IDD

Features 1

Block Diagram 2

Port Description 4

Block Description 5  
Block Parameters 6

Registers Description 6  
Timing Specification …………………………………………………………………………………… 7  
Implementation Description ……………………………………………………………………….……8 Verification …………………………………………………………………………………………….9  
Synthesis ……………………………………………………………………………………………10

# Features:

### This System provides the following operations : 1) Withdrawal Operation . 2) Deposite Operation . 3) Balance\_Service . 4) Transfer operation .

### It has a database mimicking unit defined by : -Ram block : Where it contains the available credit & the up-limit value for the user to withdraw or transfer money . It is updated for each operation that causes changes in user credit or up-limit -Rom block : It contains the ref-password for the credit card and a valid indicator to show whether this user account is valid or not .

### 

# 2.Block Diagram

# 3.Port Description

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Port Name | Width | Direction | Src/Dest | Clock | Power Domain | Description |
| clk | 1 | Input | System | - | - | Clock used to time all capturing @ posedge [TBD] |
| slow\_clk | 1 | Input | System | - | - | Clock used to count waiting timers . |
| rst\_n | 1 | Input | SYSTEM | - | - | Asynchronous active low reset |
| available\_credit\_out | CREDIT\_VAL\_SIZE | OUTPUT | USER | - | - | For displaying available credit for user in . |
| abort | 1 | OUTPUT | SYSTEM | - | - | (PULSE)A Flag to indicate an error happened during operation. |
| done | 1 | OUTPUT | SYSTEM | - | - | (PULSE)A Flag to indicate a successful operation . |
| card\_spell\_out | 1 | OUTPUT | SYSTEM | - | - | (PULSE)Control signal triggered : 1) Successful operation is done  2)Cancel at OPERATION\_CHOICE state . |
| wr\_en | 1 | OUTPUT | card\_handler | - | - | To define operation on RAM/ROM on card handler : 0:Read 1:Write |
| final\_credit | CREDIT\_VAL\_SIZE | OUTPUT | card\_handler | - | - | Indicate final credit  (Ex. In Balance service operation) |
| final\_up\_limit | UP\_LIMIT\_SIZE | OUTPUT | card\_handler | - | - | Update Final\_uplimit for a user after operations. |
| transfer\_en | 1 | OUTPUT | card\_handler | - | - | (Level)Enable transfer operation mode. |
| atm\_init | 1 | INPUT | System | - | - | Address bus |
| insert | 1 | INPUT | System |  |  | Pulse: Triggered high when card is in. |
| atm\_capacity | ATM\_CAP\_SIZE | INPUT | System | - | - | Defines the Max capcity of cach available in ATM: Max Value = 200,000$ |
| op\_choice | OP\_CHOICE\_SIZE | INPUT | USER | - | - | Available operations : 1) Cancel 000 2) Deposit 001 3) Withdrawal 010 4) balance\_service 011 |
| language\_choice | 1 | INPUT | USER | - | - | 0: Arabic 1:English |
| Input\_password | PASSWORD\_SIZE | INPUT | USER | - | - | Contains 4 numbers , Each number is represented in 4 bits |
| deposite\_value | DEPOSITE\_SIZE | INPUT | USER | - | - | Defines Deposite value |
| withdrawal\_value | WITHDRAW\_SIZE | INPUT | USER | - | - | Defines Deposite value |
| transfer\_value | TRANSFER\_SIZE | INPUT |  |  |  |  |
| enter | 1 | INPUT | USER | - | - | (positve edge deteteced) : Capture user inputs when triggered . |
| cancel | 1 | INPUT | USER | - | - | (Pulse) when triggered : Back from any operation to OPERATION\_CHOICE state . Back from OPERATION\_CHOICE state to IDLE . |
| enter | 1 | INPUT | USER | - | - | (posedge detected )  A Control signal to capture user inputs . |
| available\_credit | CREDIT\_VAL\_SIZE | INPUT | CARD\_HANDLER | - | - | Read data bus |
| ref\_password | PASSWORD\_SIZE | INPUT | CARD\_HANDLER | - | - | It indicates whether the transfer is done or still going : HIGH: Transfer is done Low: Still going or Still needs to be extended |
| valid | 1 | INPUT | CARD\_HANDLER | - | - | Indicates whether there’s an error or not : Low : Okay High : Error |
| up\_limit | UP\_LIMIT\_SIZE | INPUT | CARD\_HANDLER | - | - |  |

# 

4.Block Description   
**Block Description**

**ATM FSM:**

This block is the heart of our design it’s responsible for controlling all he outer blocks by providing the right input at the right time for them our state machine is responsible for:

1. Updating the credit of the user after a deposit, withdrawal, or a transfer.
2. Updating the Up limit of the user after withdrawal or a transfer.
3. Editing the information in the ATM’s database (i.e., RAM).
4. Spilling out the user’s card after the process is finished.
5. Aborting the process if there was an error or the user can’t do a certain operation.
6. Making the card handling work in the transfer or normal operation mode.

**Card Handling:**

This block contains two main components.

1. RAM for writing the user’s available credit & his/her up limit.
2. ROM containing the user’s password and a valid bit to see if the card is valid or not.

This block works in two modes the normal operation mode that takes the final credit and final up limit from the FSM and writes it in ram and the transfer mode that takes a new card pin and a transfer value from the FSM and updates the information of the recipient.

Chart, waterfall chart

Description automatically generated

# Block parameters :

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter Name | Type | Size | Description |
| CREDIT\_VAL\_SIZE |  | 25 | Size for CREDIT\_VALUE |
| UP\_LIMIT\_SIZE |  | 15 | Size for UP\_LIMIT VALUE |
| ATM\_CAP\_SIZE |  | 18 | Size for ATM\_CAP VALUE |
| PASSWORD\_SIZE |  | 16 | Size for h\_PASSWORD SIZE |
| DEPTH |  | 64 | Size for DEPTH |
| PINCARD\_SIZE |  | 6 | Size for PINCARD |
| WITHDRAW\_SIZE |  | 15 | Size to represent withdrawal money |
| TRANSFER\_SIZE |  | 15 | Size to represent Transfer money |
| DEPOSITE\_SIZE |  | 15 | Size to represent Deposited money |
| OP\_CHOICE\_SIZE |  | 3 | Size to represent operations featured in atm . |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Timer Name | clock | Size | Firing codition | Description |
| waiting\_timer | slow\_clock | 5 | waiting\_timer==WAITING\_TIME | Waiting timer for withdrawal and deposite operations . |
| last\_cycle\_timer | clk | $clog2(DIVISOR) | last\_cycle\_timer==DIVISOR | To avoid racing between slow\_clk counts and system clock . It maintain that the waiting\_timer goes back to zero before doing any transition. |

# 6. Timers

# 

# 7.Timing Specification:

## Transfer operation sequence : transfer_final

## Withdrawal operation sequence : withdrawal_final (2)

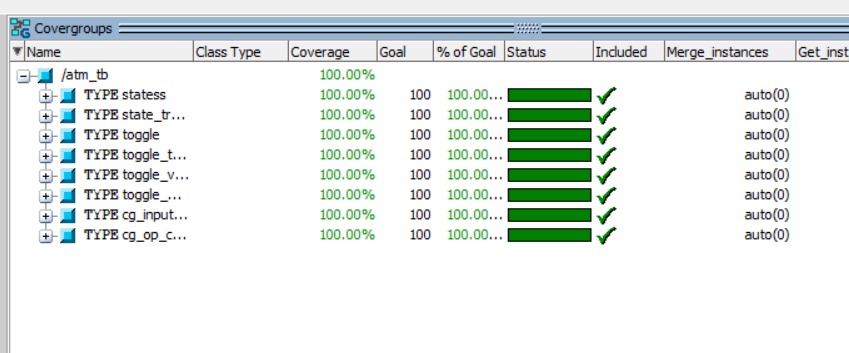
## Deposite operation sequence : Deposite

## Balance\_Service Balance_service

# Implementation Description :

# Verification [IMG_256](Test_plan.xls) (Tap image to open hyper-linked file)

# 11. Coverage / Assertions Reports : [IMG_256](..\Reports\index.html)

(Tap image to open hyper-linked file)   
  
  
**Steps to reach 100% coverage :   
1) Swapping through all possible pin\_card for all users .  
2) Randomizing Enter / Cancel occurrence an each possible states .  
3) Randomizing Input\_password (correct / false)  
4) Randomizing language\_choice   
5) Randomizing withdrawal/transfer/deposite value .  
6) Randomizing ATM capacity .**

# Bugs & Fixing :

**1) Done flag in withdrawal operation :**In Scen.8 in test plan , Feature (Done signal in withdrawal ):  
Done flag wasn’t not fired until the last\_cycle timer is finished .  
**2) we\_en in deposite operation :**  
More than assertion has failed as the wr\_en signal in deposite operation wasn’t fired when transaction operation was done .  
**3) States feed\_through :**

As all states are dependant on enter signal (user\_input) , Feed through happened between states .  
Sol.

Adding a no\_feed\_through (dead-lock signal) that disables (enter) signal sensitivity for the first clock in the state .

**4) Multi\_enters in the state :**

As all states are dependant on enter signal (user\_input) , pressign enter for more than one time makes design glitches and misbehaviour .  
Sol.

Adding a dead\_lock signal that disables (enter) signal effect for more than one time .(Enter condition is captured only one time )  
**5) Multi-clock domains :**  
 As waiting\_timer counts on posedge of slow\_clk while the whole system works on the system clock , Each state should wait till the waiting\_timer goes back down before making any transition to avoid glitches & misbehaviour .  
Sol.   
Adding a last\_cycle\_timer that counts a number of counts = DIVISOR (where Divisor = the dividing factor for the slow\_clk generator ) , It keeps counting until waiting\_timer goes back down .

# 

# 

## 

## 

#### 