Team 7

Verification Plan

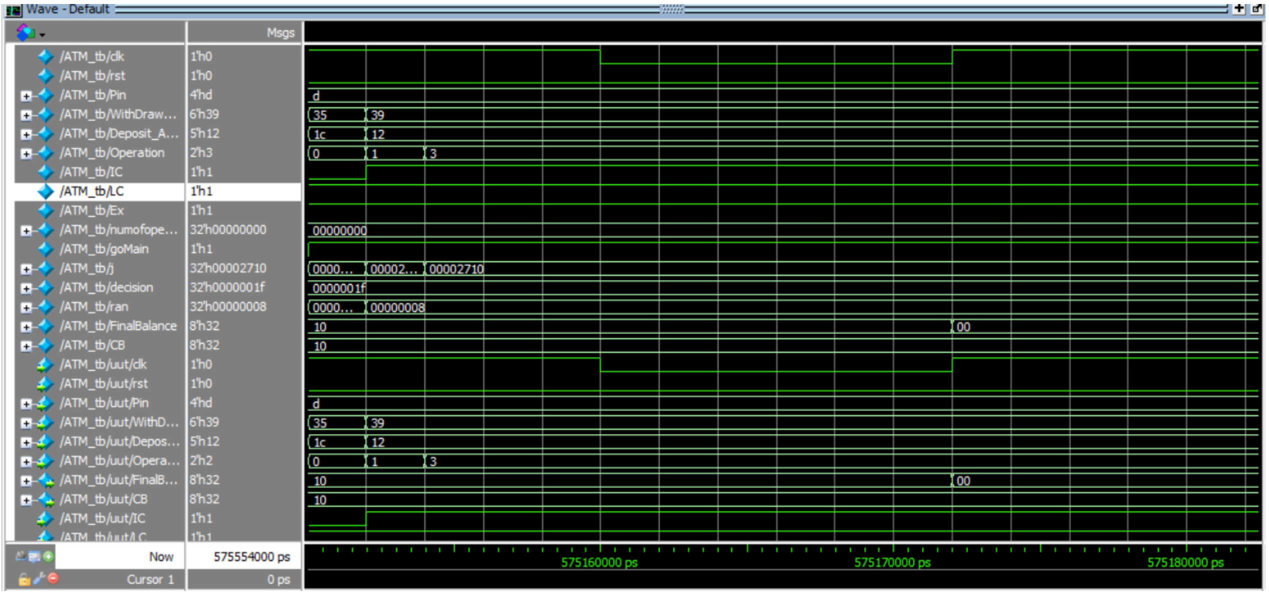
1) Directed Testing:

Directed testing is used to verify that an interface behaves as expected in response to valid/invalid transactions. Although, they are only two tests as the Randomized one cover all the states.

The code:

```
#50 rst = 1;
#50 rst = 0;

#50 IC = 1; LC = 1;
#50 Pin = 4'b1101;
    Operation = 2;
    rst = 1;

#50 rst = 0;

#50 IC = 1; LC = 1;
#50 Pin = 4'b1101;
    Operation = 2;
#50 Operation = 3;
```

The Wave Forms:



	Msgs
/ATM_tb/clk	1'h0
/ATM_tb/rst	1'h0
/ATM_tb/Pin	4'hd
/ATM_tb/WithDraw...	6'h39
/ATM_tb/Deposit_A...	5'h12
/ATM_tb/Operation	2'h3
/ATM_tb/IC	1'h1
/ATM_tb/LC	1'h1
/ATM_tb/Ex	1'h1
/ATM_tb/numofope...	32'h00000000
/ATM_tb/goMain	1'h1
/ATM_tb/j	32'h00002710
/ATM_tb/decision	32'h0000001f
/ATM_tb/ran	32'h00000008
/ATM_tb/FinalBalance	8'h32
/ATM_tb/CB	8'h32
/ATM_tb/uut/clk	1'h0
/ATM_tb/uut/rst	1'h0
/ATM_tb/uut/Pin	4'hd
/ATM_tb/uut/WithD...	6'h39
/ATM_tb/uut/Depos...	5'h12
/ATM_tb/uut/Opera...	2'h2
/ATM_tb/uut/FinalB...	8'h32
/ATM_tb/uut/CB	8'h32
/ATM_tb/uut/IC	1'h1
/ATM_tb/uut/LC	1'h1
Now	575554000 ps
Cursor 1	0 ps

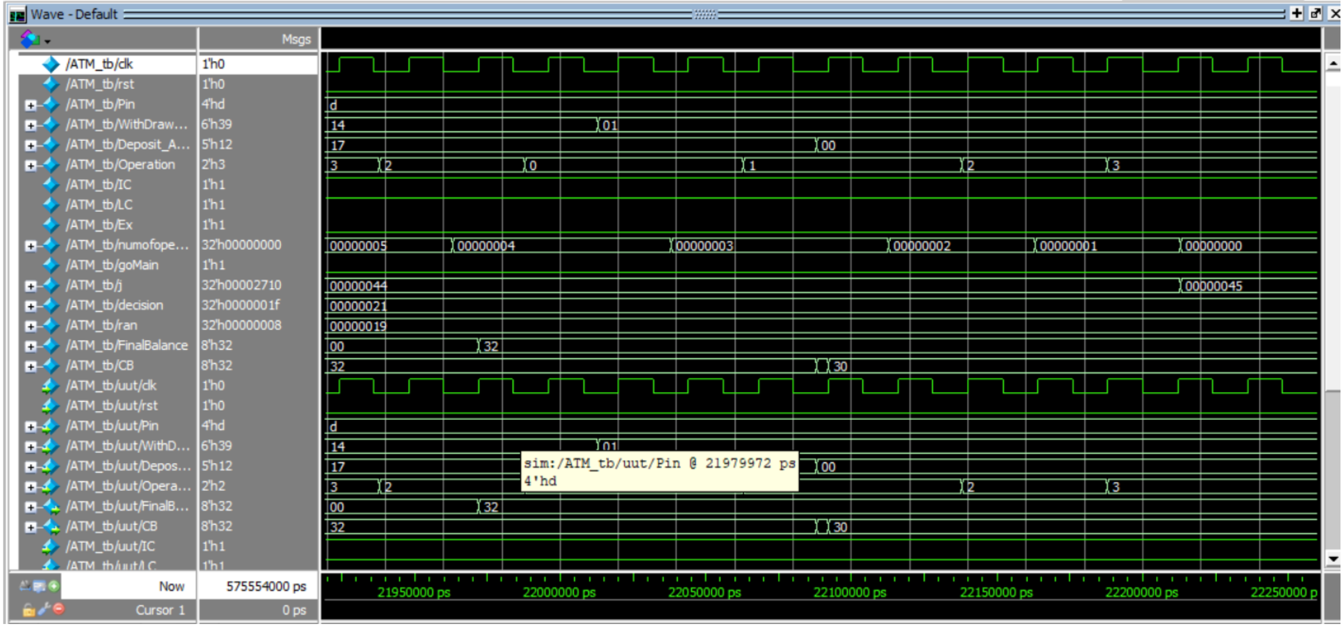
2) Constrained Randomized test

CRV is a methodology that allows you to constrain your stimulus to better target a design function, thereby allowing you to reach your coverage goal faster with accuracy. It requires the random generation of input stimuli that obey a set of declaratively specified input constraints, which are then applied to validate given design properties by simulation.

First random test code:

```
/***
directed Randomized generator to test almost all possiblites of the desgin
the flow is customized to not reset the design, although it test all other Statments and braches
not including reseting or card inserting or language choosen

each iterations is like one client doing a randomized number of operations
the last operation is to exit the ATM as default
***/

for(j=0;j<1000;j=j+1)
begin
   #25 IC = 1;            //default IC
   #25 LC = 1;            //default LC
   #25 Pin = 4'b1101;     // 4/30 probability to randomize the pin
   ran = {$random}%30;
   if(ran > 25)
      Pin = {$random}%16;

   numofoperations = 1 + {$random}%5;     //varied from 1 to 5
   #2;
   while(numofoperations > 2'b00)         //loop till excuating all the operations
   begin
      if(numofoperations == 1'b1)        //the last operation is to exit as defualt
         #25 Operation = 2'b11;          //exit operation and return to reset
      else
      begin
         #25 Operation = {$random}%3;    //operation from 0 ot 2
         if(Operation == 2'b00)          //if withdraw
         begin
            //first, randomize the withdraw amount
            #25 WithDraw_Amount = {$random}%64;

            while(WithDraw_Amount>CB)   //if not valid
            begin
               //randomized range from 0 to 39 to decide to enter another value or to leave
               #50 decision = {$random}%40;
               if(decision < 30)                    //  3/4 percent to enter another value
            #25 WithDraw_Amount = {$random}%64;
          else
          begin
            #25 goMain = 1'b1;          // 1/4 percent to return to the main menu
              WithDraw_Amount = 0;
            #25 goMain = 1'b0;
          end
        end
      end
      else if(Operation == 2'b01)               // if to deposit
      begin
          #25 Deposit_Amount = {$random}%32;       // randomize the deposit amount
      end
    end

    #25 numofoperations = numofoperations - 3'b001;    //decrement the number of remaining operations
  end
end
```

WaveForms:



	Msgs
/ATM_tb/clk	1'h0
/ATM_tb/rst	1'h0
/ATM_tb/Pin	4'hd
/ATM_tb/WithDraw...	6'h39
/ATM_tb/Deposit_A...	5'h12
/ATM_tb/Operation	2'h3
/ATM_tb/IC	1'h1
/ATM_tb/LC	1'h1
/ATM_tb/Ex	1'h1
/ATM_tb/numofope...	32'h00000000
/ATM_tb/goMain	1'h1
/ATM_tb/j	32'h00002710
/ATM_tb/decision	32'h0000001f
/ATM_tb/ran	32'h00000008
/ATM_tb/FinalBalance	8'h32
/ATM_tb/CB	8'h32
/ATM_tb/uut/clk	1'h0
/ATM_tb/uut/rst	1'h0
/ATM_tb/uut/Pin	4'hd
/ATM_tb/uut/WithD...	6'h39
/ATM_tb/uut/Depos...	5'h12
/ATM_tb/uut/Opera...	2'h2
/ATM_tb/uut/FinalB...	8'h32
/ATM_tb/uut/CB	8'h32
/ATM_tb/uut/IC	1'h1
/ATM_tb/uut/LC	1'h1
Now	575554000 ps
Cursor 1	0 ps

```
sim:/ATM_tb/uut/Pin @ 21979972 ps
4'hd
```

Second random test code:

```
//randomize the IC, LC, reseting in different states


#1 rst = 1;
#1 rst = 0;

for(j=0;j<10000;j=j+1)
begin
```

```
    rst = 0;
    ran = {$random}%30;
    #2  IC = {$random}%2;
    #2  if(ran > 20)
            rst = 1;
    #2  LC = {$random}%2;
    #2  if(ran > 20)
            rst = 1;
    #2  Pin = 4'b1101;
        if(ran > 10)
            Pin = {$random}%16;
    #2  if(ran > 20)
            rst = 1;

    #2  Operation = {$random}%3;
        ran = {$random}%30;
    #2  if(ran > 20)
            rst = 1;

    #2  WithDraw_Amount = {$random}%64;
        goMain = {$random}%2;
        Deposit_Amount = {$random}%32;
        ran = {$random}%30;
        #2  if(ran > 20)
            rst = 1;

    #2  Operation = 2'b01;
        ran = {$random}%30;
    #2  if(ran > 20)
            rst = 1;

end
```

Third random test code:

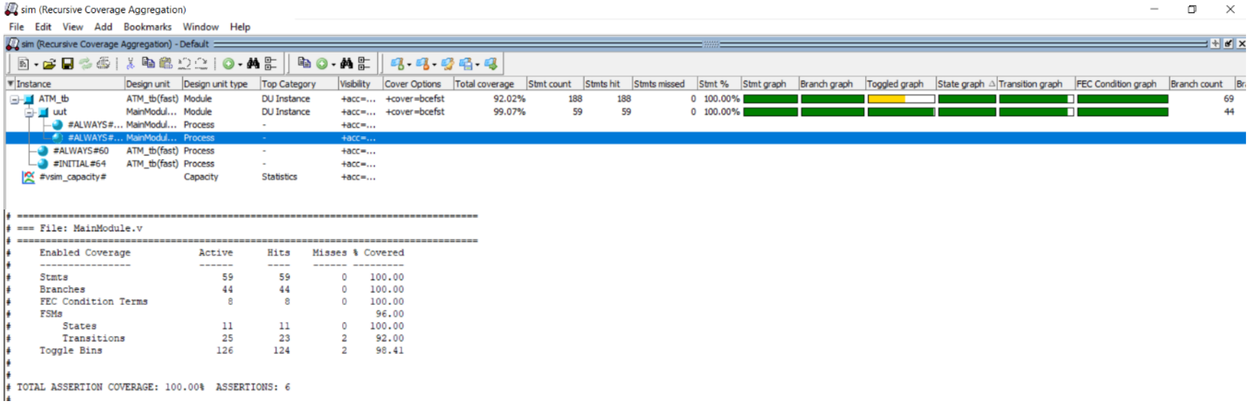
```
//Randomized test to suffle high range of state changes in the same cycle to increase coverage

#1 rst = 1;
#1 rst = 0;

for(j=0;j<10000;j=j+1)
begin
```

```
    rst = 0;
    ran = {$random}%30;
    IC = {$random}%2;
    LC = {$random}%2;
    Ex = {$random}%2;
    Pin = 4'b1101;
    if(ran > 10)
        Pin = {$random}%16;
    Operation = {$random}%3;
    WithDraw_Amount = {$random}%64;
    goMain = {$random}%2;
    Deposit_Amount = {$random}%32;

    #2  Operation = 2'b11;
end
```

3) Coverage Driven Verification

CDV: Is one of Main Critical Criteria to Judge the Effectiveness of Applied Verification Methodology. Coverage Helps us to Identify what Hasn’t Been Tested in the design

Overall coverage:



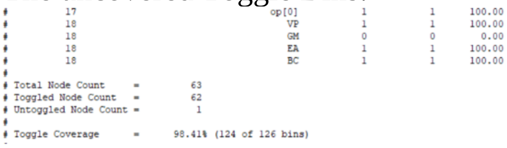
sim (Recursive Coverage Aggregation)
File
Edit
View
Add
Bookmarks
Window
Help
sim (Recursive Coverage Aggregation) - Default

Instance	Design unit	Design unit type	Top Category	Visibility	Cover Options	Total coverage	Stmt count	Stmts hit	Stmts missed	Stmt %	Stmt graph	Branch graph	Toggled graph	State graph	Transition graph	FEC Condition graph	Branch count	Br
ATM_tb	ATM_tb(fast)	Module	DU Instance	+acc=...	+cover=bcefst	92.02%	188	188	0	100.00%	🟩	🟩	🟨	🟩	🟩	🟩	69	
uut	MainModul...	Module	DU Instance	+acc=...	+cover=bcefst	99.07%	59	59	0	100.00%	🟩	🟩	🟩	🟩	🟩	🟩	44	
#ALWAYS#...	MainModul...	Process	-	+acc=...														
#ALWAYS#...	MainModul...	Process	-	+acc=...														
#ALWAYS#60	ATM_tb(fast)	Process	-	+acc=...														
#INITIAL#64	ATM_tb(fast)	Process	-	+acc=...														
#vsim_capacity#		Capacity	Statistics	+acc=...														

```
# ===============================================================================
# === File: MainModule.v
# ===============================================================================
#     Enabled Coverage              Active      Hits    Misses % Covered
#     ----------------              ------      ----    ------ ---------
#     Stmts                             59        59         0    100.00
#     Branches                          44        44         0    100.00
#     FEC Condition Terms                8         8         0    100.00
#     FSMs                                                         96.00
#         States                        11        11         0    100.00
#         Transitions                   25        23         2     92.00
#     Toggle Bins                      126       124         2     98.41
#
#
# TOTAL ASSERTION COVERAGE: 100.00%  ASSERTIONS: 6
#
```

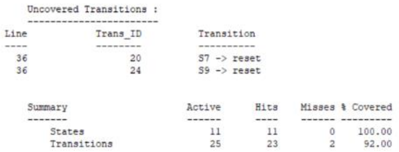
The main module reached 99.07%.

The uncovered Toggle Bins:



```
#          17                          op[0]         1          1      100.00
#          18                             VP         1          1      100.00
#          18                             GM         0          0        0.00
#          18                             EA         1          1      100.00
#          18                             BC         1          1      100.00
#
# Total Node Count     =         63
# Toggled Node Count   =         62
# Untoggled Node Count =          1
#
# Toggle Coverage      =      98.41% (124 of 126 bins)
```

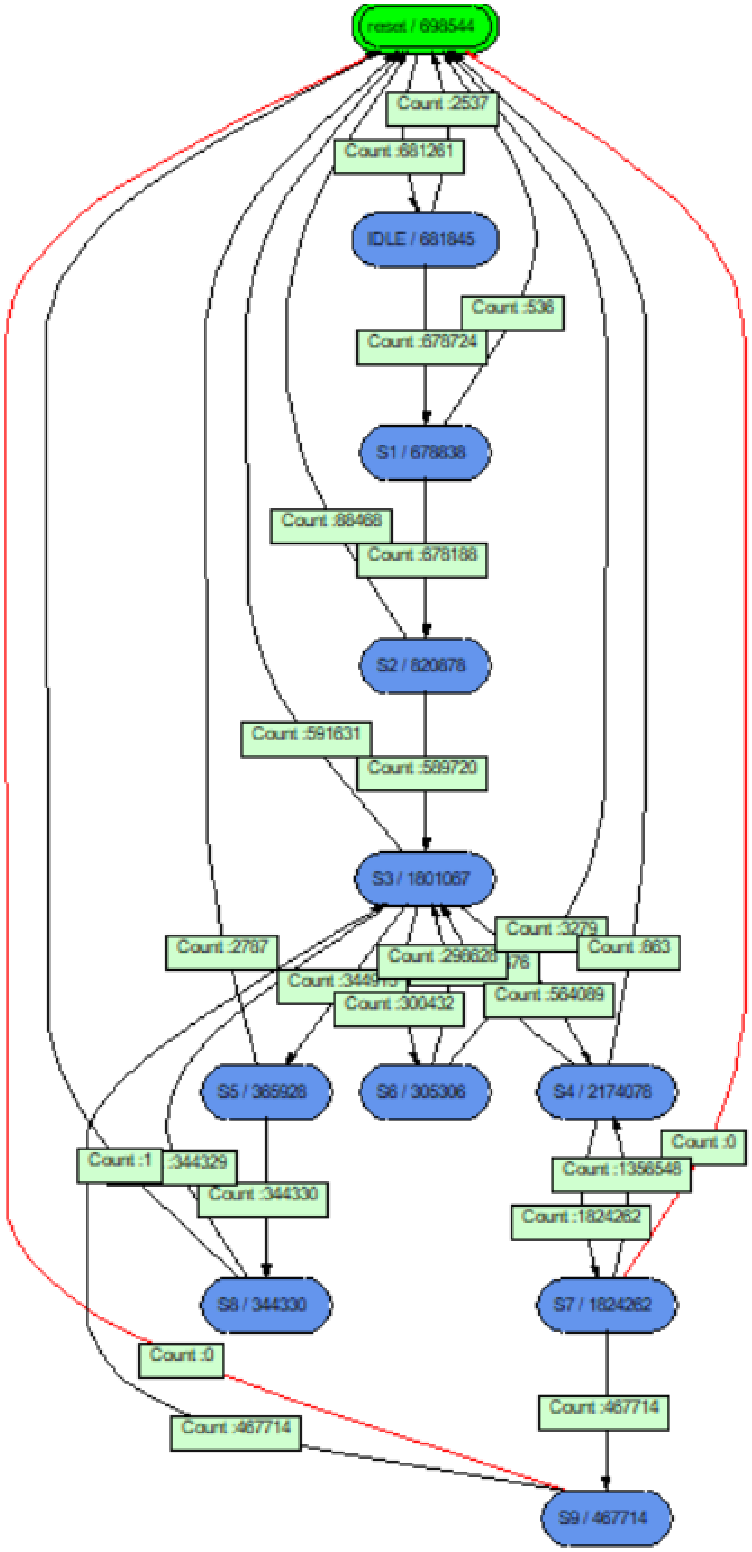
The uncovered State Transitions: note these are not recommended as they interrupt operation



```
    Uncovered Transitions :
    -----------------------
Line             Trans_ID          Transition
----             --------          ----------
  36                   20          S7 -> reset
  36                   24          S9 -> reset


    Summary                           Active      Hits    Misses % Covered
    -------                           ------      ----    ------ ---------
        States                            11        11         0    100.00
        Transitions                       25        23         2     92.00
```

Team 7

FSM Coverage:



Note: Attached the whole coverage report.

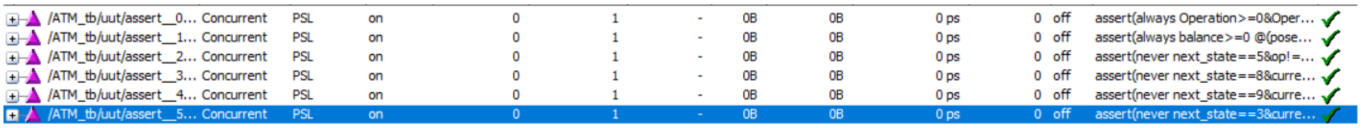
4) Assertion-Based Verification

Assertion is a statement about a specific intended behavior of the design that must hold true under normal operating conditions. We used PSL assertions to ensure that nothing went wrong during the randomized testing.

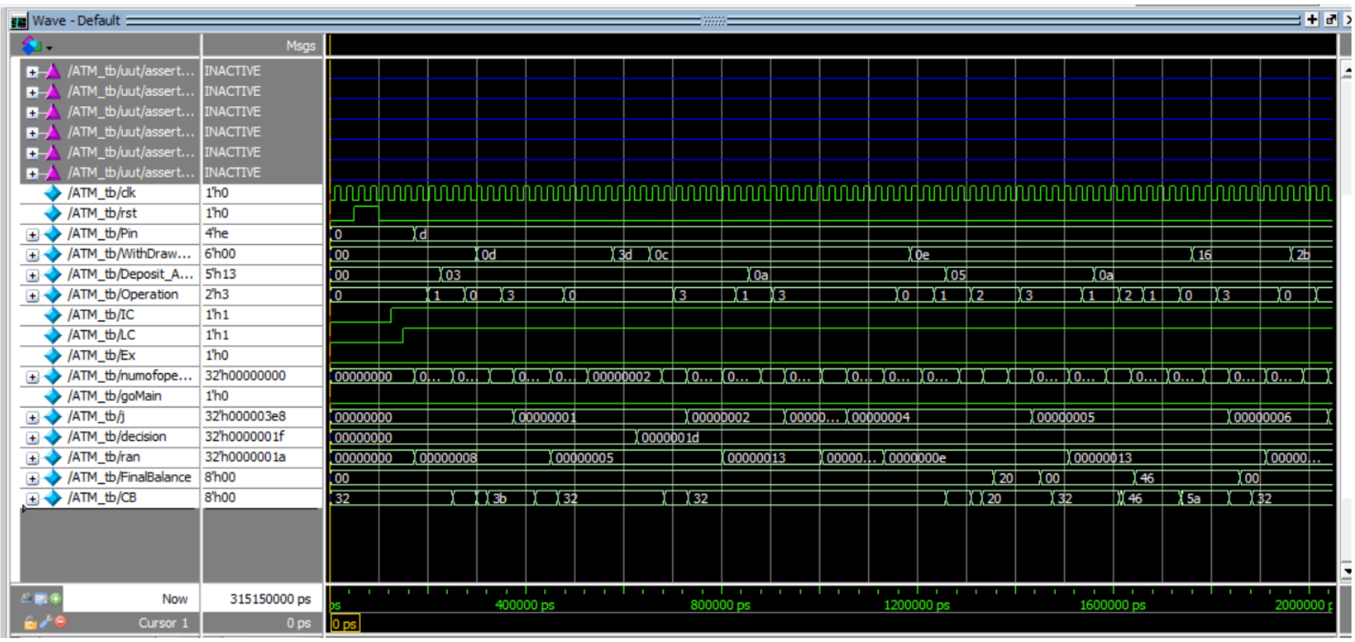
Assertion code:

```
//psl assert always(Operation >= 0 && Operation < 4) @(posedge clk);
//psl assert always(balance >=0) @(posedge clk);
//psl assert never(next_state == S5 && op != 2'b01) @(posedge clk);
//psl assert never(next_state == S8 && current_state == S5 && EA == 0) @(posedge clk);
//psl assert never(next_state == S9 && current_state == S7 && BC == 0) @(posedge clk);
//psl assert never(next_state == S3 && current_state == S2 && VP == 0) @(posedge clk);
```

Assertion results:



/ATM_tb/uut/assert__0...	Concurrent	PSL	on	0	1	-	0B	0B	0 ps	0	off	assert(always Operation>=0&Oper...	✓
/ATM_tb/uut/assert__1...	Concurrent	PSL	on	0	1	-	0B	0B	0 ps	0	off	assert(always balance>=0 @(pose...	✓
/ATM_tb/uut/assert__2...	Concurrent	PSL	on	0	1	-	0B	0B	0 ps	0	off	assert(never next_state==5&op!=...	✓
/ATM_tb/uut/assert__3...	Concurrent	PSL	on	0	1	-	0B	0B	0 ps	0	off	assert(never next_state==8&curre...	✓
/ATM_tb/uut/assert__4...	Concurrent	PSL	on	0	1	-	0B	0B	0 ps	0	off	assert(never next_state==9&curre...	✓
/ATM_tb/uut/assert__5...	Concurrent	PSL	on	0	1	-	0B	0B	0 ps	0	off	assert(never next_state==3&curre...	✓



	Msgs
/ATM_tb/uut/assert...	INACTIVE
/ATM_tb/uut/assert...	INACTIVE
/ATM_tb/uut/assert...	INACTIVE
/ATM_tb/uut/assert...	INACTIVE
/ATM_tb/uut/assert...	INACTIVE
/ATM_tb/uut/assert...	INACTIVE
/ATM_tb/clk	1'h0
/ATM_tb/rst	1'h0
/ATM_tb/Pin	4'he
/ATM_tb/WithDraw...	6'h00
/ATM_tb/Deposit_A...	5'h13
/ATM_tb/Operation	2'h3
/ATM_tb/IC	1'h1
/ATM_tb/LC	1'h1
/ATM_tb/Ex	1'h0
/ATM_tb/numofope...	32'h00000000
/ATM_tb/goMain	1'h0
/ATM_tb/j	32'h000003e8
/ATM_tb/decision	32'h0000001f
/ATM_tb/ran	32'h0000001a
/ATM_tb/FinalBalance	8'h00
/ATM_tb/CB	8'h00
Now	315150000 ps
Cursor 1	0 ps