实验三：七段数码管

一. 实验内容

采用动态扫描的方法，使八个数码管分别显示不同的数字。

二. 设计分析

编程思路：

数码管动态扫描的思想是将时间分成很小的间隔，在每个时间间隔里只允许一个数码管显示，但8个数码管按照时间间隔轮流循环显示。当一个数码管点亮的频率大于每秒50次的时候，人眼就无法分辨其亮灭的状态，看到数码管一直是亮的，这样就达到了每个数码管分别显示不同数字的效果。

按照动态扫描的思想，首先需要有8个锁存器来保存要要显示的数字，每个锁存器存储4bits数据，锁存器中的数据可以通过拨码开关进行修改，由拨码开关来确定修改哪一个锁存器、更新的值是多少。

将完整结构拆分为序列发生器1/2、序列检测器、分频器、选择器几个单元模块分别实现。

序列检测器：采用Mealy型有限状态机模型，输入序列作为条件，转移到特定状态代表检测到完整序列，输出信号。

序列产生器：采用Moore型有限状态机模型，产生对应序列。

三. VHDL源代码

顶层实体：(FPGA_EXP3_dxh.vhd)

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

entity FPGA_EXP3_dxh is
    port(sel: in std_logic_vector (2 downto 0);
    datain: in std_logic_vector(3 downto 0);
    en: in std_logic;
    clk: in std_logic;
    LED_cs: out std_logic_vector(7 downto 0);
    segment: out std_logic_vector(7 downto 0)
    );
end FPGA_EXP3_dxh;

architecture arch_FPGA_EXP3_dxh of FPGA_EXP3_dxh is
    component LED8
        port(clk: in std_logic;
        d0: in std_logic_vector(3 downto 0);
        d1: in std_logic_vector(3 downto 0);
        d2: in std_logic_vector(3 downto 0);
        d3: in std_logic_vector(3 downto 0);
```

```
        d4: in std_logic_vector(3 downto 0);
        d5: in std_logic_vector(3 downto 0);
        d6: in std_logic_vector(3 downto 0);
        d7: in std_logic_vector(3 downto 0);
        segment: out std_logic_vector(7 downto 0);
        LED_cs: out std_logic_vector(7 downto 0)
        );
    end component;

    component register0_7
        port (sel: in std_logic_vector(2 downto 0);
        datain: in std_logic_vector(3 downto 0);
        en: in std_logic;
        d0, d1, d2, d3, d4, d5, d6, d7: out std_logic_vector(3 downto 0)
        );
    end component;

    signal d0,d1,d2,d3,d4,d5,d6,d7: std_logic_vector(3 downto 0);
begin
    u1: register0_7 port map (sel, datain, en, d0, d1, d2, d3, d4, d5, d6,
d7);
    u2: LED8 port map (clk, d0, d1, d2, d3, d4, d5, d6, d7, segment, LED_cs);
end arch_FPGA_EXP3_dxh;
```

register0_7.vhd:

数据输入部分，包含3-8译码器，8个4位锁存器

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

entity register0_7 is
    port (sel: in std_logic_vector(2 downto 0);
    datain: in std_logic_vector(3 downto 0);
    en: in std_logic;
    d0, d1, d2, d3, d4, d5, d6, d7: out std_logic_vector(3 downto 0)
    );
end register0_7;

architecture arch_register0_7 of register0_7 is
    component decoder38
        port(
        A: in std_logic_vector (2 downto 0);
        B:out std_logic_vector (7 downto 0)
        );
    end component;

    component latch4
        port(
        D: in std_logic_vector(3 downto 0);
        en,cs: in std_logic;
        Q: out std_logic_vector(3 downto 0));
    end component;

    signal tmp_cs: std_logic_vector(7 downto 0);
```

```
begin
u0: latch4 port map (datain, en, tmp_cs(0), d0);
u1: latch4 port map (datain, en, tmp_cs(1), d1);
u2: latch4 port map (datain, en, tmp_cs(2), d2);
u3: latch4 port map (datain, en, tmp_cs(3), d3);
u4: latch4 port map (datain, en, tmp_cs(4), d4);
u5: latch4 port map (datain, en, tmp_cs(5), d5);
u6: latch4 port map (datain, en, tmp_cs(6), d6);
u7: latch4 port map (datain, en, tmp_cs(7), d7);
decode: decoder38 port map (sel, tmp_cs);
end arch_register0_7;
```

3-8译码器：（decoder38.vhd)

译码器的 输出信号为八个4位锁存器的片选信号

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

entity decoder38 is
    port(
    A: in std_logic_vector (2 downto 0);
    B:out std_logic_vector (7 downto 0));
end decoder38;

architecture arch_decoder38 of decoder38 is
begin
    process(A)
    begin
        case A is
            when "000" => B <= "00000001";
            when "001" => B <= "00000010";
            when "010" => B <= "00000100";
            when "011" => B <= "00001000";
            when "100" => B <= "00010000";
            when "101" => B <= "00100000";
            when "110" => B <= "01000000";
            when "111" => B <= "10000000";
            when others => null;
        end case;
    end process;
end arch_decoder38;
```

4位锁存器：（latch4.vhd）

为使结构中所包含的两个3-8译码器相同，此设计中锁存器的片选信号为高电平有效，与作业word文档中的低电平有效不同。

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

entity latch4 is
```

```
    port(
    D: in std_logic_vector(3 downto 0);
    en,cs: in std_logic;
    Q: out std_logic_vector(3 downto 0));
end latch4;

architecture arch_latch4 of latch4 is
begin
    process(cs, en)
    begin
        if (cs='1' and en='1') then
            Q <= D;
        end if;
    end process;
end arch_latch4;
:
```

LED8.vhd:

整体结构中的控制电路部分，包括分频器、8进制计数器、3-8译码器、8选1选择器、4-7译码器

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

entity LED8 is
    port(clk: in std_logic;
    d0: in std_logic_vector(3 downto 0);
    d1: in std_logic_vector(3 downto 0);
    d2: in std_logic_vector(3 downto 0);
    d3: in std_logic_vector(3 downto 0);
    d4: in std_logic_vector(3 downto 0);
    d5: in std_logic_vector(3 downto 0);
    d6: in std_logic_vector(3 downto 0);
    d7: in std_logic_vector(3 downto 0);
    segment: out std_logic_vector(7 downto 0);
    num: out std_logic_vector(2 downto 0);
    LED_cs: out std_logic_vector(7 downto 0)
    );
end LED8;

architecture arch_LED8 of LED8 is
    component Div50MHz
        port(clk: in std_logic;
        clkout: out std_logic
        );
    end component;

    component Counter8
        port (clk: in std_logic;
        q: out std_logic_vector (2 downto 0)
        );
    end component;

    component decoder38
```

```
        port(
        A: in std_logic_vector (2 downto 0);
        B:out std_logic_vector (7 downto 0)
        );
    end component;

    component selector81
        port(d0: in std_logic_vector(3 downto 0);
        d1: in std_logic_vector(3 downto 0);
        d2: in std_logic_vector(3 downto 0);
        d3: in std_logic_vector(3 downto 0);
        d4: in std_logic_vector(3 downto 0);
        d5: in std_logic_vector(3 downto 0);
        d6: in std_logic_vector(3 downto 0);
        d7: in std_logic_vector(3 downto 0);
        s: in std_logic_vector(2 downto 0);
        q: out std_logic_vector(3 downto 0)
        );
    end component;

    component decoder47
        port(
        A: in std_logic_vector (3 downto 0);
        B: out std_logic_vector (7 downto 0));
    end component;

    signal clk_tmp: std_logic;
    signal num_tmp: std_logic_vector(2 downto 0);
    signal data_tmp: std_logic_vector(3 downto 0);
begin
    u1: Div50MHz port map (clk, clk_tmp);
    u2: Counter8 port map (clk_tmp, num_tmp);
    u3: decoder38 port map (num_tmp, LED_cs);
    u4: selector81 port map (d0, d1, d2, d3, d4, d5, d6, d7, num_tmp,
data_tmp);
    u5: decoder47 port map (data_tmp, segment);
    num <= num_tmp;
end arch_LED8;
```

4-7译码器：（decoder47.vhd）

此处，将输出的段控制信号定义为8个，即包括dp段。当输入数据大于9时，其他段都不亮，只有dp段亮，以示区别。所以实际上，我把它写成了4-8译码器。

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

entity decoder47 is
    port(
    A: in std_logic_vector (3 downto 0);
    B: out std_logic_vector (7 downto 0));
end decoder47;

architecture arch_decoder47 of decoder47 is

```

```
begin
    with A select
        B <= "00000011" when "0000",
             "10011111" when "0001",
             "00100101" when "0010",
             "00001101" when "0011",
             "10011001" when "0100",
             "01001001" when "0101",
             "01000001" when "0110",
             "00011111" when "0111",
             "00000001" when "1000",
             "00001001" when "1001",
             "11111110" when others;
end arch_decoder47;
```

8进制计数器：（Counter8.vhd）

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

entity Counter8 is
    port (clk: in std_logic;
    q: out std_logic_vector (2 downto 0)
    );
end Counter8;

architecture arch_Counter8 of Counter8 is
    signal tmp_q: std_logic_vector(2 downto 0) := "000";
begin
    q <= tmp_q;
    process(clk)
    begin
        if (clk'event and clk='1') then
            if (tmp_q="111") then
                tmp_q <= "000";
            else
                tmp_q <= tmp_q + 1;
            end if;
        end if;
    end process;
end arch_Counter8;
```

分频器：（Div50MHz.vhd）

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

entity Div50MHz is
    port(clk: in std_logic;
    clkout: out std_logic
    );
end Div50MHz;
```

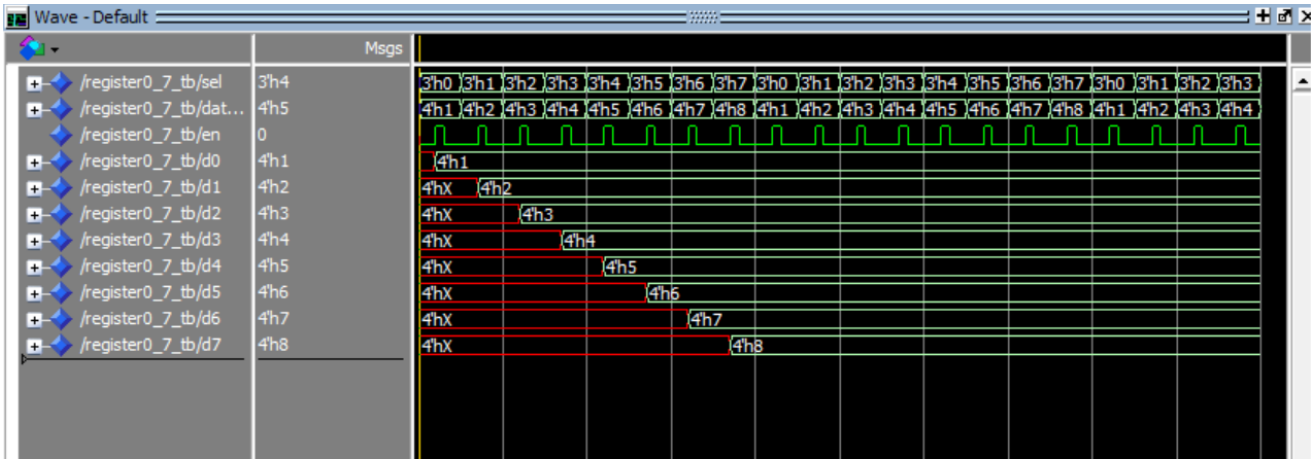
```
architecture arch_Div50MHz of Div50MHz is
    signal cnt: integer range 0 to 99999;
    signal tmp_clk: std_logic;
begin
    process(clk)
    begin
        if(clk'event and clk = '1') then
            if (cnt = 99999) then
                cnt <= 0;
                tmp_clk <= NOT tmp_clk;
            else
                cnt <= cnt + 1;
            end if;
        end if;
    end process;
    clkout <= tmp_clk;
end arch_Div50MHz;
```

8选1选择器：

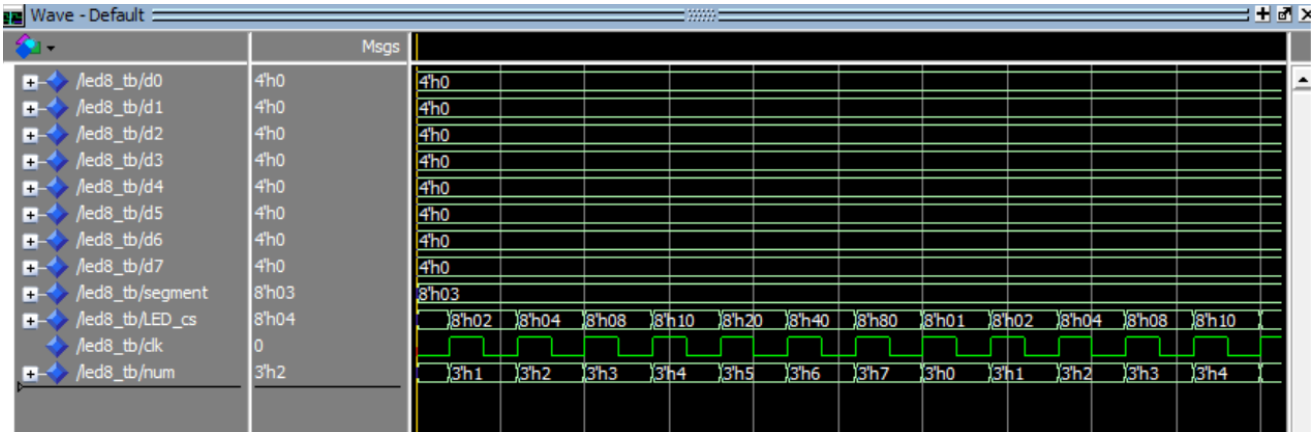
```
library ieee;
use ieee.std_logic_1164.all;

entity selector81 is
    port(d0: in std_logic_vector(3 downto 0);
    d1: in std_logic_vector(3 downto 0);
    d2: in std_logic_vector(3 downto 0);
    d3: in std_logic_vector(3 downto 0);
    d4: in std_logic_vector(3 downto 0);
    d5: in std_logic_vector(3 downto 0);
    d6: in std_logic_vector(3 downto 0);
    d7: in std_logic_vector(3 downto 0);
    s: in std_logic_vector(2 downto 0);
    q: out std_logic_vector(3 downto 0)
    );
end selector81;

architecture arch_selector81 of selector81 is
begin
    with s select
    q <= d0 when "000",
          d1 when "001",
         d2 when "010",
          d3 when "011",
          d4 when "100",
          d5 when "101",
          d6 when "110",
          d7 when "111",
          "XXXX" when others;
end arch_selector81;
```

四. 仿真结果

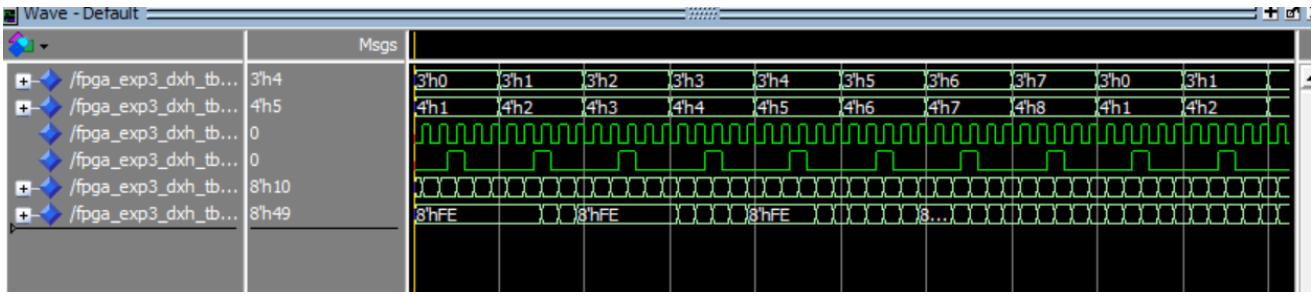
register0_7仿真结果：



LED8仿真结果：（d0~d7均置为“0000”）



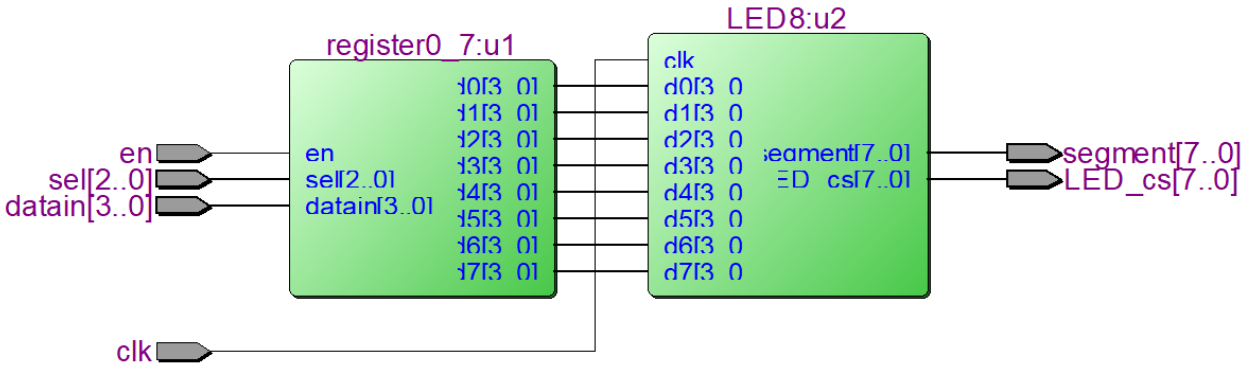
整体仿真结果：



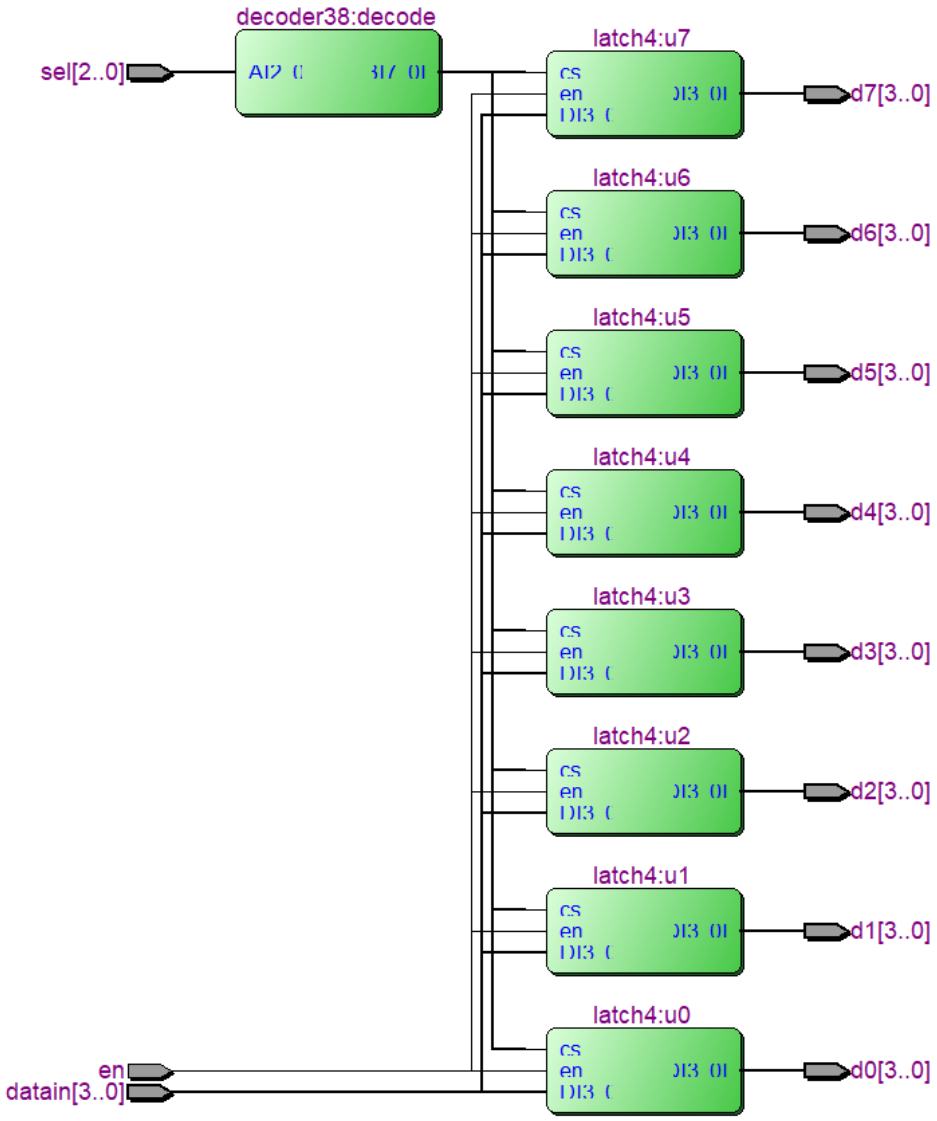
五. 中间结果

1.RTL电路图

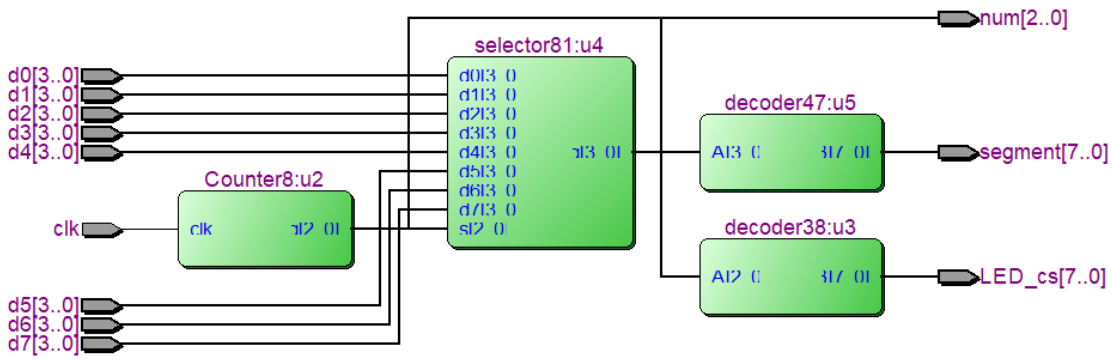
整体结构：


register0_7:u1
LED8:u2
en
sel[2..0]
datain[3..0]
clk
segment[7..0]
LED_cs[7..0]

register0_7:



LED8结构图：



2.资源占用信息

Flow Summary	
Flow Status	Successful - Fri Oct 14 22:13:31 2022
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	FPGA_EXP3_dxh
Top-level Entity Name	FPGA_EXP3_dxh
Family	Cyclone V
Device	5CEFA2F23C8
Timing Models	Final
Logic utilization (in ALMs)	53 / 9,430 (< 1 %)
Total registers	23
Total pins	25 / 224 (11 %)
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 %)

六. 硬件验证

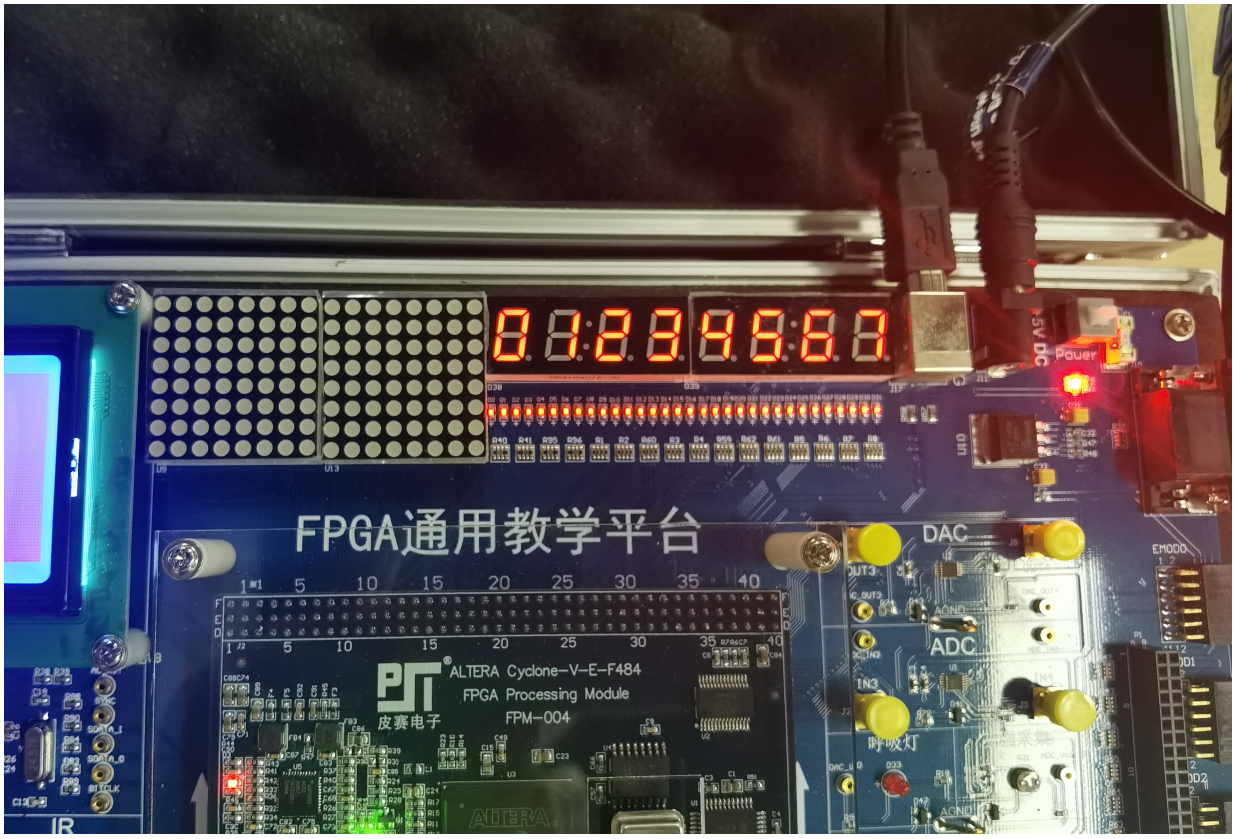
1. 管脚锁定情况:

端口	管脚
clk	PIN_W16
datain[3]	PIN_Y11
datain[2]	PIN_R11
datain[1]	PIN_U12
datain[0]	PIN_V13
en	PIN_T19
LED_cs[7]	PIN_E12
LED_cs[6]	PIN_D13
LED_cs[5]	PIN_A13
LED_cs[4]	PIN_C11
LED_cs[3]	PIN_F13
LED_cs[2]	PIN_E14
LED_cs[1]	PIN_B15

端口	管脚
LED_cs[0]	PIN_B16
segment[7]	PIN_K19
segment[6]	PIN_H18
segment[5]	PIN_K20
segment[4]	PIN_M18
segment[3]	PIN_E16
segment[2]	PIN_G13
segment[1]	PIN_G17
segment[0]	PIN_H16
sel[2]	PIN_Y20
sel[1]	PIN_W22
sel[0]	PIN_Y22

先将使能端en置零，波动拨码开关DIP12~DIP13，输入sel信号选定锁存器，然后使用DIP0~DIP3输入想要显示的数字，推动DIP15使en为高电平，输入数字显示在对应数码管上，重复上述操作，则可实现8个数码管分别显示8个不同的数字。

2. 实验效果图



七. 实验总结

在尝试对LED8模块进行仿真时，发现输出无信号。经过排查，推测是计数器环节出现问题。但是程序设计代码的逻辑并没有出错。

解决方法：在程序设计代码中，将用于计数的信号tmp_q赋初值，则可进行正常仿真。

在解码器程序中，使用了case语句。由于输入信号是std_logic_vector类型，我仅仅穷举了0/1的所有可能组合，而未考虑其他可能的取值。在综合和编译时，不会报错。但会造成仿真过程报错，提示case语句未覆盖所有取值可能。

解决方法：加一条语句:when others ->null