IC Lab Formal Verification
Bonus Report 2025 Spring

Name: 賴御安 **Student ID:** 110550168 **Account:** iclab036

(a) What is Formal verification?
What's the difference between **Formal** and **Pattern** based verification?
And list the pros and cons for each.

(1) Formal verification is a method to mathematically prove the correctness of a hardware design with respect to a certain specification or property. Formal tests all possible stimulus. The first cycle tests all combinations of uninitialized registers and at every cycle tests all combinations on inputs and undriven wires.
(2) Pattern based verification tests the hardware design by simulating with a lot of test data to find out the error in the design. Formal based verification is a method to mathematically prove the correctness of a hardware design, we write assertions and constraints then visit DUT states.
(3)
Formal verification:
Pros: We go through all reachable states, so it is more systematic and has less randomization. Therefore, it has higher quality.
Cons: The states may be exponentially large as the design size grows. We sometimes cannot reach all states in a limited amount of time.
Pattern verification:
Pros: Easy and efficient way to test our design without writing assertions and constraints. Only require the test data and the correct answers.
Cons: We need lots of simulations to reach all testcase. And also, we need to generate all test data by the designers, which means that it is easy to miss some corner cases.

(b) Explain SVA (SystemVerilog Assertions) and the roles of Assertion, Cover, and Assumption.
What is glue logic?
Why will we use **glue logic** to simplify our SVA expression?

(1) SVA is a language for expressing properties, which we can use verilog, systemverilog or VHDL to write. SVA is a powerful alternate way to write constraints, checkers and cover points for your design. The role of Assertion checks that a certain behavior must always happen. It can check invariants, IF conditions in the same cycle or after certain cycles, or sequence existences. If violated, it flags an error. The role of Cover is to check whether a scenario ever happened in the simulation or not. The role of Assumption is to specify some constraints about the design or the environment we want to assume true for the purpose of the formal verification. It helps reduce state space by ruling out illegal scenarios.

(2) Glue logic is using auxiliary logic to observe and track events. It simplifies the coding when we have complicated code or SVA.
(3) We use glue logic to simplify our SVA expression when we have a complicated SVA. Since JasperGold is fine with all SVA or SVA with some glue logic, we can use the later one to improve readability and clarity.

(c) 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?

(1)
Functional coverage:
The designers write the constraints based on the requirements, which is the verification plan. It needs time to write the constraints and we can represent all meaningful design functionality.
Code coverage:
It enumerates all branches, statements and expressions and checks the execution. It is easy to generate automatically
(2) It means that every line, branch, condition, and FSM transition has been executed, but it does not mean that the design has been verified and is functionally correct.

(d) 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.

Meaning: COI coverage is the union of all assertions COIs. It measures how much of the design is structurally reachable from the checker. Proof coverage is the subset of the COI, which includes the part that truly influences the assertion status.
Relationship: The connection of the COI coverage and the proof coverage is the checker coverage. COI Coverage checks if the assertion is structurally connected to the relevant logic, while proof Coverage checks if the assertion is strong enough to prove behavior across that logic.
Tool effort: Proof coverage requires more time since we need to run formal engines and identify more unchecked code.

(e) What are the roles of **ABVIP** and **scoreboard** separately?
Try to explain the definition, objective, and the benefit.

ABVIP:
Definition: It is an IP that contains complete checkers and RTL to verify a protocol.
Objective: It provides a reliable verification for the designers to use.
Benefit: Designers can depend on it to verify their design due to the IPs' reliability. Also, it can increase the development time.
Scoreboard:
Definition: Scoreboard is like a monitor which monitors the inputs and outputs of the DUV. It

records the expected outputs and compares with the actual outputs.
Objective: It verifies that data is not lost, duplicated, or corrupted and verifies that outputs match the expected value.
Benefit: We can ensure that every input stimulus results in a correct output. Also, we can know which stimulus is done and which is not.

(f) Among the JasperGold tools (Formal Verification, SuperLint, Jasper CDC, IMC Coverage), which one do you think is the most effective based on its functionality and typical application scenarios? Please explain your reasoning by describing a hypothetical scenario where this tool would be particularly beneficial, and discuss any potential challenges or limitations that might arise when using it.

Formal Verification is the most effective based on its functionality and typical application scenarios. For example, if we are going to design a multi-port packet router with no packet loss and FIFO ordering. When we meet the race condition problem, it could be quite difficult to detect by using simulation. But using Formal Verification, we can prove it functionally correct much easier by assertion. For the packet loss case, we can alse easily use assertion to detect. To cover all cases, formal verification is easier by coverage, which is the same as what we had done in lab10.
The potential challenges may be verifying a big design. The states will be too many and lead to a time consuming job. We also need to write the assertion carefully to prevent overconstrained or underconstrained stimulus.