A project on

# Dhaka Metro Rail Automatic Ticketing System Using Verilog HDL

Project For Fulfilment

**Fourth Year Project Work**

**Submitted by**

Name: Forhad Hossain
Roll: 1515032
Reg:1166
Session:2015-2016

Dept. of Electrical & Electronic Engineering.
Islamic University, Kushtia.

## CERTIFICATION

I am pleased to certify that the project work entitled **“Dhaka Metro Rail Automatic Ticketing System Using Verilog HDL”** submitted by **Md. Forhad Hossain**, Roll no. 1515032, Reg. no. 1166, Session: 2015-2016 has performed the project under my supervision for the fulfilment of the Fourth Year Project Work of department of Electrical & Electronic Engineering. Islamic University, Kushtia.

Signature……………………

**Dr. Md. Monjarul Alam**
Professor
Dept. of Electrical & Electronic Engineering.
Islamic University, Kushtia.

## ABSTRACT

This project is a synthesis of practical investigation, theoretical analysis and literature reading. Using Verilog HDL language to make Dhaka Metro rail automatic ticket selling system. The design of this ticketing system takes convenience, quickness and simplicity as the core, and takes saving time for passengers as the guide design. Firstly, we have studied the development of Metro rail ticketing system at home and abroad, and then studies the basic needs of this ticketing system. This project designs the Dhaka Metro rail automatic ticket selling system which composed of ticket selection module, money calculation module, change processing module and display interface module.

## INTRODUCTION

Bangladesh is now a developing country. Every government. Office has become developed by digital technologies. Government takes many steps to reduce the traffic difficulty in the capital city of Dhaka. As now they are developing a metro rail to overcome this problem. Booking the tickets through ticket counters is time consuming process. we usually face many problems at ticketing counter of exact currency to be paid while booking the reservation or unreserved tickets.

To overcome this problem, we designed electronic ticketing machine by using Verilog HDL. It will work as same as ATM machine. This machine is flexible and reliable compare to Microcontroller based design machines. The project contains the FSM (Finite State Machine) especially Moore state machine. The different states in the FSM define each operation of the electronic ticket machine.

## 1.0 DESIGN PRINCIPLE

### 1.1 Verilog HDL

Verilog HDL language clearly defines many kinds of syntax, and also defines clear simulation and simulation semantics for the structure of each syntax. For this reason, the model written in such language can be realized and verified by Verilog simulation instrument. At the same time, we can see that Verilog language has the following advantages: it can accurately, simply and clearly describe various levels of systems. The description of code has nothing to do with the specific process, which improves the repeatability of design and promotes the standardization of design. To sum up, we can see that Verilog language is a complete and excellent description language. Its ability is enough to help us complete very complex chip design or complete electronic system design.

## 1.2 Cadence Tools

Cadence Design Systems is a company that develops software, hardware, and silicon intellectual property (IP) for designing integrated circuits (ICs), printed circuit boards (PCBs), and electronic systems. They offer a range of tools and solutions to aid in various aspects of electronic design automation (EDA).

**Genus:** Genus is a leading RTL synthesis solution offered by Cadence. RTL synthesis is the process of converting a Register Transfer Level (RTL) description of a digital circuit into a gate-level representation that can be implemented in silicon.

**Encounter:** The Encounter platform from Cadence focuses on physical design and implementation of digital integrated circuits. It covers various stages of the design flow, from initial placement and routing to final sign off.

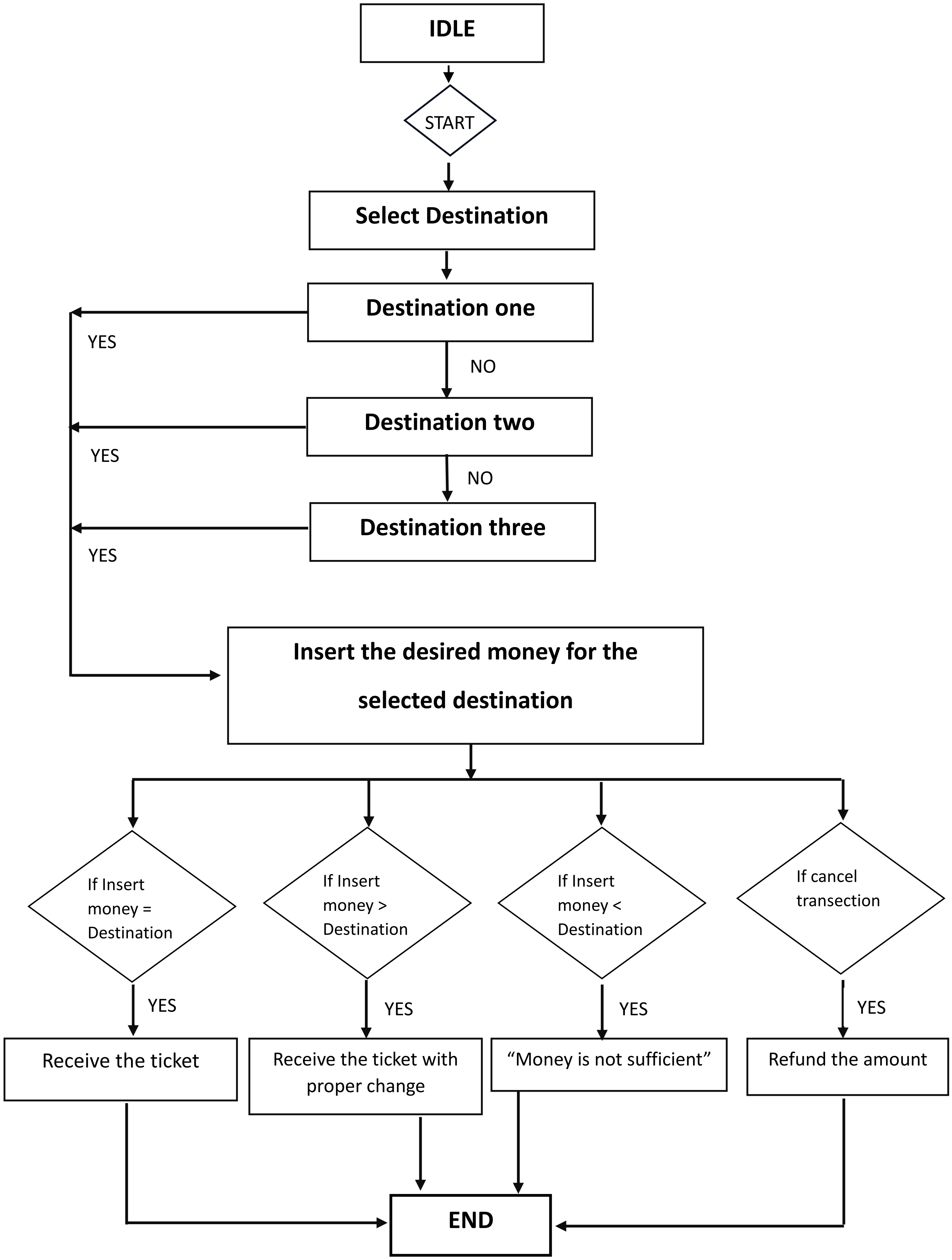
# 2.0 DESIGN METHODOLOGY

## 2.1 Operation

This project is designed for the proposed machine which can vend tickets for the three different destinations. A two bits input select line is used to select the different destinations (01 will be destination1, 10 will be destination2, 11 will be destination3). A note input signal is used for indicating different currencies. We are considering its decimal equivalent values for the value of the currencies so that it will be easy to analyze the waveforms. Our design accepts only 10, 20 and 50 taka notes. A cancel input signal is used to decline the transaction. The proposed design works on the positive edge of the signal clock and returns to initial state when reset button is pressed. Ticket1, Ticket2, and Ticket3 are the one-bit output signals represent the tickets for destination1, destination2 and destination3 respectively. Change is an output signal which returns the excess amount inserted while purchasing the ticket. Refund is an output signal returns the inserted amount on the cancellation of the transaction. The proposed machine is designed using FSM modelling and is coded in Verilog HDL language.

## 2.2 Flow Chart

Initially we have to select the desired destination. Then we will insert the currency in notes taka, if we insert the equal amount of currency for the selected destination then we will get the ticket, if insert more amount of currency then we'll receive change, if insert less amount then display will show "money is not sufficient". Also we can cancel the ticket then we'll be refunded.


IDLE
START
Select Destination
Destination one
YES
NO
Destination two
YES
NO
Destination three
YES
Insert the desired money for the selected destination
If Insert money = Destination
If Insert money > Destination
If Insert money < Destination
If cancel transection
YES
YES
YES
YES
Receive the ticket
Receive the ticket with proper change
"Money is not sufficient"
Refund the amount
END

# 3.0 SIMULATION

## 3.1 Verilog Design Code

```
module metro_rail(destination_one, destination_two, destination_three, balance, money,
select_destination, extra_cash, clock, reset)
output reg destination_one;
output reg destination_two;
output reg destination_three;
output reg [3:0]balance;

input wire [3:0]money;
input wire [1:0]select_destination;
input wire [3:0]extra_cash;
input wire clock;
input wire reset;

reg [2:0]present_state, next_state;

parameter [3:0] money_10=4'b0001;
parameter [3:0] money_20=4'b0010;
parameter [3:0] money_50=4'b0011;

parameter [1:0]select_destination_one=2'b01;
parameter [1:0]select_destination_two=2'b10;
parameter [1:0]select_destination_three=2'b11;

parameter [2:0] idle= 3'b000;
parameter [2:0] ten= 3'b001;
parameter [2:0] twenty= 3'b010;
parameter [2:0] fifty= 3'b011;

initial
begin
        present_state <= idle;
        next_state <= idle;
end
```

```
always @(posedge clock)
begin
        if(reset)
                next_state <= idle;
        else
                case(present_state)
                idle: if(money==money_10 && select_destination == select_destination_one)
                        next_state<=ten;
                else if(money==money_10 && select_destination == select_destination_two)
                        next_state<=twenty;
                else if(money==money_10 && select_destination == select_destination_three)
                        next_state<=fifty;

                else if(money==money_20 && select_destination == select_destination_two)
                        next_state<=twenty;
                else if(money==money_20 && select_destination == select_destination_one)
                        next_state<=ten;
                else if(money==money_20 && select_destination == select_destination_three)
                        next_state<=fifty;

                else if(money==money_50 && select_destination == select_destination_two)
                        next_state<=twenty;
                else if(money==money_50 && select_destination == select_destination_three)
                        next_state<=fifty;
                else if(money==money_50 && select_destination == select_destination_one)
                        next_state<=ten;


                ten: if(money==money_10 && select_destination == select_destination_one)
                        next_state<=ten;
                else if(money==money_10 && select_destination == select_destination_two)
                        next_state<=twenty;
                else if(money==money_10 && select_destination == select_destination_three)
                        next_state<=fifty;

                else if(money==money_20 && select_destination == select_destination_one)
                        next_state<=ten;
                else if(money==money_20 && select_destination == select_destination_two)
```

```
            next_state<=twenty;
else if(money==money_20 && select_destination == select_destination_three)
            next_state<=fifty;

else if(money==money_50 && select_destination == select_destination_two)
            next_state<=twenty;
else if(money==money_50 && select_destination == select_destination_three)
            next_state<=fifty;
else if(money==money_50 && select_destination == select_destination_one)
            next_state<=ten;

twenty: if(money==money_10 && select_destination == select_destination_one)
            next_state<=ten;
else if(money==money_10 && select_destination == select_destination_two)
            next_state<=twenty;
else if(money==money_10 && select_destination == select_destination_three)
            next_state<=fifty;

else if(money==money_20 && select_destination == select_destination_one)
            next_state<=ten;
else if(money==money_20 && select_destination == select_destination_two)
            next_state<=twenty;
else if(money==money_20 && select_destination == select_destination_three)
            next_state<=fifty;

else if(money==money_50 && select_destination == select_destination_two)
            next_state<=twenty;
else if(money==money_50 && select_destination == select_destination_three)
            next_state<=fifty;
else if(money==money_50 && select_destination == select_destination_one)
            next_state<=ten;

fifty: if(money==money_10 && select_destination == select_destination_one)
            next_state<=ten;
else if(money==money_10 && select_destination == select_destination_two)
            next_state<=twenty;
else if(money==money_10 && select_destination == select_destination_three)
            next_state<=fifty;
```

```
                else if(money==money_20 && select_destination == select_destination_one)
                        next_state<=ten;
                else if(money==money_20 && select_destination == select_destination_two)
                        next_state<=twenty;
                else if(money==money_20 && select_destination == select_destination_three)
                        next_state<=fifty;

                else if(money==money_50 && select_destination == select_destination_two)
                        next_state<=twenty;
                else if(money==money_50 && select_destination == select_destination_three)
                        next_state<=fifty;
                else if(money==money_50 && select_destination == select_destination_one)
                        next_state<=ten;

        default: next_state <= idle;
endcase
present_state <= next_state;
end

always @(posedge clock)
begin
        if(reset)
                present_state <= idle;
        else
        begin
                case(present_state)
                idle:begin
                        destination_one <= 1'b0;
                        destination_two <= 1'b0;
                        destination_three <= 1'b0;
                        balance = money;
                end

                ten:begin
                        if(money == money_10)
                begin
                        destination_one <= 1'b1;
```

```
            destination_two <= 1'b0;
            destination_three <= 1'b0;
            balance = money - 10;
end

else if(money == money_20)
begin
            destination_one <= 1'b1;
            destination_two <= 1'b0;
            destination_three <= 1'b0;
            balance = money - 10;
end

else if(money == money_50)
begin
            destination_one <= 1'b1;
            destination_two <= 1'b0;
            destination_three <= 1'b0;
            balance = money - 10;
end
end



twenty:begin
            if(money == money_10)
begin
            destination_one <= 1'b0;
            destination_two <= 1'b0;
            destination_three <= 1'b0;
if (extra_cash == money_10)
begin
            destination_one <= 1'b0;
            destination_two <= 1'b1;
            destination_three <= 1'b0;
end
else
            balance = 4'b0001;
```

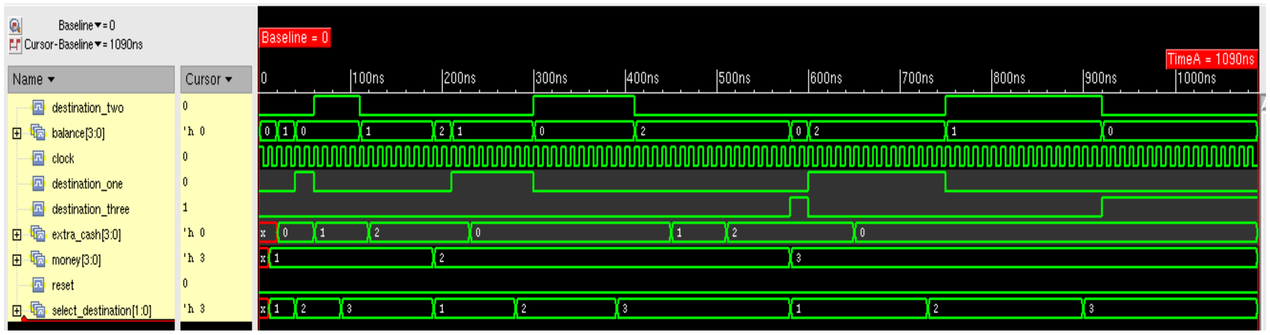
```
end

else if(money == money_20)
begin
            destination_one <= 1'b0;
            destination_two <= 1'b1;
            destination_three <= 1'b0;
            balance = money - 20;
end

else if(money == money_50)
begin
            destination_one <= 1'b0;
            destination_two <= 1'b1;
            destination_three <= 1'b0;
            balance = money - 20;
end
end



fifty:begin
            if(money == money_10)
begin
            destination_one <= 1'b0;
            destination_two <= 1'b0;
            destination_three <= 1'b0;
if (extra_cash == money_20 & money_20)
begin
            destination_one <= 1'b0;
            destination_two <= 1'b0;
            destination_three <= 1'b1;
end
else
            balance = 4'b0001;
end

else if(money == money_20)
```

```
                begin
                        destination_one <= 1'b0;
                        destination_two <= 1'b0;
                        destination_three <= 1'b0;
                if (extra_cash == money_10 & money_20)
                begin
                        destination_one <= 1'b0;
                        destination_two <= 1'b0;
                        destination_three <= 1'b1;
                end
                else
                        balance = 4'b0010;
                end

                else if(money == money_50)
                begin
                        destination_one <= 1'b0;
                        destination_two <= 1'b0;
                        destination_three <= 1'b1;
                        balance = money - 50;
                end
        end

        default: begin
                        destination_one <= 1'b0;
                        destination_two <= 1'b0;
                        destination_three <= 1'b0;
                        balance = 4'b0000;
                        end
                endcase
        end
end
endmodule
```

## 3.2 Test Bench

```
module metro_rail_tb;
wire destination_one;
wire destination_two;
wire destination_three;
wire [3:0]balance;

reg [3:0]money;
reg [1:0]select_destination;
reg [3:0]extra_cash;
reg clock;
reg reset;

metro_rail uut (destination_one, destination_two, destination_three, balance, money,
select_destination, extra_cash, clock, reset);

always #5 clock = ~clock;

initial
begin

 #0 reset= 1'b0;
 #0 clock = 1'b1;

#10 money = 4'b0001; select_destination = 2'b01;
#20 money = 4'b0001; select_destination = 2'b10;
#30 money = 4'b0001; select_destination = 2'b11;
#40 money = 4'b0010; select_destination = 2'b01;
#50 money = 4'b0010; select_destination = 2'b10;
#60 money = 4'b0010; select_destination = 2'b11;
#70 money = 4'b0011; select_destination = 2'b01;
#80 money = 4'b0011; select_destination = 2'b10;
#90 money = 4'b0011; select_destination = 2'b11;

#100 $finish;
end
endmodule
```

## Waveform:



## CONCLUSION

The system avoids queues, saves time and speeds up the ticket sales process. It has much better features like cancellation of tickets between event and the machine refunds if no tickets are available. Its design is very flexible and reliable.

In the future, we can easily improve the algorithm for more destinations and add some of the following features - the ability to pay by card and the ability to queue and process orders according to FIFO.

## REFERENCES

I. B. Caulfield & M.O Mahony , "Passenger Requirements of a Public Transport Ticketing System", Proceedings of the 8th International. IEEE Conference on Intelligent Transportation Systems Vienna, Austria, pp- 32-37,2005.

II. Fauziah Zainuddin, Norlin Mohd Ali, Roslina Mohd Sidek, Awanis Romli, Nooryati Talib & Mohd. Izham Ibrahim "Conceptual Modeling for Simulation: Steaming frozen Food Processing in Vending Machine", International Conference on Computer Science and Information Technology, University Malaysia Pahang, pp.145-149 ,2009

III. Ana Monga, Balwinder Singh, "Finite State Machine based Vending Machine Controller with Auto-Billing Features", International Journal 7. of VLSI design & Communication Systems (VLSICS) Vol.3, No.2, pp 19-28, 2012.