IC Lab Formal Verification Bonus Quick Test

2022 Spring

**Name: 陳柏鈞 Student ID: 309651050 Account: iclab093 y**

1. What is Formal verification?

What's the difference between Formal and Pattern based verification?

And list the pros and cons for each.

Ans: The formal verification will try every combination of input in each cycle.

**The formal verification:**

1. breadth-first search

2. Try every combination of input in each cycle. It will do the all possible situations.

**Pros:** 1. Less testbench effort required.

2.Using systematic method.

3. Improves productivity and schedule.

4. Leads to higher quality

**Cons:** It takes more time to verify.

**Pattern verification:**

1.depth-first search

2.Test only one state in one cycle. It will repeat the same combination, because it is random test.

**Pros:** Free to use(often). Easy to use. We can test the tasks what we think immediately.

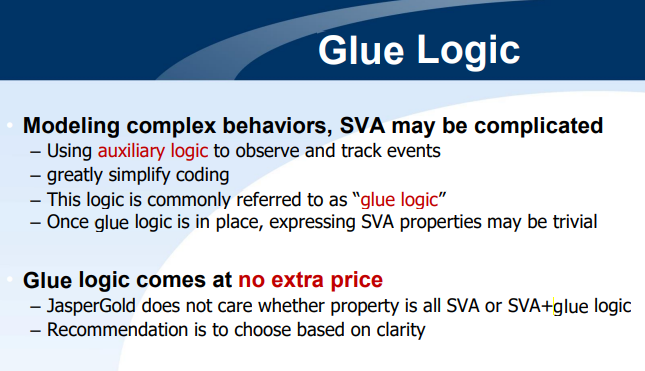
**Cons:**

1. We probably miss some case.

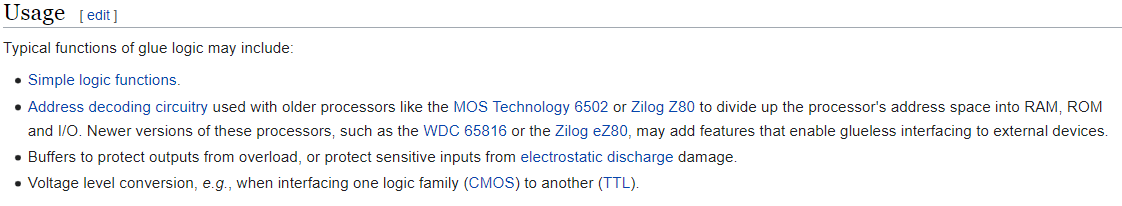
2. It takes more time to write testbench.

1. What is glue logic?

Why will we use glue logic to simplify our SVA expression?



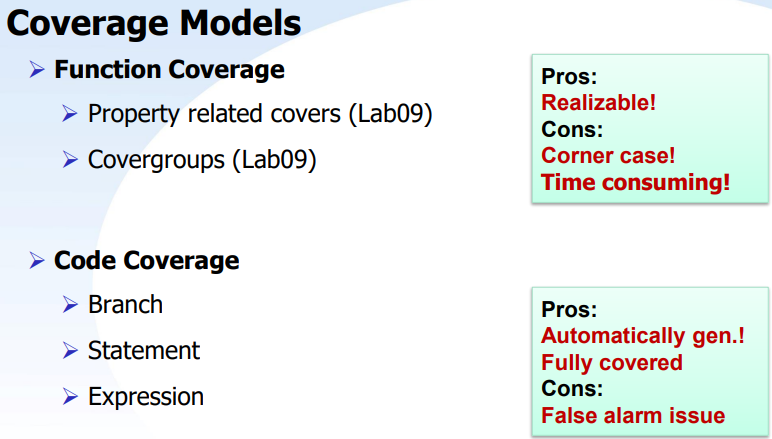
**glue logic** is the custom [logic circuitry](https://en.wikipedia.org/wiki/Digital_electronics) used to interface a number of [off-the-shelf](https://en.wikipedia.org/wiki/Commercial_off-the-shelf) [integrated circuits](https://en.wikipedia.org/wiki/Integrated_circuit).



In sum, we can use glue logic to make us have better coding style. Then we can make SVA more simplistic.

1. What is the difference between Functional coverage and Code coverage?

What’s the meaning of 100% code coverage, could we claim that our assertion is well enough for verification? Why?



**Functional coverage:**

1.Describe specific states, conditions, or sequences to be verified.

2. Requires planning, coding, and debug

3. Noise-free – nothing is don’t-care

4. Possible to represent all meaningful design functionality

5. Implements the verification plan – what needs to be verified

6. May be incomplete due to human error

**Code coverage:**

1.easy to generate automatically.

2. Guaranteed to be structurally complete – not prone to human error

3. Can be noisy – “don’t-care” or duplicate covers

4.It’s may not able to capture all meaningful design functionality

5. Standard models capture much of the meaningful behavior of the design

The 100% code coverage means that we carry out every code. But we can’t claim that our assertion is well enough for verification. The **Code coverage** can’t capture all meaningful design functionality. The **Code coverage** won’t check the correctness of calculation result.

1. What is the difference between COI coverage and proof coverage for realizing checker’s completeness? Try to explain from the meaning, relationship, and tool effort perspective.

**COI coverage:**

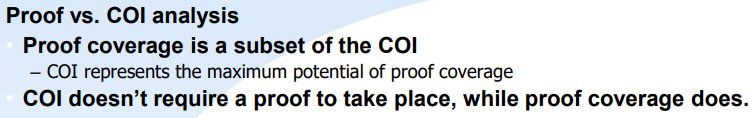
**Each assertion affected by some cover items:** The region will be called Cone-Of-Influence.

**Finds the union of the all assertion COIs:** The remaining Out of COI cover items indicate holes in the assertion set – code that is not checked by any asserts – Fast measurement – no formal engines are run.

**Proof coverage:**

**Find the region cannot truly influence assertion status**

R**epresents the portion of the design verified by formal engines**: Subset of the COI – COI represents the maximum potential of proof coverage.



1. **What are the roles of ABVIP and scoreboard separately?**

Try to explain the definition, objective, and the benefit.

**ABVIP:**

Definition: The Assertion Based Verification Intellectual Properties (ABVIPs) are a comprehensive set of checkers and RTL that check for protocol compliance.

Objective: Verifying a protocol and analyzing its completeness is a key challenge for engineers these days

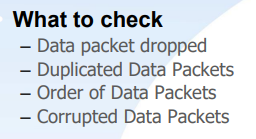
Benefit: We can use it without writing properties by ourselves.

**Scoreboard:**

Definition: Scoreboard behaves like a monitor

Objective: Observe input data and output data of DUV

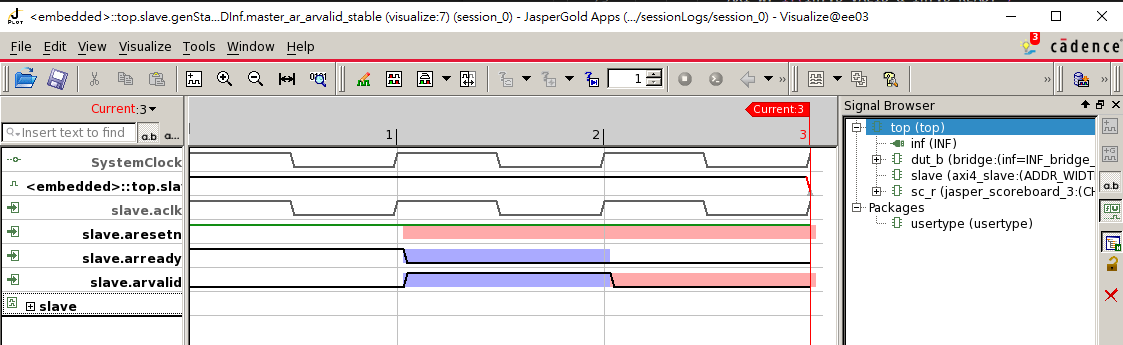
Benefit: We can use it to check the simple cases and reduce barrier for adoption.



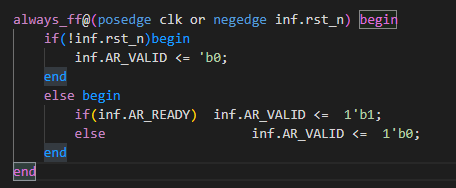
1. List four bugs in Lab Exercise

What is the answer of the Lab Exercise?

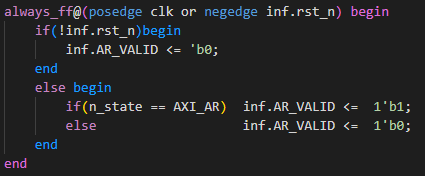
**The first bug :** AR\_VALID is not dependent on the AR\_READY. This is a wrong conditional expression.

****

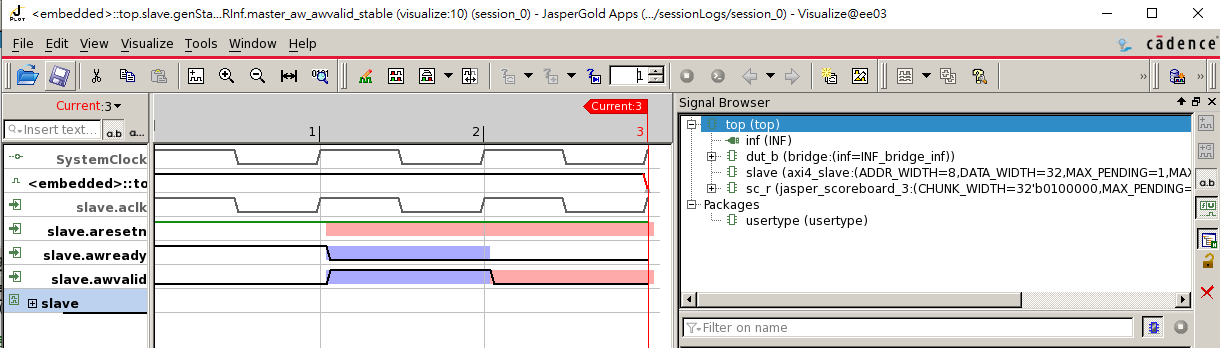
The wrong code.

****

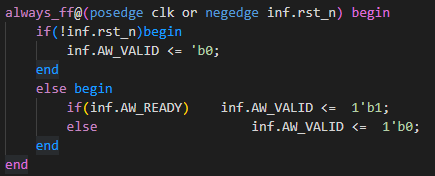
We have to change it to below. The AR\_VALID is dependent on the state.

****

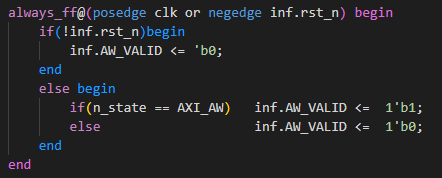
**The second bug :** the same wrong as the first bug.AW\_VALID is not dependent on the AW\_READY.

****

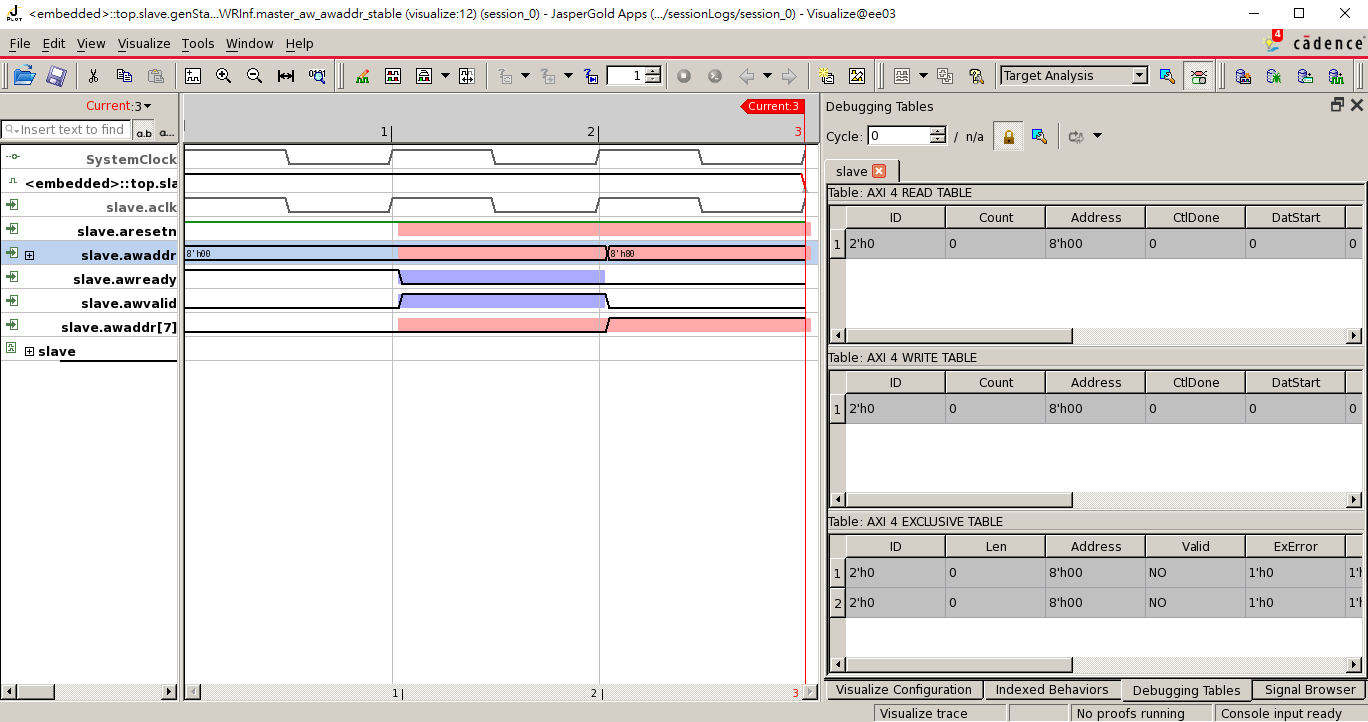
The wrong code.

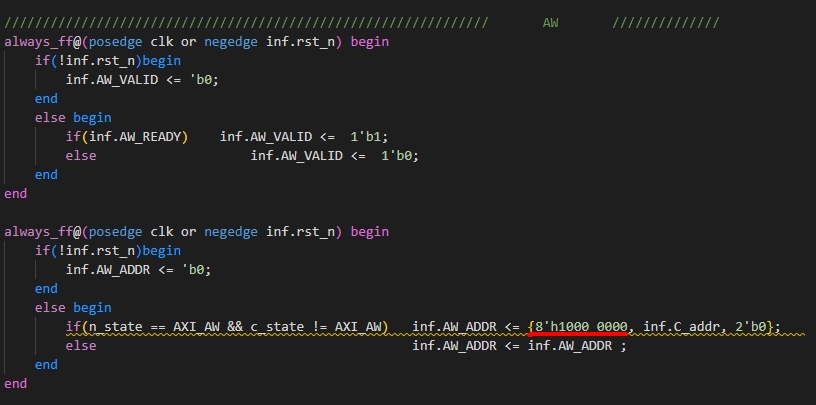
****

We have to change it to below. The AR\_VALID is dependent on the state.

****

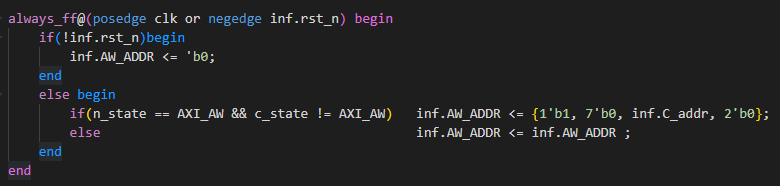
**The third bug :** awaddr is wrong.



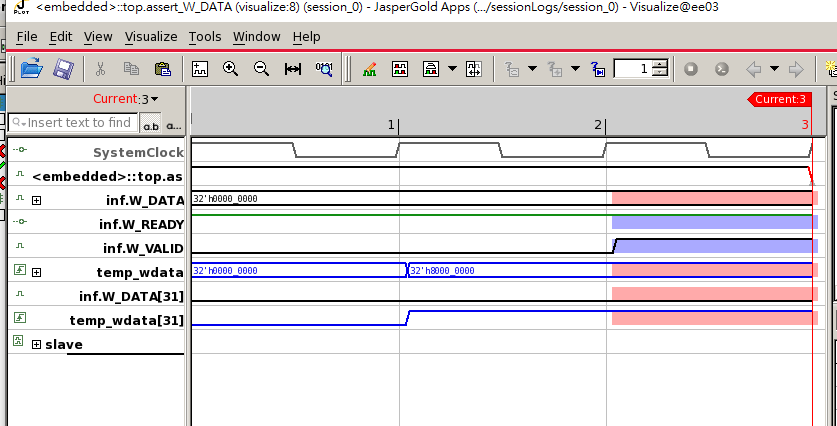


8’h1000\_0000 is equal to 8’h00 because one number is 4bits in hexadecimal system.

The correct answer is 1’b1. (below)



**The fourth bug :**  TheW\_DATA is not equal to temp\_wdata when W\_VALID & W\_READY is high.



The correct answer.

