Design and Implementation of Automated Teller Machine (FSM) Controller – Documentation

by SAN team

Verilog Code Documentation - ATM

Introduction:

The Verilog code represents an ATM (Automated Teller Machine) controller. It simulates the functionality of an ATM, allowing users to perform transactions such as withdrawal, deposit, balance inquiry, and mini statement generation. The ATM controller includes an FSM (Finite State Machine) to handle different checks and operations.

Design Overview:

The ATM controller is designed using a state machine approach. It consists of several modules that handle different aspects of the ATM operation, such as PIN verification, transaction processing, face recognition, and account lockout. The FSM ensures that the ATM performs the required checks and operations based on the user's inputs.

Modules:

The top-level module Atm is responsible for coordinating the different components of the ATM and managing the overall state of the system.

Inputs:

clk: Clock input for synchronization.

reset: Reset input to initialize the system.

pin: PIN input for PIN verification.

withdraw: Signal to initiate a withdrawal transaction.

deposit: Signal to initiate a deposit transaction.

balance_disp: Signal to display the account balance.

mini_stmt: Signal to generate a mini statement.

face_recog: Signal to initiate face recognition.

Outputs:

account_locked: Indicates whether the account is locked.

balance: Current account balance.

new_balance: Updated account balance after a transaction.

recent_transactions: Recent transaction information.

Internal Signals

state: Current state of the ATM controller.

pin_attempts: Number of invalid PIN attempts.

withdrawal_amount: Amount to be withdrawn in a transaction.

deposit_amount: Amount to be deposited in a transaction.

transaction_amount: Amount involved in the current transaction.

transaction_result: Result of the transaction.

lockout_timer: Timer to track the lockout duration.

time_counter: Counter to track time elapsed.

test_passed: Flag indicating the result of the face recognition test.

correct_pin: Variable to store the correct PIN.

transaction_count: Counter for the number of transactions in the mini statement.

transaction_history: Array to store the transaction history.

Operation/Behaviour

The Atm module implements a state machine that transitions through different states based on user inputs and internal logic. Here's a summary of the states and their behaviour:

IDLE: The system is idle, waiting for user inputs.

- If the PIN is correct, transitions to PERFORMING_TRANSACTION.
- If the PIN is incorrect, transitions to VERIFYING_PIN and increments pin_attempts.
- If the maximum PIN attempts are reached, transitions to ACCOUNT_LOCKED and sets account_locked and lockout_timer.
- VERIFYING_PIN: Verifies the PIN entered by the user.

- If the PIN is correct, transitions to PERFORMING_TRANSACTION and resets pin_attempts.
- If the PIN is incorrect, increments pin_attempts.
- If the maximum PIN attempts are reached, transitions to ACCOUNT_LOCKED and sets account_locked and lockout_timer.

PERFORMING_TRANSACTION: Handles different transaction types.

If withdraw is asserted, checks if the withdrawal amount exceeds the limit.

If the amount exceeds the limit, transitions to FACE_RECOGNITION_TEST.

If the amount is within the limit, performs the withdrawal transaction, updates the balance, recent transactions, and mini statement history, and transitions back to IDLE.

If deposit is asserted, performs the deposit transaction, updates the balance, recent transactions, and mini statement history, and transitions back to IDLE.

If balance_disp is asserted, displays the old balance and new balance, and transitions back to IDLE.

If mini_stmt is asserted, generates a mini statement with recent transactions and transitions back to IDLE.

FACE_RECOGNITION_TEST: Performs a face recognition test.

If face_recog is asserted, the face recognition test is conducted.

If the test passes and the withdrawal amount exceeds the limit, performs the withdrawal transaction, updates the balance, recent transactions, and mini statement history, and transitions back to IDLE.

If the test fails, transitions back to PERFORMING_TRANSACTION.

ACCOUNT_LOCKED: The account is locked due to too many incorrect PIN attempts.

Waits for the lockout duration to expire.

When the lockout timer elapses, clears the account lock and transitions back to IDLE.

Timing Considerations

The code operates on the rising edge of the clock (clk). Timing constraints, if any, can be specified depending on the target FPGA or ASIC implementation.

Example Usage

Instantiate the Atm module in your design and connect the necessary inputs and outputs. Here's an example:

verilog

Copy code

```
atm my_atm (
  .clk(clk),
```

```
  .reset(reset),
  .pin(pin),
  .withdraw(withdraw),
  .deposit(deposit),
  .balance_disp(balance_disp),
  .mini_stmt(mini_stmt),
  .face_recog(face_recog),
  .account_locked(account_locked),
  .balance(balance),
  .new_balance(new_balance),
  .recent_transactions(recent_transactions)
);
```

atm_tb (Testbench)

The atm_tb module is a testbench for the Atm module. It verifies the behaviour of the ATM controller under different test scenarios.

Inputs:

clk: Clock input for testbench simulation.

reset: Reset input for testbench initialization.

pin: PIN input for testbench simulation.

withdraw: Signal to initiate a withdrawal transaction in the testbench.

deposit: Signal to initiate a deposit transaction in the testbench.

balance_disp: Signal to display the account balance in the testbench.

mini_stmt: Signal to generate a mini statement in the testbench.

face_recog: Signal to simulate the face recognition test in the testbench.

Outputs:

account_locked: Indicates whether the account is locked in the testbench.

balance: Current account balance in the testbench.

new_balance: Updated account balance after a transaction in the testbench.

recent_transactions: Recent transaction information in the testbench.

Test Scenarios:

The testbench includes several test scenarios to validate different aspects of the ATM controller, such as PIN verification, withdrawal, deposit, balance display, and mini statement generation. The test scenarios cover the checks specified, including invalid PIN entry, withdrawal with face recognition test, deposit, and the display of old balance and new balance.

Simulation Results:

During simulation, the testbench verifies the expected outputs of the ATM controller based on the test scenarios. The test results are displayed using the $display system task.

Conclusion:

The provided Verilog code implements an ATM controller with PIN verification, transaction processing, and other functionality. The testbench verifies the behaviour of the ATM controller under different test scenarios, including the required checks of invalid PIN entry, withdrawal with face recognition.

Done By:

Team Name: SAN Team

Shaik. Siddiq

Shaik. Arshad

V. Navya reddy