------------
--取代版本：
--模块作者：
--完成日期：
--修改内容：
--修改文件：
----------------------------------------------------------------------------------
```

仿真结果如下图所示：

decoder 电路的仿真结果：

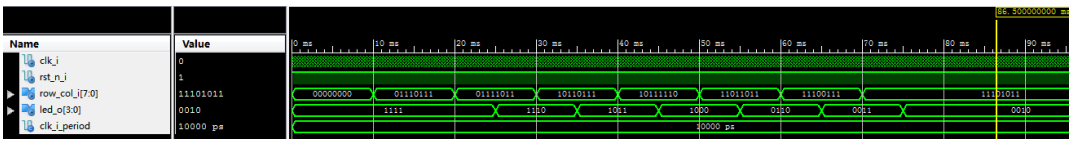

图 6-10　decoder 电路的仿真

七、FPGA 板级验证及结果分析

1、在工程中加入引脚约束文件，代码如下

```
##-----------------------------------------------------------------
##模块名称：Key4x4.ucf
##摘要提示：
##当前版本：1.0.0
##模块作者：
##完成日期：20XX 年 XX 月 XX 日
##内容提要：
##需要注意：
##-----------------------------------------------------------------
##取代版本：
##模块作者：
##完成日期：
##修改内容：
##修改文件：
##-----------------------------------------------------------------
Net "clk_50mhz_i"  LOC=P129;
Net "rst_n_i"      LOC=P85;

Net "col_i<3>"     LOC=P74;
Net "col_i<2>"     LOC=P75;
Net "col_i<1>"     LOC=P77;
Net "col_i<0>"     LOC=P82;

Net "row_o<0>"     LOC=P76;
Net "row_o<1>"     LOC=P81;
Net "row_o<2>"     LOC=P83;
Net "row_o<3>"     LOC=P86;

Net "led_o<3>"     LOC=P104;
Net "led_o<2>"     LOC=P105;
Net "led_o<1>"     LOC=P98;
Net "led_o<0>"     LOC=P103;

Net "col_i<3>"     PULLUP;
Net "col_i<2>"     PULLUP;
Net "col_i<1>"     PULLUP;
Net "col_i<0>"     PULLUP;
```

综合之后生成 bit 文件烧进实验箱中观察到如下图所示的现象：

图 7-1 按下按键 1 现象

图 7-2 按下按键 15 现象

当按下矩阵键盘上的按键的时候，四盏 LED 灯会按照按下的按键的位置（0~15）的二进制来点亮。当按下按键 1 时，LED0~3 的点亮顺序为 0001，如图 7-1 所示。而当按下按键 15 时，LED0~3 的点亮顺序为 1111，全亮，如图 7-2 所示。

当按下复位键时，四盏 LED 灯全部熄灭，如图 7-3 所示。

图 73 按下复位键现象

八、实验总结：

通过这次实验的学习，我掌握了矩阵键盘的扫描原理，了解了按键防抖的原理，学会了利用 FPGA 芯片的 50MHz 的时钟来产生自己需要的时钟信号，即掌握了分频的方法。同时，我还学会了编写译码器和按键防抖的模块，完成了将矩阵键盘的值显示在 LED 灯上的实验。

指导教师批阅意见：

成绩评定：

指导教师签字：

年　　月　　日

备注：

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。