NITTE MEENAKSHI INSTITUTE OF TECHNOLOGY

A Project On

Design and Implementation of Automated Teller Machine (FSM) Controller

Submitted by

TEAM ACE

Adithya T - 1NT20EC006

Email: 1nt20ec006.adithya@nmit.ac.in

Sufiyan Ahmed Khan - 1NT20EC153

Email: 1nt20ec153.sufiyan@nmit.ac.in

Under the Supervision of

Dr. RAJESH N

College Mentor

Nitte Meenakshi Institute of Technology

Email: rajesh.n@nmit.ac.in

Problem Statement: To design and implement an ATM(FSM) controller using Intel Quartus Prime Lite Edition.

Introduction: Automated Teller Machine is a device which is used for money transaction without the need to visit the bank for such transactions. This application is very useful when we have to just deposit or withdraw small amount of cash. But to implement this concept in an FPGA board we can use a technique called Finite State Machine (FSM). FSM consists of state in which a particular task is performed which can be unique or same as another state. We approach this problem by designing a Finite State Machine (FSM) according to the requirements provided by the establishment. Then we proceed to code the ATM accordingly with FSM designed in system Verilog language using the tool provided by intel, FPGA Design Software - Intel® Quartus® Prime.

The Finite State Machine designed will control the whole ATM machine by identifying inputs and authenticating them. Then it provides the output signal for valid input statements which enables the ATM machine to work in the optimal and secure manner.

Block Diagram:

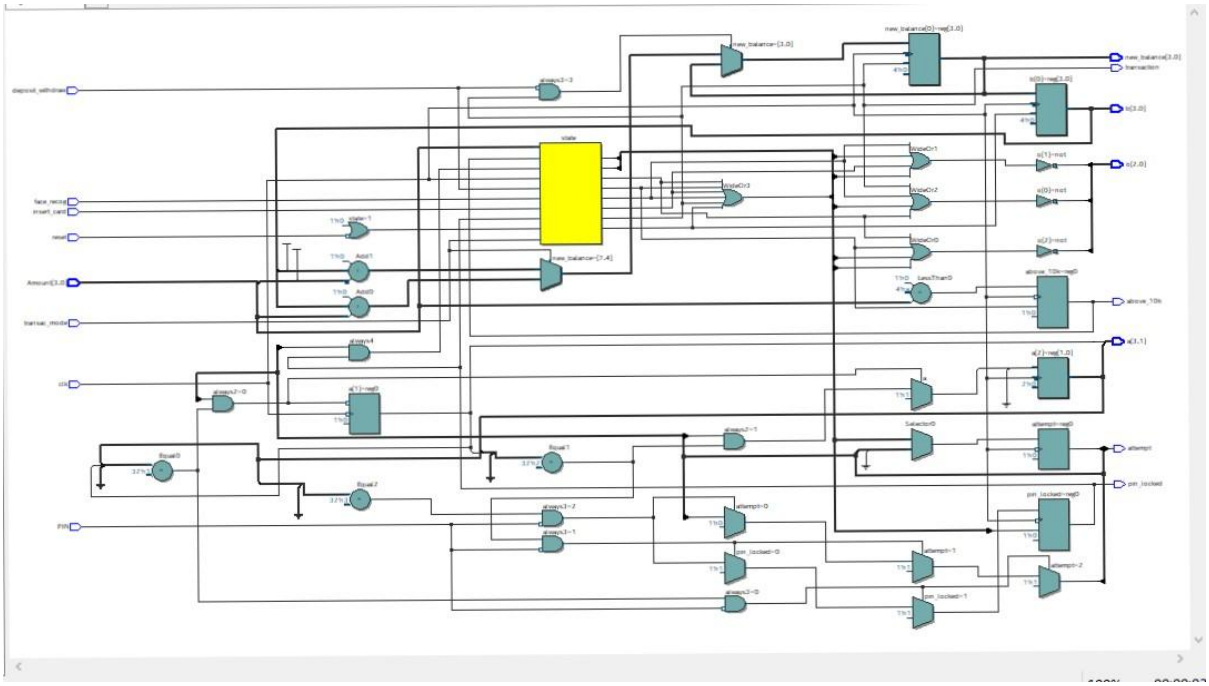

Fig:1

The above figure shows the block diagram of FSM code that we have implemented in the software and the above figure is generated from RTL viewer. The above block diagram shows us the different combinational blocks that has been used and also the registers and flip-flops that defines the state transitions and inputs and outputs that govern these transitions from one state to another. We also observe the usage of buffer circuits to maintain the signal strength of any network or wire connection thereby enhancing its performance and reduce the chances of error. Some of the connections are provided with direct ground or VDD connection. This is to provide a strong logic ‘0’ or ‘1’ to a digital logic circuit such as multiplexer and flip-flops.

FSM Model:

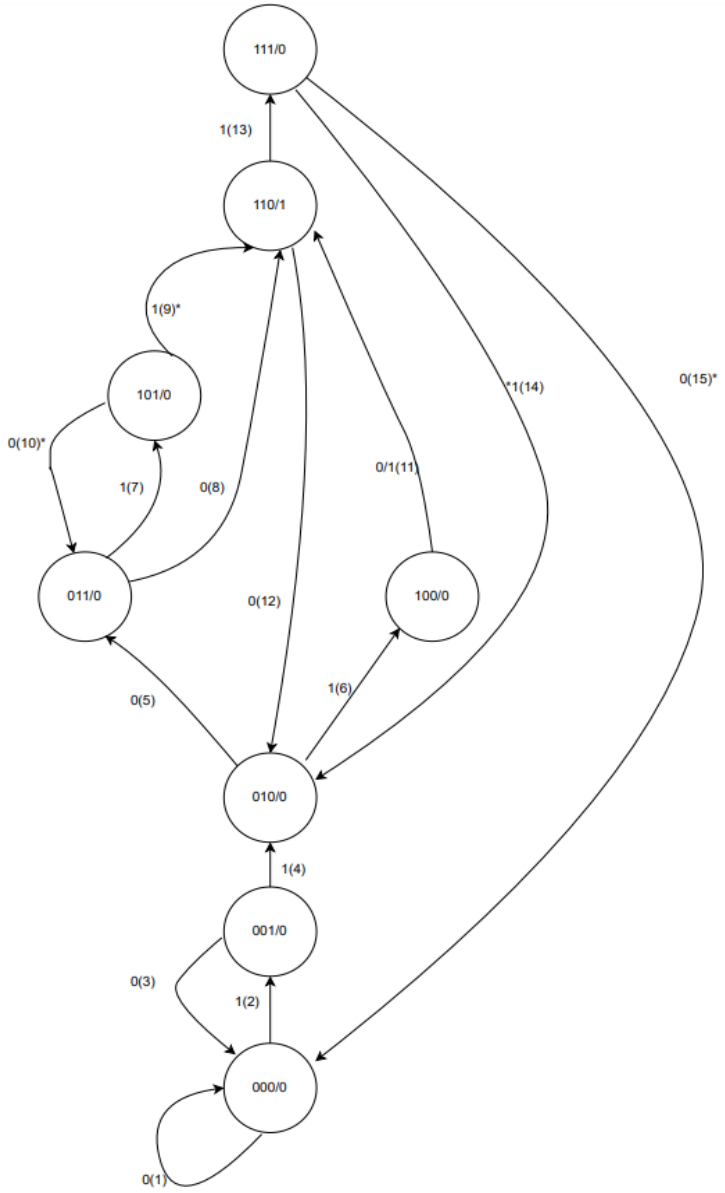

Fig: 2

The above diagram represents the Moore state machine FSM which is used to implement the ATM Controller Machine. A moore FSM is an state machine where transition of states happen based on input values. The output is defined in the state itself and it is independent of input value. The different states and their implementations are as follows:

State 0: It is represented as 000 and has an output of 0.This state functions as a main screen display which is seen in any ATM. When input is logic '0' it remains in same state and when input is logic '1' it moves to the next state.

State 1: It is represented as 001 and has an output of 0. This state functions as a Pin entry and verification display to verify the customer.When input is logic '1' it goes to next state and when input is logic '0' it moves to the previous state.

State 2: It is represented as 010 and has an output of 0. This state functions as a screen display where customer can choose to either deposit or withdraw. When input is logic '0' it goes to state 3 and when input is logic '1' it moves to the state 4.

State 3: It is represented as 011 and has an output of 0.If customer chooses to withdraw he enters this state. When input is logic '0' it goes to state 5 and when input is logic '1' it moves to the state 6.

State 4: It is represented as 100 and has an output of 0. If customer chooses to deposit he enters this state. When input is logic '0' it goes to state 6 and when input is logic '1' it moves to the state 6 as well.

State 5: It is represented as 101 and has an output of 0.If customer has to perform face recognition process he enters this state. When input is logic '0' it goes to state 3 and when input is logic '1' it moves to the state 6.

State 6: It is represented as 110 and has an output of 1. In this state the balance of customers account is updated according to his transaction mode. When input is logic '0' it goes to state 2 and when input is logic '1' it moves to the state 7.

State 7: It is represented as 111 and has an output of 0.Here the customer receives a mini-statement of his transaction and can choose to remove card or make another transaction. When input is logic '0' it goes to state 0 and when input is logic '1' it moves to the state 2.

Literature Survey:

A literature survey reveals valuable resources in the field of digital circuits and systems. "Field Programmable Gate Array Technology" by S. Trimberger (1994, Kluwer Academic Publications) provides a detailed perspective on FPGA programming and reprogramming, covering architecture, design principles, programming techniques, and practical applications. Additionally, "Finite State Machine Datapath Design, Optimization, and Implementation" (Morgan and Claypool Publishers, 2008) offers comprehensive insights into efficient FSM implementation, optimization, and design. Lastly, "System Verilog for Design: A Guide to Using System Verilog for Hardware Design and Modeling" by Stuart Sutherland, Simon Davidmann, and Peter Flake (2nd Edition, Springer) serves as an essential resource, providing invaluable insights into programming FPGAs using System Verilog for hardware designers and engineers seeking to enhance their design and modeling skills.

Approach:

By studying the above mentioned textbooks and references, we have come up with a plan to build an ATM Controller that has a Pin Entry and Verification feature where the user can have maximum of 3 attempts, after which his account will be locked for 24 hours. Usually, the common approach is to make a state for each attempt made by the user which is unnecessary and will take more states. Hence, we will be trying to create a mechanism where for every attempt the user will remain in same state and then goes to main state and there we will display that his account is locked for 24 hours. If pin entered is corrected he can move to next state where the user can either choose to deposit or withdraw cash up to 10,000 rupees for each transaction. If the withdrawal is more than 10,000 rupees a face recognition test to be conducted, if this test is successful further transaction is allowed otherwise he has to choose a transaction less than 10,000 rupees. Since we can keep a separate interface for face recognition, there is no need to add it in our FSM. Hence we are taking face recognition as an input from the interface which is used for transitioning of states Then we have to provide an

acknowledgement that transaction is successful or failed and also provide with a mini-statement at the end. Then ask the customer whether he would like to make another transaction or not.

Methodology:

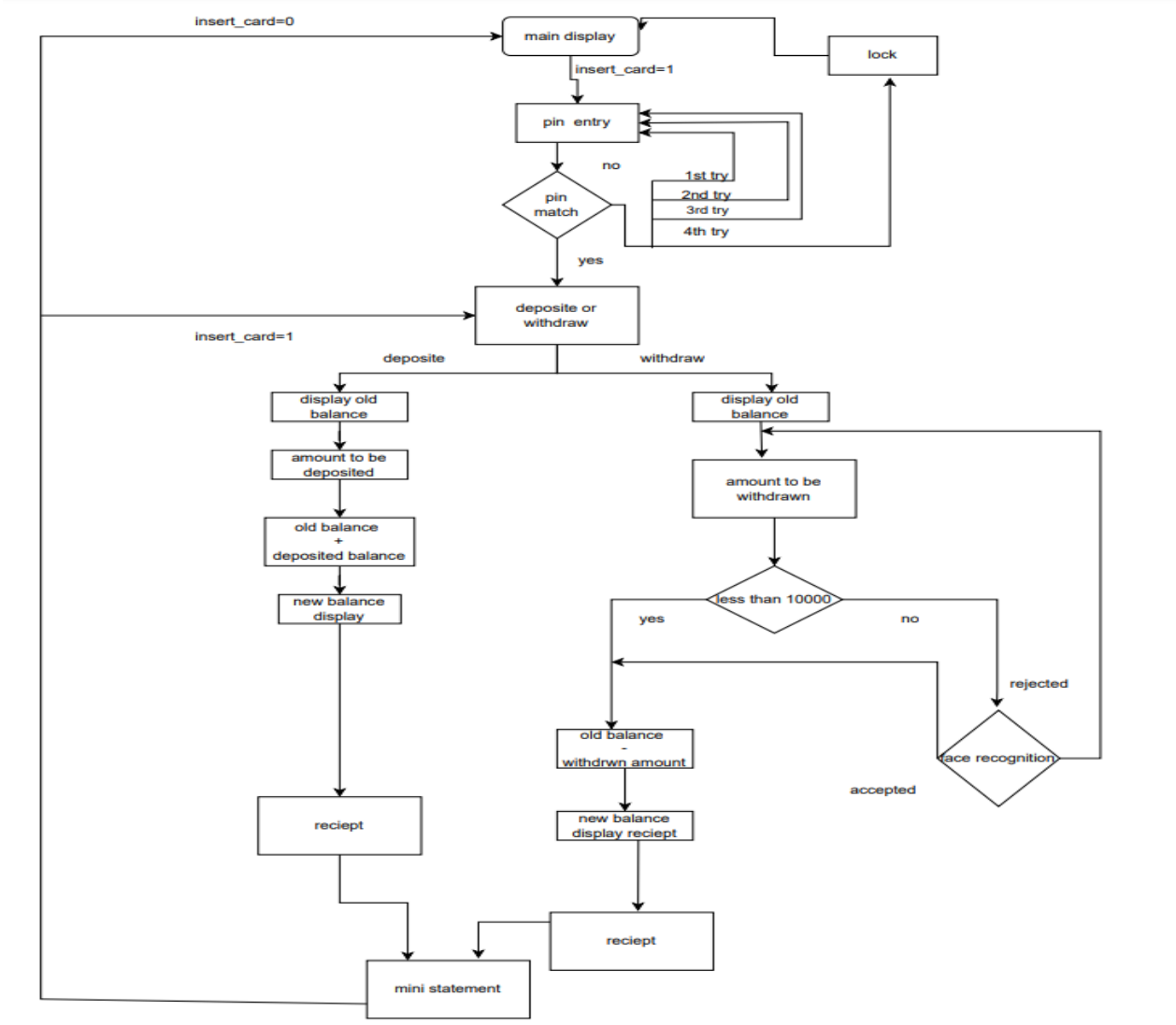

Fig: 3

When it comes to implementing the ATM Controller using FSM technique we have consider factors such as Area constraints and also power constraints. By reducing the number of states we can also reduce the registers and flip-flops required for its working. Hence we have implemented and FSM with 8 states performing all the basic functionalities of an ATM. We are using System Verilog language to code the FSM because it is user-friendly language and has many C- language functions. The main functions present in System Verilog are enumerated types used for defining the states, always procedural blocks such as always_comb which requires very less circuitry compared to the always block statement and unique case statements which functions for floating (Z) and unknown (X) value variables and conditions as well. The first state is the Main Screen Display where the user is directed to if he has accessed the machine for the first time and hasn't entered a card. This state is also used to display that the user ID is locked when his pin entered is invalid. Next he goes to the

Pin Entry and Verification state where he has to enter the correct pin within three attempts. If his verification is successful he enters the next state where he has to choose his transaction mode. When the customer chooses to deposit amount to his account he will be directed to the deposit state where he will be updated about his current balance. If the customer decides to choose withdrawal, he is directed to the withdraw state where he has to enter amount to withdraw. If the withdrawal is more than 10,000 rupees he has to go to face recognition state. But since we aren't implementing it we are assuming it as an input, where if logic '1' is returned we assume face recognition is successful and he goes to next state where he can perform the transaction. Else he is directed back to withdrawal state where he is supposed to choose amount less than 10,000 rupees. Once the user enters the Balance state, his transaction will be completed and he will receive an update of the same. Then he will be directed to next state where he will receive mini-statement of his transaction and can either choose to remove card or make another transaction. If transaction fails he is directed back to the state where he has to choose his transaction mode again.

Implementation in software

Quartus Prime is a sophisticated software suite specifically developed for FPGA and CPLD design and programming. It encompasses several crucial components that play vital roles in the design workflow. Analysis and synthesis are fundamental steps that help optimize and transform the design from a high-level RTL description into gate-level netlists. This process involves checking for any syntax errors and applying various optimizations to enhance the design's performance and efficiency. The RTL Simulator, known as ModelSim, serves as a powerful tool integrated into Quartus Prime, enabling designers to simulate and verify the functionality of their digital designs. By creating testbenches and applying stimuli, designers can observe and analyse the behaviour of their designs, leveraging the waveform viewer for detailed debugging and analysis. The Pin Planner/Fitter tool in Quartus Prime facilitates the assignment of input/output pins for the FPGA or CPLD device. Designers can visually define the connections between signals in their design and the physical pins of the target device. Additionally, constraints such as maximum operating frequencies and I/O voltage levels can be specified. The Pin Planner/Fitter then optimizes the placement and routing of the design, ensuring proper connectivity and adhering to the specified constraints. Finally, Quartus Prime enables the generation of a SOF (SRAM Object File) that contains crucial configuration data for programming the FPGA or CPLD device. This file encapsulates vital information about logic connections, memory contents and specific settings for the design. The generated SOF file can be utilised with Quartus Programmer or external programming hardware to accurately program and initialise the device with the desired design. In summary, Quartus Prime provides a comprehensive set of sophisticated tools and features that empower designers to efficiently design, simulate, analyse, and programme FPGAs and CPLDs, ultimately enabling the creation of advanced digital systems.

Result and Summary:

AREA REPORT:

RESOURCE	USAGE
ESTIMATION OF LOGIC UTILIZATION	
Combinational ALUT usage for logic	26
-- 7 input functions	0
-- 6 input functions	5
-- 5 input functions	3
-- 4 input functions	5
-- <=3 input functions	13
Dedicated logic registers	21
I/O pins	29
Total DSP Blocks	0
Maximum fan-out node	~clk input
Maximum fan-out	21
Total fan-out	202
Average fan-out	1.92

Fig: 5

From above report, we get all the information regarding the number of registers and combinational logics used . The more states used in the FSM the higher the usage of logic circuits and blocks.
The next table as shown in fig:6 tells us about the power dissipation and other factors affecting the FSM working and functionality.

POWER REPORT:

POWER ANALYSER SUMMARY	STATUS
Power Analyzer Status	SUCCESSFUL
Quartus Prime Version	18.1.0 Build 625 09/12/2018 SJ Lite Edition
Revision Name	FSM
Top-level Entity Name	FSM
Family	Cyclone V
Device	5CSEMA5F31C6
Power Models	Final
Total Thermal Power Dissipation	421.14 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	411.23 mW
I/O Thermal Power Dissipation	9.91 mW
Power Estimation Confidence	Low: user provided insufficient toggle rate data

Fig: 6

Github file upload Screenshot:

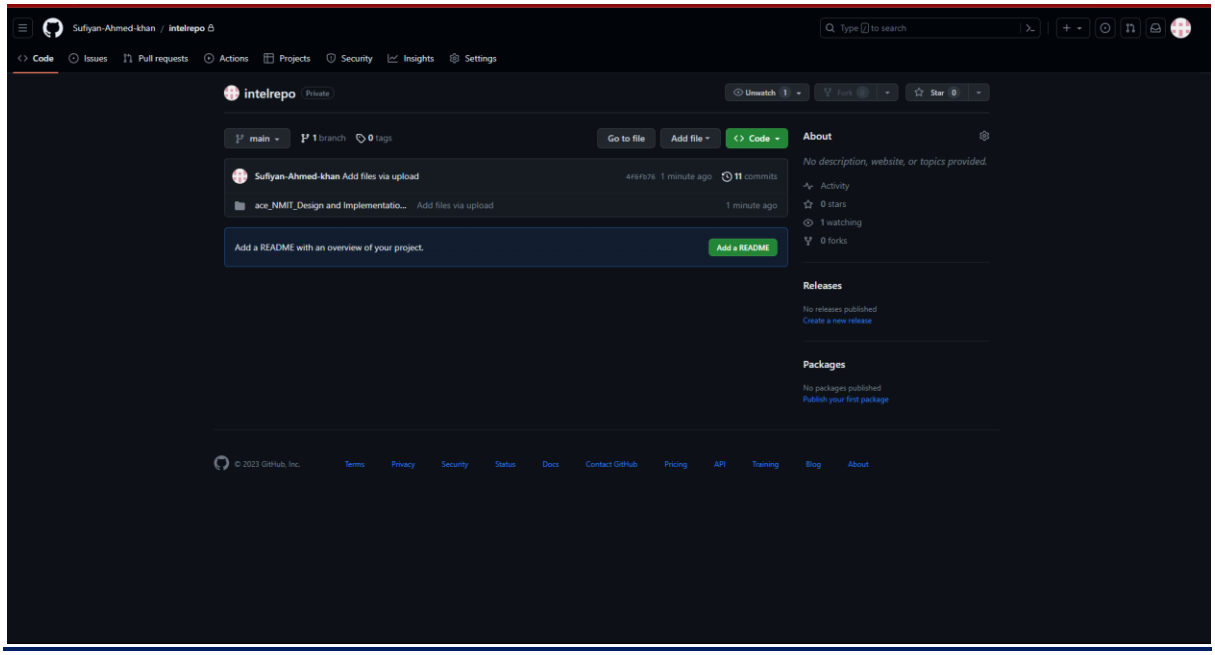

Fig: 4

The picture added above is the screenshot of the Github link generation and the folder with all the information regarding the projects have been uploaded there.

RESULT:

Hence, we have implemented an ATM controller machine using the concept of FSM. This FSM is now capable of checking pin verification and transit appropriately to respective states. Since we have used System Verilog language which is easy to code with the constraints such as power and area constraints, building an FSM with less number of states and less power consumption is possible.