同濟大學

电子信息综合专业实验报告

学院(系)	电子与信息工程学院
专　　业	电子信息工程
年　　级	2020 级
姓　　名	黄锴康
学　　号	1951706
实验课程	电子信息工程专业综合实验
任课教师	林　林
项目名称	基于 VHDL 的地铁售票机
日期	2023.7.10 - 2023.7.13

目录

一、实验内容 1
二、实验步骤及结果演示 1
2.1 总体设计思路 1
2.2 实验步骤 2
三、遇到的主要困难及解决对策 11
四、感想和体会 11

装

订

线

一、实验内容

设计地铁售票机，有两元、三元两种地铁票。只能用五角、一元两种硬币购票，投入硬币，数码管显示投入硬币金额，在投入一定数量的硬币后，有确认投币按钮，若金额数大于等于所选票价一致，给出车票，并找零。否则，退还硬币。

需要完成各个模块的 VHDL 描述及仿真，并完成该工程在硬件平台 NEXYS4 上演示。

二、实验步骤及结果演示

2.1 总体设计思路

根据以上设计要求，将项目根据输入和输出，分为以下模块：

- 输入模块：

1. 硬币接收模块：负责接收用户投入的硬币，并记录投入的金额。
2. 确认投币模块：接收用户的确认投币信号，用于确认用户已经投入了一定数量的硬币。
3. 票价选择模块：接收用户选择的地铁票票价。

- 输出模块：

1. 数码管显示模块：根据投入的硬币金额，在数码管上显示金额，同时也负责找零金额的显示。
2. 车票输出模块：根据投币金额和选择的票价进行判断，给出车票并找零。
3. 状态指示灯模块：控制 LED 指示灯的状态，例如显示购票成功、找零等状态。

同时为了不同模块的协调工作，大部分模块之间采用统一的时钟控制，而数位管显示模块需要相对低频的时钟，以满足数位管的显示特性。根据信号流，确定设计原理图如图 2.1 所示，

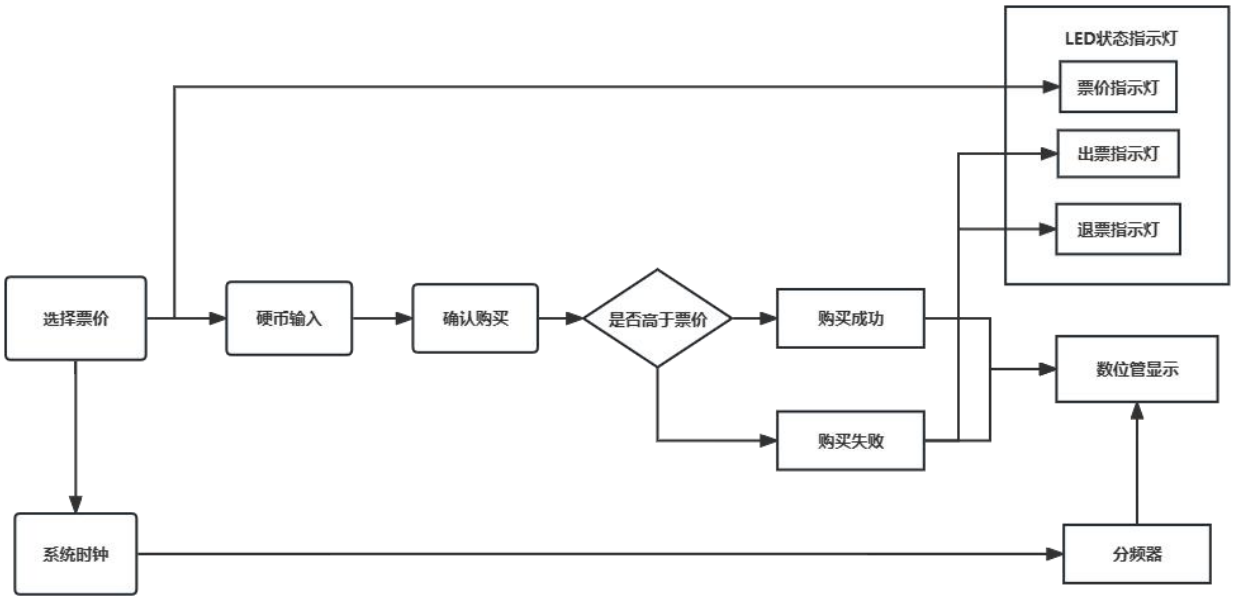

图 2.1 设计原理图

本项目中状态机主要用于用于记录投入硬币的数量，硬币的输入使得状态机进入不同的状态，确定输入可以直接跳转至票价判断，根据判断结果，进入出票状态和退钱状态，状态机原理

图如图 2.2 所示，

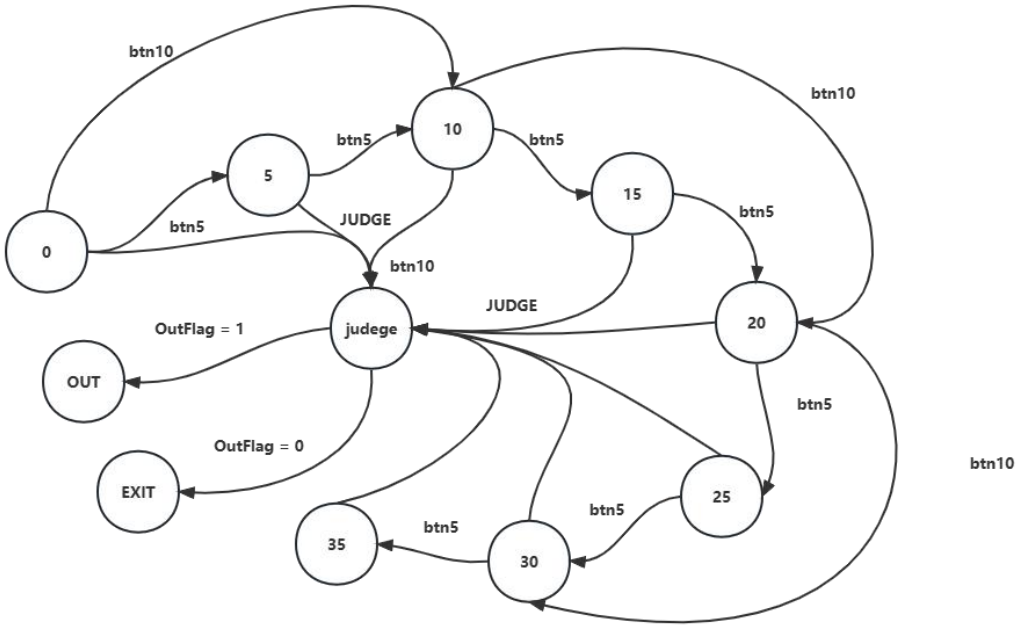

图 2.2 状态机原理图

2.2 实验步骤

- 画出顶层原理图

  根据以上原理图，以及先前的工程，顶层原理图如图 2.3 所示，

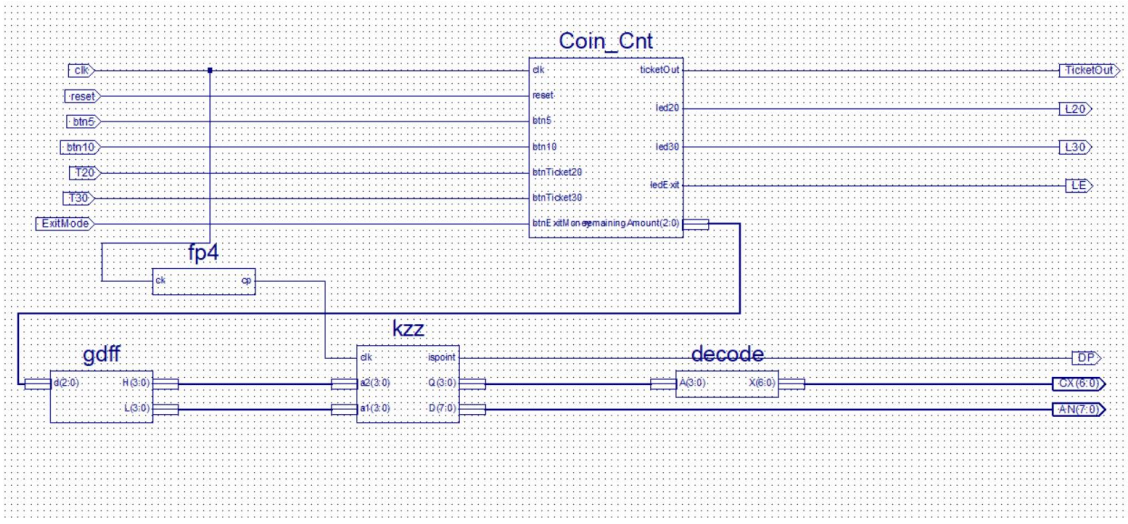

图 2.3 顶层原理图

在本项目中，用拨片开关作为输入。reset 作为系统重置，为高电平使能；T20、T30 和 ExitMode 分别为 2 元票、3 元票和确定购买输入，三者均为高电平使能；btn5 和 btn10 是硬币输入，为上升沿触发。

在顶层原理图中 gdf、fp4、kzz 以及 decode 模块都是数码管显示相关模块，由于前置课程中已经进行详细介绍，在此用数码管显示模块统称。

- 各个模块设计原理

1. 数位管显示模块

fp4 为分频器，在此为 100000 分频器，即 1ms 时钟输出。

Fp4

1ms 分频器

```
entity fp4 is
    Port ( ck : in    STD_LOGIC;
           cp : out    STD_LOGIC);
end fp4;

architecture Behavioral of fp4 is
begin
process(ck)
variable a : integer range 0 to 100000 ;
   begin
  if (ck'event and ck='1')    then
        if a=100000 -1 then
            a:=0;
        else
            a:=a+1;
        end if;
        if a< 100000 /2 then
            CP<='1';
       else
            CP<='0';
            end if;
    end if;
 end process;

end Behavioral;
```

kz 的功能是实现两位 10 进制数码的轮流输出。该模块具有两个输入向量和两个输出向量。输入向量 Q(3:0)用于轮流输出个位数和十位数的数码值，输出向量 D(7:0)用于控制对应的 8 位数码显示器的位选信号，即指示在数码显示器上显示的位置。通过控制输入向量，kz 模块可以实现两位数码值的轮流显示。注意的是在高位时，需要点亮小数点，因此要用 variable 计数此时输出数字的位数。

Kz

二选一模块

```
entity kzz is
    Port ( clk : in    STD_LOGIC;
           a2 : in    STD_LOGIC_VECTOR (3 downto 0);
           a1 : in    STD_LOGIC_VECTOR (3 downto 0);
          ispoint: out STD_LOGIC;
           Q : out    STD_LOGIC_VECTOR (3 downto 0);
           D : out    STD_LOGIC_VECTOR (7 downto 0));
end kzz;

architecture Behavioral of kzz is
```

```
begin
process (clk, a2, a1)
variable b : integer range 0 to 2 ;
begin
      if clk='1' and clk'event then
            if b= 1 then
                  b:=0;
                  Q<=a2;
                  ispoint<='0';

                  D<="11111101";
             else
                  b:=b+1;
                  Q<=a1;
                  ispoint<='1';
                  D<="11111110";
            end if;
            --D<="11111111";
      end if;
end process;

end Behavioral;
```

decode 模块的功能是将输入的二进制编码转换为对应的七段码输出，用于驱动七段数码管显示特定的数字或字符(（a、b、c、d、e、f、g)。本开发板是共阳极显示管，也就是说控制信号为低电平时，数码管被点亮。

Decode

解码器

```
entity decode is
      Port ( A : in    STD_LOGIC_VECTOR (3 downto 0);
                X : out    STD_LOGIC_VECTOR (6 downto 0));
end decode;

architecture Behavioral of decode is

begin
PROCESS(A)
BEGIN
CASE A is
    when "0000" => X <="0000001";
when "0001" => X <="1001111";
when "0010" => X <="0010010";
when "0011" => X <="0000110";
when "0100" => X <="1001100";
when "0101" => X <="0100100";
when "0110" => X <="0100000";
when "0111" => X <="0001111";
when "1000" => X <="0000000";
when "1001" => X <="0000100";
      when others => X <="0000000";
    end case;
end process;
```

```
end Behavioral;
```

gdf 的功能是将 16 进制码转换为两位 10 进制码。具体来说，在本项目中该模块接收一个 4 位的 16 进制输入信号，将其转换为两个 4 位的 10 进制输出信号。在此转换表较为特殊，每个二进制数对应 10 进制的 0.5、1.0 以此类推。

Gdf

数据转换器

```
entity gdff is
    Port ( d : in    STD_LOGIC_VECTOR (2 downto 0);
           H : out   STD_LOGIC_VECTOR (3 downto 0);
           L : out   STD_LOGIC_VECTOR (3 downto 0));
end gdff;

architecture Behavioral of gdff is

begin
--variable temp: INTEGER RANGE 0 TO 15;
PROCESS(d)
BEGIN
CASE d is
   when "000" => L<="0000"; H <="0000";
when "001" => L<="0101"; H <="0000";
when "010" => L<="0000"; H <="0001";
when "011" => L<="0101"; H <="0001";
when "100" => L<="0000"; H <="0010";
when "101" => L<="0101"; H <="0010";
when "110" => L<="0000"; H <="0011";
when others => L<="0101"; H <="0011";
   end case;
end process;
```

2. Coin_Cnt 模块

在该模块中包括 btn5（5 元按钮）、btn10（10 元按钮）、btnTicket20（选择 20 元车票按钮）、btnTicket30（选择 30 元车票按钮）、btnExitMoney（退出钱按钮）等输入信号。根据输入的按钮状态，更新相应的状态和变量。

使用两个变量 totalAmount5 和 totalAmount10 来分别记录投入的 5 元硬币和 10 元硬币的数量。根据 btn5 和 btn10 的状态变化，在时钟上升沿触发时更新硬币数量。根据 btnTicket20 和 btnTicket30 的状态变化，确定车票的价格，并相应地控制 LED 指示灯 led20 和 led30 的状态。

在时钟上升沿触发时，根据投入的硬币数量和票价进行判断，如果满足购票条件，设置 ticketOut 为高电平，并计算剩余金额 remainingAmountInt。如果投入金额不足以购买车票，设置 ledExit 为高电平，表示退还硬币。根据 remainingAmountInt 的值，将剩余金额转换为二进制形式，并赋值给输出信号 remainingAmount。

这样，通过输入模块和输出模块的交互，以及内部的状态控制和计算，实现了地铁售票机的功能。

Coin_Cnt

硬币计数模块

```
entity Coin_Cnt is
    port (
        clk : in std_logic;                    -- 时钟信号
        reset : in std_logic;                  -- 复位信号
        btn5 : in std_logic;                   -- 5 元按钮
        btn10 : in std_logic;                  -- 10 元按钮
```

```
        btnTicket20 : in std_logic;                                    -- 选择 20 元车票按钮
        btnTicket30 : in std_logic;                                    -- 选择 30 元车票按钮
        btnExitMoney : in std_logic;                                   -- 退出钱按钮

        led20 : out std_logic;                                          -- 20 元发光二极管
        led30 : out std_logic;                                          -- 30 元发光二极管
        ledExit : out std_logic;                                       -- 退出钱发光二极管
        ticketOut : out std_logic;                                     -- 是否给票
        remainingAmount : out std_logic_vector(2 downto 0)             -- 剩余金额
    );
end Coin_Cnt;

architecture Behavioral of Coin_Cnt is
    -- 内部信号
    signal totalAmount : integer range 0 to 100;                       -- 总金额
  signal totalAmount5 : integer range 0 to 100;
  signal totalAmount10 : integer range 0 to 100;
    signal TicketDone : integer range 0 to 1;
    signal ticketPrice : integer range 0 to 100;                       -- 车票价格
    signal remainingAmountInt : integer range 0 to 100;                -- 剩余金额（整数部
分）
begin

process(btn5)
    begin
      if reset ='1' or TicketDone =1 then
            totalAmount5 <= 0;
      elsif rising_edge(btn5) then
            totalAmount5 <= totalAmount5 + 5;
      end if;
    end process;

process(btn10)
    begin
      if reset ='1' or TicketDone =1 then
            totalAmount10 <= 0;
      elsif rising_edge(btn10) then
            totalAmount10 <= totalAmount10 + 10;
      end if;
    end process;

process(clk)
begin
    if rising_edge(clk)then
    ------票价控制模块,控制 LED 灯
        if reset ='1'    then
            ticketPrice <= 0;
            led20 <= '0';
            led30 <= '0';
        else
            if btnTicket20 = '1' then
                ticketPrice <= 20;
                led20 <= '1';
```

```
                led30 <= '0';
            elsif btnTicket30 = '1' then
                ticketPrice <= 30;
                led20 <= '0';
                led30 <= '1';

            else
                ticketPrice <= 0;
                led20 <= '0';
                led30 <= '0';
            end if;
        end if;
    end if;
end process;

process(clk)
begin
    if reset = '1'   then
            ------ 在复位时将总金额和车票价格归零
            TicketDone <= 0;
            ticketOut <= '0';
            remainingAmountInt<=0;
            totalAmount <=0;
            ledExit <='0';
    else
        if rising_edge(clk) then
            -------根据模式控制段式数位管。首先确定是否退出，
            totalAmount <= totalAmount5 + totalAmount10;
            if btnExitMoney = '1' then
                --TicketDone <= 1;
                if ticketPrice = 0 then
                    remainingAmountInt <= totalAmount;
                elsif totalAmount >= ticketPrice then
                    ledExit <='0';
                    remainingAmountInt <= totalAmount - ticketPrice;
                    ticketOut <= '1';
                else
                    ledExit <='1';
                    remainingAmountInt <= totalAmount;
                    ticketOut <= '0';
                end if;
            else
                TicketDone <= 0;
                ledExit <='0';
                remainingAmountInt <= totalAmount;
                ticketOut <= '0';
            end if;
            end if;
        end if;
end process;

    -- 转换剩余金额的整数部分到二进制向量
    process(remainingAmountInt)
  begin
```

```
        if reset = '1'    then
        remainingAmount <= "000";
    else

            case remainingAmountInt    is
                when 0 =>
                    remainingAmount <= "000";
                when 5 =>
                    remainingAmount <= "001";
                when 10 =>
                    remainingAmount <= "010";
                when 15 =>
                    remainingAmount <= "011";
                when 20 =>
                    remainingAmount <= "100";
                when 25 =>
                    remainingAmount <= "101";
                when 30 =>
                    remainingAmount <= "110";
                when others =>
                    remainingAmount <= "111";       -- 默认情况下，输出为 0
                end case;
     end if;
        end process;

    end Behavioral;
```

通过 testbench 对 coin_cnt 进行测试，依次投入 5 分硬币、1 元硬币，购买 2 元车票，观察该模块的输出情况，编写代码如下所示，

CC_TB

Coin_Cnt 测试代码

```
ENTITY CC_TB IS
END CC_TB;


ARCHITECTURE behavior OF CC_TB IS

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

    COMPONENT Coin_Cnt
    PORT(
          clk : IN    std_logic;
          reset : IN    std_logic;
          btn5 : IN    std_logic;
          btn10 : IN    std_logic;
          btnTicket20 : IN    std_logic;
          btnTicket30 : IN    std_logic;
          btnExitMoney : IN    std_logic;
          ticketOut : OUT    std_logic;
          led20 : OUT    std_logic;
          led30 : OUT    std_logic;
          ledExit : OUT    std_logic;
          remainingAmount : OUT    std_logic_vector(2 downto 0)
        );
    END COMPONENT;
```

```
    --Inputs
    signal clk : std_logic := '0';
    signal reset : std_logic := '0';
    signal btn5 : std_logic := '0';
    signal btn10 : std_logic := '0';
    signal btnTicket20 : std_logic := '0';
    signal btnTicket30 : std_logic := '0';
    signal btnExitMoney : std_logic := '0';

     --Outputs
    signal ticketOut : std_logic;
    signal led20 : std_logic;
    signal led30 : std_logic;
    signal ledExit : std_logic;
    signal remainingAmount : std_logic_vector(2 downto 0);

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

BEGIN

-- Instantiate the Unit Under Test (UUT)
    uut: Coin_Cnt PORT MAP (
            clk => clk,
            reset => reset,
            btn5 => btn5,
            btn10 => btn10,
            btnTicket20 => btnTicket20,
            btnTicket30 => btnTicket30,
            btnExitMoney => btnExitMoney,
            ticketOut => ticketOut,
            led20 => led20,
            led30 => led30,
            ledExit => ledExit,
            remainingAmount => remainingAmount
          );

    -- Clock process definitions
    clk_process :process
    begin
      clk <= '1';
      wait for 5 ns;
      clk <= '0';
      wait for 5 ns;
    end process;


    -- Stimulus process
    stim_proc: process
    begin

      reset <= '1';
        wait for 20 ns;
```

```
        reset <='0';
        wait for 20 ns;
        --购买 2.0 元车票，投币 0.5， 1， 0.5， 1， 0.5， 找零 0.5
        btnTicket20 <= '1';
        wait for 20 ns;
        btn5 <= '1';
        wait for 20 ns;
        btn5 <= '0';
        wait for 20 ns;
          btn10 <= '1';
        wait for 20 ns;
        btn10 <= '0';
        wait for 20 ns;
        btn5 <= '1';
        wait for 20 ns;
        btn5 <= '0';
        wait for 20 ns;
          btn10 <= '1';
        wait for 20 ns;
        btn10 <= '0';
        wait for 20 ns;
        btn5 <= '1';
        wait for 20 ns;
        btn5 <= '0';
        wait for 20 ns;
        btnExitMoney <= '1';
        wait for 20 ns;
          wait;
      end process;

  END;
```

装

订

线

仿真波形结果如图 2.4 所示，

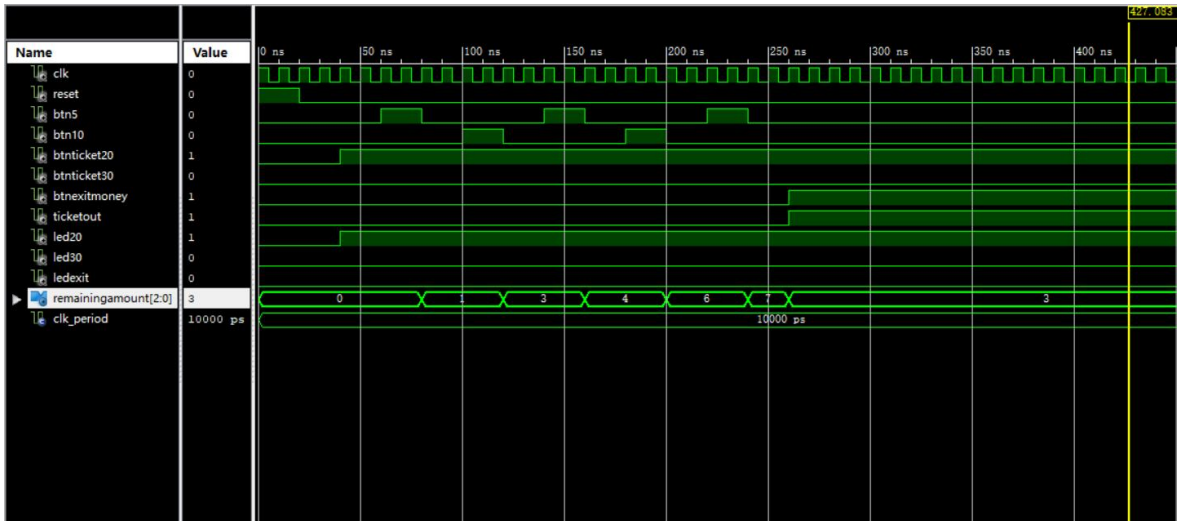

图 2.4 jsq16 仿真波形

三、遇到的主要困难及解决对策

- 状态机设计

地铁售票机的功能涉及多个状态和状态转换，因此需要合理设计状态机的状态和状态转换条件。在设计初期，存在逻辑混乱的问题，需要仔细思考逻辑以及状态转移图。

- 输入信号处理

根据输入按钮的状态变化来触发相应的操作，如投币和选择票价等。我需要处理按钮的抖动问题，确保正确地检测按钮的按下和松开，并在适当的时机更新状态和变量，同时为了防止误触，采用拨片开关触发。

- 时序逻辑与同步

在处理时钟信号和时序逻辑时，需要特别注意同步问题和时序约束。确保在正确的时钟边沿进行操作，并遵守时序要求，以避免出现不可预测的错误或功能故障。尤其是 variable 类型和 signal 类型两者数据更新的逻辑存在差异，通过这次设计，我对该类型理解更深。

四、感想和体会

- 设计思路的重要性

在开始设计之前，充分思考和规划整个系统的设计思路非常重要。需要明确功能需求、输入输出信号以及各个模块之间的关系，这有助于建立清晰的设计框架和逻辑结构。

- 状态机的应用

地铁售票机的功能包含多个状态和状态转换，使用状态机是合适的设计方法。通过定义不同的状态以及状态之间的转换条件，可以更好地组织和控制系统的行为。

- 时序约束和同步问题

在设计中，需要特别注意时序约束和同步问题。合理设计时钟边沿的使用，并遵守时序约束，可以避免时序相关的问题和功能故障。

在整个项目中，我不仅学到了有关 VHDL 语言和硬件设计的知识，还学会了系统设计的方法和技巧。通过本次设计，激发了我进一步学习和探索可编程逻辑器件设计领域的兴趣。