EE3043 Computer Architecture
Semester 251

# MILESTONE 1: DESIGN OF A VENDING MACHINE

*GVHD: Trần Hoàng Linh*

Student:
Thái Khắc Hùng - 2311306
Trương Tuấn Hải -
Phạm Vũ Hoàng Phúc -

## 1. Introduction:

Mốc đầu tiên trong học phần là thiết kế một máy bán hàng tự động bằng SystemVerilog. Sinh viên được yêu cầu tuân thủ các quy tắc lập trình cụ thể và nộp báo cáo. Nếu bạn phát hiện bất kỳ lỗi nào hoặc có đề xuất để cải thiện tài liệu này, vui lòng gửi email cho trợ giảng theo địa chỉ cxhai.sdh221@hcmut.edu.vn với tiêu đề "[CA203 FEEDBACK]".

## 2. Mục tiêu:

+ Hiểu các quy tắc lập trình.
+ Ôn lại kiến thức cơ bản về thiết kế logic và khái niệm FSM (Finite State Machine – máy trạng thái hữu hạn).
+ Thiết kế một máy bán hàng tự động bằng SystemVerilog.

**Quy tắc lập trình:**

Khi lập trình, điều quan trọng cần nhớ là bạn sẽ phải quay lại mã của mình nhiều lần cho các mục đích khác nhau như gỡ lỗi và cải tiến. Giữ cho mã nguồn sạch sẽ sẽ giúp bạn tiết kiệm được rất nhiều thời gian. Ngược lại, một tệp mã lộn xộn có thể gây choáng ngợp về mặt trực quan và làm gián đoạn tinh thần, khiến bạn tự hỏi mình đã làm gì. Trong học phần này, bạn được khuyến nghị mạnh mẽ nên tuân thủ các quy tắc lập trình được cung cấp trong đường dẫn sau.

**Đề bài:**

Máy bán hàng tự động là một loại máy phân phối nhận tiền xu hoặc tiền giấy và phát ra nước ngọt hoặc đồ ăn nhẹ. Một bộ điều khiển máy bán hàng tự động đơn giản được mô tả trong Hình 1.

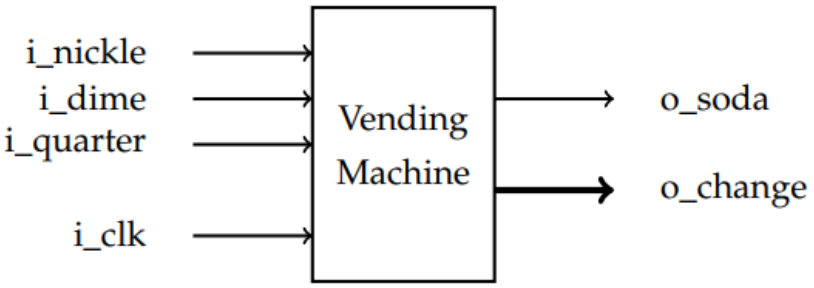



Figure 1: Vending machine's ports

Thiết kế một máy bán hàng tự động đáp ứng các yêu cầu sau:

+ Máy bán hàng tự động có khả năng chấp nhận các loại tiền xu: ¢5 (Nickel), ¢10 (Dime), ¢25 (Quarter). Tuy nhiên, tại mỗi chu kỳ xung clock, máy chỉ chấp nhận một đồng xu.
+ Khi số tiền đã nạp vượt quá ¢20, máy sẽ phát ra một lon nước ngọt và tính toán tiền thối lại chính xác.
+ Tiền thối được biểu diễn bằng dữ liệu 3-bit:

  000 → ¢0
  001 → ¢5
  010 → ¢10
  011 → ¢15
  100 → ¢20
+ Ví dụ: Khách hàng đưa vào một đồng Dime (¢10), sau đó đưa thêm một đồng Quarter (¢25). Trong chu kỳ kế tiếp, hệ thống sẽ phát ra một lon nước ngọt và trả lại ¢15 tiền thối.
+

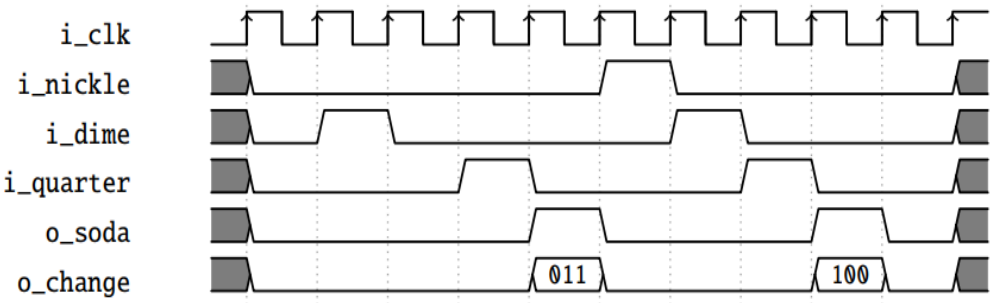



Figure 2: Waveform of an example vending machine

## 3. SPECIFICATION

| Signal | Width | Direction |
|---|---|---|
| i_clk | 1 | Input |
| i_nickle | 1 | Input |
| i_dime | 1 | Input |
| i_quarter | 1 | Input |
| o_soda | 1 | Output |
| o_change | 3 | Output |

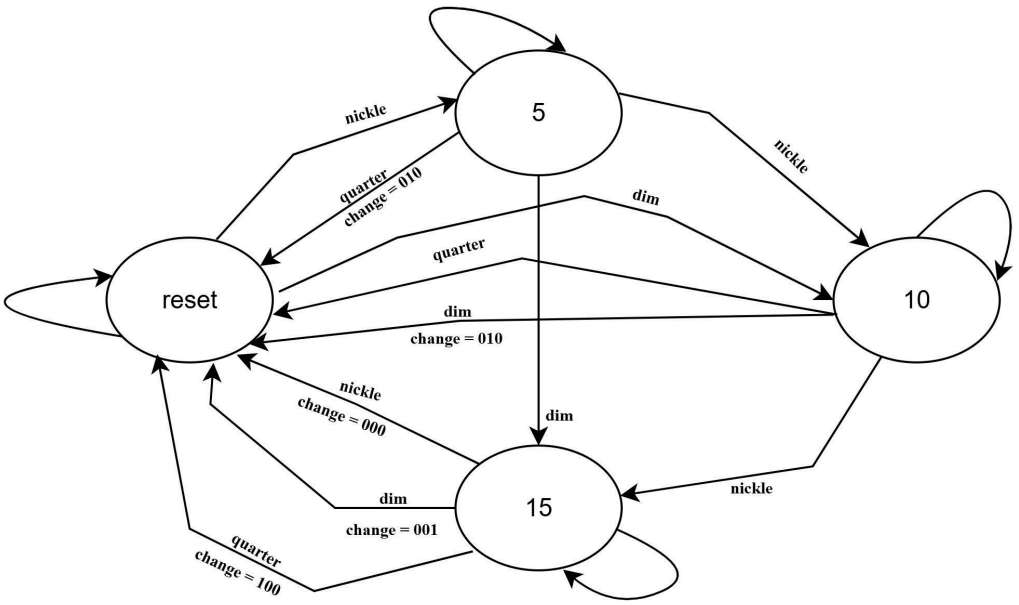
## 2. CONTROL UNIT
## FINITE STATE MACHINE

_ Nhóm quyết định sử dụng ngõ ra **Mealy** thay vì **Moore** vì có thể tối ưu được các trạng thái. Để có thể đáp ứng yêu cầu *o_soda* và *o_change* cập nhật sau một chu kỳ clock thì nhóm sẽ sử dụng *next_soda* và *next_change* ở combination và sau đó mới cập nhật bởi xung clock.

**Bảng trạng thái:** Dựa vào đề bài, ta có thể lập nên một bảng trạng thái như sau:

| **Trạng thái hiện tại** | **Trạng thái kế tiếp** | | | | **Ngõ ra Mealy** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | i_ quarter | i_ dime | i_ nickle | Else | i_quarter | | i_dime | | i_nickle | | Else | |
| | | | | | o_ soda | o_ change | o_ soda | o_ change | o_ soda | o_ change | o_ soda | o_ change |
| S0 ( 2'b00) (Chưa bỏ hoặc đã bỏ vào >= ¢20) | S0 | S2 | S1 | S0 | 1'b1 | 3'b001 | 1'b0 | 3'b000 | 1'b0 | 3'b000 | 1'b0 | 3'b000 |
| S1 (2'b01) (Đã bỏ vào | S0 | S3 | S2 | S1 | 1'b1 | 3'b010 | 1'b0 | 3'b000 | 1'b0 | 3'b000 | 1'b0 | 3'b000 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ¢5) | | | | | | | | | | | | |
| S2 (2’b10) (Đã bỏ vào ¢10) | S0 | S0 | S3 | S2 | 1’b1 | 3’b011 | 1’b1 | 3’b000 | 1’b0 | 3’b000 | 1’b0 | 3’b000 |
| S3 (2’b11) (Đã bỏ vào ¢15) | S0 | S0 | S0 | S3 | 1’b1 | 3’b100 | 1’b1 | 3’b001 | 1’b1 | 3’b000 | 1’b0 | 3’b000 |




**Phần code:** Sử dụng QuestaSim để viết code Verilog.

Code design

```
// Code your design here

module vending_machine(
  input wire i_clk,
  input wire i_nickle, i_dime, i_quarter,
  output reg o_soda,
  output reg [3:0] o_change);

  localparam S0 =2'b00, S1=2'b01, S2= 2'b10, S3= 2'b11;
  reg [1:0]current_state,next_state;
  reg next_soda;
  reg [3:0]next_change;

  always @(posedge i_clk) begin
    current_state <= next_state;
    o_soda <= next_soda;
    o_change <= next_change;
  end
```

```
always @(*) begin
  next_soda = 0;
  next_change = 3'b000;

  case (current_state)
    S0 : case ({i_quarter, i_dime, i_nickle})
      3'b100 : begin
        next_state = S0;
        next_soda = 1'b1;
        next_change = 3'b001;
      end
      3'b010 : begin
        next_state = S2;
        next_soda = 1'b0;
        next_change = 3'b000;
      end
      3'b001: begin
        next_state = S1;
        next_soda = 1'b0;
        next_change = 3'b000;
      end
      default: next_state = S0;
    endcase

    S1: case ({i_quarter, i_dime, i_nickle})
      3'b100 :begin
        next_state = S0;
        next_soda = 1'b1;
        next_change = 3'b010;
      end
      3'b010: begin
        next_state = S3;
        next_soda = 1'b0;
        next_change = 3'b000;
      end
      3'b001: begin
        next_state = S2;
        next_soda = 1'b0;
        next_change = 3'b000;
      end
      default: next_state = S1;
    endcase

     S2: case ({i_quarter, i_dime, i_nickle})
      3'b100 :begin
        next_state = S0;
        next_soda = 1'b1;
        next_change = 3'b011;
      end
      3'b010: begin
        next_state = S0;
        next_soda = 1'b1;
        next_change = 3'b000;
      end
      3'b001: begin
        next_state = S3;
        next_soda = 1'b0;
        next_change = 3'b000;
      end
```

```
      default: next_state = S2;
    endcase

    S3: case ({i_quarter, i_dime, i_nickle})
      3'b100 :begin
        next_state = S0;
        next_soda = 1'b1;
        next_change = 3'b100;
      end
      3'b010: begin
        next_state = S1;
        next_soda = 1'b1;
        next_change = 3'b001;
      end
      3'b001: begin
        next_state = S0;
        next_soda = 1'b1;
        next_change = 3'b000;
      end
      default: next_state = S3;
    endcase

    default: begin
      next_state = S0;
      next_soda = 0;
      next_change =3'b000;
    end

  endcase
 end

endmodule
```

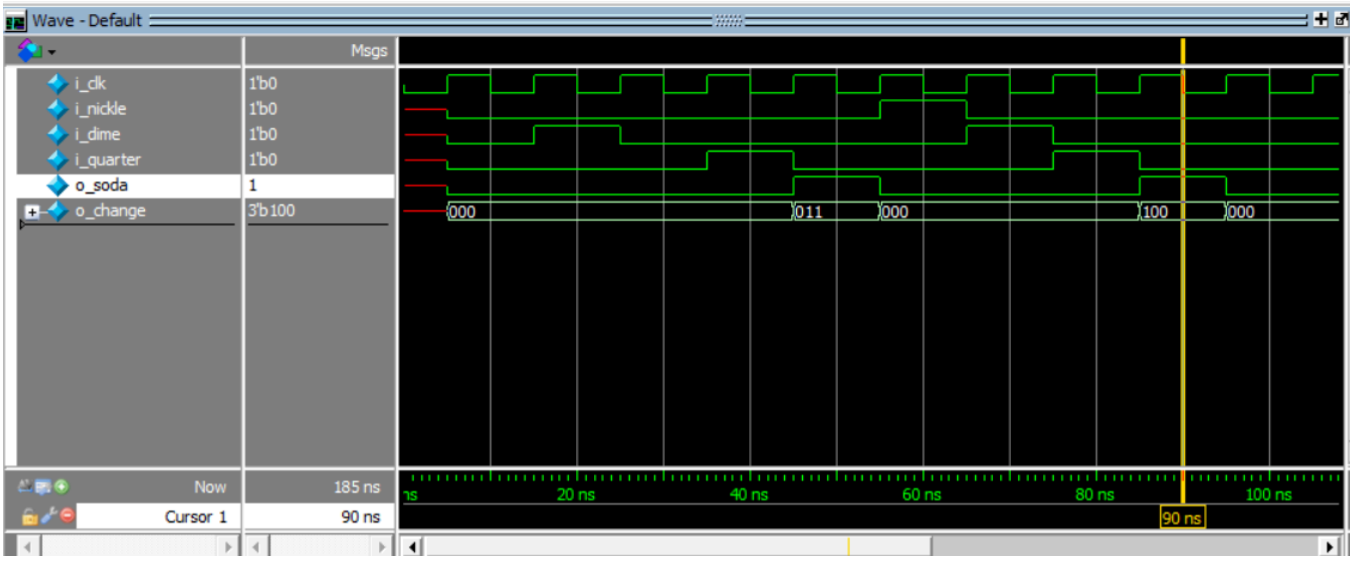
## Code testbench:

```
// Code your testbench here
module tb_vending_machine;
  reg i_clk;
  reg i_nickle, i_dime, i_quarter;
  wire o_soda;
  wire [2:0] o_change;

  vending_machine dut(
    .i_clk(i_clk),
    .i_nickle(i_nickle),
    .i_dime(i_dime),
    .i_quarter(i_quarter),
    .o_soda(o_soda),
    .o_change(o_change)
  );

  initial begin
    i_clk =0 ;
    forever #5 i_clk = ~i_clk;
```

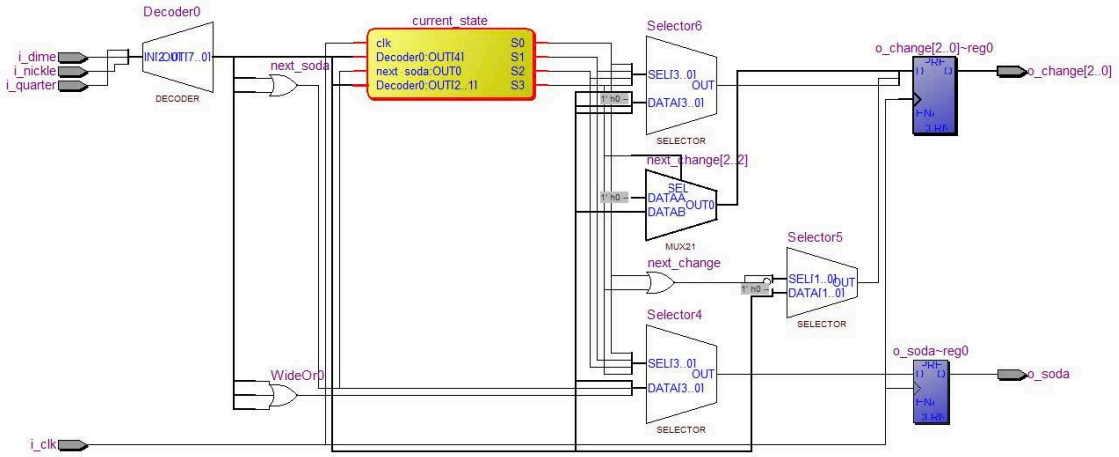
```
  end

  initial begin

    #5 i_nickle = 0;
       i_dime = 0;
       i_quarter = 0;


    #10 i_dime = 1;
    #10 i_dime = 0;

    #10 i_quarter = 1;
    #10 i_quarter = 0;

    #10 i_nickle = 1;
    #10 i_nickle = 0;
           i_dime =1;
    #10 i_dime = 0;
           i_quarter =1;
    #10 i_quarter = 0;

    #100;
    $finish;
  end
endmodule
```

KẾT QUẢ MÔ PHỎNG

WAVEFORM:



DATAPATH:




CONTROL UNIT: