ĐẠI HỌC QUỐC QIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

MILESTONE 1

DESIGN OF A VENDING MACHINE

GVHD: **Trần Hoàn Linh**

Cao Xuân Hải

Lớp: L01

Nhóm số: 16

Danh sách thành viên:

STT	Họ và tên	MSSV
1	PHẠM VIỆT HÙNG	2113592
2	Lê Trung Hiếu	2111185
3	Nguyễn Đức Hoàng	2110184

Thành phố Hồ Chí Minh – 2024

Mục lục

I. State Graph 2

II. Schamatic Module và các tín hiệu 3

1. Sơ đồ khối tổng thể thực hiện chức năng 3

2. Module Encode và mux 4

3. Adder và Sub 5

A. Adder 5

B. Sub 6

4. Bộ so sánh 7

5. Register total_reg và change_reg 9

A. Total_reg 10

B. Change_reg 11

6. Decoder 11

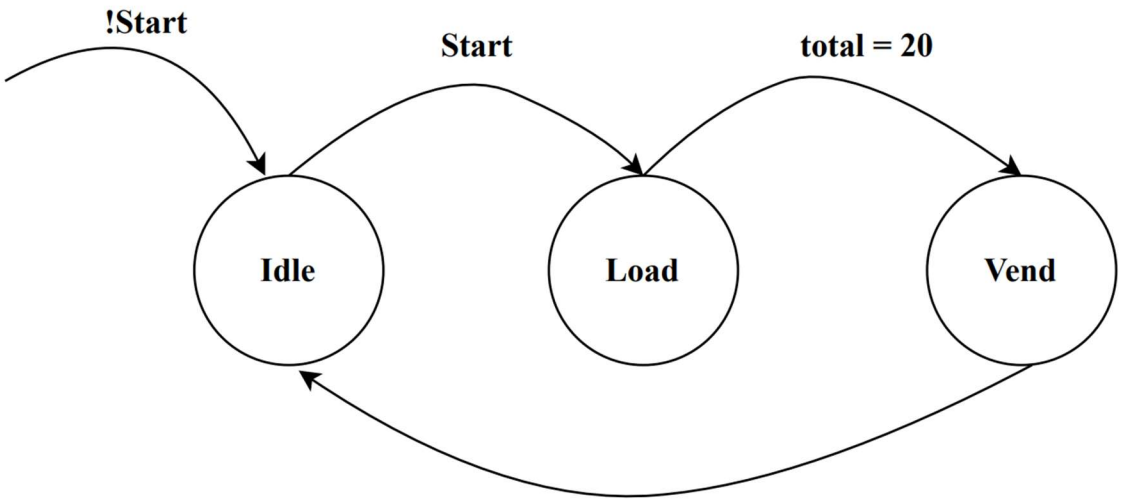
7. 7.FSM 13

8. Sơ đồ khối tổng 16

III. Testbench 17

I. State Graph

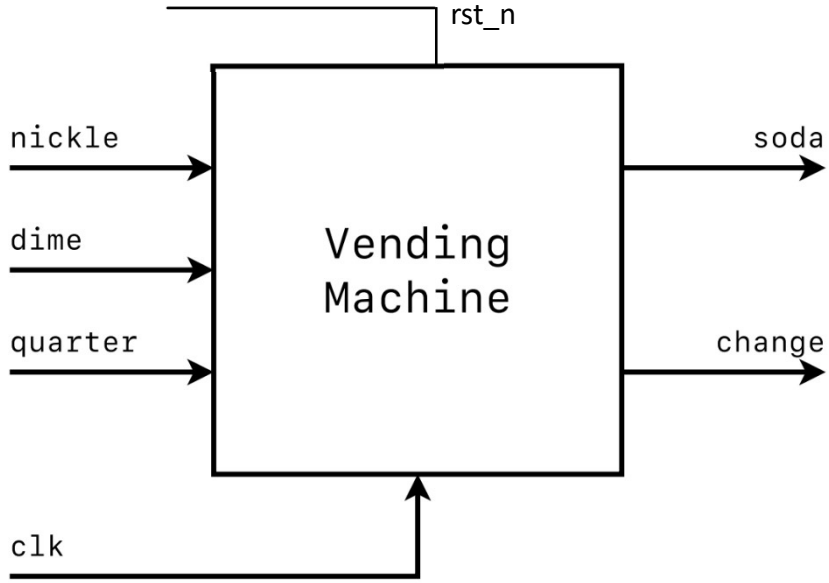
Để thực hiện hóa ý tưởng thiết kết ta chia thành 3 trạng thái để kiểm soát luồng tín hiệu ra và các register. Ba trạng thái là: Idle, Load, Vend.



State	Chức năng	Kích khởi khi	Next State
Idle	Khi chưa có xu bỏ vào máy thì trạng thái duy trì ở state Idle để chờ.	!Start	Load
Load	Khi có một đồng xu bỏ vào, tín hiệu State lên 1 thì liên tục cộng dồn tiền xu bỏ vào cho đến khi đạt được số tiền total ≥ ¢20	Start = 1	Vend
Vend	Khi total ≥ ¢20 thì cho tín hiệu output: soda lên 1 và trừ total cho 20 để tính tiền dư	Tín hiệu báo so sánh total ≥ ¢20 thỏa	Idle

II. Schamatic Module và các tín hiệu

1. Sơ đồ khối tổng thể thực hiện chức năng



Ngõ vào	Nickle, dime, quarter, clk, rst_n
Ngõ ra	Soda, change

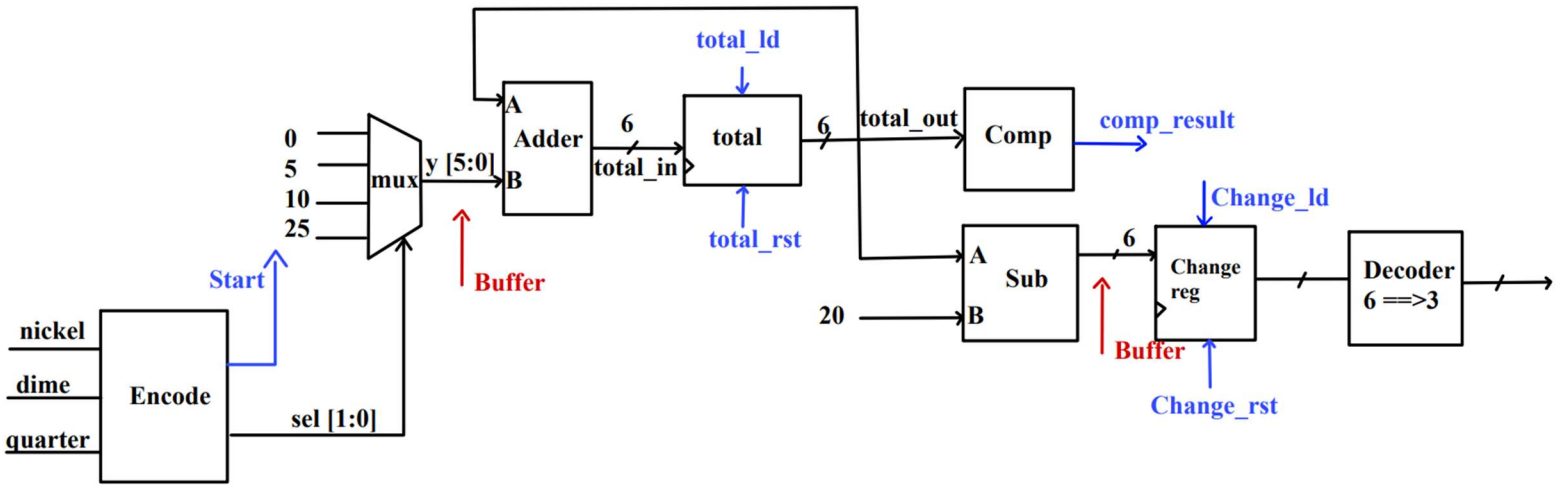


Figure 1 Sơ đồ tổng thể

Ta thêm các Buffer vào các vị trí màu đỏ để đảm bảo về mặt timing data.

Các tín hiệu điều khiển (màu xanh) là các tín hiệu điều khiển nối ra từ FSM đùng để điều khiển các register.

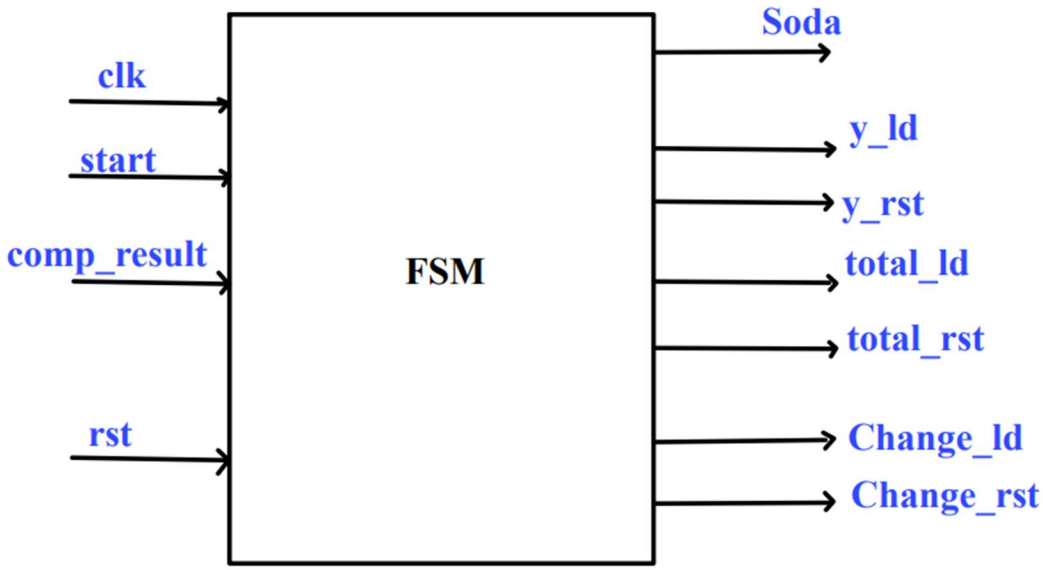


Figure 2: FSM

2. Module Encode và mux

Ngõ ra Start: Nhận biết có đồng xu được bỏ vào hay không bằng cách or 3 tín hiệu ngõ vào. Tín hiệu start=1 có nghĩa là đã có đồng xu được bỏ vào. Chuyển trạng thái từ Idle sang Load.

```
assign start = dime | quarter | nickel;
```

Ngõ ra sel với 3 bit đùng để select mux các giá trị tương đương về mệnh giá cho ra giá tiền ỏ ngõ ra y [5:0]

```
always @(*) begin
    case (sel)
        2'b00: y = 6'd0;
        2'b01: y = 6'd5;
        2'b10: y = 6'd10;
        2'b11: y = 6'd25;
        default: y = 6'd0;  // Default case (should not happen)
    endcase
end
```

3. Adder và Sub

Adder và Sub đề có nhân là bộ cla_6bit, dùng để cộng hoặc trừ các ngõ vào A cho ngõ vào B 6 bit

```
    module cla_6bit
(
    input wire [5:0] A, B,
    input wire Cin,
    output reg [5:0] S
   // output reg Co
);
    wire [5:0] P, G;
    wire [6:0] C;
    assign C[0] = Cin;

    generate
            genvar i;
            for (i = 0; i < 6; i = i + 1) begin : adder_loop
            assign G[i] = A[i] & B[i];    // Generate
        assign P[i] = A[i] ^ B[i];    // Propagate
        assign C[i+1] = G[i] | (P[i] & C[i]); // Carry
            assign S[i] = P[i] ^ C[i];    // Sum

        end
    endgenerate

            //assign Co=C[6];

endmodule
```

A. Adder

Để tính tổng giá tiền khách đã bỏ xu vào máy ta cộng dồn các giá trị từng đồng xu khách bỏ vào.

Total= Total + y

Tổng số tiền Total sẽ được lưu vào trong thanh ghi total_reg và hồi tiếp về bộ cộng để tiếp tục cộng với giá trị đồng xu y khác bỏ vào tiếp theo.

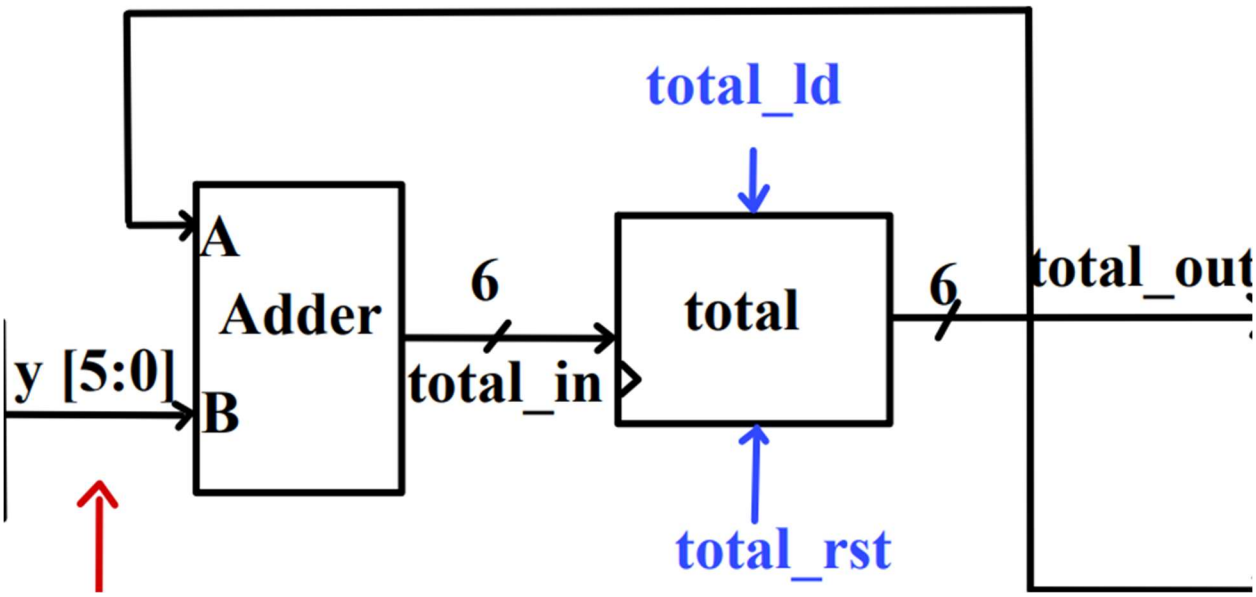


Figure 3 Sơ đồ khối bộ Adder

B. Sub

Để tính tiền thối lại khách hàng ta trừ giá trị tổng tiền đã nạp lưu trong register total_reg cho tiền một soda (¢20).

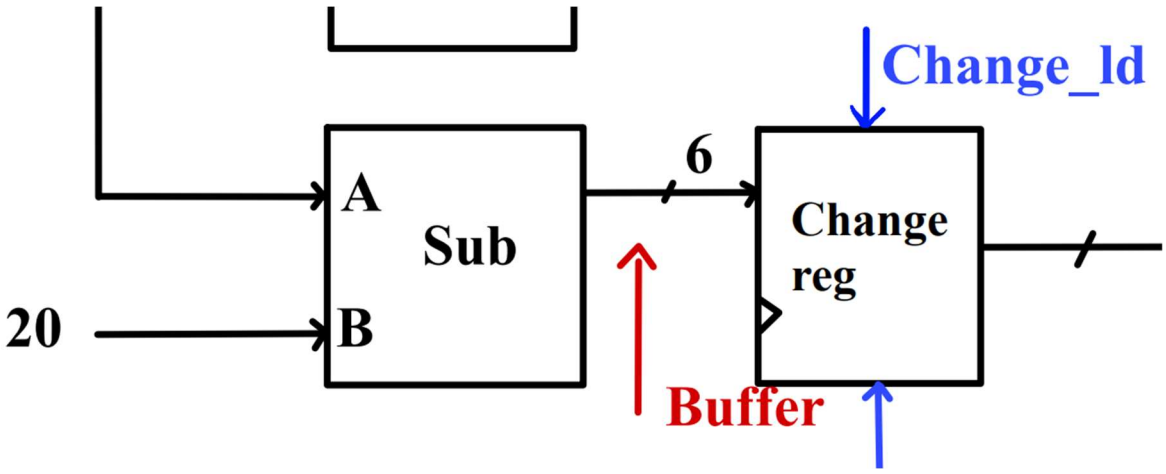


Figure 5: Sơ đồ khối bộ Sub

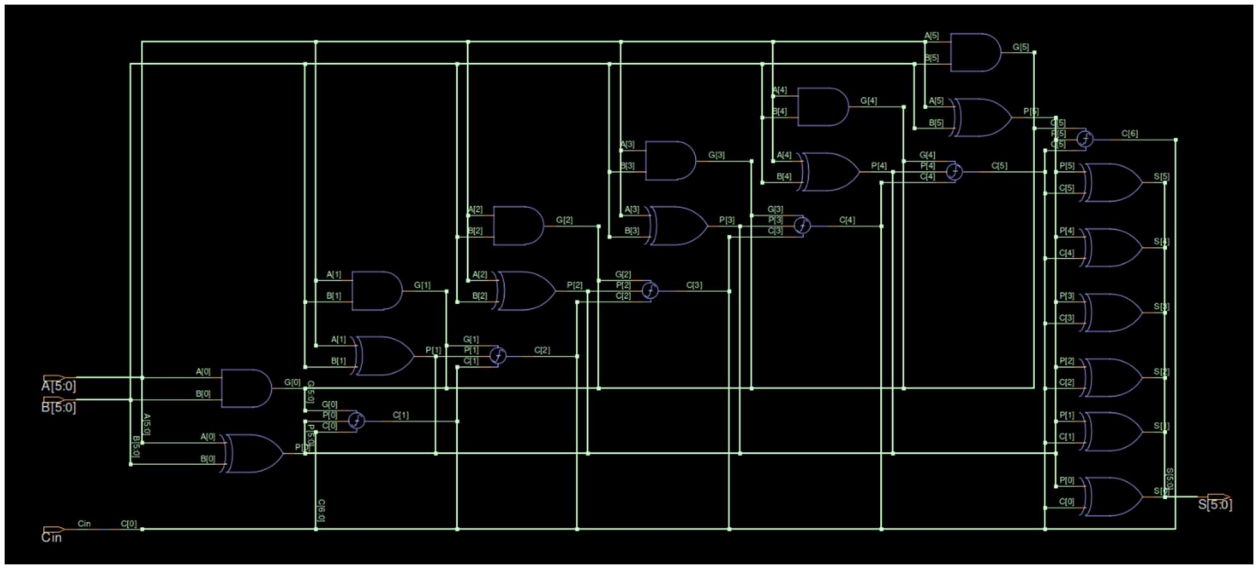


Figure 4: Schemetic CLA_Adder

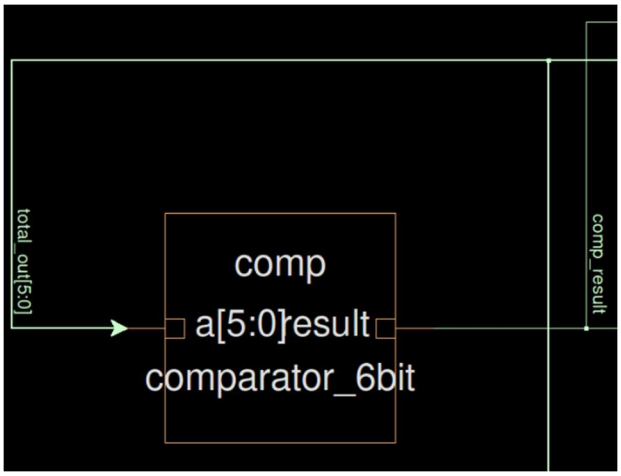
4. Bộ so sánh

Ngõ vào là giá trị của thanh ghi total, ta so sánh giá trị trong thanh ghi total với số 19. Nếu lớn hơn 19 thì ngõ ra result sẽ bằng 1

Ngõ vào total	Ngõ ra comp_result
total_out > 6'd19;	1
total_out < 6'd19;	0

```
module comparator_6bit (
    input [5:0] a,
    output reg result
);
```

```
        logic [5:0] b;
    reg [1:0] ab;
    always @(*) begin
        integer i;
        result = 1'b0;
            b =6'd19;
        for (i = 5; i >= 0; i = i - 1) begin
            ab = {a[i], b[i]};
            if (ab == 2'b10) begin
                result = 1'b1;
                break;
            end else if (ab == 2'b01) begin
                result = 1'b0;
                break;
            end
        end
    end
endmodule
```



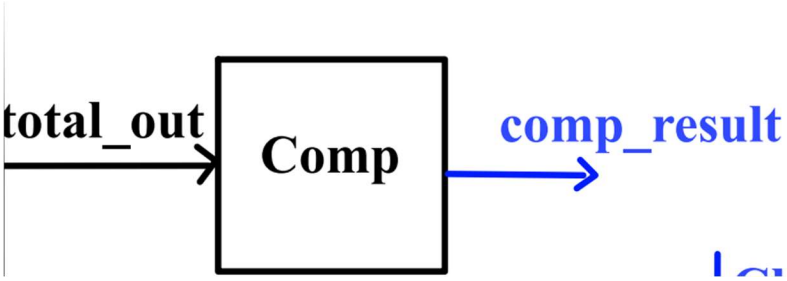


Figure 6: Ngõ vào ngõ ra của bộ so sánh

5. Register total_reg và change_reg

Total_reg và change_reg là 2 DFF 5 bit, ngõ vào D ngõ ra Q, rst và xung clk. Để enable DFF ta cho tín hiệu D vào một bộ mux ngõ vào còn lại là hồi tiếp từ Q, tương đương với việc không thay đổi ngõ vào. Sel là tín hiệu en.

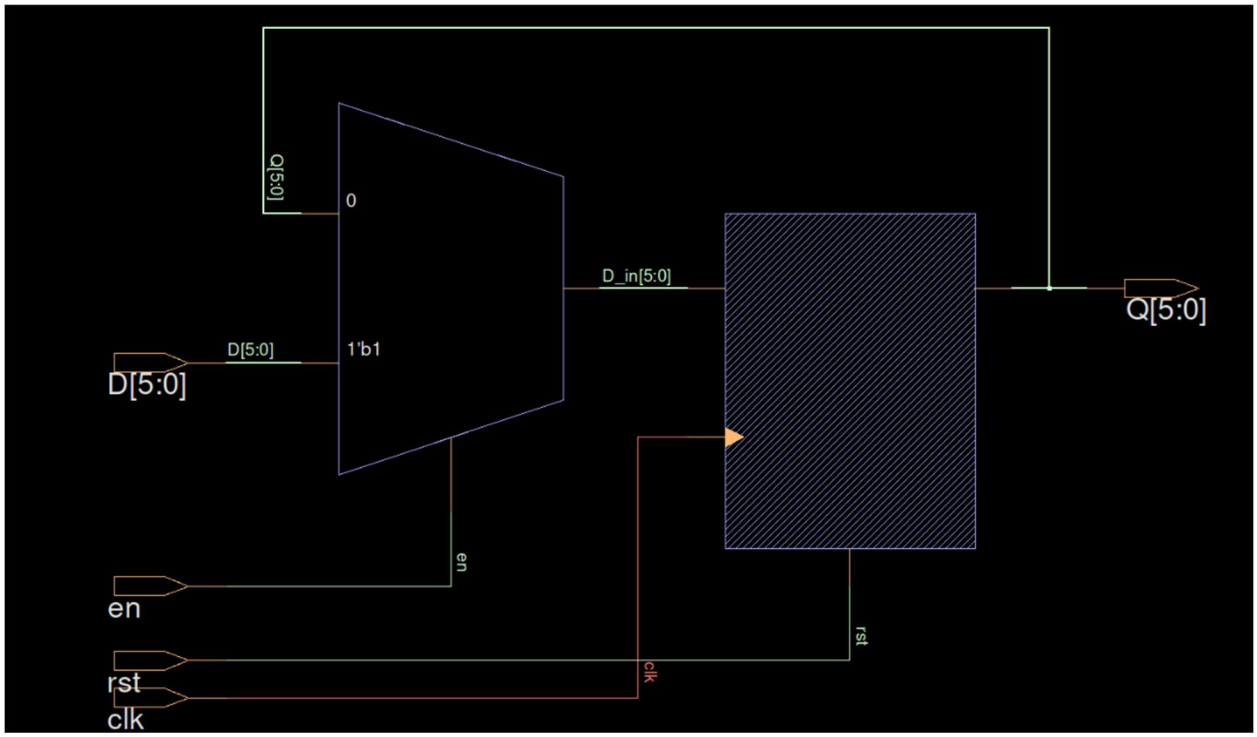


Figure 7: Schematic của Register

```
module register #(parameter WIDTH = 6)
(
    input logic rst,
    // input logic set,
    input logic en,
    input logic clk,
    input logic [WIDTH-1:0] D,
    output logic [WIDTH-1:0] Q
);

    logic [WIDTH-1:0] D_in;

    // Combinational logic to determine D_in
```

```
    always_comb begin
        if (en)
            D_in = D;
        else
            D_in = Q;
    end

    // Sequential logic to update Q on the rising edge of the clock
    always_ff @(posedge clk or posedge rst) begin
        if (rst)
            Q <= 0;
        else
            Q <= D_in;
    end

endmodule
```

A. Total_reg

Total_reg có nhân là register được nối chân như sau:

```
 register  total_reg          (.rst(total_rst),
                               .clk(clk),
                               .en(total_ld),
                               .D(total_in),
                               .Q(total_out)
                               );
```

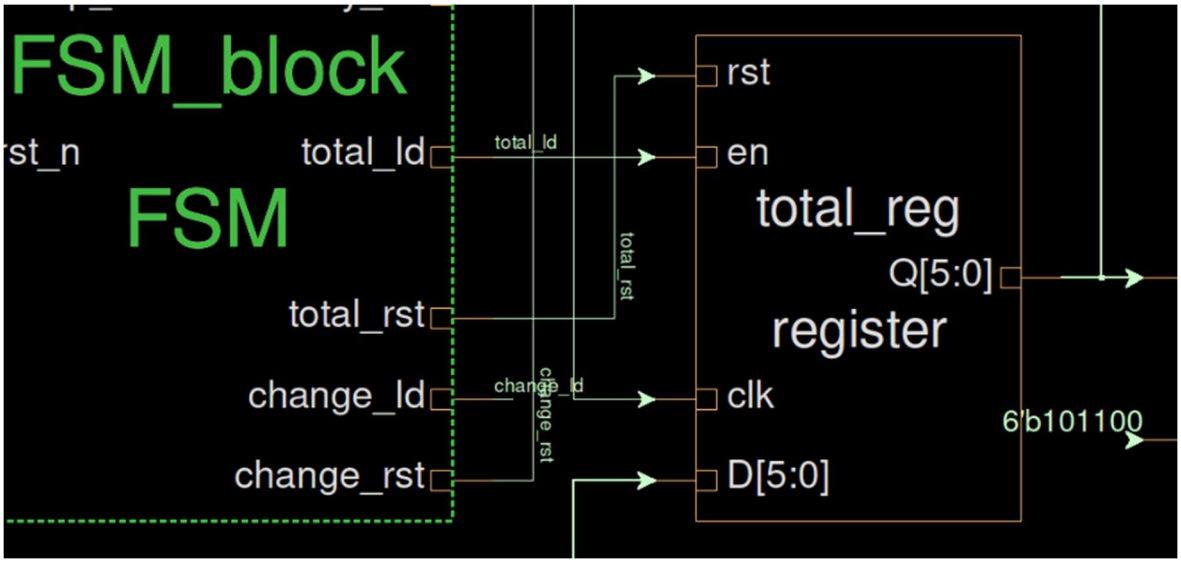


Figure 8: Sơ đồ ngõ vào ngõ ra của total_reg với các tín hiệu điều khiển được nối với FSM

B. Change_reg

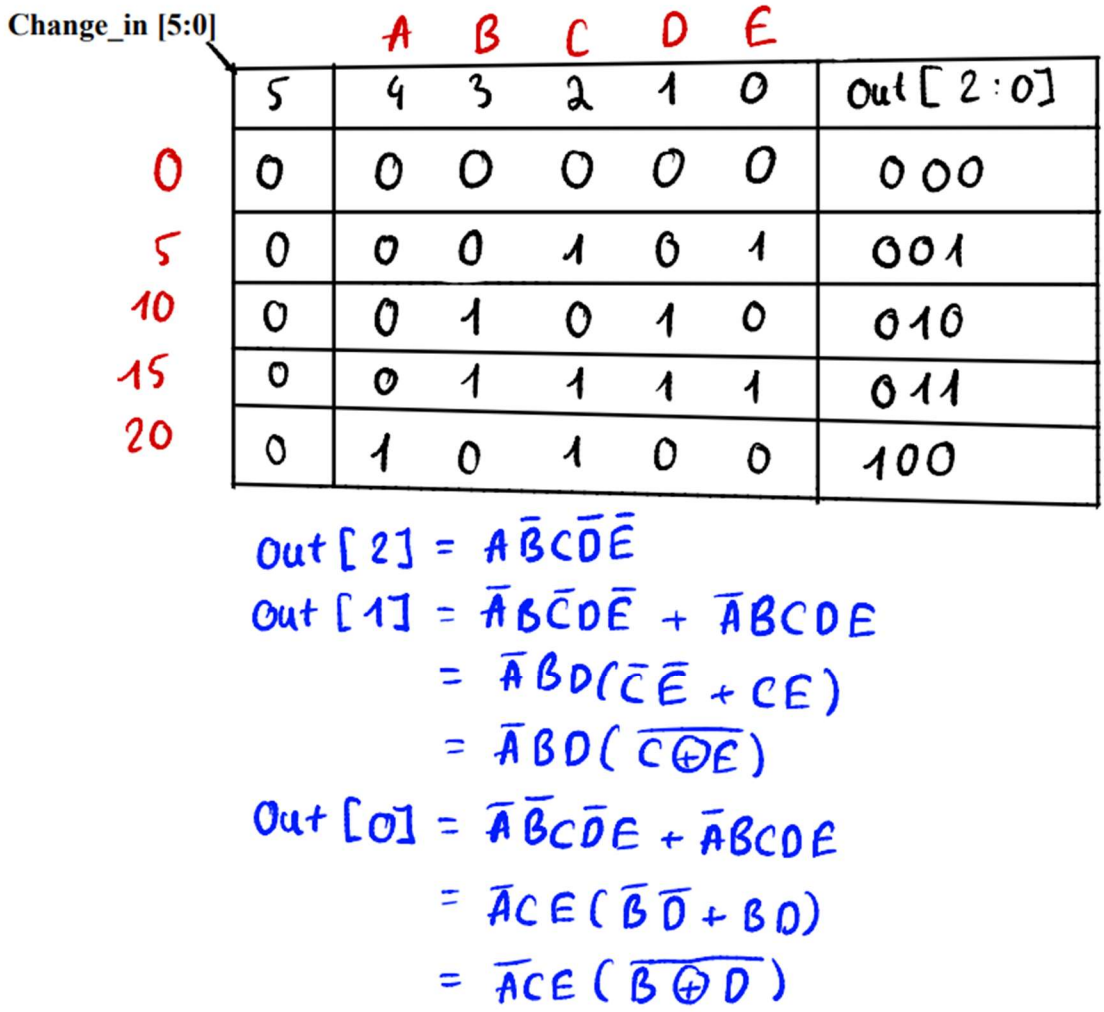
Change_reg có nhân là register được nối chân như sau:

```
register  change_reg      (.rst(change_rst),
                           .clk(clk),
                           .en(comp_result),
                           .D(sub_out),
                           .Q(change_in)
                           );
```

6. Decoder

Decoder dùng để chuyển từ 6 bit tương ứng với số tiền thối còn lại thành mã 3 bit đã được định sẵn từ trước

000 ¢0
001 ¢5
010 ¢10
011 ¢15
100 ¢20




```
module change_decode(
    input wire [5:0] in,       // Đầu vào 6-bit
    output reg [2:0] change    // Đầu ra 3-bit
);



        // Gán giá trị cho từng bit của tín hiệu change
  // Sử dụng khối always để gán giá trị cho reg
    always @(*) begin
        // Gán giá trị cho từng bit của tín hiệu change
        change[2] = in[4] & ~in[3] & in[2] & ~in[1] & ~in[0];
```

```
    change[1] = ~in[4] & in[3] & in[1] & (in[2] | ~in[0]);
    change[0] = ~in[4] & in[2] & in[0] & (in[3] | ~in[1]);
  end


endmodule
```

7. 7.FSM

Bộ FSM quyết định việc các trạng thái của máy bán nước, quyết định các bước nhảy trạng thái cũng như ngõ ra và điều khiển việc nạp các thanh ghi

FSM là một hệ gồm một bộ combination (hệ tổ hợp) để quyết định trạng thái nhảy và sequential (hệ tuận tự) dùng để thực hiện nhảy.

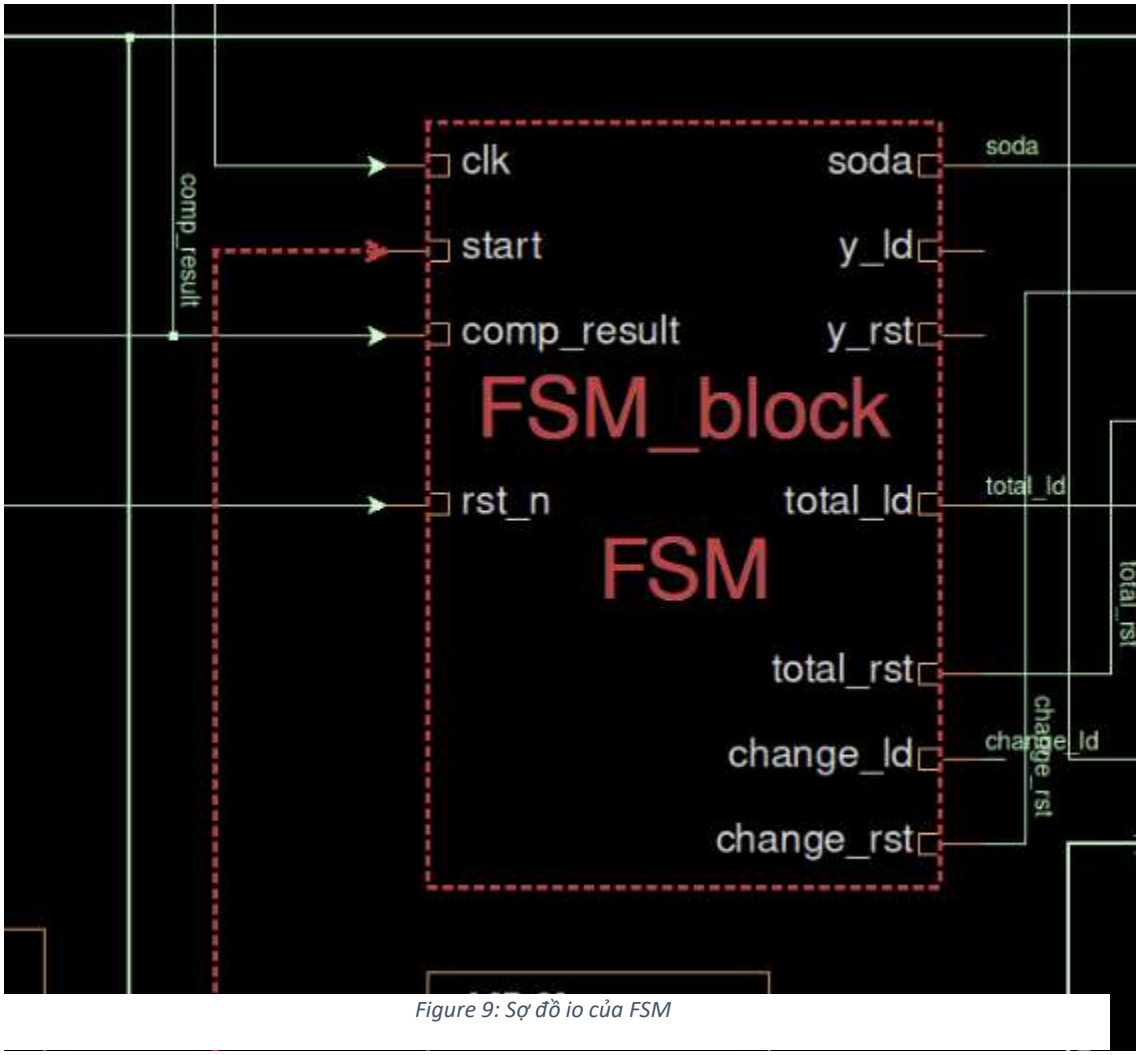


Figure 9: Sợ đồ io của FSM

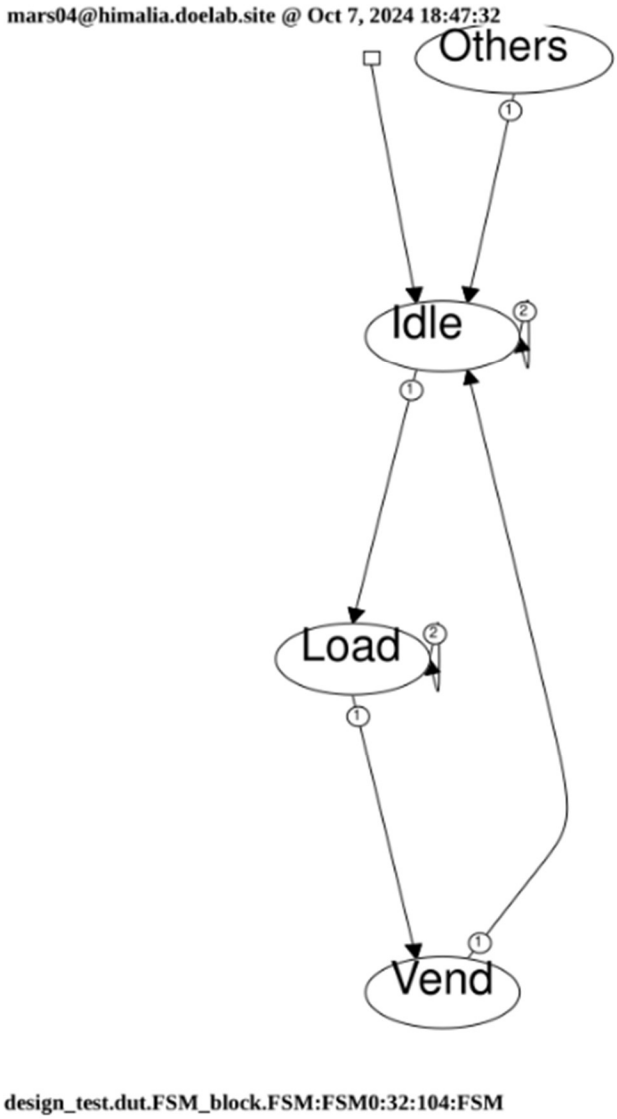
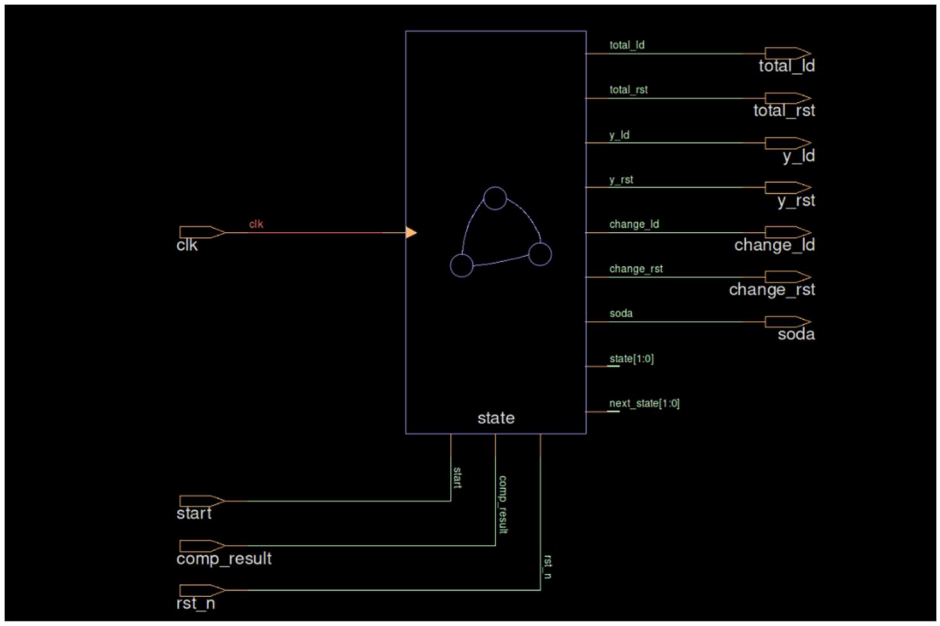


Figure 10: Bảng chuyển trạng thái mô phỏng trên tool verdi


clk
clk
state
total_ld
total_ld
total_rst
total_rst
y_ld
y_ld
y_rst
y_rst
change_ld
change_ld
change_rst
change_rst
soda
soda
state[1:0]
next_state[1:0]
start
start
comp_result
comp_result
rst_n
rst_n

8. Sơ đồ khối tổng

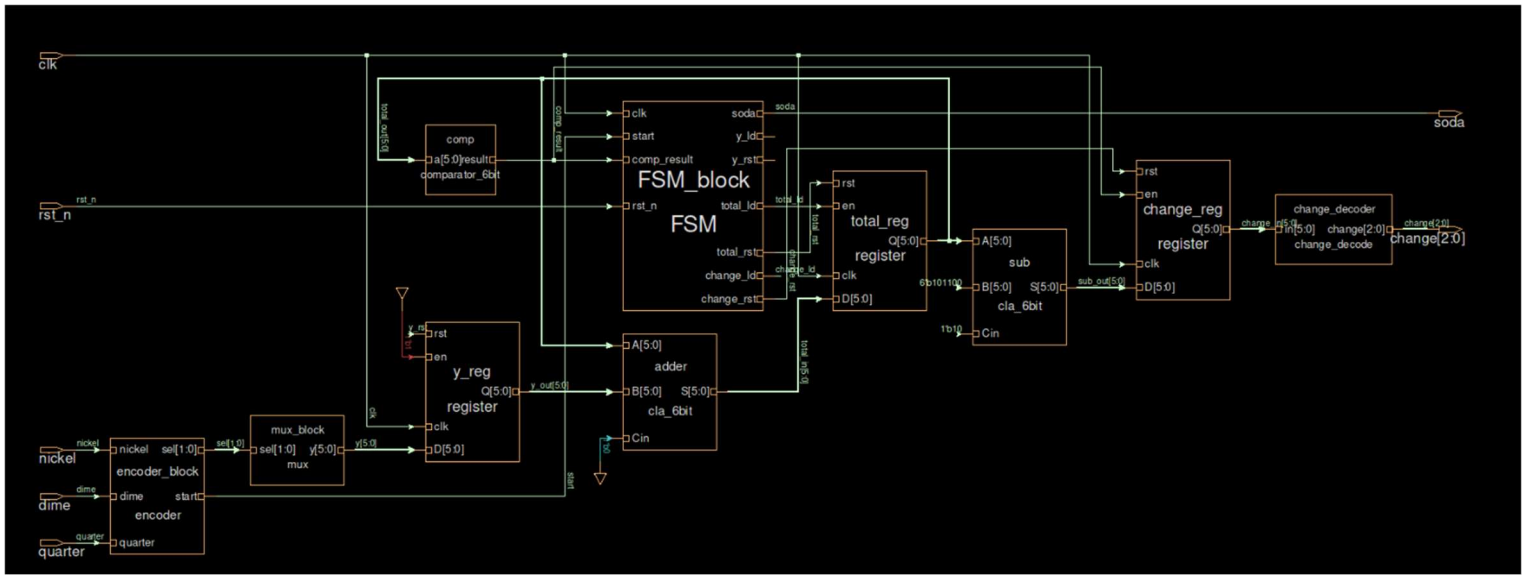


Figure 11: Sơ đồ khối tổng

9.

```
File  Edit  View  Search  Terminal  Help
../00_src/encoder.sv
../00_src/mux.sv
../00_src/cla_6bit.sv
../00_src/register.sv
../00_src/comparator_6bit.sv
../00_src/change_decode.sv
../00_src/FSM.sv
../00_src/vending_machine.sv
```

III. Testbench

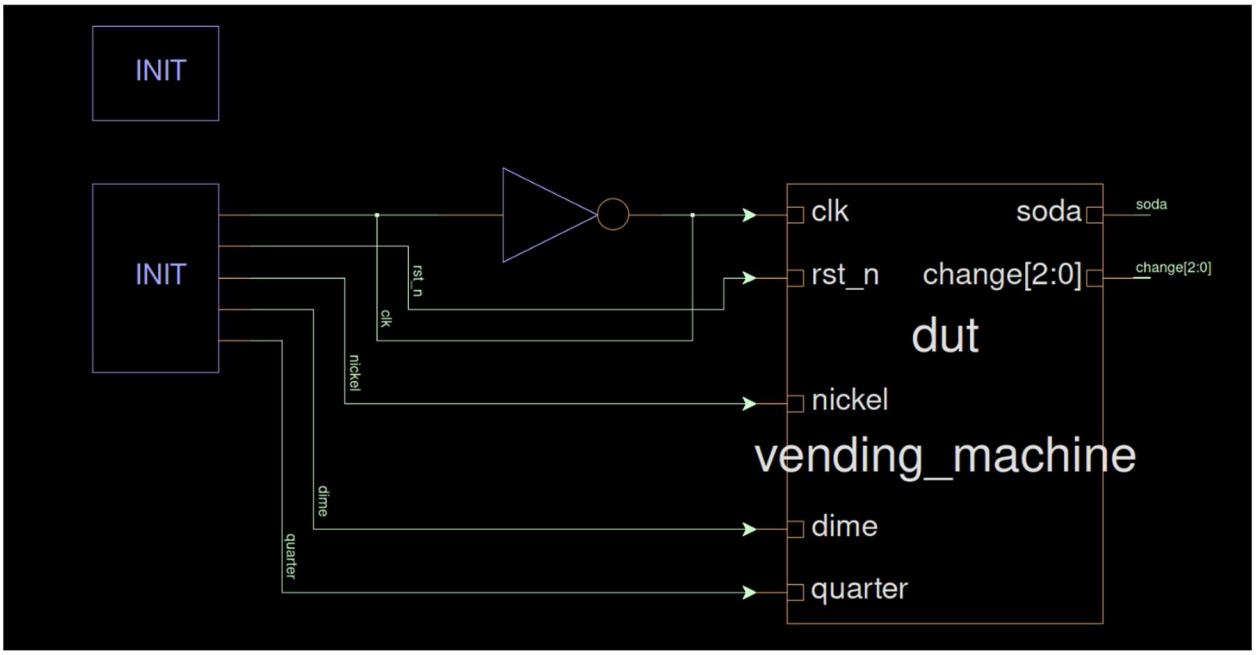


Figure 12: Sơ đồ khối top module

Figure 13: wave form

```
module design_test();
   reg clk;
   reg rst_n;           // Tín hiệu reset active-low
   reg nickel;          // Đầu vào đồng 5 xu
   reg dime;            // Đầu vào đồng 10 xu
   reg quarter;         // Đầu vào đồng 25 xu
   wire soda;           // Đầu ra phát soda
   wire [2:0] change;   // Đầu ra tiền thừa (change)
```

```
// Khởi tạo module vending_machine
vending_machine dut (
    .clk(clk),
    .rst_n(rst_n),
    .nickel(nickel),
    .dime(dime),
    .quarter(quarter),
    .soda(soda),
    .change(change)
);
initial begin
    $fsdbDumpfile("design_test.fsdb");
    $fsdbDumpvars(0, design_test, "+all","+mda");
end
  always #5 clk = ~clk;

// Khối kiểm tra
initial begin
    // Khởi tạo tín hiệu
    clk = 0;
    rst_n = 0;
    nickel = 0;
    dime = 0;
    quarter = 0;

    // Reset hệ thống
    #10;
    rst_n = 1;

    // Test case 1: Đưa vào 5 xu
    #10;
    nickel = 1;
    #10;
    nickel = 0;

    // Test case 2: Đưa vào 10 xu
    #10;
    dime = 1;
    #10;
    dime = 0;

    // Test case 3: Đưa vào 25 xu
    #10;
```

```
            quarter = 1;
            #10;
            quarter = 0;

            // Test case 4: Đưa vào 5 xu và 10 xu (tổng 15 xu)
            #10;
            nickel = 1;
            #10;
            nickel = 0;
            dime = 1;
            #10;
            dime = 0;

            // Test case 5: Đưa vào 25 xu và 10 xu (tổng 35 xu, đủ để mua soda)
            #10;
            quarter = 1;
            #10;
            quarter = 0;
            dime = 1;
            #10;
            dime = 0;

            // Đợi một thời gian để quan sát kết quả
            #50;

            // Kết thúc mô phỏng
            $finish;
        end

        // Theo dõi giá trị của tín hiệu
        initial begin
            $monitor("Time: %0t | Soda: %b | Change: %b | Nickel: %b | Dime: %b |
Quarter: %b",
                    $time, soda, change, nickel, dime, quarter);
        end

    endmodule
```