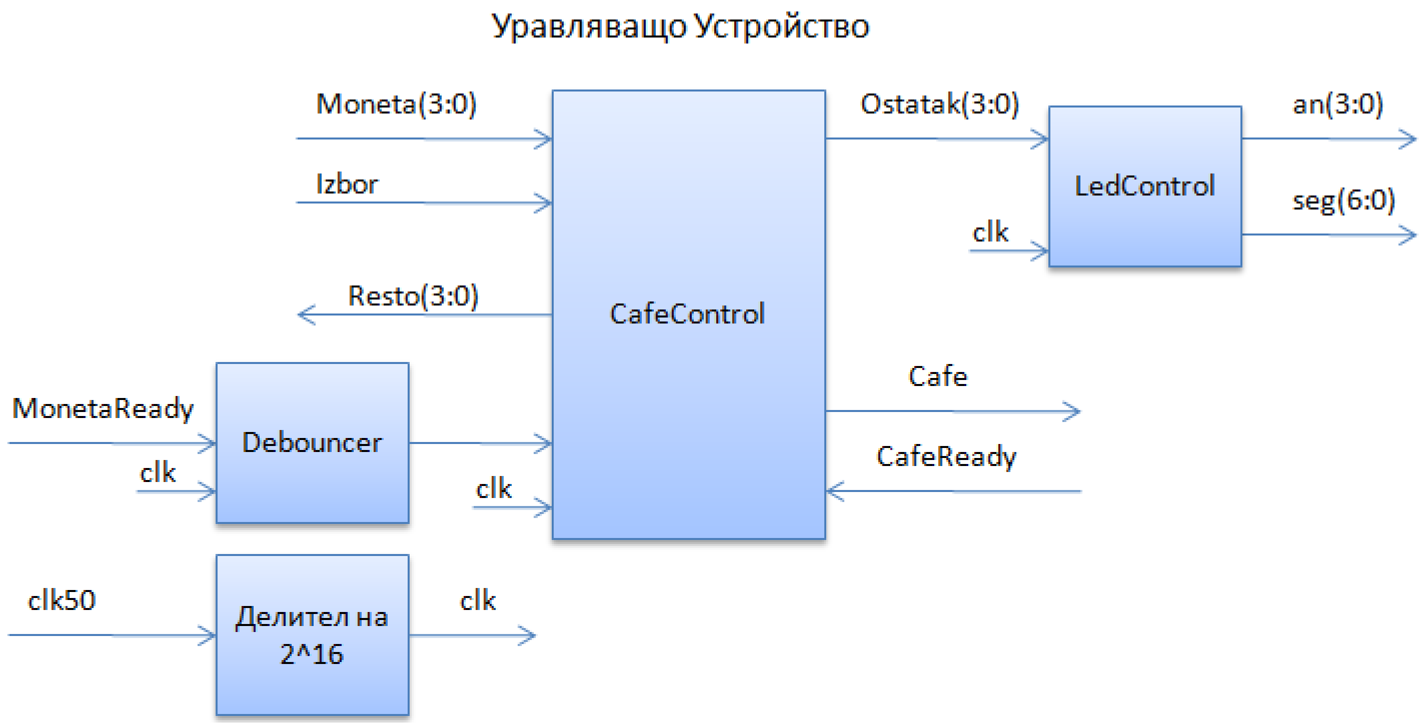
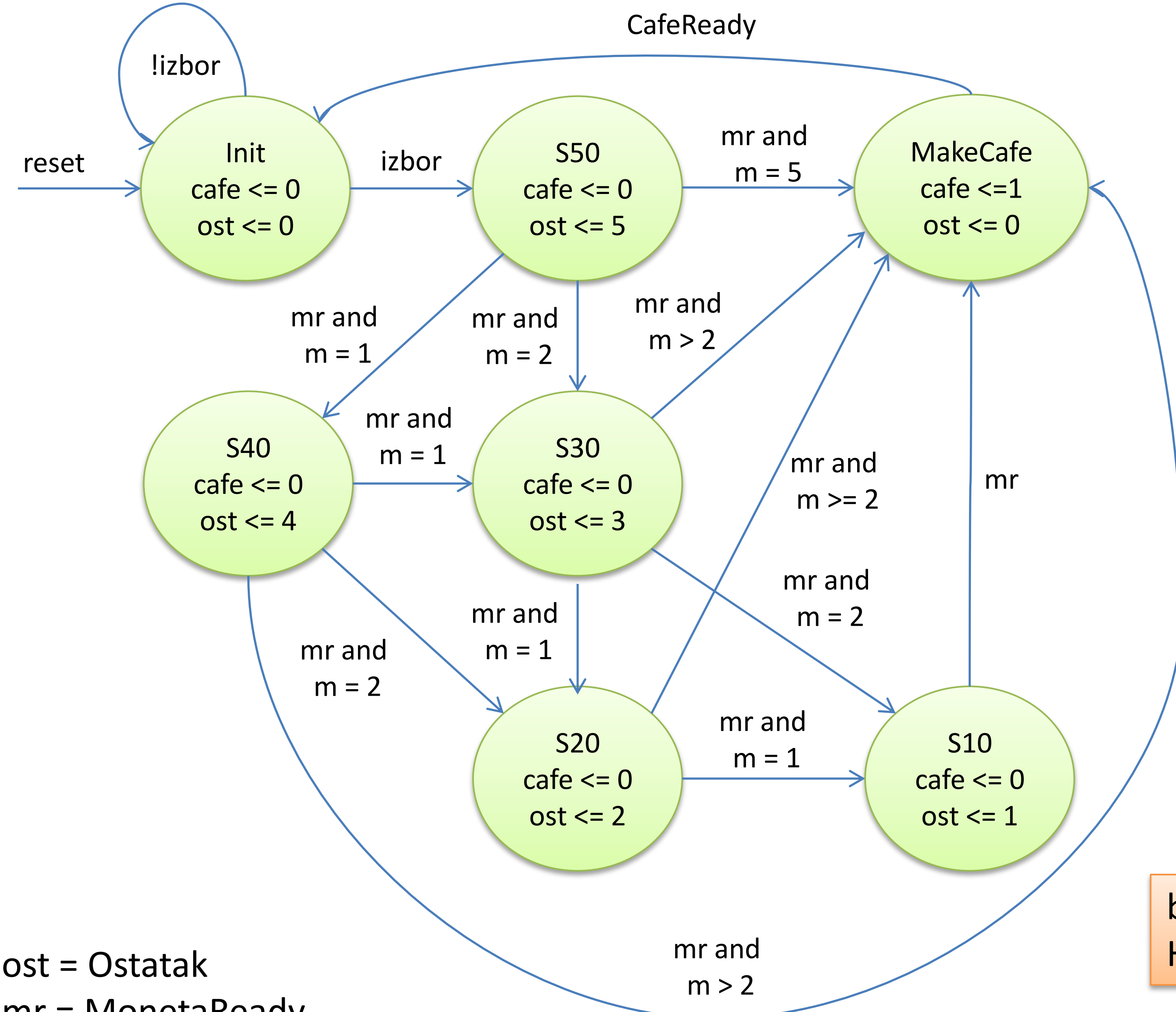
Автомат за кафе

Цената на едно кафе е 50 ст.
Автоматът приема монети от по 50, 20 и 10 стотинки.
На дисплей се показва остатъкът от сумата, която трябва да бъде платена.



Блокова схема на управляващото устройство



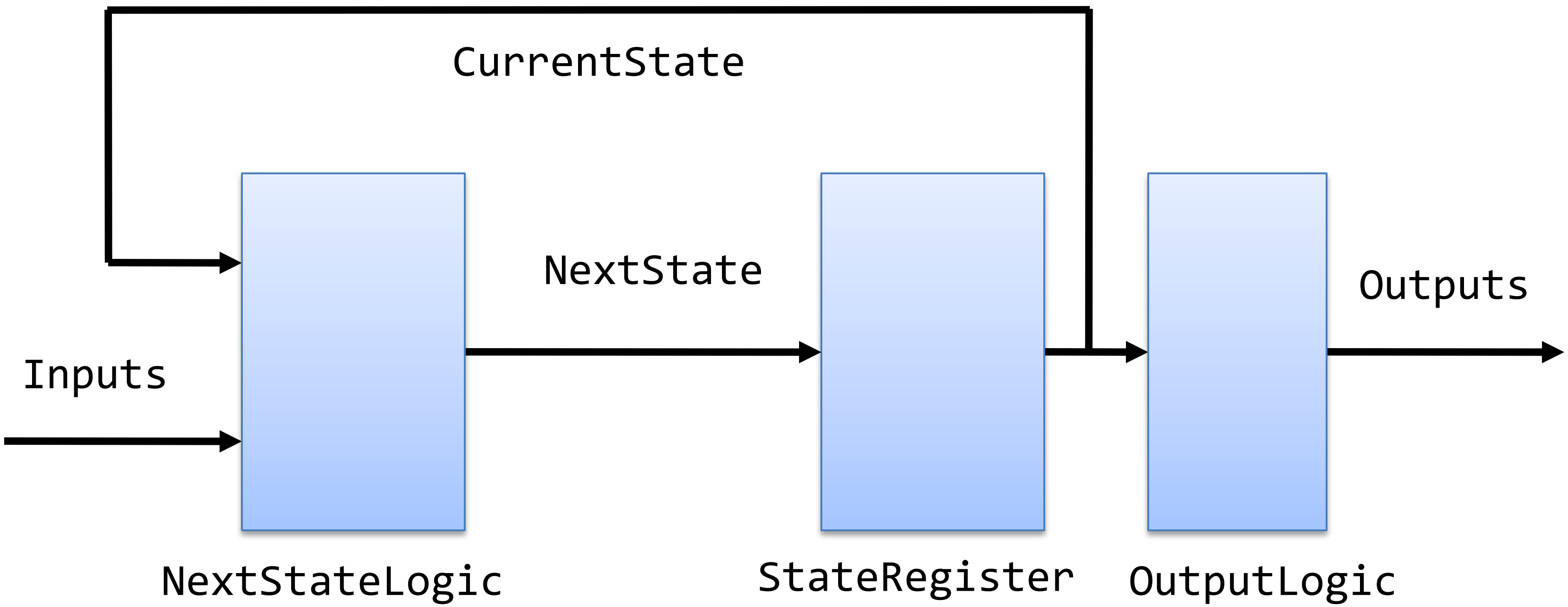
beta - версия
Не връща ресто!

ost = Ostatak
mr = MonetaReady
m = Moneta

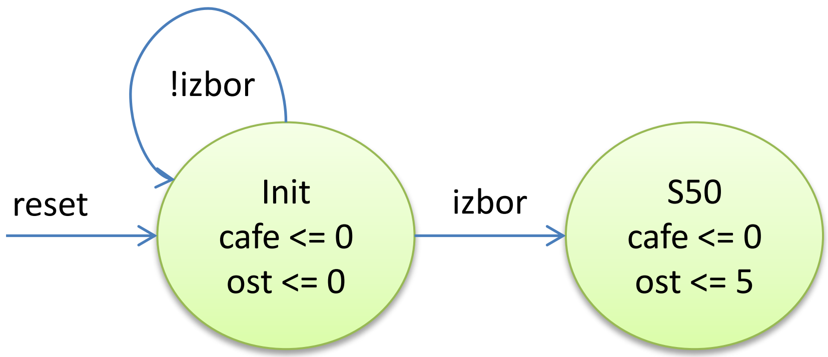
Moneta: 1,2,5 -> 10ст, 20ст, 50ст

```vhdl
entity CafeControl is
    Port ( clk : in  std_logic;
           reset : in  std_logic;
           Izbor : in std_logic;
           MonetaReady : in std_logic;
           Moneta : in std_logic_vector (3 downto 0);
           Cafe : out  std_logic;
           Ostatak : out  std_logic_vector (3 downto 0);
           Resto : out  std_logic_vector (3 downto 0);
           RestoReady : out std_logic;
           CafeReady : in  std_logic);
end CafeControl;

architecture Behavioral of CafeControl is
    type StateType is (Init, S50, S40, S20, S30, S10, MakeCafe);
    signal CurrentState, NextState : StateType;
    signal Ostatak_int, Moneta_int : integer range 0 to 5;
begin
    Moneta_int <= to_integer(unsigned(Moneta));
    Ostatak <= std_logic_vector(to_unsigned(Ostatak_int,4));
```

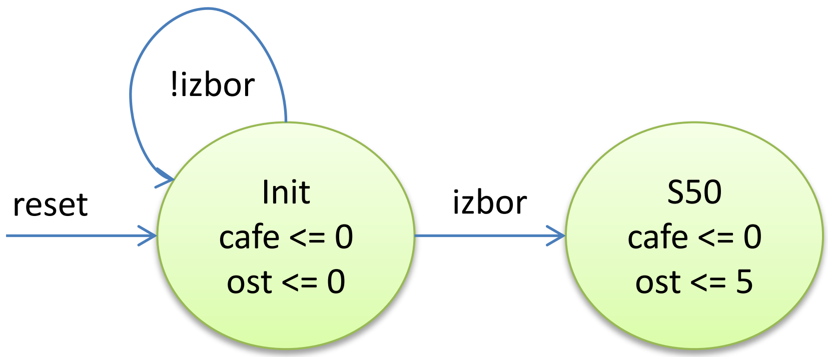
Структура на краен автомат



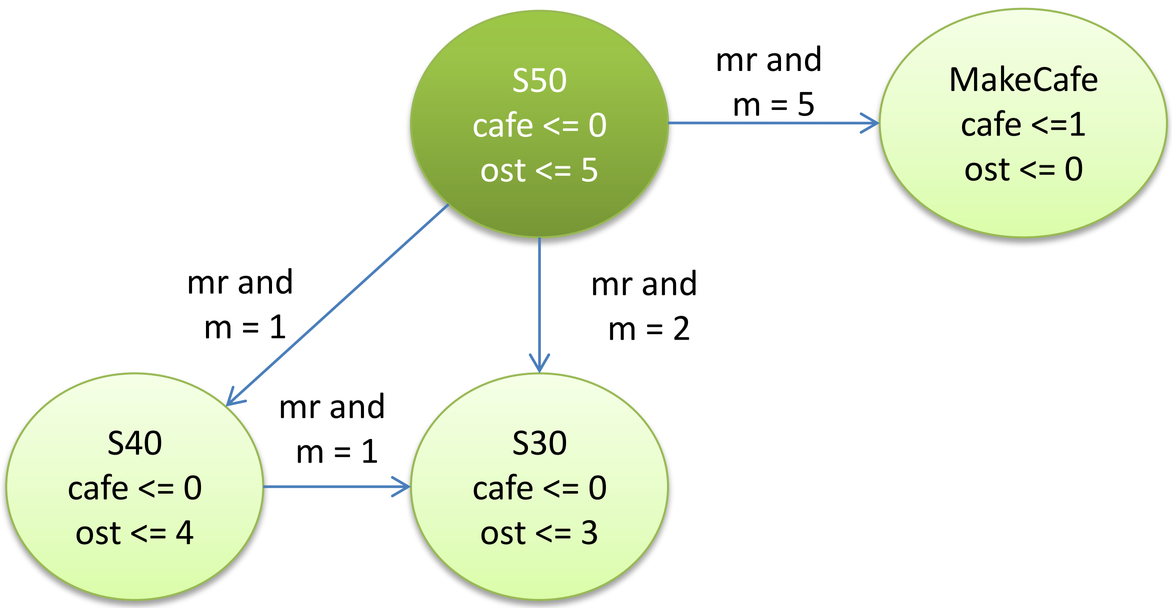
```
StateRegister: process(clk, reset)
    begin
        if reset = '1' then
            CurrentState <= Init;
        elsif rising_edge(clk) then
            CurrentState <= NextState;
        end if;
    end process;

NextStateLogic: process (CurrentState, Izbor, MonetaReady, Moneta, CafeReady)
    begin
        case CurrentState is
            ...
            ...
        end case;
    end process;

OutputLogic: process (CurrentState) begin
        case CurrentState is
            ...
            ...
        end case;
    end process;
```



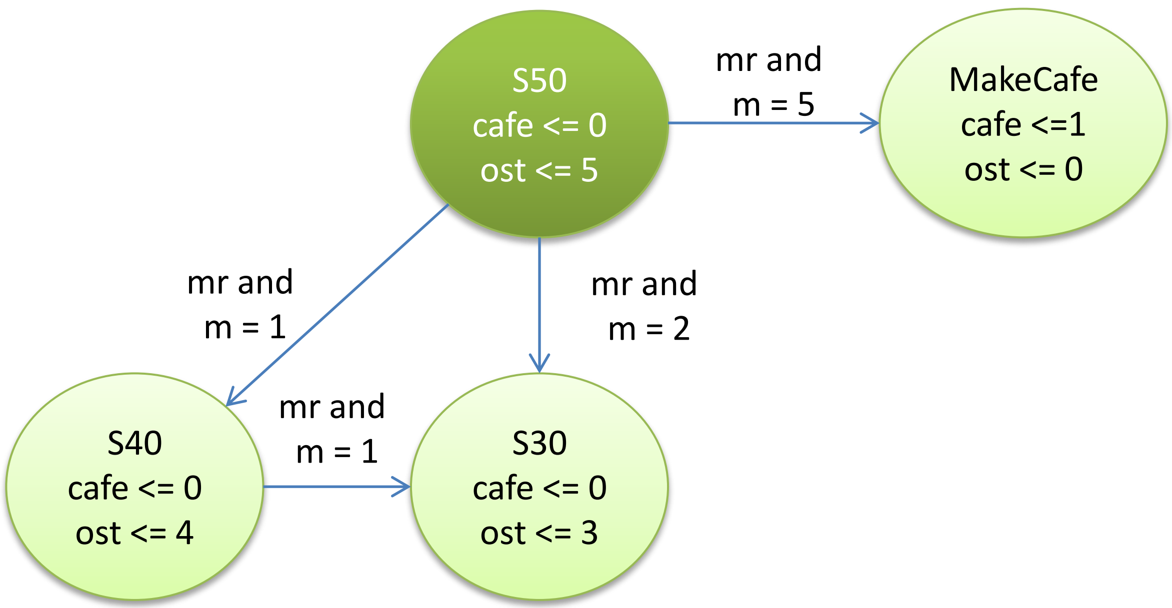

```
StateRegister: process(clk, reset) begin
        if(reset = '1')then
            CurrentState <= Init;
        elsif(rising_edge(clk))then
            CurrentState <= NextState;
        end if;
    end process;
 NextStateLogic: process (CurrentState, Izbor, MonetaReady, Moneta, CafeReady)
    begin
        case CurrentState is
        when Init =>
            if Izbor = '1' then
                NextState <= S50;
            else
                NextState <= Init;
            end if;
```




```
OutputLogic: process (CurrentState) begin
        case CurrentState is
        when Init =>
            Cafe <= '0';
            Ostatak_int <= 0;
```



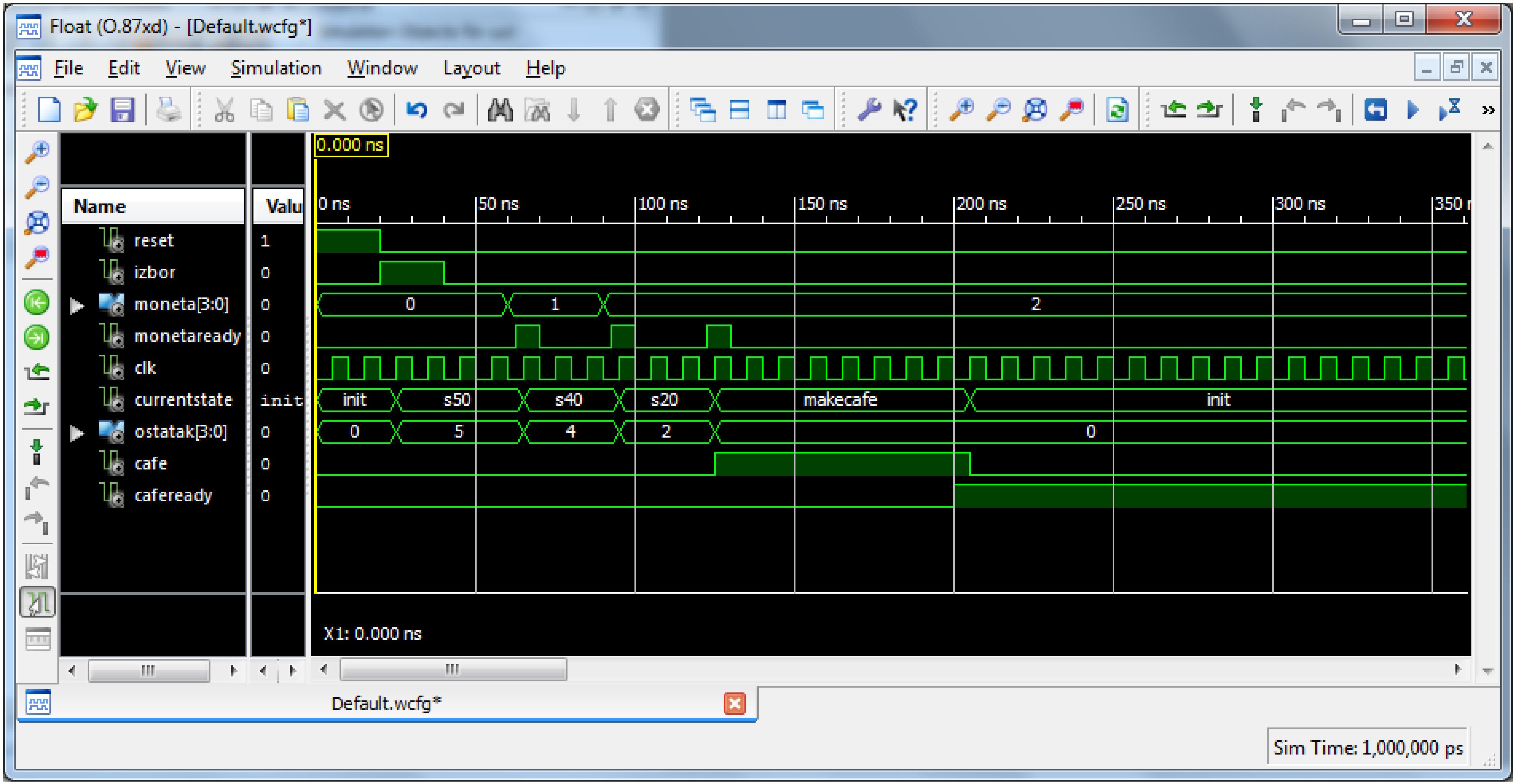

```
NextStateLogic: process (CurrentState, Izbor, MonetaReady, Moneta, CafeReady) begin
case CurrentState is
…
when S50 =>
    if MonetaReady = '1' and Moneta = 5 then
        NextState <= MakeCafe;
    elsif MonetaReady = '1' and Moneta = 2 then
        NextState <= S30;
    elsif MonetaReady = '1' and Moneta = 1 then
        NextState <= S40;
    else
        NextState <= S50;
    end if;
```

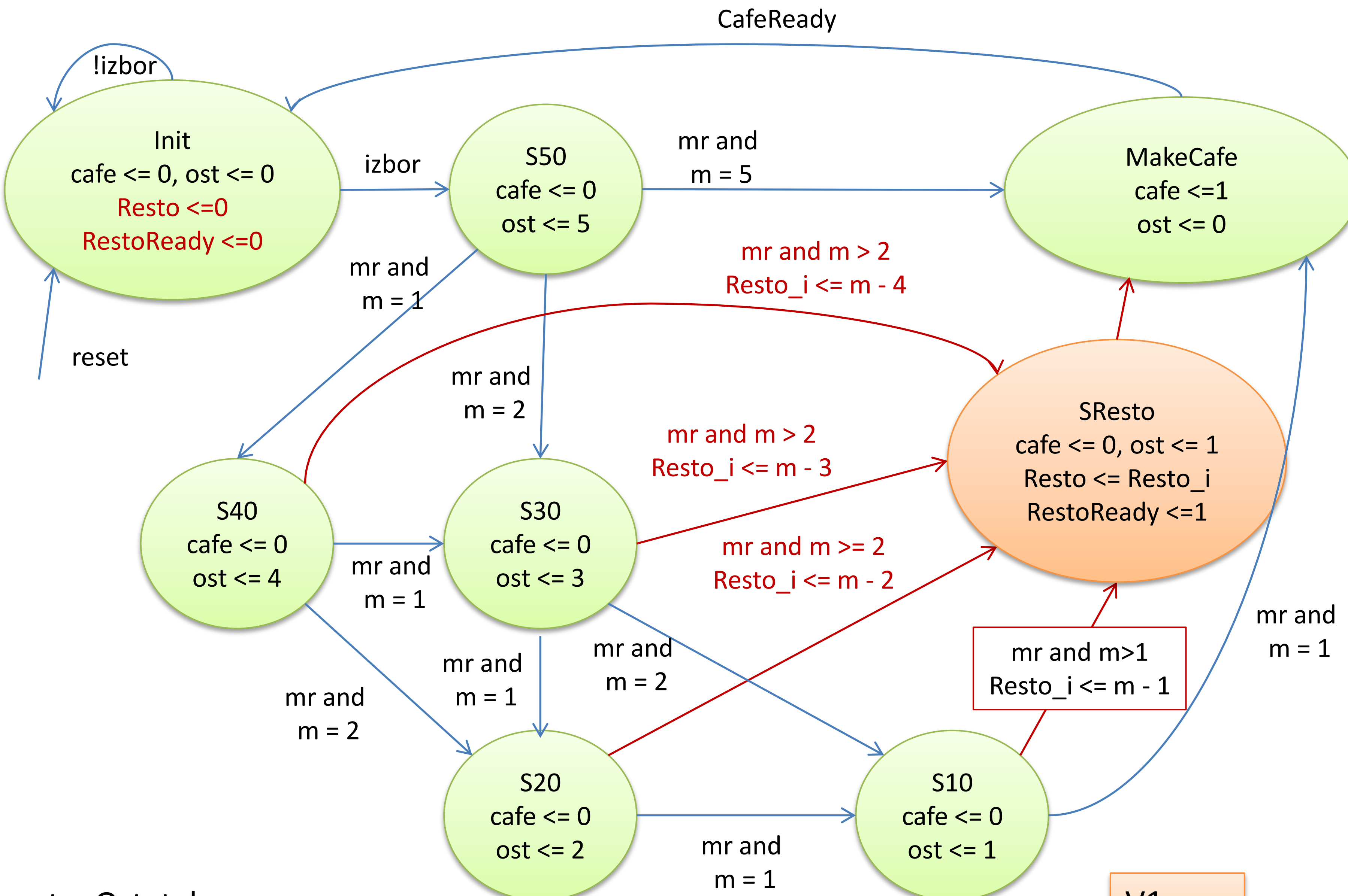



```
OutputLogic: process (CurrentState) begin
        case CurrentState is
        when Init =>
            Cafe <= '0';
            Ostatak <= 0;
        when S50 =>
            Cafe <= '0';
            Ostatak <= 5;
```

```
NextStateLogic: process (CurrentState, Izbor, MonetaReady, Moneta, CafeReady)
  begin
      case CurrentState is
      when Init =>
          if(Izbor = '1')then
              NextState <= S50;
          else
              NextState <= Init;
          end if;
...
...
      when MakeCafe =>
          if CafeReady = '1' then
              NextState <= Init;
          else
              NextState <= MakeCafe;
          end if;
      when others => NextState <= Init;
      end case;
  end process;
```

Възстановяване от невалидни състояния


Float (O.87xd) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
0.000 ns
Name
Valu
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
reset 1
izbor 0
moneta[3:0] 0
monetaready 0
clk 0
currentstate init
ostatak[3:0] 0
cafe 0
cafeready 0
0
1
2
init
s50
s40
s20
makecafe
init
0
5
4
2
0
X1: 0.000 ns
Default.wcfg*
Sim Time: 1,000,000 ps



ost = Ostatak
mr = MonetaReady
m = Moneta

Moneta: 1,2,5 -> 10ст, 20ст, 50ст

V1
- ресто