ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

NĂM HỌC 2019 – 2020

-------*-------

KỸ THUẬT SỐ NÂNG CAO

BÁO CÁO LAB

GVHD: TRẦN HOÀNG LINH

Tên	MSSV
Nguyễn Duy Tân	1713068
Nguyễn Mỹ Hằng	1711215

TP.HCM, ngày 15 tháng 12 năm 2019

BÁO CÁO THIẾT KẾ CPU LAB 1

Mục Lục

1. Mục tiêu

Mục tiêu của bài thí nghiệm này là xây dựng bộ cộng 4 bit bằng hai cách:
. Sử dụng bộ cộng toàn phần (full adder) để thực hiện (structural model).
. Sử dụng mô tả hành vi để thực hiện (behavioral model).

2. Lý thuyết hoạt động của mạch

a. Nhắc lại về bộ cộng full adder:

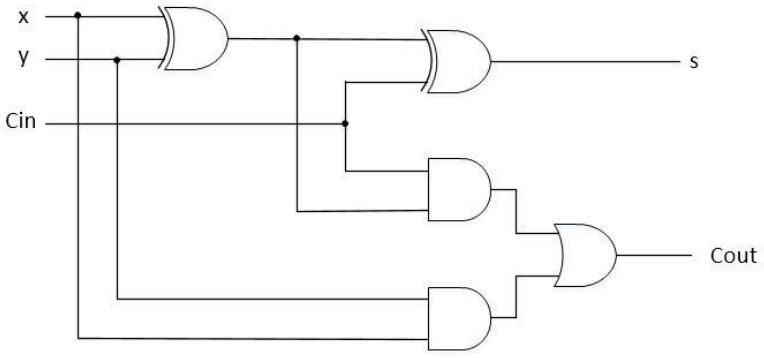
Bộ cộng full adder bao gồm ba ngõ vào là x,y và bit nhớ Cin (thông thường cho Cin = 0) và có hai ngõ ra là kết quả tổng s và bit nhớ Cout. Bảng sự thật của bộ cộng full adder được cho ở Bảng 2.1.

x	y	Cin	s	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Bảng 2.1 Bảng sự thật của bộ full adder.

b. Sơ đồ nguyên lý và nguyên lý hoạt động.

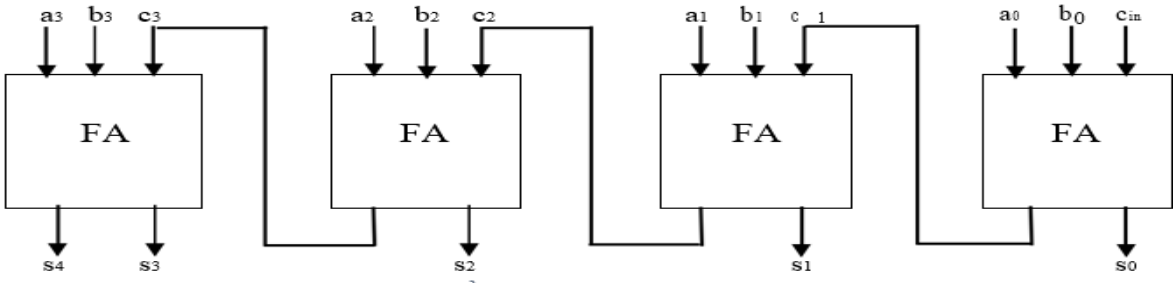
Từ bảng sự thật trên ta có sơ đồ nguyên lý của mạch full adder như Hình 2.1.



Hình 2.1 Sơ đồ nguyên lý của bộ full adder.

Như vậy để thực hiện bộ cộng 4 bit ta sử dụng 4 bộ cộng full adder với cách nối các tín hiệu như Hình 2.2. Sau khi có được sơ đồ khối ta tiến hành viết code design và code testbench.

Hình 2.2 Sơ đồ nguyên lý của bộ full adder 4 bit.



3. Code

a. Code full_adder.

```
library ieee;
use     ieee.std_logic_1164.all;
entity  fulladder   is
port(
    x,y,z: in std_logic;
    s,c: out std_logic
    );
end     fulladder;
architecture equation of fulladder is
begin
    s <= x xor y xor z;
    c <= (x and y) or (y and z) or (x and z);
end equation;
```

b. Code lab_1.

```vhdl
library ieee;
use     ieee.std_logic_1164.all;
entity  lab1 is
port(
    cin: in std_logic;
    a,b: in std_logic_vector(3 downto 0);
    s: out std_logic_vector(3 downto 0);
    cout: out std_logic
    );
end lab1;
architecture structure of lab1 is
signal c : std_logic_vector(1 to 3);
component fulladder
port(
    x,y,z: in std_logic;
    s,c: out std_logic
    );
end component;
begin
stage0: fulladder port map (a(0),b(0),cin,s(0),c(1));
stage1: fulladder port map (a(1),b(1),c(1),s(1),c(2));
stage2: fulladder port map (a(2),b(2),c(2),s(2),c(3));
stage3: fulladder port map (a(3),b(3),c(3),s(3),cout);
end structure;
```

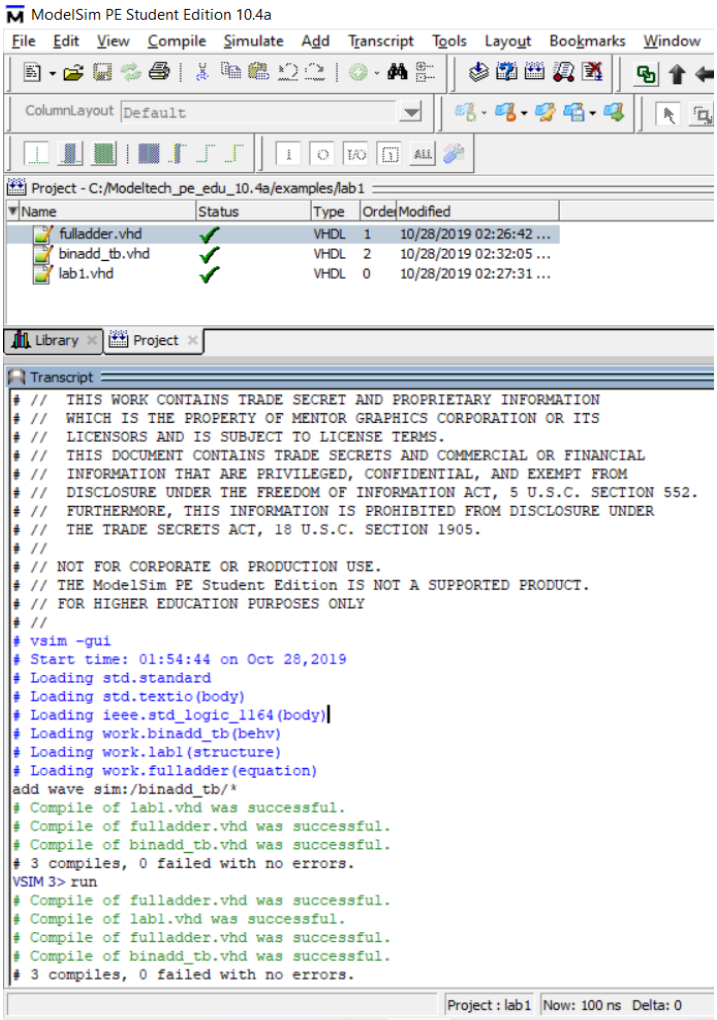
c. Code test.

```vhdl
library ieee;
use  ieee.std_logic_1164.all;
entity binadd_tb is
end binadd_tb;

architecture behv of binadd_tb is
component lab1
port(
     a,b: in std_logic_vector(3 downto 0);
     cin: in std_logic;
     s: out std_logic_vector(3 downto 0);
     cout: out std_logic
     );
end component;

signal a: std_logic_vector(3 downto 0) := (others => '0');
signal b: std_logic_vector(3 downto 0) := (others => '0');
signal cin : std_logic := '0';
signal s: std_logic_vector(3 downto 0);
signal cout : std_logic;

 begin
     uut: lab1 port map(
     a => a,
     b => b,
     cin => cin,
```

```
s => s,
cout => cout
);

strim_proc_a: process
begin
a <= "0100";
wait for 100 ns;
a <= "0111";
wait for 100 ns;
a <= "0110";
wait for 100 ns;
a <= "0101";
wait for 100 ns;
a <= "1100";
wait;
end process;

strim_proc_b: process
begin
b <= "1111";
wait for 100 ns;
b <= "0001";
wait for 100 ns;
b <= "0011";
wait for 100 ns;
b <= "1010";
wait for 100 ns;
b <= "1011";
wait;
end process;

strim_proc_cin: process
begin
cin <= '0';
wait for 100 ns;
wait;
end process;

end;
```

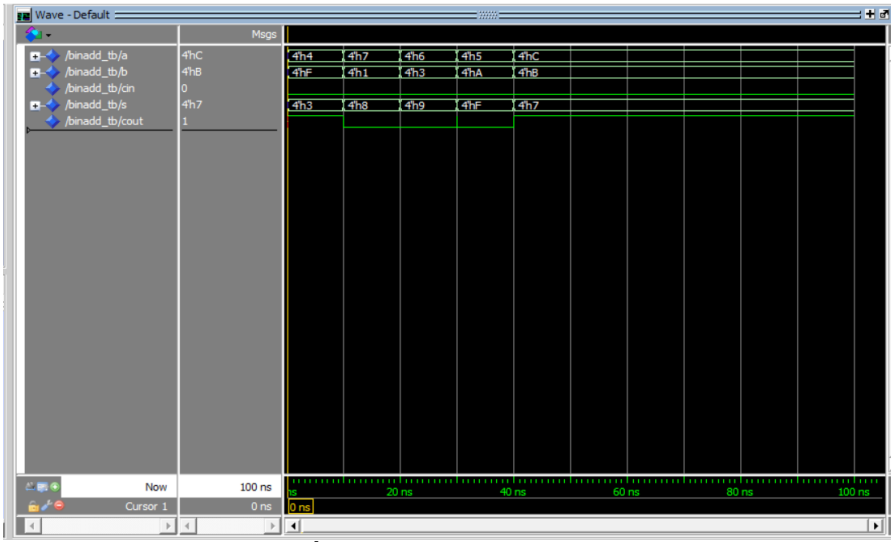
4. Chạy chương trình và mô phỏng trên ModelSim.

a. Chạy chương trình.



Hình 4.1 Màn hình chạy chương trình trên ModelSim.

b. Kết quả chạy mô phỏng.



Hình 4.2 Kết quả dạng sóng trên ModeSim.

5. Bản sự thật và kết luận

a	b	Cin	s	Cout
0H		0		
1H		0		
2H		0		
3H		0		
4H	FH	0	3H	1
5H	AH	0	FH	0
6H	3H	0	9H	0
7H	1H	0	8H	0
8H		0		
9H		0		
AH		0		
BH		0		
CH	BH	0	7H	1
DH		0		
EH		0		
FH		0		

Bảng 5.1 Bảng sự thật trích trừ 64 trường hợp của bộ full adder 4 bit.

Kết luận: Mô phỏng đúng với bản sự thật.

LAB 2: THIẾT KẾ VÀ CÀI ĐẶT BỘ ALU BẰNG VERILOG

I. Mục tiêu:

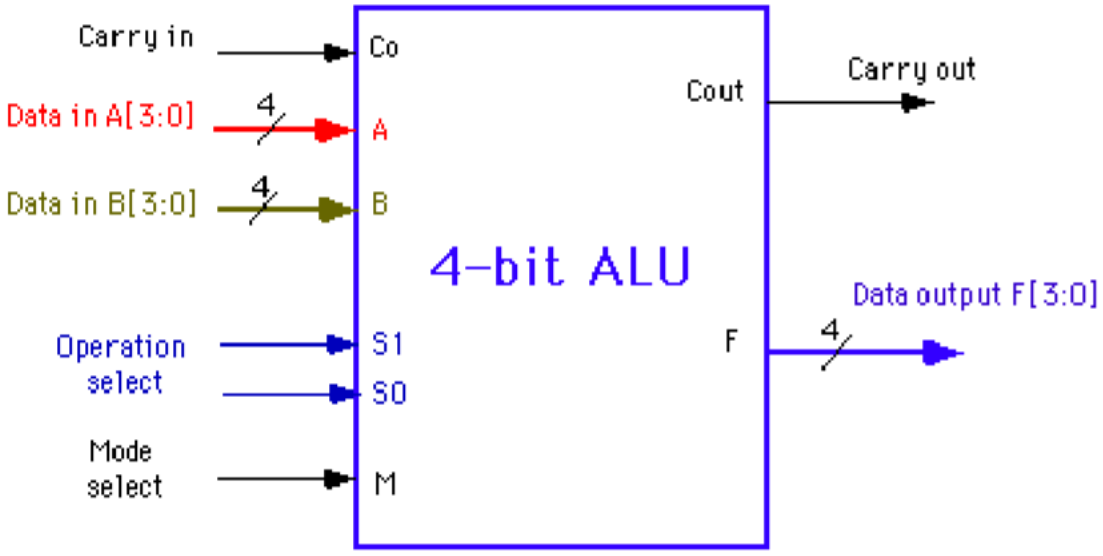
- Thiết kế bộ ALU.
- Cài đặt bộ ALU bằng Verilog sau đó kiểm tra hoạt động.

II. Yêu cầu thiết kế

a. Bảng chức năng:

M	S1	S0	Chức năng	Tác vụ
0	0	0	Ai.Bi	AND
0	0	1	Ai+Bi	OR
0	1	0	Ai(+)Bi	XOR
0	1	1	~Ai(+)Bi	XNOR
1	0	0	A+C0	Cộng A với Carry
1	0	1	A+B+C0	Cộng A, B và Carry
1	1	0	A+B'+C0	Cộng A với bù B và Carry
1	1	1	A'+B+C0	Cộng B với bù A và Carry

b. Sơ đồ khối:



Input: C0, [3:0]A, [3:0]B, [1:0]S, M
Output: [3:0]F, Cout

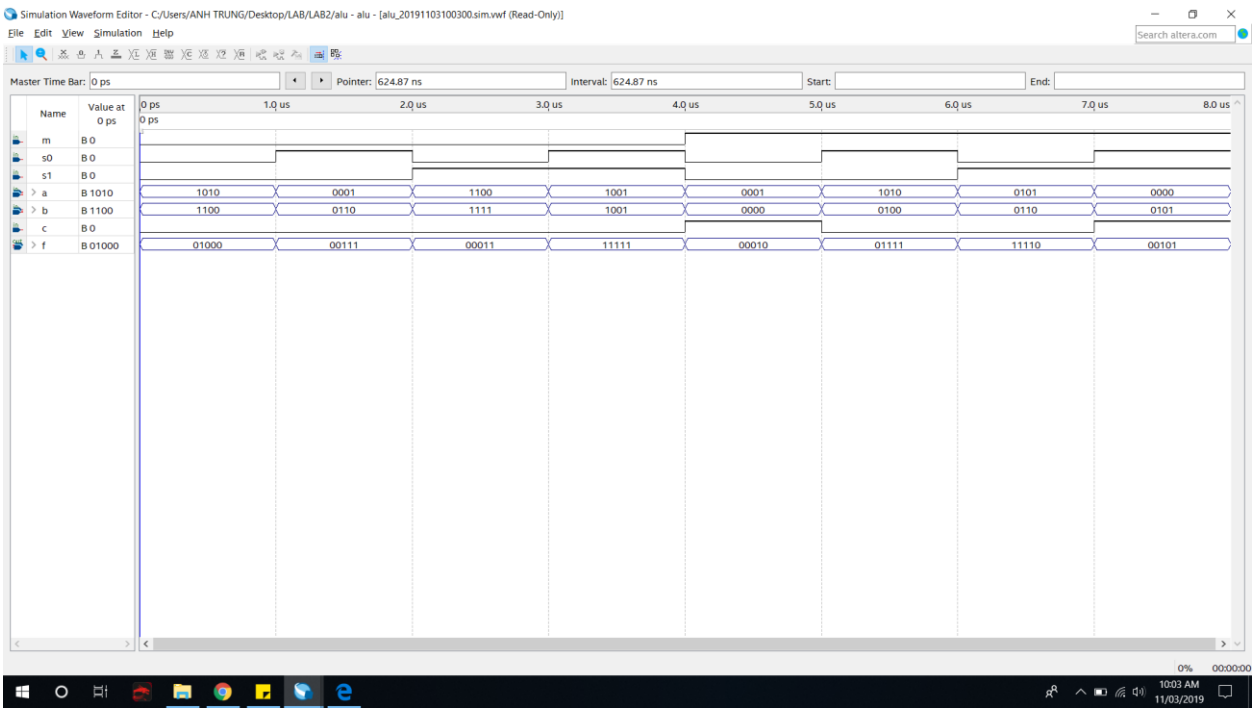
III. Nội dung file code sử dụng Verilog:

alu.v

```
1  module alu(a,b,c,m,s0,s1,f);
2  input [3:0]a,b;
3  input c,m,s0,s1;
4  output [4:0]f;
5  reg [4:0]f;
6
7  always@(*)
8     if (!m)
9        case({s1,s0})
10          2'b00: f=a&b;
11          2'b01: f=a|b;
12          2'b10: f=a^b;
13          default: f=a~^b;
14       endcase
15    else
16       case({s1,s0})
17          2'b00: f=c?(a+4'b0001):(a+4'b0000);
18          2'b01: f=c?(a+b+4'b0001):(a+b+4'b0001);
19          2'b10: f=c?(a+~b+4'b0001):(a+~b+4'b0000);
20          default: f=c?(~a+b+4'b0001):(~a+b+4'b0000);
21       endcase
22 endmodule
23
```

***Nhận xét:** Dùng mô tả hành vi với always@() để cài đặt bộ ALU, ta có đoạn code như trên với 2 mode phân biệt và 4 chức năng cho mỗi mode.

IV. Dạng sóng ra:



Chọn bộ dữ liệu đầu vào với đủ 8 chức năng đã xây dựng trên bộ ALU ta có dạng sóng ngõ ra như hình trên (Cout=F[4]).

V. Nhận xét:

- Từ kết quả thu được khi quan sát dạng sóng ngõ ra, ta thấy bộ ALU đã vận hành và cho kết quả đúng như mong muốn với việc xử lí các phép toán logic 4-bit hay các phép toán số học 4-bit.
- Vậy bộ ALU đã cài đặt thỏa yêu cầu thiết kế.

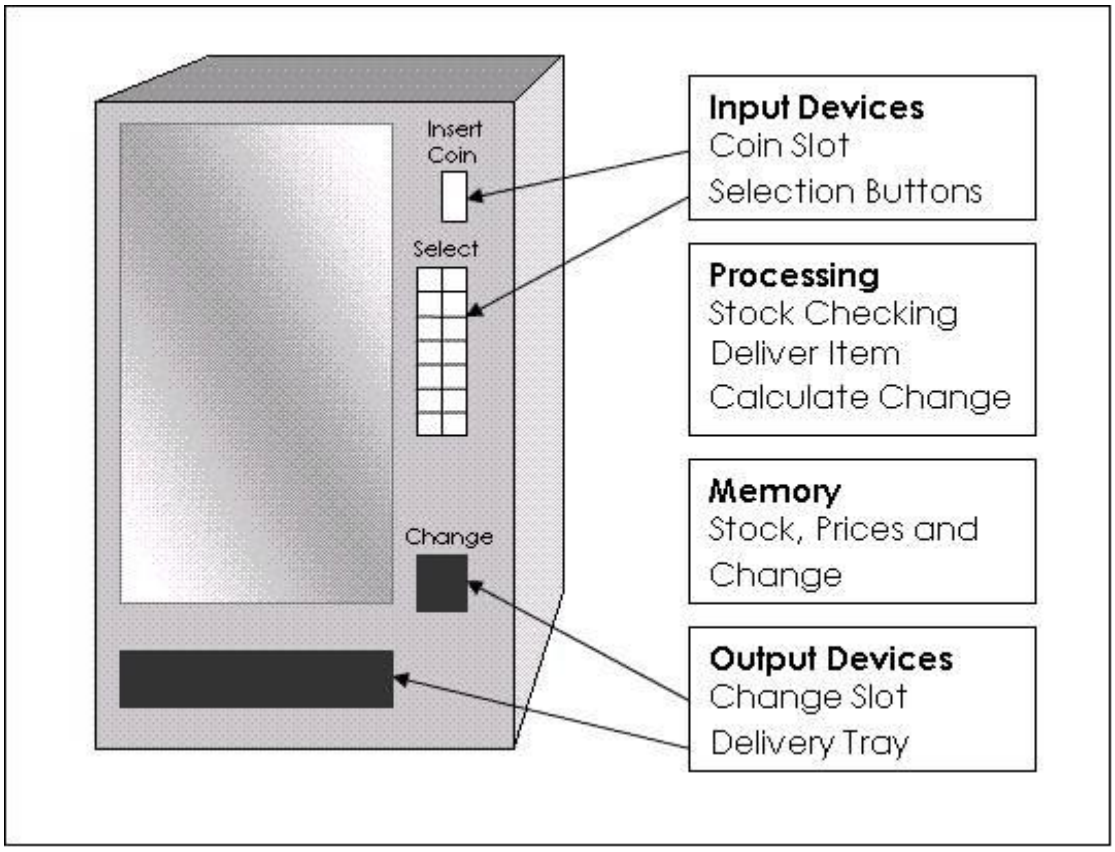
LAB 3

Thiết kế máy bán nước ngọt tự động

I. YÊU CẦU THIẾT KẾ:

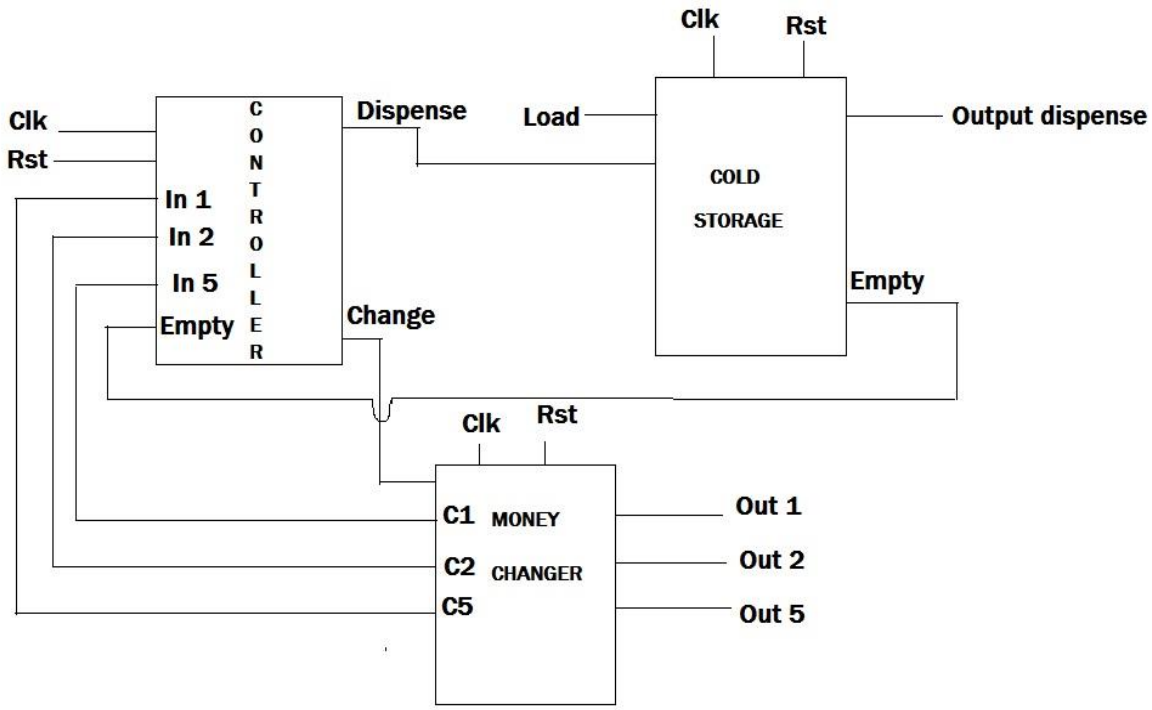
Thiết kế máy bán nước ngot tự động (vending machine) tuân thủ các nguyên tắc sau:

- Nước ngọt (Soda) giá 9000, nước suối (Water) giá 7000
- Máy nhận xu: 1000, 2000 và 5000 (N, D, Q)
- Số tiền trả lại sao cho số xu ít nhất (Give change in the smallest # coins possible)
- Nếu số tiền bỏ vào lớn hơn 9000 máy sẽ tự trả lại tiền vừa bỏ vào sau.
- Máy có nút Coin Return (CR) dùng để trả lại hết tiền vừa bỏ vào.
- Nếu không có nút nào được ấn thì máy trạng thái giữ nguyên trạng thái cũ.
- Các ngõ ra:
    - Coin Return out (trả hết tiền khi CR được bấm)
    - Water out (WO) (mua nước suối)
    - Soda out (SO) (mua nước ngọt)
    - Change (CO) (trả tiền thừa)



II. SƠ ĐỒ KHỐI:

Sơ đồ khối của Vending machine:



III. CODE VERILOG:

```
module fsm(clk,reset,coin,vend,state,CO);

input clk;
input reset;
input [2:0]coin;
output vend;
output [2:0]state;
output [2:0]CO;

reg vend;
reg [2:0]CO;
wire [2:0]coin;


parameter [2:0]N=3'b001;
parameter [2:0]D=3'b010;
parameter [2:0]N_D=3'b011;
parameter [2:0]D_D=3'b100;
parameter [2:0]Q=3'b101;
```

```
parameter [2:0]ttnghi=3’b000;
parameter [2:0]FIVE=3’b001;
parameter [2:0]TEN=3’b010;
parameter [2:0]FIFTEEN=3’b011;
parameter [2:0]TWENTY=3’b100;
parameter [2:0]TWENTYFIVE=3’b101;

reg [2:0]state,next_state;

always @(state or coin)
begin next_state=0;
case(state)
    ttnghi: case(coin)
                N: next_state=FIVE;
                D: next_state=TEN;
                Q: next_state=TWENTYFIVE;
                default: next_state=ttnghi;
            endcase
    FIVE: case(coin)
                N: next_state=TEN;
                D: next_state=FIFTEEN;
                Q: next_state=TWENTYFIVE;
                default: next_state=FIVE;
                endcase
    TEN: case(coin)
                N: next_state=FIFTEEN;
                D: next_state=TWENTY;
                Q: next_state=TWENTYFIVE;
                default: next_state=TEN;
                endcase FIFTEEN:
            case(coin)
                N: next_state=TWENTY;
                D: next_state=TWENTYFIVE;
                Q: next_state=TWENTYFIVE;
                default: next_state=FIFTEEN;
            endcase
    TWENTY: case(coin)
                N: next_state=TWENTYFIVE;
                D: next_state=TWENTYFIVE;
                Q: next_state=TWENTYFIVE;
                default: next_state=TWENTY;
            endcase
    TWENTYFIVE: next_state=ttnghi;
    default : next_state=ttnghi;
endcase
end
always @(clk)
```

```
    begin
        if(reset)
            begin state <= ttnghi;
                    vend <= 1'b0;
            end
        else state <= next_state;
    case (state)
            ttnghi: begin vend <= 1'b0;
                            CO <=3'd0;
                    end
            FIVE: begin vend <= 1'b0;
                            if (coin==Q) CO <=N;
                            else CO <=3'd0;
                    end
            TEN: begin vend <= 1'b0;
                        if (coin==Q) CO <=D;
                        else CO <= 3'd0;
                    end
            FIFTEEN : begin vend <= 1'b0;
                        if (coin==Q) CO <=N_D;
                        else CO <= 3'd0;
                    end
            TWENTY : begin vend <= 1'b0;
                            if (coin==D) CO <=N;
                            else if (coin==Q) CO <=D_D;
                                    else CO <= 3'd0;
                        end
            TWENTYFIVE : begin vend <= 1'b1;
                                    CO <=3'd0;
                            end
            default: state <= ttnghi;
    endcase
end
endmodule
```