**TITLE**: Design and Simulation of a Coffee Vending Machine in VHDL

**OBJECTIVE**: To design and simulate working of a coffee vending machine using Xilinx ISE Simulator

**THEORY**

A vending machine is an automated machine that dispenses items like snacks, drinks, and other goods when you insert money or use a credit card. They're often found in places like offices, schools, hospitals, and airports. They are a convenient way to buy a quick snack or drink.

Here's how they work:

1. Choose an item: Look at the pictures or buttons on the machine to find what you want.

2. Pay: Insert cash, a credit card, or use a mobile payment app.

3. Collect your item: The machine will dispense your item into a tray or slot.

**System Design**

As discussed above, vending machine has three core states choose an item, pay money and collect item. Utilizing these, we develop three states our coffee vending machine can have, namely, idle, put\_money and coffee\_out. Also, the cash that is inserted may be of different denominations. To handle this, we assign different states according to the cash input.

Suppose that the coffee vending machine dispenses coffee at Rs. 20. Then for this, several combinations may occur. Some of the possible occurrences are:

1. entering no money
2. putting 10 followed by another 10 -> getting the coffee
3. putting 10 followed by 20 -> receiving change -> getting a coffee
4. putting 10 followed by 50 -> receiving change -> getting a coffee
5. putting 20 and getting the coffee
6. putting 50 => receiving change => getting a coffee.

Thus, for each cash denomination, we define additional states in\_1, in\_2, in\_3, in\_5, in\_6 and for change of Rs. 10, state change\_1.

**States**

1. put\_money: this is the first state after ‘idle’, here the customer puts in the cash.
2. in\_5: If the customer put 50, the machine moves to this state. Pay attention that the machine gives back a 20 change without dispensing the coffee yet.
3. change\_1: after giving a 20 change for the customer who paid 50,we need to give back another 10 change. Here, in this state we do exactly that. again, note that the coffee still has not been dispensed yet.
4. in\_1, in\_3, in\_6: we get to those states depending on the sum of money inserted by the customer.
5. coffee\_out: dispensing out the coffee.

**Input**

cashIn: the cash provided to vending machine

clk: clock that drives fsm of vending machine

rstn: a signal that resets the fsm to it’s initial state

**Output**

coffee\_out: coffee is dispensed

cashOut: change money is provided to the user

**Assumptions**

1. In this design we will ignore the capacity of the stock, which means, we’ll assume that there will always be can in the vending machine.
2. Also, we can assume that only one action could be made in every “clock cycle” or state.
3. first give back the change then dispense coffee.

**FSM**

The Finite State Machine designed on the basis of above given states, inputs and outputs as well as assumptions is shown below:

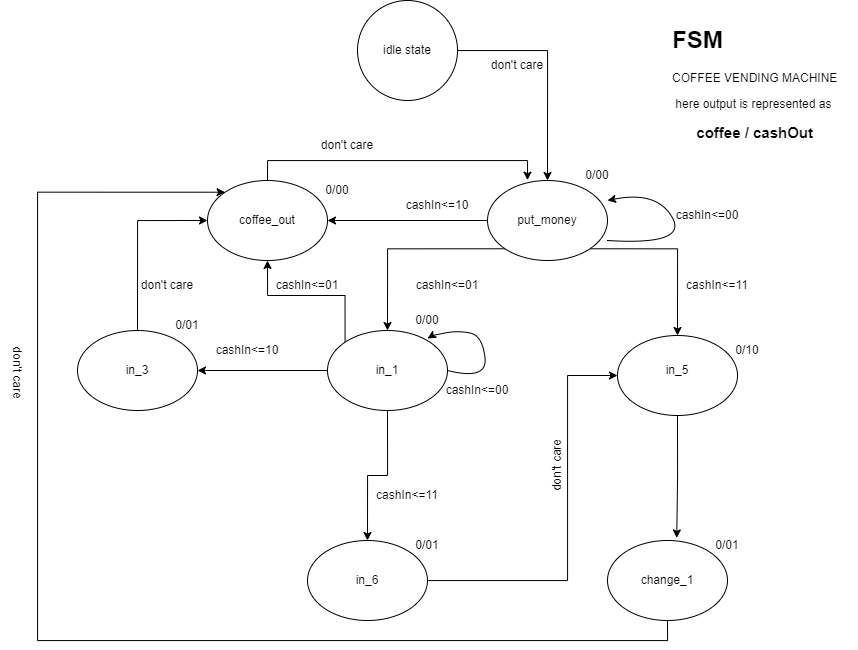


Figure 1: Moore Machine for Coffee Vending Machine

**CODE**

**VHDL**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity vendingmachine is

Port ( clk : in STD\_LOGIC;

rstn : in STD\_LOGIC;

cashIn : in STD\_LOGIC\_VECTOR (1 downto 0);

coffee : out STD\_LOGIC;

cashOut : out STD\_LOGIC\_VECTOR (1 downto 0));

end vendingmachine;

architecture Behavioral of vendingmachine is

type state\_type is ( idle,--start state/reset

put\_money, --waiting to enter money

in\_1,in\_3,in\_6,in\_5, --represent the current sum of money after returning change

change\_1, --should return change of Rs 10

coffee\_out --dispence coffee can.

); --type of state machine.

signal current\_s,next\_s: state\_type; --current and next state declaration.

begin

--comb

process(CLK,rstn)

begin

if(rstn = '0') then

current\_s <= idle; --defualt state is on RESET

elsif(clk'event and clk = '1') then

current\_s <= next\_s;

end if;

end process;

process(current\_s,cashIn)

begin

case current\_s is

when idle =>

coffee <= '0';

cashOut <= "00";

next\_s <= put\_money;

when put\_money => --wait for money to be entered

if(cashIn = "00")then

coffee <= '0';

cashOut <= "00";

next\_s <= put\_money;

elsif(cashIn = "01")then --insert Rs 10

coffee <= '0';

cashOut <= "00";

next\_s <= in\_1;

elsif(cashIn = "10")then --insert Rs 20

coffee <= '0';

cashOut <= "00";

next\_s <= coffee\_out;

elsif(cashIn = "11")then --insert Rs 30

coffee <= '0';

cashOut <= "00";

next\_s <= in\_5;

end if;

------------------------------------------------------

when in\_1 =>

if(cashIn = "00") then--stay on the same state

coffee <= '0';

cashOut <= "00";

next\_s <= in\_1;

elsif(cashIn = "01") then--inserted another 10

coffee <= '0';

cashOut <= "00";

next\_s <= coffee\_out;

elsif(cashIn = "10") then--inserted another 20

coffee <= '0';

cashOut <= "00";

next\_s <= in\_3;

elsif(cashIn = "11") then

coffee <= '0';

cashOut <= "10";

next\_s <= in\_6;

end if;

------------------------------------------------------

when in\_3 =>

coffee <= '0';

cashOut <= "01";

next\_s <= coffee\_out;

------------------------------------------------------

when in\_6 =>

coffee <= '0';

cashOut <= "01";

next\_s <= in\_5;

------------------------------------------------------

when in\_5 => -- input = Rs 50

coffee <= '0';

cashOut <= "10";

next\_s <= change\_1;

------------------------------------------------------

when change\_1 => -- input = Rs 50

coffee <= '0';

cashOut <= "01";

next\_s <= coffee\_out;

------------------------------------------------------

when coffee\_out =>

coffee <= '1';

cashOut <= "00";

next\_s <= put\_money;

end case;

end process;

end Behavioral;

**Testbench**

LIBRARY ieee;

USE ieee.std\_logic\_1164.ALL;

USE ieee.std\_logic\_unsigned.all;

USE ieee.numeric\_std.ALL;

ENTITY vmtb\_vhd IS

END vmtb\_vhd;

ARCHITECTURE behavior OF vmtb\_vhd IS

constant CLK\_PERIOD: time := 10 ns;

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT vendingmachine

PORT(

clk : IN std\_logic;

rstn : IN std\_logic;

cashIn : IN std\_logic\_vector(1 downto 0);

coffee : OUT std\_logic;

cashOut : OUT std\_logic\_vector(1 downto 0)

);

END COMPONENT;

--Inputs

SIGNAL clk : std\_logic := '0';

SIGNAL rstn : std\_logic := '0';

SIGNAL cashIn : std\_logic\_vector(1 downto 0) := (others=>'0');

--Outputs

SIGNAL coffee : std\_logic;

SIGNAL cashOut : std\_logic\_vector(1 downto 0);

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: vendingmachine PORT MAP(

clk => clk,

rstn => rstn,

cashIn => cashIn,

coffee => coffee,

cashOut => cashOut

);

CLK\_process: process

begin

while now < 200 ns loop

CLK <= '0';

wait for CLK\_PERIOD/2;

CLK <= '1';

wait for CLK\_PERIOD/2;

end loop;

wait;

end process CLK\_process;

tb : PROCESS

BEGIN

-- Reset

rstn <= '0';

wait for CLK\_PERIOD;

rstn <= '1';

wait for CLK\_PERIOD;

-- Scenario 1: No money

cashIn <= "00";

wait for CLK\_PERIOD \* 2;

-- Scenario 2: Insert 10 followed by another 10

cashIn <= "01";

wait for CLK\_PERIOD;

cashIn <= "01";

wait for CLK\_PERIOD \* 2;

-- Scenario 3: Insert 10 followed by 20

cashIn <= "01";

wait for CLK\_PERIOD;

cashIn <= "10";

wait for CLK\_PERIOD \* 2;

-- Scenario 4: Insert 10 followed by 50

cashIn <= "01";

wait for CLK\_PERIOD;

cashIn <= "11";

wait for CLK\_PERIOD \* 2;

-- Scenario 5: Insert 20

cashIn <= "10";

wait for CLK\_PERIOD \* 2;

-- End simulation

wait;

END PROCESS;

END;

**BLOCK DIAGRAM**

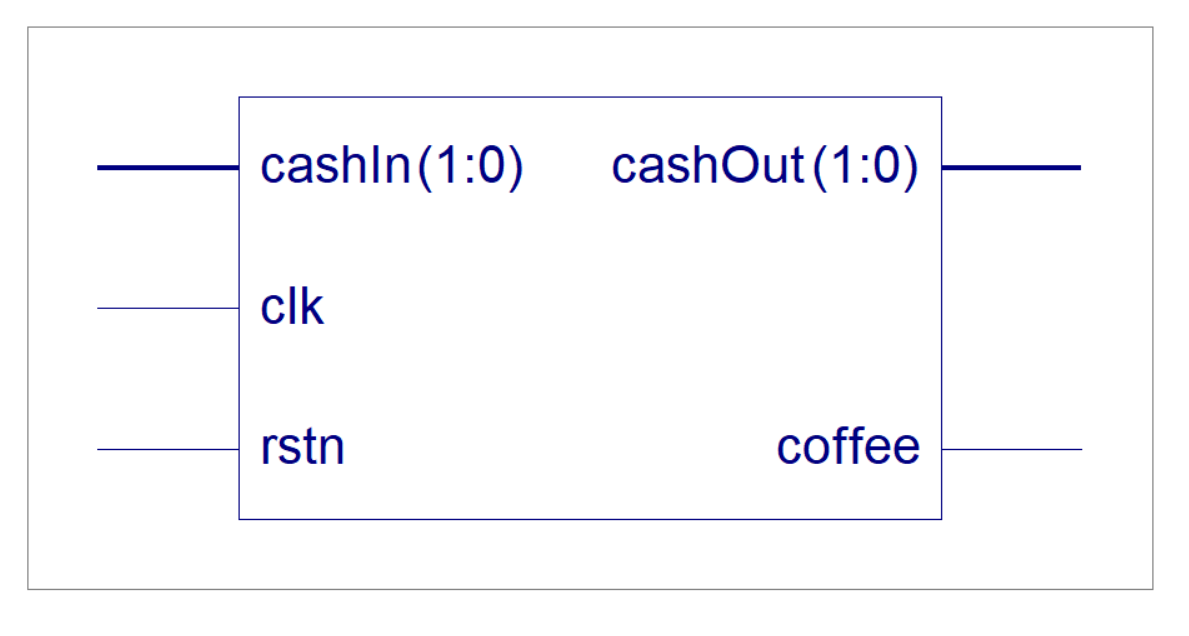
****

Figure 2: Block Diagram of Coffee Vending Machine

**TIMING DIAGRAMS**

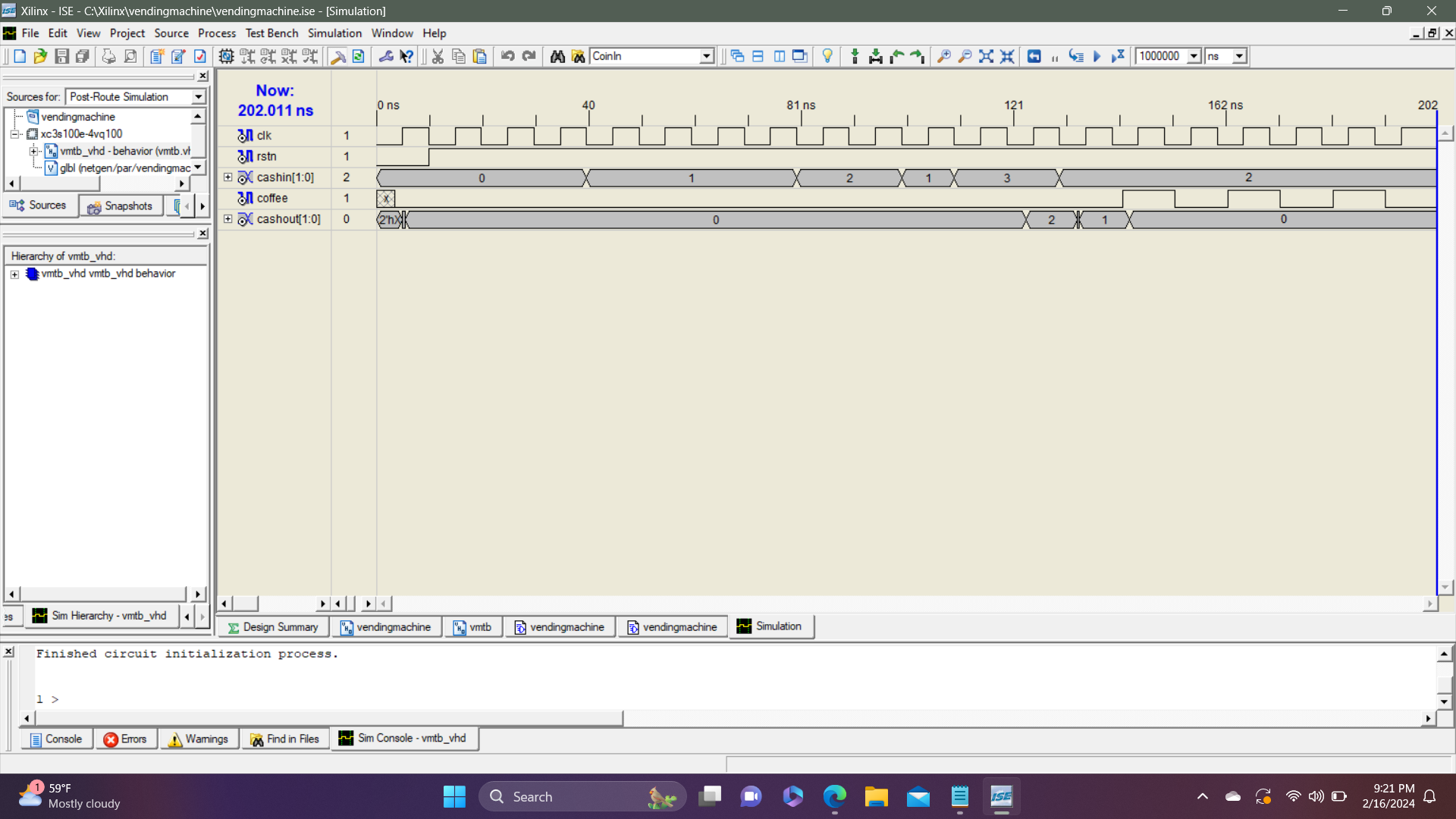


Figure 3:Timing diagram for above testbench

****

Figure 4: Timing Diagram for input 00

**CONCLUSION AND DISCUSSION**

Thus, the coffee vending machine was designed, implemented and simulated in Xilinx. The simulation and coding were done on earlier version of Xilinx 8.1 due to compatibility issues with the system. The simulation shows the dispensing of coffee when the cash in amount is exactly 20.

Although not every cash input scenario was tested, the test combinations were chosen carefully to make sure that the main functions of the machine were tested thoroughly. The test scenarios included exact change, cash with less or more change, and multiple item purchase scenarios. While these test scenarios provided a good basis for verifying the behavior of the machine, it was recognized that more test cases could improve the machine’s reliability and robustness.

Hence, design and simulation of working of a vending machine using Xilinx ISE Simulator was done.