CSE312

ELECTRONIC DESIGN AUTOMATION

JUNIOR CESS

FALL 2023

PROJECT #1 [ATM]

Name	ID
Seif Yasser Ahmed	21P0102
Mohamed Salah Fathy	21P0117
Ali Tarek Abdelmonem	21P0123
Gaser Zaghlol Hassan	21P0052
Youssef Tamer Hossam	21P0138
Mohamed Mostafa Mamdouh	21P0244
Tarek Hadef Mostafa	21P0199

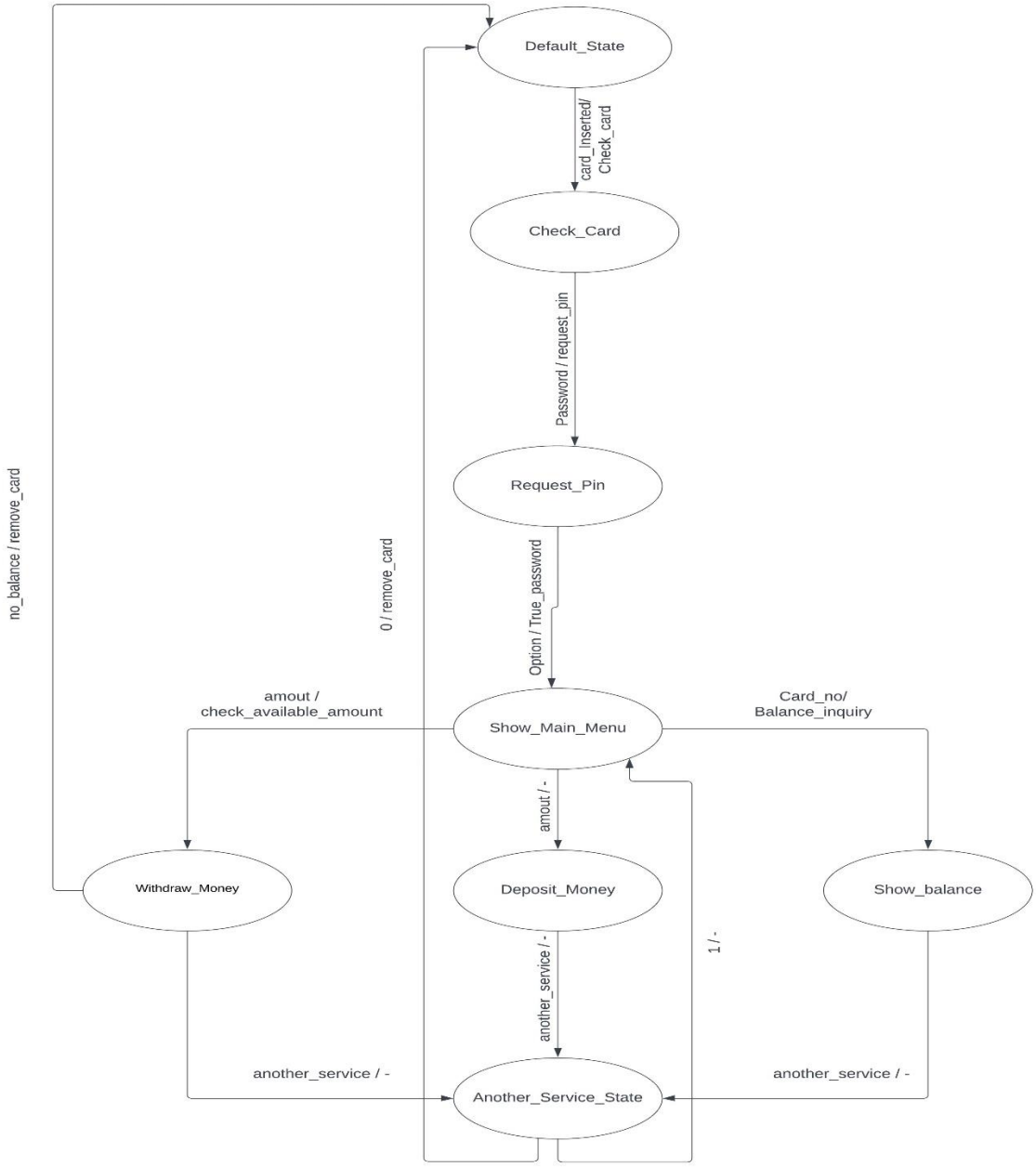
Table of Contents

Introduction

Our ATM System project aims to provide a solution that allows users to perform essential banking operations. From card insertion to language selection, password verification, and main menu navigation, our system is designed with user experience and security in mind, our project is implemented using a tool called QuestaSim and a HDL called Verilog, when the user inserts a card, the ATM does the following features:

- **Language Selection:** Our ATM system offers multilingual support, allowing users to choose their preferred language for a personalized banking experience. Whether it's English, French, German, or Italian, users can seamlessly interact with the ATM in a language they are comfortable with.
- **Password Verification:** Security is a top priority in the design of our ATM system. Users will be required to enter a secure PIN (Personal Identification Number) to access their accounts. The system employs robust authentication mechanisms to ensure the confidentiality of user information.
- **Main Menu Options:**
    - **Deposit:** Users can easily deposit money into their accounts through a straightforward and intuitive interface. The system ensures accuracy and reliability in processing deposits.
    - **Withdraw:** With our ATM system, users can withdraw cash quickly and efficiently. The system considers factors such as available balance and transaction limits to provide a smooth withdrawal experience.
    - **Show Balance:** Users can check their account balances at any time. The system displays the current balance, providing users with real-time financial information.

System Architecture



Verification Plan

We made a reference model in C++, like our model.

Reference Model

1. When the user inserts his card, the model enters an input ‘CARD” to begin the process.
2. First the model asks about the language to deal with

```
Please select language (00=ENGLISH, 01=FRENCH, 10=GERMAN): 00
Enter your password: Pass1
Welcome! Please select an option:
1. Show Balance
2. Withdraw Money
3. Deposit Money
4. Remove Card
5. Another Service
1
Your current balance is: $1000
```

3. Then it asks about the password of the card and if it’s correct it shows the main menu of the machine
4. The machine shows the available options to choose from
5. First is the show balance option that shows the user’s balance in this card.
6. Second is the withdrawal of money.

```
Welcome! Please select an option:
1. Show Balance
2. Withdraw Money
3. Deposit Money
4. Remove Card
5. Another Service
2
Enter the amount you want to withdraw: 500
Successfully withdrawn $500.
```

7. Then it asks him to enter the amount he wants to withdraw and if the amount is bigger than the actual amount in the card it shows an error that the amount of withdrawal is higher than your balance.

8. Third is the deposit money.

```
3
Enter the amount you want to deposit: 450
Successfully deposited $450.
Welcome! Please select an option:
1. Show Balance
2. Withdraw Money
3. Deposit Money
4. Remove Card
5. Another Service
1
Your current balance is: $2450
```

9. Fourth is the removal of the card to end the process immediately.

```
Your current balance is: $2450
Welcome! Please select an option:
1. Show Balance
2. Withdraw Money
3. Deposit Money
4. Remove Card
5. Another Service
4
Please wait while your card is ejected.
Card removed. Thank you for using our ATM.
```

10. Fifth option is another service which contains many other options:

```
5. Another Service
5
1. Change PIN
2. View Transaction History
3. Back to Main Menu
Please choose an additional service:
enter ur choice:1
```

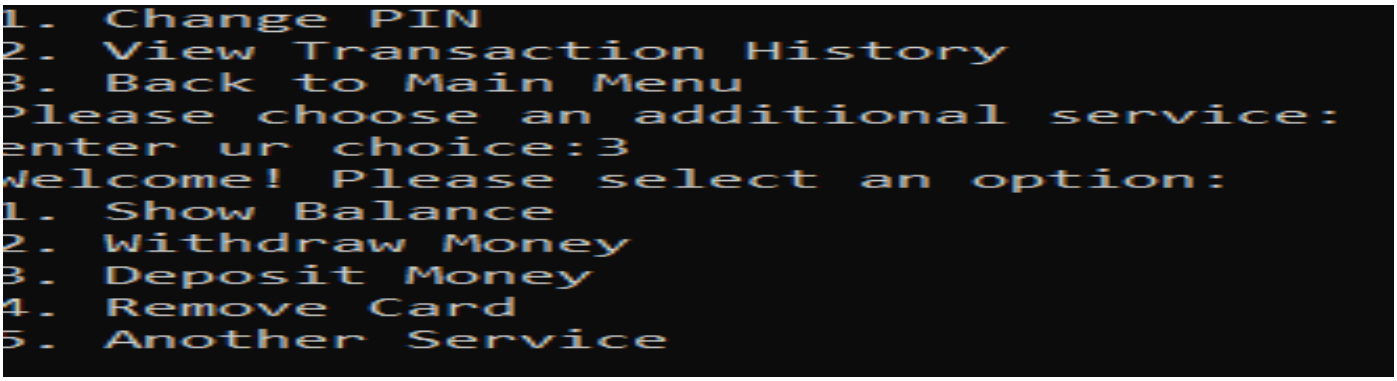
a. First is the change pin service.

```
enter ur choice:1
Enter your current PIN: Pass1
Enter your new PIN: pass2
Your PIN has been successfully changed.
```

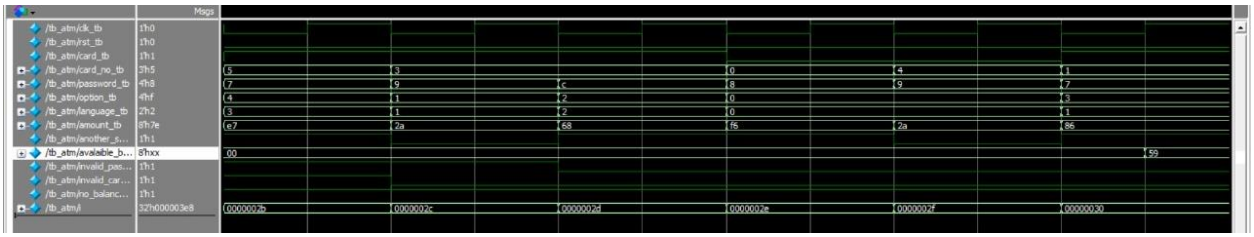
b. Second is the transaction history.

```
1. Change PIN
2. View Transaction History
3. Back to Main Menu
Please choose an additional service:
enter ur choice:2
show balance:
1000
```

Third is back to the main menu.



DUT Results



Coverage

Questa Coverage Report

Number of tests run:	1
Passed:	1
Warning:	0
Error:	0
Fatal:	0

List of tests included in report...

List of global attributes included in report...

List of Design Units included in report...

Coverage Summary by Structure:

Design Scope	Hits %	Coverage %
tb_atm	77.27%	85.72%
uut	77.68%	85.99%

Coverage Summary by Type:

Total Coverage:					77.27%	85.72%
Coverage Type	**Bins**	**Hits**	**Misses**	**Weight**	**% Hit**	**Coverage**
Statements	166	165	1	1	99.39%	**99.39%**
Branches	36	35	1	1	97.22%	**97.22%**
Toggles	254	151	103	1	59.44%	**59.44%**
FSMs	28	23	5	1	82.14%	**86.84%**
States	9	9	0	1	100.00%	**100.00%**
Transitions	19	14	5	1	73.68%	**73.68%**

Report generated by Questa (ver. 10.6c) on Saturday 16 December 2023 19:08:41 with command line:
coverage report -html -htmldir covhtmlreport -assert -directive -cvg -code bcefst -threshL 50 -threshH 90

RTL

Ports

```
module atm (
    clk,
    rst,
    card,
    card_no,
    password,
    option,
    language,
    another_service,
    amount,
    invalid_password,
    no_balance,
    invalid_card,
    avalaible_balance
);
```

States

```
27    localparam English = 2'b00;
28    localparam French = 2'b01;
29    localparam German = 2'b10;
30    localparam Italy = 2'b11;
31
32    localparam default_state = 4'b0000;  //0
33    localparam show_language = 4'b0001;  //1
34    localparam request_pin = 4'b0010;  //2
35    localparam show_main_menu_English = 4'b0100;  //4
36    localparam show_balance = 4'b0111;  //7
37    localparam withdraw_money = 4'b1000;  //8
38    localparam deposit_money = 4'b1001;  //9
39    localparam another_service_state = 4'b101;  //10
40    localparam check_card = 4'b1011;  //11
41
42
43    reg invalid_amount_flag;
44    reg found;
45    reg [3:0] current_state;
46    reg [3:0] next_state;
47    reg [1:0] index;
48    reg [3:0] password_reg;
49    reg [7:0] amount_reg;
50    reg [3:0] accounts_password[9:0];
51    reg [7:0] accounts_balances[9:0];
52
```

Data Base

```
initial begin
  avalaible_balance = 8'b00000000;
  accounts_password[0] = 4'b1111;
  accounts_password[1] = 4'b1110;
  accounts_password[2] = 4'b1101;
  accounts_password[3] = 4'b1100;
  accounts_balances[0] = 8'b00000110;
  accounts_balances[1] = 8'b01010101;
  accounts_balances[2] = 8'b01010101;
  accounts_balances[3] = 8'b01010101;
end
```

Transition always Block.

```
always @(posedge clk or posedge rst) begin
  if (rst) begin
    current_state <= default_state;
  end else current_state <= next_state;
end
```

Default State

```
76      always @(*) begin
77        case (current_state)
78          default_state: begin
79            invalid_password = 1'b1;
80            invalid_card = 1'b1;
81            no_balance = 1'b1;
82            next_state = check_card;
83          end
```

Check Card

```
check_card: begin
  if (card) begin
    if (card_no == 3'b000) begin
      invalid_card = 1'b0;
      next_state   = request_pin;
    end else if (card_no == 3'b001) begin
      invalid_card = 1'b0;
      next_state   = request_pin;
    end else if (card_no == 3'b010) begin
      invalid_card = 1'b0;
      next_state   = request_pin;
    end else if (card_no == 3'b011) begin
      invalid_card = 1'b0;
      next_state   = request_pin;
    end else begin
      invalid_card = 1'b1;
      next_state   = default_state;
    end
  end else next_state = default_state;
end
```

Request Pin

```
request_pin: begin
  if (invalid_card == 1'b0) begin
    if (password == accounts_password[card_no]) begin
      next_state = show_language;
      invalid_password = 1'b0;
    end else begin
      next_state = default_state;
      invalid_password = 1'b1;
    end
  end else begin
    next_state = default_state;
  end
end
```

Show Language

```
show_language: begin
  begin
    if (language == English) begin
      next_state = show_main_menu_English;
    end else begin
      next_state = default_state;
    end
  end
end
```

Show Main Menu for English

```
show_main_menu_English: begin
  if (option == 3'b000) begin
    next_state = show_main_menu_English;
  end else if (option == 3'b001) begin
    next_state = deposit_money;
  end else if (option == 3'b010) begin
    next_state = withdraw_money;
  end else if (option == 3'b011) begin
    next_state = show_balance;
  end else begin
    next_state = show_main_menu_English;
  end
end
```

Options in the main Menu

Show Balance

```
show_balance: begin
  avalaible_balance = accounts_balances[card_no];
  next_state = another_service_state;
end
```

Deposit Money

```
deposit_money: begin
  accounts_balances[card_no] = accounts_balances[card_no] + amount;
  next_state = another_service_state;
end
```

Withdrawal Money

```
withdraw_money: begin
  no_balance = 1'b1;
  if (amount > accounts_balances[card_no]) begin
    no_balance = 1'b0;
  end

  if (no_balance == 0) begin
    accounts_balances[card_no] = accounts_balances[card_no] - amount;
    next_state = another_service_state;
  end else begin
    next_state = default_state;
  end
end
```

Another Service

```
    another_service_state: begin
      if (another_service) begin
        next_state = show_main_menu_English;
      end else begin
        next_state = default_state;
      end
    end
  endcase
end
```

Conclusion:

Finally, This project simulates a real-life ATM by which a card is expected from the user to be able to carry out the necessary transactions such as Deposit, withdrawal money, show balance, etc.., in order to modify this account as he desires taking into consideration his previous transactions.

The Verilog implementation, which is based on FPGAs, encapsulates the hardware behavior of the ATM, including card input, transaction processing, and account update. The project transforms high-level functionality into low-level hardware operations via the Verilog description, allowing the system to work at the hardware level.

The C++ reference model, on the other hand, acts as a dependable and adaptable benchmark for the Verilog implementation. The C++ approach, by offering a high-level software abstraction, enables rigorous testing and validation of the system's functioning. To ensure that the ATM behaves correctly, the reference model can run a range of test scenarios, such as depositing monies, withdrawing money, displaying account balances, and updating account information.

The verification procedure is critical for ensuring the correctness and dependability of the ATM system. The establishment of a test bench allows for comprehensive testing, and the use of coverage metrics aids in determining the success of the tests. The project team can discover areas that may require more testing by carefully reviewing the coverage %, ensuring a more thorough test, ensuring a more robust and dependable ATM system.