

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



Đồ án môn học Thiết kế Luận lý (CO3091)

VENDING MACHINE

Hệ thống máy bán hàng  
tự động

GVHD: Thầy Huỳnh Phúc Nghị

Thành viên	MSSV	Nhiệm vụ
Trương Anh Khôi	2211701	Làm báo cáo
Phạm Hà Bảo Long	2211893	Code
Lê Phú Cường	2210425	Lên ý tưởng và Code chính
Đoàn Tường Chánh Dao	2210644	Code
Đoàn Đức Bình	2210302	Làm báo cáo



## Mục lục

<b>1 Giới thiệu</b>	<b>2</b>
1.1 Máy bán hàng tự động (Vending Machine) . . . . .	2
1.2 Công cụ và thiết bị sử dụng . . . . .	2
<b>2 Tổng quan về sản phẩm</b>	<b>5</b>
2.1 Lý thuyết sơ lược về sản phẩm . . . . .	5
2.2 Tính năng được ứng dụng . . . . .	5
<b>3 Thiết kế hệ thống</b>	<b>7</b>
3.1 Sơ đồ khối . . . . .	7
3.1.1 Các khối chức năng . . . . .	7
3.1.2 Đặc tả các khối chức năng . . . . .	8
3.2 Luồng xử lý . . . . .	8
<b>4 Hiện thực (Verilog Code)</b>	<b>13</b>
4.1 Module Input . . . . .	13
4.2 Module Check . . . . .	13
4.3 Module Calculate . . . . .	13
4.4 Module Insert Money . . . . .	14
4.5 Module Insert Price . . . . .	14
4.6 Module Payment . . . . .	14
4.7 Module Vending Machine (Main) . . . . .	15
<b>5 Đánh giá tổng quan thiết kế</b>	<b>31</b>
<b>6 Kết luận</b>	<b>32</b>

## 1 Giới thiệu

### 1.1 Máy bán hàng tự động (Vending Machine)

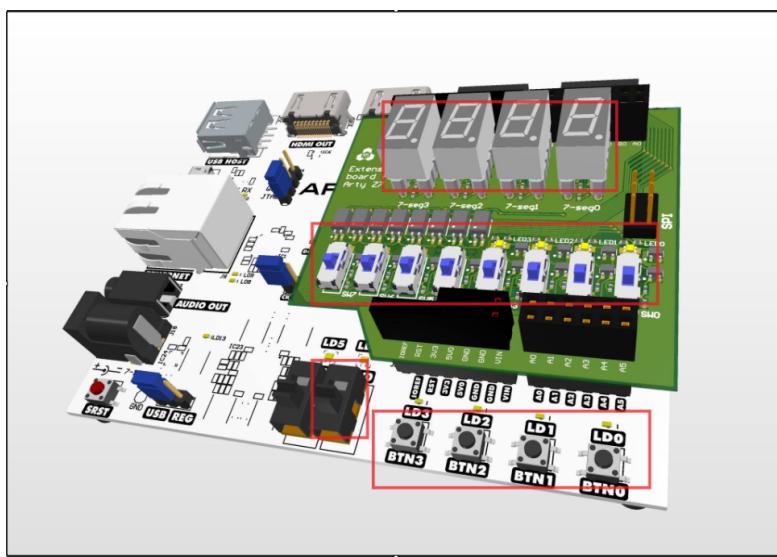
Bán hàng tự động là một hình thức bán hàng rất phổ biến ở các nước phát triển. Lợi ích của bán hàng tự động là tiết kiệm được chi phí trong kinh doanh khi không cần phải thuê nhân viên bán hàng cũng như tìm kiếm mặt bằng thuê cửa hàng. Ngoài ra, người mua hàng có thể được mua với giá tốt nhất. Nguyên tắc hoạt động cơ bản của máy bán hàng tự động là lưu trữ lượng sản phẩm với giá thành cho từng sản phẩm. Khi có người mua hàng, tùy thuộc vào sản phẩm được chọn, máy sẽ tính ra lượng tiền tương ứng cần thanh toán. Sau đó, người mua hàng phải trả tiền và nhận sản phẩm, đồng thời máy sẽ trả tiền thừa cho người mua hàng (nếu có) và kết thúc giao dịch.

#### Các chức năng cơ bản:

- Hiển thị thông tin sản phẩm: Tên sản phẩm, số lượng, giá thành.
- Lựa chọn mặt hàng: Cho phép người dùng lựa chọn mặt hàng và số lượng.
- Thanh toán: Tiến hành thanh toán với tiền của người dùng.
- Giao hàng: Hệ thống thực hiện giao hàng cho người dùng.
- Cập nhật kho hàng: Tự động cập nhật số lượng hàng hóa còn lại sau mỗi giao dịch.
- Cho phép người quản trị viên cập nhật kho hàng.

### 1.2 Công cụ và thiết bị sử dụng

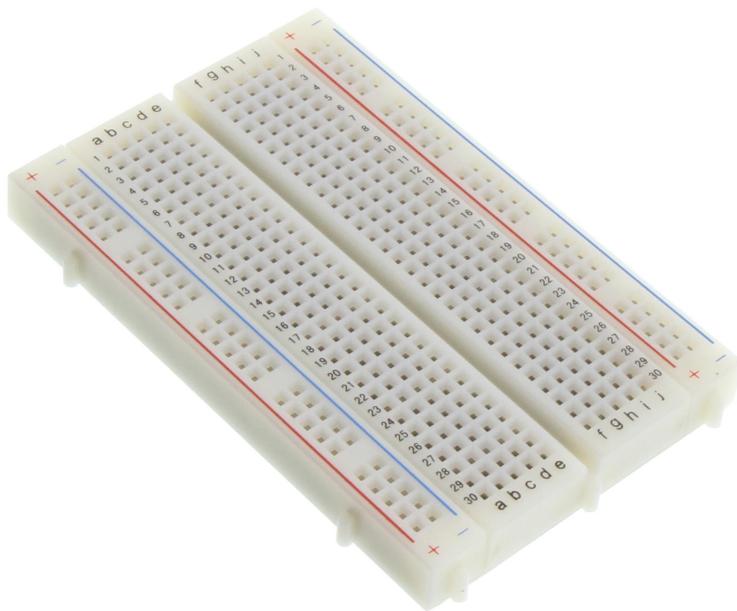
Dự án được thực hiện trên Phần mềm Vivado sử dụng ngôn ngữ Verilog và Board Arty Z7.





Hình 2: Màn hình LCD 16x2

Thiết bị sử dụng	Số lượng
Board Arty Z7	1
Board Arty Z7 extension	1
Button	4
Switchs (Board Arty Z7 extension)	8
Switchs (Board Arty Z7)	1
Màn hình LCD	1
Breadboard	1



Hình 3: Breadboard



## 2 Tổng quan về sản phẩm

### 2.1 Lý thuyết sơ lược về sản phẩm

Sản phẩm của nhóm xây dựng dựa trên những chức năng cơ bản của máy bán hàng tự động với đặc tả cụ thể như sau:

- **Hiển thị thông tin của sản phẩm:** Tên sản phẩm, số lượng và giá thành. Những thông tin này được hiển thị thông qua màn hình LCD. Người dùng có thể sử dụng Button để xem lần lượt qua các sản phẩm có trên hệ thống.
- **Chọn mặt hàng và số lượng:** Người dùng sử dụng các switchs để chọn mặt hàng và số lượng cần mua. Khi người dùng chọn mặt hàng hoặc số lượng bị nhầm người dùng có thể sử dụng các button để trả về nhập lại mặt hàng hoặc số lượng.
- **Hiển thị thông tin thanh toán:** Tổng số tiền người dùng cần thanh toán sẽ được hiển thị trên LCD. Người dùng cũng có thể sử dụng các Button để quay trở lại chọn số lượng hoặc mặt hàng nếu có nhu cầu.
- **Người dùng nhập tiền:** Người dùng có thể nhập số tiền của họ bằng các switchs. Người dùng có thể dùng Button để quay lại xem thông tin thanh toán nếu quên số tiền.
- **Thanh toán:** Hệ thống sẽ đổi chiếu số tiền của người dùng và số tiền cần thanh toán.
  - Nếu đủ: Tiến hành giao hàng.
  - Nếu không đủ: Hệ thống sẽ báo lỗi.
- **Giao hàng:** Hệ thống sẽ mô phỏng giao hàng thông qua các LED đơn và trả lại tiền thừa nếu có thông qua LCD.
- **Cập nhật kho hàng:** Hệ thống sẽ cập nhật kho hàng sau mỗi lần giao dịch thành công.
- **Quản trị viên:** Hệ thống cho phép người quản trị có thể cập nhật lại kho hàng thông qua các Switchs và Button như khi mua hàng.
- **Thông báo cho người dùng:** Hệ thống sử dụng các LED đơn và LCD để thông báo cho người dùng biết đang ở giai đoạn nào.

### 2.2 Tính năng được ứng dụng

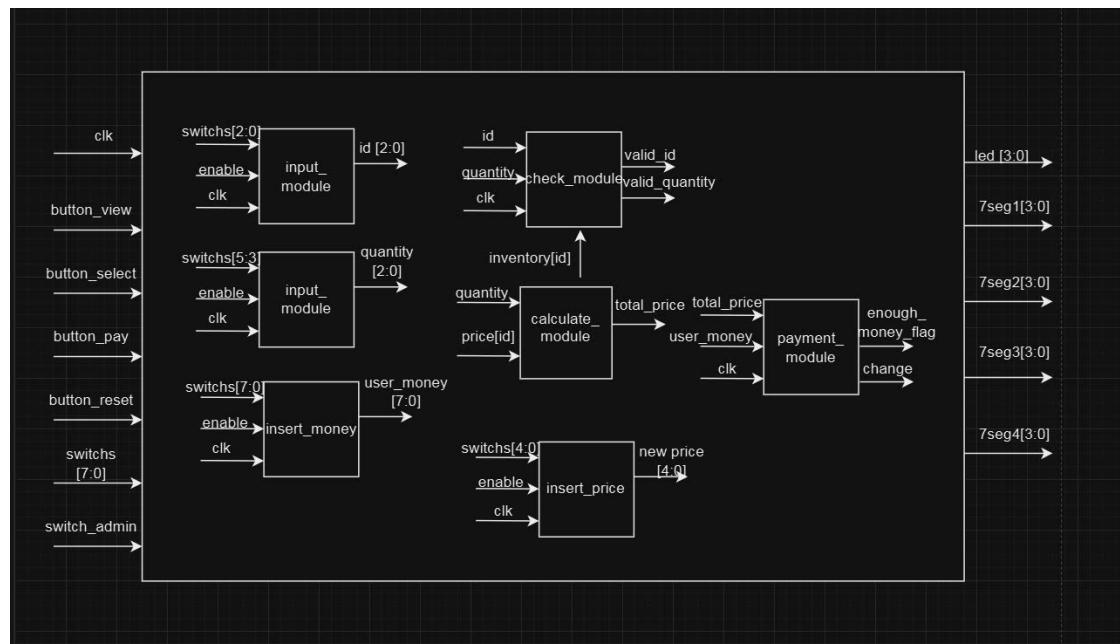
- Hiển thị thông tin sản phẩm
  - Hiển thị các sản phẩm hiện có, giá thành và số lượng.
- Quy trình giao dịch tự động
  - Hệ thống nhận yêu cầu từ người dùng (chọn sản phẩm và số lượng).
  - Tính toán tổng chi phí mà người dùng cần thanh toán.
  - Xác nhận thanh toán, giao hàng và trả tiền thừa nếu cần.
- Cập nhật trạng thái kho hàng
  - Tự động điều chỉnh số lượng hàng hóa trong kho sau mỗi lần giao dịch.
  - Thông báo khi sản phẩm được chọn hết hàng.



- Giao diện đơn giản và trực quan
  - Sử dụng LED, Switch, Button để mô phỏng giao dịch.
  - Hiệu ứng LED và màn hình LCD báo hiệu hoàn thành giao dịch.

### 3 Thiết kế hệ thống

#### 3.1 Sơ đồ khái



Hình 4: Vending machine module

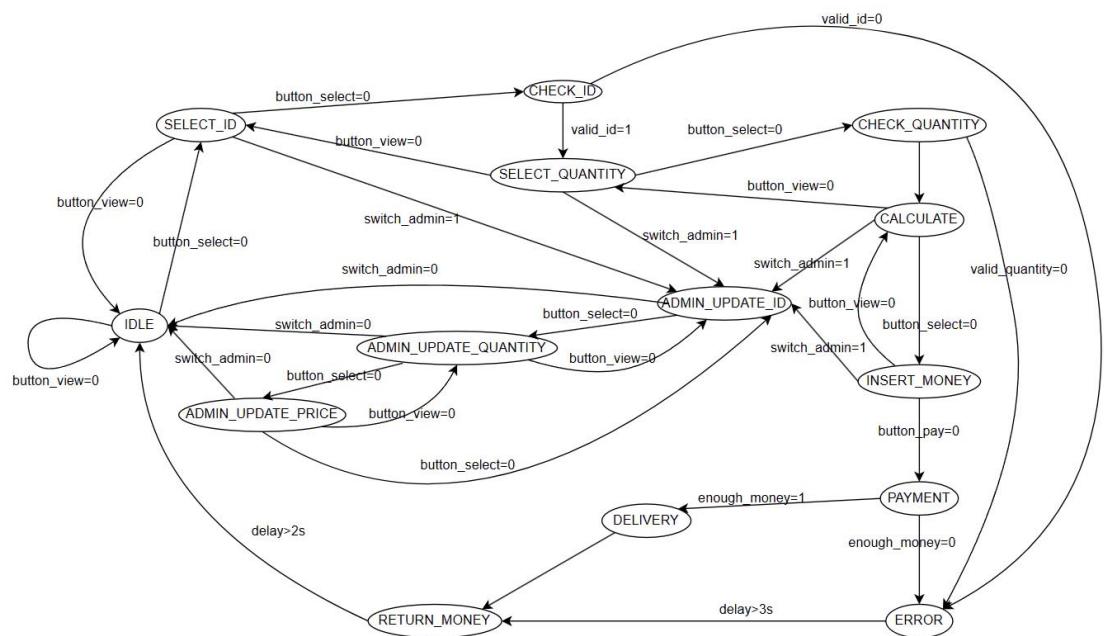
##### 3.1.1 Các khái chức năng

- Input Module** - Được sử dụng để người dùng nhập ID và số lượng cần mua hoặc để người quản trị nhập ID và số lượng mới cần cập nhật.
- Insert Money Module** - Được sử dụng để người nhập số tiền vào.
- Insert Price Module** - Được sử dụng để người quản trị nhập giá tiền mới cho sản phẩm cần cập nhật.
- Check Module** - Khối kiểm tra ID và số lượng của sản phẩm mà người dùng nhập vào có hợp lệ hay không.
- Calculate Module** - Khối tính toán số tiền mà người dùng cần thanh toán.
- Payment Module** - Khối xử lý thanh toán.
- Main Module** - Quản lý tất cả đầu vào và đầu ra của hệ thống.

### 3.1.2 Đặc tả các khối chức năng

Module	Input	Output	Chức năng
Input	switch[7:0]	id[2:0], quantity[2:0]	Nhập id cần mua hoặc chỉnh sửa từ 3 bit đầu của switch[7:0], tương tự với số lượng là 3 bit tiếp theo của switch[7:0]
Insert Money	switch[7:0]	user_money[7:0]	Nhập tiền từ switch[7:0]
Insert Price	switch[4:0]	new_price[4:0]	Nhập giá cho sản phẩm cần chỉnh sửa giá tiền từ switch[4:0]
Check	id, quantity, inventory[id]	valid_id, valid_quantity	Kiểm tra tính hợp lệ của id và quantity có tồn tại hay không
Calculate	quantity, price[id]	total_price	Tính tổng số tiền cần trả dựa trên giá của sản phẩm và số lượng sản phẩm người dùng chọn
Payment	total_price, user_money	change, enough_money_flag	Thực hiện thanh toán từ tiền của người dùng và giá của sản phẩm. Trả tiền thừa nếu có

## 3.2 Luồng xử lý



Hình 5: Sơ đồ chuyển trạng thái



### ĐẶC TẢ MÁY TRẠNG THÁI:

- **Ở trạng thái IDLE:** Ở trạng thái này, người dùng có thể xem lần lượt các sản phẩm hiện có.
  - **Led đơn 3, 2, 1, 0:** off, off, off, off.
  - **Màn hình LCD:** Dùng để hiển thị lần lượt các sản phẩm hiện có.
  - **Button 3 (view):** Dùng để hiển thị lần lượt các sản phẩm hiện có.
  - **Button 2 (select):** Dùng để chuyển trạng thái sang chọn ID của sản phẩm cần mua.
  - **Button 1 (pay):** Không làm gì cả.
  - **Button 0 (reset):** Trở về trạng thái IDLE.
  - **Các switchs:** Không làm gì cả.
  - **Switch Admin:** Chuyển trạng thái sang ADMIN UPDATE ID.
- **Ở trạng thái SELECT ID:** Ở trạng thái này, người dùng sẽ nhập ID của sản phẩm cần mua thông qua các switchs.
  - **Led đơn 3, 2, 1, 0:** off, off, off, on.
  - **Màn hình LCD:** Dùng để hiển thị sản phẩm hiện được chọn.
  - **Button 3 (view):** Trở về trạng thái IDLE.
  - **Button 2 (select):** Dùng để xác nhận ID của sản phẩm cần mua và chuyển sang trạng thái CHECK ID.
  - **Button 1 (pay):** Không làm gì cả.
  - **Button 0 (reset):** Trở về trạng thái IDLE.
  - **Switch 0, 1 và 2:** Dùng để nhập ID của sản phẩm.
  - **Các switchs còn lại:** Không làm gì cả.
  - **Switch Admin:** Chuyển trạng thái sang ADMIN UPDATE ID.
- **Ở trạng thái CHECK ID:** Đây là trạng thái trung gian nên sẽ kiểm tra ID có hợp lệ hay không.
  - **Nếu hợp lệ:** Chuyển sang trạng thái SELECT QUANTITY.
  - **Nếu không hợp lệ:** Chuyển sang trạng thái ERROR.
- **Ở trạng thái SELECT QUANTITY:** Ở trạng thái này, người dùng sẽ nhập số lượng cần mua thông qua các switchs,
  - **Led đơn 3, 2, 1, 0:** off, off, on, on.
  - **Màn hình LCD:** Dùng để hiển thị số lượng sản phẩm hiện được chọn.
  - **Button 3 (view):** Trở về trạng thái SELECT ID.
  - **Button 2 (select):** Dùng để xác nhận số lượng của sản phẩm cần mua và chuyển sang trạng thái CHECK QUANTITY.
  - **Button 1 (pay):** Không làm gì cả.
  - **Button 0 (reset):** Trở về trạng thái IDLE.



- **Switch 0, 1 và 2:** Dùng để nhập số lượng của sản phẩm cần mua.
  - **Các switchs còn lại:** Không làm gì cả.
  - **Switch Admin:** Chuyển trạng thái sang ADMIN UPDATE ID.
- **Ở trạng thái CHECK QUANTITY:** Đây là trạng thái trung gian nên sẽ kiểm tra ID có hợp lệ hay không.
    - **Nếu hợp lệ:** Chuyển sang trạng thái CALCULATE.
    - **Nếu không hợp lệ:** Chuyển sang trạng thái ERROR.
  - **Ở trạng thái CALCULATE:** Trạng thái này sẽ hiển thị tổng số tiền người dùng cần thanh toán trên LCD.
    - **Led đơn 3, 2, 1, 0:** off, off, on, on.
    - **Màn hình LCD:** Dùng để hiển thị tổng số tiền cần thanh toán.
    - **Button 3 (view):** Trở về trạng thái SELECT QUANTITY.
    - **Button 2 (select):** Dùng để chuyển sang trạng thái INSERT MONEY.
    - **Button 1 (pay):** Không làm gì cả.
    - **Button 0 (reset):** Trở về trạng thái IDLE.
    - **Các switchs:** Không làm gì cả.
  - **Ở trạng thái INSERT MONEY:** Ở trạng thái này, người dùng nhập số lượng của mình vào thông qua các switchs.
    - **Led đơn 3, 2, 1, 0:** off, on, on, on.
    - **Màn hình LCD:** Dùng để hiển thị số tiền mà người dùng nhập vào.
    - **Button 3 (view):** Trở về trạng thái CALCULATE.
    - **Button 2 (select):** Dùng để xác nhận số lượng của sản phẩm cần mua và chuyển sang trạng thái CHECK QUANTITY.
    - **Button 1 (pay):** Chuyển sang trạng thái PAYMENT.
    - **Button 0 (reset):** Trở về trạng thái IDLE.
    - **Các switchs từ 0 - 7:** Dùng để người dùng nhập tiền vào.
    - **Switch Admin:** Chuyển trạng thái sang ADMIN UPDATE ID.
  - **Ở trạng thái PAYMENT:** Đây là trạng thái trung gian nên sẽ kiểm tra số tiền người dùng nhập vào đủ hay không.
    - **Nếu đủ:** Hệ thống sẽ chuyển sang trạng thái DELIVERY để giao hàng.
    - **Nếu không đủ:** Hiển thị sẽ chuyển sang trạng thái ERROR.
  - **Ở trạng thái DELIVERY:** Ở trạng thái này, sẽ hiển thị hiệu ứng giao hàng thông qua các Led đơn và LCD.
    - **Led đơn 3, 2, 1, 0:** on, on, on, on.
    - **Màn hình LCD:** Dùng để hiển thị trạng thái giao hàng.
    - **Sau 3 giây:** Chuyển sang trạng thái RETURN MONEY.



- **Ở trạng thái ERROR:** Đây là trạng thái lỗi, thông báo cho người dùng thông qua Led 7 đoạn.
  - **Led 7 đoạn 1, 2, 3, 4:** Hiển thị số 9.
  - **Màn hình LCD:** Dùng để hiển thị trạng thái lỗi.
  - **Sau 3 giây:** Chuyển sang trạng thái RETURN MONEY.
- **Ở trạng thái RETURN MONEY:** Ở trạng thái, số tiền thừa của người dùng sau khi thanh toán sẽ hiển thị trên Led 7 đoạn.
  - **Màn hình LCD:** Dùng để hiển thị tiền thừa của người dùng.
  - **Sau 2 giây:** Chuyển sang trạng thái IDLE.
- **Ở trạng thái ADMIN UPDATE ID:** Đây là trạng thái cho quản trị viên nhập ID của sản phẩm cần chỉnh sửa thông qua các switchs.
  - **Led đơn 3, 2, 1, 0:** 4 đèn sáng lên cùng lúc rồi trở về off, off, off, on.
  - **Màn hình LCD:** Dùng để hiển thị ID sản phẩm cần chỉnh sửa.
  - **Button 3 (view):** Không làm gì cả.
  - **Button 2 (select):** Dùng để xác nhập ID cần chỉnh sửa và chuyển sang trạng thái ADMIN UPDATE QUANTITY.
  - **Button 1 (pay):** Không làm gì cả.
  - **Button 0 (reset):** Trở về trạng thái IDLE.
  - **Switch 0, 1 và 2:** Dùng để nhập ID của sản phẩm cần chỉnh sửa.
  - **Các switchs còn lại:** Không làm gì cả.
  - **Switch Admin:** Chuyển trạng thái sang IDLE.
- **Ở trạng thái ADMIN UPDATE QUANTITY:** Đây là trạng thái cho quản trị viên nhập số lượng của sản phẩm cần chỉnh sửa thông qua các switchs.
  - **Led đơn 3, 2, 1, 0:** off, off, on, on.
  - **Màn hình LCD:** Dùng để hiển thị số lượng mới của sản phẩm cần chỉnh sửa.
  - **Button 3 (view):** Chuyển về trạng thái ADMIN UPDATE ID.
  - **Button 2 (select):** Dùng để xác nhập số lượng mới và chuyển sang trạng thái ADMIN UPDATE PRICE.
  - **Button 1 (pay):** Không làm gì cả.
  - **Button 0 (reset):** Trở về trạng thái IDLE.
  - **Switch 0, 1 và 2:** Dùng để nhập số lượng mới của sản phẩm.
  - **Các switchs còn lại:** Không làm gì cả.
  - **Switch Admin:** Chuyển trạng thái sang IDLE.
- **Ở trạng thái ADMIN UPDATE PRICE:** Đây là trạng thái cho quản trị viên nhập giá mới của sản phẩm cần chỉnh sửa thông qua các switchs.
  - **Led đơn 3, 2, 1, 0:** off, on, on, on.



- **Màn hình LCD:** Dùng để hiển thị giá mới của sản phẩm.
- **Button 3 (view):** Chuyển về trạng thái ADMIN UPDATE QUANTITY.
- **Button 2 (select):** Dùng để xác nhập giá mới và chuyển sang trạng thái ADMIN UPDATE ID.
- **Button 1 (pay):** Không làm gì cả.
- **Button 0 (reset):** Trở về trạng thái IDLE.
- **Switch 0, 1, 2 và 3:** Dùng để nhập giá mới của của sản phẩm.
- **Các switchs còn lại:** Không làm gì cả.
- **Switch Admin:** Chuyển trạng thái sang IDLE.



## 4 Hiện thực (Verilog Code)

### 4.1 Module Input

```
1 module input_modulless2(
2     input clk,
3     input [2:0] switchs,
4     input enable,
5     output reg [2:0] id_or_quantity
6 );
7 always @(posedge clk) begin
8     if(enable) begin
9         id_or_quantity <= switchs;
10    end
11 end
12 endmodule
```

### 4.2 Module Check

```
1 module check_modulless2(
2     input clk,
3     input [2:0] selected_id,
4     input [2:0] selected_quantity,
5     input [2:0] inventory_id,
6     output reg valid_id,
7     output reg valid_quantity
8 );
9 always@ (posedge clk) begin
10     if(selected_id >= 3'b000 && selected_id <= 3'b111) begin
11         valid_id <= 1;
12     end else begin
13         valid_id <= 0;
14     end
15     if(selected_quantity >= 1 && selected_quantity <= inventory_id) begin
16         valid_quantity <= 1;
17     end else begin
18         valid_quantity <= 0;
19     end
20 end
21 endmodule
```

### 4.3 Module Calculate

```
1 module calculate_modulless2(
2     input [2:0] selected_quantity,
3     input [4:0] price_id,
4     output reg [7:0] total_price
5 );
6 always@(*) begin
7     total_price = price_id*selected_quantity;
```



```
8     end
9 endmodule
```

#### 4.4 Module Insert Money

```
1 module insert_money_modulless2(
2     input clk,
3     input [7:0] switchs,
4     input enable,
5     output reg [7:0] user_money
6 );
7
8     always@(posedge clk)begin
9         if(enable) begin
10             user_money <= switchs;
11         end
12     end
13 endmodule
```

#### 4.5 Module Insert Price

```
1 module insert_price(
2     input clk,
3     input [4:0] switchs,
4     input enable,
5     output reg [4:0] new_price
6 );
7
8     always@(posedge clk)begin
9         if(enable) begin
10             new_price <= switchs;
11         end
12     end
13 endmodule
```

#### 4.6 Module Payment

```
1 module payment_modulless2(
2     input clk,
3     input [7:0] total_price,
4     input [7:0] total_money,
5     output reg enough_money_flag,
6     output reg [7:0] remaining_money
7 );
8     always @ (posedge clk) begin
9         enough_money_flag = 0;
10        if(total_money >= total_price) begin
11            enough_money_flag = 1;
12            remaining_money = total_money - total_price;
```



```
13     end
14 else begin
15     enough_money_flag = 0;
16     remaining_money = total_money;
17 end
18 end
19 endmodule
```

## 4.7 Module Vending Machine (Main)

```
1 timescale 1ns / 1ps
2 module vending_machiness2(
3     input clk,
4     input button_pay,
5     input button_view,
6     input button_select,
7     input button_reset,
8     input switch_admin,
9     input [7:0] switchs,
10    output reg [3:0] seven_segment1,
11    output reg [3:0] seven_segment2,
12    output reg [3:0] seven_segment3,
13    output reg [3:0] seven_segment4,
14    output reg [3:0] led,
15    output reg [7:0] lcd_data, // Data to be sent to LCD
16    output reg rs,
17    output reg en,
18    output rw // Control signals for LCD
19 );
20
21 reg [2:0] selected_id;
22 reg [2:0] selected_quantity;
23 wire [2:0] id;
24 wire [2:0] quantity;
25
26 wire [2:0] admin_id;
27 wire [2:0] admin_quantity;
28 reg [2:0] id_for_update;
29 reg [2:0] new_quantity;
30 wire [4:0] admin_price;
31 reg [4:0] new_price;
32
33
34 wire valid_id;
35 wire valid_quantity;
36
37 wire [7:0] sum_price;
38 reg [7:0] total_price;
39
40 wire [7:0] user_money;
41 reg [7:0] total_money;
```



```
43
44     wire enough_money_flag;
45
46     wire [7:0] remaining_money;
47     reg [7:0] change;
48
49     reg [2:0] product_inventory [0:7];
50     reg [4:0] product_price [0:7];
51     reg [13:0] state;
52     reg [2:0] current_product;
53     reg [31:0] delay_counter;
54
55
56     reg [18:0] delay_lcd;
57     reg [7:0] data_rom[37:0];
58     integer data_pos;
59
60     localparam IDLE = 14'b0100000000000000;
61     localparam SELECT_ID = 14'b0000000000000001;
62     localparam SELECT_QUANTITY = 14'b0000000000000010;
63     localparam CHECK_ID = 14'b0010000000000000;
64     localparam CHECK_QUANTITY = 14'b0001000000000000;
65     localparam CALCULATE = 14'b0000100000000000;
66     localparam PAYMENT = 14'b00000000000100;
67     localparam DELIVERY = 14'b0000010000000000;
68     localparam ERROR = 14'b0000001000000000;
69     localparam ADMIN_UPDATE_ID = 14'b00000000001000;
70     localparam ADMIN_UPDATE_QUANTITY = 14'b000000000010000;
71     localparam ADMIN_UPDATE_PRICE = 14'b00000000100000;
72     localparam INSERT_MONEY = 14'b00000001000000;
73     localparam RETURN_MONEY = 14'b10000001000000;
74
75
76     initial begin
77         product_inventory[3'b000] = 3'd5;
78         product_inventory[3'b001] = 3'd5;
79         product_inventory[3'b010] = 3'd5;
80         product_inventory[3'b011] = 3'd5;
81         product_inventory[3'b100] = 3'd5;
82         product_inventory[3'b101] = 3'd5;
83         product_inventory[3'b110] = 3'd5;
84         product_inventory[3'b111] = 3'd5;
85
86         product_price[3'b000] = 5'd5;
87         product_price[3'b001] = 5'd10;
88         product_price[3'b010] = 5'd15;
89         product_price[3'b011] = 5'd20;
90         product_price[3'b100] = 5'd25;
91         product_price[3'b101] = 5'd30;
92         product_price[3'b110] = 5'd30;
93         product_price[3'b111] = 5'd30;
94
95         current_product = 3'b000;
```



```
96      state = IDLE;
97      delay_counter = 0;
98      change = 8'd0;
99      selected_id = 3'b000;
100     selected_quantity = 3'b000;
101     new_price = 5'b00000;
102     id_for_update = 3'b000;
103     new_quantity = 3'b000;
104     total_money = 8'b00000000;
105     total_price = 8'b00000000;
106     change = 8'b00000000;
107   end
108
109   input_modulless2 input_handle_id(.clk(clk), .switchs(switchs[2:0]),
110   .enable(state[0]),
111   .id_or_quantity(id)
112 );
113
114   input_modulless2 input_handle_quantity(.clk(clk), .switchs(switchs
115   [5:3]),
116   .enable(state[1]),
117   .id_or_quantity(quantity)
118 );
119
120   input_modulless2 input_handle_admin_update_id(.clk(clk), .switchs(
121   switchs[2:0]),
122   .enable(state[3]),
123   .id_or_quantity(admin_id)
124 );
125
126   input_modulless2 input_handle_admin_update_quantity(.clk(clk), .
127   switchs(switchs[5:3]),
128   .enable(state[4]),
129   .id_or_quantity(admin_quantity)
130 );
131
132
133   check_modulless2 check(.clk(clk), .selected_id(selected_id),
134   .selected_quantity(selected_quantity),
135   .inventory_id(product_inventory[selected_id]),
136   .valid_id(valid_id),
137   .valid_quantity(valid_quantity));
138
139   calculate_modulless2 calculate(.selected_quantity(selected_quantity),
140   .price_id(product_price[selected_id]),
141   .total_price(sum_price));
142
143   insert_money_modulless2 insert_money(.clk(clk), .switchs(switchs),
144   .enable(state[6]),
145   .user_money(user_money)
146 );
```



```
146     insert_price insert_price_admin(.clk(clk), .switchs(switchs[4:0]), .
147         enable(state[5]),
148         .new_price(admin_price)
149     );
150
150     payment_modulless2 payment(.clk(clk), .total_price(total_price),
151         .total_money(total_money),
152         .enough_money_flag(enough_money_flag),
153         .remaining_money(remaining_money)
154     );
155
156
157     reg button_view_prev;
158     reg button_select_prev;
159     reg button_pay_prev;
160     reg button_reset_prev;
161
162
163     assign rw=1'b0;
164
165     always @(posedge clk) begin
166         delay_lcd<=delay_lcd+1;
167         if (delay_lcd<50000) begin
168
169             en<=1'b1;
170             lcd_data<=data_rom[data_pos];
171         end
172         else begin
173             if (delay_lcd>=50000 && delay_lcd<250000) begin
174                 delay_lcd<=delay_lcd+1;
175                 en<=1'b0;
176             end
177             else if (delay_lcd==250000) begin
178                 delay_lcd<=0;
179                 data_pos<=data_pos+1;
180             end
181         end
182         if (data_pos<5) begin
183             rs<=1'b0;
184         end
185         else if (data_pos>=5 && data_pos<21) begin
186             rs<=1'b1;
187         end
188         else if (data_pos==21) begin
189             rs<=1'b0;
190         end
191         else if (data_pos>21 && data_pos <38) begin
192             rs<=1'b1;
193         end
194         else if (data_pos>=38) begin
195             data_pos<=4;
196         end
197     end
198 
```



```
198
199
200    always @(posedge clk) begin
201        button_view_prev <= button_view;
202        button_select_prev <= button_select;
203        button_pay_prev <= button_pay;
204        button_reset_prev <= button_reset;
205        case(state)
206            IDLE: begin
207                led <= 4'b0000;
208                // seven_segment1 <= seven_segment_encode(
209                //    current_product);
210                // seven_segment2 <= seven_segment_encode(
211                //    product_inventory[current_product]);
212                // seven_segment3 <= seven_segment_encode(
213                //    product_price[current_product] / 10);
214                // seven_segment4 <= seven_segment_encode(
215                //    product_price[current_product] % 10);
216                delay_counter <= 0;
217                data_rom[0] <= 8'h38;
218                data_rom[1] <= 8'h0c;
219                data_rom[2] <= 8'h06;
220                data_rom[3] <= 8'h01;
221                data_rom[4] <= 8'h80;
222                data_rom[5] <= 8'h49; //I
223                data_rom[6] <= 8'h44; //D
224                data_rom[7] <= 8'h3a; //:
225                data_rom[8] <= 8'h20;
226                data_rom[9] <= 8'h30+current_product;
227                data_rom[10] <= 8'h20;
228                data_rom[11] <= 8'h20;
229                data_rom[12] <= 8'h20;
230                data_rom[13] <= "S";
231                data_rom[14] <= "L";
232                data_rom[15] <= ":";
233                data_rom[16] <= 8'h20;
234                data_rom[17] <= 8'h30+product_inventory[current_product];
235            ];
236                data_rom[18] <= 8'h20;
237                data_rom[19] <= 8'h20;
238                data_rom[20] <= 8'h20;
239                data_rom[21] <= 8'hc0;
240                data_rom[31] <= 8'h20;
241                data_rom[22] <= "P";
242                data_rom[23] <= "R";
243                data_rom[24] <= "I";
244                data_rom[25] <= "C";
245                data_rom[26] <= "E";
246                data_rom[27] <= ":";
247                data_rom[28] <= " ";
248                data_rom[29] <= (product_price[current_product] / 4'd10
249                    )+8'h30;
```



```
244         data_rom[30]<=( product_price[current_product] % 4'>
245             d10)+8'h30;
246         data_rom[32]<=" ";
247         data_rom[33]<=" ";
248         data_rom[34]<=" ";
249         data_rom[35]<=" ";
250         data_rom[36]<=" ";
251         data_rom[37]<=" ";
252     //    data_rom[38]<=" ";
253     //    seven_segment1 <= current_product;
254     //    seven_segment2 <= product_inventory[current_product];
255     //    seven_segment3 <= product_price[current_product] /
256     //        4'd10;
257     //    seven_segment4 <= product_price[current_product] %
258     //        4'd10;
259
260     if (button_view_prev && !button_view) begin
261         current_product <= current_product + 1;
262         if (current_product == 3'b111) begin
263             current_product <= 3'b000;
264         end
265     end
266
267     if (button_select_prev && !button_select) begin
268         state <= SELECT_ID;
269     end
270
271     if(switch_admin) begin
272         state <= ADMIN_UPDATE_ID;
273     end
274     if(button_reset_prev && !button_reset) begin
275         state <= IDLE;
276         current_product <= 3'b000;
277     end
278
279     SELECT_ID: begin
280         seven_segment1 <= seven_segment_encode(4'd0);
281         seven_segment2 <= seven_segment_encode(4'd0);
282         seven_segment3 <= seven_segment_encode(4'd0);
283         seven_segment4 <= seven_segment_encode(selected_id);
284     ;
285         led <= 4'b0001;
286         selected_id <= id;
287     //         seven_segment1 <= 4'd0;
288     //         seven_segment2 <= 4'd0;
289     //         seven_segment3 <= 4'd0;
290     //         seven_segment4 <= selected_id;
291         data_rom[5]<=8'h20;//I
292         data_rom[6]<="S";//D
293         data_rom[7]<="E";//:
294         data_rom[8]<="L";
```



```
292         data_rom[9] <= "E";
293         data_rom[10] <= "C";
294         data_rom[11] <= "T";
295         data_rom[12] <= " ";
296         data_rom[13] <= "I";
297         data_rom[14] <= "D";
298         data_rom[15] <= ":";
299         data_rom[16] <= 8'h20;
300         data_rom[17] <= 8'h30+selected_id;
301         data_rom[18] <= 8'h20;
302         data_rom[19] <= 8'h20;
303         data_rom[20] <= 8'h20;
304         data_rom[21] <= 8'hc0;
305         data_rom[22] <= 8'h20;
306         data_rom[23] <= 8'h20;
307         data_rom[24] <= 8'h20;
308         data_rom[25] <= 8'h20;
309         data_rom[26] <= 8'h20;
310         data_rom[27] <= 8'h20;
311         data_rom[28] <= 8'h20;
312         data_rom[29] <= 8'h20;
313         data_rom[30] <= 8'h20;
314         data_rom[31] <= 8'h20;
315         data_rom[32] <= " ";
316         data_rom[33] <= " ";
317         data_rom[34] <= " ";
318         data_rom[35] <= " ";
319         data_rom[36] <= " ";
320         data_rom[37] <= " ";
321         if (button_select_prev && !button_select) begin
322             state <= CHECK_ID;
323         end
324
325         if(switch_admin) begin
326             state <= ADMIN_UPDATE_ID;
327         end
328         if (button_view_prev && !button_view) begin
329             state <= IDLE;
330         end
331         if(button_reset_prev && !button_reset) begin
332             state <= IDLE;
333             current_product <= 3'b000;
334         end
335     end
336
337     SELECT_QUANTITY: begin
338 //      seven_segment1 <= seven_segment_encode(4'd0);
339 //      seven_segment2 <= seven_segment_encode(4'd0);
340 //      seven_segment3 <= seven_segment_encode(4'd0);
341 //      seven_segment4 <= seven_segment_encode(
342         selected_quantity);
343         led <= 4'b0011;
            selected_quantity <= quantity;
```



```
344 //  
345 //    seven_segment1 <= selected_id;  
346 //    seven_segment2 <= 4'd0;  
347 //    seven_segment3 <= 4'd0;  
348 //    seven_segment4 <= selected_quantity;  
349     data_rom[5]<=8'h20;//I  
350     data_rom[6]<="Q";//D  
351     data_rom[7]<="U";//:  
352     data_rom[8]<="A";  
353     data_rom[9]<="N";  
354     data_rom[10]<="T";  
355     data_rom[11]<="I";  
356     data_rom[12]<="T";  
357     data_rom[13]<="Y";  
358     data_rom[14]<=":";  
359     data_rom[15]<=" ";  
360     data_rom[16]<=8'h30+selected_quantity;  
361     data_rom[17]<=8'h20;  
362     data_rom[18]<=8'h20;  
363     data_rom[19]<=8'h20;  
364     data_rom[20]<=8'h20;  
365 if (button_select_prev && !button_select) begin  
366     state <= CHECK_QUANTITY;  
367 end  
368  
369 if(switch_admin) begin  
370     state <= ADMIN_UPDATE_ID;  
371 end  
372  
373 if(button_reset_prev && !button_reset) begin  
374     state <= IDLE;  
375     current_product <= 3'b000;  
376 end  
377 if (button_view_prev && !button_view) begin  
378     state <= SELECT_ID;  
379 end  
380 end  
381  
382 CHECK_ID: begin  
383 //    seven_segment1 <= 4'd0;  
384 //    seven_segment2 <= 4'd0;  
385 //    seven_segment3 <= 4'd0;  
386 //    seven_segment4 <= selected_id;  
387 if (valid_id) begin  
388     state <= SELECT_QUANTITY;  
389 end else begin  
390     state <= ERROR;  
391 end  
392  
393 end  
394 CHECK_QUANTITY: begin  
395 //    seven_segment1 <= selected_id;  
396 //    seven_segment2 <= 4'd0;
```



```
397 //    seven_segment3 <= 4'd0;
398 //    seven_segment4 <= selected_quantity;
399 if(valid_quantity) begin
400     state <= CALCULATE;
401 end else begin
402     state <= ERROR;
403 end
404
405 end
406 CALCULATE: begin
407 //    seven_segment1 <= seven_segment_encode(4'd0);
408 //    seven_segment2 <= seven_segment_encode(4'd0);
409 //    seven_segment3 <= seven_segment_encode(total_price
410 //        / 10);
411 //    seven_segment4 <= seven_segment_encode(total_price
412 //        % 10);
413 //    total_price <= sum_price;
414 //    seven_segment1 <= 4'd0;
415 //    seven_segment2 <= total_price / 100;
416 //    seven_segment3 <= (total_price % 100) / 10;
417 //    seven_segment4 <= total_price % 10;
418
419 data_rom[5]<="T"; //D
420 data_rom[6]<="0"; //:
421 data_rom[7]<="T";
422 data_rom[8]<="A";
423 data_rom[9]<="L";
424 data_rom[10]<=" ";
425 data_rom[11]<="P";
426 data_rom[12]<="R";
427 data_rom[13]<="I";
428 data_rom[14]<="C";
429 data_rom[15]<="E";
430 data_rom[16]<=":";
431 data_rom[17]<=8'h20;
432 data_rom[18]<=8'h30+total_price / 100;
433 data_rom[19]<=8'h30+(total_price % 100) / 10;
434 data_rom[20]<=8'h30+total_price % 10;
435
436 if(button_select_prev && !button_select) begin
437     state <= INSERT_MONEY;
438 end
439
440 if(switch_admin) begin
441     state <= ADMIN_UPDATE_ID;
442 end
443 if(button_reset_prev && !button_reset) begin
444     state <= IDLE;
445     current_product <= 3'b000;
446 end
447 if(button_view_prev && !button_view) begin
448     state <= SELECT_QUANTITY;
449 end
```



```
448
449     end
450 //    INSERT_MONEY: begin
451 //        seven_segment1 <= seven_segment_encode(4'd0);
452 //        seven_segment2 <= seven_segment_encode(4'd0);
453 //        seven_segment3 <= seven_segment_encode(total_money
454 //            / 10);
455 //        seven_segment4 <= seven_segment_encode(total_money
456 //            % 10);
457 //        led <= 4'b0111;
458 //        total_money <= user_money;
459 //        change <= remaining_money;
460 //        seven_segment1 <= 4'd0;
461 //        seven_segment2 <= total_money / 100;
462 //        seven_segment3 <= (total_money % 100) / 10;
463 //        seven_segment4 <= total_money % 10;
464 //        data_rom[5]<="E"; //D
465 //        data_rom[6]<="N"; //:
466 //        data_rom[7]<="T";
467 //        data_rom[8]<="E";
468 //        data_rom[9]<="R";
469 //        data_rom[10]<=" ";
470 //        data_rom[11]<="M";
471 //        data_rom[12]<="O";
472 //        data_rom[13]<="N";
473 //        data_rom[14]<="E";
474 //        data_rom[15]<="Y";
475 //        data_rom[16]<=":";
476 //        data_rom[17]<=8'h20;
477 //        data_rom[18]<=8'h30+total_money / 100;
478 //        data_rom[19]<=8'h30+(total_money % 100) / 10;
479 //        data_rom[20]<=8'h30+ total_money % 10;
480 if(button_pay_prev && !button_pay) begin
481     state <= PAYMENT;
482 end
483
484 if(switch_admin) begin
485     state <= ADMIN_UPDATE_ID;
486 end
487 if(button_reset_prev && !button_reset) begin
488     state <= IDLE;
489     current_product <= 3'b000;
490 end
491 if (button_view_prev && !button_view) begin
492     state <= CALCULATE;
493 end
494 PAYMENT: begin
495
496     if (enough_money_flag) begin
497         state <= DELIVERY;
498         led <= 4'b1111;
499     end
500     else begin
```



```
499         state <= ERROR;
500     end
501 end
502
503 DELIVERY: begin
504     led <= 4'b1111;
505 //             seven_segment1 <= seven_segment_encode(4'd0);
506 //             seven_segment2 <= seven_segment_encode(4'd0);
507 //             seven_segment3 <= seven_segment_encode(
508 //                 remaining_money / 10);
509 //                 seven_segment4 <= seven_segment_encode(
510 //                     remaining_money % 10);
511 //                     seven_segment1 <= 4'd0; // F
512 //                     seven_segment2 <= 4'd0; // F
513 //                     seven_segment3 <= 4'd0; // F
514 //                     seven_segment4 <= 4'd0; // F
515     data_rom[5] = "S";
516     data_rom[6] = "U";
517     data_rom[7] = "C";
518     data_rom[8] = "C";
519     data_rom[9] = "E";
520     data_rom[10] = "S";
521     data_rom[11] = "S";
522     data_rom[12] = " ";
523     data_rom[13] = " ";
524     data_rom[14] = " ";
525     data_rom[15] = " ";
526     data_rom[16] = " ";
527     data_rom[17] = " ";
528     data_rom[18] = " ";
529     data_rom[19] = " ";
530     data_rom[20] = " ";
531     if (delay_counter < 300000000) begin
532         delay_counter <= delay_counter + 1;
533     end
534     else begin
535         product_inventory[selected_id] <=
536             product_inventory[selected_id] -
537             selected_quantity;
538         state <= RETURN_MONEY;
539         delay_counter <= 0;
540     end
541 end
542
543 ERROR: begin
544     led <= 4'b0000;
545 //             seven_segment1 <= 4'd9; // F
546 //             seven_segment2 <= 4'd9; // F
547 //             seven_segment3 <= 4'd9; // F
548 //             seven_segment4 <= 4'd9; // F
549     data_rom[5] = "E";
550     data_rom[6] = "R";
551     data_rom[7] = "R";
552     data_rom[8] = "O";
```



```
548     data_rom[9] = "R";
549     data_rom[10] = " ";
550     data_rom[11] = " ";
551     data_rom[12] = " ";
552     data_rom[13] = " ";
553     data_rom[14] = " ";
554     data_rom[15] = " ";
555     data_rom[16] = " ";
556     data_rom[17] = " ";
557     data_rom[18] = " ";
558     data_rom[19] = " ";
559     data_rom[20] = " ";
560   if (delay_counter < 300000000) begin
561     delay_counter <= delay_counter + 1;
562   end
563 else begin
564   state <= RETURN_MONEY;
565   delay_counter <= 0;
566 end
567 end
568 RETURN_MONEY: begin
569   led <= 4'b0000;
570
571 // seven_segment1 <= 4'd0;
572 // seven_segment2 <= change / 100;
573 // seven_segment3 <= (change % 100) / 10;
574 // seven_segment4 <= change % 10;
575   data_rom[5] <= "C"; //D
576   data_rom[6] <= "H"; //:
577   data_rom[7] <= "A";
578   data_rom[8] <= "N";
579   data_rom[9] <= "G";
580   data_rom[10] <= "E";
581   data_rom[11] <= ":" ;
582   data_rom[12] <= " ";
583   data_rom[13] <= 8'h30+change / 100;
584   data_rom[14] <= 8'h30+(change % 100) / 10;
585   data_rom[15] <= 8'h30+change % 10;
586   data_rom[16] <= " ";
587   data_rom[17] <= 8'h20;
588   data_rom[18] <= 8'h20;
589   data_rom[19] <= 8'h20;
590   data_rom[20] <= 8'h20;
591   if (delay_counter < 200000000) begin
592     delay_counter <= delay_counter + 1;
593   end
594 else begin
595   state <= IDLE;
596   delay_counter <= 0;
597   current_product <= 3'b000;
598   change <= 8'd0;
599 end
600 end
```



```
601 ADMIN_UPDATE_ID: begin
602     if (delay_counter < 100000000) begin
603         delay_counter <= delay_counter + 1;
604         led <= 4'b1111;
605         end
606     else begin
607         led <= 4'b0001;
608
609     end
610
611 //    seven_segment1 <= seven_segment_encode(4'd0);
612 //    seven_segment2 <= seven_segment_encode(4'd0);
613 //    seven_segment3 <= seven_segment_encode(4'd0);
614 //    seven_segment4 <= seven_segment_encode(
615 //        id_for_update);
616 //        id_for_update <= admin_id;
617 //        seven_segment1 <= 4'd0;
618 //        seven_segment2 <= 4'd0;
619 //        seven_segment3 <= 4'd0;
620 //        seven_segment4 <= id_for_update;
621 //        data_rom[5] = "U";
622 //        data_rom[6] = "P";
623 //        data_rom[7] = "D";
624 //        data_rom[8] = "A";
625 //        data_rom[9] = "T";
626 //        data_rom[10] = "E";
627 //        data_rom[11] = " ";
628 //        data_rom[12] = "I";
629 //        data_rom[13] = "D";
630 //        data_rom[14] = ":";
631 //        data_rom[15] = " ";
632 //        data_rom[16] = 8'h30+id_for_update;
633 //        data_rom[17] = " ";
634 //        data_rom[18] = " ";
635 //        data_rom[19] = " ";
636 //        data_rom[20] = " ";
637 //        data_rom[21] = 8'hc0;
638 //        data_rom[22] = " ";
639 //        data_rom[23] = " ";
640 //        data_rom[24] = " ";
641 //        data_rom[25] = " ";
642 //        data_rom[26] = " ";
643 //        data_rom[27] = " ";
644 //        data_rom[28] = " ";
645 //        data_rom[29] = " ";
646 //        data_rom[30] = " ";
647 //        data_rom[31] = " ";
648 //        data_rom[32] = " ";
649 //        data_rom[33] = " ";
650 //        data_rom[34] = " ";
651 //        data_rom[35] = " ";
652 //        data_rom[36] = " ";
653 //        data_rom[37] = " ";
```



```
653         if(button_select_prev && !button_select) begin
654             state <= ADMIN_UPDATE_QUANTITY;
655         end
656
657         if(!switch_admin)begin
658             state <= IDLE;
659         end
660         if(button_reset_prev && !button_reset) begin
661             state <= IDLE;
662             current_product <= 3'b000;
663         end
664     end
665     ADMIN_UPDATE_QUANTITY: begin
666         led <= 4'b0011;
667         // seven_segment1 <= seven_segment_encode(4'd0);
668         // seven_segment2 <= seven_segment_encode(4'd0);
669         // seven_segment3 <= seven_segment_encode(4'd0);
670         // seven_segment4 <= seven_segment_encode(
671             new_quantity);
672         new_quantity <= admin_quantity;
673         seven_segment1 <= id_for_update;
674         seven_segment2 <= 4'd0;
675         seven_segment3 <= 4'd0;
676         seven_segment4 <= new_quantity;
677         data_rom[5] = "U";
678         data_rom[6] = "P";
679         data_rom[7] = "D";
680         data_rom[8] = "A";
681         data_rom[9] = "T";
682         data_rom[10] = "E";
683         data_rom[11] = " ";
684         data_rom[12] = "Q";
685         data_rom[13] = "U";
686         data_rom[14] = "A";
687         data_rom[15] = "N";
688         data_rom[16] = ":";
689         data_rom[17] = " ";
690         data_rom[18] = 8'h30 + new_quantity;
691         data_rom[19] = " ";
692         data_rom[20] = " ";
693         if(button_select_prev && !button_select) begin
694             product_inventory[id_for_update] <=
695                 new_quantity;
696             state <= ADMIN_UPDATE_PRICE;
697         end
698
699         if(!switch_admin)begin
700             state <= IDLE;
701         end
702         if(button_reset_prev && !button_reset) begin
703             state <= IDLE;
704             current_product <= 3'b000;
705         end
```



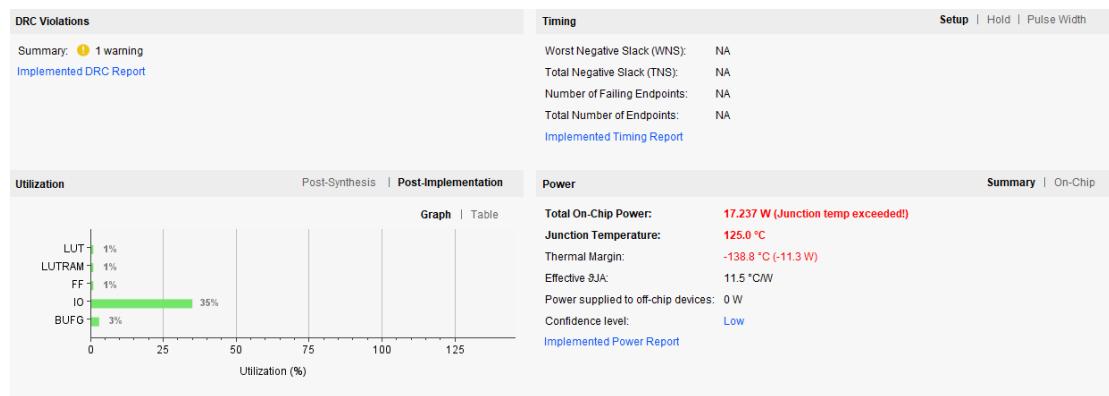
```
704         if (button_view_prev && !button_view) begin
705             state <= ADMIN_UPDATE_ID;
706         end
707     end
708     ADMIN_UPDATE_PRICE: begin
709         led <= 4'b0111;
710         // seven_segment1 <= seven_segment_encode(4'd0);
711         // seven_segment2 <= seven_segment_encode(4'd0);
712         // seven_segment3 <= seven_segment_encode(
713         new_price / 10);
714         // seven_segment4 <= seven_segment_encode(
715         new_price % 10);
716         new_price <= admin_price;
717         // seven_segment1 <= 4'd0;
718         // seven_segment2 <= 4'd0;
719         // seven_segment3 <= new_price / 10;
720         // seven_segment4 <= new_price % 10;
721         data_rom[5] = "U";
722         data_rom[6] = "P";
723         data_rom[7] = "D";
724         data_rom[8] = "A";
725         data_rom[9] = "T";
726         data_rom[10] = "E";
727         data_rom[11] = " ";
728         data_rom[12] = "P";
729         data_rom[13] = "R";
730         data_rom[14] = "I";
731         data_rom[15] = "C";
732         data_rom[16] = "E";
733         data_rom[17] = ":";
734         data_rom[18] = " ";
735         data_rom[19] = 8'h30 + new_price / 10;
736         data_rom[20] = 8'h30 + new_price % 10;
737         if(button_select_prev && !button_select) begin
738             product_price[id_for_update] <= new_price
739             ;
740             delay_counter <= 0;
741             state <= ADMIN_UPDATE_ID;
742         end
743         if(!switch_admin)begin
744             current_product <= 3'b000;
745             state <= IDLE;
746         end
747         if(button_reset_prev && !button_reset) begin
748             state <= IDLE;
749             current_product <= 3'b000;
750         end
751         if (button_view_prev && !button_view) begin
752             state <= ADMIN_UPDATE_QUANTITY;
```



```
754           end
755       end
756   default: state <= IDLE;
757 endcase
758 end
759 endmodule
```



## 5 Đánh giá tổng quan thiết kế



Hình 6: Project Summary

### Những mặt đạt được:

- Quá trình Synthesis và Implementation đã hoàn thành.
- Thiết kế không có lỗi lớn về cấu trúc mạch (không có Critical Warnings hoặc Errors).
- DRC (Design Rule Check) chỉ báo 1 cảnh báo (warning) nhỏ, nghĩa là các quy tắc thiết kế cơ bản đã được đáp ứng.
- Sử dụng tài nguyên FPGA hiệu quả
  - LUT (Look-Up Table): Chỉ sử dụng 1%
  - LUTRAM và FF (Flip-Flop): Sử dụng rất ít (1%).
  - BUFG (Global Clock Buffer): Chỉ dùng 3%.
  - IO: Sử dụng 35%.

### Những mặt cần cải thiện:

- Vấn đề nhiệt độ (Junction Temperature Exceeded!)
  - Junction Temperature hiện tại là 125.0°C vượt quá ngưỡng an toàn, có thể gây hỏng thiết bị.
  - Công suất tiêu thụ (Total On-Chip Power) là 17.237 W, quá cao so với mức thông thường.
- Chưa có báo cáo timing đầy đủ.

Tóm lại, thiết kế đã hoàn thiện các chức năng cơ bản và đáp ứng yêu cầu tổng hợp, nhưng để có thể triển khai trên phần cứng thực tế, cần cải thiện thêm về tối ưu công suất, nhiệt độ và timing để đảm bảo thiết kế hoạt động ổn định và hiệu quả.



## 6 Kết luận

Với nguồn tài liệu tham khảo hạn chế, nhóm đã gặp khó khăn từ lúc bắt đầu khi không biết phải bắt đầu từ đâu. Tuy vậy, nhóm đã lên kế hoạch và hoàn thiện từng phần của dự án. Dự án thiết kế mạch máy bán hàng tự động đã hoàn thiện các chức năng cơ bản bao gồm lựa chọn sản phẩm, hiển thị thông tin, xử lý thanh toán và quản lý số lượng hàng hóa. Sự tích hợp giữa các thành phần phần cứng như công tắc, đèn LED và màn hình 7 đoạn với bộ điều khiển FPGA đã giúp hệ thống hoạt động ổn định và chính xác, đáp ứng các yêu cầu ban đầu.

### *Phương hướng phát triển trong tương lai:*

- **Kết nối IoT:** Tích hợp mạng không dây để theo dõi trạng thái máy từ xa, như số lượng hàng tồn hoặc doanh thu.
- **Hệ thống thanh toán hiện đại:** Hỗ trợ thanh toán bằng mã QR hoặc thẻ từ thay vì chỉ sử dụng nút nhấn và thanh toán vật lý.
- **Tăng cường trải nghiệm người dùng:** Sử dụng màn hình cảm ứng hoặc giao diện người dùng thân thiện hơn.

Những cải tiến này không chỉ làm tăng giá trị của hệ thống mà còn góp phần nâng cao khả năng ứng dụng thực tế, mở rộng phạm vi sử dụng trong các lĩnh vực khác nhau. Dự án sẽ tiếp tục là tiền đề cho các nghiên cứu và phát triển chuyên sâu hơn trong tương lai.



## Tài liệu tham khảo

- [1] Nguyen Dinh Phu. Thiết kế mạch giao tiếp LCD 20x4 hiển thị chuỗi dùng FPGA và VHDL. Truy cập từ <https://www.youtube.com/watch?v=yc4H0g8W0xA&t=80s>
- [2] All About FPGA. Learn FPGA 9: Displaying message on 16x2 LCD using EDGE Spartan 7 FPGA kit. Truy cập từ <https://www.youtube.com/watch?v=WiIKw81cRlQ>
- [3] XIAMEN AMOTEC DISPLAY CO.,LTD. SPECIFICATIONS OF LCD MODULE. Truy cập từ <https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf>