

# NYCU-EE ICLAB – Autumn 2024

## Lab10 Exercise: Coverage & Assertion

### Verification: Storck Trading Program (From Lab09)

#### Data Preparation

1. Extract test data from TA's directory:  
**% tar xvf ~iclabTA01/Lab10.tar**
2. The extracted LAB directory contains:  
Exercise/
3. DO NOT paste the pseudo\_DRAM.sv from Lab09 to the Lab10.

#### Description

You need to write the verification pattern for the Program from Lab09. You need to complete following things:

1. **PATTERN.sv** (Lab10/Exercise/00\_TESTBED/PATTERN.sv)  
Generate pattern data.  
Send pattern data to Program.sv and make sure that it will achieve coverage goals.  
You also need to **check** the correctness of the output signals of the design.
2. **CHECKER.sv** (Lab10/Exercise/00\_TESTBED/CHECKER.sv)  
Write your cover groups and assertions here.

#### Specifications

##### Coverage:

1. Each case of **Formula\_Type** should be select at least 150 times.
2. Each case of **Mode** should be select at least 150 times.
3. Create a **cross bin** for the SPEC1 and SPEC2. Each combination should be selected at least 150 times. (Formula\_A,B,C,D,E,F,G,H) x (Insensitive, Normal, Sensitive)
4. Output signal inf.warn\_msg should be “No\_Warn”, “Date\_Warn”, “Data\_Warn“,,”Risk\_Warn, each at least 50 times. (Sample the value when inf.out\_valid is high)
5. **Create the transitions bin** for the inf.D.act[0] signal from [Index\_Check:Check\_Valid\_Date] to [Index\_Check:Check\_Valid\_Date]. Each transition should be hit at least 300 times. (sample the value at posedge clk iff inf.sel\_action\_valid)
6. Create a covergroup for **variation of Update action** with auto\_bin\_max = 32, and each bin have to hit at least one time.

Notice:

↳ maximum number of  
bins to create

1. When you send the pattern to the Program.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 demo, TA will also use wrong design (4 cases) 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.
  - "./00\_run\_cov" for normal design
  - "./00\_run\_cov FAIL\_X" for error answer design, e.g. "./00\_run\_cov FAIL\_1"
  - 4 Error Answer Cases, from FAIL\_1 to FAIL\_4
4. Under 01\_RTL, you can use bellowing commands to check your coverage result.
  - A. "00\_run\_cov" for running your design and generate the coverage related files.
  - B. "02\_cov\_detail" for generating the detail report (01\_RTL/Report/Coverage\_Detail.log) of your coverage result.
  - C. "03\_cov\_summary" for generating the brief report (01\_RTL/Report/Coverage\_Summary.log) of your coverage result. You should ensure both of CoverGroup Average/Covered to 100% coverage.

name	CoverGroup Average	CoverGroup Covered
Checker	100.00%	100.00% (80/80)

- D. "imc &" for starting Cadence IMC (Incisive Metrics Center) in gui mode. (Please refer to appendix)
- E. "imc -batch" for starting Cadence IMC (Incisive Metrics Center) in batch (shell) mode.

#### Assertion:

1. All outputs signals (Program.sv) should be zero after reset.
2. Latency should be less than 1000 cycles for each operation.
3. If action is completed (complete=1), warn\_msg should be 2'b0 (No\_Warn).
4. Next input valid will be valid 1-4 cycles after previous input valid fall.
5. All input valid signals won't overlap with each other.
6. Out\_valid can only be high for **exactly** one cycle.
7. Next operation will be valid 1-4 cycles after out\_valid fall.
8. The input date from pattern should adhere to the real calendar. (ex: 2/29, 3/0, 4/31, 13/1 are illegal cases)

9. The AR\_VALID signal should not overlap with the AW\_VALID signal.

Notice:

1. You can't use PATTERN.sv to check the spec above.
2. 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 can directly copy the messages provided by TA in 00\_TESTBED/message.txt . For normal design, please use "01\_run"; For testing error design, use "01\_run SPEC\_X\_Y", please refer to the table in grading policy.
3. The definition of cycle and latency is the same as Lab09.

### Self Testing Flow

---

#### 1. ./00\_run\_cov (For Coverage)

TA's design + Your Checker + Your Pattern (including your dram.dat)

#### 2. ./01\_run (For Assertion)

TA's design + Your Checker + TA's Pattern (including your dram.dat)

### TA's Demo Flow

---

#### 1. (TA's DESIGN + TA's CHECKER + **Your PATTERN**):

Test your pattern if it can pass TA's design and checker, and detect the error answer in TA's design (FAIL1~FAIL\_4)

#### 2. (TA's DESIGN + TA's PATTERN+ **Your CHECKER**):

Test your checker if it can pass and find every hidden error in TA's design and pattern. (SPEC\_1\_1 ~ SPEC\_9\_2)

#### 3. (TA's DESIGN + **Your CHECKER** + **Your PATTERN (including your dram.dat)**):

Test your pattern and checker if their coverage rate can reach 100%, and measure their simulation time of coverage.

### Note

---

#### 3. Grading Policy

- Coverage: 50%
- 10%: Pass each item of the command "./00\_run\_cov" in the table below.
- 30%: 100% Coverage (requires passing the previous item).
- 10%: Simulation time of coverage (**You need to pass all specs in coverage and assertion and will get the simulation time score.**).

9 / 3

- **Assertion: 50%**
- Please refer to the table for each item's (./01\_run) point (Total 10 items).

```

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

```

	Command	Print From	You should Print	Points
./00_run_cov	(None)	PATTERN.sv	Congratulations	2%
	FAIL_1~FAIL_4	PATTERN.sv	Wrong Answer	8%
<div style="display: flex; align-items: center;"> <div style="border: 1px solid black; border-radius: 50%; width: 40px; height: 40px; display: flex; align-items: center; justify-content: center; margin-right: 10px;">50</div> <div style="border-left: 1px solid black; padding-left: 10px;"> ./01_run </div> </div>	(None)	PATTERN.sv	Congratulations	6%
	SPEC_1_1 ~ SPEC_1_4	CHECKER.sv	Assertion 1 is violated	5%
	SPEC_2_1 ~ SPEC_2_4	CHECKER.sv	Assertion 2 is violated	5%
	SPEC_3_1 ~ SPEC_3_2	CHECKER.sv	Assertion 3 is violated	5%
	SPEC_4_1 ~ SPEC_4_4	CHECKER.sv	Assertion 4 is violated	5%
	SPEC_5_1 ~ SPEC_5_4	CHECKER.sv	Assertion 5 is violated	5%
	SPEC_6_1	CHECKER.sv	Assertion 6 is violated	4%
	SPEC_7_1	CHECKER.sv	Assertion 7 is violated	4%
	SPEC_8_1 ~ SPEC_8_6	CHECKER.sv	Assertion 8 is violated	6%
	SPEC_9_1	CHECKER.sv	Assertion 9 is violated	5%

(The info that need to print can be copied from 00\_TESTBED/message.txt)

The second demo will be **30% off**.

**4. 1<sup>st</sup> demo: 11/25 (MON) 12:00; 2<sup>nd</sup> demo: 11/27 (WED) 12:00**

**5. Please submit the following files by 09\_submit (Will be automatically packaged by 00\_tar):**

- PATTERN.sv
- CHECKER.sv
- dram.dat

- Remember using **dram.dat** when you read your data in pattern. Or you will be failed at demo. You can use any software language (e.g. Python, C/Cpp) to generate.
- Usertype.sv

**6. Since the purpose of this Lab is to use SystemVerilog to do verification. You should generate pattern in the PATTERN.sv directly instead of using read file method.**

- Don't use any wire/reg/submodule/parameter name called **\*error\***, **\*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.
- If there is an unknown in your dram.dat, you will fail this lab.

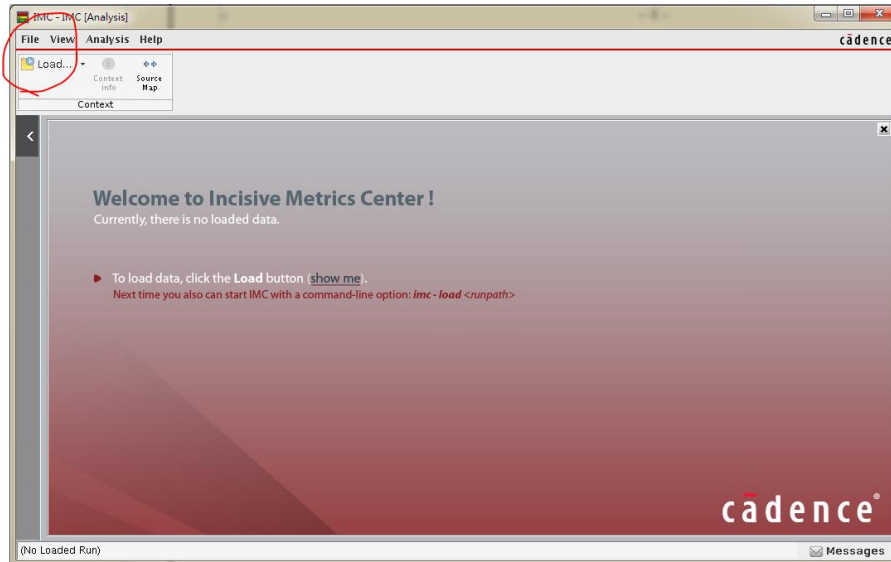
## Using GUI Mode of Cadence IMC (Incisive Metrics Center)

### 1. % ./00\_run\_cov

```
4 18:05 TESTBED.SV -> ./00_TESTBED/TESTBED.SV
8 18:30 cov_work
8 18:27 imc.log
8 18:30 icrun.log
```

### 2. % imc &

### 3. Load the coverage database (default path: /01\_RTL/cov\_work/scope/test)



## Panel

**List of Analytic Page**

Name	Overall Average Grade	Overall Covered	Assertion Status Grade
Verification Metrics	95.83%	40 / 42 (95.24%)	100%
Types	91.67%	38 / 40 (95%)	100%
Unit	n/a	0 / 0 (n/a)	n/a
TESTBED	83.33%	36 / 38 (94.74%)	n/a
SCORE	n/a	0 / 0 (n/a)	n/a
PATTERN	n/a	0 / 0 (n/a)	n/a
Checker	100%	2 / 2 (100%)	100%
Instances	100%	2 / 2 (100%)	100%
Unit	n/a	0 / 0 (n/a)	n/a
TESTBED	100%	2 / 2 (100%)	100%

**Assertion Status Grade**

Name	Overall Average Grade	Overall Covered
Overall	91.67%	38 / 40 (95%)
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	91.67%	38 / 40 (95%)
Assertion	100%	2 / 2 (100%)
CoverGroup	83.33%	36 / 38 (94.74%)

**Details of Types**

Name	Overall Average Grade	Overall Covered
Overall	91.67%	38 / 40 (95%)
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	91.67%	38 / 40 (95%)
Assertion	100%	2 / 2 (100%)
CoverGroup	83.33%	36 / 38 (94.74%)

Detail:  
Contribution of coverage grade  
Attributes  
Source code

## 4. Covergroup analysis

The screenshot shows the Cadence IMC - IMC [Analysis - Summary] window. The main panel displays the Verification Hierarchy with a tree view on the left and a table of metrics on the right. The tree view includes categories like Verification Metrics, Types, Instances, and Checker. The table shows the Overall Average Grade, Overall Covered, and Assertion Status Grade for each category. The Checker panel on the right shows the Relative Elements and the Checker details, including the Overall Average Grade and Overall Covered for the checker.

Name	Overall Average Grade	Overall Covered	Assertion Status Grade
Verification Metrics	100%	216 / 216 (100%)	100%
Types	100%	108 / 108 (100%)	100%
Instances	100%	108 / 108 (100%)	100%
Checker	100%	108 / 108 (100%)	100%

## 5. Detail of items inside covergroup

The screenshot shows the Cadence IMC - IMC [Analysis - Functional] window. The main panel displays the CoverGroup Local Grade and the BINS OF: latency\_check panel. The CoverGroup Local Grade shows the Overall Average Grade and Overall Covered for the covergroup. The BINS OF: latency\_check panel shows the details of the bins, including the Name, Overall Average Grade, Overall Covered, and Score. Red circles and arrows highlight the steps to choose the covergroup and click the item. A text box states: "Every coverage spec should be 100%".

3. Detail information of your bins are shown here

1. Choose the covergroup you want to see

2. Click the item

Every coverage spec should be 100%

Name	Overall Average Grade	Overall Covered	Score
auto(12-15)	100%	1 / 1 (100%)	4
auto(16-19)	100%	1 / 1 (100%)	253
auto(20-23)	100%	1 / 1 (100%)	254
auto(24-27)	100%	1 / 1 (100%)	256
auto(28-31)	100%	1 / 1 (100%)	3
auto(32-35)	100%	1 / 1 (100%)	254
auto(36-39)	100%	1 / 1 (100%)	504
auto(40-43)	100%	1 / 1 (100%)	9
auto(44-47)	100%	1 / 1 (100%)	3
auto(48-51)	100%	1 / 1 (100%)	251
auto(52-55)	100%	1 / 1 (100%)	254
auto(56-59)	100%	1 / 1 (100%)	4
auto(60-63)	100%	1 / 1 (100%)	253
auto(64-67)	100%	1 / 1 (100%)	3
auto(68-71)	100%	1 / 1 (100%)	257
auto(72-75)	100%	1 / 1 (100%)	252
auto(76-79)	100%	1 / 1 (100%)	4
auto(80-83)	100%	1 / 1 (100%)	252
auto(84-87)	100%	1 / 1 (100%)	509
auto(88-91)	100%	1 / 1 (100%)	2
auto(92-95)	100%	1 / 1 (100%)	3
auto(96-99)	100%	1 / 1 (100%)	255
auto(100-103)	100%	1 / 1 (100%)	253
auto(104-107)	100%	1 / 1 (100%)	25

Attributes: latency\_check

Value
1
Cover Group
CoverGroup Average Grade
CoverGroup Covered
CoverGroup Ignored
CoverGroup Total



## 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' with 100% coverage and 'prop\_check' with 66.67% coverage. 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	Score
latency_check	100%	32 / 32 (100%)	0
prop_check	66.67%	4 / 6 (66.67%)	55

Name	Overall Average Grade	Overall Covered	Score
s0	0%	0 / 1 (0%)	0
s1	100%	1 / 1 (100%)	5780
s2	100%	1 / 1 (100%)	55
s3	100%	1 / 1 (100%)	30
s4	100%	1 / 1 (100%)	1255
s5	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 )
  
```

