	Academic Year:	**2022/2023**	**Term:**	**second term**	
	Course Code:	**Elective4**	**Course Title:**	**EDA**	

Cairo University

Faculty of Engineering

Electronics and Communications Engineering Department – 4th Year

ATM DESIGN & Verification PROJECT

Name	Sec	BN
احمد سامح عبدالصبور منصور	1	4
ايهاب مصطفى فرغلي صالح	1	46
بسنت عاطف جاد الكريم	1	47
عبدالرحمن احمد عواد محمود	2	43
مصطفى خالد فاروق عبدالشافي	4	18
منى منصور امين محمد	4	30

Table of Contents

Table of Figures:

List of Tables:

1. Project Description

1.1. Digital Design:

The project aims at practicing Digital Design by implementing the core of the **bank ATM** design. The student should assume all auxiliary devices like card handling, money counting, and timers exist. As well as assume account information like passwords, account numbers and balances exist locally with no need for database connection. ATM System Can contain the following auxiliaries:

- Card handling
- Language used
- Card password
- Timers
- Operation (Deposit – Withdraw – Balance service)
- Balance exists
- Deposit value
- Withdraw value

1.2. Digital Verification:

The project aims also at practicing Digital Verification by implementing environment Testbench that covers (Test Stimulus using variation of directed, constraint random), Self-Checking TB Using Reference Model from Step#1, define design properties or assertion using SVA, create coverage model using combination between cover properties, cover groups and enable code coverage and create coverage report for Statement, Branch, FSM Coverage.

2- Design Flow

2.1 System Architecture:

The system architecture was done using **visio** program. In the architecture, we assumed that user's data (passwords, balances) exists in a memory and we have 4 main modules:

• Card handling

• User interface

• Timer

• FSM

The architecture is shown in figure (1).

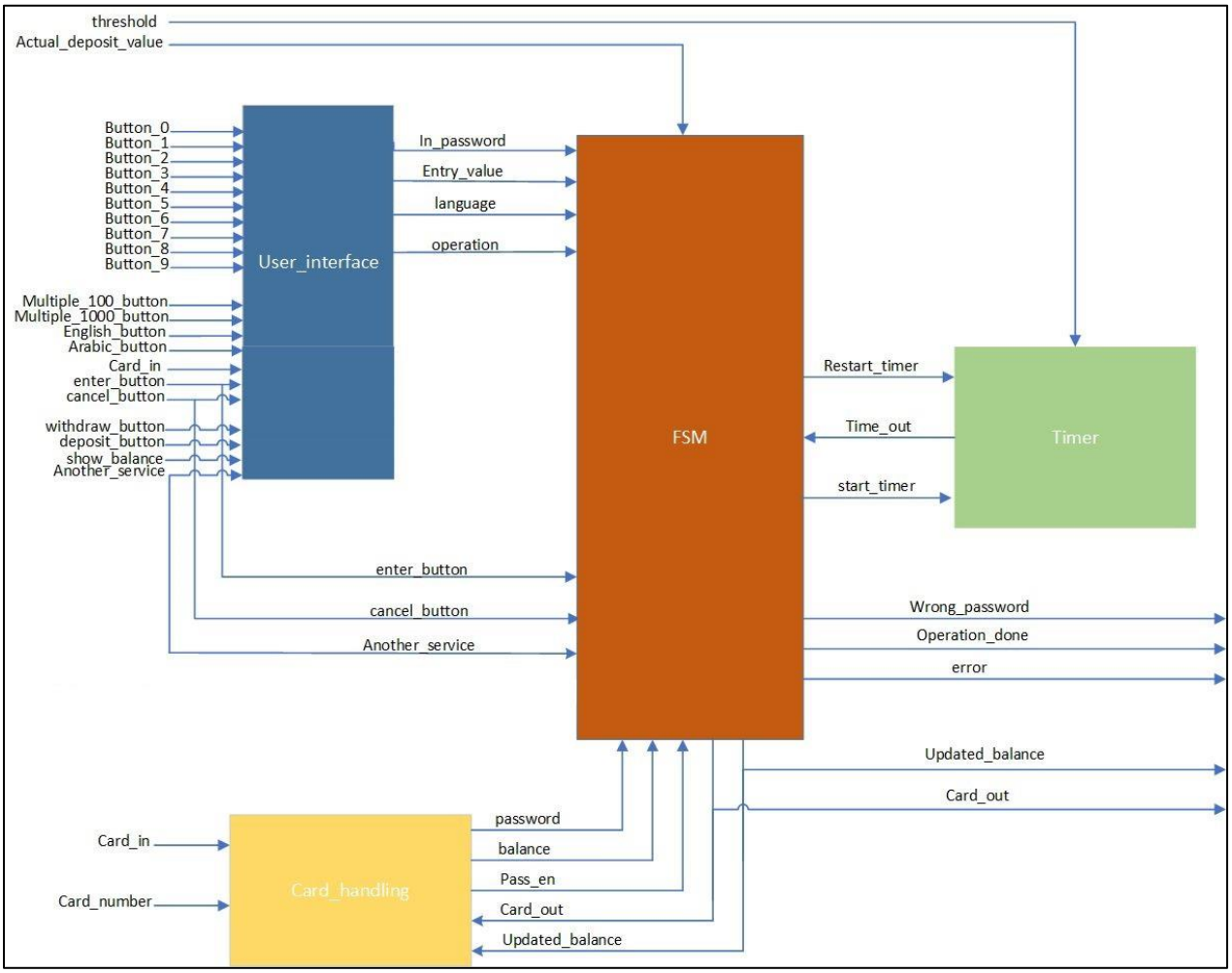


Figure 1 The architecture of Design

2.2 Signals:

Signal name	Direction	Signal width	Description
Card handling Signals			
Card_in	**IN**	**BIT**	Pulse of ATM response to Card number
Card_number	**IN**	**[5:0]**	The Card number
Password	**OUT**	**[15:0]**	The Card Password
Balance	**OUT**	**[19:0]**	The balance found in card after operations
pass_en	**OUT**	**BIT**	Pulse of ATM response to Card number if it
card_out	**IN**	**BIT**	Signal rises for one clock edge when system back to idle state
Updated_balance	**IN**	**[19:0]**	Updated value of balance during operation
operation_done	**IN**	**BIT**	Signal to check if operation done or fail
User interface signals			
Button_0 Button_1 Button_2 Button_3 Button_4 Button_5 Button_6 Button_7 Button_8 Button_9	**IN**	**BIT**	Buttons of ATM interface
Multiple_100_button Multiple_1000_button	**IN**	**BIT**	Buttons used to specify the value of deposit and withdraw
English_button Arabic_button	**IN**	**BIT**	Buttons to choose the language between Arabic / English
Card_in	**IN**	**BIT**	Pulse of ATM response to Card number
enter_button cancel_button	**IN**	**BIT**	Enter button of ATM interface Cancel button of ATM interface
Withdraw_button Deposit_button Show_balance Another_service	**IN**	**BIT**	Buttons to choose Withdraw/Deposit/ShowBalance/AnotherService operation
In_password	**OUT**	**[15:0]**	The entered password
Entry_value	**OUT**	**[19:0]**	The value specified for Withdraw/Deposit operation

Language	**OUT**	**[1:0]**	The chosen language
Operation	**OUT**	**[1:0]**	The chosen operation
Timer signals			
Threshold	**IN**	**[31:0]**	Timer for time out of ATM
Restart_timer	**IN**	**BIT**	Button to restart the timer
start_timer	**IN**	**BIT**	Button to start the timer
time_out	**OUT**	**BIT**	Signal make ATM goes to IDLE
FSM signals			
In_password	**IN**	**[15:0]**	The entered password
Entry_value	**IN**	**[19:0]**	The value specified for Withdraw/Deposit operation
Language	**IN**	**[1:0]**	The chosen language
Operation	**IN**	**[1:0]**	The chosen operation
enter_button cancel_button	**IN**	**BIT**	Enter button of ATM interface Cancel button of ATM interface
Actual_deposit_value	**IN**	**[19:0]**	The value of actual deposit which user entered
Another_service	**IN**	**BIT**	Button to use another service after finishing operation
Password	**IN**	**[15:0]**	The entered password
Balance	**IN**	**[19:0]**	The balance found in card after operations
Pass_en	**IN**	**BIT**	Pulse of ATM response to Card number
Card_out	**OUT**	**BIT**	Signal rises for one clock edge when system back to idle state
Updated_balance	**OUT**	**[19:0]**	The updated value of balance during operation
Time_out	**IN**	**BIT**	Signal make ATM goes to IDLE
Start_timer	**OUT**	**BIT**	Signal make ATM goes to IDLE
Restart_timer	**OUT**	**BIT**	Signal to restart the timer
Wrong_password	**OUT**	**BIT**	Signal to check if password pass or fail
Operation_done	**OUT**	**BIT**	Signal to check if operation pass or fail

error	**OUT**	**BIT**	Signal rises when error happens like (password is fail / entry value for withdraw larger than balance)

2.3 FSM Diagram of the system

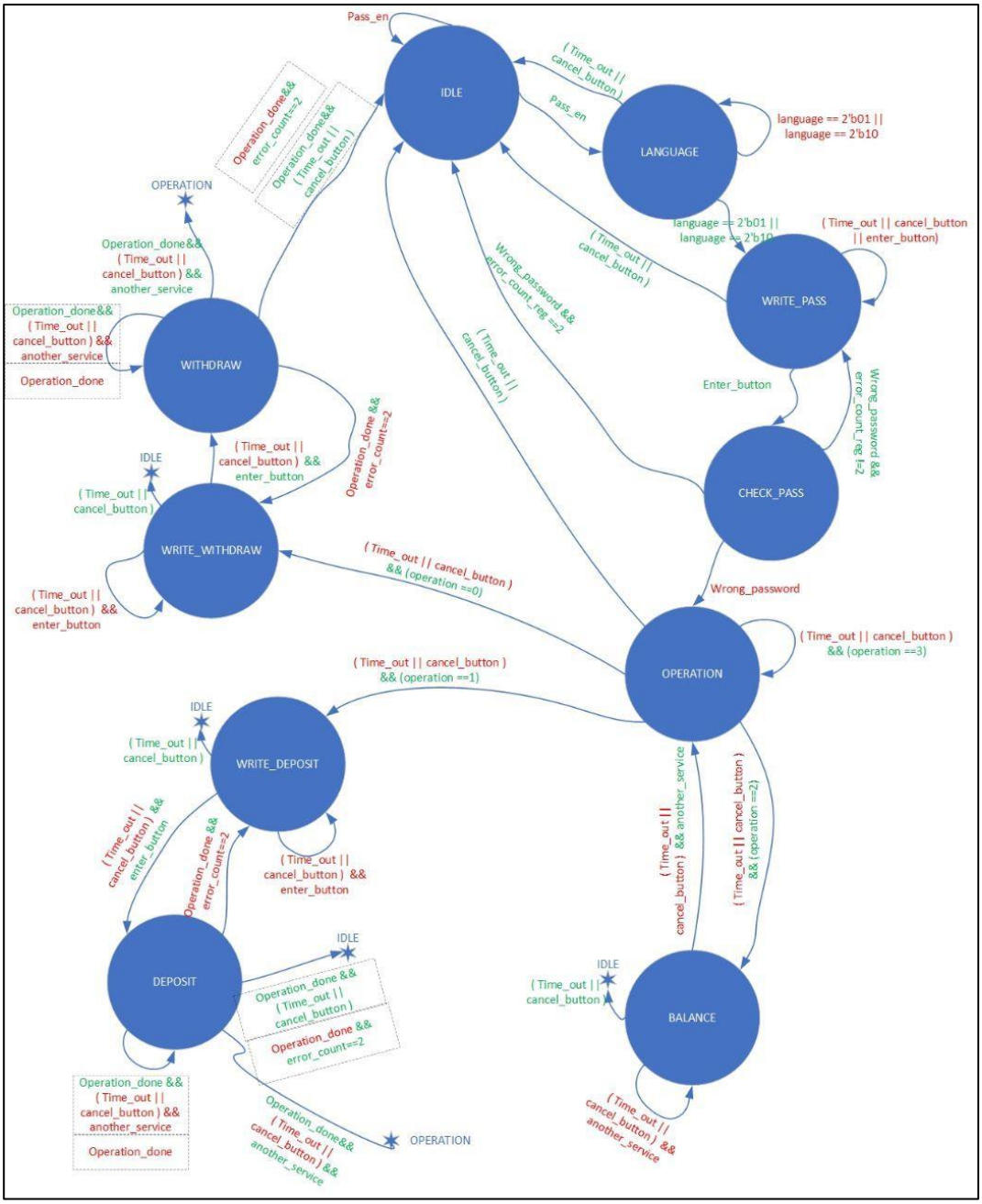


Figure 2 FSM Diagram

Table 1:FSM Transition

State	Transition	Signals
IDLE	IDLE → IDLE IDLE → LANGUAGE	pass_en=0 pass_en=1
LANGUAGE	LANGUAGE → IDLE LANGUAGE → WRITE_PASS LANGUAGE → LANGUAGE	time_out=1 , cancel=1 language = 2'b01 or 2'b10 Otherwise
WRITE_PASS	WRITE_PASS → IDLE WRITE_PASS → CHEK_PASS WRITE_PASS → WRITE_PASS	time_out=1 , cancel=1 enter_button=1 Otherwise
CHECK_PASS	CHECK_PASS → IDLE CHECK_PASS → WRITE_PASS CHECK_PASS → OPERATION	wrong_pass=1 && error_count_reg=2'b10 wrong_pass=1 && error_count_reg < 2'b10 Otherwise
OPERATION	OPERATION → IDLE OPERATION → WRITE_WITHDRAW OPERATION → WRITE_DEPOSIT OPERATION → BALANCE OPERATION → OPERATION	time_out=1 , cancel=1 operation='b00 operation='b01 operation='b10 Otherwise
WRITE_WITHDRAW	WRITE_WITHDRAW → IDLE WRITE_WITHDRAW → WITHDRAW WRITE_WITHDRAW → WRITE_WITHDRAW	time_out=1 , cancel=1 enter_button=1 Otherwise
WITHDRAW	WITHDRAW → IDLE WITHDRAW → OPERATION WITHDRAW → WITHDRAW WITHDRAW → WRITE_WITHDRAW	time_out=1 , cancel=1 , operation_done=1 , (error_count_reg =2'b10 && operation_done=0) another_service=1 , operation_done=1 another_service=0 , operation_done=1 Otherwise
WRITE_DEPOSIT	WRITE_DEPOSIT → IDLE WRITE_DEPOSIT → DEPOSIT WRITE_DEPOSIT → WRITE_DEPOSIT	time_out=1 , cancel=1 enter_button=1 Otherwise

DEPOSIT	**DEPOSIT → IDLE** **DEPOSIT → OPERATION** **DEPOSIT → DEPOSIT** **DEPOSIT → WRITE_DEPOSIT**	**time_out=1 , cancel=1 , operation_done=1 , (error_count_reg =2'b10 && operation_done=0)** **another_service=1 , operation_done=1** **another_service=0 , operation_done=1** **Otherwise**
BALANCE	**BALANCE → IDLE** **BALANCE → OPERATION** **BALANCE → BALANCE**	**time_out=1 , cancel=1** **another_service=1** **Otherwise**

2.4 High level model of the system

This phase was done using MATLAB to make a model that describes the system behavior.

The MATLAB model is attached with the project files.

2.5 RTL phase of the system

This phase was done using Verilog to implement the design architecture.

The RTL code is attached with the project files.

3- Project Verification Flow

3.1 VERIFICATION PLAN

3.1.1 UVM-System Verilog Architecture:

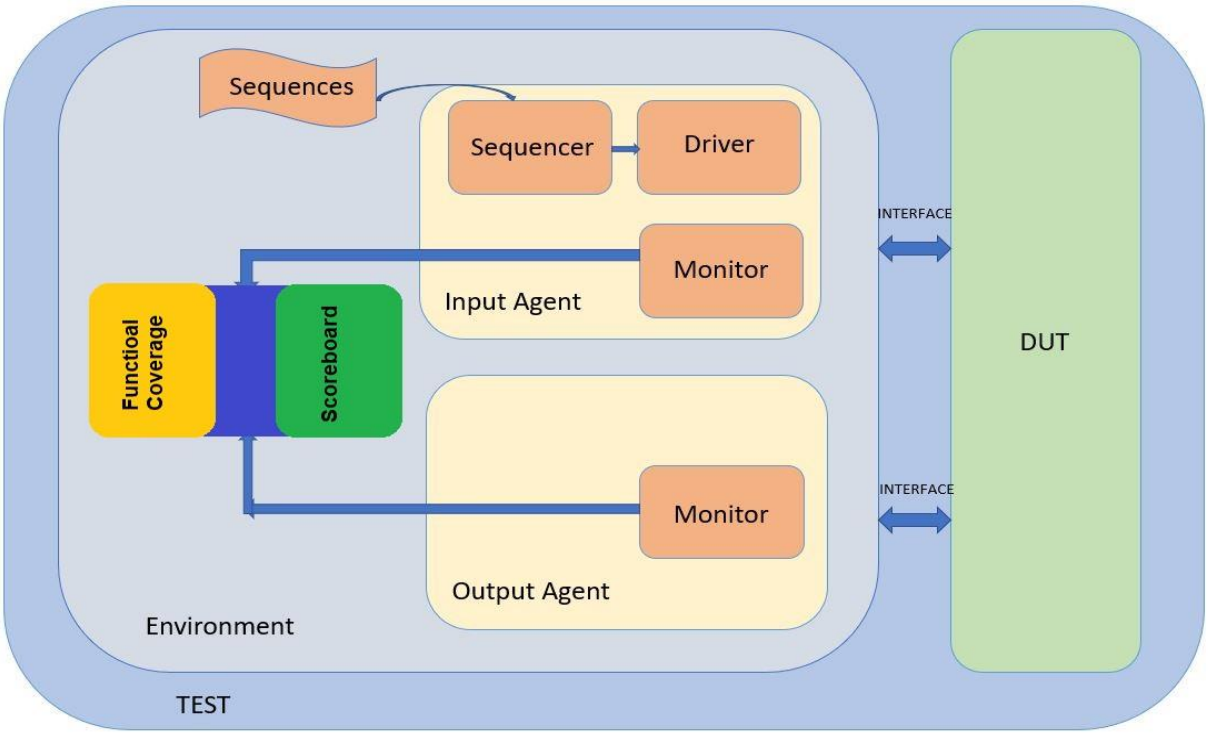


Figure 3 UVM Architecture

3.1.2 Sequence items:

ATM_sequence_item

3.1.3 Sequences:

ATM_sequence_operation_toIDLE

ATM_sequence_anotherService

ATM_sequence_errors

ATM_sequence_timeouts

3.1.4 Virtual sequences:

We will create some testes that will cover some of our sequences.

• **Test_all_mode**: (test all modes for all operation modes by all the following scenarios).

• **Test_Scenario1:**

(Withdraw → IDLE → Deposit → IDLE → Show Balance → IDLE).

• **Test_Scenario2:**

(Withdraw → Another Service → Deposit→ Another Service →
Show Balance)

• **Test_Scenario3**:

(Deposit → Three Errors →IDLE → Wrong Password → IDLE →

Withdraw → One Error (Success) → IDLE → Withdraw → Three Errors → IDLE).

• **Test_Scenario4**:

(IDLE → Language → IDLE (timeout)→ ------ →Write pass→IDLE

(timeout) →------ → Operation → IDLE (timeout) → ------ →Write Withdraw →

IDLE (timeout) → ---- →Withdraw → IDLE (timeout) → -------→Write Deposit → IDLE

(timeout). → ------→ Deposit → IDLE (timeout) →------- → Balance → IDLE (timeout)

3.2 Interface signals:

Table 2: Interface signals

Signal name	Dir "driver of active agent"	Dir "monitor of active agent"	Signal width	Description
Clk	-	-	**Bit**	System clock
Rst	**OUT**	**IN**	**Bit**	System reset
Threshold	**OUT**	**IN**	**[31:0]**	Timer for time out of ATM
card number	**OUT**	**IN**	**[5:0]**	Card number of user
card_in	**OUT**	**IN**	**Bit**	Pulse of ATM response to Card number
button_0 button_1 button_2 button_3 button_4 button_5 button_6 button_7 button_8 button_9	**OUT**	**IN**	**Bit**	Buttons of ATM interface
enter_button	**OUT**	**IN**	**Bit**	Enter button of ATM interface
cancel_button	**OUT**	**IN**	**Bit**	Cancel button of ATM interface
withdraw_button	**OUT**	**IN**	**Bit**	Button for withdraw operation
deposit_button	**OUT**	**IN**	**Bit**	Button for deposit operation
show_balance	**OUT**	**IN**	**Bit**	Button to show balance operation
another_service	**OUT**	**IN**	**Bit**	Button to do another operation
English_button	**OUT**	**IN**	**Bit**	Button to use English language
Arabic_button	**OUT**	**IN**	**Bit**	Button to use Arabic language
multiple_100_butt multiple_1000_but	**OUT**	**IN**	**Bit**	Buttons used to specify the value of deposit and withdraw
actual_deposit_value	**OUT**	**IN**	**[19:0]**	The value of actual deposit value

updated_balance	**OUT**	**IN**	**[19:0]**	The value of updated balance
wrong_password	**OUT**	**IN**	**Bit**	Signal show if password is wrong
operation_done	**OUT**	**IN**	**Bit**	Signal show if operation done—
Error	**OUT**	**IN**	**Bit**	Signal show if there is error
card_out	**OUT**	**IN**	**Bit**	Signal rises for one clock edge when system back to idle state

3.3 Constraints Randomization:

Bit entry_int (Random signal) → Constraint inside [1:9]

- Using it for randomization entry deposit value.
- Using it for randomization entry withdraw value.
- Using it for randomization entry password buttons.

3.4 Assertions:

3.4.1 Assertions table:

Table 3:Assertions

Assertion	Sequence	Status
Cancel_Cardout	After Cancel_Button is pressed -> Card_out =1 or Card_out = 0 and Idle state	**Pass**
Timeout_cardout	After Timer timeout ->Card_out = 1 or Card_out =0 and Idle state	**Pass**
Cancel_starttimer	After Cancel_Button is pressed => Start_timer=0 (Timer stopped)	**Pass**
Timeout_Starttimer	After Timer Timeout is pressed => Start_timer=0 (Timer stopped)	**Pass**
Pass_checkpass	checking transition IDLE > Language > Write_Password > Check_Password -> Start_timer = 0 (Timer Stopped while checking password)	**Pass**
Error_idle	Checks that 3 errors return to Idle state -> (Current_State == 0)	**Pass**
Withdraw	checking withdraw operation until Enter button -> Enter_button = 1	**Pass**
withdraw_done	checking that withdraw operation is done successfully -> Operation_done = 1;	**Pass**

withdraw_another	checking withdraw operation with another service (given that cancel button not pressed or timer has timed out) -> (next_state == 4)	**Pass**
Deposit	checking Deposit operation until Enter button -> Enter_button = 1	**Pass**
Deposit_done	checking that Deposit operation is done successfully -> Operation_done = 1;	**Pass**
Deposit_another	checking Deposit operation with another service (given that cancel button not pressed or timer has timed out) -> (next_state == 4)	**Pass**
ShowBalance	checking ShowBalance operation until Enter button -> Enter_button = 1	**Pass**
ShowBalance _done	checking that ShowBalance operation is done successfully -> Operation_done = 1;	**Pass**
ShowBalance _another	checking ShowBalance operation with another service (given that cancel button not pressed or timer has timed out) -> (next_state == 4)	**Pass**
balance_equal	Checking that balance gets updated after every deposit and withdraw -> (UpdatedBalance==Balance_memory [Card_number]) == 1	**Pass**
Wrong_password	Checking if (in_password != user_password) that Wrong_Passwordis raised -> Wrong_Password = 1	**Pass**

3.4.2 Assertions Waveform & Percentage & Pass

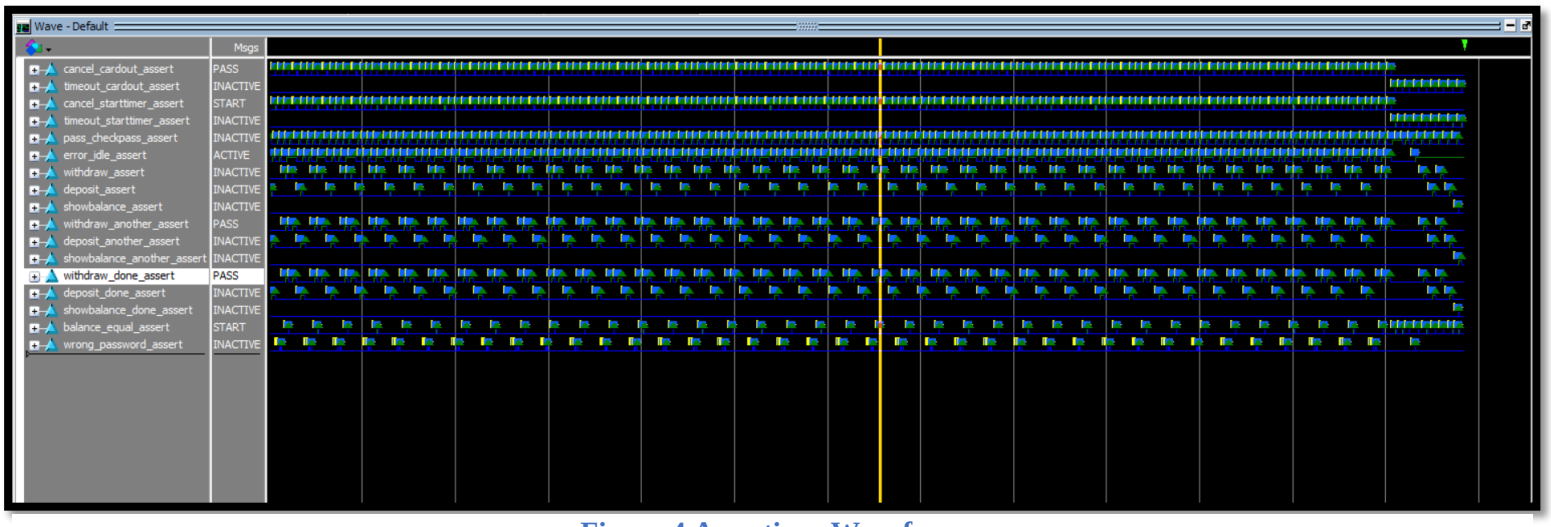
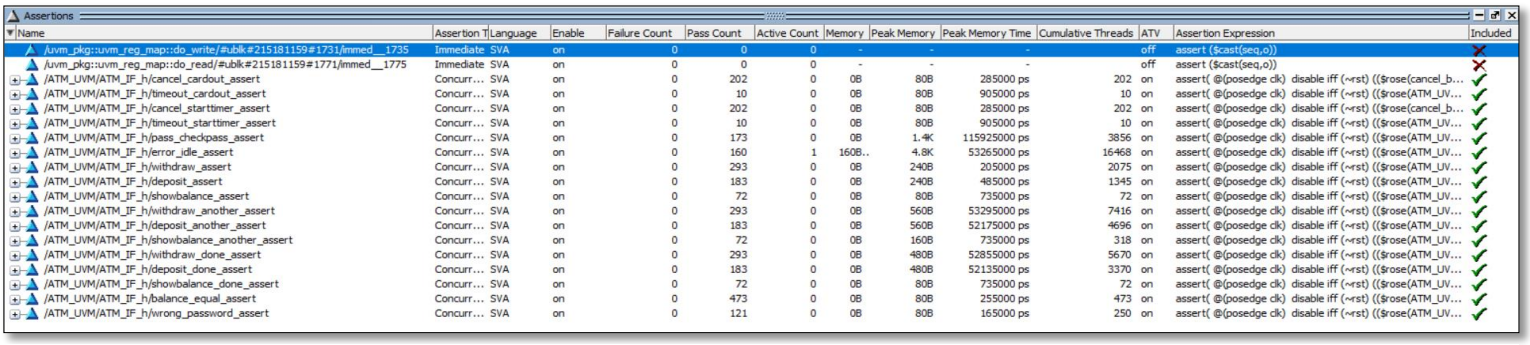


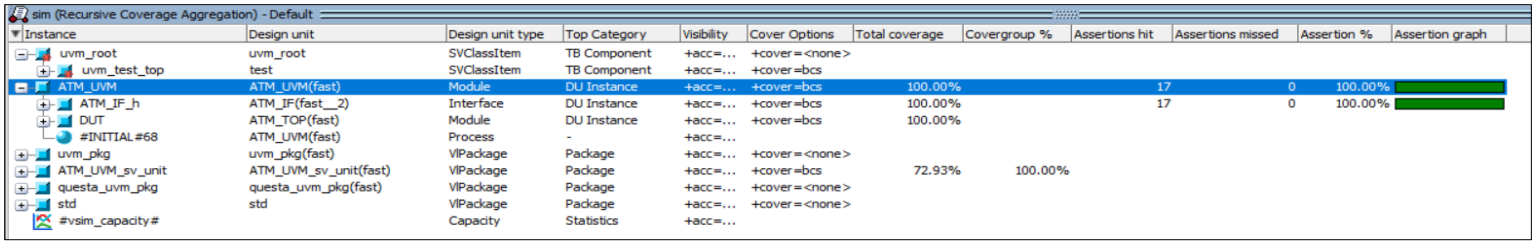
Figure 4 Assertions Waveform



Assertions

Name	Assertion T	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Included
/uvm_pkg::uvm_reg_map::do_write/#ublk#215181159#1731/immed__1735	Immediate	SVA	on	0	0	0	-	-	-		off	assert ($cast(seq,o))	✗
/uvm_pkg::uvm_reg_map::do_read/#ublk#215181159#1771/immed__1775	Immediate	SVA	on	0	0	0	-	-	-		off	assert ($cast(seq,o))	✗
/ATM_UVM/ATM_IF_h/cancel_cardout_assert	Concurr...	SVA	on	0	202	0	0B	80B	285000 ps	202	on	assert(@(posedge clk) disable iff (~rst) (($rose(cancel_b...	✓
/ATM_UVM/ATM_IF_h/timeout_cardout_assert	Concurr...	SVA	on	0	10	0	0B	80B	905000 ps	10	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/cancel_starttimer_assert	Concurr...	SVA	on	0	202	0	0B	80B	285000 ps	202	on	assert(@(posedge clk) disable iff (~rst) (($rose(cancel_b...	✓
/ATM_UVM/ATM_IF_h/timeout_starttimer_assert	Concurr...	SVA	on	0	10	0	0B	80B	905000 ps	10	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/pass_checkpass_assert	Concurr...	SVA	on	0	173	0	0B	1.4K	115925000 ps	3856	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/error_idle_assert	Concurr...	SVA	on	0	160	1	160B..	4.8K	53265000 ps	16468	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/withdraw_assert	Concurr...	SVA	on	0	293	0	0B	240B	205000 ps	2075	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/deposit_assert	Concurr...	SVA	on	0	183	0	0B	240B	485000 ps	1345	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/showbalance_assert	Concurr...	SVA	on	0	72	0	0B	80B	735000 ps	72	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/withdraw_another_assert	Concurr...	SVA	on	0	293	0	0B	560B	53295000 ps	7416	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/deposit_another_assert	Concurr...	SVA	on	0	183	0	0B	560B	52175000 ps	4696	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/showbalance_another_assert	Concurr...	SVA	on	0	72	0	0B	160B	735000 ps	318	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/withdraw_done_assert	Concurr...	SVA	on	0	293	0	0B	480B	52855000 ps	5670	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/deposit_done_assert	Concurr...	SVA	on	0	183	0	0B	480B	52135000 ps	3370	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/showbalance_done_assert	Concurr...	SVA	on	0	72	0	0B	80B	735000 ps	72	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/balance_equal_assert	Concurr...	SVA	on	0	473	0	0B	80B	255000 ps	473	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓
/ATM_UVM/ATM_IF_h/wrong_password_assert	Concurr...	SVA	on	0	121	0	0B	80B	165000 ps	250	on	assert(@(posedge clk) disable iff (~rst) (($rose(ATM_UV...	✓

Figure 5 Assertions pass counts



sim (Recursive Coverage Aggregation) - Default

Instance	Design unit	Design unit type	Top Category	Visibility	Cover Options	Total coverage	Covergroup %	Assertions hit	Assertions missed	Assertion %	Assertion graph
uvm_root	uvm_root	SVClassItem	TB Component	+acc=...	+cover=<none>						
uvm_test_top	test	SVClassItem	TB Component	+acc=...	+cover=bcs						
ATM_UVM	ATM_UVM(fast)	Module	DU Instance	+acc=...	+cover=bcs	100.00%		17	0	100.00%	
ATM_IF_h	ATM_IF(fast__2)	Interface	DU Instance	+acc=...	+cover=bcs	100.00%		17	0	100.00%	
DUT	ATM_TOP(fast)	Module	DU Instance	+acc=...	+cover=bcs	100.00%					
#INITIAL#68	ATM_UVM(fast)	Process	-	+acc=...							
uvm_pkg	uvm_pkg(fast)	VlPackage	Package	+acc=...	+cover=<none>						
ATM_UVM_sv_unit	ATM_UVM_sv_unit(fast)	VlPackage	Package	+acc=...	+cover=bcs	72.93%	100.00%				
questa_uvm_pkg	questa_uvm_pkg(fast)	VlPackage	Package	+acc=...	+cover=<none>						
std	std	VlPackage	Package	+acc=...	+cover=<none>						
#vsim_capacity#		Capacity	Statistics	+acc=...							

Figure 6 Assertions Precentages

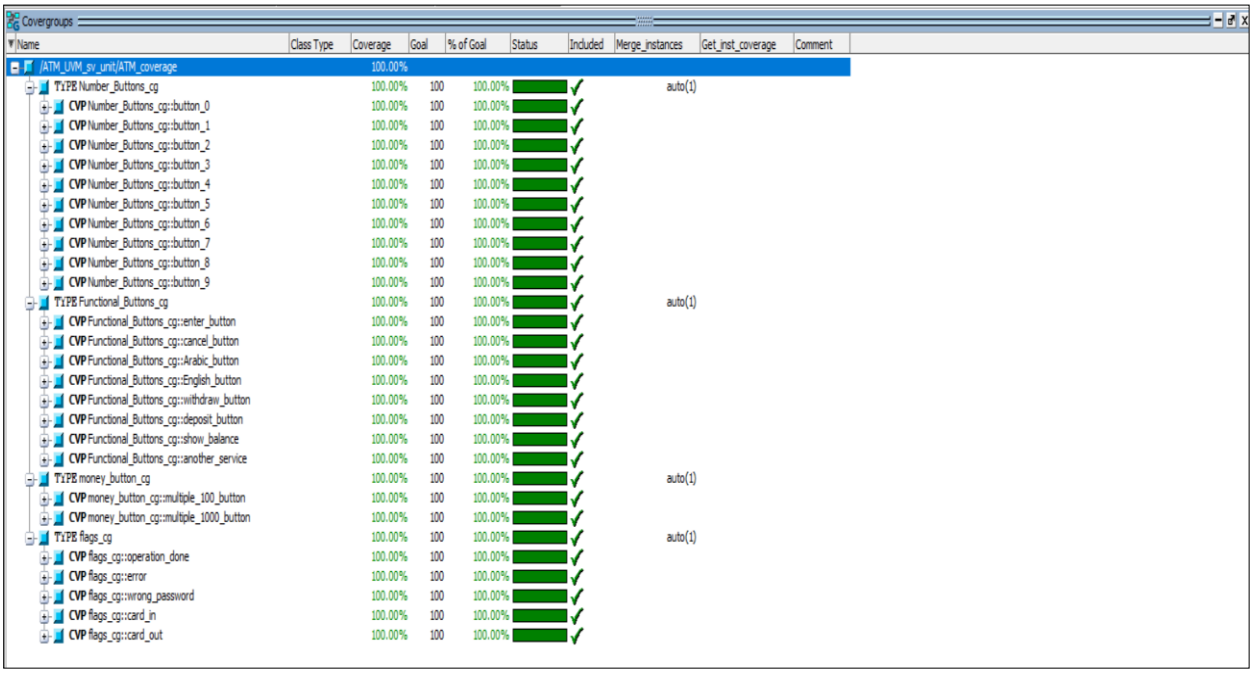
3.5 Coverage

3.5.1 Cover Groups:

Table 4:Cover Groups

❖ **Number_Buttons_cg** • button_0 • button_1 • button_2 • button_3 • button_4 • button_5 • button_6 • button_7 • button_8 • button_9	❖ **Functional_Buttons_cg** • enter_button • cancel_button • Arabic_button • English_button • withdraw_button • deposit_button • show_balance • another_service
❖ **money_button_cg** • multiple_100_button • multiple_1000_button	❖ **flags_cg** • operation_done • error • wrong_password • card_in • card_out

3.5.2 Functional COVERAGE:

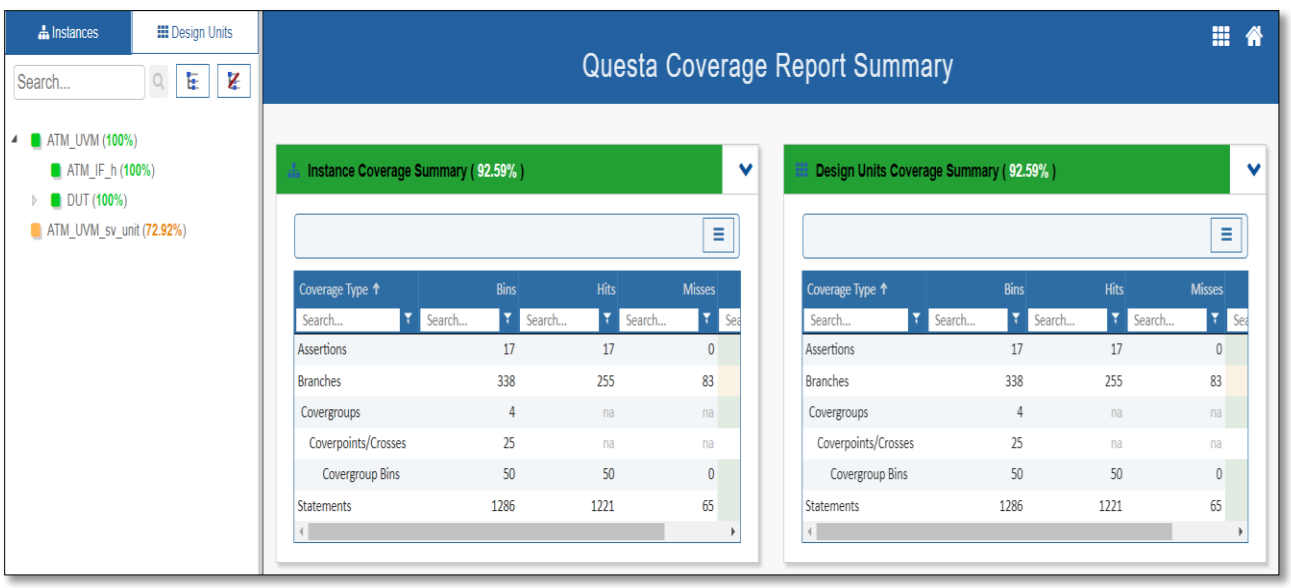


Covergroups

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances	Get_inst_coverage	Comment
/ATM_UVM_sv_unit/ATM_coverage		100.00%							
TYPE Number_Buttons_cg		100.00%	100	100.00%			auto(1)		
CVP Number_Buttons_cg::button_0		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_1		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_2		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_3		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_4		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_5		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_6		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_7		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_8		100.00%	100	100.00%					
CVP Number_Buttons_cg::button_9		100.00%	100	100.00%					
TYPE Functional_Buttons_cg		100.00%	100	100.00%			auto(1)		
CVP Functional_Buttons_cg::enter_button		100.00%	100	100.00%					
CVP Functional_Buttons_cg::cancel_button		100.00%	100	100.00%					
CVP Functional_Buttons_cg::Arabic_button		100.00%	100	100.00%					
CVP Functional_Buttons_cg::English_button		100.00%	100	100.00%					
CVP Functional_Buttons_cg::withdraw_button		100.00%	100	100.00%					
CVP Functional_Buttons_cg::deposit_button		100.00%	100	100.00%					
CVP Functional_Buttons_cg::show_balance		100.00%	100	100.00%					
CVP Functional_Buttons_cg::another_service		100.00%	100	100.00%					
TYPE money_button_cg		100.00%	100	100.00%			auto(1)		
CVP money_button_cg::multiple_100_button		100.00%	100	100.00%					
CVP money_button_cg::multiple_1000_button		100.00%	100	100.00%					
TYPE flags_cg		100.00%	100	100.00%			auto(1)		
CVP flags_cg::operation_done		100.00%	100	100.00%					
CVP flags_cg::error		100.00%	100	100.00%					
CVP flags_cg::wrong_password		100.00%	100	100.00%					
CVP flags_cg::card_in		100.00%	100	100.00%					
CVP flags_cg::card_out		100.00%	100	100.00%					

Figure 7 Functional Coverage

3.5.3 CODE COVERAGE:



Instances | Design Units

Search...

- ATM_UVM (100%)
  - ATM_IF_h (100%)
  - DUT (100%)
  - ATM_UVM_sv_unit (72.92%)

Questa Coverage Report Summary

Instance Coverage Summary (92.59%)

Coverage Type	Bins	Hits	Misses
Assertions	17	17	0
Branches	338	255	83
Covergroups	4	na	na
Coverpoints/Crosses	25	na	na
Covergroup Bins	50	50	0
Statements	1286	1221	65

Design Units Coverage Summary (92.59%)

Coverage Type	Bins	Hits	Misses
Assertions	17	17	0
Branches	338	255	83
Covergroups	4	na	na
Coverpoints/Crosses	25	na	na
Covergroup Bins	50	50	0
Statements	1286	1221	65

Figure 8 Code Coverage

3.5.4 FSM COVERAGE:

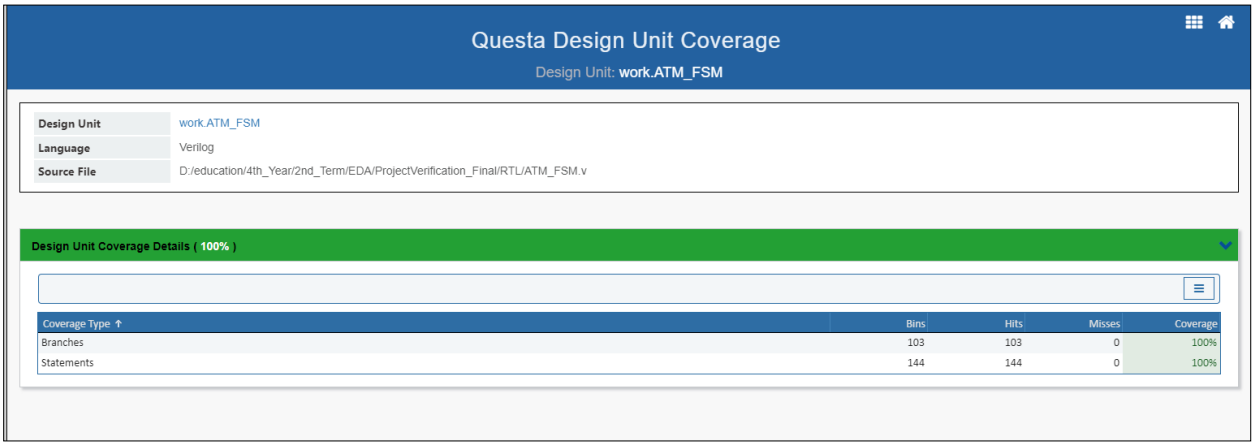


Figure 9 FSM Coverage

3.6 Scoreboard Self checking:

Table 5: Checks in Scoreboard

Statement	Description	Signals
Deposit Mode operation checking	Checks that the deposit value through serial line is being added to the user's balance and the updated balance is equivalent to the user's balance after the operation is done.	actual_deposit_value, updated_balance, operation_done
Withdraw Mode operation checking	Checks that the withdraw value through serial line is being subtracted from the user's balance and the updated balance is equivalent to the user's balance after the operation is done.	withdraw_button, multiple_100_button, multiple_1000_button, updated_balance, operation_done

4. APPENDIX:

The GitHub repository link:

https://github.com/EngMostafaKhaled/ATM---based-bank-system-Design-verification