

NCTU-EE IC LAB - FALL 2019

Lab10 Exercise

Verification: CTS (From Lab09) Coverage and Assertion

Data Preparation

1. Extract test data from TA's directory:

```
% tar xvf ~iclabta01/Lab10.tar
```

2. The extracted LAB directory contains:

- a. EXERCISE/

Description

you need to write the verification pattern for the CTS (from Lab09). You need to complete following things:

1. PATTERN.sv (Lab10/ EXERCISE /00_TESTBED/PATTERN.sv)

Generate pattern data.

Send pattern data to payment.sv and make sure that it will achieve coverage goals.

You also need to **do the reset** first.

2. CHECKER.sv (Lab10/ EXERCISE/00_TESTBED/CHECKER.sv)

Write your covergroups and assertions here.

Specifications

Coverage: (TA's DESIGN + TA's CHECKER + Your PATTERN)

1. Create a covergroup including coverpoint **inf.D.d_pw[0]**.

There are four bins which range 0001~0011, 0100~0111, 1000~1011, 1100~1111 in binary, respectively.

Each bin should be hit at least 10 time.

(sample the value at posedge clk when passwd_valid is high)

2. Create a covergroup including coverpoint **inf.D.d_id[0]** with auto bin max = 256.

(means that you need to divide the ID signal into 256 bins averagely)

And each bin has to be hit at least 1 time.

(sample the value at posedge clk when id_valid is high)

3. Create a covergroup including coverpoint **inf.D.d_act[0]**

There are five actions for **inf.D.d_act[0]**: trnsf, deposit, check_blc, change_pw, exit.

Create the transition bins from one action to itself or others (**except from exit to exit**)

such as: trnsf to trnsf, trnsf to deposit, trnsf to check_blc and so on.

There are total 24 transition bins. Each transition bin should be hit at least 10 times.

(sample the value at posedge clk when act_valid is high)

4. Create a covergroup including coverpoint **inf.err_msg**

Every case of **inf.err_msg** **except No_Err** should occur at least 10 times.

(sample the value at negedge clk when inf.out_valid is high)

5. Create a covergroup including coverpoints **inf.complete** and **inf.out_balance**

The bins of **inf.complete** need be 0 and 1 hit at least 100 times.

The bins of **inf.out_balance** should range 1~17695, 17696~29492, 29493~36045, 36046~47842, 47843~65536, respectively. And each needs to be hit at least 20 times.

Create a cross bin for the above **inf.complete** and **inf.out_balance**.

Each combination should occur at least 20 times.

(sample the value at negedge clk when inf.out_valid is high)

Ex: (0 or 1) x (1~17695, 17696~29492, 29493~36045, 36046~47842, 47843~65536)

Note: When you send the pattern to the payment.sv, you need to follow the input specs from the Lab09.

For example, all outputs should be zero after reset.

You should write some assertion in your CHECKER.sv to check.

If you violate the following assertion, this part you would be **fail**.

And if you violate the specs and assertions didn't discover but TA discover during demo, you will also **fail**.

Assertion: (TA's DESIGN + TA's PATTERN+ **Your CHECKER**)

***You should hand in the code of this part.**

1. All outputs should be zero after reset.
2. If action is completed, inf.err_msg must be 4'b0.
3. When aciotn is trnsf, the gap length between id_valid and amnt_valid is at least 1 cycle.

Notice that your assertion warning messages should be "Assertion X is violated", where X is number.

Note

1. Grading Policy

- Coverage 70%
 - Spec 1- 10%
 - Spec 2- 10%
 - Spec 3- 10%
 - Spec 4- 10%,
 - Spec 5- 10%
- Simulation time 20%
- Assertion 30%
 - Assertion 1- 10%
 - Assertion 2- 10%
 - Assertion 3- 10%

You need to pass all specs and will get the simulation time score.

The second demo will be 30% off. The highest second demo score will be 70.

2. Please upload the following file on newE3 platform before 12:00 at noon on 12/9:

- PATTERN_iclabxx.sv, CHECKER_iclabxx.sv

If you use .txt file to read your pattern, please also upload that file.

Or you will be failed at demo.

And if you have modified the Usertype_PKG.sv, please also upload that file too.

(Usertype_PKG_iclabXX.sv)

Using Cadence IMC (Incisive Metrics Center)

1. % irun TESTBED.sv -define RTL -debug -coverage U -covoverwrite ./02_run_cov)

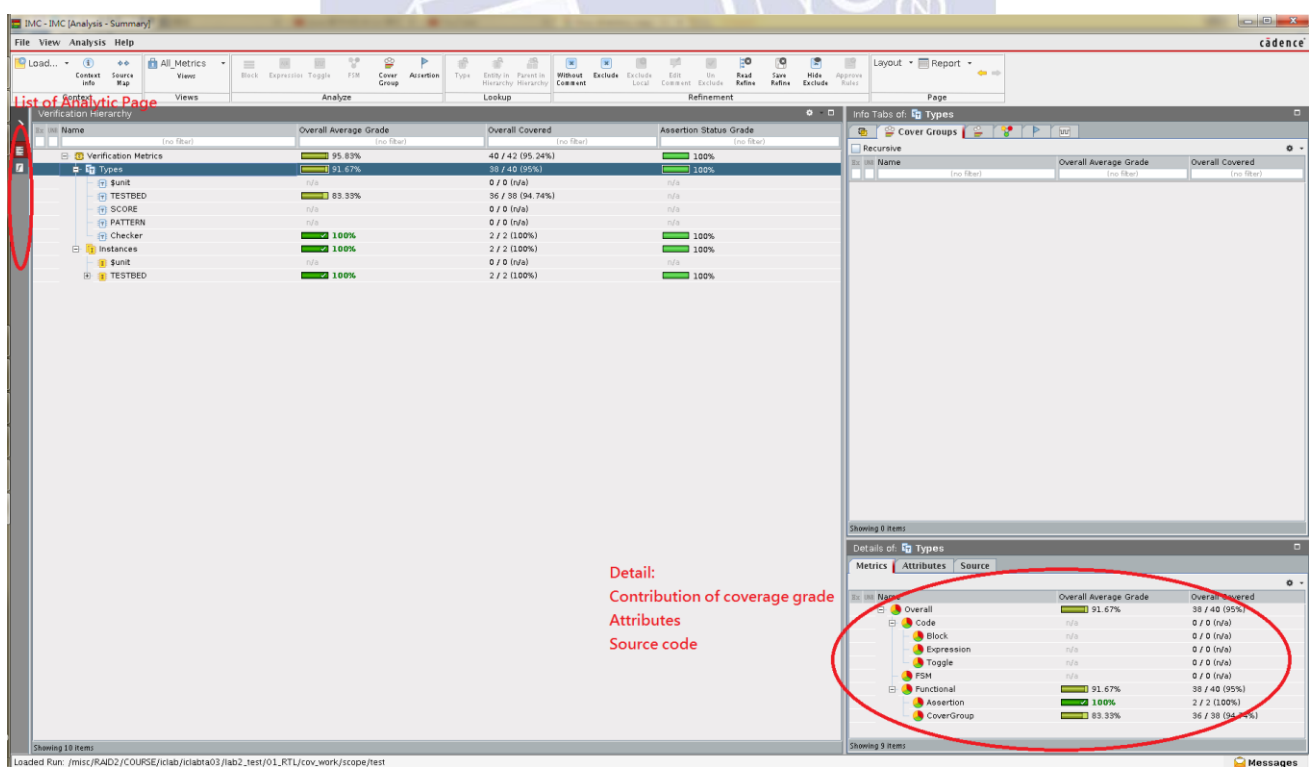
```
4 18:03 TESTBED.SV -> ./02_run_cov/TESTBED.SV
8 18:30 cov_work
8 18:27 imc.log
8 18:30 irun.log
```

2. % imc &

3. Load the coverage database (default path: /01_RTL/cov_work/scope/test)



Panel



4. Covergroup analysis

The screenshot shows the Cadence BMC Analysis - Summary window. The Verification Hierarchy on the left lists various analysis types. The main table displays the Overall Average Grade and Overall Covered status for different analysis types. A red circle highlights the 'Cover Group Analysis' option in the context menu, with a red arrow pointing to it and the text: "Right click on your module, then choose 'Cover Group Analysis' for more advanced information".

Name	Overall Average Grade	Overall Covered	Assertion Status Grade
Verification Metrics	91.67%	40 / 43 (93.24%)	100%
Types	91.67%	36 / 40 (90%)	100%
TESTBED	83.33%	0 / 0 (n/a)	n/a
Block Analysis	83.33%	0 / 0 (n/a)	n/a
Expression Analysis	100%	2 / 2 (100%)	100%
Toggle Analysis	100%	2 / 2 (100%)	100%
FSM Analysis	100%	0 / 0 (n/a)	n/a
Cover Group Analysis	100%	2 / 2 (100%)	100%
Assertion Analysis	100%	2 / 2 (100%)	100%

Info Tabs of: TESTBED

Name	Overall Average Grade	Overall Covered
cov	83.33%	36 / 38 (94.74%)

Showing 1 item:

Details of: TESTBED

Metrics: Attributes Source

Name	Overall Average Grade	Overall Covered
Overall	83.33%	36 / 38 (94.74%)
Code	n/a	0 / 0 (n/a)
Expression	n/a	0 / 0 (n/a)
Toggle	n/a	0 / 0 (n/a)
FSM	n/a	0 / 0 (n/a)
Functional	83.33%	36 / 38 (94.74%)
Assertion	n/a	0 / 0 (n/a)
CoverGroup	83.33%	36 / 38 (94.74%)

Showing 3 items:

5. Detail of items inside covergroup

The screenshot shows the Cadence BMC Analysis - Functional window. The Cover Groups analysis is displayed, showing the Overall Average Grade and Overall Covered status for different cover groups. A red circle highlights the 'cov' cover group, with a red arrow pointing to it and the text: "1. Choose the covergroup you want to see". Another red circle highlights the 'latency_check' item, with a red arrow pointing to it and the text: "2. Click the item". A red circle highlights the 'BINS OF: latency_check' table, with a red arrow pointing to it and the text: "3. Detail information of your bins are shown here".

Overall Local Grade: 83.33% (no filter)

CoverGroup Local Grade: 83.33% (no filter)

Assertion Local Grade: n/a

Name	Overall Average Grade	Overall Covered
cov	83.33%	36 / 38 (94.74%)

Showing 1 item:

ITEMS OF: cov

Name	Overall Average Grade	Overall Covered
latency_check	100%	32 / 32 (100%)
prop_check	66.67%	4 / 6 (66.67%)

Showing 2 items:

BINS OF: latency_check

Name	Overall Average Grade	Overall Covered	Score
auto(4.7)	100%	1 / 1 (100%)	4
auto(6.11)	100%	1 / 1 (100%)	253
auto(12.15)	100%	1 / 1 (100%)	254
auto(16.19)	100%	1 / 1 (100%)	3
auto(20.23)	100%	1 / 1 (100%)	254
auto(24.27)	100%	1 / 1 (100%)	504
auto(28.31)	100%	1 / 1 (100%)	9
auto(32.35)	100%	1 / 1 (100%)	3
auto(36.39)	100%	1 / 1 (100%)	4
auto(40.43)	100%	1 / 1 (100%)	251
auto(44.47)	100%	1 / 1 (100%)	254
auto(48.51)	100%	1 / 1 (100%)	4
auto(52.55)	100%	1 / 1 (100%)	253
auto(56.59)	100%	1 / 1 (100%)	3
auto(60.63)	100%	1 / 1 (100%)	2
auto(64.67)	100%	1 / 1 (100%)	257
auto(68.71)	100%	1 / 1 (100%)	252
auto(72.75)	100%	1 / 1 (100%)	4
auto(76.79)	100%	1 / 1 (100%)	252
auto(80.83)	100%	1 / 1 (100%)	509
auto(84.87)	100%	1 / 1 (100%)	2
auto(88.91)	100%	1 / 1 (100%)	3
auto(92.95)	100%	1 / 1 (100%)	255
auto(96.99)	100%	1 / 1 (100%)	253
auto(100.03)	100%	1 / 1 (100%)	255
auto(104.07)	100%	1 / 1 (100%)	2

Showing 32 items:

Details of: latency_check

Attributes Source

A. Value	
1	

Cover Group: cov

CoverGroup Average Grade: 100%

CoverGroup Covered: 32.0

CoverGroup Ignored: 0.0

CoverGroup Total: 32.0

Showing 2 items:

Loaded Run: /misc/RAD2/COURSE/iclab/iclabna03/lab2_test/01_RTL/cov/work/scope/test

6. Check your source code

The screenshot shows the Cadence JMC [Analysis - Functional] interface. The main window displays coverage analysis results for the 'prop_check' testbench. The 'COVER GROUPS' pane on the left shows the 'cov' group with an overall average grade of 83.33% and overall coverage of 36 / 38 (94.74%). The 'ITEMS OF: cov' pane shows 'latency_check' at 100% and 'prop_check' at 66.67%. The 'BINS OF: prop_check' pane shows a table of bins with their respective grades and scores. The 'Details of: prop_check' pane shows the source code for the 'prop_check' testbench, with the 'Source' button highlighted by a red circle and the text 'Check your source code here'.

Name	Overall Average Grade	Overall Covered
cov	83.33%	36 / 38 (94.74%)

Name	Overall Average Grade	Overall Covered
latency_check	100%	32 / 32 (100%)
prop_check	66.67%	4 / 6 (66.67%)

Name	Overall Average Grade	Overall Covered	Score
a0	0%	0 / 1 (0%)	0
a1	100%	1 / 1 (100%)	5780
a2	100%	1 / 1 (100%)	55
a3	100%	1 / 1 (100%)	30
a4	100%	1 / 1 (100%)	1255
a5	0%	0 / 1 (0%)	0


```

30 latency_check: coverpoint pf_latency(
31   option auto_bin_max = 32;
40 )
41
42 prop_check: coverpoint pf_prop(
43   bins s0 = {0..10};
44   bins s1 = {11..20};
45   bins s2 = {21..30};
46   bins s3 = {31..40};
47 )
  
```

