

NCTU-EE IC LAB - Spring 2023

Lab10 Exercise

Verification: OS (From Lab09) Coverage

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 OS(from Lab9). You need to complete following things:

1. **PATTERN.sv** (Lab10/ Exercise /00_TESTBED/PATTERN.sv)

Generate pattern data.

Send pattern data to OS.sv and bridge.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 cover groups and assertions here.

Specifications

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

1. Create a covergroup including coverpoint inf.D.d_money (means 0~15 bits of input signal D when typing your money amount). There are five bins which range 16'd0~16'd12000, 16'd12001~16'd24000, 16'd24001~16'd36000, 16'd36001~16'd48000, 16'd48001~16'd60000, respectively. Each bin should be hit at least 10 time. (sample the value at posedge clk when amnt_valid is high)
2. Create a covergroup including coverpoint inf.D.d_id[0] (means 0~7 bits of input signal D when typing your ID) with auto_bin_max = 256. (means that you need to divide the inf.D.d_id[0] signal into 256 bins averagely). And each bin has to be hit at least 2 time. (sample the value at posedge clk when id_valid is high)
3. Create a covergroup including coverpoint inf.D.d_act[0] (means 0~3 bits of input signal D when typing your action). There are four actions for inf.D.d_act[0]: buy, check, deposit, return. Create the transition bins from one action to itself or others. such as: buy to check, buy to deposit, buy to return and so on. There are total 16 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.d_item[0](Large 、 Medium 、 Small). Each of these three bins needs at least 20 times. (sample the value at posedge clk when item_valid is high)
5. Create a covergroup including coverpoint inf.err_msg. Every case of inf.err_msg except No_Err should occur at least 20 times. (sample the value at negedge clk when inf.out_valid is high)
6. Create a covergroup including coverpoints inf.complete. The bins of inf.complete need to be 0 and 1, and each bin should be hit at least 200 times. (sample the value at negedge clk when inf.out_valid is high)

Notice:

1. When you send the **pattern** and **dram** to the OS.sv, you **need to follow the specs** from the Lab09. For example, all input valid signals won't overlap with each other. You can write some assertions in your CHECKER.sv to check. If you violate the specs and your assertions didn't discover but TA discover during demo, you will fail.
2. **After passing the last pattern, your PATTERN should finish immediately. (Still remember not to violate any specs.)**
3. During the demo, TA will also use wrong design to test if your pattern can check the correctness of the output signals of the design. When the answer is wrong, you should stop the program immediately and display "**Wrong Answer**" on the terminal.
4. **Do not print anything else except from "Wrong Answer" when the answer is wrong , or you will fail this lab.**

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

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

1. All outputs signals (including OS.sv and bridge.sv) should be zero after reset.
2. If action is completed, err_msg must be 4'b0.
3. If action is not completed, out_info should be 32'b0.
4. All input valid can only be high for **exactly** one cycle.
5. The gap between each input valid is at least 1 cycle and at most 5 cycles(including the correct input sequence).
6. The five valid signals won't overlap with each other.(id_valid, act_valid, amnt_valid, item_valid , num_valid)
7. Out_valid will be high for one cycle.
8. Next operation will be valid 2-10 cycles after out_valid fall.

9. Latency should be less than 10000 cycle for each operation.

Notice that once the spec is violated, you should stop the program immediately and show the assertion message on the terminal. Your assertion warning messages should be “**Assertion X is violated**”, where X is the number of assertions. You **must** directly copy the messages provided by TA in student.txt, **or you will fail this lab**. The definition of cycle and latency is the same as Lab09.

Note

1. Grading Policy

- **Pattern + Coverage + Assertion: 70%**
- **Simulation time of coverage: 30%**

```
-----
                        Congratulations!
                    You have passed all patterns!
-----
Simulation complete via $finish(1) at time 3700 NS + 0
./PATTERN.v:242    $finish;
ncsim> exit
```

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

The second demo will be 30% off.

2. Please upload the following file in Lab10/EXERCISE/09_SUBMIT

- PATTERN.sv
- CHECKER.sv
- dram.dat (Your dram.dat **must** follow the spec in Lab09.)
- Usertype_OS.sv
- No need to submit clk period, we will use 1ns as period when demoing.
- **./01_submit**

3. Deadlines:

- 1st_deadline : 2022/05/08(Mon.) 12:00:00
- 2nd_deadline: 2022/05/10(Wed.) 12:00:00

4. Since the purpose of this Lab is to use SystemVerilog to do verification. You should **generate a pattern in the PATTERN.sv directly** instead of using the read file method.
5. Don't use any wire/reg/submodule/parameter name called ***error***, ***congratulation***, ***latch*** or ***fail*** otherwise you will fail the lab. Note: * means any char in front of or behind the word. e.g: error_note is forbidden.
6. Please do not display any other information on the CHECKER.sv and PATTERN.sv except for the information written in the student.txt.

Using Cadence IMC (Incisive Metrics Center)

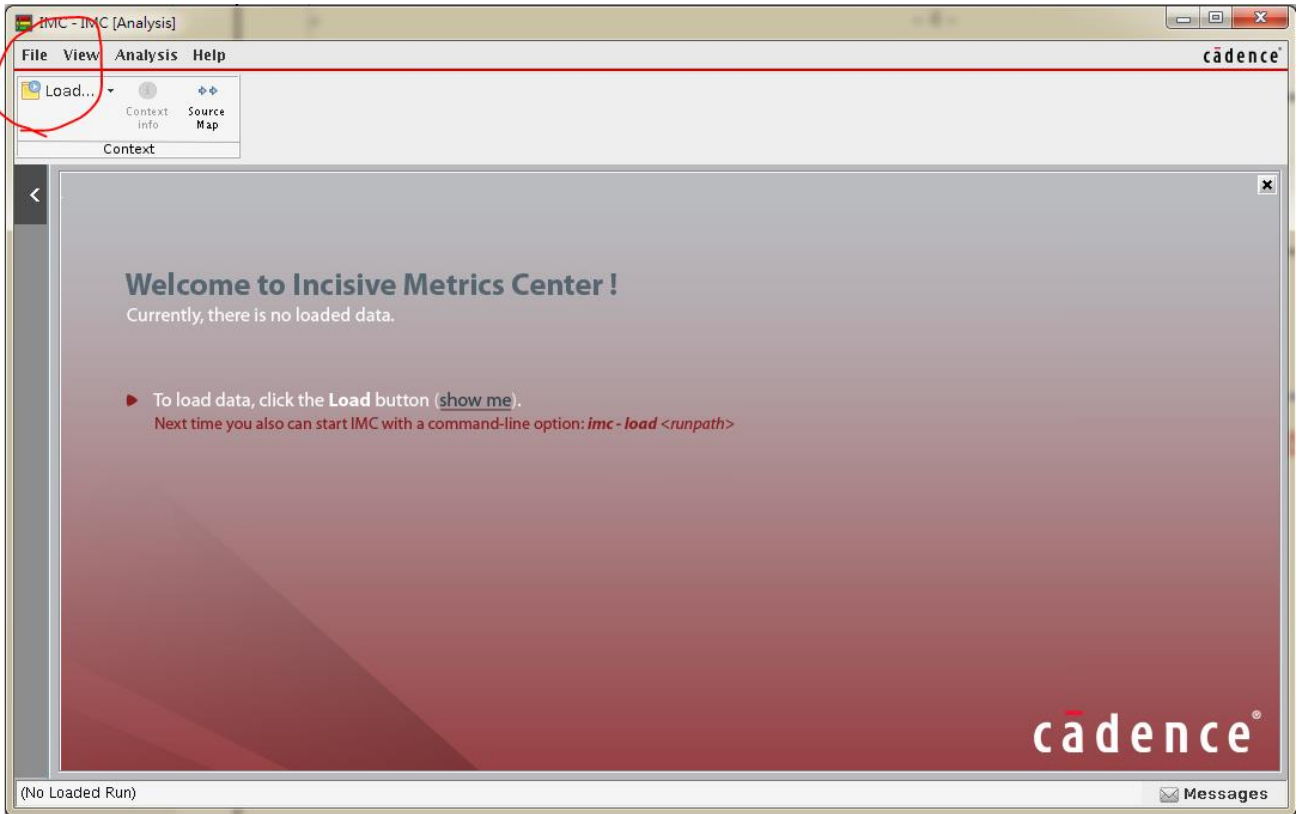
1. % irun TESTBED.sv -define RTL -debug -coverage U -covoverwrite (./02_run_cov)

```

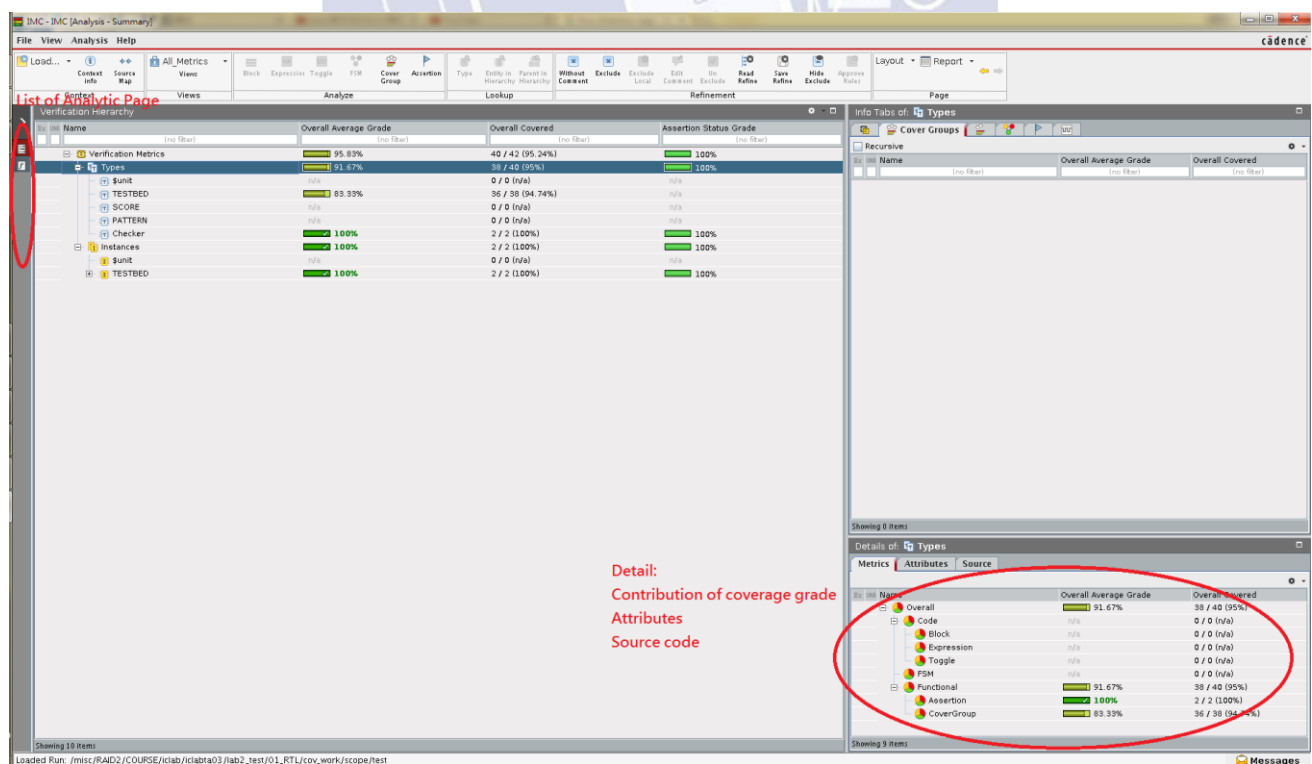
4 19:03 TESTBED.SV -> /1700_TESTBED/TESTBED.S
8 18:30 cov_work
8 18:27 imc.log
8 18:30 i-run.log

```

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



Panel



4. Covergroup analysis

The screenshot shows the Cadence IMC - IMC [Analysis - Summary] window. The 'Verification Hierarchy' tab is active, displaying a tree structure of verification metrics. The 'Checker' tab is also visible, showing a table of metrics with columns for Name, Overall Average Grade, and Overall Covered. The 'Checker' tab shows a table with the following data:

Name	Overall Average Grade	Overall Covered
check_int	100%	100 / 100 (100%)

The 'Checker' tab also shows a table of metrics with columns for Name, Overall Average Grade, and Overall Covered. The 'Checker' tab shows a table with the following data:

Name	Overall Average Grade	Overall Covered
Overall	100%	100 / 100 (100%)
Code	n/a	0 / 0 (n/a)
Block	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	100%	100 / 100 (100%)
Assertion	100%	11 / 11 (100%)
CoverGroup	100%	97 / 97 (100%)

5. Detail of items inside covergroup

The screenshot shows the Cadence IMC - IMC [Analysis - Functional] window. The 'CoverGroup Local Grade' is 83.33%. The 'BINS OF: latency_check' table shows the following data:

Name	Overall Average Grade	Overall Covered	Score
auto(4-5)	100%	1 / 1 (100%)	4
auto(4-7)	100%	1 / 1 (100%)	253
auto(6-11)	100%	1 / 1 (100%)	254
auto(12-15)	100%	1 / 1 (100%)	256
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%)	9
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%)	9
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(100-103)	100%	1 / 1 (100%)	255
auto(104-107)	100%	1 / 1 (100%)	2

Annotations in the image include:

- 1. Choose the covergroup you want to see (pointing to the 'cov' covergroup in the left pane)
- 2. Click the item (pointing to the 'latency_check' item in the left pane)
- 3. Detail information of your bins are shown here (pointing to the 'BINS OF: latency_check' table)

Every coverage spec should be 100%

Loaded Run: /misc/RAD2/COURSE/iclab/iclab03/lab2_test/01_RTL/cov_work/scope/test

6. Check your source code

The screenshot displays the Cadence BMC - BMC [Analysis - Functional] interface. The main window shows the 'COVER GROUPS' section with a table of coverage data. The 'prop_check' group is highlighted, showing an overall average grade of 83.33% and overall coverage of 36 / 38 (94.74%). Below this, the 'ITEMS OF: cov' section shows a table with two items: 'latency_check' (100% grade, 32 / 32 coverage) and 'prop_check' (66.67% grade, 4 / 6 coverage). A red circle highlights the 'Source' button in the 'Details of: prop_check' section, with a red arrow pointing to it and the text 'Check your source code here'.

Name	Overall Average Grade	Overall Covered
cov	83.33%	36 / 38 (94.74%)
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%)	9780
a2	100%	1 / 1 (100%)	55
a3	100%	1 / 1 (100%)	30
a4	100%	1 / 1 (100%)	1255
a5	0%	0 / 1 (0%)	0

Check your source code here

Source

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

