VHDL Code

Simulation & Implementation

in ISE Xilinx

vending_machine_processor:

contains: accumulator8, Adder8. full_Adder, comparator8, comparator_1bit, comparator_co, mux21, subtractor8

Mohammad Niknam

Reference:

https://github.com/MohammadNiknam17/vending_machine_processor

Block Diagram:

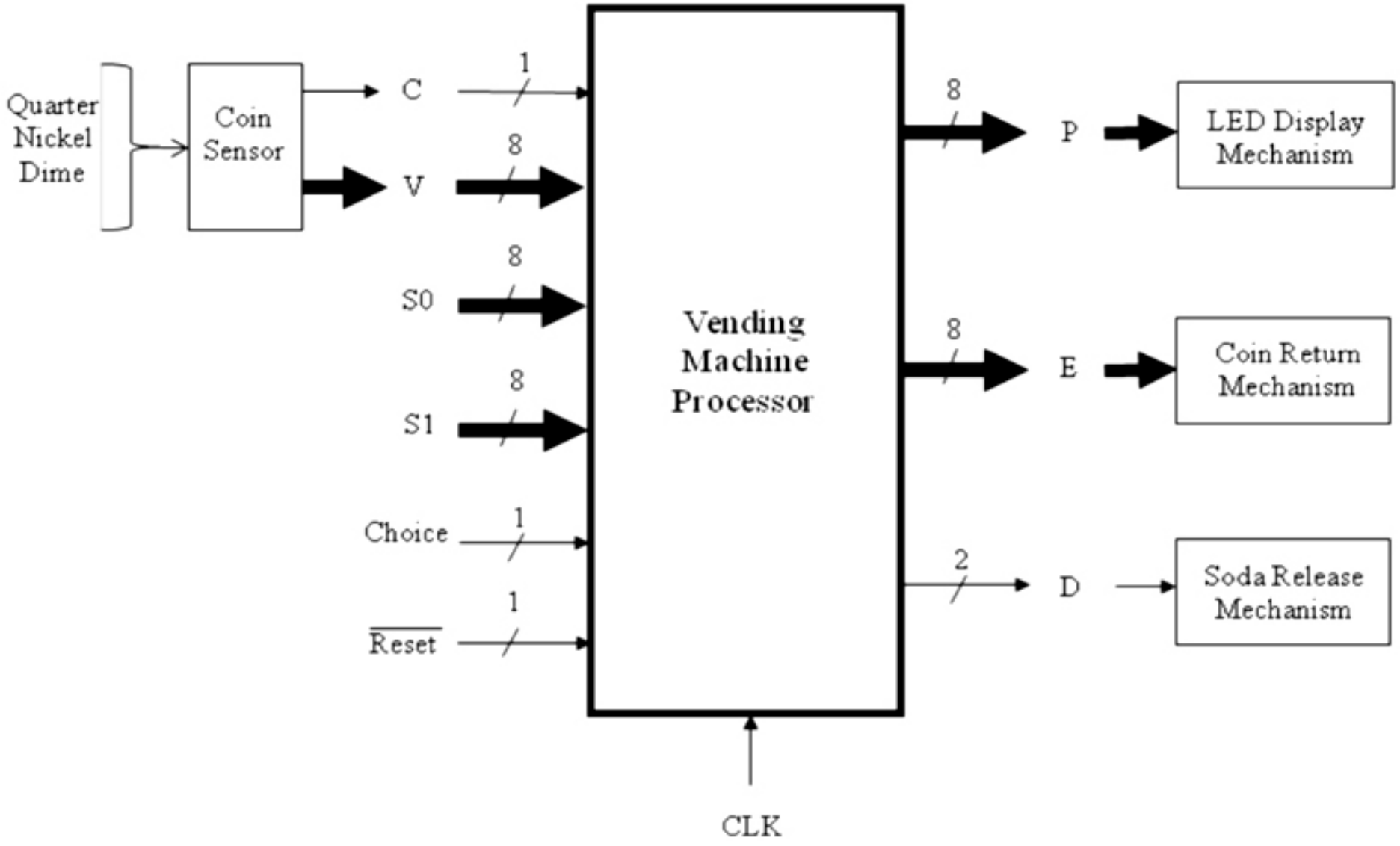


Figure 1

Project Describtion:

Basic 8-bit vending machine processor design which offers support for two drinks and currency of 5 cent, 10 cent, 25 cent denominations.
As an engineer working in a high tech company, asked to implement a RTL description of a vending machine processor using VHDL to control a vending machine.
Figure 1 shows the block diagram of the vending machine.

Here is how the vending machine is supposed to work:
The machine has a single coin slot that accepts nickels, dimes and quarters, one at a time. A coin sensor provides the processor with a 1-bit input C that becomes 1 when a coin is detected, and an 8-bit input V indicating the coin's value in cents. Two 8-bit inputs SO and S1 indicate the cost of the two types of soda drink. Note that the value of SO and S1 can be set by the vending machine owner. Choice is a 1-bit input to the vending machine indicating the type of soda drink selected (Choice 0 is for soda drink type 0 with the cost S0 and Choice 1 is for soda drink type 1 with cost of S1).

If the amount of money deposited is less than the cost of the soda, the processor generates an 8-bit output P to display the amount of money deposited in cents. Once the processor has seen coins whose value equals or exceeds the cost of a soda, the processor should set an 2-bit output D for one clock cycle, causing a soda to be dispensed (00 or 11 for no action, 01 for the release of soft drink type 0, and 10 for the
release of soft drink type 1).

The vending machine also should generate the 8-bit output E indicating the change to be returned in cents.

Question Link:

https://www.chegg.com/homework-help/questions-and-answers/junior-engineer-working-high-tech-company-asked-implement-rtl-description-vending-machine--q36105103

vending_machine:

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  vending_machine_processor
-- Module Name:  vending_machine - rtl
library ieee;
USE ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

entity vending_machine is
    port(
        nRST : in std_logic;  --negative reset
        clk : in std_logic;
        C : in std_logic; --from coin sensor: becomes 1 when a coin is detected.
        V : in std_logic_vector(7 downto 0); --received coin's value in cents from coin sensor.
        S0 : in std_logic_vector(7 downto 0); --price of choice 0, defined by vending machin owner.
        S1 : in std_logic_vector(7 downto 0); --price of choice 1, defined by vending machin owner.
        choice : in std_logic;
        P : out std_logic_vector(7 downto 0); --Display
        E : out std_logic_vector(7 downto 0); --return change
        D : out std_logic_vector(1 downto 0) --soda dispensation
        );
end vending_machine;

architecture rtl of vending_machine is
component accumulator8 is
    port(
        clk: in std_logic;
        nRST_acc: in std_logic;
        C : in std_logic;   --becomes 1 when a coin is detected.
        data_in : in std_logic_vector(7 downto 0);
        data_out : out std_logic_vector(7 downto 0)
        );
end component;

component subtractor8 is
    port(
        a : in std_logic_vector(7 downto 0);
        b : in std_logic_vector(7 downto 0);
        result : out std_logic_vector(7 downto 0)
        );
end component;

component comparator8 is
    port(
        a : in std_logic_vector(7 downto 0);
        b : in std_logic_vector(7 downto 0);
        g_out : out std_logic;
        e_out : out std_logic;
        l_out : out std_logic);
end component;

component mux21 is
    port(
        A : in std_logic_vector(7 downto 0);
        B : in std_logic_vector(7 downto 0);
        s : in std_logic;
        output : out std_logic_vector(7 downto 0)
        );
end component;

type FSMTYPE is (INIT_STATE, Coin_Reception, soda_dispensation);

signal CSTATE, NSTATE : FSMTYPE;
signal balance, price, price_reg, coins_to_return : std_logic_vector(7 downto 0);
signal price_choice_reg_EN, balance_greater, balance_equal, balance_lower: std_logic;
signal choice_reg, dispensation_EN : std_logic;
signal nRST_acc : std_logic;

```

vending_machine:

```
  begin
      price_registration : process( CLK )
      begin
          if (CLK'event and CLK = '1') then
              if (price_choice_reg_EN = '1') then
                  price_reg <= price;
                  choice_reg <= choice;
              end if ;
          end if ;
      end process ; -- price_registration


      state_registration : process( CLK )
      begin
          if (CLK'event and CLK = '1') then
              if (nRST = '0') then
                  CSTATE <= INIT_STATE;
              else
                  CSTATE <= NSTATE;
              end if ;
          end if ;
      end process ; -- state_registration

      soda_dispensation_proc: process(clk)
      begin
          if (CLK'event and CLK = '1') then
              if (dispensation_EN = '1') then
                  if(choice_reg = '0') then
                      D <= "01";  --S0
                  else
                      D <= "10";  --S1
                  end if;
              else
                  D <= "00";
              end if;
          end if;
      end process; --soda_dispensation_Proc;

      next_state : process( CSTATE, balance, C, balance_equal, balance_greater,
      coins_to_return)
      begin
          NSTATE <= CSTATE;
          nRST_acc <= '1';
          price_choice_reg_EN <= '0';
          dispensation_EN <= '0';
          p <= (others => '0');
          E <= (others => '0');

          case( CSTATE ) is
              when INIT_STATE =>
                  nRST_acc <= '0';
                  price_choice_reg_EN <= '1';
                  E <= balance;
                  if (C = '1') then
                      nRST_acc <= '1';
                      NSTATE <= Coin_Reception;
                  end if ;

              when Coin_Reception =>
                  P <= balance;
                  if (balance_equal = '1' or balance_greater = '1') then
                      NSTATE <= soda_dispensation ;
                  end if ;

              when soda_dispensation =>
                  dispensation_EN <= '1';
                  E <= coins_to_return;
                  nRST_acc <= '0';
                  NSTATE <= INIT_STATE;

              when others =>
          end case ;
      end process ; -- next_state
```

vending_machine:

```
147       mux : mux21 port map(S0, S1, choice, price);
148       accumulator : accumulator8 port map (clk, nRST_acc, C, V, balance);
149       comparator : comparator8 port map (balance, price_reg, balance_greater,
          balance_equal, balance_lower);
150       subtractor : subtractor8 port map (balance, price_reg, coins_to_return);
151
152   end rtl;
153
154
```

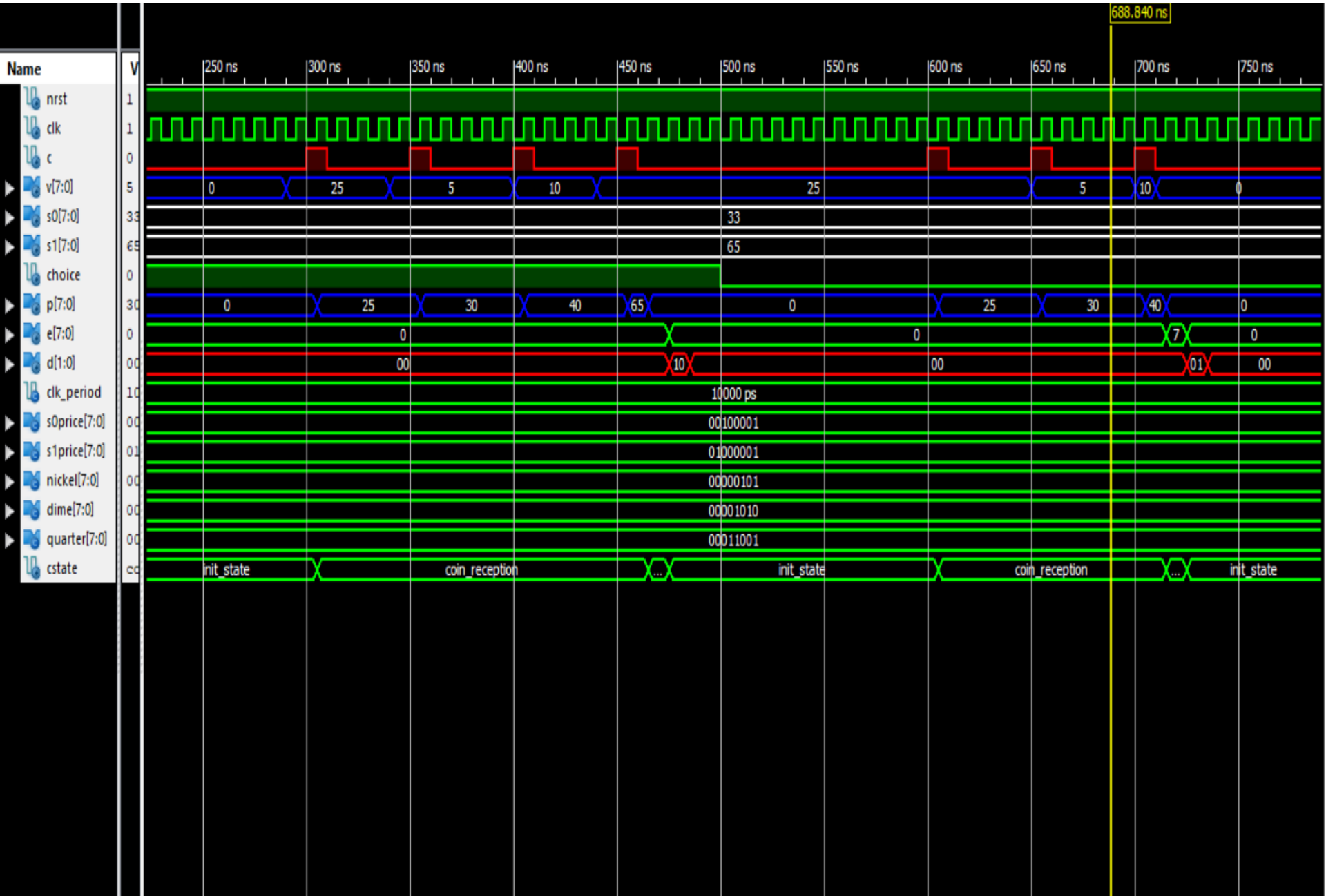
vending_machine_TB:

```
1   -- Engineer: Mohammad Niknam
2   -- Project Name:  vending_machine_processor
3   -- Module Name:  vending_machine_TB
4   library ieee;
5   USE ieee.std_logic_1164.all;
6   use IEEE.NUMERIC_STD.ALL;
7
8   ENTITY vending_machine_TB IS
9   END vending_machine_TB;
10
11  ARCHITECTURE behavior OF vending_machine_TB IS
12      -- Component Declaration for the Unit Under Test (UUT)
13
14      COMPONENT vending_machine
15      PORT(
16           nRST : IN  std_logic;
17           clk : IN  std_logic;
18           C : IN  std_logic;
19           V : IN  std_logic_vector(7 downto 0);
20           S0 : IN  std_logic_vector(7 downto 0);
21           S1 : IN  std_logic_vector(7 downto 0);
22           choice : IN  std_logic;
23           P : OUT  std_logic_vector(7 downto 0);
24           E : OUT  std_logic_vector(7 downto 0);
25           D : OUT  std_logic_vector(1 downto 0)
26          );
27      END COMPONENT;
28
29     --Inputs
30     signal nRST : std_logic := '0';
31     signal clk : std_logic := '0';
32     signal C : std_logic := '0';
33     signal V : std_logic_vector(7 downto 0) := (others => '0');
34     signal S0 : std_logic_vector(7 downto 0) := (others => '0');
35     signal S1 : std_logic_vector(7 downto 0) := (others => '0');
36     signal choice : std_logic := '0';
37
38      --Outputs
39     signal P : std_logic_vector(7 downto 0);
40     signal E : std_logic_vector(7 downto 0);
41     signal D : std_logic_vector(1 downto 0);
42
43     -- Clock period definitions
44     constant clk_period : time := 10 ns;
45
46     constant S0price : std_logic_vector(7 downto 0) := "00100001"; --soda S0 price=
       33 cent
47     constant S1price : std_logic_vector(7 downto 0) := "01000001"; --soda S1 price=
       65 cent
48
49     constant Nickel : std_logic_vector(7 downto 0) := "00000101"; --us_coin value= 5
       cent
50     constant Dime : std_logic_vector(7 downto 0) := "00001010"; --us_coin value= 10
       cent
51     constant Quarter : std_logic_vector(7 downto 0) := "00011001"; --us_coin value=
       25 cent
52
53  BEGIN
54      -- Instantiate the Unit Under Test (UUT)
55     uut: vending_machine PORT MAP (
56            nRST => nRST,
57            clk => clk,
58            C => C,
59            V => V,
60            S0 => S0,
61            S1 => S1,
62            choice => choice,
63            P => P,
64            E => E,
65            D => D
66          );
```

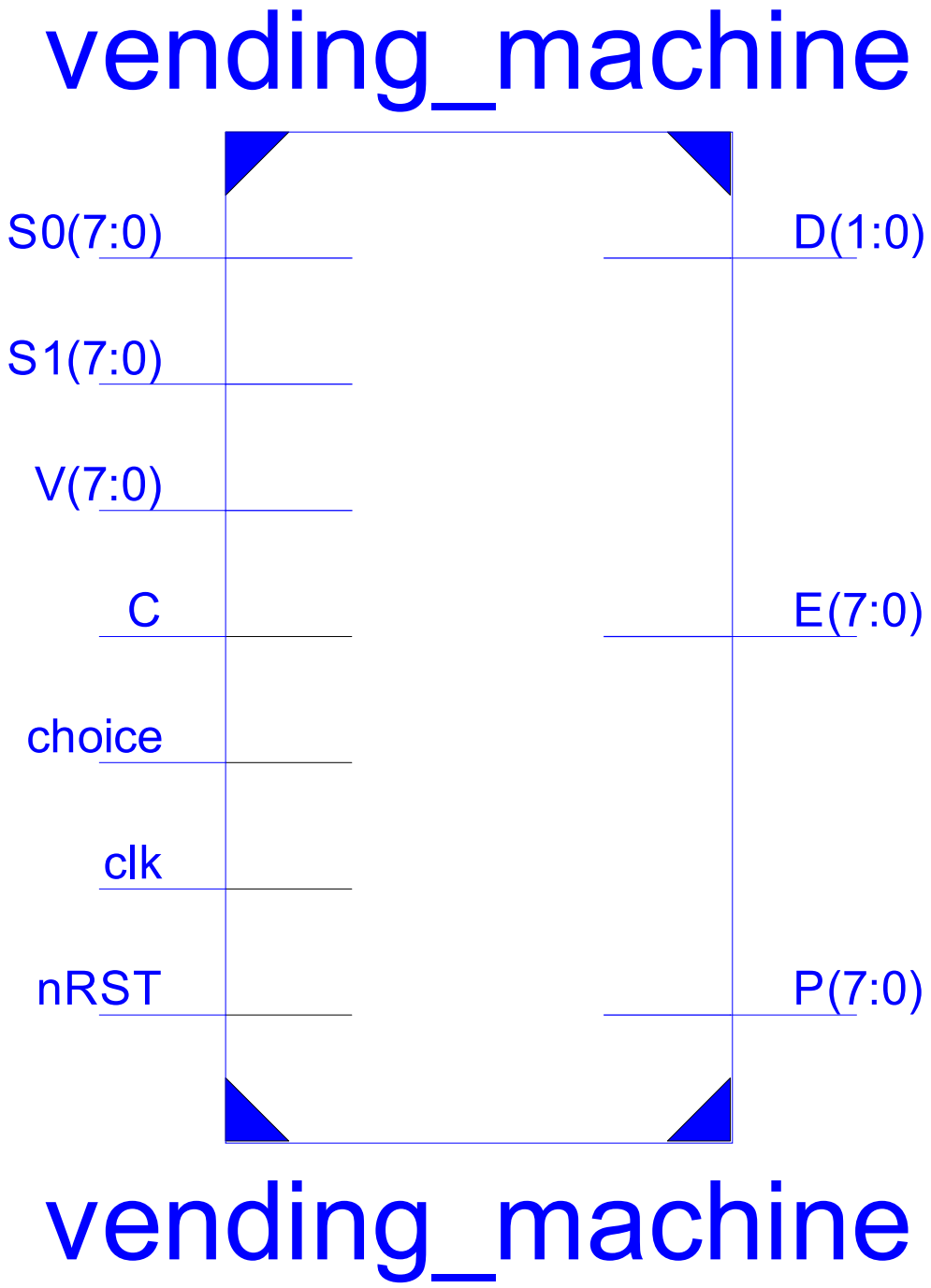
vending_machine_TB:

```
77      CLK <= not(CLK) after clk_period/2;
78      nRST <= '0', '1' after 150 ns;  -- hold reset state for 400 ns.
79      s0 <= S0price; --soda S0 price= 33 cent
80      S1 <= S1price; --soda S1 price= 65 cent
81      choice <= '1', '0' after 500 ns;
82      C <= '0', '1' after 300 ns, '0' after 310 ns, '1'after 350ns, '0' after 360 ns,
83                '1' after 400 ns, '0' after 410 ns, '1'after 450ns, '0' after 460 ns,
84                '1' after 600 ns, '0' after 610 ns, '1'after 650ns, '0' after 660 ns,
85                '1' after 700 ns, '0' after 710 ns;
86
87      v <= (others => '0'), Quarter after 290 ns, Nickel after 340 ns, Dime after 400
        ns, Quarter after 440 ns,
88      Quarter after 600 ns, Nickel after 650 ns, Dime after 700 ns, (others => '0')
        after 710 ns;
89  END;
90
```

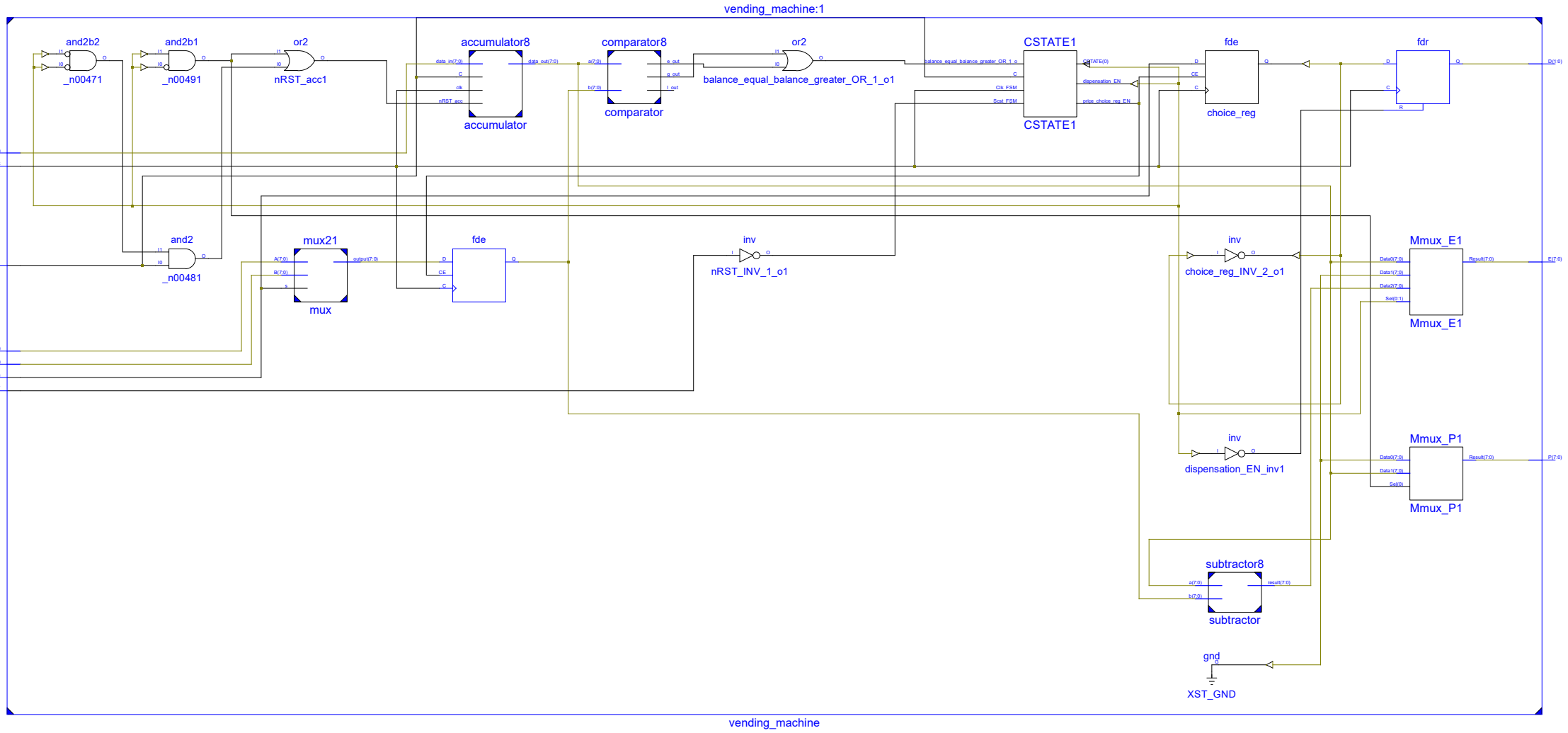
vending_machine Simulation:



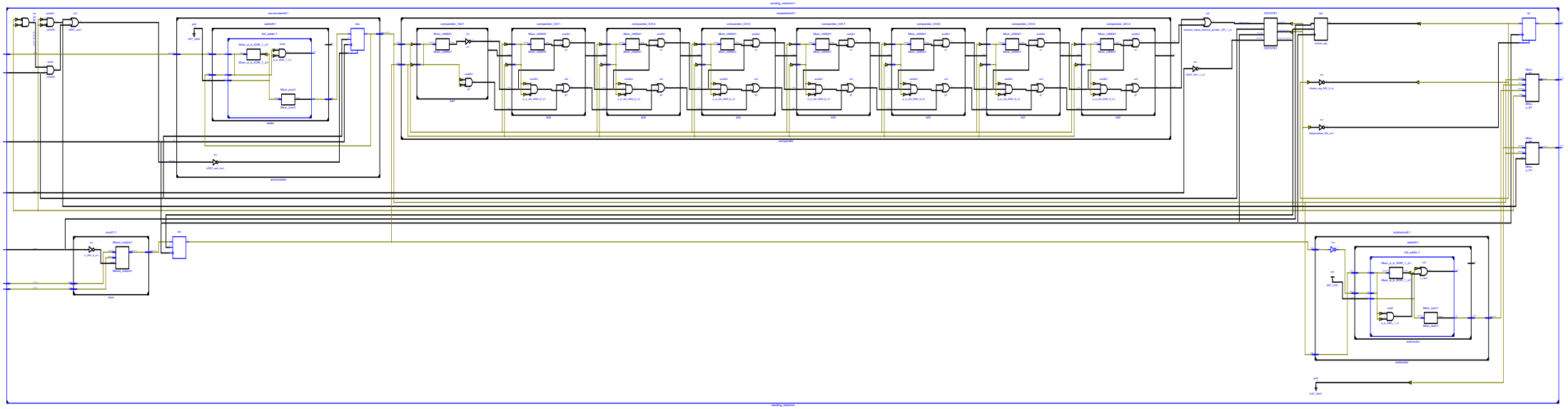
vending_machine Synthesize & RTL Schematic:



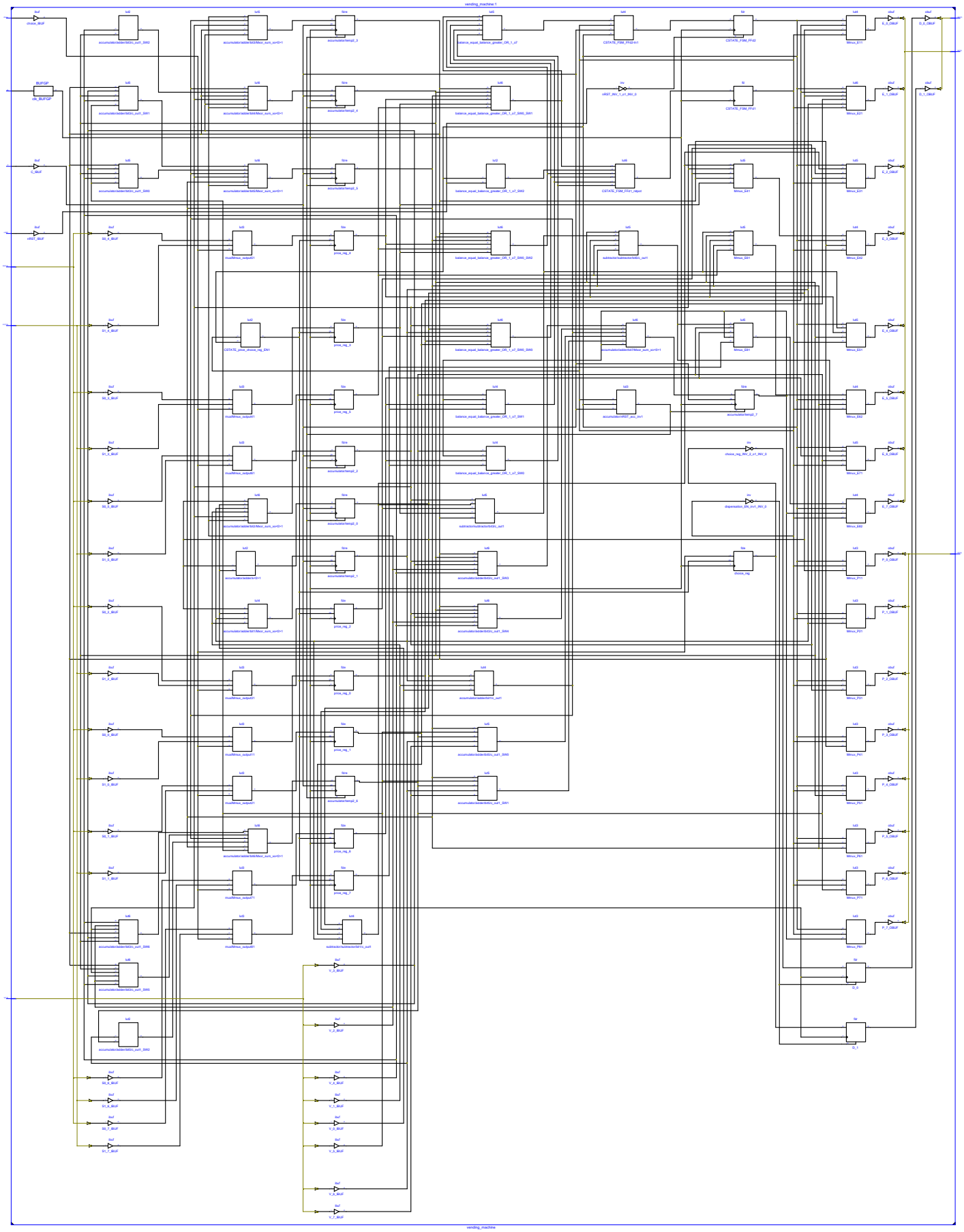
vending_machine Synthesize & RTL Schematic:



vending_machine Synthesize & RTL Schematic:



vending_machine Technology Schematic:

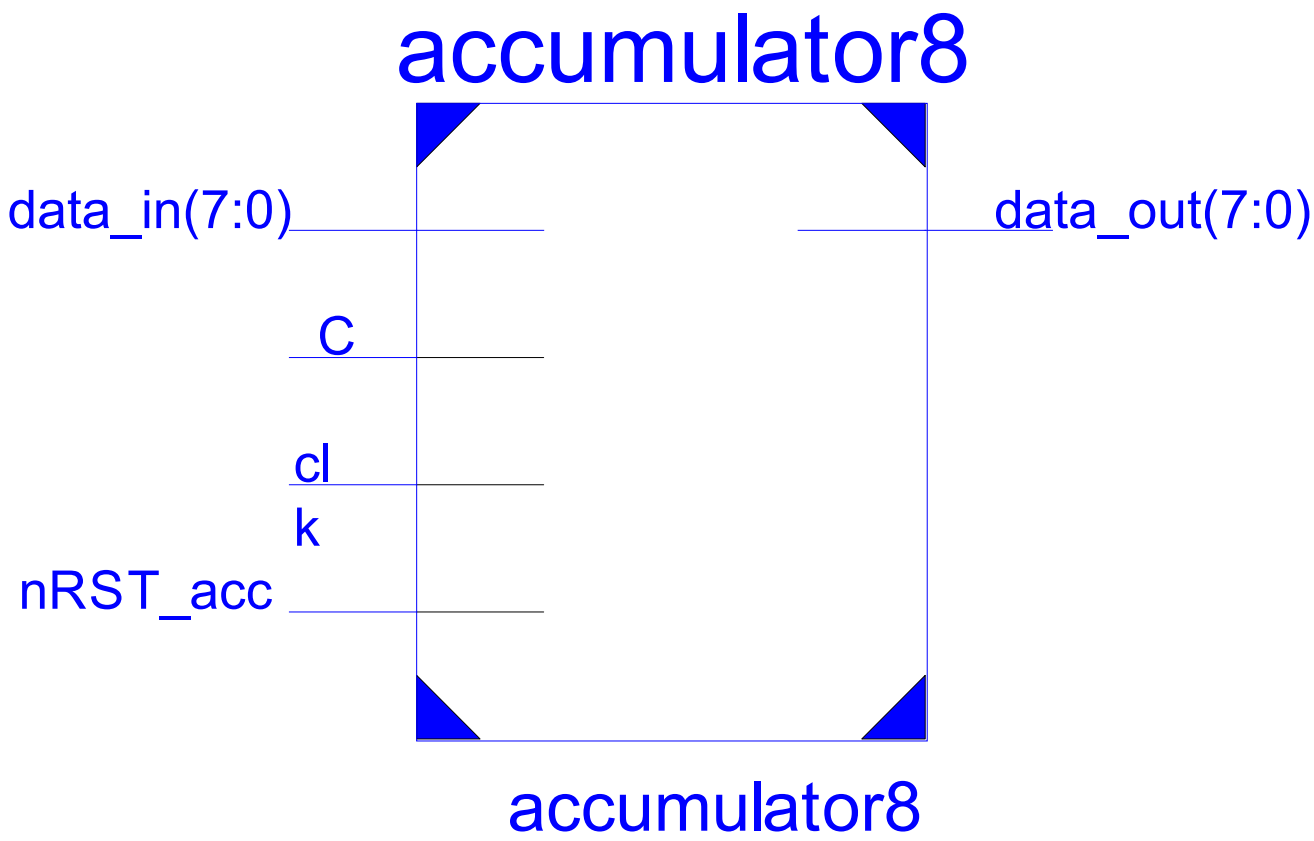


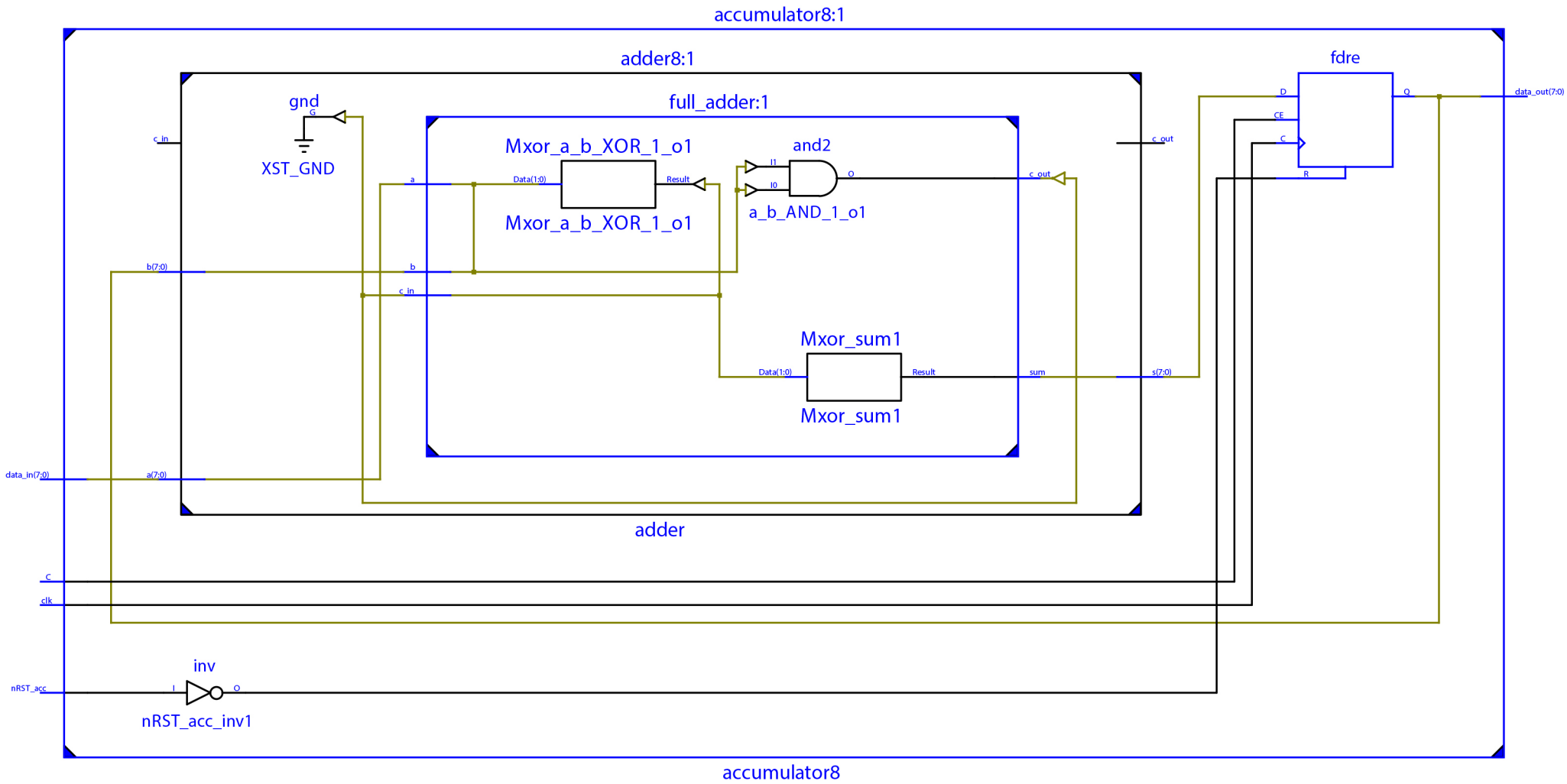
accumulator8:

VHDL CODE

```
1   -- Engineer: Mohammad Niknam
2   -- Project Name:  vending_machine_processor
3   -- Module Name:  accumulator8
4   library ieee;
5   USE ieee.std_logic_1164.all;
6   use IEEE.NUMERIC_STD.ALL;
7
8   entity accumulator8 is
9       port(
10          clk : in std_logic;
11          nRST_acc : in std_logic;
12          C : in std_logic;   --becomes 1 when a coin is detected.
13          data_in : in std_logic_vector(7 downto 0);
14          data_out : out std_logic_vector(7 downto 0)
15          );
16  end accumulator8;
17
18  architecture rtl of accumulator8 is
19  component adder8 IS
20    port(
21      a : in std_logic_vector(7 downto 0);
22      b : in std_logic_vector(7 downto 0);
23      c_in: in std_logic;
24      s: out std_logic_vector(7 downto 0);
25      c_out: out std_logic
26      );
27  end component;
28
29  signal temp1 : std_logic_vector(7 downto 0);
30  signal temp2 : std_logic_vector(7 downto 0);
31  signal c_in : std_logic;
32  signal c_out : std_logic;
33
34  begin
35      c_in <= '0';
36
37      adder : adder8 port map (data_in, temp2, c_in, temp1, c_out);
38
39      reg: process(clk)
40      begin
41          if (clk'event and clk = '1') then
42              if nRST_acc = '0' then
43                  temp2 <= (others => '0');
44              elsif (C = '1') then
45                  temp2 <= temp1;
46              end if;
47          end if;
48      end process reg;
49
50      data_out <= temp2;
51  end rtl;
52
```

accumulator8 Synthesize & RTL Schematic:





Adder8:

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  vending_machine_processor
-- Module Name:  Adder8
library ieee;
USE ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

ENTITY adder8 IS
  PORT(
    a : in std_logic_vector(7 downto 0);
    b : in std_logic_vector(7 downto 0);
    c_in: in std_logic;
    s: out std_logic_vector(7 downto 0);
    c_out: out std_logic);
end adder8;

architecture model of adder8 is
signal c: std_logic_vector(7 downto 1) := (others => '0');
    COMPONENT full_adder
      PORT(
         a : IN STD_LOGIC;
         b : IN STD_LOGIC;
         c_in : IN STD_LOGIC;
         sum : OUT STD_LOGIC;
         c_out : OUT STD_LOGIC);
    END COMPONENT;
BEGIN
    bit0 : full_adder PORT MAP (a(0),b(0),c_in,s(0),c(1));
    bit1 : full_adder PORT MAP (a(1),b(1),c(1),s(1),c(2));
    bit2 : full_adder PORT MAP (a(2),b(2),c(2),s(2),c(3));
    bit3 : full_adder PORT MAP (a(3),b(3),c(3),s(3),c(4));
    bit4 : full_adder PORT MAP (a(4),b(4),c(4),s(4),c(5));
    bit5 : full_adder PORT MAP (a(5),b(5),c(5),s(5),c(6));
    bit6 : full_adder PORT MAP (a(6),b(6),c(6),s(6),c(7));
    bit7 : full_adder PORT MAP (a(7),b(7),c(7),s(7),c_out);
END model;
```

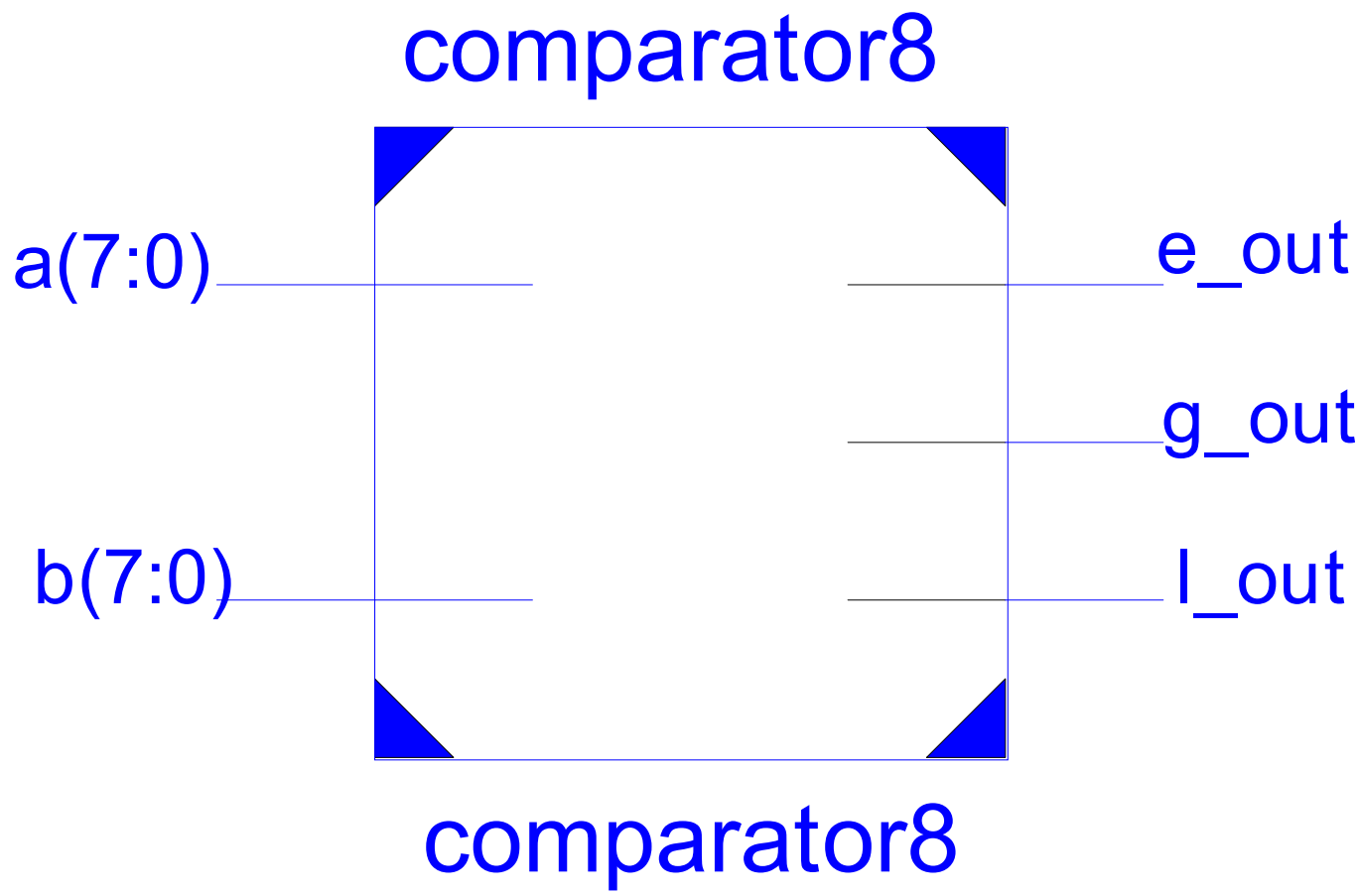
full_Adder:

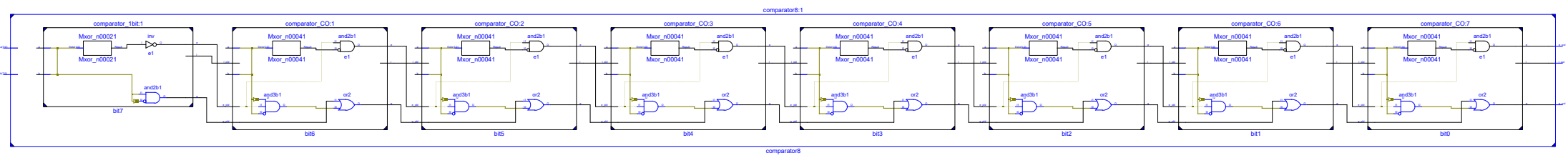
```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  vending_machine_processor
-- Module Name:  full_adder
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

ENTITY full_adder IS
  PORT(
    a : IN STD_LOGIC;
    b : IN STD_LOGIC;
    c_in : IN STD_LOGIC;
    sum : OUT STD_LOGIC;
    c_out : OUT STD_LOGIC);
END full_adder;

ARCHITECTURE model OF full_adder IS
BEGIN
 sum <= a XOR b XOR c_in;
 c_out <= (a AND b) OR (c_in AND (a XOR b));
END model;

```

Comparator8:

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  vending_machine_processor
-- Module Name:  comparator8- structural
library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

entity comparator8 is
    port(
        a : in std_logic_vector(7 downto 0);
        b : in std_logic_vector(7 downto 0);
        g_out : out std_logic;
        e_out : out std_logic;
        l_out : out std_logic);
end comparator8;

architecture structural of comparator8 is
    signal g : std_logic_vector(7 downto 0);
    signal e : std_logic_vector(7 downto 0);
    signal l : std_logic_vector(7 downto 0);

    component comparator_1bit
    port(
        a : in std_logic;
        b : in std_logic;
        g : out std_logic;
        e : out std_logic;
        l : out std_logic);
    end component;

    component comparator_CO  --comparator_consider_order
    port(
        g_old : in std_logic;
        e_old : in std_logic;
        l_old : in std_logic;
        a : in std_logic;
        b : in std_logic;
        g : out std_logic;
        e : out std_logic;
        l : out std_logic);
    end component;

begin
    bit7 : comparator_1bit port map (a(7), b(7), g(6), e(6), l(6));
    bit6 : comparator_CO port map (g(6), e(6), l(6), a(6), b(6), g(5), e(5), l(5));
    bit5 : comparator_CO port map (g(5), e(5), l(5), a(5), b(5), g(4), e(4), l(4));
    bit4 : comparator_CO port map (g(4), e(4), l(4), a(4), b(4), g(3), e(3), l(3));
    bit3 : comparator_CO port map (g(3), e(3), l(3), a(3), b(3), g(2), e(2), l(2));
    bit2 : comparator_CO port map (g(2), e(2), l(2), a(2), b(2), g(1), e(1), l(1));
    bit1 : comparator_CO port map (g(1), e(1), l(1), a(1), b(1), g(0), e(0), l(0));
    bit0 : comparator_CO port map (g(0), e(0), l(0), a(0), b(0), g_out, e_out, l_out);

end structural;
```

Comparator8 Synthesize & RTL Schematic:





Comparator C comparator_consider_order

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  vending_machine_processor
-- Module Name:  comparator_CO: comparator_consider_order - structural
library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

entity comparator_CO is  --comparator_consider_order
    port(
        g_old : in std_logic;
        e_old : in std_logic;
        l_old : in std_logic;
        a : in std_logic;
        b : in std_logic;
        g : out std_logic;
        e : out std_logic;
        l : out std_logic);
end comparator_CO;

architecture structural  of comparator_CO is
begin
    g <= (a and (not b) and e_old) or g_old;
    e <= (a xnor b) and e_old;
    l <= ((not a) and b and e_old) or l_old;
end structural ;

```

Comparator t:

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name:  vending_machine_processor
-- Module Name:  comparator_1bit- structural
library ieee;
use ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

entity comparator_1bit is
    port(
        a : in std_logic;
        b : in std_logic;
        g : out std_logic;
        e : out std_logic;
        l : out std_logic);
end comparator_1bit;

architecture structural of comparator_1bit is
begin
    g <= (a and (not b));
    e <= (a xnor b);
    l <= ((not a) and b);
end structural;

```

subtractor8

```
1   -- Engineer: Mohammad Niknam
2   -- Project Name:  vending_machine_processor
3   -- Module Name:  subtractor8 - structural
4   library ieee;
5   USE ieee.std_logic_1164.all;
6   use IEEE.NUMERIC_STD.ALL;
7
8   entity subtractor8 is
9       port(
10          a : in std_logic_vector(7 downto 0);
11          b : in std_logic_vector(7 downto 0);
12          result : out std_logic_vector(7 downto 0)
13          );
14  end subtractor8;
15
16  architecture structural of subtractor8 is
17  component adder8 IS
18    port(
19      a : in std_logic_vector(7 downto 0);
20      b : in std_logic_vector(7 downto 0);
21      c_in: in std_logic;
22      s: out std_logic_vector(7 downto 0);
23      c_out: out std_logic
24      );
25  end component;
26
27  signal c_in : std_logic;
28  signal c_out : std_logic;
29  signal not_b : std_logic_vector(7 downto 0);
30
31  begin
32      not_b <= (not b);
33      c_in <= '1';
34      subtractor : adder8 port map (a, not_b, c_in, result, c_out);
35  end structural;
36
```

mux21

```vhdl
-- Engineer: Mohammad Niknam
-- Project Name: vending_machine_processor
-- Module Name:  mux21
library ieee;
USE ieee.std_logic_1164.all;
use IEEE.NUMERIC_STD.ALL;

entity mux21 is
    port(
        A : in std_logic_vector(7 downto 0);
        B : in std_logic_vector(7 downto 0);
        s : in std_logic;
        output : out std_logic_vector(7 downto 0)
        );
end mux21;

architecture model of mux21 is
begin
    output <= A when (s = '0') else
                B when (s = '1');
end model;
```

Reference:

https://github.com/MohammadNiknam17/vending_machine_processor