七段数码管动态显示电路设计

Pb16060240
黄骏达

实验目的：

1. 熟悉 8 位七段数码管动态显示的控制
2. 熟练掌握 VHDL test bench 的编写和仿真流程
3. 熟练掌握 Altera FPGA 的开发环境、设计步骤和流程

8 位数码管原理：

1. 首先七段数码管由 7 位数据信号进行内容控制，七位数据总线分别控制七段数码管的一段，当数码管为共阳极数码管时，数码管的所有阳极共线，并连接在高电平上。

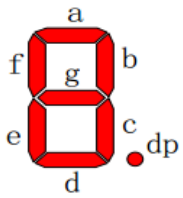
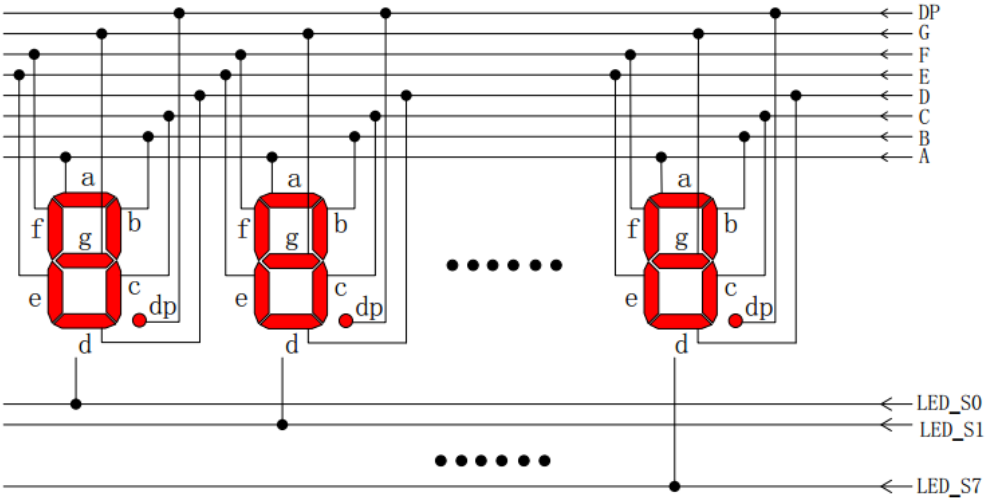


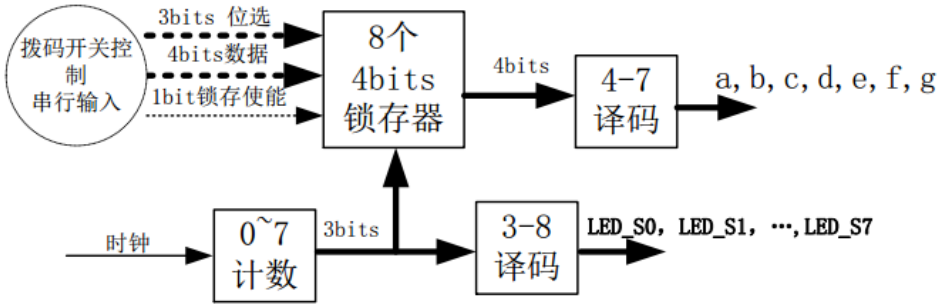
图 1　七段数码管构成图

2. 8 位是指需要同时控制 8 个七位数码管进行输出，通过一个控制器，轮流扫描 8 个数码管，即可实现 8 个数码管的轮流显示。

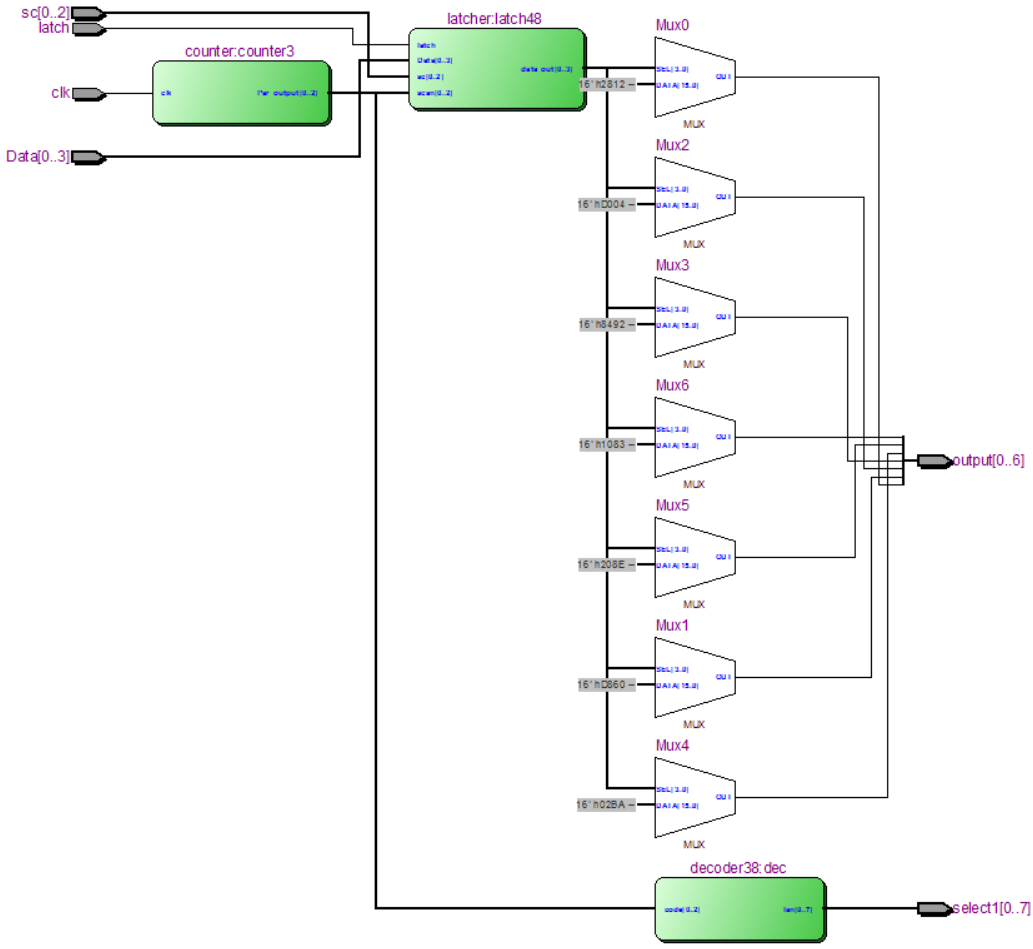


实验设计：

1. 如下为模块化电路设计基本架构：



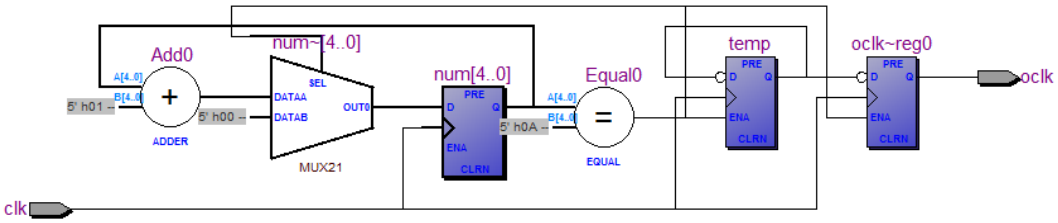
2. 首先设计 4-7 译码器，并将 4-7 译码器作为顶层实体，电路图如下所示：



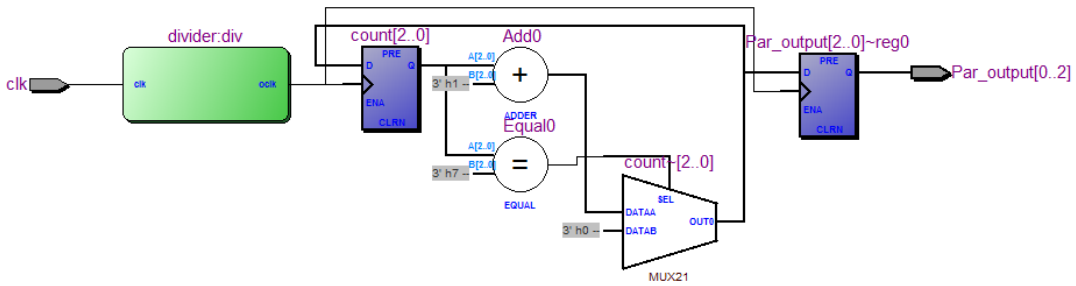
3. 设计锁存器，锁存器能够被 4 位输入进行控制，其中 1 位控制数据信号的锁存，其中 3 位实现对于 8 个内部虚拟锁存器的选择。然后由 4 位数据输入，将数码管上的显示数据锁存在这 8 个内部虚拟锁存器的其中之一之中。由于锁存器电路较为复杂，将在后面贴出。

4. 另一部分是扫描电路，扫描电路由一个分频器，一个 8 进制计数器和一个 3-8 译码器组成。

5. 分频器将初始 50m 时钟进行分频输出，电路如下所示（不过，分频器在此可有可无）：



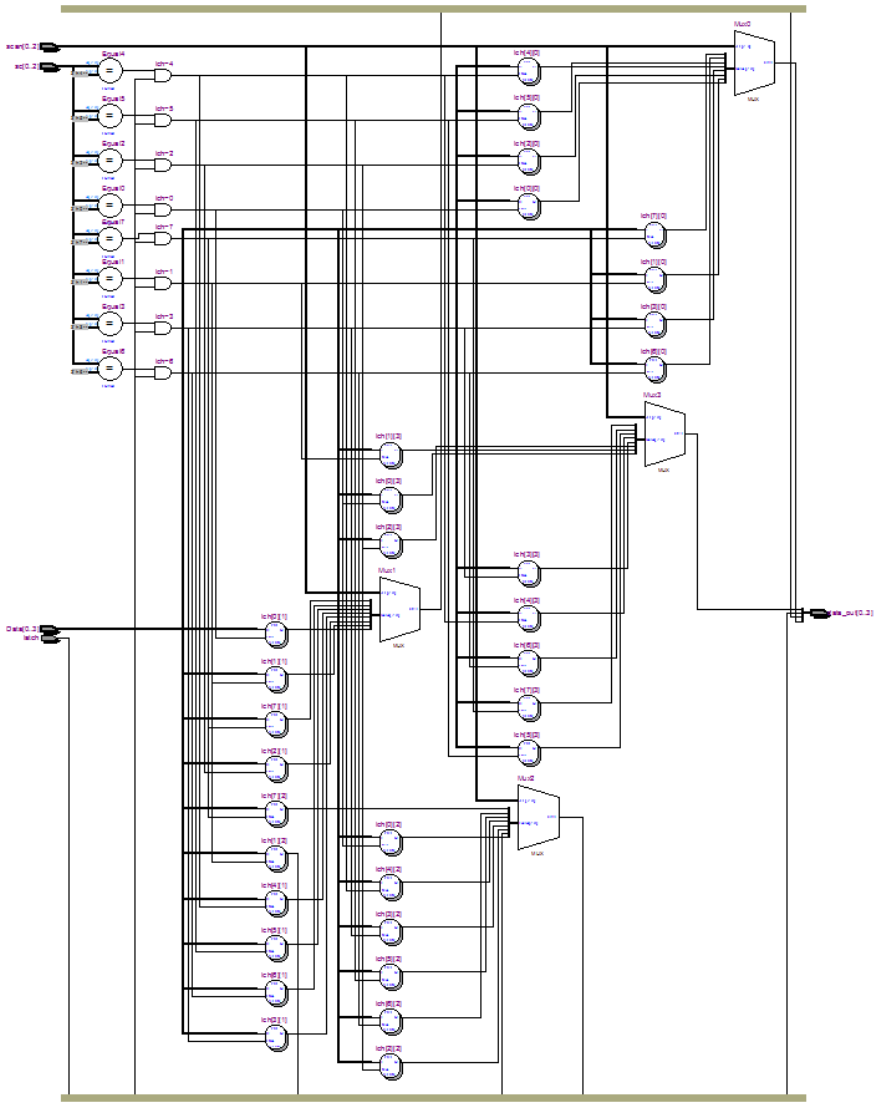
6. 8 进制计数器将分频器的分频时钟输出进行计数，并将数值一个输出给锁存器，进行数码管映射数据的扫描信号，一个输出给 3-8 译码器，八进制计数器电路图如下所示：



7. 3-8 译码器负责将 3 位八进制计数器的输出进行译码并输出给数码管选择控制总线，在此次电路设计中，由于已经完全掌握使用 case 来进行译码，因此使用了 others 与 for 循环进行组合，降低了编写代码的复杂度，但同时在电路中发现，尽管代码简洁，但电路相对于 case 的电路复杂很多，因此并不推崇使用，电路附后。

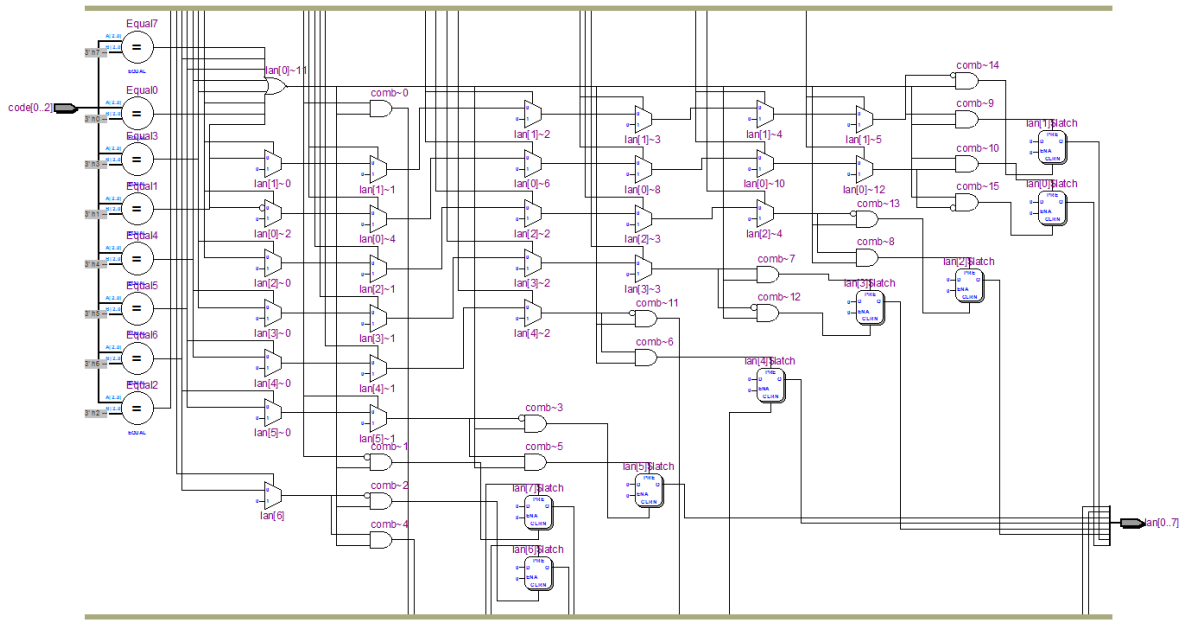
电路图：

- 锁存器



- 38 译码器

(由于使用新方法，因此电路复杂，并不推崇)



VHDL 源代码：

(counter.vhd)8 进制计数器

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity counter is
port(clk : in std_logic;
        Par_output : out std_logic_vector(0 to 2));
end entity;

architecture behave of counter is

component divider is
    port (clk : IN std_logic;
            oclk : OUT std_logic);
end component;

signal fclk : std_logic;

begin
    div : divider port map (clk,fclk);

    process(fclk)
```

```
    variable count : integer range 0 to 7 := 0;
    begin
        if (fclk'event and fclk = '1') then
            if (count = 7 ) then
                count := 0;
            else
                count := count + 1;
            end if;
            Par_output <= conv_std_logic_vector(count,3);
        end if;
    end process;
end behave;
```

(decoder38.vhd)3-8 译码器

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;

entity decoder38 is
port(code : in std_logic_vector(0 to 2);
        lan : out std_logic_vector(0 to 7));
end entity decoder38;

architecture behave of decoder38 is

begin
    process(code)
    begin

        for i in 0 to 7 loop
            if (code = conv_std_logic_vector(i,3)) then
                lan <= ( i => '1',others => '0');
            end if;
        end loop;

    end process;
end behave;
```

(Digital.vhd)顶层实体：4-7 译码器

```
library ieee;
use ieee.std_logic_1164.all;

entity digital is --47
port(Data : in std_logic_vector(0 to 3);
```

```
            sc : in std_logic_vector(0 to 2);
            latch : in std_logic;
            output: out std_logic_vector(0 to 6);
            clk : in std_logic;
            select1 : out std_logic_vector(0 to 7));

end entity;

architecture behave of digital is

component counter is
port(clk : in std_logic;
            Par_output : out std_logic_vector(0 to 2));
end component;

component latcher is
      port(Data : in std_logic_vector(0 to 3);
                  sc : in std_logic_vector(0 to 2);
                  latch : in std_logic;
                  scan : in std_logic_vector(0 to 2);
                  data_out: out std_logic_vector(0 to 3));
end component;

component decoder38 is
port(code : in std_logic_vector(0 to 2);
            lan : out std_logic_vector(0 to 7));
end component;

signal data_output : std_logic_vector(0 to 3);
signal temp_bus : std_logic_vector(0 to 2);

begin
      counter3: counter port map(clk,temp_bus);
      latch48 : latcher port map (Data,sc,latch,temp_bus,data_output);
      dec : decoder38 port map(temp_bus,select1);

      process(data_output)
      variable a0:std_logic_vector(0 to 6) := "0000001";
      variable a1:std_logic_vector(0 to 6) := "1001111";
      variable a2:std_logic_vector(0 to 6) := "0010010";
      variable a3:std_logic_vector(0 to 6) := "0000110";
      variable a4:std_logic_vector(0 to 6) := "1001100";
      variable a5:std_logic_vector(0 to 6) := "0100100";
      variable a6:std_logic_vector(0 to 6) := "0100000";
```

```
        variable a7:std_logic_vector(0 to 6) := "0001111";
        variable a8:std_logic_vector(0 to 6) := "0000000";
        variable a9:std_logic_vector(0 to 6) := "0000100";
        variable aA:std_logic_vector(0 to 6) := "0001000";
        variable aB:std_logic_vector(0 to 6) := "1100000";
        variable aC:std_logic_vector(0 to 6) := "0110001";
        variable aD:std_logic_vector(0 to 6) := "1000010";
        variable aE:std_logic_vector(0 to 6) := "0110000";
        variable aF:std_logic_vector(0 to 6) := "0111000";

        begin
        case data_output is
                when "0000" => output <= a0;
                when "0001" => output <= a1;
                when "0010" => output <= a2;
                when "0011" => output <= a3;
                when "0100" => output <= a4;
                when "0101" => output <= a5;
                when "0110" => output <= a6;
                when "0111" => output <= a7;
                when "1000" => output <= a8;
                when "1001" => output <= a9;
                when "1010" => output <= aA;
                when "1011" => output <= aB;
                when "1100" => output <= aC;
                when "1101" => output <= aD;
                when "1110" => output <= aE;
                when "1111" => output <= aF;
                when others => output <= "XXXXXXX";
        end case;
        end process;
end behave;
```

(divider.vhd)分频器

```
library ieee;
use ieee.std_logic_1164.all;

entity divider is
        port (clk : IN std_logic;
                        oclk : OUT std_logic);
end entity;

architecture behave of divider is
begin
```

```
    process(clk)
        variable num: integer range 0 to 31;
        variable temp:std_logic := '1';
        begin
        if (clk = '1' and clk'event) then
            if (num = 10) then
                num := 0; temp := not temp; oclk <= temp;
            else  num := num + 1;
            end if;
        end if;

    end process;
end behave;
```

(latcher.vhd)锁存器

```
library ieee;
use ieee.std_logic_1164.all;

entity latcher is
    port(Data : in std_logic_vector(0 to 3);
            sc : in std_logic_vector(0 to 2);
            latch : in std_logic;
            scan : in std_logic_vector(0 to 2);
            data_out: out std_logic_vector(0 to 3));
end entity latcher;

architecture behave of latcher is

type lch_array is array(0 to 7) of std_logic_vector(0 to 3);
signal lch : lch_array;
signal temp: std_logic := '1';

begin

        lch(0) <= Data when (latch = '1' and sc = "000");
        lch(1) <= Data when (latch = '1' and sc = "001");
        lch(2) <= Data when (latch = '1' and sc = "010");
        lch(3) <= Data when (latch = '1' and sc = "011");
        lch(4) <= Data when (latch = '1' and sc = "100");
        lch(5) <= Data when (latch = '1' and sc = "101");
        lch(6) <= Data when (latch = '1' and sc = "110");
        lch(7) <= Data when (latch = '1' and sc = "111");
```

```
    process(scan)
    begin
        case scan is
            when "000" => data_out <= lch(0);
            when "001" => data_out <= lch(1);
            when "010" => data_out <= lch(2);
            when "011" => data_out <= lch(3);
            when "100" => data_out <= lch(4);
            when "101" => data_out <= lch(5);
            when "110" => data_out <= lch(6);
            when "111" => data_out <= lch(7);
            when others => data_out <= "XXXX";
        end case;
    end process;

end behave;
```

(digital_tb.vhd)test bench

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

entity digital_tb is
end entity;

architecture bh of digital_tb is

component Digital is
port(Data : in std_logic_vector(0 to 3);
        sc : in std_logic_vector(0 to 2);
        latch : in std_logic;
        output: out std_logic_vector(0 to 6);
        clk : in std_logic);
end component;

signal data : std_logic_vector(0 to 3);
signal sc : std_logic_vector(0 to 2);
signal latch : std_logic;
signal output : std_logic_vector(0 to 6);
signal clk : std_logic;

begin
```

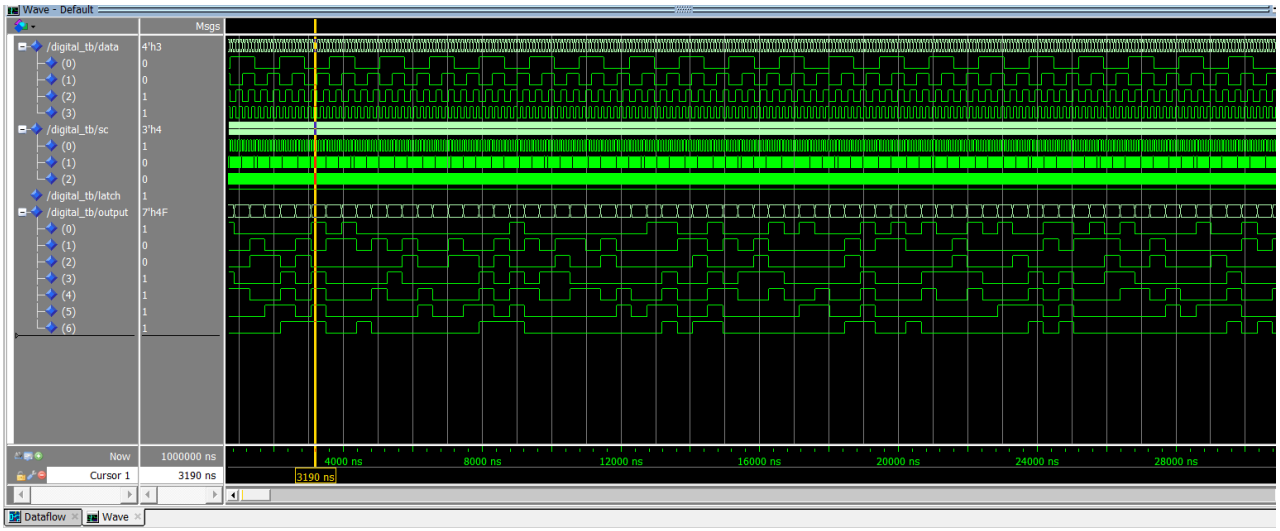
```
    u1 : Digital port map (data,sc,latch,output,clk);

    process
    begin
    clk <= '1';
    wait for 10 ns;
    clk <= '0';
    wait for 10 ns;
    end process;

    process
    begin
        latch <= '1';
        for i in 0 to 15 loop
        data <= CONV_STD_LOGIC_VECTOR(i,4);
            for j in 0 to 7 loop
            sc <= conv_STD_LOGIC_VECTOR(j,3);
            wait for 10 ns;
            end loop;
        wait for 10 ns;
        end loop;
    end process;

end bh;
```

仿真结果记录：



FPGA 验证结果记录：

Latch 所映射的开关打开到 1 时，将数据信息以及锁存器片选信息锁存进入锁存器，从而在扫描时将在对应的片选数码管上对数据进行更新。

实验总结：

1. 此次实验引入使用 arith 库，主要函数为 conv_std_logic_vector，将 integer 类型转化为 std_logic_vector 类型，从而使得很多地方能够使用 for 循环时的代码大为简化，使代码复杂度降低，但也由于未对比电路复杂度的变化，因此主要使用实在 test bench 中，从而减小对于实体电路的影响。
2. 此次实验引入 ppt 中未提及的 others 用法，out <= (I => ' 1' , others => '0'),类似于位操作，将向量的 i 号位赋值 1，同时将其他位赋值 0。这种方法虽然能够避免使用 case 一类的语句从而简化代码量，但是实际上却是使得代码可读性降低，同时使电路情况变得极其复杂。

   因此，从这种情况看来，这种写法最方便的是快速开发，在实际应用时非常不推崇此种代码写法。
3. 已较熟练的掌握 modelsim 来进行 debug，收获颇多，并且编写代码时间大大降低。