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:

每種 100

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:

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)

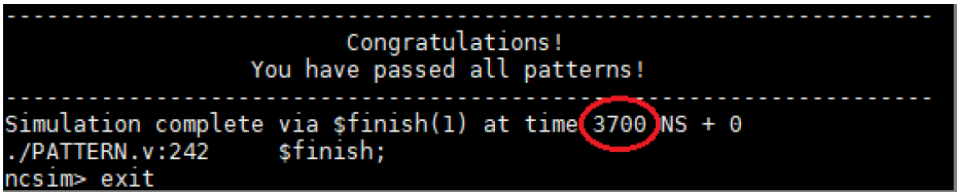
TA's Demo Flow

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_1)
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.).

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



Command		Print From	You should Print	Points
./00_run_cov	(None)	PATTERN.sv	Congratulations	2%
	FAIL_1~FAIL_4	PATTERN.sv	Wrong Answer	8%
./01_run	(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.

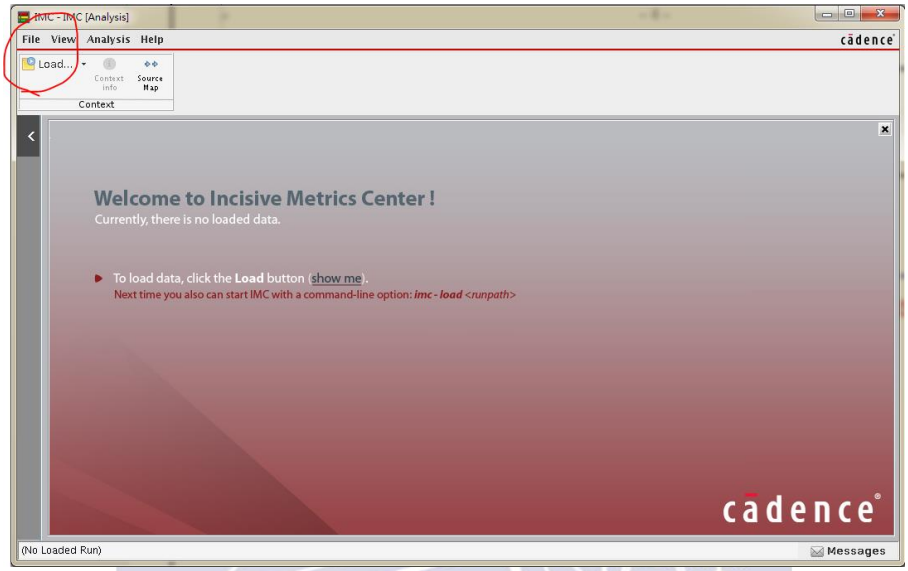
4. **1st demo: 11/25 (MON) 12:00; 2nd demo: 11/27 (WED) 12:00**
5. **Please submit the following files by 09_submit (Will be automatically packaged by 00_tar):**
   - PATTERN.sv
   - CHECKER.sv
   - dram.dat
   - Remember using dram.dat when you read your data in pattern. Or you will be failed at demo. 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.
7. 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.
8. If there is an unknown in your dram.dat, you will fail this lab.

Using GUI Mode of Cadence IMC (Incisive Metrics Center)

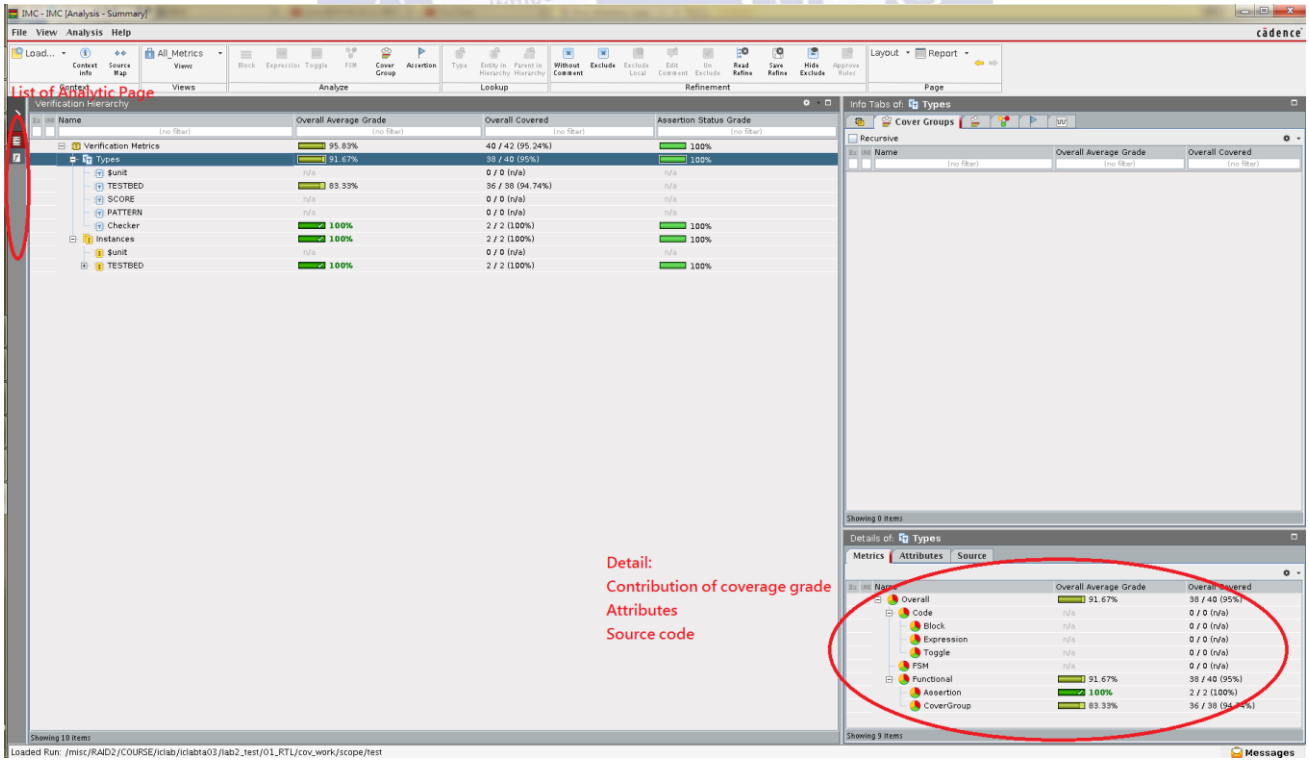
1. % ./00_run_cov

```
8 18:30 cov_work
8 18:27 imc.log
8 18:30 irun.log
```

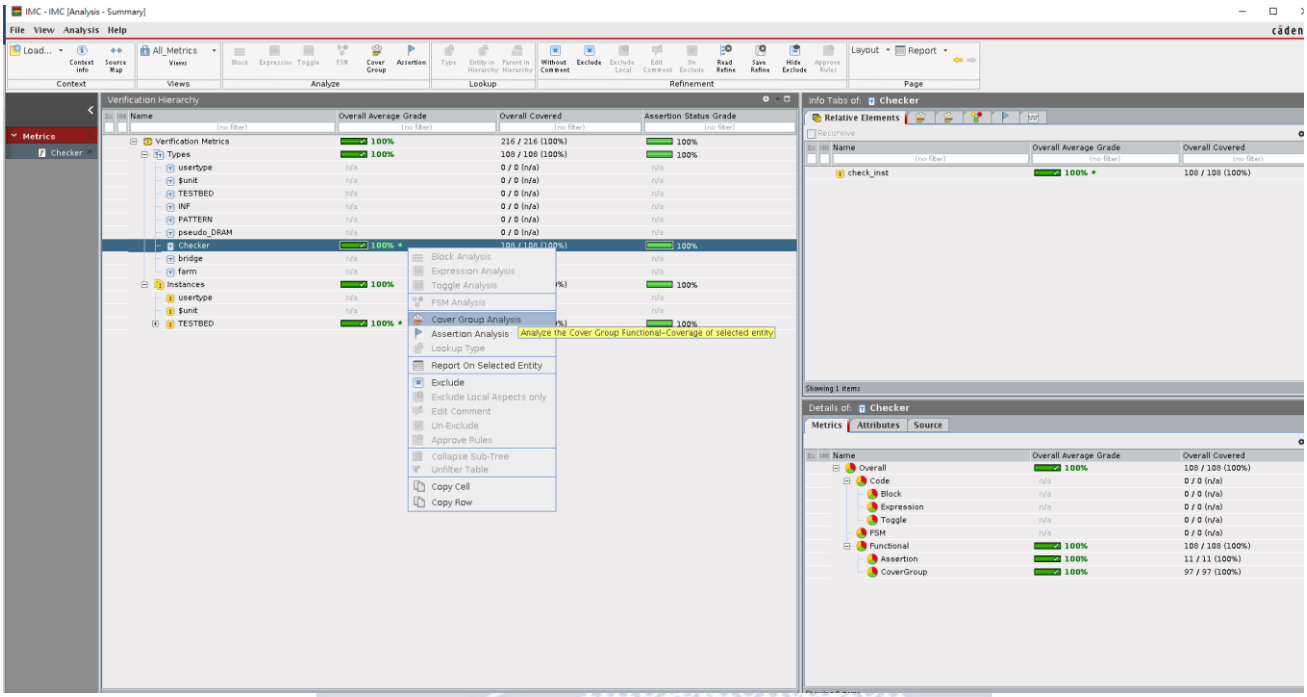
2. % imc &
3. Load the coverage database (default path: /01_RTL/cov_work/scope/test)



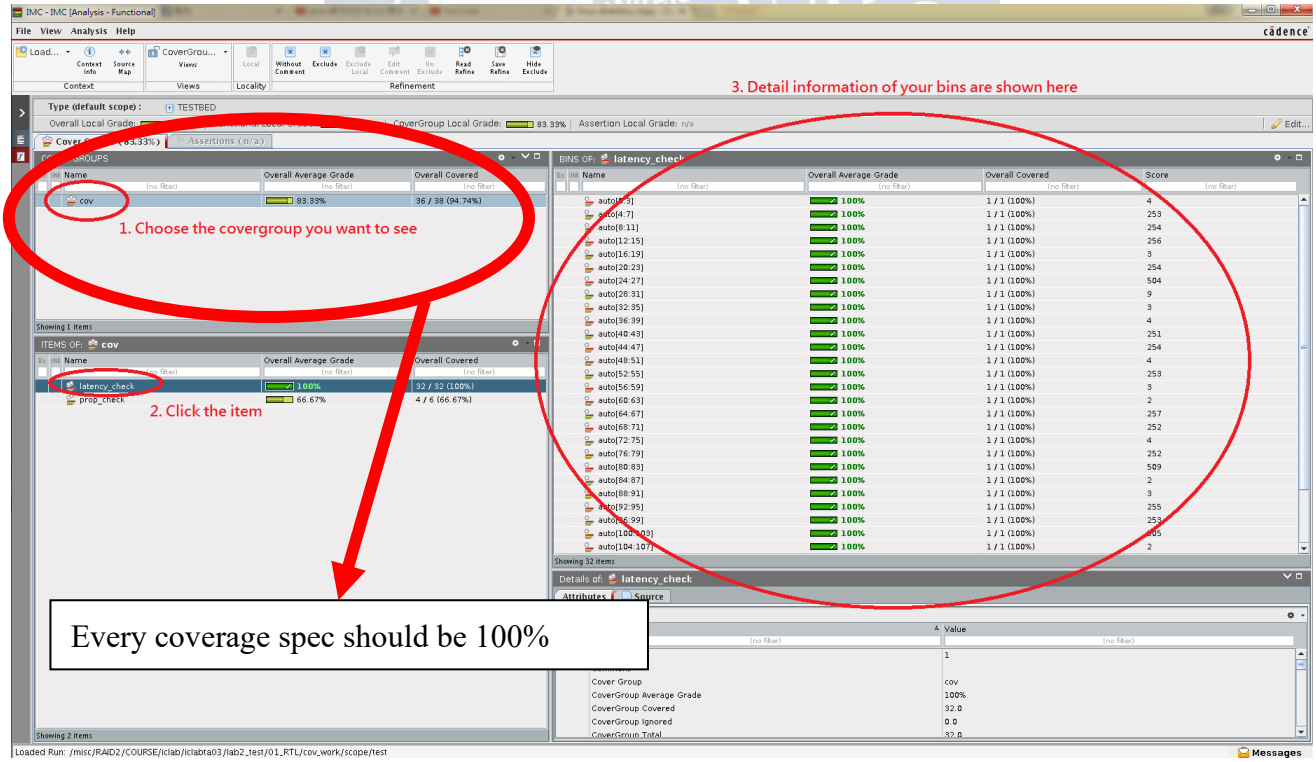
Panel



4. Covergroup analysis



5. Detail of items inside covergroup



6. Check your source code