**ĐẠI HỌC QUỐC QIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**

**KHOA ĐIỆN – ĐIỆN TỬ**

**A blue and white logo

Description automatically generated-----------------------------**

**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](#_Toc179220381)

[**II.** **Schamatic Module và các tín hiệu** 3](#_Toc179220382)

[**1.** **Sơ đồ khối tổng thể thực hiện chức năng** 3](#_Toc179220383)

[**2.** **Module Encode và mux** 4](#_Toc179220384)

[**3.** **Adder và Sub** 5](#_Toc179220385)

[**A.** **Adder** 5](#_Toc179220386)

[**B.** **Sub** 6](#_Toc179220387)

[**4.** **Bộ so sánh** 7](#_Toc179220388)

[**5.** **Register total\_reg và change\_reg** 9](#_Toc179220389)

[**A.** **Total\_reg** 10](#_Toc179220390)

[**B.** **Change\_reg** 11](#_Toc179220391)

[**6.** **Decoder** 11](#_Toc179220392)

[**7.** **7.FSM** 13](#_Toc179220393)

[**8.** **Sơ đồ khối tổng** 16](#_Toc179220394)

[**III.** **Testbench** 17](#_Toc179220395)

# **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.

A diagram of a process

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

# **Schamatic Module và các tín hiệu**

## **Sơ đồ khối tổng thể thực hiện chức năng**

A diagram of a vending machine

Description automatically generated

rst\_n



|  |  |
| --- | --- |
| Ngõ vào | Nickle, dime, quarter, clk, rst\_n |
| Ngõ ra | Soda, change |

A diagram of a computer program

Description automatically generated

Figure 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.

A diagram of a computer component

Description automatically generated

Figure : FSM

## **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

## **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

### **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.

A diagram of a number of objects

Description automatically generated

Figure Sơ đồ khối bộ Adder

### **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).

A diagram of a block diagram

Description automatically generatedA diagram of a circuit

Description automatically generated

Figure : Schemetic CLA\_Adder

Figure : Sơ đồ khối bộ Sub

## **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

A black and blue text on a white background

Description automatically generatedA computer screen shot of a computer program

Description automatically generatedendmodule

Figure : Ngõ vào ngõ ra của bộ so sánh

## **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.

A computer screen shot of a diagram

Description automatically generated module register #(parameter WIDTH = 6)

Figure : Schematic của Register

(

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

### **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)

);

A computer screen with text and arrows

Description automatically generated

Figure : Sơ đồ ngõ vào ngõ ra của total\_reg với các tín hiệu điều khiển được nối với FSM

### **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)

);

## **Decoder**

A number of numbers on a white background

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

A table with numbers and symbols

Description automatically generated

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.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

A computer screen shot of a computer program

Description automatically generated 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 : Sợ đồ io của FSM

A diagram of a diagram

Description automatically generated

Figure : Bảng chuyển trạng thái mô phỏng trên tool verdi

A screenshot of a computer

Description automatically generated

## **Sơ đồ khối tổng**

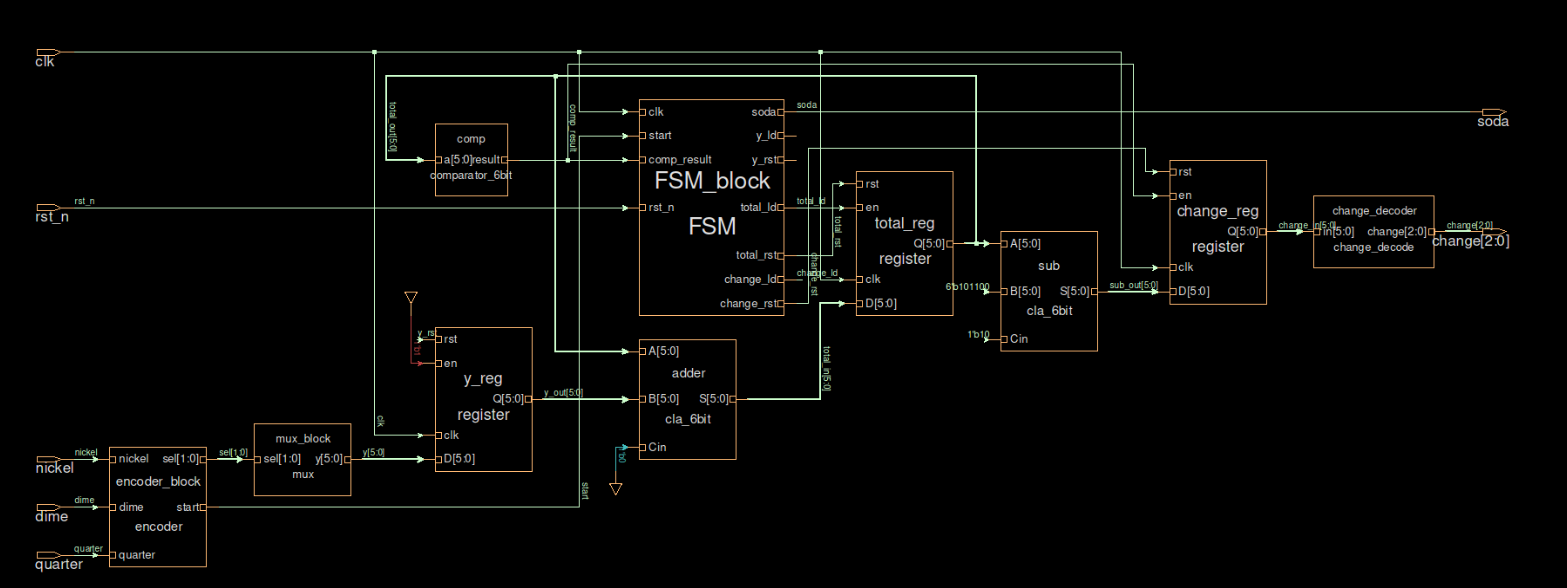
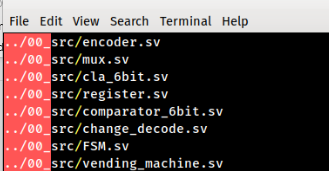
1. 

Figure : Sơ đổ khối tỏng

# **Testbench**

A computer screen with green and blue lines

Description automatically generated

Figure 13: wave form

A computer screen shot of a diagram

Description automatically generated

Figure 12: Sơ đồ khối top module

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