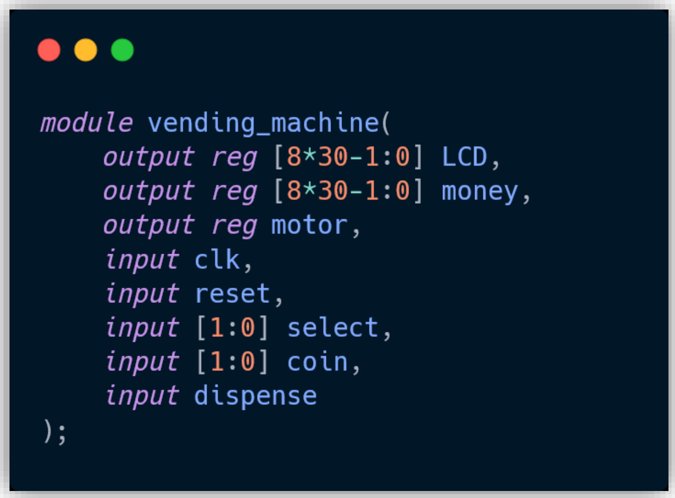
"In the name of God"

Design and implementation of a digital vending machine controller using Verilog

By Zahra Azizi

This project aims to design a digital vending machine with 4 different products at different prices.



```
module vending_machine(
    output reg [8*30-1:0] LCD,
    output reg [8*30-1:0] money,
    output reg motor,
    input clk,
    input reset,
    input [1:0] select,
    input [1:0] coin,
    input dispense
);
```

Fig. 1 - module

- Module definition

Outputs

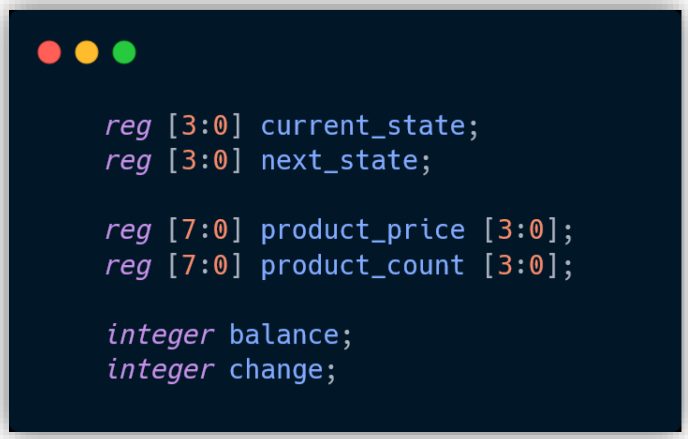
LCD = Displaying current states, an array of 30 * 8 bits.
money = The amount to be returned or the amount that the user has entered.
motor = When the product is being dispensed, will become 1.

Inputs

clk = Clock signal.
reset = When it becomes 1, the machine goes to idle state.

select = For choosing the product by the user.
coin = The amount of money that the user has entered.
dispense = When the product is being dispensed, the machine will go to the next state only if this becomes 1.

- Variables
  1) current_state
  2) next_state :This is defined based on the current state
  3) product_price: An array of 4 * 8 bits.
  4) product_count
  5) balance
  6) change



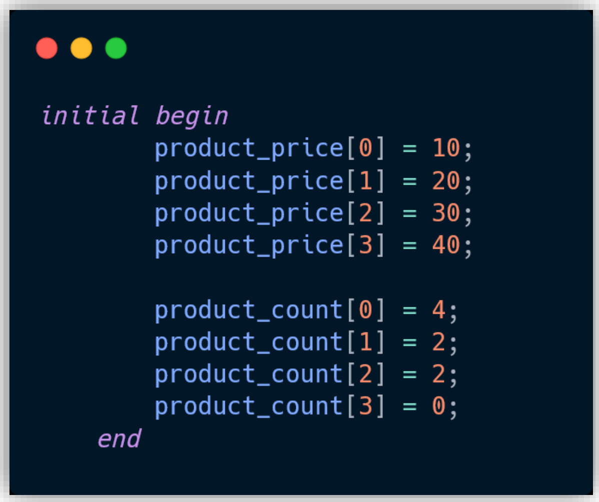
```
reg [3:0] current_state;
reg [3:0] next_state;

reg [7:0] product_price [3:0];
reg [7:0] product_count [3:0];

integer balance;
integer change;
```

Fig. 2 - Variables

- Assumptions

Assumptions for the products are as follows:

Name	Initial count	Price
Product 1	4	10c
Product 2	2	20c
Product 3	2	30c
Product 4	0	40c



```
initial begin
        product_price[0] = 10;
        product_price[1] = 20;
        product_price[2] = 30;
        product_price[3] = 40;

        product_count[0] = 4;
        product_count[1] = 2;
        product_count[2] = 2;
        product_count[3] = 0;
    end
```

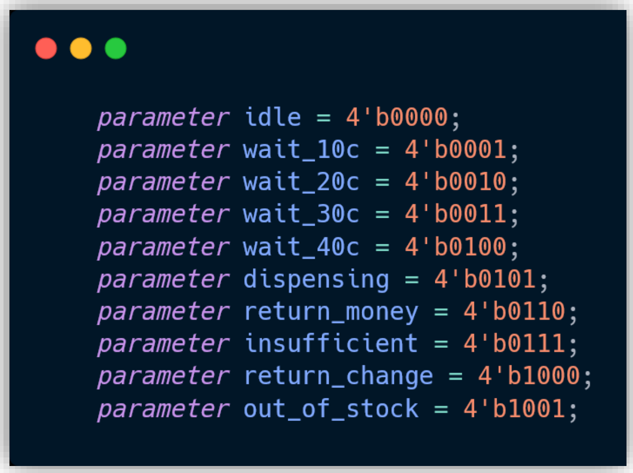
Fig. 3 – Assumptions

- States

This machine includes 10 states:

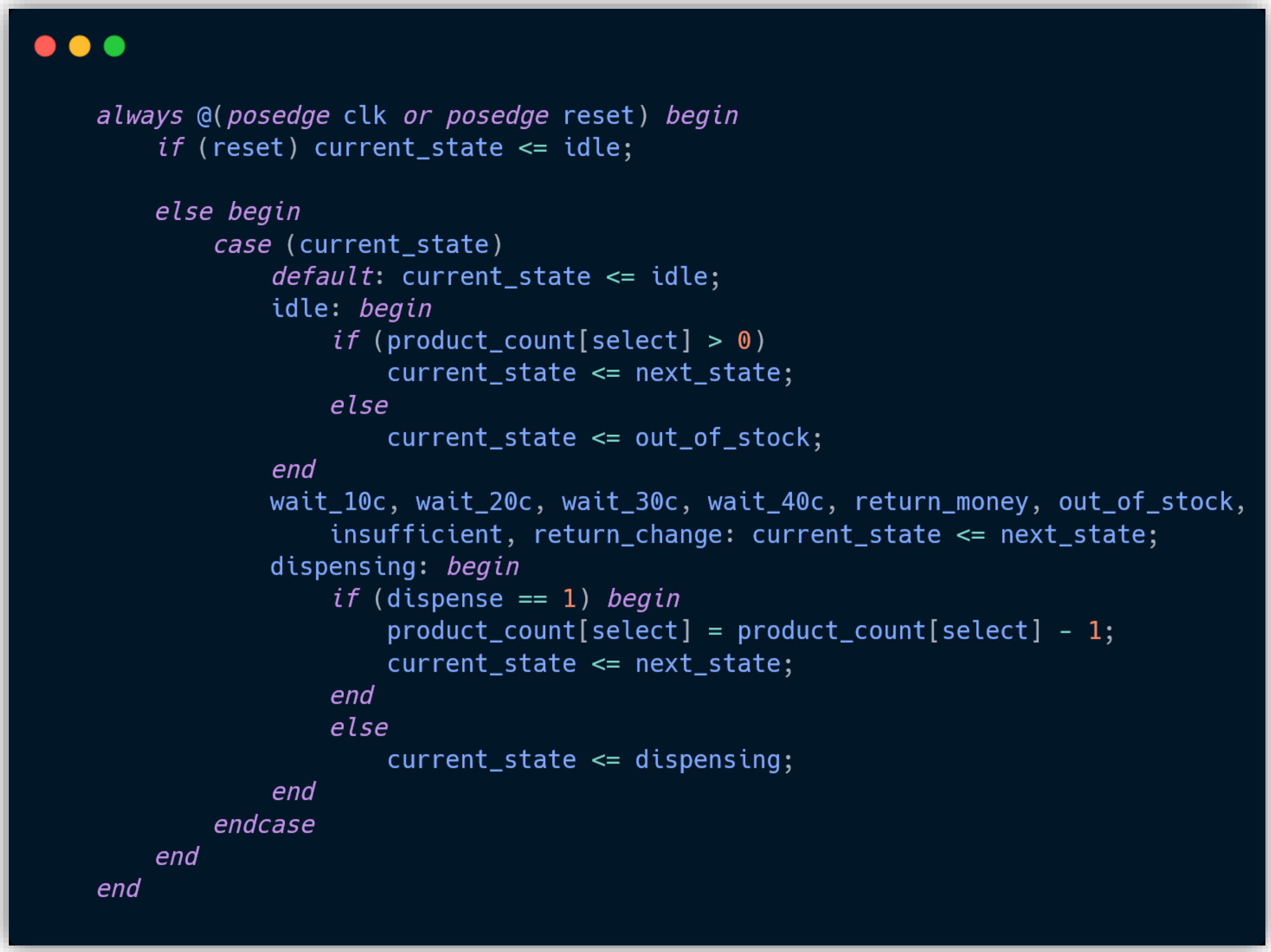
1. **idle:** The machine doesn't do anything and is waiting for the user to choose a product. If the user chooses an out-of-stock product, it will go to the **out_of_stock** state.

2. **wait_10c:** The user has picked the first product and the machine is waiting for coins to be inserted.
3. **wait_20c:** The user has picked the second product and the machine is waiting for coins to be inserted.
4. **wait_30c:** The user has picked the third product and the machine is waiting for coins to be inserted.
5. **wait_40c:** The user has picked the fourth product and the machine is waiting for coins to be inserted.
6. **dispensing:** The machine is dispensing the product. This state will end when dispense becomes 1. After this state, the number of the selected product is decreased by one.
7. **return_money:** If there isn’t enough money, the whole money will be returned.
8. **insufficient:** It means the money is not enough and it will go to the **return_money** state.
9. **return_change**
10. **out_of_stock**



```
    parameter idle = 4'b0000;
    parameter wait_10c = 4'b0001;
    parameter wait_20c = 4'b0010;
    parameter wait_30c = 4'b0011;
    parameter wait_40c = 4'b0100;
    parameter dispensing = 4'b0101;
    parameter return_money = 4'b0110;
    parameter insufficient = 4'b0111;
    parameter return_change = 4'b1000;
    parameter out_of_stock = 4'b1001;
```

Fig. 4 - states



```
always @(posedge clk or posedge reset) begin
    if (reset) current_state <= idle;

    else begin
        case (current_state)
            default: current_state <= idle;
            idle: begin
                if (product_count[select] > 0)
                    current_state <= next_state;
                else
                    current_state <= out_of_stock;
            end
            wait_10c, wait_20c, wait_30c, wait_40c, return_money, out_of_stock,
                insufficient, return_change: current_state <= next_state;
            dispensing: begin
                if (dispense == 1) begin
                    product_count[select] = product_count[select] - 1;
                    current_state <= next_state;
                end
                else
                    current_state <= dispensing;
            end
        endcase
    end
end
```

Fig. 5 – general state transition

- State transitions

State transitions are shown in the below diagram:

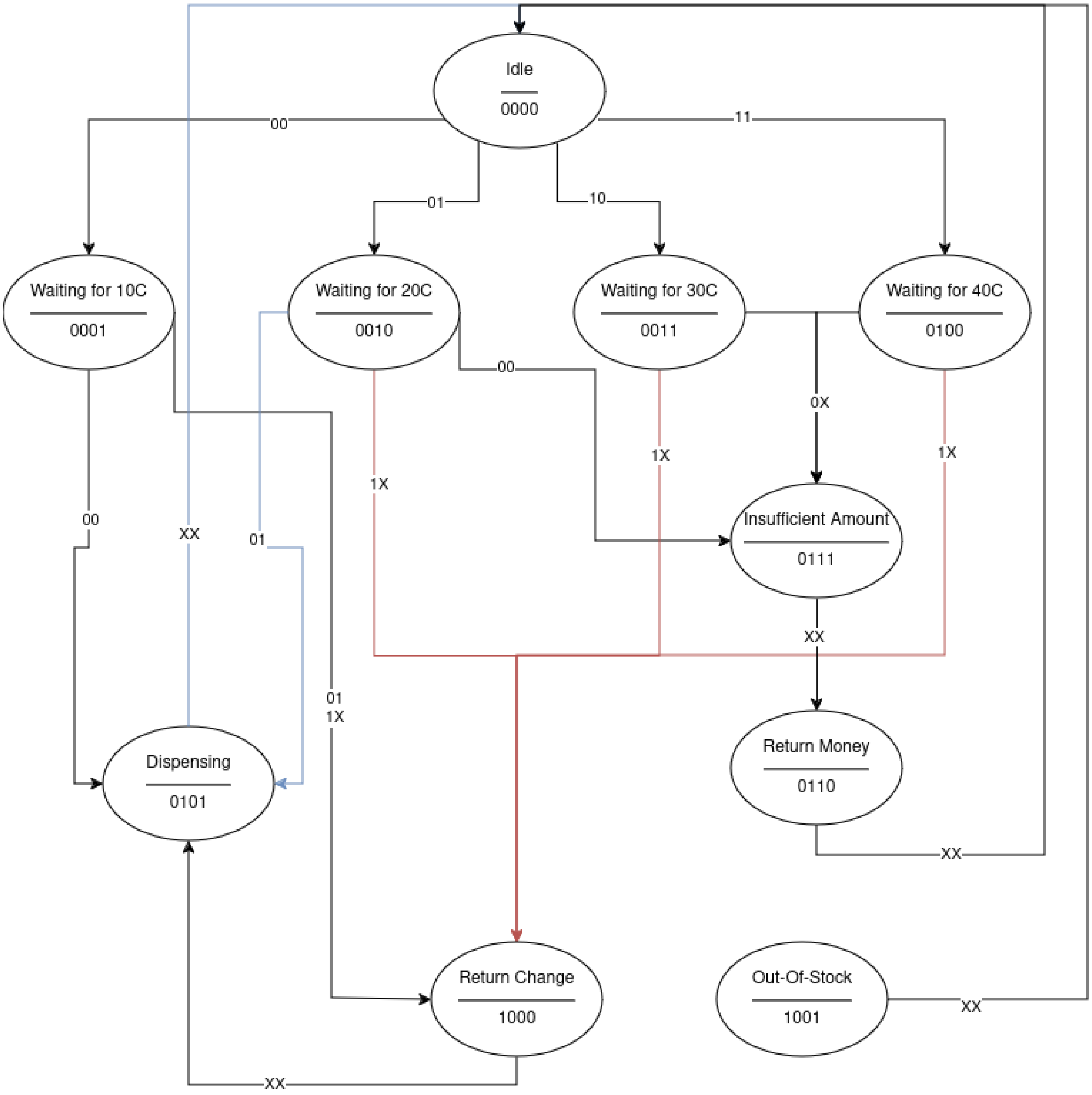
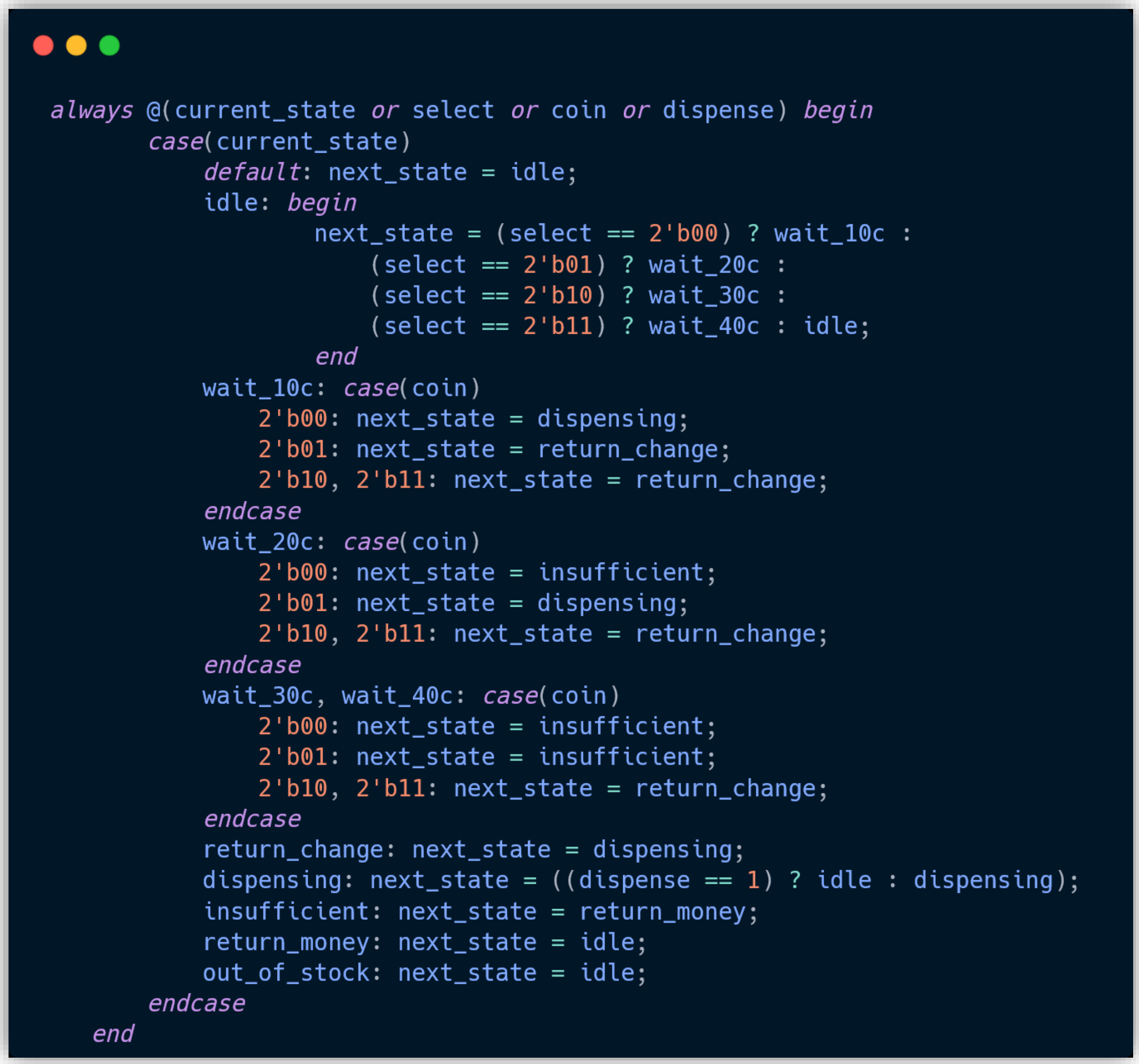


Fig. 6 –State diagram



```
always @(current_state or select or coin or dispense) begin
       case(current_state)
           default: next_state = idle;
           idle: begin
                   next_state = (select == 2'b00) ? wait_10c :
                       (select == 2'b01) ? wait_20c :
                       (select == 2'b10) ? wait_30c :
                       (select == 2'b11) ? wait_40c : idle;
                   end
           wait_10c: case(coin)
               2'b00: next_state = dispensing;
               2'b01: next_state = return_change;
               2'b10, 2'b11: next_state = return_change;
           endcase
           wait_20c: case(coin)
               2'b00: next_state = insufficient;
               2'b01: next_state = dispensing;
               2'b10, 2'b11: next_state = return_change;
           endcase
           wait_30c, wait_40c: case(coin)
               2'b00: next_state = insufficient;
               2'b01: next_state = insufficient;
               2'b10, 2'b11: next_state = return_change;
           endcase
           return_change: next_state = dispensing;
           dispensing: next_state = ((dispense == 1) ? idle : dispensing);
           insufficient: next_state = return_money;
           return_money: next_state = idle;
           out_of_stock: next_state = idle;
       endcase
   end
```

Fig. 7 – state transitions

The machine starts at an idle state and according to the user's choice, it will go to one of the wait_xxc states. For instance, if the user chooses 01, it will go to wait_20c. If there wasn't any product left, it would go to an out-of-stock state without waiting for money, then it would go to an idle state.

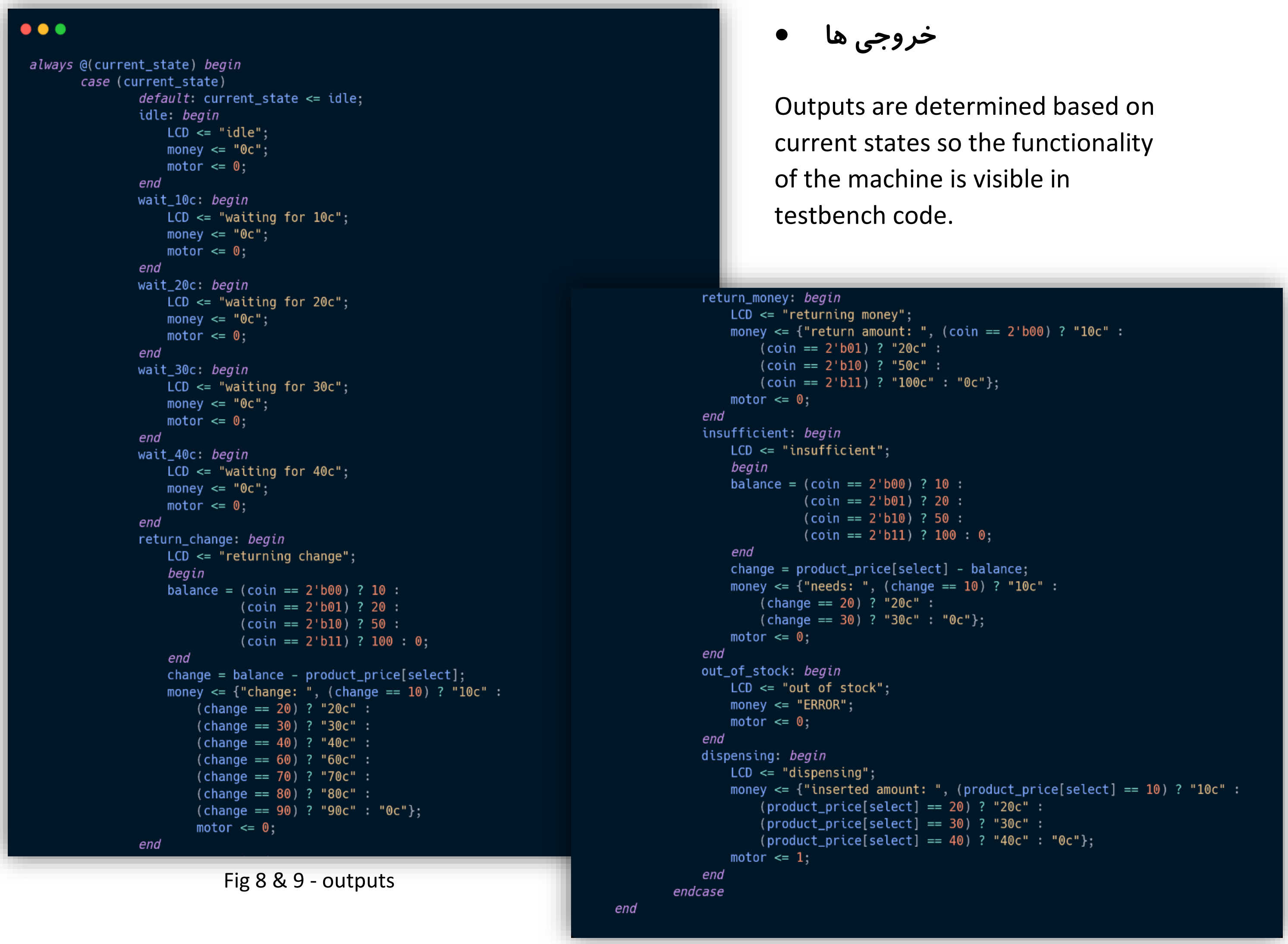
After entering wait_xxc, according to the money inserted, the next state is defined:

- If the money is sufficient, the next state is dispensing.
- if not, the next state is insufficient then return_money.

- If the amount that the user has inserted is more than the price, the machine will return the change, and then it will dispense the product.

- خروجی ها

Outputs are determined based on current states so the functionality of the machine is visible in testbench code.



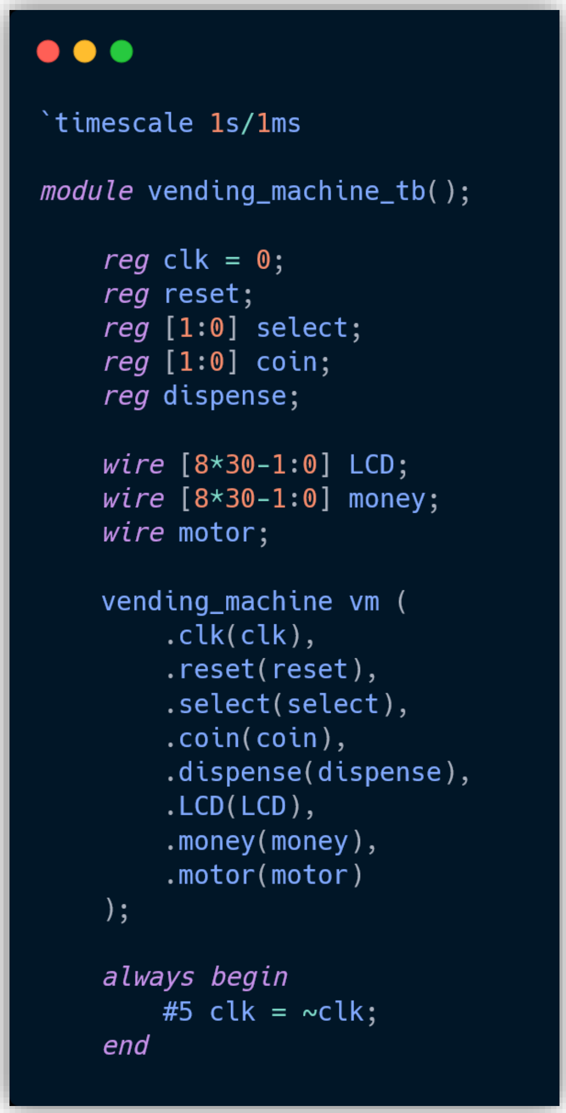
```
always @(current_state) begin
        case (current_state)
                default: current_state <= idle;
                idle: begin
                    LCD <= "idle";
                    money <= "0c";
                    motor <= 0;
                end
                wait_10c: begin
                    LCD <= "waiting for 10c";
                    money <= "0c";
                    motor <= 0;
                end
                wait_20c: begin
                    LCD <= "waiting for 20c";
                    money <= "0c";
                    motor <= 0;
                end
                wait_30c: begin
                    LCD <= "waiting for 30c";
                    money <= "0c";
                    motor <= 0;
                end
                wait_40c: begin
                    LCD <= "waiting for 40c";
                    money <= "0c";
                    motor <= 0;
                end
                return_change: begin
                    LCD <= "returning change";
                    begin
                    balance = (coin == 2'b00) ? 10 :
                              (coin == 2'b01) ? 20 :
                              (coin == 2'b10) ? 50 :
                              (coin == 2'b11) ? 100 : 0;
                    end
                    change = balance - product_price[select];
                    money <= {"change: ", (change == 10) ? "10c" :
                        (change == 20) ? "20c" :
                        (change == 30) ? "30c" :
                        (change == 40) ? "40c" :
                        (change == 60) ? "60c" :
                        (change == 70) ? "70c" :
                        (change == 80) ? "80c" :
                        (change == 90) ? "90c" : "0c"};
                        motor <= 0;
                end
```

```
                return_money: begin
                    LCD <= "returning money";
                    money <= {"return amount: ", (coin == 2'b00) ? "10c" :
                        (coin == 2'b01) ? "20c" :
                        (coin == 2'b10) ? "50c" :
                        (coin == 2'b11) ? "100c" : "0c"};
                    motor <= 0;
                end
                insufficient: begin
                    LCD <= "insufficient";
                    begin
                    balance = (coin == 2'b00) ? 10 :
                              (coin == 2'b01) ? 20 :
                              (coin == 2'b10) ? 50 :
                              (coin == 2'b11) ? 100 : 0;
                    end
                    change = product_price[select] - balance;
                    money <= {"needs: ", (change == 10) ? "10c" :
                        (change == 20) ? "20c" :
                        (change == 30) ? "30c" : "0c"};
                    motor <= 0;
                end
                out_of_stock: begin
                    LCD <= "out of stock";
                    money <= "ERROR";
                    motor <= 0;
                end
                dispensing: begin
                    LCD <= "dispensing";
                    money <= {"inserted amount: ", (product_price[select] == 10) ? "10c" :
                        (product_price[select] == 20) ? "20c" :
                        (product_price[select] == 30) ? "30c" :
                        (product_price[select] == 40) ? "40c" : "0c"};
                    motor <= 1;
                end
            endcase
    end
```

Fig 8 & 9 - outputs

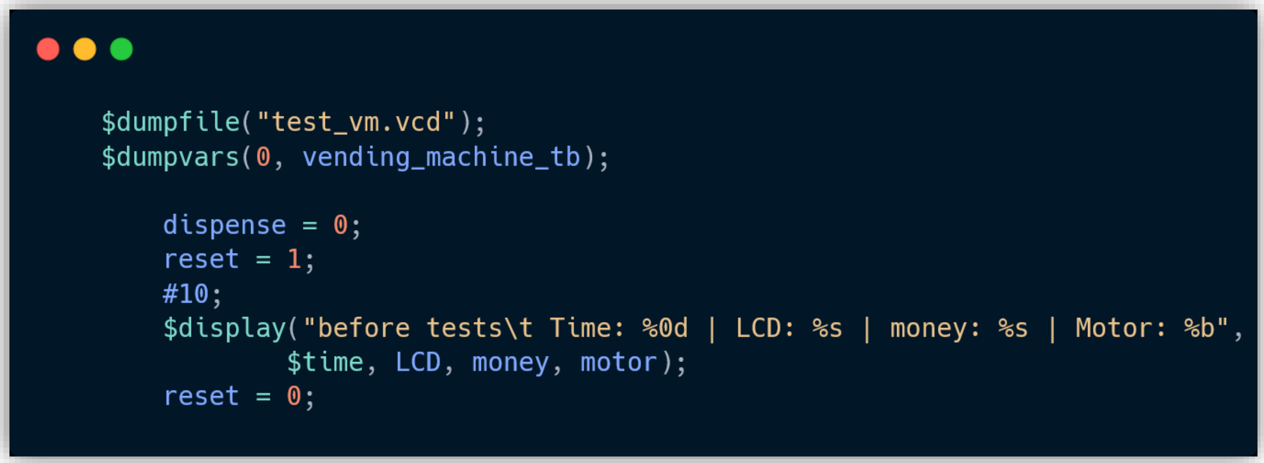
- Testbench

To verify the machine, we need to test it. We will create an instance of it according to the inputs and the outputs. In 5 time units, the clock signal will change. There are 7 test scenarios. After each test, there is a delay of 10 time units so the changes are visible in the terminal output. Before the start of the tests, the machine is reset. Time unit is 1 second and precision is 1 millie second.

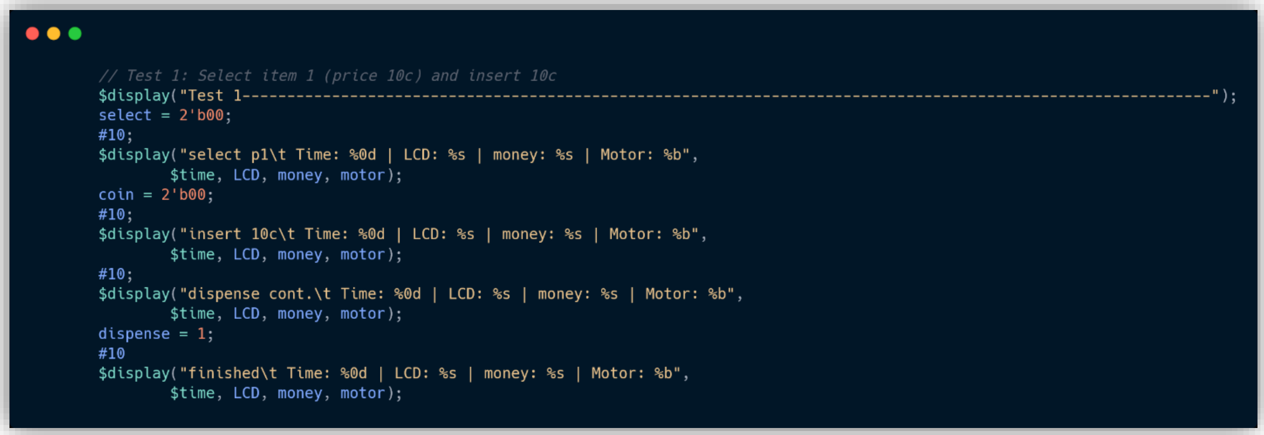


```
`timescale 1s/1ms

module vending_machine_tb();

    reg clk = 0;
    reg reset;
    reg [1:0] select;
    reg [1:0] coin;
    reg dispense;

    wire [8*30-1:0] LCD;
    wire [8*30-1:0] money;
    wire motor;

    vending_machine vm (
        .clk(clk),
        .reset(reset),
        .select(select),
        .coin(coin),
        .dispense(dispense),
        .LCD(LCD),
        .money(money),
        .motor(motor)
    );

    always begin
        #5 clk = ~clk;
    end
```

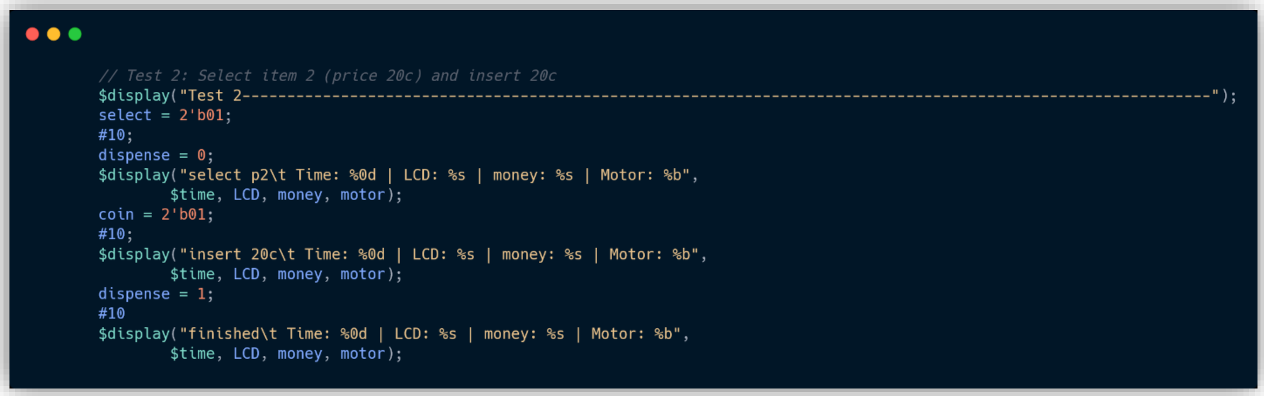
Fig. 10 – instanciation and clock signal



```
    $dumpfile("test_vm.vcd");
    $dumpvars(0, vending_machine_tb);

        dispense = 0;
        reset = 1;
        #10;
        $display("before tests\t Time: %0d | LCD: %s | money: %s | Motor: %b",
                $time, LCD, money, motor);
        reset = 0;
```

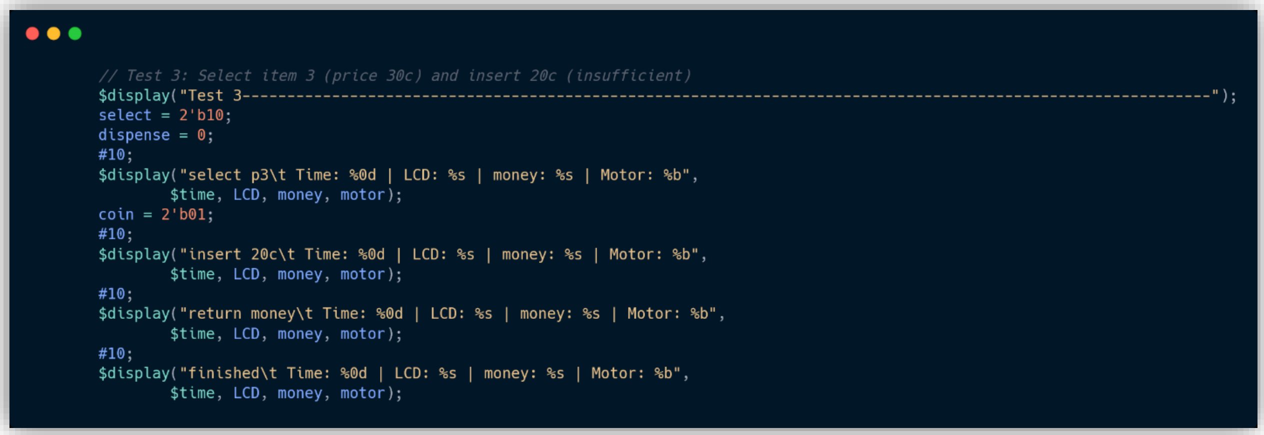


```
    // Test 1: Select item 1 (price 10c) and insert 10c
    $display("Test 1-----------------------------------------------------------------------------------------------------------");
    select = 2'b00;
    #10;
    $display("select p1\t Time: %0d | LCD: %s | money: %s | Motor: %b",
            $time, LCD, money, motor);
    coin = 2'b00;
    #10;
    $display("insert 10c\t Time: %0d | LCD: %s | money: %s | Motor: %b",
            $time, LCD, money, motor);
    #10;
    $display("dispense cont.\t Time: %0d | LCD: %s | money: %s | Motor: %b",
            $time, LCD, money, motor);
    dispense = 1;
    #10
    $display("finished\t Time: %0d | LCD: %s | money: %s | Motor: %b",
            $time, LCD, money, motor);
```

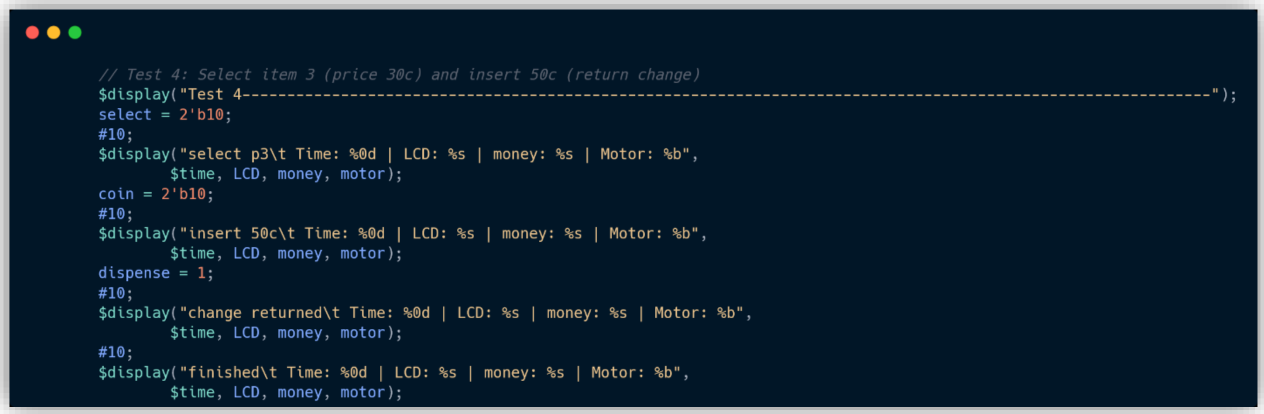


```
    // Test 2: Select item 2 (price 20c) and insert 20c
    $display("Test 2-----------------------------------------------------------------------------------------------------------");
    select = 2'b01;
    #10;
    dispense = 0;
    $display("select p2\t Time: %0d | LCD: %s | money: %s | Motor: %b",
            $time, LCD, money, motor);
    coin = 2'b01;
    #10;
    $display("insert 20c\t Time: %0d | LCD: %s | money: %s | Motor: %b",
            $time, LCD, money, motor);
    dispense = 1;
    #10
    $display("finished\t Time: %0d | LCD: %s | money: %s | Motor: %b",
            $time, LCD, money, motor);
```

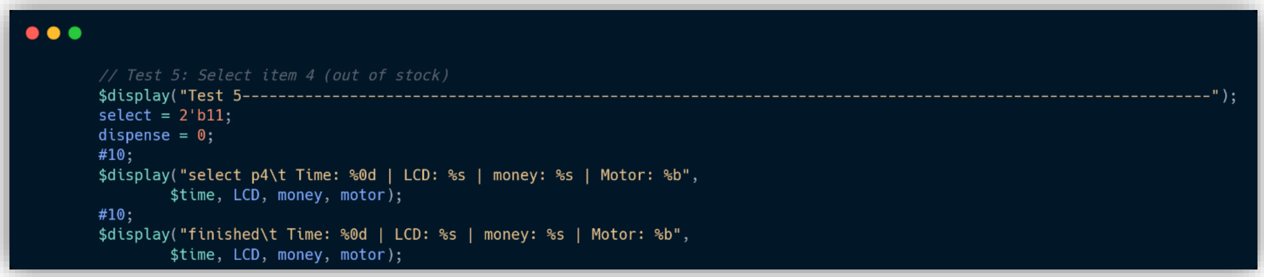
Fig. 11-13 – Initializing and test 1 & 2



```
// Test 3: Select item 3 (price 30c) and insert 20c (insufficient)
$display("Test 3------------------------------------------------------------------------------------------------------------");
select = 2'b10;
dispense = 0;
#10;
$display("select p3\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
coin = 2'b01;
#10;
$display("insert 20c\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
#10;
$display("return money\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
#10;
$display("finished\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
```

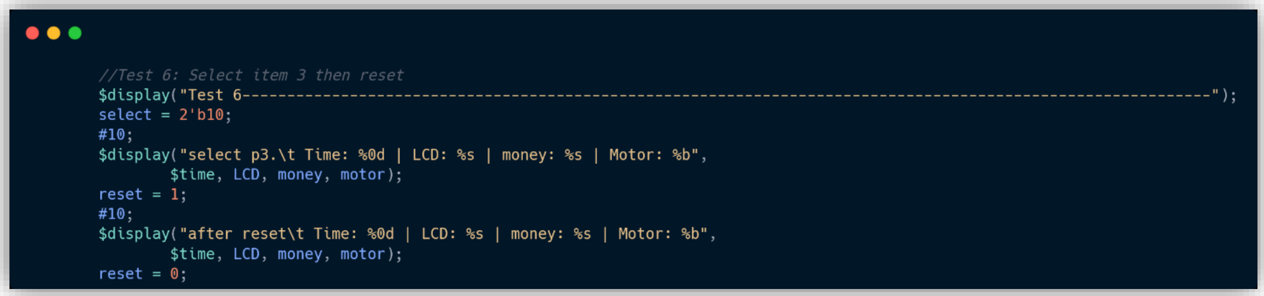


```
// Test 4: Select item 3 (price 30c) and insert 50c (return change)
$display("Test 4------------------------------------------------------------------------------------------------------------");
select = 2'b10;
#10;
$display("select p3\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
coin = 2'b10;
#10;
$display("insert 50c\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
dispense = 1;
#10;
$display("change returned\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
#10;
$display("finished\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
```

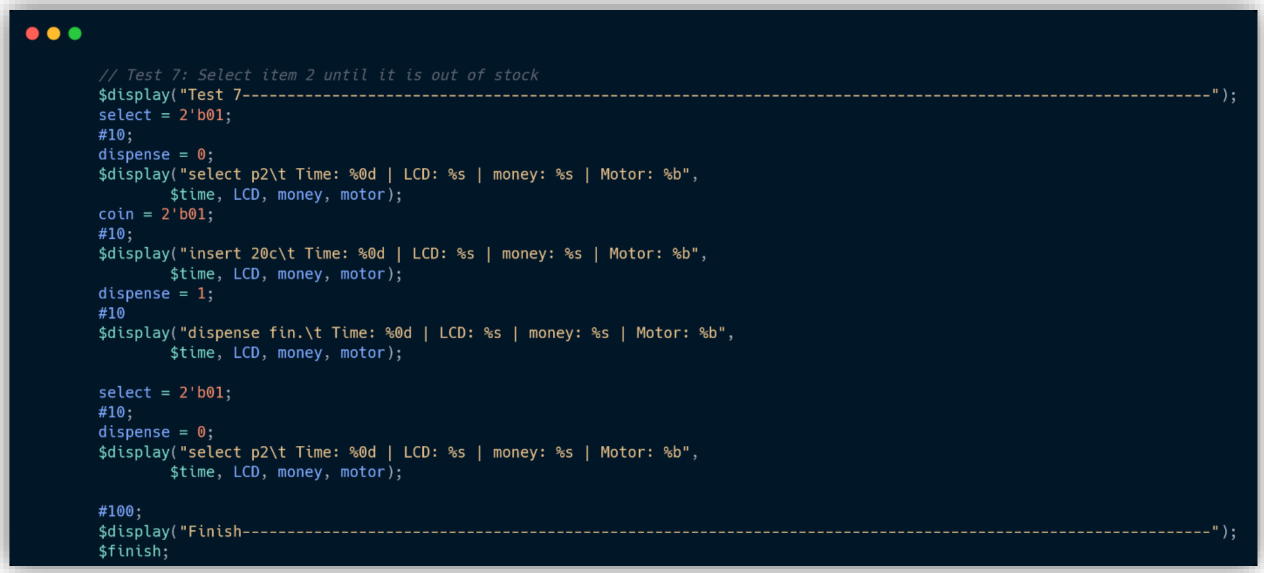


```
// Test 5: Select item 4 (out of stock)
$display("Test 5------------------------------------------------------------------------------------------------------------");
select = 2'b11;
dispense = 0;
#10;
$display("select p4\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
#10;
$display("finished\t Time: %0d | LCD: %s | money: %s | Motor: %b",
        $time, LCD, money, motor);
```

Fig. 14-16 – test 3-5

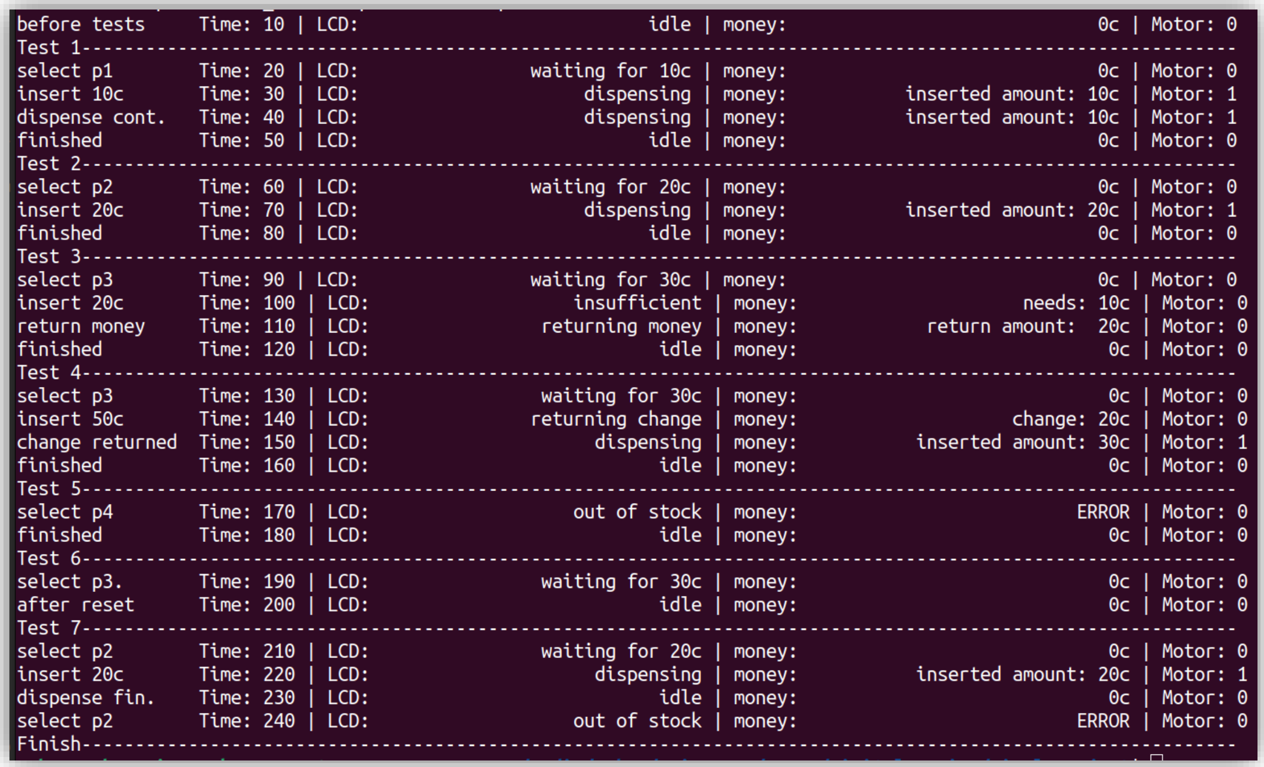


```
        //Test 6: Select item 3 then reset
        $display("Test 6------------------------------------------------------------------------------------------------------------");
        select = 2'b10;
        #10;
        $display("select p3.\t Time: %0d | LCD: %s | money: %s | Motor: %b",
                $time, LCD, money, motor);
        reset = 1;
        #10;
        $display("after reset\t Time: %0d | LCD: %s | money: %s | Motor: %b",
                $time, LCD, money, motor);
        reset = 0;
```



```
        // Test 7: Select item 2 until it is out of stock
        $display("Test 7------------------------------------------------------------------------------------------------------------");
        select = 2'b01;
        #10;
        dispense = 0;
        $display("select p2\t Time: %0d | LCD: %s | money: %s | Motor: %b",
                $time, LCD, money, motor);
        coin = 2'b01;
        #10;
        $display("insert 20c\t Time: %0d | LCD: %s | money: %s | Motor: %b",
                $time, LCD, money, motor);
        dispense = 1;
        #10
        $display("dispense fin.\t Time: %0d | LCD: %s | money: %s | Motor: %b",
                $time, LCD, money, motor);

        select = 2'b01;
        #10;
        dispense = 0;
        $display("select p2\t Time: %0d | LCD: %s | money: %s | Motor: %b",
                $time, LCD, money, motor);

        #100;
        $display("Finish------------------------------------------------------------------------------------------------------------");
        $finish;
```

Fig. 17-18 – test 6 & 7



```
before tests    Time: 10 | LCD:                            idle | money:                             0c | Motor: 0
Test 1------------------------------------------------------------------------------------------------------------
select p1       Time: 20 | LCD:                 waiting for 10c | money:                             0c | Motor: 0
insert 10c      Time: 30 | LCD:                      dispensing | money:           inserted amount: 10c | Motor: 1
dispense cont.  Time: 40 | LCD:                      dispensing | money:           inserted amount: 10c | Motor: 1
finished        Time: 50 | LCD:                            idle | money:                             0c | Motor: 0
Test 2------------------------------------------------------------------------------------------------------------
select p2       Time: 60 | LCD:                 waiting for 20c | money:                             0c | Motor: 0
insert 20c      Time: 70 | LCD:                      dispensing | money:           inserted amount: 20c | Motor: 1
finished        Time: 80 | LCD:                            idle | money:                             0c | Motor: 0
Test 3------------------------------------------------------------------------------------------------------------
select p3       Time: 90 | LCD:                 waiting for 30c | money:                             0c | Motor: 0
insert 20c      Time: 100 | LCD:                    insufficient | money:                     needs: 10c | Motor: 0
return money    Time: 110 | LCD:                 returning money | money:            return amount:  20c | Motor: 0
finished        Time: 120 | LCD:                            idle | money:                             0c | Motor: 0
Test 4------------------------------------------------------------------------------------------------------------
select p3       Time: 130 | LCD:                 waiting for 30c | money:                             0c | Motor: 0
insert 50c      Time: 140 | LCD:                returning change | money:                    change: 20c | Motor: 0
change returned Time: 150 | LCD:                      dispensing | money:           inserted amount: 30c | Motor: 1
finished        Time: 160 | LCD:                            idle | money:                             0c | Motor: 0
Test 5------------------------------------------------------------------------------------------------------------
select p4       Time: 170 | LCD:                    out of stock | money:                          ERROR | Motor: 0
finished        Time: 180 | LCD:                            idle | money:                             0c | Motor: 0
Test 6------------------------------------------------------------------------------------------------------------
select p3.      Time: 190 | LCD:                 waiting for 30c | money:                             0c | Motor: 0
after reset     Time: 200 | LCD:                            idle | money:                             0c | Motor: 0
Test 7------------------------------------------------------------------------------------------------------------
select p2       Time: 210 | LCD:                 waiting for 20c | money:                             0c | Motor: 0
insert 20c      Time: 220 | LCD:                      dispensing | money:           inserted amount: 20c | Motor: 1
dispense fin.   Time: 230 | LCD:                            idle | money:                             0c | Motor: 0
select p2       Time: 240 | LCD:                    out of stock | money:                          ERROR | Motor: 0
Finish------------------------------------------------------------------------------------------------------------
```

Fig. 19 – terminal output

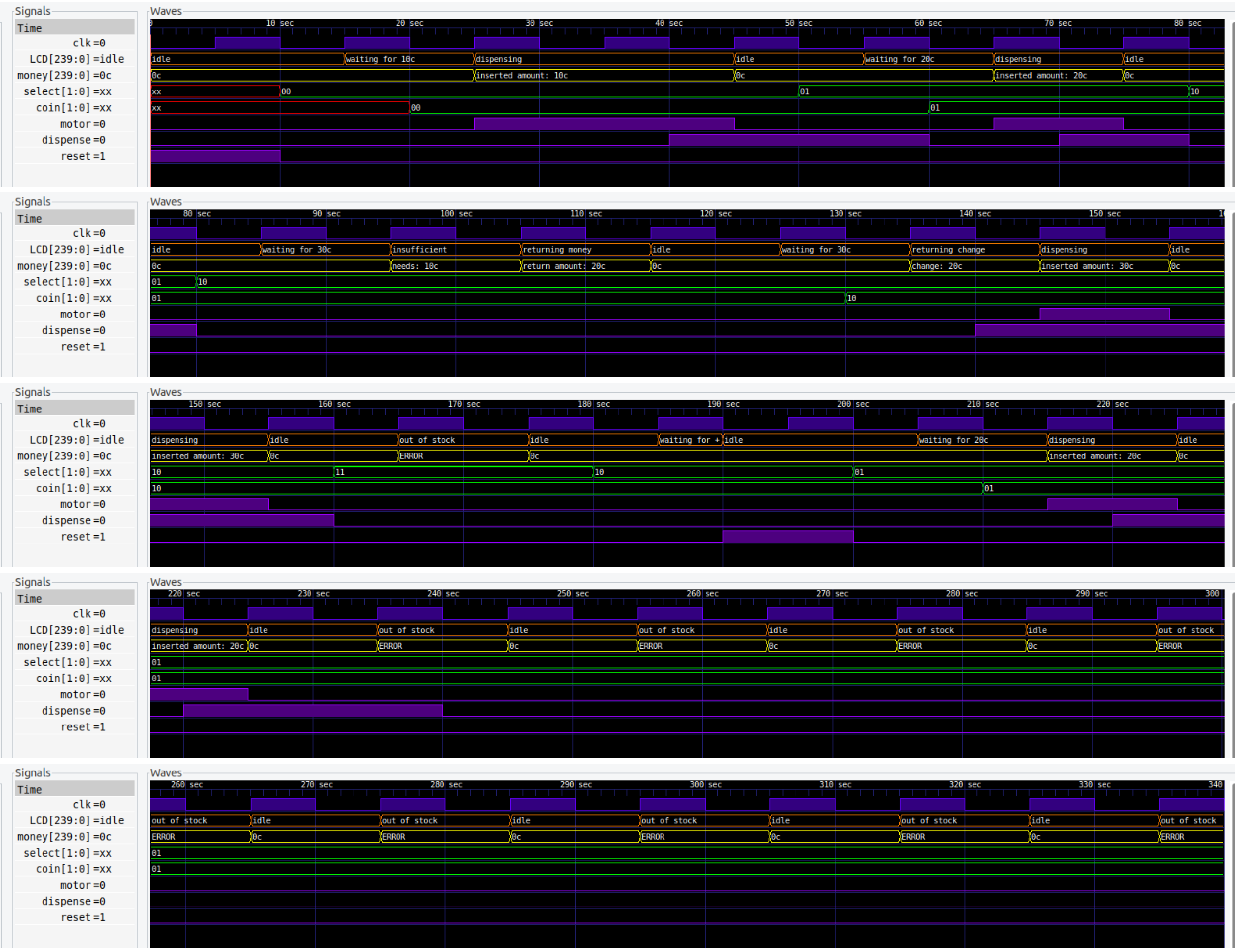


Fig. 20 -24 - waves

Compiled by **Icarus Verilog** and waves by **gtkwave**.

These files are attached to this report:

1. vending_machine.v
2. vending_machine_tb.v
3. test_vm.vcd
4. vending_machine_wave

***