

# VHDL Code

## Simulation & Implementation

### in ISE Xilinx

## **vending machine processor:**

contains: package, accumulator8, Adder8. full\_Adder,  
comparator8, comparator\_1bit, comparator\_co, mux21,  
subtractor8

**version 1.0 : Designed Structural-Modular, based on components**

**version 2.0 : Designed Behavioral-Package used, based on procedures-functions**

**Mohammad Niknam**

**Reference:**

[https://github.com/MohammadNiknam17/vending\\_machine\\_processor](https://github.com/MohammadNiknam17/vending_machine_processor)

## Block Diagram:

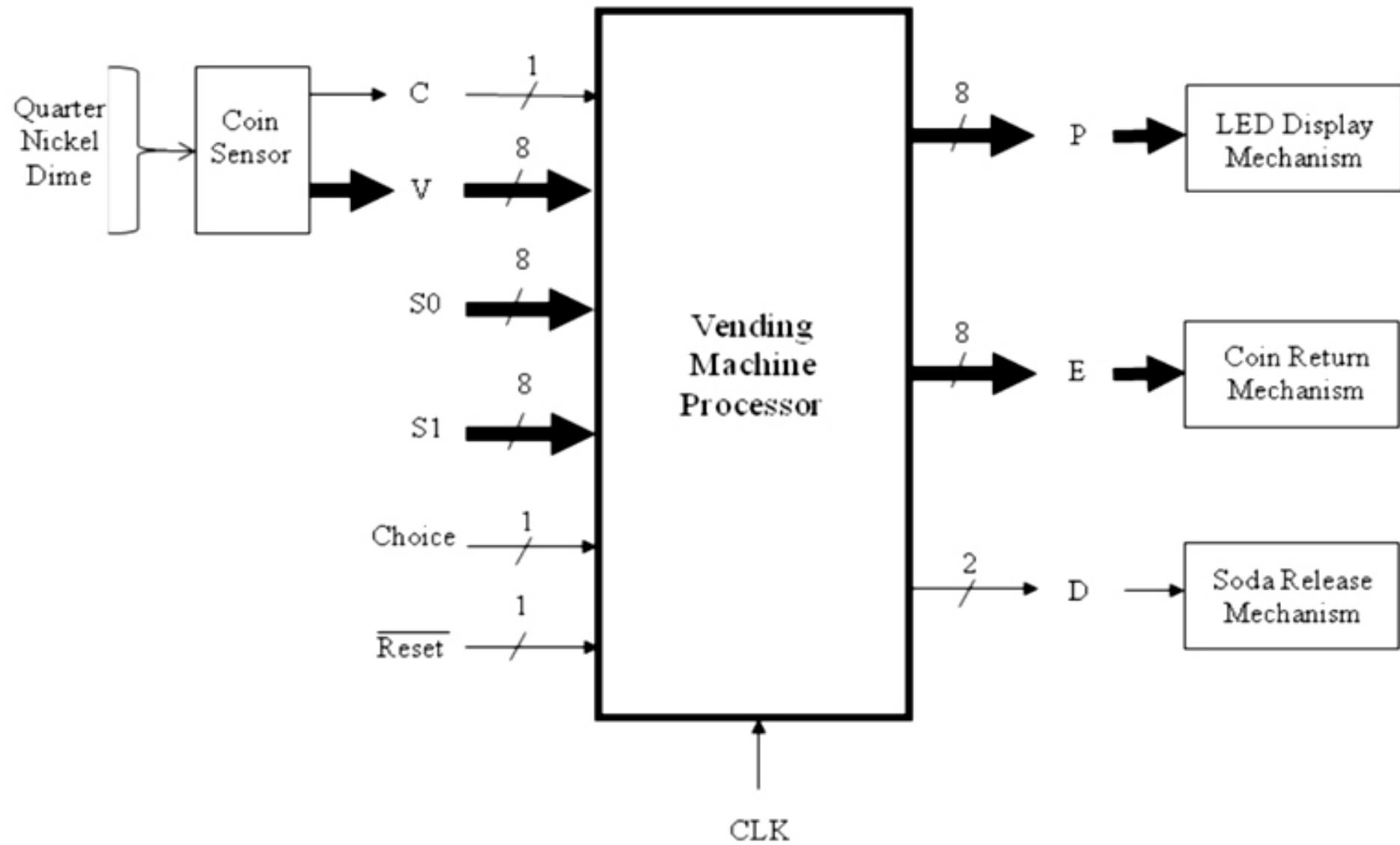


Figure 1

# Project Description:

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 S0 and S1 indicate the cost of the two types of soda drink. Note that the value of S0 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 CODE

```
1 -- Engineer: Mohammad Niknam
2 -- Project Name: vending_machine_processor
3 -- Module Name: vending_machine - rtl
4 library ieee;
5 USE ieee.std_logic_1164.all;
6 use IEEE.NUMERIC_STD.ALL;
7
8 entity vending_machine is
9 port(
10     nRST : in std_logic; --negative reset
11     clk : in std_logic;
12     C : in std_logic; --from coin sensor: becomes 1 when a coin is detected.
13     V : in std_logic_vector(7 downto 0); --received coin's value in cents from
14     coin sensor.
15     S0 : in std_logic_vector(7 downto 0); --price of choice 0, defined by
16     vending machin owner.
17     S1 : in std_logic_vector(7 downto 0); --price of choice 1, defined by
18     vending machin owner.
19     choice : in std_logic;
20     P : out std_logic_vector(7 downto 0); --Display
21     E : out std_logic_vector(7 downto 0); --return change
22     D : out std_logic_vector(1 downto 0) --soda dispensation
23 );
24 end vending_machine;
25
26 architecture rtl of vending_machine is
27 component accumulator8 is
28 port(
29     clk: in std_logic;
30     nRST_acc: in std_logic;
31     C : in std_logic; --becomes 1 when a coin is detected.
32     data_in : in std_logic_vector(7 downto 0);
33     data_out : out std_logic_vector(7 downto 0)
34 );
35 end component;
36
37 component subtractor8 is
38 port(
39     a : in std_logic_vector(7 downto 0);
40     b : in std_logic_vector(7 downto 0);
41     result : out std_logic_vector(7 downto 0)
42 );
43 end component;
44
45 component comparator8 is
46 port(
47     a : in std_logic_vector(7 downto 0);
48     b : in std_logic_vector(7 downto 0);
49     g_out : out std_logic;
50     e_out : out std_logic;
51     l_out : out std_logic);
52 end component;
53
54 component mux21 is
55 port(
56     A : in std_logic_vector(7 downto 0);
57     B : in std_logic_vector(7 downto 0);
58     s : in std_logic;
59     output : out std_logic_vector(7 downto 0)
60 );
61 end component;
62
63 type FSMTYPE is (INIT_STATE, Coin_Reception, soda_dispensation);
64
65 signal CSTATE, NSTATE : FSMTYPE;
66 signal balance, price, price_reg, coins_to_return : std_logic_vector(7 downto 0);
67 signal price_choice_reg_EN, balance_greater, balance_equal, balance_lower: std_logic;
68 signal choice_reg, dispensation_EN : std_logic;
69 signal nRST_acc : std_logic;
```

# vending\_machine:

VHDL CODE

```
71 begin
72     price_registration : process( CLK )
73     begin
74         if (CLK'event and CLK = '1') then
75             if (price_choice_reg_EN = '1') then
76                 price_reg <= price;
77                 choice_reg <= choice;
78             end if ;
79         end if ;
80     end process ; -- price_registration
81
82
83 state_registration : process( CLK )
84 begin
85     if (CLK'event and CLK = '1') then
86         if (nRST = '0') then
87             CSTATE <= INIT_STATE;
88         else
89             CSTATE <= NSTATE;
90         end if ;
91     end if ;
92 end process ; -- state_registration
93
94 soda_dispensation_proc: process(clk)
95 begin
96     if (CLK'event and CLK = '1') then
97         if (dispensation_EN = '1') then
98             if(choice_reg = '0') then
99                 D <= "01"; --S0
100            else
101                D <= "10"; --S1
102            end if;
103        else
104            D <= "00";
105        end if;
106    end if;
107 end process; --soda_dispensation_Proc;
108
109 next_state : process( CSTATE, balance, C, balance_equal, balance_greater,
110 coins_to_return)
111 begin
112     NSTATE <= CSTATE;
113     nRST_acc <= '1';
114     price_choice_reg_EN <= '0';
115     dispensation_EN <= '0';
116     p <= (others => '0');
117     E <= (others => '0');
118
119     case( CSTATE ) is
120         when INIT_STATE =>
121             nRST_acc <= '0';
122             price_choice_reg_EN <= '1';
123             E <= balance;
124             if (C = '1') then
125                 nRST_acc <= '1';
126                 NSTATE <= Coin_Reception;
127             end if ;
128
129         when Coin_Reception =>
130             P <= balance;
131             if (balance_equal = '1' or balance_greater = '1') then
132                 NSTATE <= soda_dispensation ;
133             end if ;
134
135         when soda_dispensation =>
136             dispensation_EN <= '1';
137             E <= coins_to_return;
138             nRST_acc <= '0';
139             NSTATE <= INIT_STATE;
140
141             when others =>
142         end case ;
143     end process ; -- next_state
```

## vending\_machine:

VHDL CODE

```
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,
150     balance_equal, balance_lower);
151     subtractor : subtractor8 port map (balance, price_reg, coins_to_return);
152 end rtl;
153
154
```

# vending\_machine\_TB:

VHDL CODE

```
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=
47     33 cent
47     constant S1price : std_logic_vector(7 downto 0) := "01000001"; --soda S1 price=
48     65 cent
49
50     constant Nickel : std_logic_vector(7 downto 0) := "00000101"; --us_coin value= 5
51     cent
52     constant Dime : std_logic_vector(7 downto 0) := "00001010"; --us_coin value= 10
53     cent
54     constant Quarter : std_logic_vector(7 downto 0) := "00011001"; --us_coin value=
55     25 cent
56
56 BEGIN
57     -- Instantiate the Unit Under Test (UUT)
58     uut: vending_machine PORT MAP (
59         nRST => nRST,
60         clk => clk,
61         C => C,
62         V => V,
63         S0 => S0,
64         S1 => S1,
65         choice => choice,
66         P => P,
67         E => E,
68         D => D
69     );

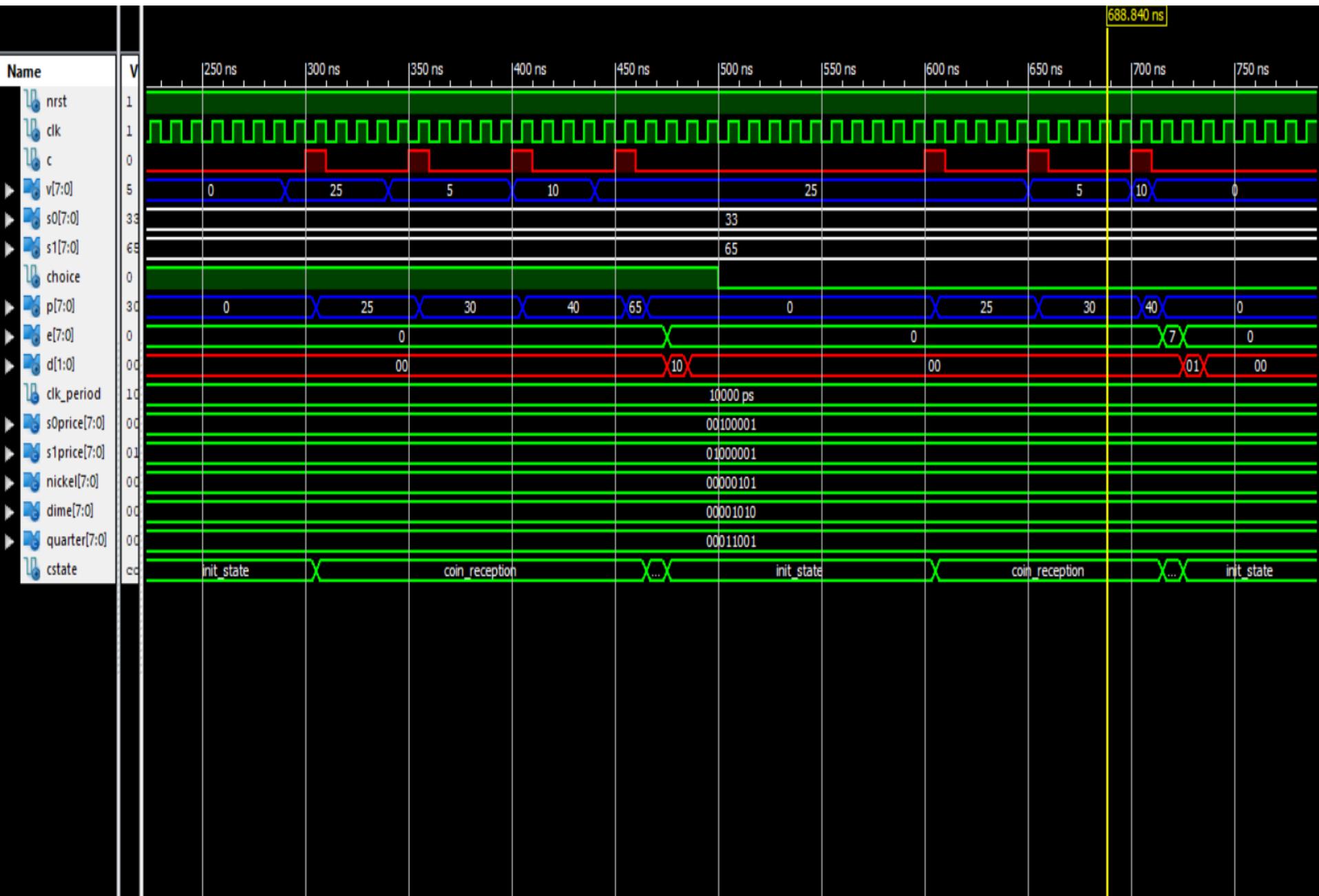
```

## vending\_machine\_TB:

VHDL CODE

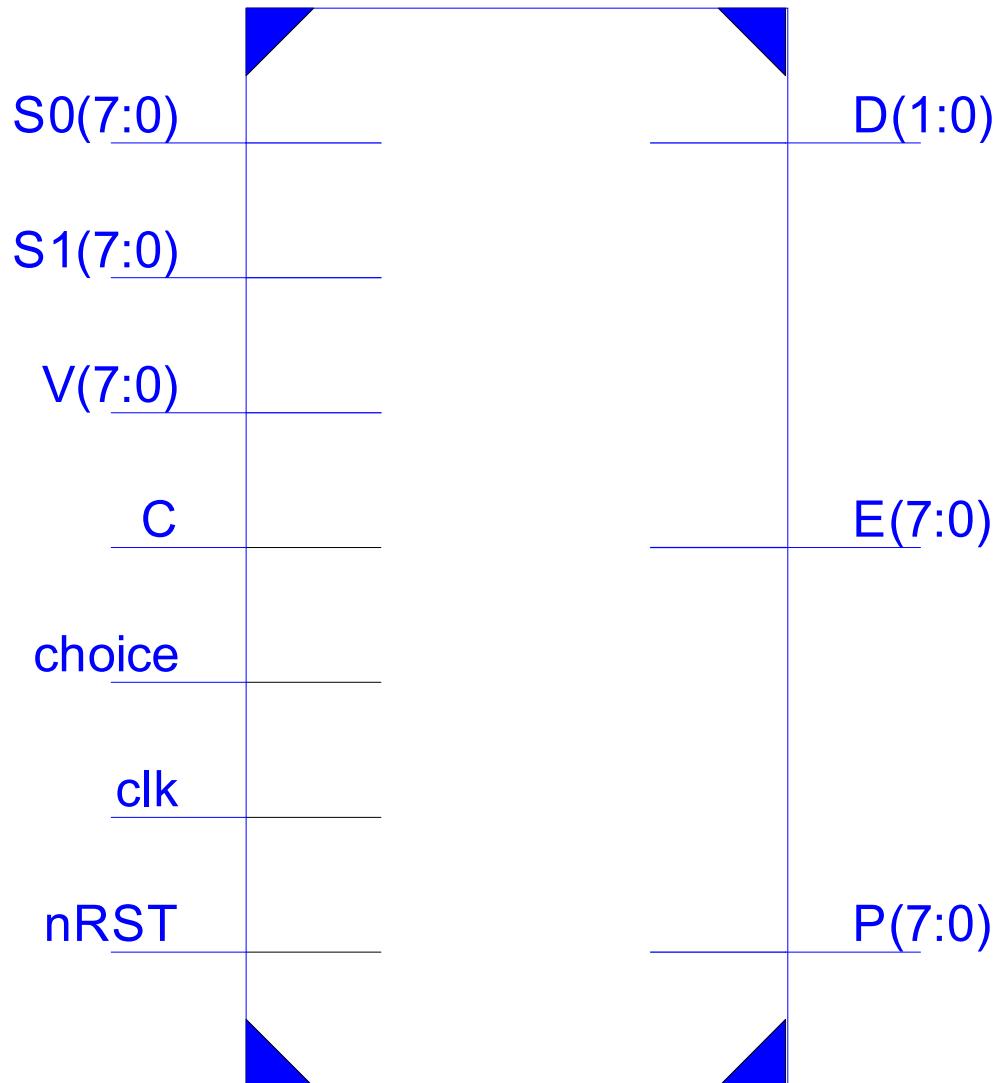
```
77 CLK <= not(CLK) after clk_period/2;
78 nRST <= '0', '1' after 150 ns; -- hold reset state for 400 ns.
79 s0 <= S0price; --soda S0 price= 33 cent
80 S1 <= S1price; --soda S1 price= 65 cent
81 choice <= '1', '0' after 500 ns;
82 C <= '0', '1' after 300 ns, '0' after 310 ns, '1' after 350ns, '0' 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
88 ns, Quarter after 440 ns,
89 Quarter after 600 ns, Nickel after 650 ns, Dime after 700 ns, (others => '0')
90           after 710 ns;
91
92 END;
```

# vending\_machine Simulation:



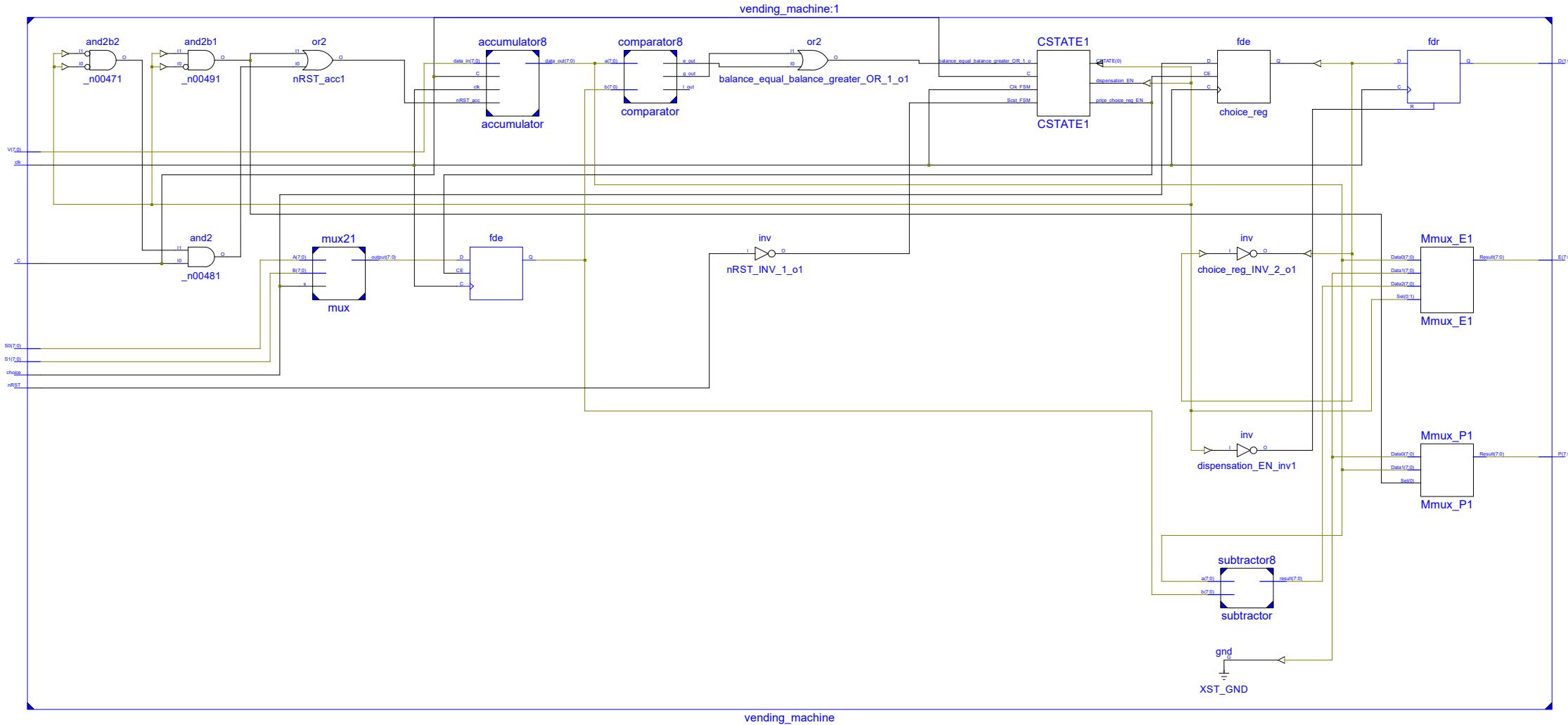
## vending\_machine Synthesize & RTL Schematic:

# vending\_machine

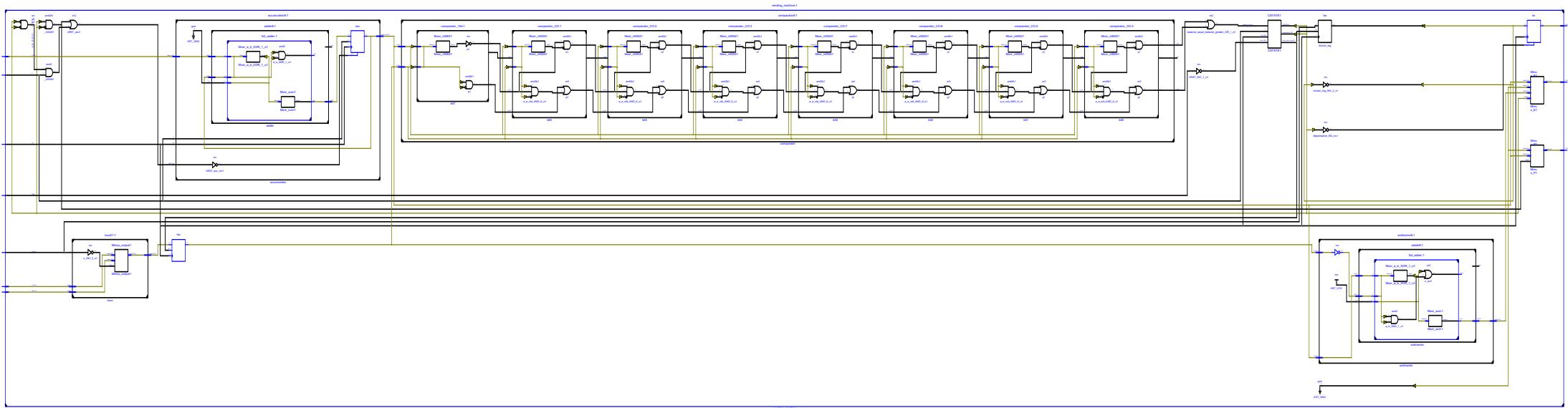


# vending\_machine

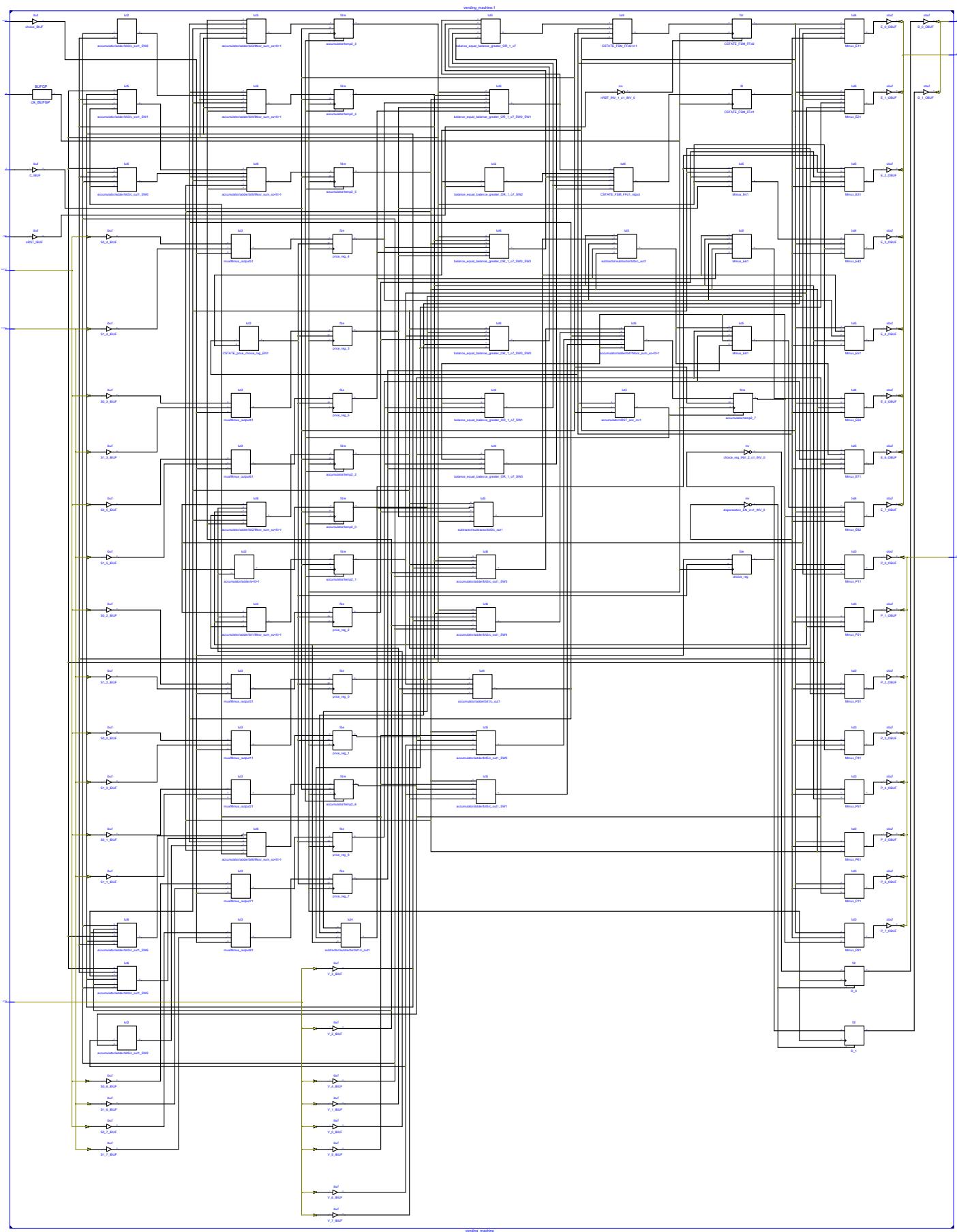
# vending\_machine Synthesize & RTL Schematic:



# vending\_machine Synthesize & RTL Schematic:



# vending\_machine Technology Schematic:

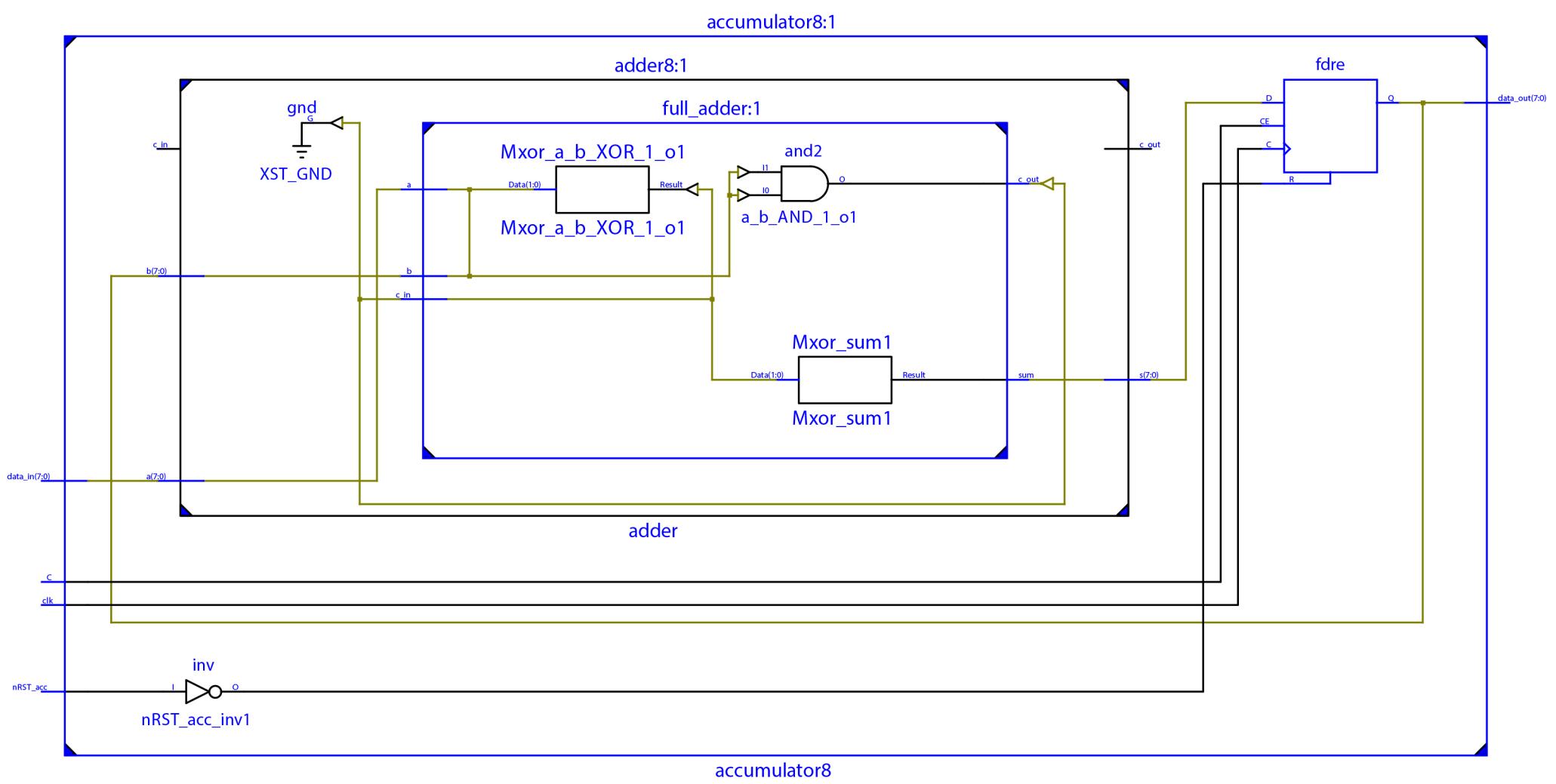
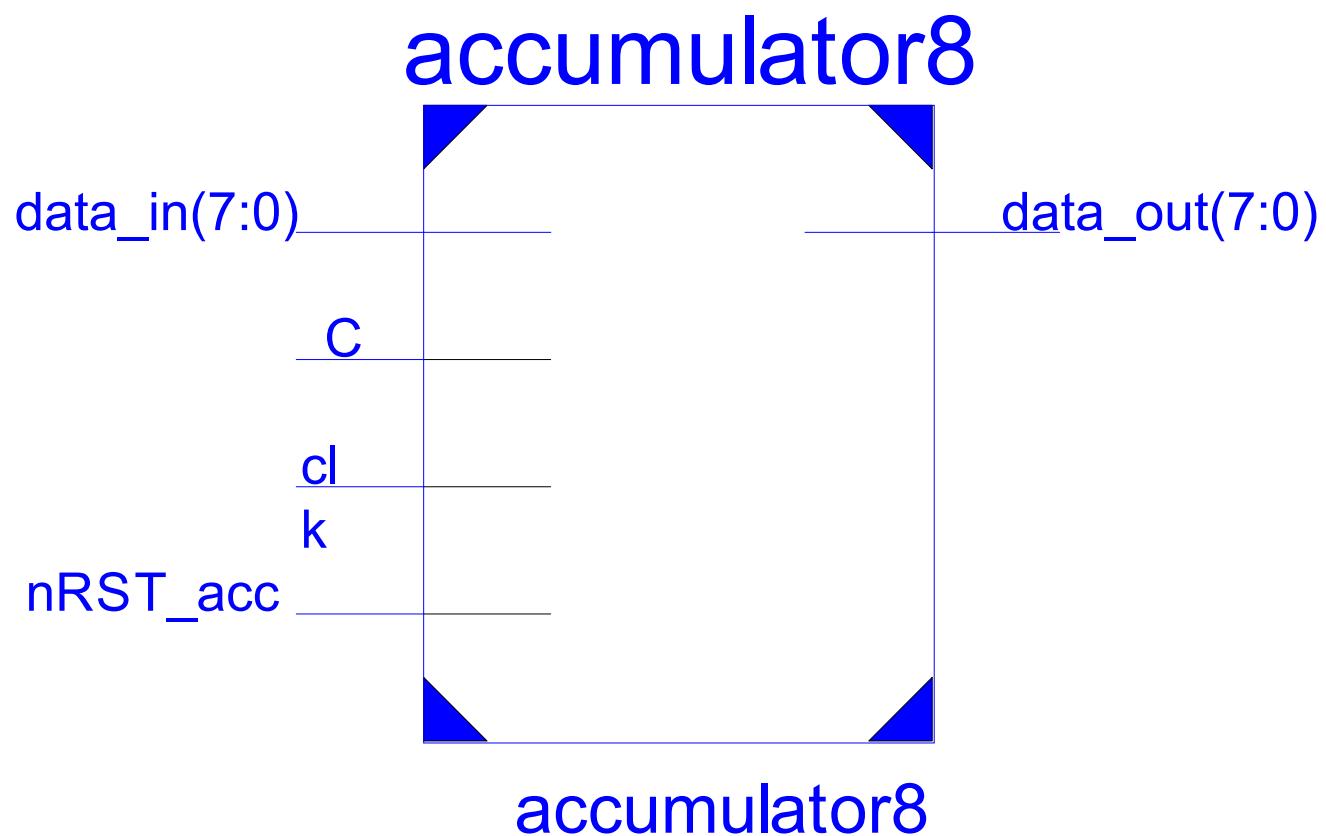


# accumulator8:

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 CODE

```
1 -- Engineer: Mohammad Niknam
2 -- Project Name: vending_machine_processor
3 -- Module Name: Adder8
4 library ieee;
5 USE ieee.std_logic_1164.all;
6 use IEEE.NUMERIC_STD.ALL;
7
8 ENTITY adder8 IS
9 PORT(
10    a : in std_logic_vector(7 downto 0);
11    b : in std_logic_vector(7 downto 0);
12    c_in: in std_logic;
13    s: out std_logic_vector(7 downto 0);
14    c_out: out std_logic);
15 end adder8;
16
17 architecture model of adder8 is
18 signal c: std_logic_vector(7 downto 0) := (others => '0');
19 COMPONENT full_adder
20 PORT(
21    a : IN STD_LOGIC;
22    b : IN STD_LOGIC;
23    c_in : IN STD_LOGIC;
24    sum : OUT STD_LOGIC;
25    c_out : OUT STD_LOGIC);
26 END COMPONENT;
27 BEGIN
28    bit0 : full_adder PORT MAP (a(0),b(0),c_in,s(0),c(1));
29    bit1 : full_adder PORT MAP (a(1),b(1),c(1),s(1),c(2));
30    bit2 : full_adder PORT MAP (a(2),b(2),c(2),s(2),c(3));
31    bit3 : full_adder PORT MAP (a(3),b(3),c(3),s(3),c(4));
32    bit4 : full_adder PORT MAP (a(4),b(4),c(4),s(4),c(5));
33    bit5 : full_adder PORT MAP (a(5),b(5),c(5),s(5),c(6));
34    bit6 : full_adder PORT MAP (a(6),b(6),c(6),s(6),c(7));
35    bit7 : full_adder PORT MAP (a(7),b(7),c(7),s(7),c_out);
36 END model;
```

# full\_Adder:

VHDL CODE

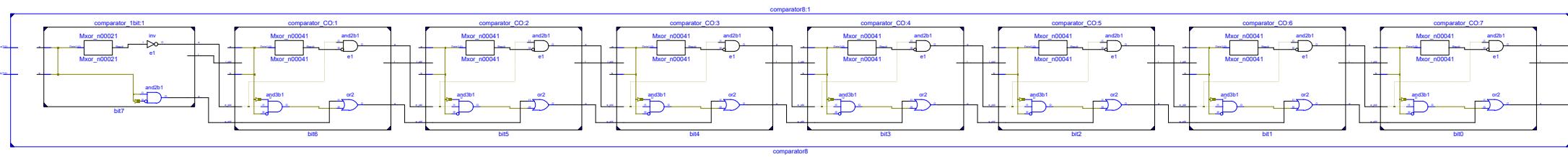
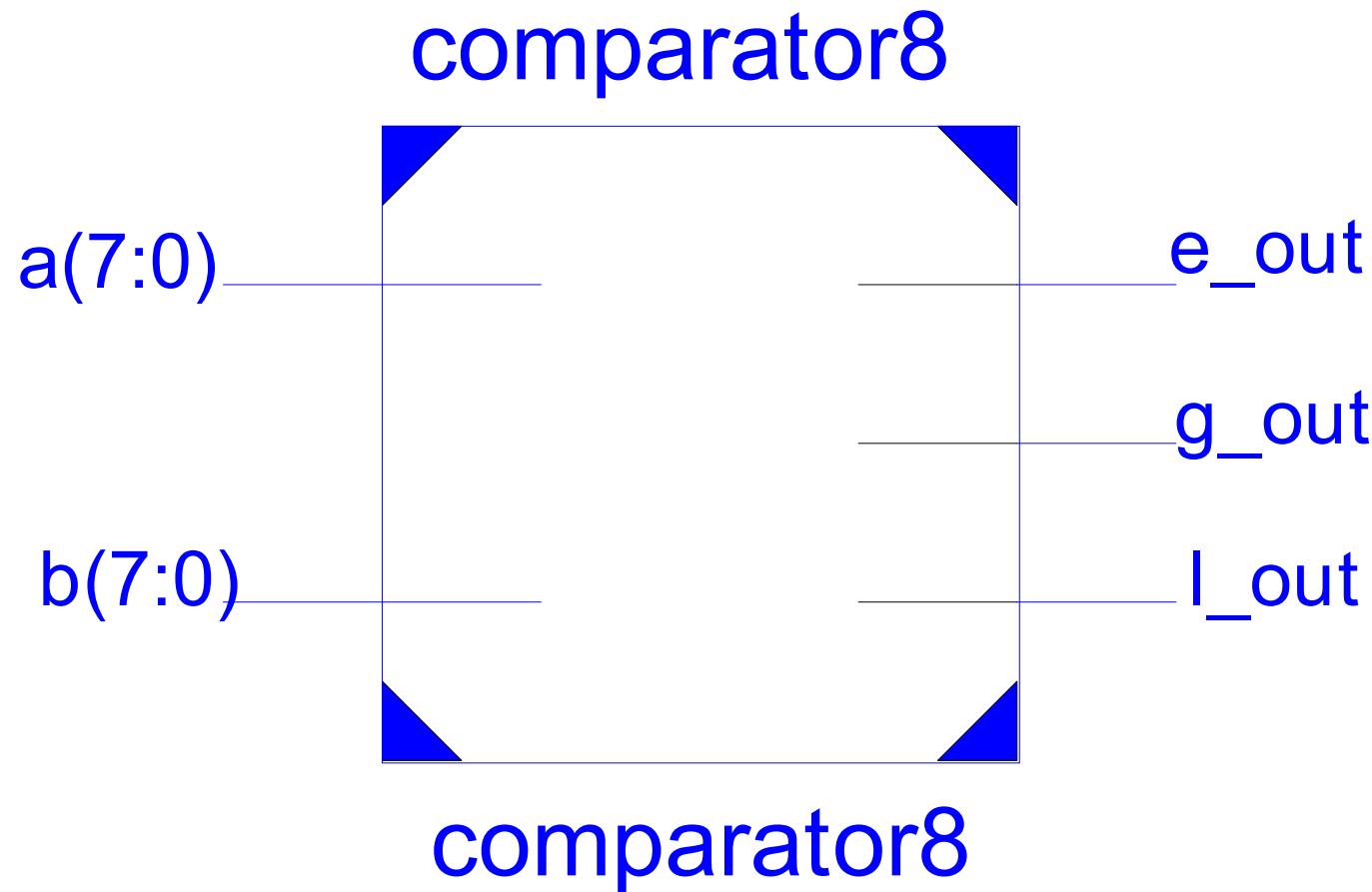
```
1 -- Engineer: Mohammad Niknam
2 -- Project Name: vending_machine_processor
3 -- Module Name: full_adder
4 LIBRARY ieee;
5 USE ieee.std_logic_1164.all;
6 use IEEE.NUMERIC_STD.ALL;
7
8 ENTITY full_adder IS
9 PORT(
10    a : IN STD_LOGIC;
11    b : IN STD_LOGIC;
12    c_in : IN STD_LOGIC;
13    sum : OUT STD_LOGIC;
14    c_out : OUT STD_LOGIC);
15 END full_adder;
16
17 ARCHITECTURE model OF full_adder IS
18 BEGIN
19    sum <= a XOR b XOR c_in;
20    c_out <= (a AND b) OR (c_in AND (a XOR b));
21 END model;
22
```

# Comparator8:

VHDL CODE

```
1 -- Engineer: Mohammad Niknam
2 -- Project Name: vending_machine_processor
3 -- Module Name: comparator8- structural
4 library ieee;
5 use ieee.std_logic_1164.all;
6 use IEEE.NUMERIC_STD.ALL;
7
8 entity comparator8 is
9 port(
10    a : in std_logic_vector(7 downto 0);
11    b : in std_logic_vector(7 downto 0);
12    g_out : out std_logic;
13    e_out : out std_logic;
14    l_out : out std_logic);
15 end comparator8;
16
17 architecture structural of comparator8 is
18 signal g : std_logic_vector(7 downto 0);
19 signal e : std_logic_vector(7 downto 0);
20 signal l : std_logic_vector(7 downto 0);
21
22 component comparator_1bit
23 port(
24    a : in std_logic;
25    b : in std_logic;
26    g : out std_logic;
27    e : out std_logic;
28    l : out std_logic);
29 end component;
30
31 component comparator_CO --comparator_consider_order
32 port(
33    g_old : in std_logic;
34    e_old : in std_logic;
35    l_old : in std_logic;
36    a : in std_logic;
37    b : in std_logic;
38    g : out std_logic;
39    e : out std_logic;
40    l : out std_logic);
41 end component;
42
43 begin
44    bit7 : comparator_1bit port map (a(7), b(7), g(6), e(6), l(6));
45    bit6 : comparator_CO port map (g(6), e(6), l(6), a(6), b(6), g(5), e(5), l(5));
46    bit5 : comparator_CO port map (g(5), e(5), l(5), a(5), b(5), g(4), e(4), l(4));
47    bit4 : comparator_CO port map (g(4), e(4), l(4), a(4), b(4), g(3), e(3), l(3));
48    bit3 : comparator_CO port map (g(3), e(3), l(3), a(3), b(3), g(2), e(2), l(2));
49    bit2 : comparator_CO port map (g(2), e(2), l(2), a(2), b(2), g(1), e(1), l(1));
50    bit1 : comparator_CO port map (g(1), e(1), l(1), a(1), b(1), g(0), e(0), l(0));
51    bit0 : comparator_CO port map (g(0), e(0), l(0), a(0), b(0), g_out, e_out, l_out);
52
53 end structural;
```

# Comparator8 Synthesize & RTL Schematic:



# Comparator\_CO: comparator\_consider\_order

VHDL CODE

```
1 -- Engineer: Mohammad Niknam
2 -- Project Name: vending_machine_processor
3 -- Module Name: comparator_CO: comparator_consider_order - structural
4 library ieee;
5 use ieee.std_logic_1164.all;
6 use IEEE.NUMERIC_STD.ALL;
7
8 entity comparator_CO is --comparator_consider_order
9 port(
10     g_old : in std_logic;
11     e_old : in std_logic;
12     l_old : in std_logic;
13     a : in std_logic;
14     b : in std_logic;
15     g : out std_logic;
16     e : out std_logic;
17     l : out std_logic);
18 end comparator_CO;
19
20 architecture structural of comparator_CO is
21 begin
22     g <= (a and (not b) and e_old) or g_old;
23     e <= (a xnor b) and e_old;
24     l <= ((not a) and b and e_old) or l_old;
25 end structural ;
26
```

# Comparator\_1bit:

VHDL CODE

```
1 -- Engineer: Mohammad Niknam
2 -- Project Name: vending_machine_processor
3 -- Module Name: comparator_1bit- structural
4 library ieee;
5 use ieee.std_logic_1164.all;
6 use IEEE.NUMERIC_STD.ALL;
7
8 entity comparator_1bit is
9 port(
10     a : in std_logic;
11     b : in std_logic;
12     g : out std_logic;
13     e : out std_logic;
14     l : out std_logic);
15 end comparator_1bit;
16
17 architecture structural of comparator_1bit is
18 begin
19     g <= (a and (not b));
20     e <= (a xnor b);
21     l <= ((not a) and b);
22 end structural;
23
```

# subtractor8:

VHDL CODE

```
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 CODE

```
1  -- Engineer: Mohammad Niknam
2  -- Project Name: vending_machine_processor
3  -- Module Name: mux21
4  library ieee;
5  USE ieee.std_logic_1164.all;
6  use IEEE.NUMERIC_STD.ALL;
7
8  entity mux21 is
9    port(
10      A : in std_logic_vector(7 downto 0);
11      B : in std_logic_vector(7 downto 0);
12      s : in std_logic;
13      output : out std_logic_vector(7 downto 0)
14    );
15 end mux21;
16
17 architecture model of mux21 is
18 begin
19   output <= A when (s = '0') else
20             B when (s = '1');
21 end model;
```

**version 2.0 :**

**Designed Behavioral-Package used  
based on procedures-functions**

# Version2.0 : vending\_machine:

VHDL CODE

```
1  -- Engineer: Mohammad Niknam
2  -- Project Name: vending_machine_processor
3  -- Module Name: vending_machine - rtl
4  library ieee;
5  USE ieee.std_logic_1164.all;
6  use IEEE.NUMERIC_STD.ALL;
7  library work;
8  use work.my_pack.ALL;
9
10 entity vending_machine is
11 port(
12     nRST : in std_logic;  --negative reset
13     clk : in std_logic;
14     C : in std_logic; --from coin sensor: becomes 1 when a coin is detected.
15     V : in std_logic_vector(7 downto 0); --received coin's value in cents from
16     coin sensor.
17     S0 : in std_logic_vector(7 downto 0); --price of choice 0, defined by
18     vending machin owner.
19     S1 : in std_logic_vector(7 downto 0); --price of choice 1, defined by
20     vending machin owner.
21     choice : in std_logic;
22     P : out std_logic_vector(7 downto 0); --Display
23     E : out std_logic_vector(7 downto 0); --return change
24     D : out std_logic_vector(1 downto 0) --soda dispensation
25 );
26 end vending_machine;
27
28 architecture rtl of vending_machine is
29 type FSMTYPE is (INIT_STATE, Coin_Reception, soda_dispensation);
30 signal CSTATE, NSTATE : FSMTYPE;
31 signal balance, price, price_reg, coins_to_return : std_logic_vector(7 downto 0);
32 signal price_choice_reg_EN, balance_greater, balance_equal, balance_lower: std_logic;
33 signal choice_reg, dispensation_EN : std_logic;
34 signal nRST_acc : std_logic;
35
36 begin
37     price_registration : process( CLK )
38 begin
39         if (CLK'event and CLK = '1') then
40             if (price_choice_reg_EN = '1') then
41                 price_reg <= price;
42                 choice_reg <= choice;
43             end if ;
44         end if ;
45     end process ; -- price_registration
46
47     state_registration : process( CLK )
48 begin
49         if (CLK'event and CLK = '1') then
50             if (nRST = '0') then
51                 CSTATE <= INIT_STATE;
52             else
53                 CSTATE <= NSTATE;
54             end if ;
55         end if ;
56     end process ; -- state_registration
57
58     soda_dispensation_proc: process(clk)
59 begin
60         if (CLK'event and CLK = '1') then
61             if (dispensation_EN = '1') then
62                 if(choice_reg = '0') then
63                     D <= "01"; --S0
64                 else
65                     D <= "10"; --S1
66                 end if;
67             else
68                 D <= "00";
69             end if;
70         end if;
71     end process; --soda_dispensation_Proc;
```

# Version2.0 : vending\_machine:

VHDL CODE

```
78      next_state : process( CSTATE, balance, C, balance_equal, balance_greater,
79      coins_to_return)
80      begin
81          NSTATE <= CSTATE;
82          nRST_acc <= '1';
83          price_choice_reg_EN <= '0';
84          dispensation_EN <= '0';
85          p <= (others => '0');
86          E <= (others => '0');

87          case( CSTATE ) is
88              when INIT_STATE =>
89                  nRST_acc <= '0';
90                  price_choice_reg_EN <= '1';
91                  E <= balance;
92                  if (C = '1') then
93                      nRST_acc <= '1';
94                      NSTATE <= Coin_Reception;
95                  end if ;

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

101             when soda_dispensation =>
102                 dispensation_EN <= '1';
103                 E <= coins_to_return;
104                 nRST_acc <= '0';
105                 NSTATE <= INIT_STATE;

106                 when others =>
107             end case ;
108         end process ; -- next_state
109
110
111
112
113     price <= mux21 (S0, S1, choice);
114     accumulator8 (clk, nRST_acc, C, balance, V, balance);
115     comparator8 (balance, price_reg, balance_greater, balance_equal, balance_lower);
116     coins_to_return <= subtractor8 (balance, price_reg);
117
118 end rtl;
119
120
```

# Version2.0 : my\_pack:

VHDL CODE

```
1  -- Engineer: Mohammad Niknam
2  -- Project Name: vending_machine_processor
3  -- Module Name: my_pack
4
5  library IEEE;
6  use IEEE.STD_LOGIC_1164.ALL;
7  use IEEE.NUMERIC_STD.ALL;
8
9  package my_pack is
10
11    function mux21(
12      signal A, B : std_logic_vector(7 downto 0);
13      signal s : std_logic)
14      return std_logic_vector;
15
16
17    function subtractor8(
18      signal A, B : std_logic_vector(7 downto 0))
19      return std_logic_vector;
20
21
22    procedure comparator8(
23      signal a, b : in std_logic_vector(7 downto 0);
24      signal g_out, e_out, l_out : out std_logic);
25
26
27    procedure accumulator8(
28      signal clk , nRST_acc, C : in std_logic;
29      signal data_in, data_old : in std_logic_vector(7 downto 0);
30      signal data_out : out std_logic_vector(7 downto 0));
31
32
33    procedure register8(
34      signal clk, nRST : in std_logic;
35      signal regin : in std_logic_vector(7 downto 0);
36      signal regout : out std_logic_vector(7 downto 0));
37
38  end my_pack;
39
```

# Version2.0 : my\_pack:

VHDL CODE

```
39 package body my_pack is
40
41     function mux21(
42         signal A, B : std_logic_vector(7 downto 0);
43         signal s : std_logic)
44         return std_logic_vector is
45             variable output : std_logic_vector(7 downto 0);
46         begin
47             case s is
48                 when '0' => output := A;
49                 when others => output := B;
50             end case ;
51             return output;
52     end mux21;
53
54
55     function subtractor8(
56         signal A, B : std_logic_vector(7 downto 0))
57         return std_logic_vector is
58             variable Avar, Bvar : unsigned(7 downto 0);
59             variable result : std_logic_vector(7 downto 0);
60         begin
61             Avar := unsigned(A);
62             Bvar := unsigned(B);
63             Bvar := not(Bvar) + "00000001";
64             result := std_logic_vector(Avar + Bvar);
65             return result;
66     end subtractor8;
67
68
69
70     procedure comparator8(
71         signal a, b : in std_logic_vector(7 downto 0);
72         signal g_out, e_out, l_out : out std_logic) is
73         begin
74             if (a > b) then
75                 g_out <= '1';
76                 e_out <= '0';
77                 l_out <= '0';
78             elsif (a = b) then
79                 g_out <= '0';
80                 e_out <= '1';
81                 l_out <= '0';
82             else
83                 g_out <= '0';
84                 e_out <= '0';
85                 l_out <= '1';
86             end if;
87     end comparator8;
88
89
90     procedure accumulator8(
91         signal clk , nRST_acc, C : in std_logic;
92         signal data_in, data_old : in std_logic_vector(7 downto 0);
93         signal data_out : out std_logic_vector(7 downto 0)) is
94         begin
95             if (clk'event and clk = '1') then
96                 if nRST_acc = '0' then
97                     data_out <= (others => '0');
98                 elsif (C = '1') then
99                     data_out <= std_logic_vector(unsigned(data_old) +
100                         unsigned(data_in));
101                     end if;
102                 end if;
103         end accumulator8;
```

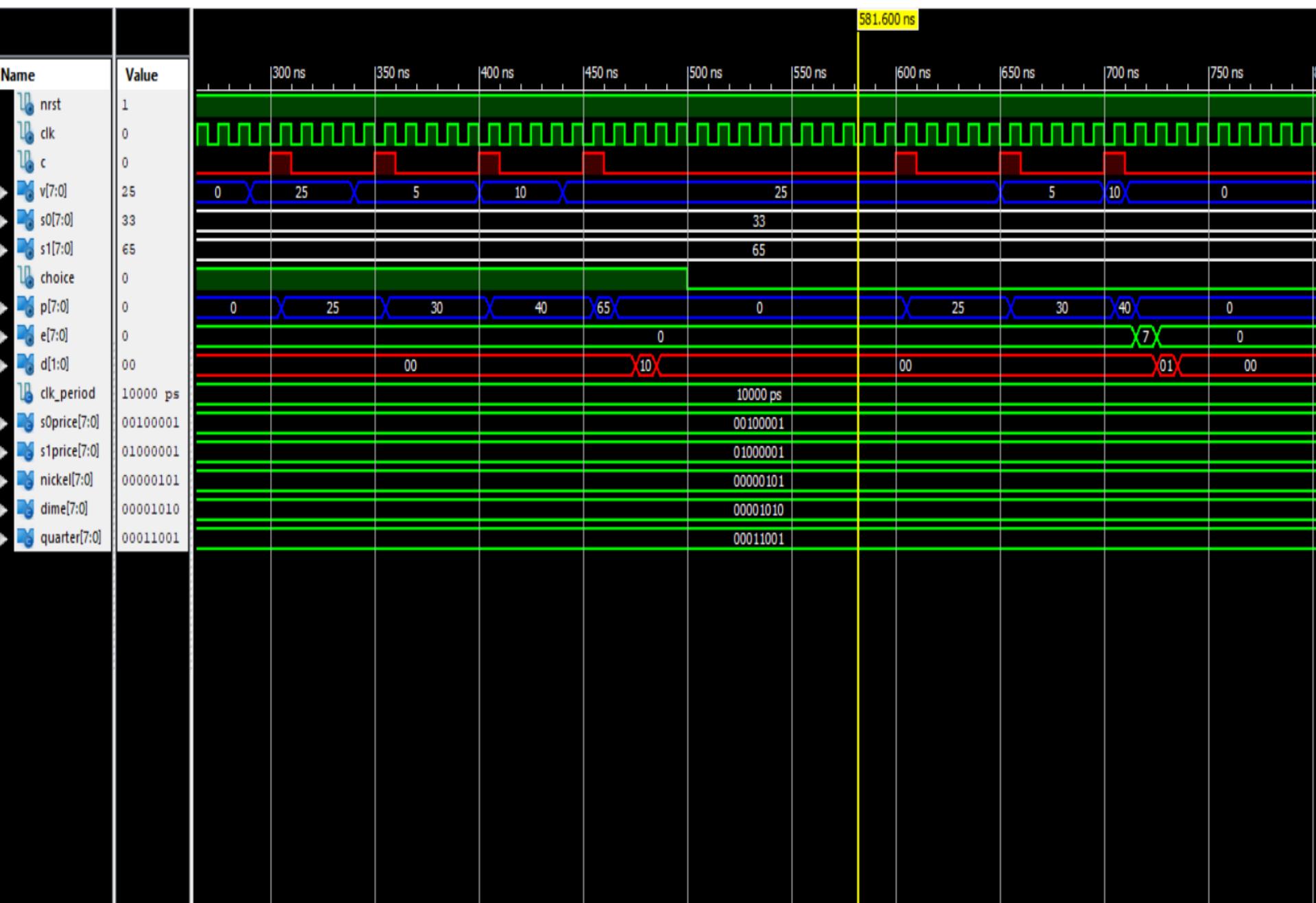
## Version2.0 : my\_pack:

VHDL CODE

```
104  procedure register8(
105    signal clk, nRST: in std_logic;
106    signal regin : in std_logic_vector(7 downto 0);
107    signal regout : out std_logic_vector(7 downto 0)) is
108      begin
109        if (CLK'event and CLK = '1') then
110          if(nRST = '0') then
111            regout <= "00000000";
112          else
113            regout <= regin;
114            end if;
115            end if;
116      end register8;
117
118  end my_pack;
119
120
```

notice: procedure register8 not used in this project.

# Version2.0: vending\_machine Simulation: resulted same as version 1.0



## **Reference:**

[https://github.com/MohammadNiknam17/vending\\_machine\\_processor](https://github.com/MohammadNiknam17/vending_machine_processor)