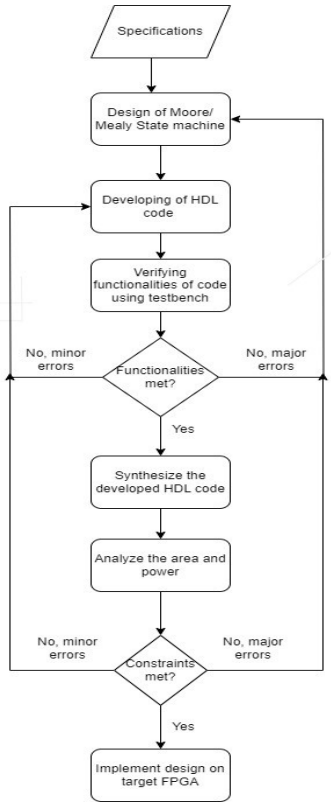
Intel Unnati Industrial Training SUMMER 2023

Title: DESIGN AND IMPLEMENTATION OF ATM(FSM) CONTROLLER
Team members: Aayush pandey,Bhargava P,Berwyn suhas A(team-proffesionals)
Institution: Nitte Meenakshi institute of technology
Institute Mentor: Dr. Rajesh N. **Industry Mentor**: Mr Abhishek Nandy
Date: 14 July 2023

1. Introduction

The layout of modern banking systems must include an Automated Teller Machine (ATM). Users using an ATM can do a variety of financial transactions without the help of a bank personnel, including cash withdrawals, deposits, balance inquiries, and fund transfers. In this project, a finite state machine (FSM) for an ATM system was built using the hardware description language Verilog. In this project, a Verilog-based Automated Teller Machine (FSM) Controller is designed and put into operation. Verilog is a widely used hardware description language in the realm of digital design. The main goal of this project is to create a reliable and effective controller that closely resembles the behavior of an ATM, enabling the efficient execution of its essential features such card authentication, transaction processing ,face recognition, receipt generation etc.



2. Literature survey

The literature survey identifies the most important sources that are pertinent to the project. The chosen papers, articles, blogs, and tutorials offer information on creating and using FSMs, comprehending Verilog syntax and modelling methods, enhancing performance, and verifying intricate designs. The project's understanding of ATM controller architecture, FSM implementation, and overall system reliability can be improved by incorporating these insights.

Textbook: "Verilog HDL: A Practical Primer" by J. Bhasker

Verilog HDL and its practical applications are thoroughly covered in this book. The syntax, data types, operators, modules, and behavioural modelling of Verilog are all covered. Advanced subjects including system tasks and functions, testbenches, and synthesis considerations are also covered . An organised and thorough learning environment for Verilog HDL is provided covering the key Verilog concepts that aid in developing a thorough understanding of Verilog programming, which is required for building the ATM controller successfully.

NPTEL Video Lecture Series: "Digital System Design" by Prof. S. Srinivasan

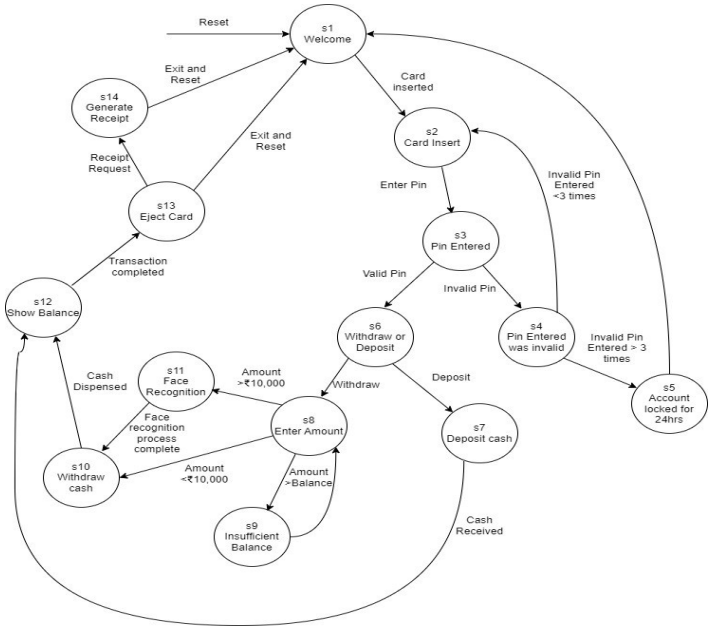
This NPTEL video lecture series provides a comprehensive overview of digital system design, including topics such as combinational and sequential circuits, FSMs, and HDLs like Verilog. The lectures delve into practical implementation techniques, design methodologies, and verification strategies. The NPTEL video lectures offer an extensive resource for understanding digital system design concepts and their practical application using Verilog. The lectures can provide in-depth knowledge on FSMs, hardware description languages, and effective design methodologies, which are directly relevant to the project.

Article: "Designing Finite State Machines for Intel FPGAs"

This article focuses on designing Finite State Machines (FSMs) specifically for Intel FPGAs. It discusses various aspects of FSM design, including state encoding techniques, state transition logic, output generation, and optimization strategies. The article also highlights tools and features available in Intel FPGA development software for FSM design and analysis. As we are using Verilog for implementing the ATM controller on an FPGA, this article provides valuable guidance specific to Intel FPGAs. It covers important considerations and optimization techniques that can be applied to our FSM design using Intel FPGA development tools.

3. Finite State Machine Design

The design of the FSM for the ATM system involves defining the states, inputs, outputs, and transitions. The FSM implementation can be done using mealy or moore technique.For this project we have choosen the moore implementation and the FSM for the ATM controller is given below.



States:

1. Welcome(s1): The initial state of the ATM system, waiting for a user to insert a bank card.
2. Card Inserted(s2): The state after the user has inserted a valid bank card.
3. PIN Entered(s3): The state after the user has entered a valid Personal Identification Number (PIN).
4. Valid PIN(s4):The state if the pin entered was a valid pin.
5. Account lock(s5):If the invalid pin was entered greater than 3 times this state locks the account.
6. Withdraw or deposit(s6): The state where the user selects the desired transaction (e.g., withdrawal, deposit).
7. Deposit (s7): The state where the amount is deposited to the account.
8. Enter amount(s8): The state in which the user enters the amount.
9. Insufficient balance(s9):If the entered amount is greater than the balance.
10. Withdraw cash(s10):the state in which cash is withdrawn and old and new balance is displayed.
11. Face recognition(s11):If the enterd amount is greater than 10,000 face recognition would be performed.
12. Show balance(s12):the state which displays the balance left in the account.
13. Eject card(s13):the state in which the transaction is completed and card is ejected.
14. Generate receipt(s14):the state in which the receipt is generated after the transaction is completed.

4. Description of the code:

The code starts by specifying **threshold**, which identifies the amount (for this project, Rs 10,000) that must be withdrawn before face recognition will be used to verify identity. For the purpose of conveniently encoding boolean values, it also defines **true** and **false**.

The main module, "atm," is responsible for implementing the FSM controller for the ATM. It takes the inputs **clk** (clock), **reset**, **cardnumber**, **pin**, **amount**, **withdraw**, **reciept_req**, and provides outputs **remaining_balance** and **LED** (used to display the state on the FPGA and it is just the number assigned to each state.)

The FSM states are defined using parameters **welcome**, **card_inserted**, **pin_entered**,etc. Each state represents a specific phase or operation of the ATM.

The FSM state transitions are described inside the always block, which activates on the positive edge of the clock signal (**posedge clk**). The code predicts the subsequent state based on the input conditions and the current state, and it also displays the results of each state.The FSM's state is stored in the register **state**.

The card numbers, related PINs, and account balances for various accounts are stored in the **card_database**, **pin_database,** and **balance** arrays. These arrays are initialised by the code with example information for three accounts.

In the initial state, "welcome," the code displays a message asking the user to insert their card and checks if a **cardnumber** is received. If no card number is detected (**!cardnumber**), an error message is displayed, and the FSM remains in the "welcome" state. If a card number is detected, the code searches for a match in the **card_database** array. If a match is found, the FSM transitions to the "card_inserted" state. Otherwise, an error message is displayed, and the FSM returns to the "welcome" state.

In the "card_inserted" state, the code prompts the user to enter their PIN. If a **pin** is entered, the FSM transitions to the "pin_entered" state. Otherwise, it remains in the "card_inserted" state.

In the "pin_entered" state, the code determines whether the PIN entered matches the PIN for the relevant card number that is recorded in the **pin_database** array. The FSM changes to the "withdraw_deposit" state if the PIN is valid. If not, a warning message appears and the FSM enters the "invalid_pin" state.

In the "invalid_pin" state, The code increases the count variable(increment by 1), which reflects the total number of unsuccessful PIN attempts. The FSM switches to the "account_lock" state if the count reaches or exceeds 3. The FSM returns to the "card_inserted" state if the count is less than 3.

The display message stating that the account is locked for 24 hours is displayed by the "account_lock" state. This state Sets the lock varible to true which locks the account for 24 hours. After then, the FSM returns to the "welcome" state.

The "withdraw_deposit" state prompts the user to choose between withdrawal or deposit. If the **withdraw** signal is high, the FSM transitions to the "enter_amount" state. Otherwise, it moves to the "deposit" state.

In the "deposit" state, the code prompts the user to enter the amount to be deposited (**amount**). The deposited amount is stored in the **dep_amount** variable, and a success message is displayed. The FSM then transitions to the "show_balance" state.

The user is prompted to enter the amount to be withheld when the state is "enter_amount". The function determines whether the amount is higher than the account's balance (**balance[acc]).** The FSM enters the "insufficient_funds" state if it is. The FSM enters the "face_recognition" state if the **amount** is greater than the specified **threshold** but less than the account balance. It switches to the "withdraw_cash" state if not.The "insufficient_funds" state displays a message indicating that there are insufficient funds. The FSM then returns to the "enter_amount" state.

In the "withdraw_cash" state, a success message is displayed which says that the cash was withdrawn successfully. The FSM transitions to the "show_balance" state.

The "face_recognition" state displays a message indicating that face recognition will be performed since the withdrawal amount exceeds the defined threshold. The FSM then transitions to the "withdraw_cash" state.

In the "show_balance" state, the code displays the old account balance (**balance[acc]**). If it is a withdrawal, the **remaining_balance** is updated by subtracting the withdrawal amount from the balance. If it is a deposit, the **remaining_balance** is updated by adding the deposited amount. The new balance is displayed, and the FSM transitions to the "eject_card" state.

In the "eject_card" state, a message indicating that the card is ejected is displayed. If **reciept_req** is high, the FSM transitions to the "generate_reciept" state. Otherwise, it returns to the "welcome" state.

The "generate_reciept" state displays a receipt message, including the card number (**cardnumber**) and other details like total amount withdrawn or deposited ,the balance left.either the total amount. The FSM then transitions back to the "welcome" state.

5. FPGA implementation

For this project we are implementing the Verilog code on the cyclone V FPGA board using a virtual board provided by labsland.as the number of pins in the FPGA are limited we have designed two versions of the code. The first version is used for demonstrating the ATM FSM with the given parameters which were specified in the problem statement like a threshold of 10,000,card numbers, etc. The first version exceeds the given number of pins on the board hence we designed a second version which is same as the first version but with lesser number of bits and we removed some additional functions.

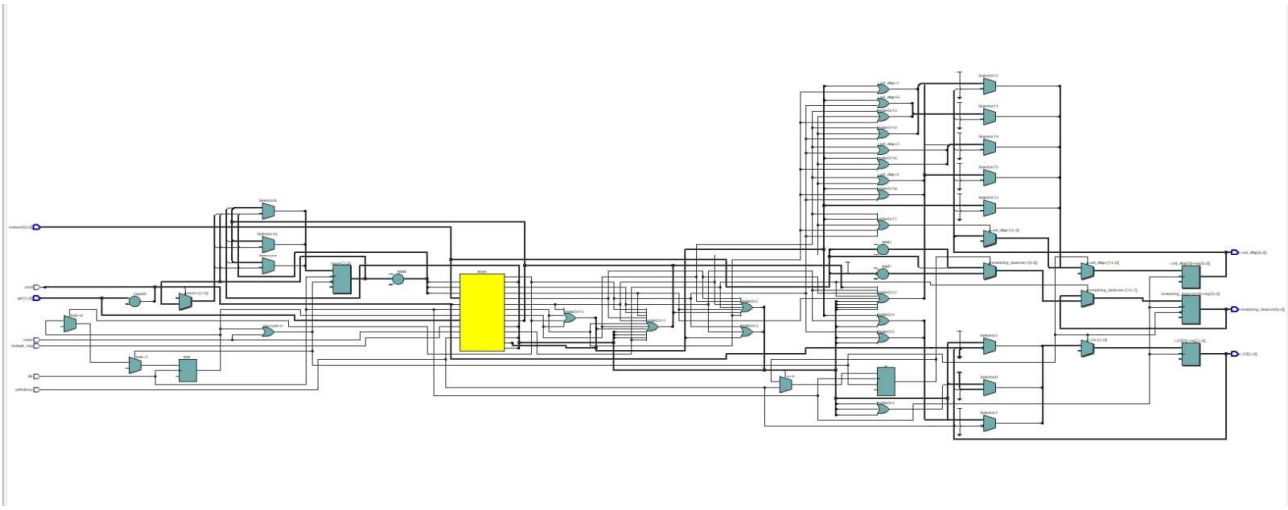
6. Results

Transcript

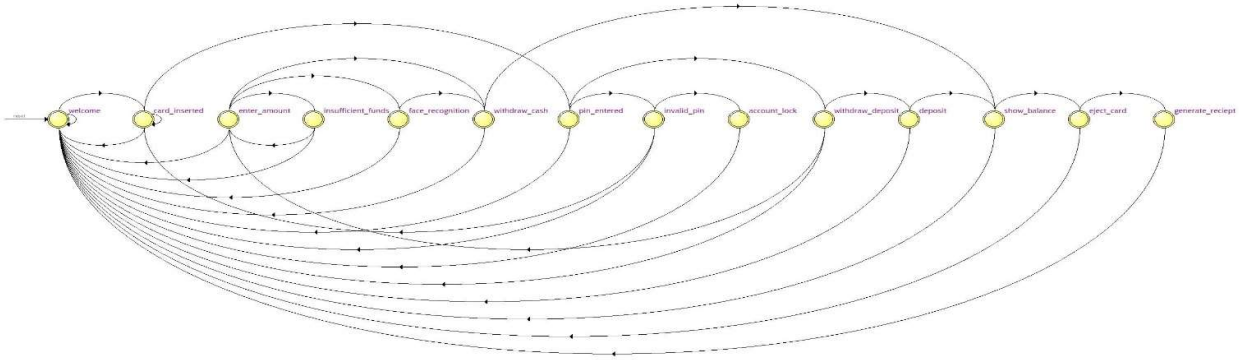
 # please insert your card(enter card number)
 # error in getting card number try again
 # please insert your card(enter card number)
 # Card number 2136 not found!
 # please insert your card(enter card number)
 # please enter the pin
 # pin entered was invalid please try again
 # pin error try 1
 # please enter the pin
 # pin was entered succesfully
 # do you want to withdraw or deposit the money?

 # enter the amount to be withdrawn
 # the cash was withdrawn succesfully
 # the old balance was 13000
 # the balance now is 12000
 # the card is ejected
 # reciept
 # card number:2133
 # the total amount withdrawn: 1000
 # the total balance left is:12000
 # please insert your card(enter card number)
 # please enter the pin
 # pin was entered succesfully
 # do you want to withdraw or deposit the money?
 # enter the amount to be deposited
 # the money was deposited succesfully
 # the old balance was 15000
 # the balance now is 17000
 # the card is ejected
 # reciept
 # card number:1234
 # the total amount deposited: 2000
 # the total balance left is:17000
 # please insert your card(enter card number)
 # please enter the pin
 # pin was entered succesfully
 # do you want to withdraw or deposit the money?
 # enter the amount to be withdrawn
 # the entered amount is greater than 10,000 face recognition will be performed
 # the cash was withdrawn succesfully
 # the old balance was 13000
 # the balance now is 2500
 # the card is ejected
 # reciept
 # card number:2133
 # the total amount withdrawn:10500
 # the total balance left is: 2500
 # please insert your card(enter card number)
 # please enter the pin
 # pin was entered succesfully
 # do you want to withdraw or deposit the money?
 # enter the amount to be withdrawn
 # not enough balance!try again
 # enter the amount to be withdrawn
 # the cash was withdrawn succesfully
 # the old balance was 12000
 # the balance now is 9500
 # the card is ejected
 # reciept
 # card number:1556
 # the total amount withdrawn: 2500
 # the total balance left is: 9500
 # please insert your card(enter card number)
 # please enter the pin
 # pin entered was invalid please try again
 # pin error try 1
 # please enter the pin
 # pin entered was invalid please try again
 # pin error try 2
 # please enter the pin
 # pin entered was invalid please try again
 # pin error try 3
 # your account 1234 is locked for the next 24hours

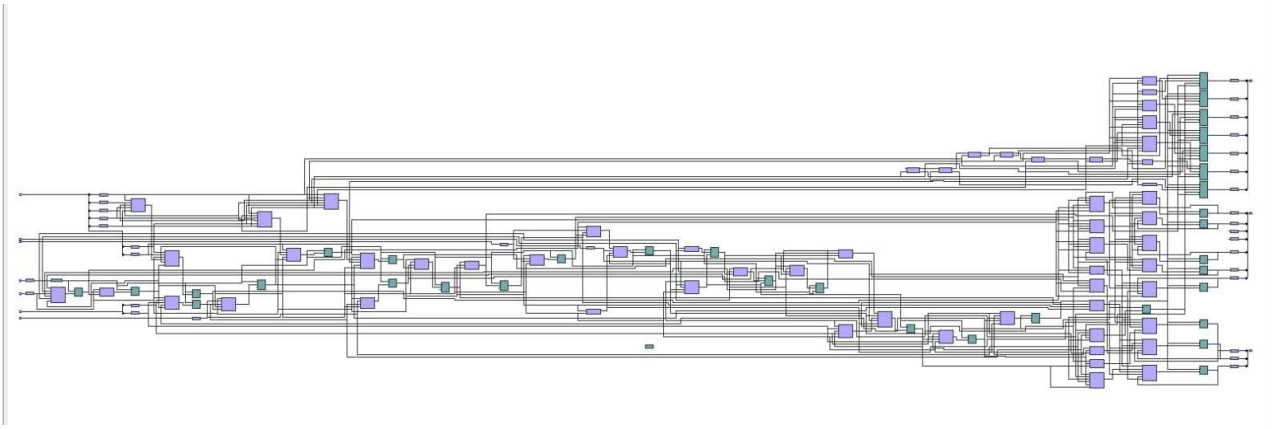
RTL schematic



FSM generated by the software



post fitting layout



Power report:

Power Analyzer Status : Successful - Fri Jul 14 11:25:35 2023

Quartus Prime Version : 18.1.0 Build 625 09/12/2018 SJ Lite Edition

Revision Name : atm6

Top-level Entity Name : atm6

Family : Cyclone V

Device : 5CSEMA5F31C6

Power Models : Final

Total Thermal Power Dissipation : 421.36 mW

Core Dynamic Thermal Power Dissipation : 0.00 mW

Core Static Thermal Power Dissipation : 411.23 mW

I/O Thermal Power Dissipation : 10.13 mW

Power Estimation Confidence : Low: user provided insufficient toggle rate data

Area report:

```
+-------------------------------------------------------------------------------+
; Fitter/Area Summary                                                           ;
+---------------------------------+---------------------------------------------+
; Fitter Status                  ; Successful - Fri Jul 14 11:22:07 2023       ;
; Quartus Prime Version          ; 18.1.0 Build 625 09/12/2018 SJ Lite Edition ;
; Revision Name                  ; atm6                                        ;
; Top-level Entity Name          ; atm6                                        ;
; Family                         ; Cyclone V                                   ;
; Device                         ; 5CSEMA5F31C6                                ;
; Timing Models                  ; Final                                       ;
; Logic utilization (in ALMs)    ; 37 / 32,070 ( < 1 % )                       ;
; Total registers                ; 34                                          ;
; Total pins                     ; 31 / 457 ( 7 % )                            ;
; Total virtual pins             ; 0                                           ;
; Total block memory bits        ; 0 / 4,065,280 ( 0 % )                       ;
; Total RAM Blocks               ; 0 / 397 ( 0 % )                             ;
; Total DSP Blocks               ; 0 / 87 ( 0 % )                              ;
; Total HSSI RX PCSs             ; 0                                           ;
; Total HSSI PMA RX Deserializers ; 0                                          ;
; Total HSSI TX PCSs             ; 0                                           ;
; Total HSSI PMA TX Serializers  ; 0                                           ;
; Total PLLs                     ; 0 / 6 ( 0 % )                               ;
; Total DLLs                     ; 0 / 4 ( 0 % )                               ;
+---------------------------------+---------------------------------------------+
```

7. References

https://www.youtube.com/watch?v=CeD2L6KbtVM&list=PL803563859BF7ED8C&index=1

1.6.4.2. Verilog HDL State Machines (intel.com)

8. Screenshots