Table of Contents

List of Figures and Tables

List of Abbreviations, Acronyms & Terms Used

ATM is used for Automated Teller Machines

FPGA is used for Field Programmable Gate Array

FSM is used for Finite State Machine

PIN is used for Personal Identification Number

HDL is used for Hardware Description Language

DSD is used for Digital System Design

ISE is used for Integrated Software Environment

PAF is used for Pakistan Air Force

LED is used for Light Emitting Diode

ASM is used for Algorithmic State Machine

1 – Abstract

ATM (automated teller machine) is a very essential tool required for the society in order to facilitate the need of safe transaction of money. In the growing technological world, people want their work to be very simple in order to save time. As we know some new technology is becoming very popular in banking sector which is referred as ATMs. A finite state machine is used to model a simple ATM in this project. The developed design is modeled using Verilog HDL language which hardware description language. The simulation is carried out using Xilinx tool and then it is burned on FPGA.

Keywords

FPGA, HDL, ATM, FPGA Spartan 3 development board

2 – Introduction

2.1 – Background

We had worked on ATM in 5th semester in our semester project. We implemented ATM on Visual Studio 2010 using its feature Windows Form Application and then connected that with a database that was developed on MySql. When choosing project for DSD, we choose ATM implementation as we had already work on this topic and want to continue it. Also FPGA is a very hot topic in fields these days and it has good scope.

2.2 – History

The ATM was invented by Scot John Shepherd-Barron. The world's first ATM was installed in a branch of Barclays in the northern London borough of Enfield, Middlesex, in 1967. Inspiration had struck Mr. Shepherd-Barron, now 82, while he was in the bath. A mechanical cash dispenser was developed and built by Luther George Simjian and installed in 1939 in New York City by the City Bank of New York. The first person to use this machine was Reg Varney of “On the Buses” frame a British Television program from the 1960s.The idea of a PIN stored on the card was developed by a British Engineer John Rose in 1965. The modern networked ATM was invented in Dallas, Texas, by Don Wetzel in 1968.

Figure 2.1 Reg Varney using the first ATM in 1967

2.3 - Significance of Work

We have implemented ATM on FPGA using FSM on Xilinx ISE. We are learning DSD as our semester subject and FSM is a very important part of this field. By working on this project practically we are able to understand FSM and FPGAs better.
We have also do search work for this project which enhance our knowledge about FPGAs more clearly.

2.4 – Scope

This project is implemented on Xilinx and burned on FPGA. FPGAs are an ideal fit for many different markets due to their programmable nature. Field-programmable gate array (FPGA) technology continues to gain momentum, and the worldwide FPGA market is expected to grow to $3.5 billion USD by 2015.Also in Pakistan FPGAs are used in PAF. They have a high scope in market today.
When we talk about ATMs, we all know how important they are in our daily life as they provide us 24 hours service any time anywhere.

2.5– Limitations

When we started this project we tried to implement ATM as real ATM works but when burning on FPGA we faced some problems and limitations.

- Firstly, we can’t show output on FPGA or on Xilinx simulator as user saw output on a real ATM or as we showed ATM output in our previous project on Visual Studio as VS shows graphical output. We can only show output on Xilinx in numbers and on FPGA using LEDs.
- FPGA has only 16 input switches as shown in fig 2.2 and output LEDs which mean all of my project inputs including clock and rest can only be 16 bits or less than 16 bits, same with my outputs, that is why in this project my all inputs are of 1 or 2 bits. Amount and PIN that should be 15 and 4 bits respectively are chosen to be 2 bits for this project.
- During simulation we faced problem in states transitions.
- ATM implementation for such inputs and outputs are done in field but as we are have lack of inputs that’s why we implement the prototype of ATM.

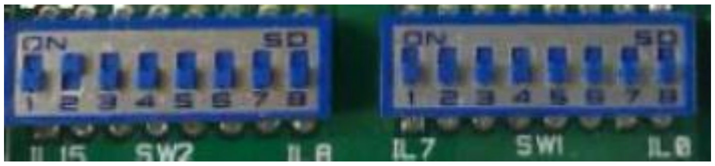


Figure 2.2 16 input switches

3 - Solution

3.1 – Description

We use Xilinx ISE tool to code for ATM. Technique used to code ATM is FSM and language is Verilog HDL. Simple ATM works here that is initially control is on idle state that we will show on FPGA using a LED, then if user insert card control goes to next state (scan_card) and goes on checking the conditions and move control from state to state and showing output using LEDs. Flow Chart for the design is shown in fig 3.1.

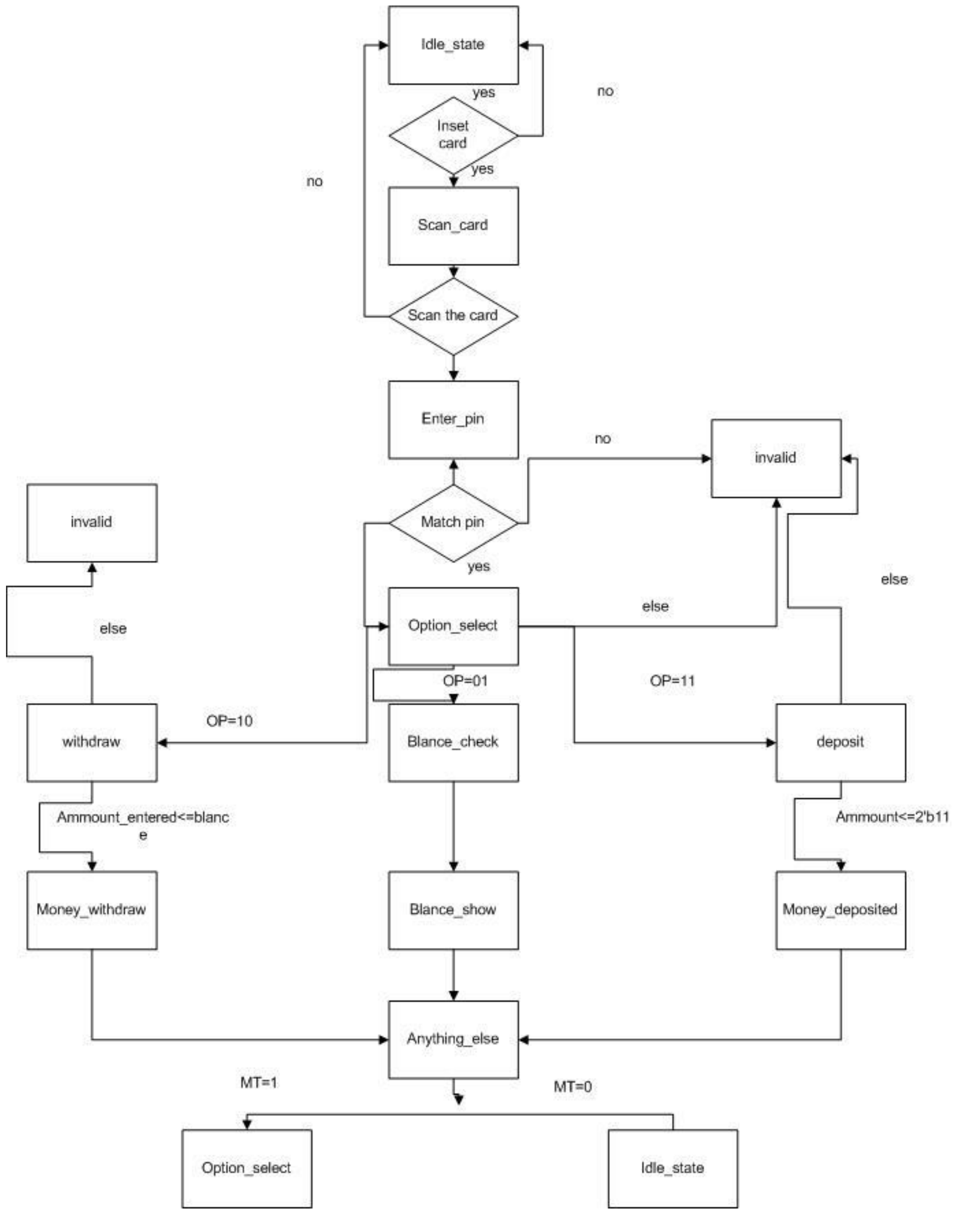


Figure 3.1 ASM diagram for ATM

3.2 – Detailed Design

3.2.1 -Moore Machine State Diagram for ATM

Basically FSM is a representation of the different transition taken place in a system. A state machine is an effective way to implement the control functions. The Moore Machine for the overall ATM controller is represented in fig 3.2. The description of state code and state input signal are tabulated in table 3.1 and table 3.2.

Operation:

Initially the system is in idle state and when it is ready to operate, S0 state is selected which represents the scan card. When the user gets the card inserted, scan line goes high which represents the next state S1. In state S2 user allowed to enter the pin. When the invalid pin is entered, signal will again goes to invalid state S4.

When the valid pin is entered i.e. pin get matched, control signal will lead to the state S3 in which user can select the option.

There are three types of transaction options can be made as follows.

1. Withdraw
2. Balance check
3. deposit

Deposit state is represented by S7 in which the user allowed to deposit the amount through amount entering. The amount entered will be checked whether it overflowed or not, if not then control goes to S10 state. If balance got overflowed, control signal will go to invalid state S4. Once verification is over, the signal will goes to the anything else state S11 from which user will be allowed to make more transaction by taking to state S3 else to the state S0.

Withdraw state is represented by S5 and balance check state with S6 respectively.

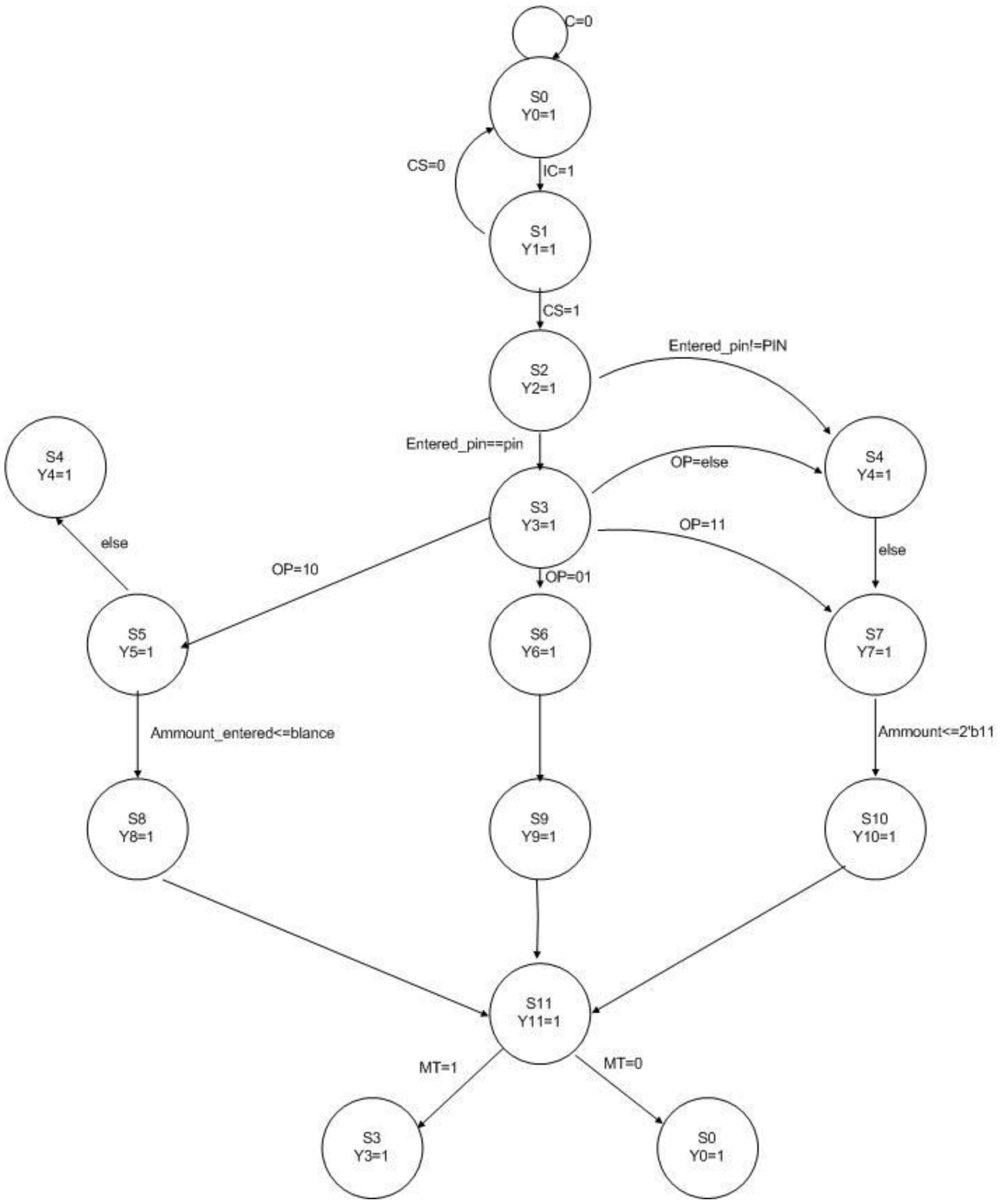


Figure 3.2 Moore Machine Diagram for ATM

Withdraw state is represented by S5 and balance check state with S6 respectively.

Table 3.1 State Code Description

State Code	State Description
S0	Idle State
S1	Scan Card
S2	Enter Pin
S3	Option Select
S4	Invalid
S5	Withdraw
S6	Balance Check
S7	Deposit
S8	Money Withdraw
S9	Balance Show
S10	Money Deposited
S11	Anything Else

Table 3.2 Input Signal Description

Signal Code	Signal Description
IC	Insert Card
CS	Card Scan
OP	Option Select
MT	More Transaction

3.2.2 – Verilog HDL Coding for ATM

```
module
source(clk,rst,entered_pin,ammount_entered,OP,IC,CS,MT,blance,pin,y0,y1,y2,y3,y4
,y5,y6,y7,y8,y9,y10,y11);
input wire clk,rst,IC,CS,MT; //IC=insert_card,CS=card_scan,MT=more_transaction
input wire [1:0] entered_pin;
input wire [1:0] ammount_entered;
input wire [1:0] OP; //OP= option that is selected by the user
output reg y0,y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11;
input wire [1:0] blance;
input wire  [1:0] pin;
reg [1:0] blance_reg;
reg [1:0] pin_reg;
parameter  [3:0] idle_state=4'b0000, // s0
                        scan_card=4'b0001,  //s1
                        enter_pin=4'b0010,  //s2
                        option_select=4'b0011, //s3
                        invalid=4'b0100,     //s4
```

```
                                        withdraw=4'b0101,  //s5
                                        blance_check=4'b0110,  //s6
                                        deposit=4'b0111,    //s7
                                        money_withdraw=4'b1000,  //s8
                                        blance_show=4'b1001,  //s9
                                        money_deposited=4'b1010,  //s10
                                        anything_else=4'b1011;  //s11

reg [3:0] current_state, next_state;

always @(*)
begin
y0 <= 0;
        y1 <= 0;
        y2 <= 0;
        y3 <= 0;
        y4 <= 0;
        y5 <= 0;
        y6 <= 0;
        y7 <= 0;
        y8 <= 0;
        y9 <= 0;
        y10 <= 0;
        y11 <= 0;
next_state <= 0;
blance_reg <= blance;
pin_reg <= pin;
case(current_state)
        idle_state:begin
                                                  if(IC)
                                                       begin
                                                       next_state <= scan_card;
                                                       y1 <= 1;
                                                       end
                                                  else
                                                       begin
                                                       next_state <= idle_state;
```

```
                                                y0 <= 1;
                                                end
                                end
                scan_card: begin
                                        if(CS)
                                                begin
                                                next_state <= enter_pin;
                                                y2 <= 1;
                                                end
                                        else
                                                begin
                                                next_state <= idle_state;
                                                y0 <= 1;
                                                end
                                end
                enter_pin: begin
                                        if(entered_pin == pin_reg)
                                                begin
                                                        next_state <=
        option_select;

                                                        y3 <= 1;
                                                end
                                        else
                                                begin
                                                next_state <= invalid;
                                                y4 <= 1;
                                                end
                                end
                invalid: begin
                                    next_state <= idle_state;
                                    y0 <= 1;
                               end
                option_select:begin
                                        if(OP == 2'b10)
                                                begin
                                                next_state <= withdraw;
                                                y5 <= 1;
```

```
                                                            end
                                                    else if(OP == 2'b01)
                                                            begin
                                                            next_state <= blance_check;
                                                            y6 <= 1;
                                                            end
                                                    else if(OP == 2'b11)
                                                            begin
                                                            next_state <= deposit;
                                                            y7 <= 1;
                                                            end
                                                    else
                                                            begin
                                                            next_state <= invalid;
                                                            y4 <= 1;
                                                            end
                                            end
            withdraw: begin
                                                    if(ammount_entered <= blance_reg)
    //if ammount entered is less that blance in account
                                                            begin
                                                            next_state <= money_withdraw;
                                                            y8 <= 1;
                                                            end
                                                    else
                                                            begin
                                                            next_state <= invalid;
                                                            y4 <= 1;
                                                            end
                                    end
            blance_check: begin
                                                    next_state <= blance_show;
                                                    y9 <= 1;
                                            end
            deposit: begin
                                                    if(ammount_entered <= 11) //if entered
ammount is 16 bit num
```

```
                                                            begin
                                                            next_state <= money_deposited;
                                                            y10 <= 1;
                                                            end
                                                      else
                                                            begin
                                                            next_state <= invalid;
                                                            y4 <= 1;
                                                            end
                                          end
                        money_withdraw: begin
                                                            next_state <= anything_else;
                                                            y11 <= 1;
                                                       end
                        blance_show: begin
                                                      next_state <= anything_else;
                                                      y11 <= 1;
                                                end
                        money_deposited: begin
                                                                  next_state <=
               anything_else;
                                                                  y11 <= 1;
                                                       end
                        anything_else: begin
                                                            if(MT)
                                                                  begin
                                                                  next_state <=
               option_select;
                                                                  y3 <= 1;
                                                                  end
                                                            else
                                                                  begin
                                                                  next_state <= idle_state;
                                                                  y0 <= 1;
                                                                  end
                                                      end
               endcase
```

```
end

always @(posedge clk or negedge rst)
begin
    if(!rst)
    begin
            current_state<=idle_state;

            end
    else
            current_state<=next_state;
end
endmodule
```

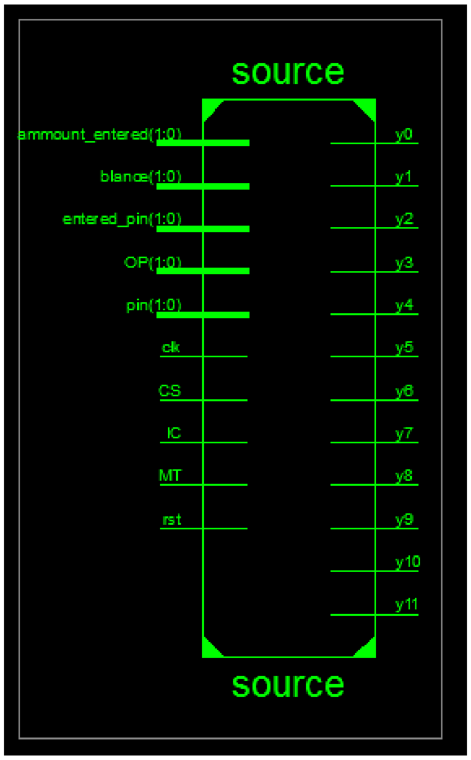
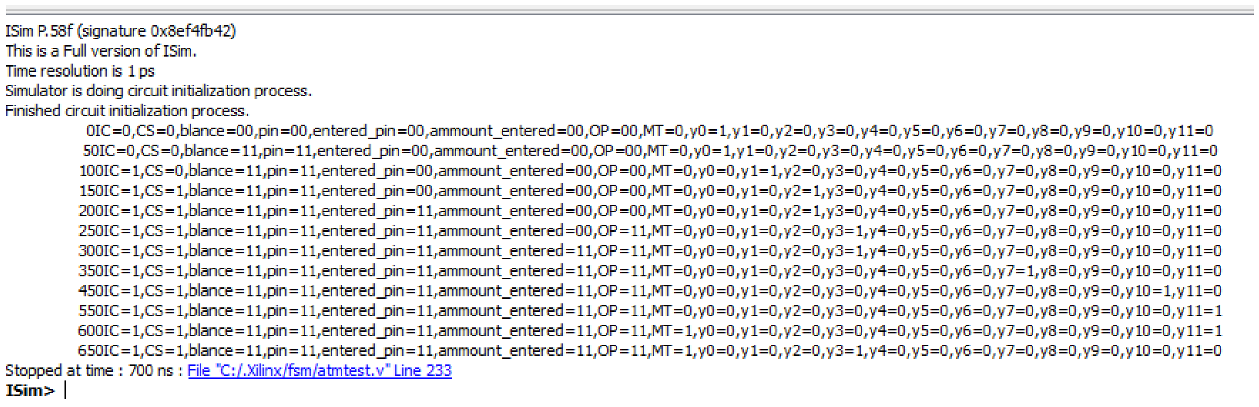


Figure 3.3 RTL

3.2.3- Testing for ATM on Xilinx

Testing is done on this code by giving different inputs to the code and outputs are checked as shown in fig 3.4.



```
ISim P.58f (signature 0x8ef4fb42)
This is a Full version of ISim.
Time resolution is 1 ps
Simulator is doing circuit initialization process.
Finished circuit initialization process.
              0IC=0,CS=0,blance=00,pin=00,entered_pin=00,ammount_entered=00,OP=00,MT=0,y0=1,y1=0,y2=0,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
             50IC=0,CS=0,blance=11,pin=11,entered_pin=00,ammount_entered=00,OP=00,MT=0,y0=1,y1=0,y2=0,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
            100IC=1,CS=0,blance=11,pin=11,entered_pin=00,ammount_entered=00,OP=00,MT=0,y0=0,y1=1,y2=0,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
            150IC=1,CS=1,blance=11,pin=11,entered_pin=00,ammount_entered=00,OP=00,MT=0,y0=0,y1=0,y2=1,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
            200IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=00,OP=00,MT=0,y0=0,y1=0,y2=1,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
            250IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=00,OP=11,MT=0,y0=0,y1=0,y2=0,y3=1,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
            300IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=11,OP=11,MT=0,y0=0,y1=0,y2=0,y3=1,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
            350IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=11,OP=11,MT=0,y0=0,y1=0,y2=0,y3=0,y4=0,y5=0,y6=0,y7=1,y8=0,y9=0,y10=0,y11=0
            450IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=11,OP=11,MT=0,y0=0,y1=0,y2=0,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=1,y11=0
            550IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=11,OP=11,MT=0,y0=0,y1=0,y2=0,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=1
            600IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=11,OP=11,MT=1,y0=0,y1=0,y2=0,y3=0,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=1
            650IC=1,CS=1,blance=11,pin=11,entered_pin=11,ammount_entered=11,OP=11,MT=1,y0=0,y1=0,y2=0,y3=1,y4=0,y5=0,y6=0,y7=0,y8=0,y9=0,y10=0,y11=0
Stopped at time : 700 ns : File "C:/.Xilinx/fsm/atmtest.v" Line 233
ISim> |
```

Figure 3.4 Simulation Results

3.3 – Implementation Details

Coding once complied and simulated then burned on FPGA. Burning on FPGA requires several steps as follows:

1: Make new project by selecting Spartan3, XC3S400 & PQ208 as FPGA family, device name & package respectively. Use all other property values as defaults.
2: Add a new source by write clicking on the project icon.
3. Select Verilog module as type of source to be added.

4. Write desired Verilog code in the .V file.

5. After that go to Assign Package Pins (or I/O pin planning) in User Constraints menu. Assign pins to all inputs & outputs by writing the pin numbers against each input & output in "loc" tab in the format e.g. p01, p14, p128 etc. (pins assign to this project are shown in table 3.3).
6. Synthesize the code.

7. After that Double click on Implement Design.

8. Go to properties of "Generate Programming File" menu. Go to Startup Options and change the FPGA Start-Up clock to JTAG Clock. Go to Readback Options & check the "Create Mask File" option. Then run "Generate Programming File".
9. Connect your FPGA device with computer using JTAG cable & switch the power of device ON.

10. Go to properties of "Configure Device (iMPACT)" in "Configure Target Device" menu & select the "Boundary-Scan mode". After that run "Configure Device (iMPACT)".

11. While iMPACT is running, right click & add the generated bit file of your project. After that right click on the displayed device & click program. Then test your FPGA device by changing inputs & observe results.

Table 3.3 Pin assigns to Input/Output

Inputs	Pin Location	Outputs	Pin Location
Blance[0]	P139	Y0	P161
Blance[1]	P138	Y1	P172
IC	P137	Y2	P156
CS	P135	Y3	P171
Entered_pin[0]	P133	Y4	P155
Entered_pin[1]	P132	Y5	P169
OP[0]	P131	Y6	P154
OP[1]	P130	Y7	P168
Amount_entered[0]	P128	Y8	P152
Amount_entered[1]	P126	Y9	P167
MT	P125	Y10	P150
clk	P76	Y11	
rst	P78		

4 - Discussion & Related Work / Discussion on Simulation Results

Our simulation results may differ a little bit from our expectations. The outputs shown are doubled each time and we tried a lot to recover this problem by re constructing our project and by changing all the possible inputs and outputs we can.

However if we ignore the doubled results the project works perfectly.

5 – Conclusion / Results

Basically, ATM controller allows the user to interact with the memory and hence the security level is increased. This also makes the transaction in account gets easier. This work helps to understand the concept of Finite State Machine and designing architecture of a system. The familiarization of the procedure to develop the State Machine diagram of a system through system level analysis is improved. The ATM block diagram is studied and the corresponding Moore Machine State diagram is also analyzed. The FSM is modeled in Verilog HDL and verified for all the appropriate scenarios.

The simulation results have been verified for the different appropriate test cases. Finally the developed model is taken to the Xilinx tool and done the Synthesis using the FPGA family of Spartan 3E.

6 –References

[1] KHAN, DR SHOAIB AHMED., 2011. Digital Design Of SPS – A Practical Approach. United Kingdom: John Wiley & Sons, Ltd.

[2] CHU, PONG.P. , 2008. FPGA Prototyping by Verilog examples. Canada: John Wiley & Sons, Ltd.

[3] Shukla, Kuldeep B., Rao, Hetal N., Joshi Arjun H. , 2013. Implementation of ATM algorithm through VHDL,2, 91-94.

[4] Ana Monga, Balwinder Singh. 2012. Finite State Machine based Vending Machine Controller with Auto-Billing Features, 3(2), 19-28.

[5]Ali Kaichouhi, Xilinx Tutorial Spartan 3 FPGA Board Programming Examples.

[6] Ten Steps To Writing and Effective Abstract, http://www.sfedit.net.

[7] Xilinx® ISE Simulator ((ISim)) with Verilog Test Fixture Tutorial, 2010.Diligent.