VHDL Project

ELG 4137: Principles & Application of VLSI Design

Winter 2019

Prof: Dr. Dafu Lou

TA: Shady Mohammad

Group Member:

Baohui Xie (7685551)

Yicong Zhang (7863181)

Experiment Date: 2019-03-20/03-27

Submission Date: 2018-04-05

Table of Contents

Table of figures

Introduction

In this project, a soda vending machine will be designed. There are five entities in our design, which are accumulator (an 8-bits full adder and an 8-bit register), 8-bit multiplexer, 8-bits comparator, Soda selector (output displace), and casher. All the VHDL codes and simulation's results will be demonstrated in the following report.

Hardware/Software

- Computer

- Quartus II (VHDL)

Design Layout

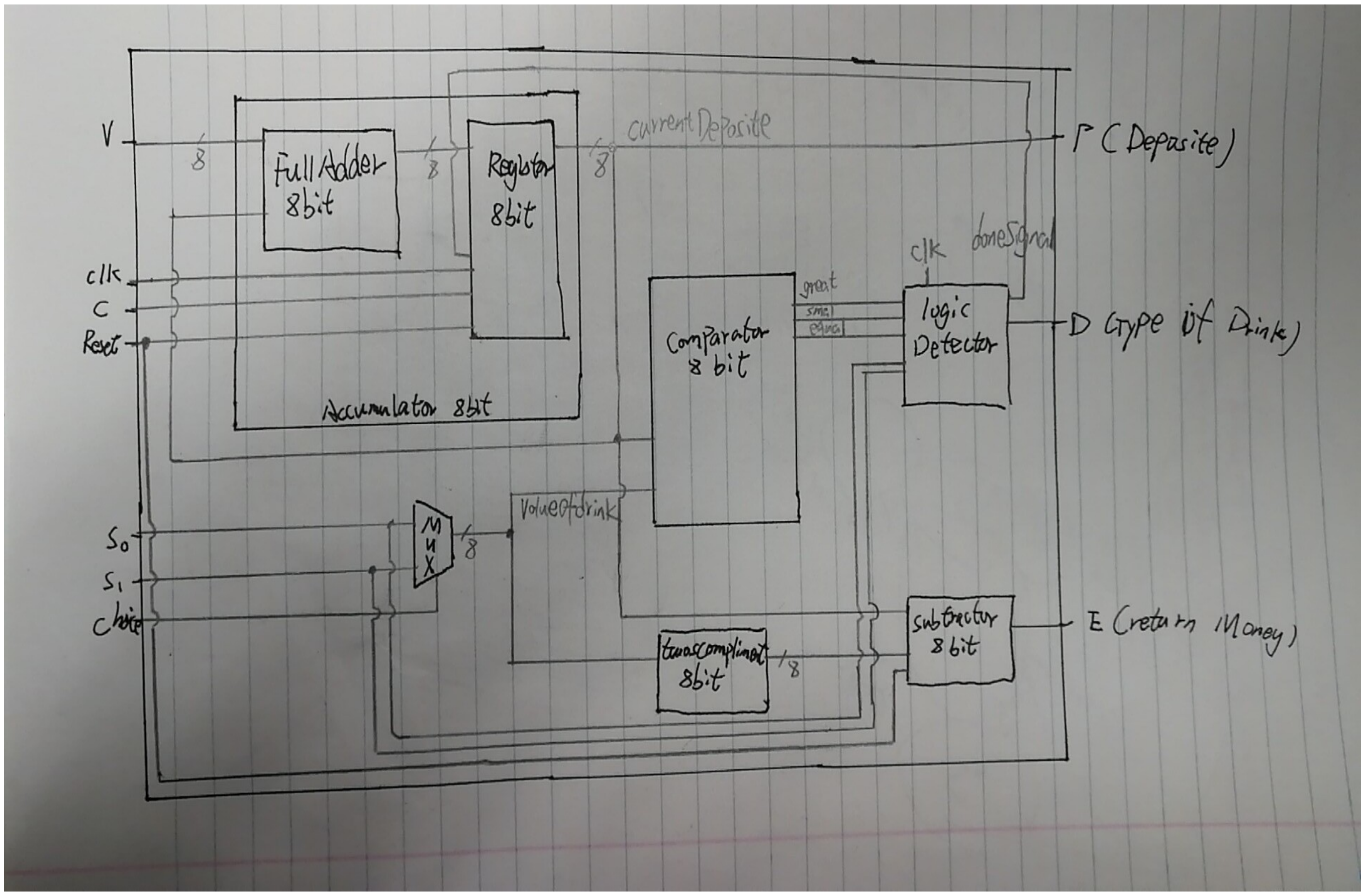

Figure 1. Design Lay out

State Diagram

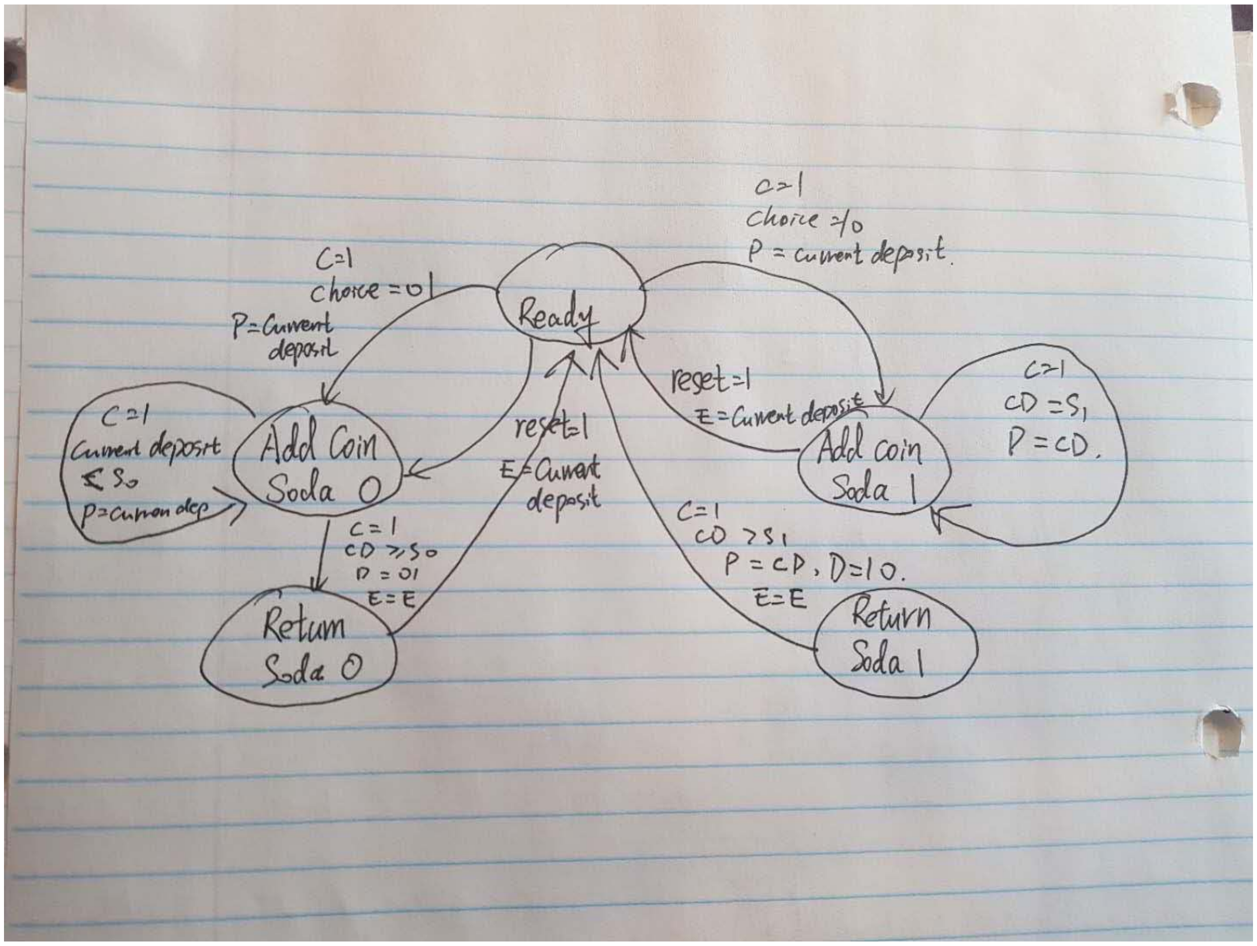

Figure 2. State diagram for the vending machine

Codes

```
-----------------notgate-----------

library IEEE;

use IEEE.std_logic_1164.all;

entity notgate is

        port(i : in STD_LOGIC;

                o : out STD_LOGIC);

end notgate;


architecture behav of notgate is

begin
```

```
	O <= not(I);
end behav;

----------------And2--------------------
library IEEE;
use IEEE.std_logic_1164.all ;
entity and_gate is
	port(I1,  I2:in STD_LOGIC;
		O:out STD_LOGIC);
end and_gate;

architecture behav of and_gate is
begin
	O <= (I1 and I2) after 10 ns;
end behav;

----------------Xor2-------------------
library IEEE ;
use IEEE.std_logic_1164.all ;
entity exor_gate is
	port(I1, I2:in STD_LOGIC;
		O:out STD_LOGIC);
end exor_gate;

architecture behav of exor_gate is
begin
	O <= (I1 xor I2);
end behav;
```

```
------------------or2---------------------
library IEEE ;
use IEEE.std_logic_1164.all ;
entity or_gate is
            port(I1, I2:in STD_LOGIC;
                        O:out STD_LOGIC);
end or_gate;

architecture behav of or_gate is
begin  O <= (I1 or I2);
end behav;

----------------- nand4gate----------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY nand4gate IS PORT(
i1, i2, i3, i4: IN STD_LOGIC;
o: OUT STD_LOGIC);
END nand4gate;
ARCHITECTURE Dataflow OF nand4gate IS
BEGIN
o <= NOT(i1 AND i2 AND i3 AND i4);
END Dataflow;

----------------nand2gate---------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY nand2gate IS PORT(
i1, i2: IN STD_LOGIC;
o: OUT STD_LOGIC);
END nand2gate;
ARCHITECTURE Dataflow OF nand2gate IS
BEGIN
o <= NOT(i1 AND i2);
END Dataflow;

----------------- nor2gate----------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY nor2gate IS PORT(
i1, i2: IN STD_LOGIC;
o: OUT STD_LOGIC);
END nor2gate;
ARCHITECTURE Dataflow OF nor2gate IS
BEGIN
o <= NOT(i1 OR i2);
END Dataflow;



-----------------half_adder----------------
library IEEE;
use IEEE.std_logic_1164.all;
entity half_adder is
        port( A,B :in STD_LOGIC; --input
                Sum, Carry: out STD_LOGIC);--output ports
end half_adder;

architecture structure_view of half_adder is
```

```
component notgate
            port(I : in STD_LOGIC;
                        O : out STD_LOGIC);
end component;
component and_gate
            port(I1,I2 : in STD_LOGIC;
                        O: out STD_LOGIC);
end component;
component or_gate
            port(I1,I2 : in STD_LOGIC;
                        O : out STD_LOGIC);
end component;
signal w,x,y,z : STD_LOGIC;
begin
G1: notgate port map (A,x);
G2: notgate port map (B,y);
G3: and_gate port map (x,B,z);
G4: and_gate port map (A,y,w);
G5: or_gate  port map (z,w,Sum);
G6: and_gate port map (A,B,Carry);
end structure_view;


-------------------Full_adder----------------
library IEEE ;
use IEEE.std_logic_1164.all ;
entity full_adder is
            port ( A,B, Cin: in STD_LOGIC; --input ports
                        Sum, Cout : out STD_LOGIC); --output ports
end full_adder;
```

```
architecture structure_view of full_adder is
component half_adder
            port (A,B: in STD_LOGIC;
                        Sum, Carry: out STD_LOGIC);
end component;
component or_gate
            port(I1,I2 : in STD_LOGIC;
                        O : out STD_LOGIC);
end component;
signal Y0, Z0, Z1 : STD_LOGIC;
begin
            HA0: HALF_ADDER port map(A,B,Y0,Z0);
            HA1: HALF_ADDER port map(Cin,Y0,Sum,Z1);
            OG:  or_gate    port map (Z0, Z1, Cout);
end structure_view;

----------------------------adder8------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity adder8 is
port ( a : in std_logic_vector(7 downto 0);
 b : in std_logic_vector(7 downto 0);
 cin : in std_logic;
 sum : out std_logic_vector(7 downto 0);
 cout : out std_logic);
end adder8;
```

```
architecture structural of adder8 is


component full_adder is
  port (
    a, b, cin : in  std_logic;            -- inputs
    sum, Cout : out std_logic);           -- sum & carry
end component;
signal carry : std_logic_vector(6 downto 0);


begin


U1 : full_adder port map (a(0),b(0),cin,sum(0),carry(0));
U2 : for i in 1 to 6 generate
U3 : full_adder port map (a(i),b(i),carry(i-1),sum(i),carry(i));
end generate;
U4 : full_adder port map (a(7),b(7),carry(6),sum(7),cout);


end structural;


-------------8 bit register------------
library IEEE;
use IEEE.std_logic_1164.all;
entity Register8bit is
            port( regin :in std_logic_vector(7 downto 0);
                        Clk, sync_reset,C,toReset: in std_logic;
                        regout: out std_logic_vector(7 downto 0));
end Register8bit;


architecture behavioral of Register8bit is
```

```
begin
        process(Clk)
        begin
        if rising_edge(Clk) then
                if (sync_reset = '1'or toReset='1')then
                        regout <= (others => '0');
                elsif(C='1') then
                        regout <= regin;
                elsif sync_reset = 'X' or Clk = 'X' then
                        regout <= (others => 'X');
                end if;
        end if;
        end process;
end behavioral;


--------------------3 to 1 mux-------------------
library IEEE;
use IEEE.std_logic_1164.all;
entity mux3to1 is
        port(s: in std_logic;
                 S0,S1,S2: in std_logic_vector(7 downto 0);
                 output : out std_logic_vector(7 downto 0));
end mux3to1;


architecture behavior of mux3to1 is
begin
        output<=S0 When s='0' else
                         S1 WHEN s='1' else
                         S2;
```

```
end behavior;



----------------Accumulater8bit-----------------
library IEEE;
use IEEE.std_logic_1164.all;
entity Accumulater8bit is
		port( data_in :in std_logic_vector(7 downto 0);
				ACClk, ACreset,ACenable,ACtoReset: in std_logic;
				data_out: out std_logic_vector(7 downto 0));
end Accumulater8bit;

architecture structural of Accumulater8bit is
component adder8
port (A, B: in std_logic_vector (7 downto 0);
				cin: in std_logic;
				cout: out std_logic;
				sum: out std_logic_vector (7 downto 0));
end component;
component Register8bit
		port( regin :in std_logic_vector(7 downto 0);
				Clk, sync_reset,C,toReset: in std_logic;
				regout: out std_logic_vector(7 downto 0));
end component;
signal temp1,temp2: std_logic_vector (7 downto 0):= (others => '0');
begin
		FA: adder8 port map (A => data_in, B => temp2, cin => '0', sum => temp1);
		REG: Register8bit port map (regin => temp1, Clk => ACClk, sync_reset => ACreset,
C=>ACenable,toReset=>ACtoReset,regout => temp2);
```

```
        data_out <= temp2;
end structural;

----------------- 1BIT comparator entity ---------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY onebitcomparator IS PORT(
a, b, c, d: IN STD_LOGIC;
c_out, d_out,eq: OUT STD_LOGIC);
END onebitcomparator;

ARCHITECTURE struct OF onebitcomparator IS
COMPONENT notgate PORT(
i: IN STD_LOGIC;
o: OUT STD_LOGIC);
END COMPONENT;
COMPONENT nand2gate PORT(
i1, i2: IN STD_LOGIC;
o: OUT STD_LOGIC);
END COMPONENT;
COMPONENT nand4gate PORT(
i1, i2, i3, i4: IN STD_LOGIC;
o: OUT STD_LOGIC);
END COMPONENT;

COMPONENT nor2gate PORT(
i1, i2: IN STD_LOGIC;
o: OUT STD_LOGIC);
END COMPONENT;
```

```
SIGNAL nota, notb, notc, notd, s1, s2: STD_LOGIC;

BEGIN


G1: notgate PORT MAP(a,nota);

G2: notgate PORT MAP(b,notb);

G3: notgate PORT MAP(c,notc);

G4: notgate PORT MAP(d,notd);

G5: nand4gate PORT MAP(a, notb, notc, notd, s1);

G6: nand2gate PORT MAP(s1, notc, c_out);

G7: nand4gate PORT MAP(nota, b, notc, notd, s2);

G8: nand2gate PORT MAP(s2, notd, d_out);


END struct;




----------------- eightbitcomparator ---------------------

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

ENTITY eightbitcomparator IS PORT(

a: IN STD_LOGIC_VECTOR(7 DOWNTO 0);

b: IN STD_LOGIC_VECTOR(7 DOWNTO 0);

aGreater, aSmaller, eq: OUT STD_LOGIC);

END eightbitcomparator;

ARCHITECTURE struct OF eightbitcomparator IS


COMPONENT onebitcomparator PORT(

a, b, c, d: IN STD_LOGIC;

c_out, d_out,eq: OUT STD_LOGIC);
```

```
END COMPONENT;

COMPONENT nor2gate PORT(
i1, i2: IN STD_LOGIC;
o: OUT STD_LOGIC);
END COMPONENT;

SIGNAL ci, di: STD_LOGIC_VECTOR(7 DOWNTO 0);

BEGIN

G0: onebitcomparator PORT MAP(a(7), b(7), '0', '0', ci(7), di(7));
G1: onebitcomparator PORT MAP(a(6), b(6), ci(7), di(7), ci(6), di(6));
G2: onebitcomparator PORT MAP(a(5), b(5), ci(6), di(6), ci(5), di(5));
G3: onebitcomparator PORT MAP(a(4), b(4), ci(5), di(5), ci(4), di(4));
G4: onebitcomparator PORT MAP(a(3), b(3), ci(4), di(4), ci(3), di(3));
G5: onebitcomparator PORT MAP(a(2), b(2), ci(3), di(3), ci(2), di(2));
G6: onebitcomparator PORT MAP(a(1), b(1), ci(2), di(2), ci(1), di(1));
G7: onebitcomparator PORT MAP(a(0), b(0), ci(1), di(1), ci(0), di(0));
G8: onebitcomparator PORT MAP(a(0), b(0), ci(1), di(1), aGreater, aSmaller);
G9: nor2gate PORT MAP(ci(0), di(0), eq);

END struct;

--------------------------logicDetector----------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY logicDetector IS PORT(
	Greater,Smaller,equal,Clk : in std_logic;
```

```
        S1,S0, SodaType: in  std_logic_vector(7 downto 0);
        D : out std_logic_vector(1 downto 0);
        enable,done: out std_logic);
END logicDetector;

architecture behavioral of logicDetector is
begin
        process(Clk)
        begin
        if rising_edge(Clk) then
                if (Greater='1' and SodaType=S1) then
                        enable <= '1'; --return extra money
                        D <= "10"; --return soda1
                        done<='1';
                elsif (Greater='1' and SodaType=S0) then
                        enable <= '1'; --return extra money
                        D <= "01"; --return soda1
                        done<='1';
                elsif (equal='1' and SodaType=S1) then
                        enable <= '0'; --no return extra money
                        D <= "10"; --return soda1
                        done<='1';
                elsif (equal='1' and SodaType=S0) then
                        enable <= '0'; --no return extra money
                        D <= "01"; --return soda1
                        done<='1';
                else
                        enable <= '0'; --no return extra money
                        D <= "00"; --no action
```

```
                                        done<='0';
                                end if;
                        end if;
                        end process;
end behavioral;
-----------------------twoscompliment------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity twoscompliment is
 port (  --Inputs
          A : in std_logic_vector (7 downto 0);
          --Outputs
          Y : out std_logic_vector (7 downto 0));
end twoscompliment;

architecture twoscompliment_v1 of twoscompliment is
   constant ONE:   UNSIGNED(Y'RANGE) := (0 => '1', others => '0');
begin
    Y <= std_logic_vector(unsigned (not A) + ONE);
end twoscompliment_v1;

-----------------------------subtractor----------------------------
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all;
ENTITY subtractor IS
generic(nbit: natural :=8);
PORT(
            enabled,reset: in std_logic;

            SodaTyped,depositMoney : in  std_logic_vector(7 downto 0);

            E: out std_logic_vector(nbit-1 downto 0));

END subtractor;


architecture behav of subtractor is


            signal complimentResult: std_logic_vector(7 downto 0);

            constant ONE:   UNSIGNED(complimentResult'RANGE) := (0 => '1', others => '0');

begin

            process(enabled)

            begin

            if enabled='1' then

                        complimentResult <=std_logic_vector(unsigned(depositMoney(7 downto 0)) +
unsigned(SodaTyped(7 downto 0)));

                        E <= std_logic_vector(unsigned (not complimentResult) + ONE);

            elsif reset='1' then

                        E <=depositMoney;

            else

                        E<=(others=>'0');

            end if;

            end process;

end behav;


--------------------------vendingMachine------------------------

LIBRARY IEEE;

USE IEEE.STD_LOGIC_1164.ALL;

ENTITY vendingMachine IS PORT(

            C,Choice,Reset,clk : in std_logic;
            S1,S0,V : in  std_logic_vector(7 downto 0);
            D : out std_logic_vector(1 downto 0);
            P,E: out std_logic_vector(7 downto 0));
END vendingMachine;

architecture behav of vendingMachine is
            component Accumulater8bit is
            port( data_in : in std_logic_vector(7 downto 0);
                                    ACClk,ACreset,ACenable,ACtoReset : in std_logic;
                                    data_out : out std_logic_vector(7 downto 0));
            end component;
            component eightbitcomparator is
                        port (a,b : in STD_LOGIC_VECTOR(7 downto 0);
                                    aGreater, aSmaller, eq: out STD_LOGIC);
            end component;
            component mux3to1 is
            port(s: in std_logic;
                         S0,S1,S2: in std_logic_vector(7 downto 0);
                         output : out std_logic_vector(7 downto 0));
            end component;
            component logicDetector IS PORT(
                        Greater,Smaller,equal,Clk : in std_logic;
                        S1,S0, SodaType: in  std_logic_vector(7 downto 0);
                        D : out std_logic_vector(1 downto 0);
                        enable,done: out std_logic);
            END component;
            component twoscompliment is
                        port (
          A : in std_logic_vector (7 downto 0);
```

```
        Y : out std_logic_vector (7 downto 0));
                end component;
          component subtractor IS PORT(
                enabled,reset: in std_logic;
                SodaTyped,depositMoney : in  std_logic_vector(7 downto 0);
                E: out std_logic_vector(7 downto 0));
          END component;


          signal currentDeposite,valueOfDrink,compliment,noThings:std_logic_vector(7 downto 0);
          signal g,s,compareE,enable,doneSignal,DReset:std_logic;


begin
          noThings<=(others=>'X');
          G1: Accumulater8bit PORT MAP (V,clk,Reset,C,doneSignal,currentDeposite);
          P <= currentDeposite;
          G2: mux3to1 port map (Choice, S0, S1, noThings, valueOfDrink);
          G3: eightbitcomparator port map (currentDeposite,valueOfDrink,g,s,compareE);
          G4: logicDetector port map (g,s,compareE,clk,S1,S0,valueOfDrink,D,enable,doneSignal);
          G5: twoscompliment port map (valueOfDrink,compliment);
          G6: subtractor port map (enable,Reset,compliment,currentDeposite,E);
end behav;
---------------------------------------------------------------------
--00000000--0--------00001011--11--------00010110--22
--00000001--1--------00001100--12--------00010111--23
--00000010--2--------00001101--13--------00011000--24
--00000011--3--------00001110--14--------00011001--25
--00000100--4--------00001111--15--------
--00000101--5--------00010000--16--------
--00000110--6--------00010001--17--------
```

--00000111--7--------00010010--18--------

--00001000--8--------00010011--19--------

--00001001--9--------00010100--20--------

--00001010--10-------00010101--21--------

Simulation Results

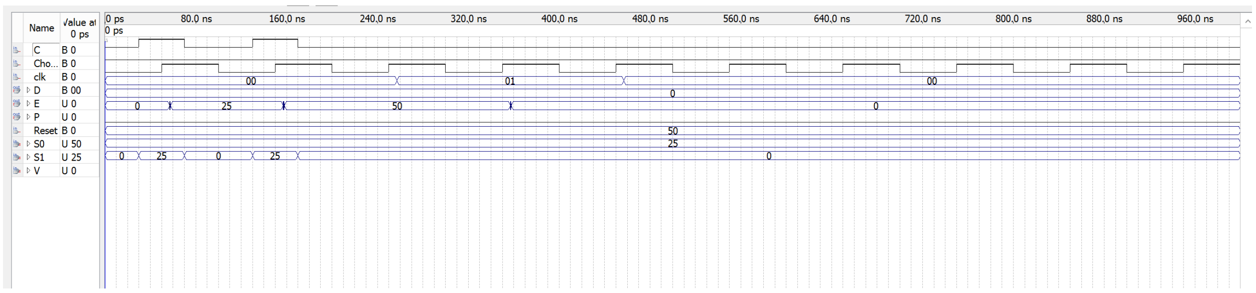

Figure 3. Result 1

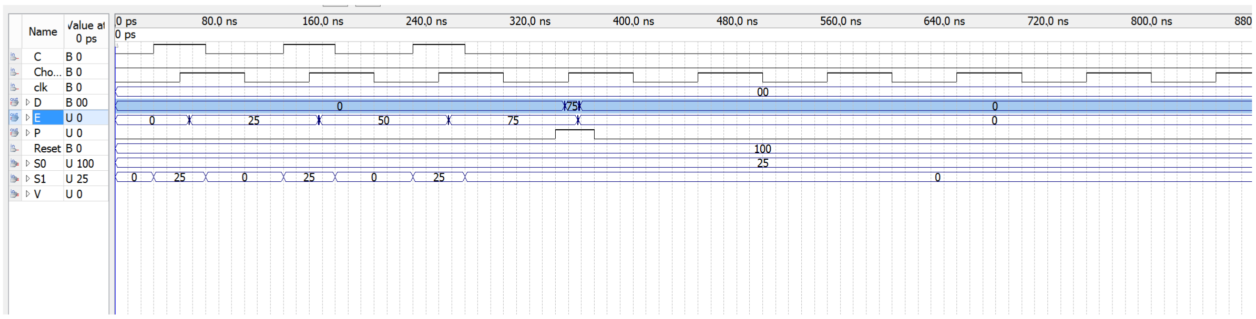

Figure 4. Result 2

Conclusion

The logic of the design includes three scenarios. The first scenario is that the user gives the exact same amount of money for the specific soda. Then the machine would give the soda without any change of money. The second scenario is that the user gives money which exceed the price of the specific soda. Then the machine would give the soda and the change. The last scenario is that the user doesn't give enough money to the machine, and there is no soda coming out form the machine. After many testing and debugging. These three scenarios are pointing to the three parts of the codes, and each of them are working. This means we have designed this soda vending machine correctly, and we have finished this project successfully.