



(CSE312) Electronic Design Automation

Junior CESS

FALL 2022

Project (1) ATM Machine

Name	ID
Ahmed Ehab Mohamed El-Baramony	21P0261
Ahmed Mohamed Mohamed	22P0283
Hesham Mohamed El-Affandi	21P0054
Omar Tamer Omar Kamel	21P0032
Fatma Ayman Mohamed Abdel-Salam	2100327
Nancy Amr Hussien Mansour	2101421
Youssef Habil	21P0187

Table of Contents

Table Of Figures	2
Design Document.....	3
1. State Diagram.....	3
2. Input Description.....	3
3. State Description	4
4. State Explanation	5
5. Detailed explanation	6
Verification Plan	7
1. Introduction.....	7
2. Inputs	7
3. Outputs.....	7
4. Instantiation	7
5. Clock Generation.....	8
6. Test Cases	8
7. Assertion	11
8. Results	12
Coverage	12
Waveform.....	12
9. Reference Model	12

Table Of Figures

Figure 1: ATM FSM.....	3
Figure 2: ATM Module Instantiation.....	7
Figure 3: Clock Generation.....	8
Figure 4: Test Case (1)	8
Figure 5: Test Case (2)	8
Figure 6: Test Case (3)	9
Figure 7: Test Case (4)	9
Figure 8: Test case (5)	9
Figure 9: Test Case (5)	10
Figure 10: Assertion Code.....	11
Figure 11: Assertion Test	11
Figure 12: Test Bench Waveform.....	12

Design Document

1. State Diagram

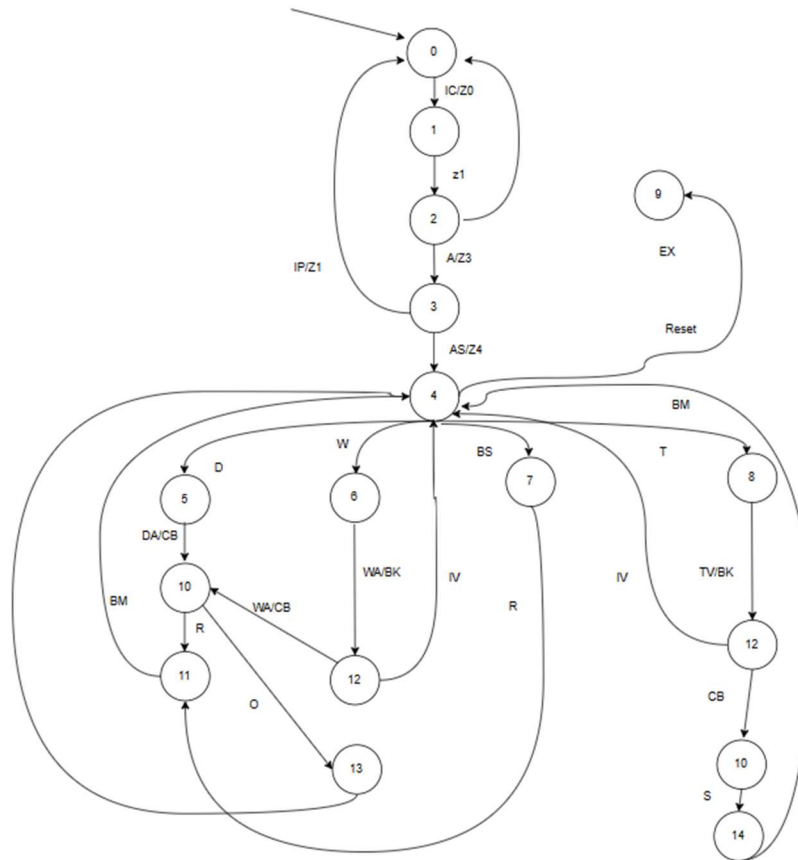


Figure 1: ATM FSM

2. Input Description

Input	Description
WA	Withdrawal Amount
DA	Deposit Amount
TA	Transfer Amount
AN	Account Number
LC	Language Chosen
BC	Balance Check
EX	Exit
Op	Operation

3. State Description

States (ABV)	States description / Actions
IC/ Z0	Insert card / language choice
Z1	Request pin
EX	Remove card
A/Z3	Authorization/ send pin to database
IP	Invalid pin
AS/Z4	Authorization success / main menu
D	Deposit
DA /CB	deposit amount/ Change balance
R	Receipt
BM	Back to main menu
W	Withdraw
WA/BK	Withdraw amount /Balance check
O	Money out
IV	Invalid value
BS	Balance Service
T	transfer
TV/BK	Transfer value / check balance
S	Successful
VP	Valid pin

4. State Explanation

- **S0:** This is the waiting, where the ATM system is ready to have a card inserted.
- **S1:** In this state, the customer will enter the pin
- **S2:** Here, the pin will be compared with the database
- **S3:** In this state, either the pin is valid so we'll jump to the next state or if the pin is invalid the system will go back to state 0.
- **S4:** There will be list of options shown to the customer representing the different operations that can be done.
- **S5:** Represents the first operation which is the deposit operation
- **S6:** Represents the second operation which is withdraw
- **S7:** Represents the third operation, which is to show that the balance exists. If it does then we'll go to S11 where the receipt will be out.
- **S8:** Represents the transfer money from an account to another operation
- **S9:** This state represents the exit state
- **S10:** In this state, we will enter the deposit amount and the system will change the customer's balance
- **S11:** Is the "receipt out" state
- **S12:** In this state, we will enter the amount to be withdrawn/transferred and the system will check if this is a convenient amount by moving to S10. If the amount entered is valid, then we will move to S13/S14. Otherwise, we will go back to the main menu S4
- **S13:** Is the state where the money is handed out and go back to main menu.
- **S14:** Is the state that will show that the transfer operation was successful.

5. Detailed explanation

Our ATM system starts with S0, where the system is waiting for a card to be inserted. Once a Card is detected, the system will ask the user to enter a pin code. This pin code will be compared against a database. If the PIN code is valid, then the system is able to go to the next state, which is S4 (main menu). Otherwise, the system will output the card and goes back to the zero state. After the system is in state 4, the user will be asked to enter a specific option number to choose the operation that needs to be done. If the user chooses s5, then he has to enter the deposit amount. The system will update his new balance, output a receipt with the new balance, and back to the S4. If the user chooses s6, he has to enter the withdrawal amount; the system has to check this amount with his balance account. If the withdrawal amount is convenient, then the process is continued and this amount is subtracted from the original account, money is handed out, and returning to state 4. If the user chooses s7, then he needs to see his current balance account. Therefore, a receipt will be output. Finally, if the user chooses s8, he will probably transfer money from his account to another account. Therefore, he will enter the value to be transferred and the system will check his current account. If the value is valid then the process is continued by letting him enter the account number to be transferred to and the transfer process is successful. Otherwise, the system goes back to state 4. If the user wants to exit from the system, he will be able to do this by choosing the exit option from main menu (state 4).

Verification Plan

1. Introduction

This Verilog test bench is designed to verify the functionality of an ATM module. It instantiates an ATM unit and tests various scenarios by simulating clock cycles, resetting the system, and applying different operations (deposit, withdrawal, transfer, balance service).

2. Inputs

- **clk**: clock signal of the module.
- **rst**: reset signal of the module.
- **Withdraw_Amount**: used for money entered for withdraw operation.
- **Transfer_Amount**: used for money entered for transfer operation.
- **Deposit_Amount**: used for money entered for deposit operation.
- **Operation**: used to choose one of the available operations which are:
 - **Deposit**: Deposited money to the account
 - **Withdraw**: withdraw money from the account
 - **Balance service**: balance inquiry
 - **Transfer**: transfer money to another account
 - **Exit**: terminate system

3. Outputs

- **FinalBalance**: Balance of the user's account
- **Final_DstBalance**: Balance of the account to which is transferred

4. Instantiation

In the test bench we take an instance of the ATM module to apply different test cases.

```
//INSTANCE OF ATM
ATM A1(clk,rst,Account_Number,PIN,Destination_Account,
Withdraw_Amount,Transfer_Amount,Deposit_Amount,Operation,1'b1,
FinalBalance,Final_DstBalance);
```

Figure 2: ATM Module Instantiation

5. Clock Generation

To generate a clock for the instance of the ATM module, we need to write a loop that runs all the time of the test.

```
//CLOCK GENERATION
initial begin
    clk=1'b0;
    forever begin
        #1 clk=~clk;
    end
end
```

Figure 3: Clock Generation

6. Test Cases

Test 1:

We begin by testing the reset functionality by resetting `rst = 1` which basically resets the variables in the system.

```
@(negedge clk);
rst=1;
@(negedge clk);
if (FinalBalance != 0) begin
    $display ("Clear has a Problem");
end
```

Figure 4: Test Case (1)

Test 2:

Disabling the `rst`, and testing with false account data and destination account.

```
rst<=0;
Account_Number<=12'hffa;
PIN<=12'h729;
Destination_Account<=12'hffe;
```

Figure 5: Test Case (2)

Test 3:

Changing the account number to a legitimate account and destination account from the database, but with wrong account PIN-Code.

```
Account_Number=12'hfff;  
PIN=12'hff1;  
Destination_Account<=12'h456;
```

Figure 6: Test Case (3)

Test 4:

Then change the PIN-Code to the right one and begin the operations.

```
repeat(3) @(negedge clk);  
PIN=12'hfff;
```

Figure 7: Test Case (4)

Test 5:

Here we begin to test the operation, starting with the deposit operation by depositing a certain amount.

```
Operation<=0;  
Deposit_Amount<=12'h11f;  
Transfer_Amount<=12'h012;  
Withdraw_Amount<=12'h013;
```

Figure 8: Test case (5)

Test 6:

Then testing the withdraw operation, with operation = 1

Test 7:

Then testing the balance service operation, with operation = 2

Test 8:

Then testing the deposit operation, with operation = 0

Test 9:

Then testing the transfer operation with operation = 3, by transferring a certain amount to the destination account balance.

Test 10:

Then changing the amounts to a larger amount to test the failure scenarios, and testing the withdraw operation, with operation = 1

```
Deposit_Amount<=12'h11f;  
Transfer_Amount<=12'hffe;  
Withdraw_Amount<=12'hffe;
```

Figure 9: Test Case (5)

Test 11:

Then testing the balance service operation, with operation = 2

Test 12:

Then testing the deposit operation, with operation = 0

Test 13:

Then testing the transfer operation with operation = 3, by transferring a certain amount to the destination account balance.

Test 14:

Then testing an invalid input operation with operation = 6, to test the failing scenario.

Test 15:

Then testing exiting the system, with operation = 4

7. Assertion

```
//psl assert never (A1.next_state == A1.S12 && A1.current_state == A1.S8 && A1.F == 0) @(negedge clk);
//psl assert never (A1.next_state == A1.S12 && A1.current_state == A1.S6 && A1.WithDraw_Amount == 0) @(negedge clk);
//psl assert never (A1.next_state == A1.S10 && A1.current_state == A1.S5 && A1.Deposit_Amount == 0) @(negedge clk);
//psl assert never (A1.next_state == A1.S4 && A1.current_state == A1.S2 && A1.VP == 0) @(negedge clk);
// psl assert never ((A1.op != 3'b000) && (A1.next_state == A1.S5)) @(negedge A1.clk);
// psl assert never ((A1.op != 3'b001) && (A1.next_state == A1.S6)) @(negedge A1.clk);
// psl assert never ((A1.op != 3'b010) && (A1.next_state == A1.S7)) @(negedge A1.clk);
// psl assert never ((A1.op != 3'b011) && (A1.next_state == A1.S8)) @(negedge A1.clk);
```

Figure 10: Assertion Code


Assertions count	Assertions hit	Assertions missed	Assertion %	Assertion graph
8	8	0	100.00%	

Figure 11: Assertion Test

Pin Validity:

Assert that if current_state=S2 (REQUEST PIN) **but** VP=0, next_state will **never** be S4 (Main_Menu)

Deposit_Amount Transition:

Assert that if current_state=S5 (Deposit) **but** D_amount=0, next_state will **never** be S10 (CheckBalance)

Withdraw_Amount Transition:

Assert that if current_state=S6 (Withdraw) **but** W_amount=0, next_state will **never** be S12 (CheckBalance)

Transfer_Amount Transition:

Assert that if current_state=S8 (Transfer) **but** F=0, next_state will **never** be S12 (CheckBalance)

Deposit Operation Transition:

Asserts that if the operation code is not 3'b001 (deposit), the next state will **never** be S5.

Withdraw Operation Transition:

Asserts that if the operation code is not 3'b000 (Withdraw), the next state will **never** be S6.

Balance Operation Transition:

Asserts that if the operation code is not 3'b010 (Balance), the next state will **never** be S7.

Transfer Operation Transition:

Asserts that if the operation code is not 3'b011 (transfer), the next state will **never** be S8.

8. Results

Coverage

Branch Coverage:	86.66% (52/60 Hits)
FSM States Coverage:	100% (15/15 Hits)
Statement Coverage:	88.65% (86/97 Hits)
Toggle Coverage:	75.32% (232/308 Hits)

Total Coverage: 82.03%

(For detailed coverage details refer to the coverage report with the project files)

Waveform

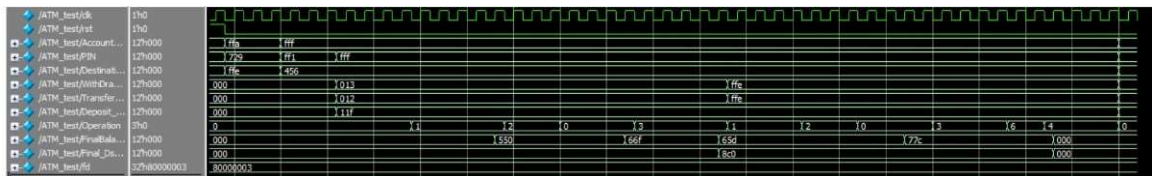


Figure 12: Test Bench Waveform

9. Reference Model:

A Reference Model is made with a high-level programming language to simulate the project (**C++ Programming Language**), and then tested with the same inputs of the above test cases and then outputs are monitored, the outputs then is written to an output file as well as the path of the states, file is presented with the reference model files.

(check "outputs_cpp.txt" for detailed outputs of the reference model test cases)

N.B.

The database used in Verilog & Reference Model are both identical (Same Exact Data), but in different extensions.

- **Verilog uses a (.txt) file** that contains the data in hexadecimal.
- **Reference Model uses a (.csv) file** that contain the data in decimal.

We did face some technical difficulties to read from a (.csv) in Verilog, so we did this as a solution to have the **same exact data verified in Both Models**.

Thank You