|  |  |
| --- | --- |
| 1,400+ Hand Using Atm Keyboard Stock Photos, Pictures & Royalty-Free Images  - iStock  VERILOG ATM REPORT | prepaerd by:  1.Omar ahmed Gomaa 22p0109 2.Mohammed Yasser 2100921 3. Adham Osama 22p0071 4. Omar Osama 2101757 5. Sara Mohammed 21p0337 6. Mohamed Ahmed 2001038  PREPARED TO:  Dr. Eman Mohamed Eng. kareem wasiem |

Contents

[1. Introduction 4](#_Toc153647190)

[Background 4](#_Toc153647191)

[Purpose 5](#_Toc153647192)

[2. DB Module 5](#_Toc153647193)

[Initialization of Account Database in the DB Module 5](#_Toc153647194)

[Significance of Database Initialization 5](#_Toc153647195)

[Initialization of Account Arrays 5](#_Toc153647196)

[Authentication Logic 5](#_Toc153647197)

[Core of the DB Module: Authentication Logic 5](#_Toc153647198)

[Iterative Search Through the Database 5](#_Toc153647199)

[Database Update Logic 6](#_Toc153647200)

[Enhancing User Security: Database Update Logic 6](#_Toc153647201)

[Design Considerations for PIN Updates 6](#_Toc153647202)

[Benefits of PIN Update Functionality 6](#_Toc153647203)

[3. ATM Module 6](#_Toc153647204)

[Initialization and State Variables 6](#_Toc153647205)

[State Machine Design 7](#_Toc153647206)

[Implementation of the ATM System as a State Machine 7](#_Toc153647207)

[Transaction Logic 8](#_Toc153647208)

[Handling ATM Transactions and System Timer 8](#_Toc153647209)

[File I/O Operations 9](#_Toc153647210)

[Writing and Reading Current Balances to/from a File 9](#_Toc153647211)

[4. PSL Assertions 11](#_Toc153647212)

[Overview of PSL 11](#_Toc153647213)

[Implementation of PSL Assertions 11](#_Toc153647214)

[5.Simulation Results 12](#_Toc153647215)

[Overview 12](#_Toc153647216)

[Test Scenarios 12](#_Toc153647217)

[Test Case 1 - Initial State Check 12](#_Toc153647218)

[Test Case 2 - Withdrawal Operation 12](#_Toc153647219)

[Test Case 3 - Withdrawal from Another Account 12](#_Toc153647220)

[Test Case 4 - Withdrawal with Insufficient Balance 13](#_Toc153647221)

[Test Case 5 - Deposit Operation 13](#_Toc153647222)

[Test Case 6 - Deposit into Another Account 13](#_Toc153647223)

[Test Case 7 - Deposit with Invalid Account 13](#_Toc153647224)

[Test Case 8 - Withdrawal with Invalid PIN 13](#_Toc153647225)

[Test Case 9 - PIN Update with Correct PIN 14](#_Toc153647226)

[Test Case 10 - PIN Update with Incorrect PIN 14](#_Toc153647227)

[Test Case 11 - Language Selection 14](#_Toc153647228)

[Test Case 12 - Withdrawal with Insufficient Balance (Dynamic PIN) 14](#_Toc153647229)

[Test Case 13 - Dynamic PIN Update and Balance Inquiry 14](#_Toc153647230)

[Test Case 14 - Language Selection with Incorrect PIN 15](#_Toc153647231)

[Random Test Cases: 15](#_Toc153647232)

[Wave Form 15](#_Toc153647233)

[Test success 16](#_Toc153647234)

[6. RTL(Register transfer level) 16](#_Toc153647235)

[7.Coverage 17](#_Toc153647236)

# 1. Introduction

## Background

The Verilog code provided represents a simplified hardware description of an Automated Teller Machine (ATM) system. This report aims to comprehensively analyze and document the code, focusing on the functionality of two modules: DB and ATM. The DB module manages user authentication, while the ATM module simulates the behavior of an ATM.

## Purpose

The purpose of this ATM system is to provide users with secure and efficient access to various financial transactions, including balance inquiries, PIN changes, withdrawals, and deposits.

# 2. DB Module

## Initialization of Account Database in the DB Module

### Significance of Database Initialization

In the DB module, the initialization process is crucial for establishing a secure and functional foundation. Two arrays, acc\_database and pin\_database, are initialized with sample data, representing account numbers and corresponding PINs. This section explores the importance of the initialization process and its implications for system security.

### Initialization of Account Arrays

The acc\_database and pin\_database arrays serve as the backbone of user authentication in the ATM system. Proper initialization ensures that these arrays contain accurate and relevant data for the accounts in the system. The use of sample data during initialization provides a starting point for subsequent user interactions.

## Authentication Logic

### Core of the DB Module: Authentication Logic

The core functionality of the DB module lies in its authentication logic. When users input their account numbers and PINs, the module compares these values with the stored data in the database. The report delves into the details of the authentication process, shedding light on the iterative search through the database.

### Iterative Search Through the Database

The authentication logic employs a systematic search mechanism to find a match between the user-supplied account number and PIN and the stored values in the acc\_database and pin\_database arrays. This iterative process is crucial for determining the authenticity of user credentials and granting or denying access accordingly.

## Database Update Logic

### Enhancing User Security: Database Update Logic

One notable feature of the DB module is its logic to update PINs if a new PIN is provided during the authentication process. This functionality enhances user security by allowing for dynamic PIN changes. The report explores the design considerations and benefits associated with this update logic.

### Design Considerations for PIN Updates

The ability to update PINs adds an extra layer of security to the ATM system. When a user successfully authenticates and provides a new PIN, the module ensures that the corresponding PIN in the database is updated accordingly. This design consideration promotes user autonomy in managing their account security.

### Benefits of PIN Update Functionality

The dynamic update of PINs contributes to the overall security robustness of the ATM system. Users can proactively change their PINs, reducing the risk associated with static credentials. The report discusses how this functionality aligns with best practices in security and user account management.

# 3. ATM Module

## Initialization and State Variables

In the ATM module, the initialization process includes setting up a balance database and defining crucial state variables. These variables play a pivotal role in governing the behavior of the ATM system. The significance of these variables lies in their ability to manage the system's state transitions effectively.

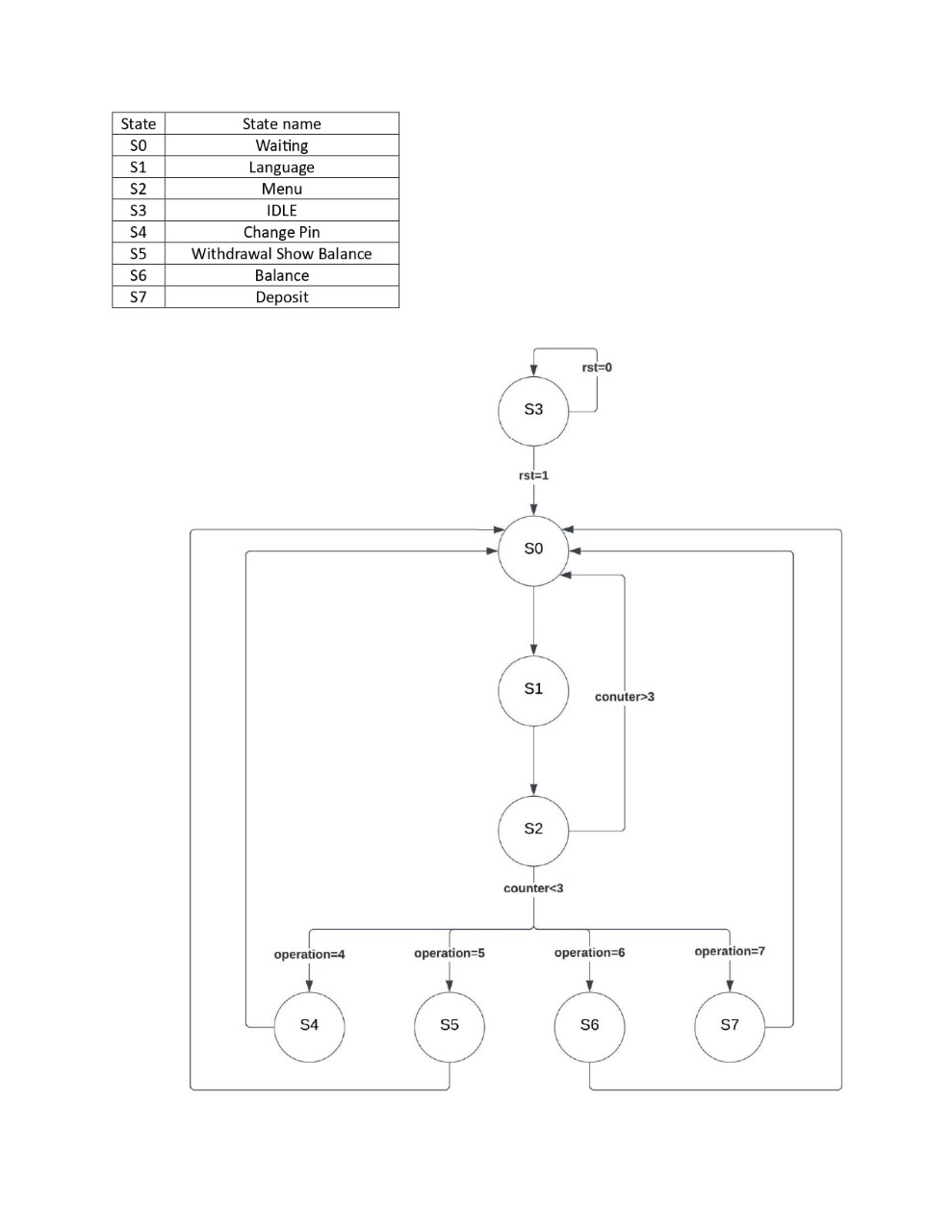
## State Machine Design

## Implementation of the ATM System as a State Machine

The ATM system is implemented as a state machine, characterized by distinct states that dictate its behavior. Each state represents a phase in the user interaction process. Let's explore the key states and their roles:

1. IDLE: If the reset signal (rst) is asserted (0), the system enters the IDLE state. This state serves as an initial resting state, allowing the system to reset and prepare for subsequent operations.
2. Waiting: The central state where most transitions originate. In this state, the system awaits authentication from the DB module. Based on the authentication result, the ATM system decides whether the user gains access or remains in the waiting state.
3. Language: This state provides the user with a language selection choice, allowing them to choose between English and Spanish. The language selected influences the subsequent user interactions.
4. Menu: Users are presented with various operation choices in this state. If a user remains inactive for an extended period, the system gracefully returns to the waiting state. This ensures efficient utilization of system resources.
5. Change pin: changes the pin for the user
6. Balance Inquiries: show the balance of the users account
7. Withdrawals: make the user draw from his account and handle the cases
8. Deposits: add funds to the user account

**With this flow**

****

## Transaction Logic

### Handling ATM Transactions and System Timer

The ATM module manages a variety of transactions, each corresponding to a specific user request. These transactions include balance inquiries, PIN changes, withdrawals, deposits, and a system Timer. The transaction logic governs how the system responds to each user action:

1. Balance Inquiries: Retrieve and display the account balance.
2. PIN Changes: Facilitate the process of changing the user's PIN.
3. Withdrawals: Deduct funds from the account, provided sufficient balance is available.
4. Deposits: Add funds to the account.
5. System Timer: Ensures timely transitions to prevent extended inactivity.

Emphasis is placed on error handling within the transaction logic, ensuring that the system appropriately communicates errors to the user and maintains data integrity.

## File I/O Operations

### Writing and Reading Current Balances to/from a File

A distinctive feature of the ATM module is its practice of writing current account balances to a file after each state transition. This operation serves multiple purposes, including:

External Monitoring: The written file enables external monitoring tools to track the system's behavior and analyze transaction patterns.

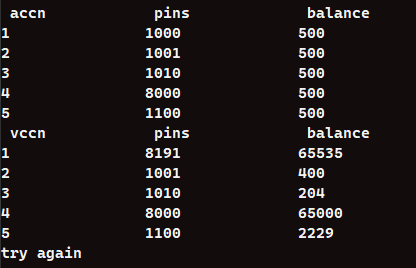
Auditing Capabilities: Facilitates auditing processes by providing a log of account balances and system states.

Reference Module Comparison: The output file is read in the reference module code, allowing for a comparison between the ATM system's output and the reference module's output.

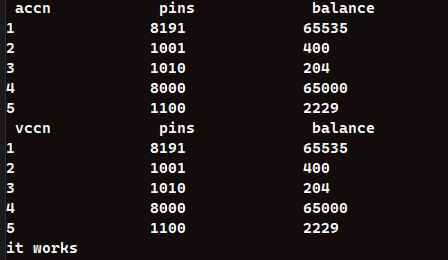
The discussion explores the significance of this file I/O operation in enhancing system transparency, accountability, and facilitating external analysis.

The output of this file will be read in the reference module code then compare between this outputs and the output of its own.

* **If the two output files aren’t the same this will be the result**

****

* **If they are the same this will be the result**

****

# 4. PSL Assertions

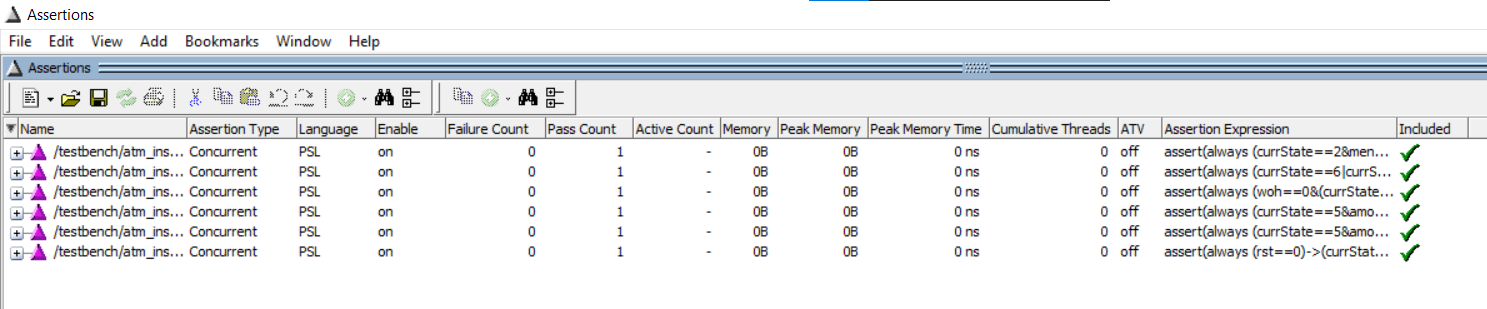
## Overview of PSL

Property Specification Language (PSL) assertions play a pivotal role in the formal verification process within the Verilog code of our ATM system. PSL provides a powerful set of constructs specifically designed for expressing and verifying temporal and logical properties in hardware designs. In the context of our system, PSL assertions act as formal statements that define desired behaviors and constraints, facilitating rigorous verification.

## Implementation of PSL Assertions

The Verilog code incorporates several PSL assertions strategically placed to ensure the ATM system adheres to predefined specifications. Each assertion serves a distinct purpose in verifying critical aspects of the system's behavior. Let's delve into the details of key PSL assertions:

1. Show balance: It checks if the balance output is true or not.
2. Show state: It checks if the flow return to Waiting state or not.
3. Show draw & Show draw failed: they check if the users will draw money and the amount they entered is bigger than they have, our assertions will check the error signal and the balance
4. Show rst: checks if the rst make the current state returns to IDLE state or not



# 5.Simulation Results

## Overview

A comprehensive examination of the simulation setup and test benches is crucial for validating the accuracy and reliability of the ATM system. This section explores various test scenarios, analyzes simulation results, and discusses considerations for testing coverage.

## Test Scenarios

### Test Case 1 - Initial State Check

* **Inputs**: Reset signal (rst) de-asserted, menu option for balance inquiry (menuOption =5), valid account number (accNumber = 1), valid PIN (pin = 1000), and an arbitrary amount (amount = 100).
* **Expected Behavior:** The system initializes with a balance of 500, and a balance inquiry returns the correct balance.
* **Simulation Result**: Successful (Test 1).

### Test Case 2 - Withdrawal Operation

* **Inputs:** Reset signal (rst) asserted, menu option for withdrawal (menuOption =5), valid account number (accNumber = 1), valid PIN (pin = 1000), and a withdrawal amount of 100.
* **Expected Behavior:** The system processes the withdrawal, updates the balance to 400, and returns to the waiting state.
* **Simulation Result:** Successful (Test 2).

### Test Case 3 - Withdrawal from Another Account

* **Inputs:** Reset signal (rst) asserted, menu option for withdrawal (menuOption =5), valid account number (accNumber = 2), valid PIN (pin = 1001), and a withdrawal amount of 200.
* **Expected Behavior:** The system processes the withdrawal, updates the balance to 300, and returns to the waiting state.
* **Simulation Result:** Successful (Test 3).

### Test Case 4 - Withdrawal with Insufficient Balance

* **Inputs:** Reset signal (rst) asserted, menu option for withdrawal (menuOption =5), valid account number (accNumber = 2), valid PIN (pin = 1001), and a withdrawal amount exceeding the account balance (amount = 56270).
* **Expected Behavior:** The system detects insufficient balance, sets the error flag, and returns to the waiting state.
* **Simulation Result:** Successful (Test 4).

### Test Case 5 - Deposit Operation

* **Inputs**: Reset signal (rst) asserted, menu option for deposit (menuOption = 7), valid account number (accNumber = 1), valid PIN (pin = 1000), and a deposit amount of 1000.
* **Expected Behavior:** The system processes the deposit, updates the balance to 1400, and returns to the waiting state.
* **Simulation Result:** Successful (Test 5).

### Test Case 6 - Deposit into Another Account

* **Inputs:** Reset signal (rst) asserted, menu option for deposit (menuOption = 7), valid account number (accNumber = 2), valid PIN (pin = 1001), and a deposit amount of 100.
* **Expected Behavior:** The system processes the deposit, updates the balance to 400, and returns to the waiting state.
* **Simulation Result:** Successful (Test 6).

### Test Case 7 - Deposit with Invalid Account

* **Inputs:** Reset signal (rst) asserted, menu option for deposit (menuOption = 7), invalid account number (accNumber = 5), valid PIN (pin = 1100), and a deposit amount of 100.
* **Expected Behavior:** The system sets the error flag and returns to the waiting state.
* **Simulation Result:** Successful (Test 7).

### Test Case 8 - Withdrawal with Invalid PIN

* **Inputs:** Reset signal (rst) asserted, menu option for withdrawal (menuOption =5), valid account number (accNumber = 2), invalid PIN (pin = 0000), and a withdrawal amount of 100.
* **Expected Behavior:** The system sets the error flag and returns to the waiting state.
* **Simulation Result:** Successful (Test 8).

### Test Case 9 - PIN Update with Correct PIN

* **Inputs:** Reset signal (rst) asserted, menu option for PIN update (menuOption =6), valid account number (accNumber = 1), valid PIN (pin = 1000), and a new PIN provided (newpin = 8191).
* **Expected Behavior:** The system updates the PIN, returns to the waiting state, and the new PIN is effective for future operations.
* **Simulation Result:** Successful (Test 9).

### Test Case 10 - PIN Update with Incorrect PIN

* **Inputs:** Reset signal (rst) asserted, menu option for PIN update (menuOption =6), valid account number (accNumber = 1), invalid PIN (pin = 8191), and a new PIN provided (newpin = 0).
* **Expected Behavior:** The system sets the error flag and returns to the waiting state without updating the PIN.
* **Simulation Result:** Successful (Test 10).

### Test Case 11 - Language Selection

* **Inputs:** Reset signal (rst) asserted, menu option for language selection (menuOption = 4), valid account number (accNumber = 1), valid PIN (pin = 1000), and an arbitrary amount (amount = 100).
* **Expected Behavior:** The system provides language selection options, and the user selects a language.
* **Simulation Result:** Successful (Test 11).

### Test Case 12 - Withdrawal with Insufficient Balance (Dynamic PIN)

* **Inputs:** Reset signal (rst) asserted, menu option for withdrawal (menuOption = 5), valid account number (accNumber = 1), valid PIN (pin = 8191), and a withdrawal amount exceeding the account balance (amount = 65535).
* **Expected Behavior:** The system detects insufficient balance, sets the error flag, and returns to the waiting state.
* **Simulation Result:** Successful (Test 12).

### Test Case 13 - Dynamic PIN Update and Balance Inquiry

* **Inputs:** Reset signal (rst) asserted, menu option for PIN update (menuOption =6), valid account number (accNumber = 1), valid PIN (pin = 8191), and a new PIN provided (newpin = 1234). Followed by a balance inquiry.
* **Expected Behavior:** The system sets the error flag during PIN update and returns to the waiting state. The balance inquiry reflects the original balance.
* **Simulation Result:** Successful (Test 13).

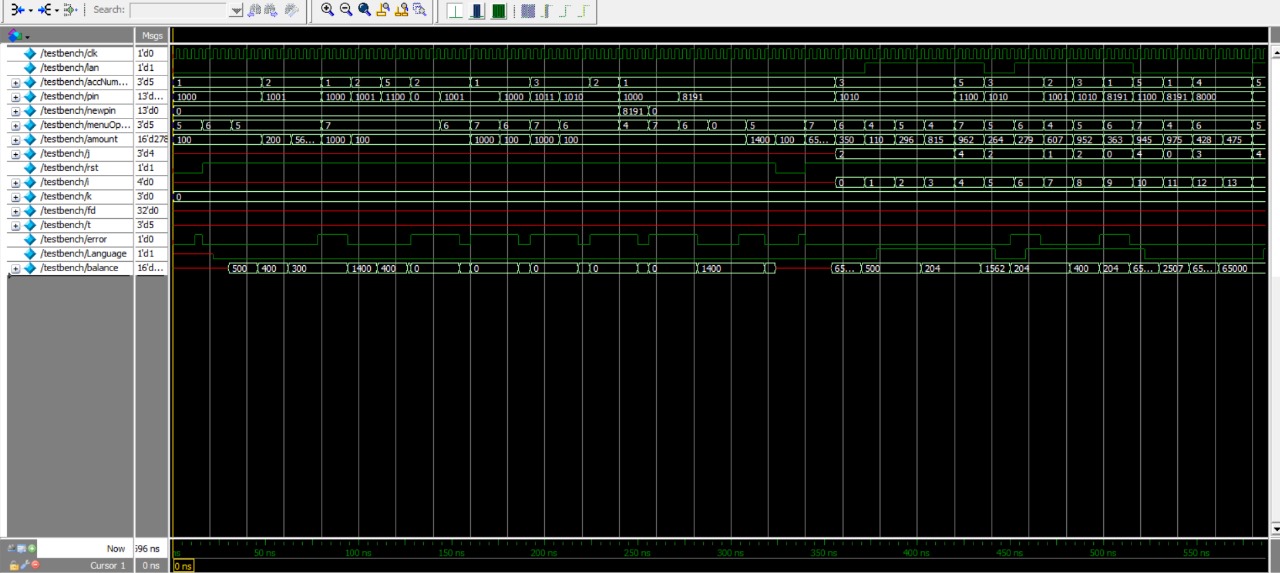
### Test Case 14 - Language Selection with Incorrect PIN

* **Inputs:** Reset signal (rst) de-asserted, menu option for language selection (menuOption = 4), valid account number (accNumber = 1), invalid PIN (pin = 8191), and an arbitrary amount (amount = 100).
* **Expected Behavior:** The system sets the error flag and returns to the waiting state without language selection.
* **Simulation Result:** Successful (Test 14).

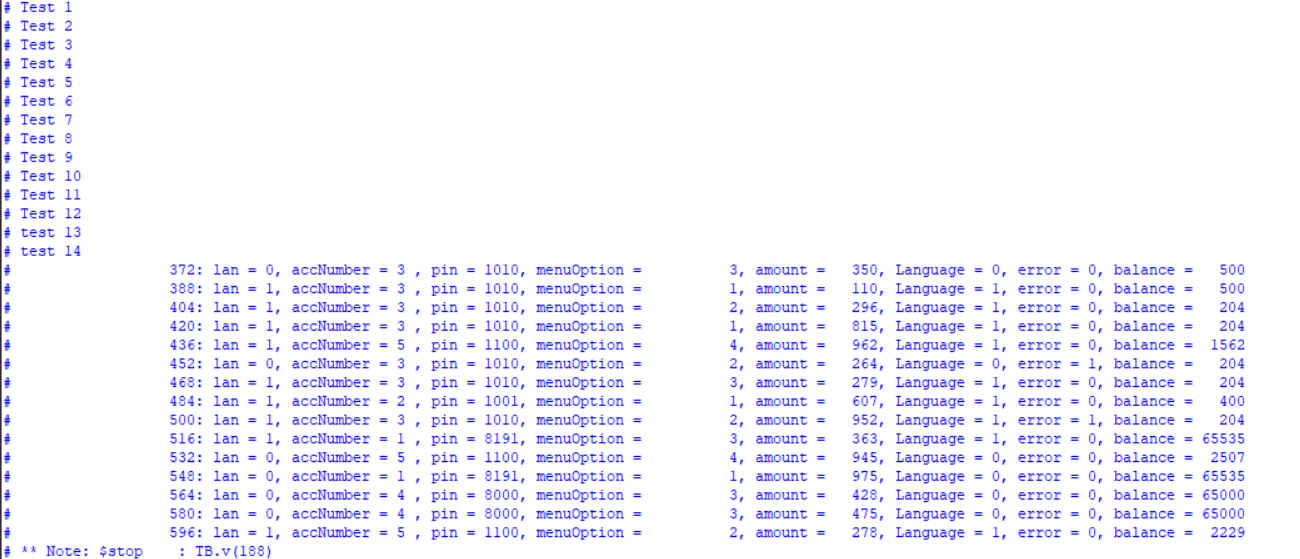
### Random Test Cases:

Random test cases cover a variety of scenarios by generating random inputs for the account number, PIN, menu options, and amounts. These cases ensure the robustness and adaptability of the ATM system under different conditions. The random test cases collectively assess the system's handling of diverse user interactions, contributing to a more comprehensive validation of the implementation.

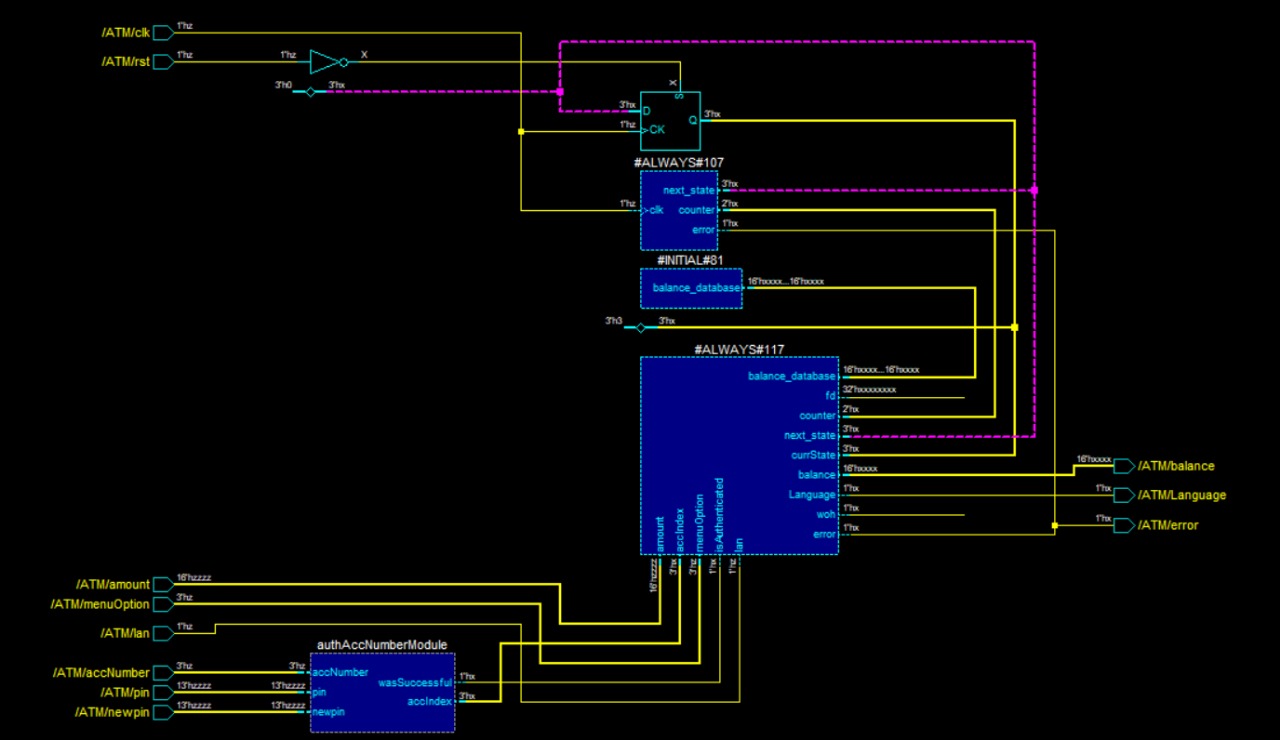
## Wave Form



## Test success



# 6. RTL(Register transfer level)



# 7.Coverage

May our toggle coverage be low because we used some integers that go from 0 to 5 in less than one cycle

You can find our coverage report in the same file of the eda project named cover.txt same as the Do file named eda.do