实验报告

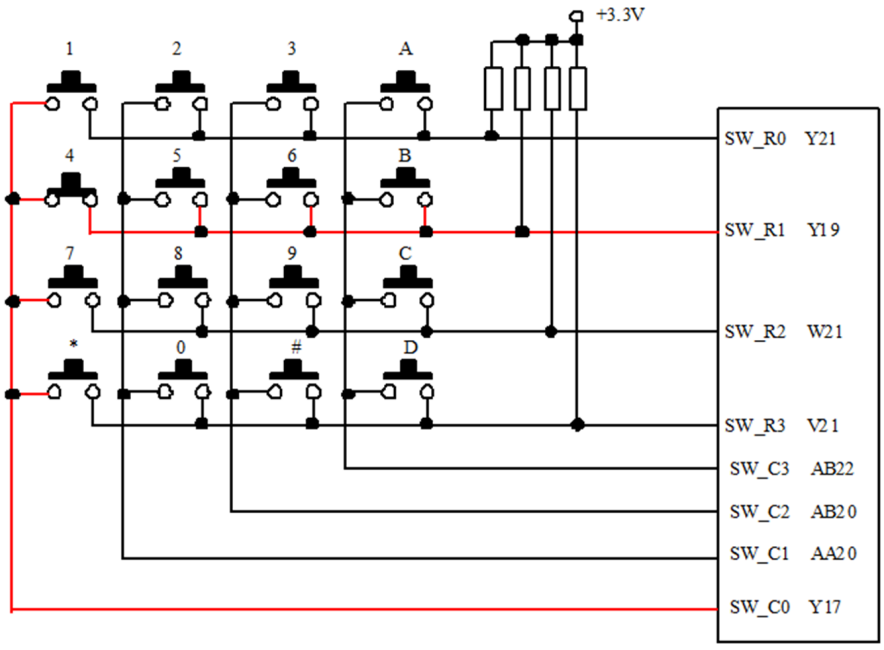
实验题目：矩阵键盘　　姓名：牟真伟　　学号：PB20051061

实验内容

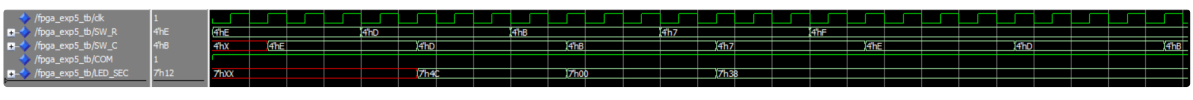
设计一个4x4矩阵键盘读取电路，每次按下矩阵键盘上一个键时，在一位七段数码管上显示按下的键上面的值，当松开按键时，数码管显示不变。

设计分析

矩阵键盘读取电路分四条行信号线和四条列信号线，四条行信号线在按键未按下时，均被上拉到高电平，当按键被按下时，行信号线为按下按键所在列的列信号线的电平。当在列信号线设置为"1110","1101","1011","0111"的扫描信号时，列信号依次为低电平，再每个扫描期间，读取四条行信号线的电平，当某行信号线读取到低电平时，说明此时列信号为低电平的列有按键按下，根据此时列信号线和行信号线的位置，即可判断当前按下的按键位置。可以设置一个时钟信号，再每个时钟周期设置列扫描信号，读取行信号，再将列信号和行信号进行拼接，找出其对应的按键位置（4位），再将按键位置信号发送给一个47译码器，得出按下的按键对应数码管应该显示的段选信号，位选信号选择一个置1，即可显示当前按下按键的值。为防止扫描速率过快，导致信号不能同步，可将输入的时钟信号进行一定分频处理。



仿真结果记录



clk为时钟信号，仿真程序采用4分频，SW_R为行信号，SW_C为列扫描信号，COM为数码管的位选端，LED_SEC为数码管的段选端。

在每个分频后的时钟的上升沿，根据当前的列扫描信号SW_C和读取的行信号SW_R得出当前按下的按键，LED_SEC为显示按键按下值需要的数码管的段选信号。

在第二个分频时钟的上升沿，此时列扫描信号为"1110"(EH)，行读取信号为"1011"(DH),为第一列，第二行,得出按下按键的值为4，LED_SEC为"1001100"(4CH)。

在第三个分频时钟的上升沿，此时列扫描信号为"1101"(DH)，行读取信号为"1101"(BH),为第二列，第三，行得出按下按键的值为8，LED_SEC为"0000000"(00H)。

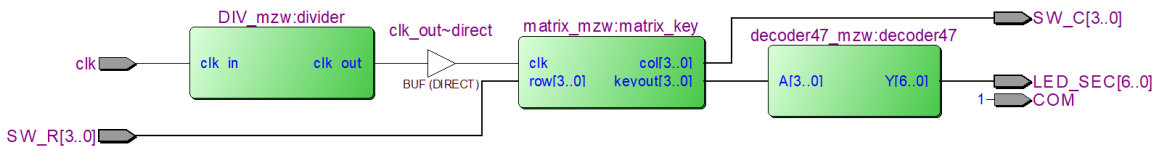
当行读取信号为"1111"(FH)时，说明此时无按键按下，LED_SEC保持不变。

中间结果记录

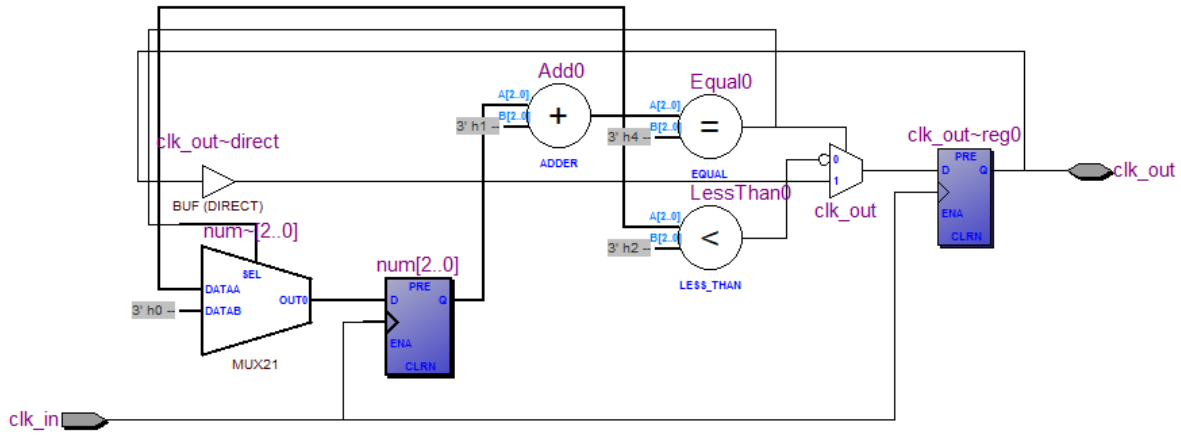
全编译后资源占用情况：

Flow Summary	
Flow Status	Successful - Sat Oct 22 07:57:29 2022
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	FPGA_EXP5_mzw
Top-level Entity Name	FPGA_EXP5_mzw
Family	Cyclone V
Device	5CEFA2F23C8
Timing Models	Final
Logic utilization (in ALMs)	12 / 9,430 (< 1 %)
Total registers	14
Total pins	17 / 224 (8 %)
Total virtual pins	0
Total block memory bits	0 / 1,802,240 (0 %)
Total DSP Blocks	0 / 25 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI PMA RX ATT Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total HSSI PMA TX ATT Serializers	0
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 4 (0 %)

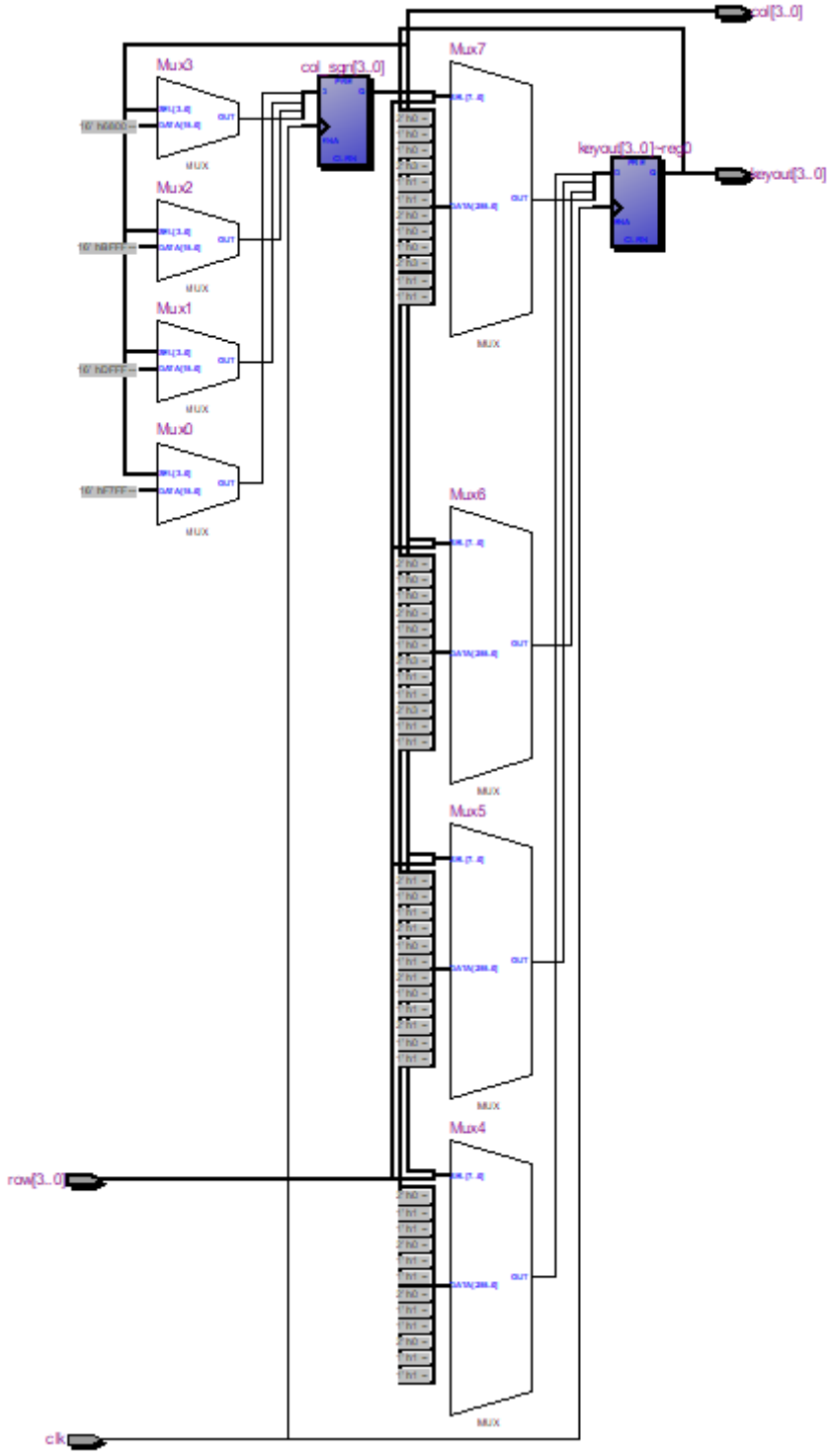
电路总RTL结构图：



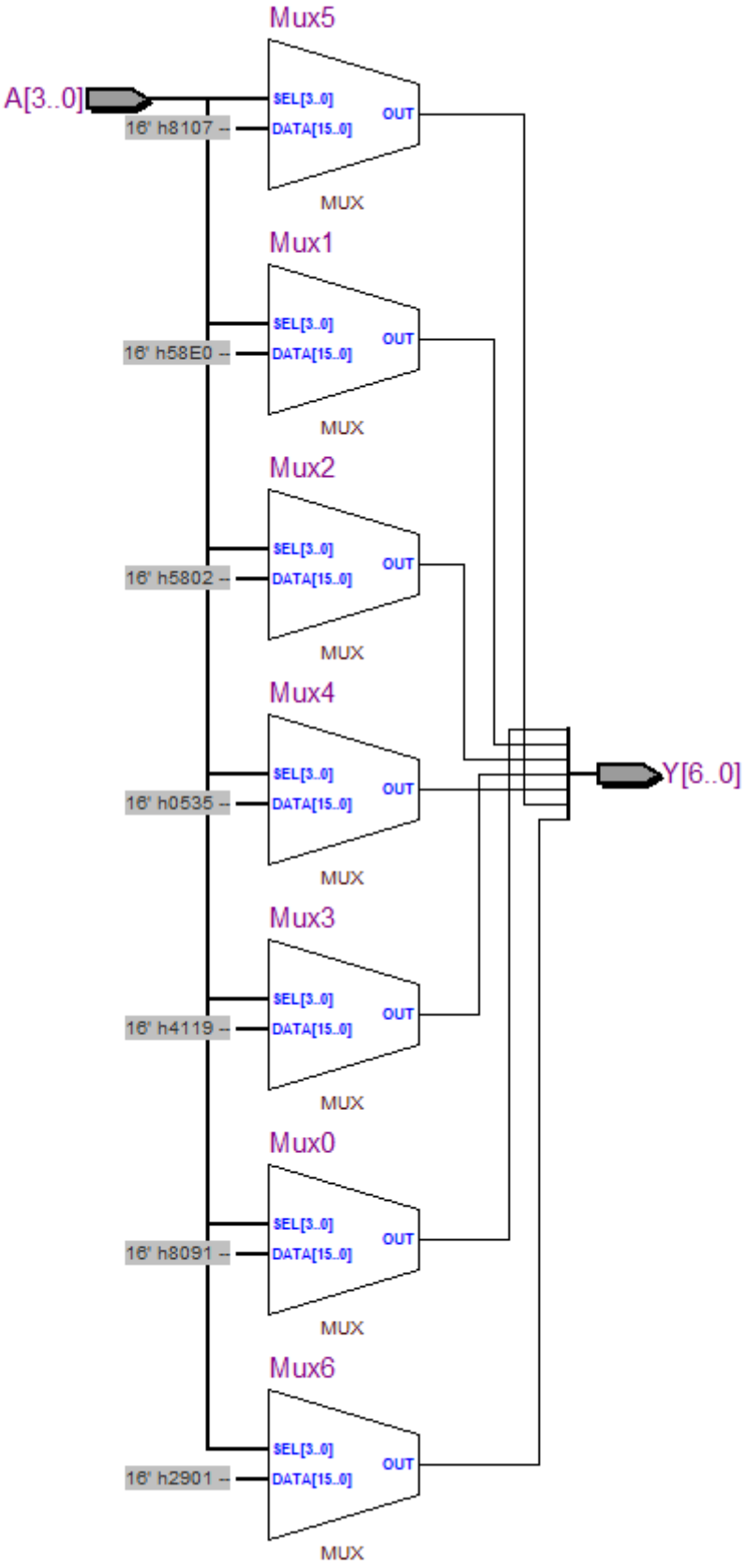
分频器RTL结构图：



矩阵键盘扫描读取器RTL结构图：


col[3..0]
Mux3
col_sgnl[3..0]
Mux7
keyout[3..0]~reg0
keyout[3..0]
Mux2
Mux1
Mux0
Mux6
Mux5
Mux4
row[3..0]
clk
MUX

译码器RTL结构图：



FPGA验证结果记录

实验管脚约束情况如下：

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
clk	Input	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V (default)		12mA (default)	
COM	Output	PIN_B16	7A	B7A_N0	PIN_B16	2.5 V (default)		12mA (default)	1 (default)
LED_SEC[6]	Output	PIN_K19	7A	B7A_N0	PIN_K19	2.5 V (default)		12mA (default)	1 (default)
LED_SEC[5]	Output	PIN_H18	7A	B7A_N0	PIN_H18	2.5 V (default)		12mA (default)	1 (default)
LED_SEC[4]	Output	PIN_K20	7A	B7A_N0	PIN_K20	2.5 V (default)		12mA (default)	1 (default)
LED_SEC[3]	Output	PIN_M18	5B	B5B_N0	PIN_M18	2.5 V (default)		12mA (default)	1 (default)
LED_SEC[2]	Output	PIN_E16	7A	B7A_N0	PIN_E16	2.5 V (default)		12mA (default)	1 (default)
LED_SEC[1]	Output	PIN_G13	7A	B7A_N0	PIN_G13	2.5 V (default)		12mA (default)	1 (default)
LED_SEC[0]	Output	PIN_G17	7A	B7A_N0	PIN_G17	2.5 V (default)		12mA (default)	1 (default)
SW_C[3]	Output	PIN_AB22	4A	B4A_N0	PIN_AB22	2.5 V (default)		12mA (default)	1 (default)
SW_C[2]	Output	PIN_AB20	4A	B4A_N0	PIN_AB20	2.5 V (default)		12mA (default)	1 (default)
SW_C[1]	Output	PIN_AA20	4A	B4A_N0	PIN_AA20	2.5 V (default)		12mA (default)	1 (default)
SW_C[0]	Output	PIN_Y17	4A	B4A_N0	PIN_Y17	2.5 V (default)		12mA (default)	1 (default)
SW_R[3]	Input	PIN_V21	4A	B4A_N0	PIN_V21	2.5 V (default)		12mA (default)	
SW_R[2]	Input	PIN_W21	4A	B4A_N0	PIN_W21	2.5 V (default)		12mA (default)	
SW_R[1]	Input	PIN_Y19	4A	B4A_N0	PIN_Y19	2.5 V (default)		12mA (default)	
SW_R[0]	Input	PIN_Y21	4A	B4A_N0	PIN_Y21	2.5 V (default)		12mA (default)	

clk为50MHz时钟，采用20000分频。COM位选端接实验箱上最右端的数码管，LED_SEC接数码管段选端。SW_C接矩阵键盘列信号线，SW_R接矩阵键盘行信号线。

实验现象如下：

当按下矩阵键盘上某个按键时，数码管上显示按下按键的值，松开按键后，数码管显示值不变。

实验总结

本次实验通过矩阵键盘读取电路的设计，学习了矩阵键盘扫描读取电路的原理，矩阵键盘和多位数码管显示都采用了扫描的原理，可以在节省引脚的情况下，通过快速地扫描单个单元，实现类似扫描全部单元的效果。扫描电路的关键是做好信号同步，信号同步的关键一方面是要选择合适的时钟信号，另一方面，尽量在关键的信号读取和输出电路上加上时钟信号，以达到更好的同步效果。

VHDL源代码

```
-- FPGA_EXP5_mzw @PB20051061 牟真伟

library ieee;
use ieee.std_logic_1164.all;
entity FPGA_EXP5_mzw is
    port(
        clk :IN std_logic;
        SW_R :IN std_logic_vector(3 downto 0);
        SW_C :OUT std_logic_vector(3 downto 0);
        COM :OUT std_logic;
        LED_SEC :OUT std_logic_vector(6 downto 0)
    );
end FPGA_EXP5_mzw;

architecture arch_FPGA_EXP5 of FPGA_EXP5_mzw is
    component DIV_mzw port(
        clk_in :IN std_logic;
        clk_out :INOUT std_logic
    );
    end component;

    component decoder47_mzw port(
        A :IN std_logic_vector(3 downto 0);
```

```
        Y :OUT std_logic_vector(6 downto 0) -- 6543210 -> abcdefg
    );
    end component;

    component matrix_mzw port(
            clk: in std_logic;
            row:  in std_logic_vector(3 downto 0);
            col: out std_logic_vector(3 downto 0);
            keyout: out std_logic_vector(3 downto 0)
    );
    end component;
    signal clk_out :std_logic;
    signal location :std_logic_vector(3 downto 0);
begin
    divider :DIV_mzw port map(clk,clk_out);
    decoder47 :decoder47_mzw port map(location,LED_SEC);
    matrix_key :matrix_mzw port map(clk_out,SW_R,SW_C,location);
    COM <= '1';
end arch_FPGA_EXP5;


-- 矩阵键盘扫描读取电路 @ PB20051061 牟真伟

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity matrix_mzw is
    port(
        clk: in std_logic;
        row:  in std_logic_vector(3 downto 0);
        col: out std_logic_vector(3 downto 0);
        keyout: out std_logic_vector(3 downto 0)
    );
end matrix_mzw;

architecture arch_matrix of matrix_mzw is
    signal col_sgn:  std_logic_vector(3 downto 0);
    signal con: std_logic_vector(7 downto 0);
begin
--分频，在1~10KHZ左右的时钟进行键盘扫描
    --产生列扫描信号
    process(clk)
    begin
        if (clk'event and clk = '1') then
            case col_sgn is
                when "1110" => col_sgn<="1101";
                when "1101" => col_sgn<="1011";
                when "1011" => col_sgn<="0111";
                when "0111" => col_sgn<="1110";
                when others => col_sgn<="1110";
```

```
                end case;
            end if;
    end process;
--对行信号和列信号进行组合--
    col<=col_sgn;
    con<= col_sgn & row;--列行
--对组合的扫描信号进行判断，并输出按键指示信号以及编码--
    process(clk)
    begin
        if (clk'event and clk = '1') then
            case con is
                when "11101110" => keyout<="0000";
                when "11011110" => keyout<="0001";
                when "10111110" => keyout<="0010";
                when "01111110" => keyout<="0011";
                when "11101101" => keyout<="0100";
                when "11011101" => keyout<="0101";
                when "10111101" => keyout<="0110";
                when "01111101" => keyout<="0111";
                when "11101011" => keyout<="1000";
                when "11011011" => keyout<="1001";
                when "10111011" => keyout<="1010";
                when "01111011" => keyout<="1011";
                when "11100111" => keyout<="1100";
                when "11010111" => keyout<="1101";
                when "10110111" => keyout<="1110";
                when "01110111" => keyout<="1111";
                when others => null;
            end case;
        end if;
    end process;
end arch_matrix;

-- 矩阵键盘与数码管译码器 @PB20051061 牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity decoder47_mzw is
    port(
        A :IN std_logic_vector(3 downto 0);
        Y :OUT std_logic_vector(6 downto 0) -- 6543210 -> abcdefg
    );
end decoder47_mzw;

architecture arch_decoder47 of decoder47_mzw is

begin
    process(A)
    begin
```

```
        case A is
            when "0000" => Y <= "1001111";--1
            when "0001" => Y <= "0010010";--2
            when "0010" => Y <= "0000110";--3
            when "0011" => Y <= "0001000";--A
            when "0100" => Y <= "1001100";--4
            when "0101" => Y <= "0100100";--5
            when "0110" => Y <= "0100000";--6
            when "0111" => Y <= "1100000";--b
            when "1000" => Y <= "0001111";--7
            when "1001" => Y <= "0000000";--8
            when "1010" => Y <= "0000100";--9
            when "1011" => Y <= "0110001";--C
            when "1100" => Y <= "0110000";--*E
            when "1101" => Y <= "0000001";--0
            when "1110" => Y <= "0111000";--#F
            when "1111" => Y <= "1000010";--d
            when others => null;
        end case;
    end process;
end arch_decoder47;


-- 分频器 @PB20051061牟真伟

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity DIV_mzw  is
    port(
        clk_in :IN std_logic;
        clk_out :INOUT std_logic:='0'
    );
end DIV_mzw;

architecture arch_DIV of DIV_mzw is
begin
    process(clk_in)
            --variable num: integer range 0 to 20000;  --2000分频
            variable num: integer range 0 to 4 := 0;  --仿真4分频
    begin
         if (clk_in'event and clk_in='1') then
                        num := num + 1;
              --if  (num = 20000)  then
                    if  (num = 4)  then --仿真2分频
                        num := 0;
              --elsif  (num < 10000)  then
                    elsif  (num < 2)  then --仿真2分频
```

```
                        clk_out <= '0';
                else
                        clk_out <= '1';
                end if;
        end if;
    end process;
end arch_DIV;

-- FPGA_EXP5_tb  @ PB20051061 牟真伟

library ieee;
use ieee.std_logic_1164.all;

entity FPGA_EXP5_tb is
end FPGA_EXP5_tb;

architecture arch_FPGA_EXP5_tb of FPGA_EXP5_tb is
    component FPGA_EXP5_mzw port(
        clk :IN std_logic;
        SW_R :IN std_logic_vector(3 downto 0);
        SW_C :INOUT std_logic_vector(3 downto 0);
        COM :OUT std_logic;
        LED_SEC :OUT std_logic_vector(6 downto 0)
    );
    end component;

    signal clk :std_logic;
    signal SW_R :std_logic_vector(3 downto 0);
    signal SW_C :std_logic_vector(3 downto 0);
    signal COM :std_logic;
    signal LED_SEC :std_logic_vector(6 downto 0);

begin
    fpga_exp5 :FPGA_EXP5_mzw port map(clk,SW_R,SW_C,COM,LED_SEC);
    process
    begin
        clk <= '0';
        wait for 10 ns;
        clk <= '1';
        wait for 10 ns;
    end process;

    process
    begin
        SW_R <= "1110";
        wait for 80 ns;
        SW_R <= "1101";
        wait for 80 ns;
        SW_R <= "1011";
        wait for 80 ns;
```

```
        SW_R <= "0111";
        wait for 80 ns;
        SW_R <= "1111";
        wait for 500 ns;
    end process;
end arch_FPGA_EXP5_tb;
```