基于有限状态机的自动售票系统设计

一、输入输出端口信号分析

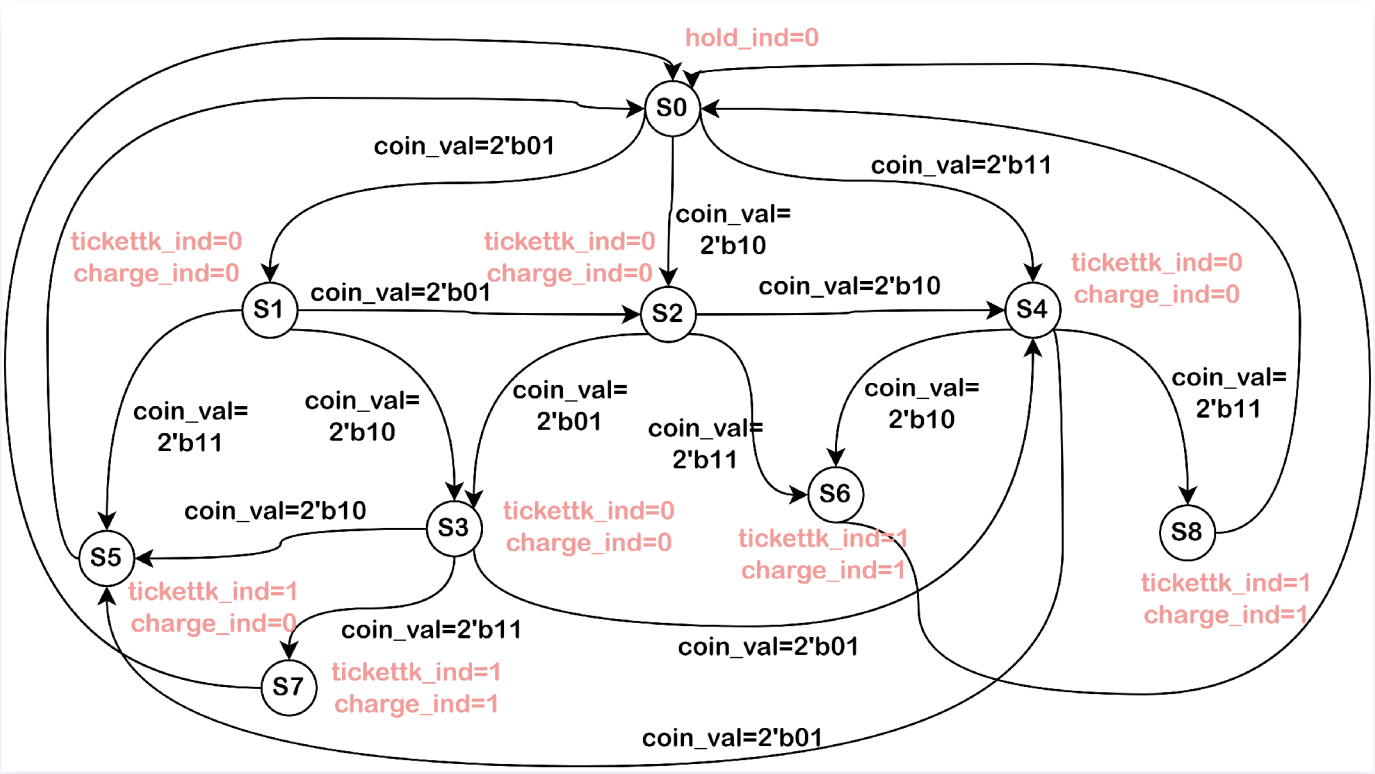
- 输入信号：clk, rst;
- 输入信号：操作开始：op_start; // op_start = 1 开始操作
- 输入信号：投币币值：coin_val; // 2'b00 表示 0 元，2'b01 表示 5 元，2'b10 表示 10 元，2'b11 表示 20 元;
- 输入信号：取消操作指示：cancel_flag; // cancel_flag = 1 表示取消操作
- 输出信号：机器是否可用：hold_ind; // hold_ind = 0表示可以使用
- 输出信号：取票信号：tickettk_ind; // tickettk_ind = 1 表示取票
- 输出信号：找零与退币标志：charge_ind; // charge_ind = 1 表示找零
- 输出信号：找零与退币币值：charge_val; // 3'b000 表示找 5 元，3'b001 表示找 10 元，3'b010 表示找 15 元，3'b011 表示找 20 元，3'b100 表示找 25 元，3'b101 表示找 30 元，3'b110 表示找 35 元，3'b111 表示找 40 元;

二、系统状态分析

2.1 状态说明

- S0: 初始状态;
- S1: 已投币 5 元;
- S2: 已投币 10 元;
- S3: 已投币 15 元;
- S4: 已投币 20 元;
- S5: 已投币 25 元;
- S6: 已投币 30 元;
- S7: 已投币 35 元;
- S8: 已投币 40 元;

2.2 状态转移图



2.3 状态转移关系

1. 始态 S0 (coin=0元)，根据状态转移条件 coin_val 分别转移到 S1(coin=5元)/S2(coin=10元)/S3(coin=15元);
2. 状态 S1(coin=5元)，根据状态转移条件 coin_val 分别转移到 S2(coin=10元)/S3(coin=15元)/S5(coin=25元);
3. 状态 S2(coin=10元)，根据状态转移条件 coin_val 分别转移到 S3(coin=15元)/S4(coin=20元)/S6(coin=30元);
4. 状态 S3(coin=15元)，根据状态转移条件 coin_val 分别转移到 S4(coin=20元)/S5(coin=25元)/S7(coin=35元);
5. 状态 S4(coin=20元)，根据状态转移条件 coin_val 分别转移到 S5(coin=25元)/S6(coin=30元)/S8(coin=40元);
6. 状态 S5(coin=25元)，无条件转移到始态 S0;
7. 状态 S6(coin=30元)，无条件转移到始态 S0;
8. 状态 S7(coin=35元)，无条件转移到始态 S0;
9. 状态 S8(coin=40元)，无条件转移到始态 S0;

2.4 系统工作流程

1. 在 S0 状态下，如果检测到 op_start = 1，开始检测是否有投币，如果有，一次新的售票操作开始;

2. 在状态 S1/S2/S3/S4 下，如果检测到 cancel_flag = 1，则取消操作，状态退回到 S0，并退回相应的币值，否则进行投币；
3. 在状态 S5 下，售票不找零；在状态 S6/S7/S8 下，售票并找零；
4. 在状态 S5/S6/S7/S8 操作完后，都返回状态 S0，等待下一轮新的操作开始；
5. 只有在 S0 状态下，hold_ind = 0，可以发起新一轮操作，其他状态都为 1.

三、VHDL语言描述

```
library IEEE;

entity saler is
    Port ( clk : in bit;
           rst : in bit;
           op_start : in bit;
           coin_val : in bit_vector(1 downto 0);
           cancel_flag : in bit;
           hold_ind : buffer bit;
           tickettk_ind : buffer bit;
           refund_ind : buffer bit;
           refund_val : buffer bit_vector(3 downto 0);
           ccur_state : buffer bit_vector(3 downto 0));
end saler;

architecture Behavioral of saler is

type states is (s0, s1, s2, s3, s4, s5, s6, s7, s8);
signal state: states;

begin
    saling_system: process(clk, rst, op_start, cancel_flag)
    begin
        -- start the saling system and initialize
        if(op_start = '1' or rst = '1') then
            state <= s0;
            tickettk_ind <= '0';
            refund_ind <= '0';
            refund_val <= "0000";
            hold_ind <= '0';
            ccur_state <= "0000";
        elsif(cancel_flag = '1') then     -- cancel the operations
        -- refund for canceling the operation
            case state is
                    when s1 => refund_val <= "0001";
                               refund_ind <= '1';
                    when s2 => refund_val <= "0010";
                               refund_ind <= '1';
```

```
39                      when s3 => refund_val <= "0011";
40                                 refund_ind <= '1';
41                      when s4 => refund_val <= "0100";
42                                 refund_ind <= '1';
43                      when s5 => refund_val <= "0101";
44                                 refund_ind <= '1';
45                      when s6 => refund_val <= "0110";
46                                 refund_ind <= '1';
47                      when s7 => refund_val <= "0111";
48                                 refund_ind <= '1';
49                      when s8 => refund_val <= "1000";
50                                 refund_ind <= '1';
51                      when others => refund_val <= "0000";
52                                 refund_ind <= '0';
53              end case;
54          -- states transfer
55          elsif(clk = '1' and hold_ind = '0') then     -- the machine is available

56              case state is
57                  when s0 => case coin_val is
58                                 when "01" => state <= s1;
59                                     ccur_state <= "0001";
60                                 when "10" => state <= s2;
61                                     ccur_state <= "0010";
62                                 when "11" => state <= s4;
63                                      ccur_state <= "0100";
64                                 when others => state <= s0;
65                                       ccur_state <= "0000";
66                             end case;
67                  when s1 => case coin_val is
68                                 when "01" => state <= s2;
69                                     ccur_state <= "0010";
70                                 when "10" => state <= s3;
71                                      ccur_state <= "0011";
72                                 when "11" => state <= s5;
73                                      ccur_state <= "0101";
74                                 when others => state <= s1;
75                                      ccur_state <= "0001";
76                             end case;
77                  when s2 => case coin_val is
78                                 when "01" => state <= s3;
79                                      ccur_state <= "0011";
80                                 when "10" => state <= s4;
81                                      ccur_state <= "0100";
82                                 when "11" => state <= s6;
83                                      ccur_state <= "0110";
84                                 when others => state <= s2;
85                                      ccur_state <= "0010";
```

```
                                end case;
                    when s3 => case coin_val is
                                    when "01" => state <= s4;
                                    ccur_state <= "0100";
                                    when "10" => state <= s5;
                                    ccur_state <= "0101";
                                    when "11" => state <= s7;
                                    ccur_state <= "0111";
                                    when others => state <= s3;
                                    ccur_state <= "0011";
                                end case;
                    when s4 => case coin_val is
                                    when "01" => state <= s5;
                                    ccur_state <= "0101";
                                    when "10" => state <= s6;
                                    ccur_state <= "0110";
                                    when "11" => state <= s8;
                                    ccur_state <= "1000";
                                    when others => state <= s1;
                                    ccur_state <= "0001";
                                end case;
                    when s5 => tickettk_ind <= '1';
                                hold_ind <= '1';
                    when s6 => tickettk_ind <= '1';
                                hold_ind <= '1';
                    when s7 => tickettk_ind <= '1';
                                hold_ind <= '1';
                    when s8 => tickettk_ind <= '1';
                                hold_ind <= '1';
                    when others => refund_ind <= '1';    -- uncommon state to refund
and exit
                                hold_ind <= '1';
                end case;
            -- get the last states
            elsif(clk = '1' and tickettk_ind ='1') then
            -- refund for extra model
                case state is
                     when s6 => refund_val <= "0001";
                                refund_ind <= '1';
                     when s7 => refund_val <= "0010";
                                refund_ind <= '1';
                     when s8 => refund_val <= "0011";
                                refund_ind <= '1';
                     when others => refund_val <= "0000";
                                refund_ind <= '0';
                 end case;
            end if;
        end process;
```

```
133
134   end Behavioral;
```

四、仿真配置

仿真配置 Pipline: 系统启动 → 购票不找零→ 购票找零→ 取消操作

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity saler_sim is
5   end saler_sim;
6
7   architecture Behavioral of saler_sim is
8
9   component saler is
10      Port ( clk : in bit;
11             rst : in bit;
12             op_start : in bit;
13             coin_val : in bit_vector(1 downto 0);
14             cancel_flag : in bit;
15             hold_ind : buffer bit;
16             tickettk_ind : buffer bit;
17             refund_ind : buffer bit;
18             refund_val : buffer bit_vector(3 downto 0);
19             ccur_state : buffer bit_vector(3 downto 0));
20  end component;
21
22  signal clk, rst, op_start, cancel_flag, hold_ind, tickettk_ind, refund_ind : bit
    := '0';
23  signal coin_val : bit_vector(1 downto 0) := "00";
24  signal refund_val, ccur_state : bit_vector(3 downto 0) := "0000";
25  constant clk_period : time := 10 ns;
26
27  begin
28      UUT: saler port map(
29             clk => clk,
30             rst => rst,
31             op_start => op_start,
32             coin_val => coin_val,
33             cancel_flag => cancel_flag,
34             hold_ind => hold_ind,
35             tickettk_ind => tickettk_ind,
36             refund_ind => refund_ind,
37             refund_val => refund_val,
38             ccur_state => ccur_state );
```

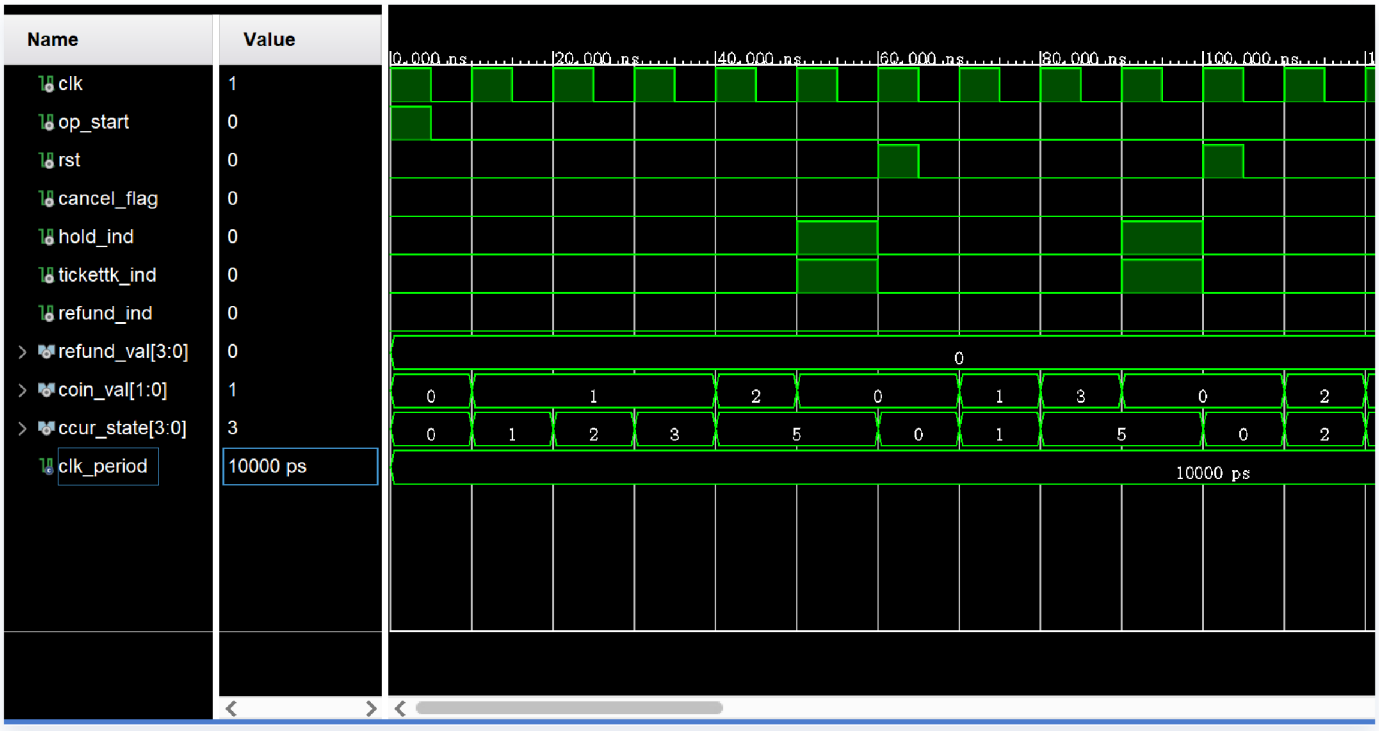
```
39
40        -- clk production
41        process
42            begin
43                clk <= '1';
44                wait for clk_period / 2;
45                clk <= '0';
46                wait for clk_period / 2;
47        end process;
48
49
50
51        process
52            begin
53
54                -- start the saling system
55                op_start <= '1';
56                wait for clk_period / 2;
57                op_start <= '0';
58                wait for clk_period / 2;
59
60                -- check coining
61                coin_val <= "01";
62                wait for clk_period;
63                coin_val <= "01";
64                wait for clk_period;
65                coin_val <= "01";
66                wait for clk_period;
67                coin_val <= "10";
68                wait for clk_period;
69                coin_val <= "00";
70                wait for clk_period;
71
72                -- reset
73                rst <= '1';
74                wait for clk_period / 2;
75                rst <= '0';
76                wait for clk_period / 2;
77
78                -- check coining
79                coin_val <= "01";
80                wait for clk_period;
81                coin_val <= "11";
82                wait for clk_period;
83                coin_val <= "00";
84                wait for clk_period;
85
86                -- reset
```

```
87              rst <= '1';
88              wait for clk_period / 2;
89              rst <= '0';
90              wait for clk_period / 2;
91
92              -- check refund
93              coin_val <= "10";
94              wait for clk_period;
95              coin_val <= "11";
96              wait for clk_period;
97              coin_val <= "00";
98              wait for clk_period;
99              coin_val <= "00";
100             wait for clk_period;
101
102             -- reset
103             rst <= '1';
104             wait for clk_period / 2;
105             rst <= '0';
106             wait for clk_period / 2;
107
108             -- check refund
109             coin_val <= "11";
110             wait for clk_period;
111             coin_val <= "11";
112             wait for clk_period;
113             coin_val <= "00";
114             wait for clk_period;
115             coin_val <= "00";
116             wait for clk_period;
117
118             -- reset
119             rst <= '1';
120             wait for clk_period / 2;
121             rst <= '0';
122             wait for clk_period / 2;
123
124             -- check cancel
125             coin_val <= "10";
126             wait for clk_period;
127             coin_val <= "01";
128             wait for clk_period;
129             cancel_flag <= '1';
130             wait for clk_period;
131             cancel_flag <= '0';
132             wait for clk_period;
133
134             -- rst
```

```
135             rst <= '1';
136             wait for clk_period / 2;
137             rst <= '0';
138             wait for clk_period / 2;
139 
140         end process;
141 end Behavioral;
```

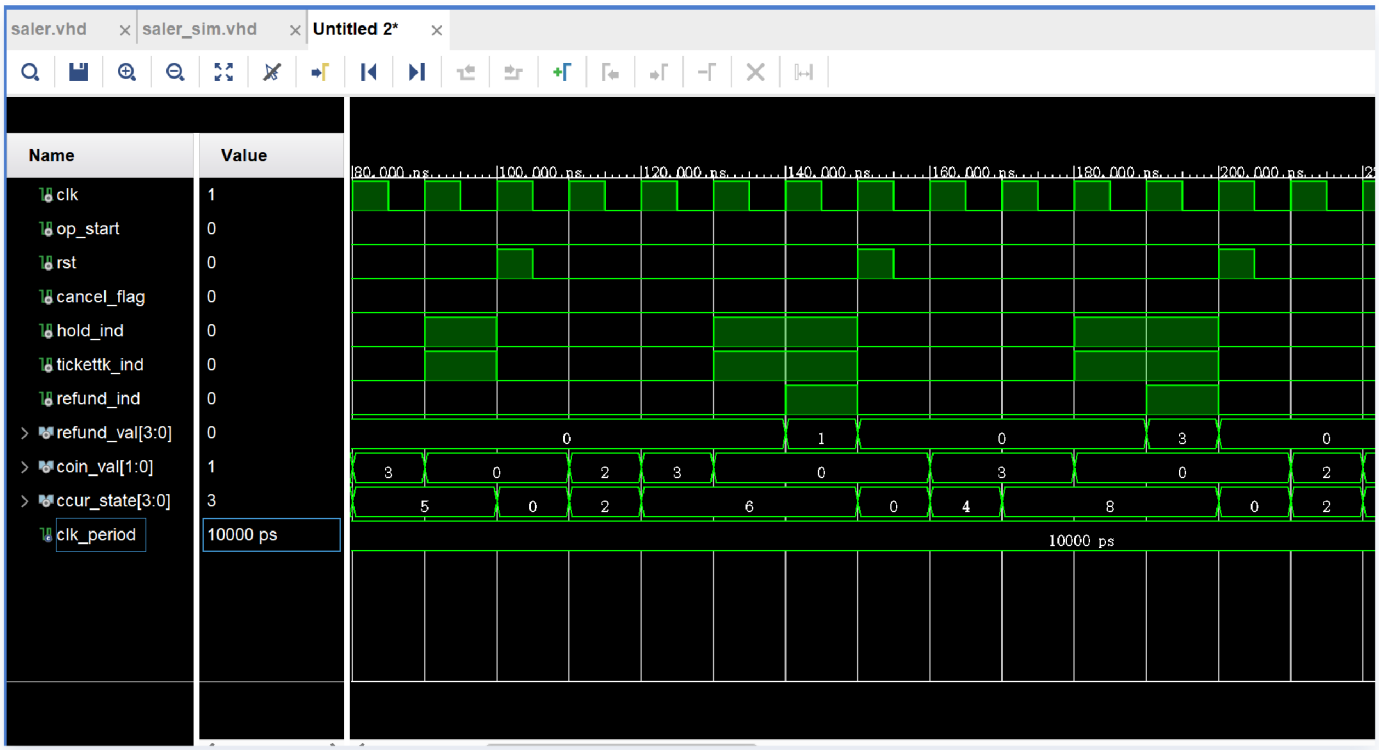
五、测试代码编写、仿真与结果分析

5.1 买票不找零：5元 + 5元 + 5元 + 10元 / 5元 + 20元



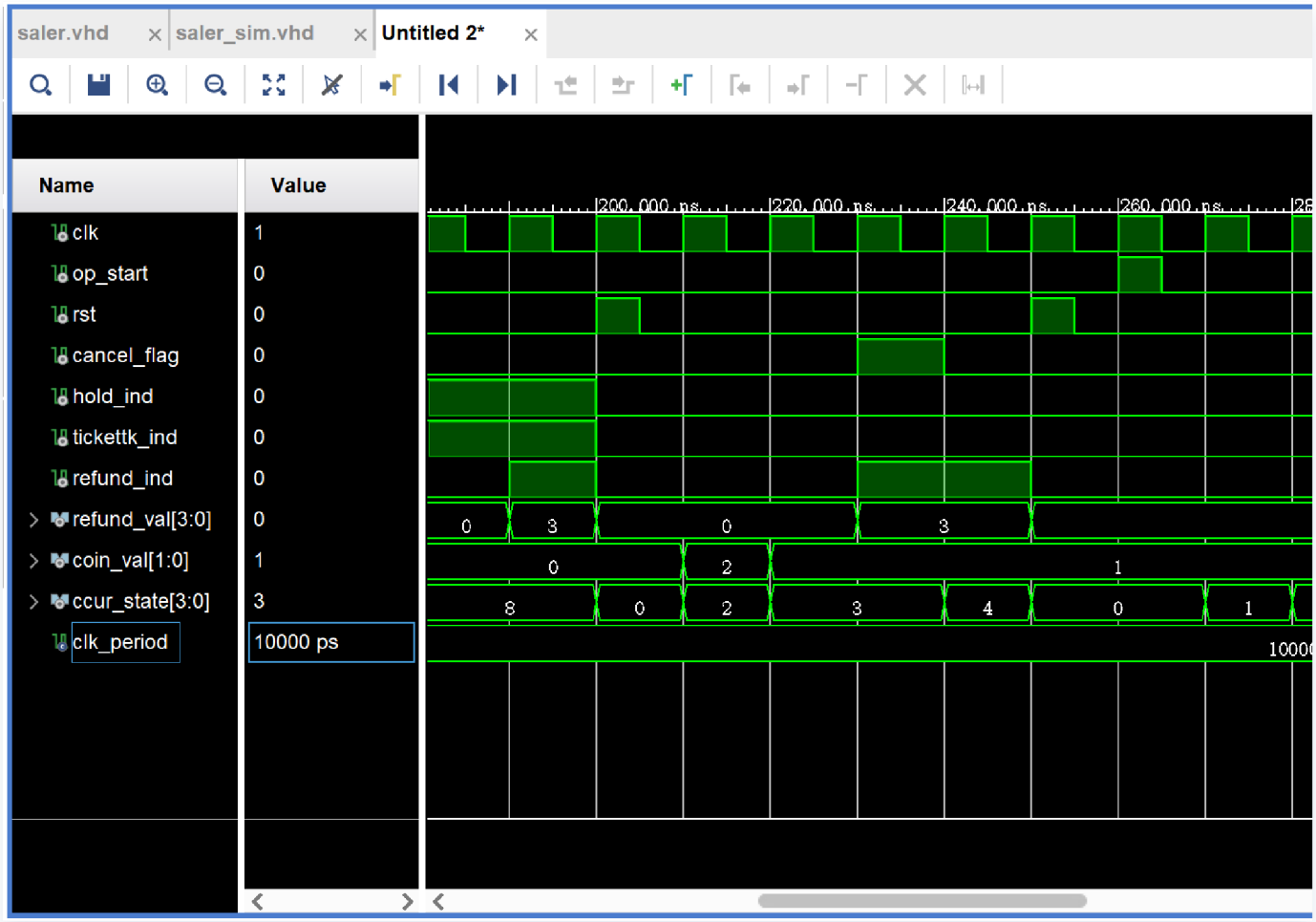
- 结果分析：
  - 5元 + 5元 + 5元 + 10元：从仿真结果可以看出，在 op_start 产生一个上升沿信号时，系统初始化并开始启动工作；coin_val = 1 (5元）持续三个 CLK，使得状态从 S0 迁移到 S3，再次 coin_val = 2（10元）使得状态迁移到 S5 (已投币25元)；下一个时钟周期检测到可以取票，此时设置系统被占用，并计算不找零；一轮操作结束后，系统异步复位初始化所有状态；
  - 5元 + 20元：系统复位后，coin_val = 1 (5元）、coin_val = 3 (20元）使得状态从 S0 迁移到 S1再迁移到 S5；下一个时钟周期检测到可以取票，此时设置系统被占用，并计算不找零；一轮操作结束后，系统异步复位初始化所有状态；

5.2 买票找零：10元 + 20元 / 20元 + 20元



- 结果分析：
  - 10元 + 20元：从仿真结果可以看出，在系统复位后，coin_val = 2 (10元）、coin_val = 3 (20元）使得状态从 S0 迁移到 S2 再迁移到 S6；下一个时钟周期检测到可以取票，并计算需要找零，找零 refund_val=1（5元）；一轮操作结束后，系统异步复位初始化所有状态；
  - 20元 + 20元：从仿真结果可以看出，在系统复位后，coin_val = 3 (20元）持续两个CLK，使得状态从 S0 迁移到 S4 再迁移到 S8；下一个时钟周期检测到可以取票，并计算需要找零，找零 refund_val=3（15元）；一轮操作结束后，系统异步复位初始化所有状态；

5.3 取消操作并退款：10元+5元



- 结果分析：
  - 10元 + 5元 ： 从仿真结果可以看出，在系统复位后，coin_val = 2 (10元）、coin_val = 1 (5元）使得状态从 S0 迁移到 S2 再迁移到 S3；此时用户取消操作，使得cancel_flag = 1，则系统进行自动退款，退款refund_val = 3(15元）；下一个时钟周期对系统异步复位并初始化所有状态；