IC Lab Formal Verification Bonus Report 2024 Fall

Name: 邱士豪　　**Student ID:** 313591005　　**Account:** iclab154

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

My answer:
Formal based verification is a systematic process that uses mathematical reasoning to verify the design intent during the implementation of RTL. It can algorithmically checks all possible input values that may change over time.
Formal based verification verifies all possible cases, while pattern based verification only checks limited possibilities.
Formal based verification ensures the correctness of the design, but its resource intensity is extremely high for larger block design due to the larger search space.
Pattern based verification allows the designers to reach a certain level of coverage of the design in the initial testing stage. However, it may not cover some corner cases which lead to system failure.

(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?

My answer:
Assertion will check if the design meets the specifications we defined or perform the specified behavior.
Cover will track and monitor whether the specific input values or input combinations are hit during the simulation.
Assumption will restrict the design to perform under specified conditions and allow the verification tool to ignore some failure caused by those unwanted conditions.

Glue logic means using auxiliary logic to observe and track events. We use glue logic because glue logic comes at no extra price. JasperGold does not care whether property is all SVA or glue logic.

(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?

My answer:
Functional coverage: Determine if the user-specified states, conditions and sequences are covered.
Code coverage: Enumerates all possible input values by using brute force automatically.

100% code coverage implies that all the targets that users define are all covered, and it also gives high confidence to the correctness of the design under those targets. However, we cannot claim that our assertion is well enough because there still might be some unexpected, incomplete or incorrect assertions.

(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.

My answer:
COI coverage provides faster measurement because no formal engines are needed, and it does not require a proof like proof coverage does.
Proof coverage is actually a subset of COI coverage. It utilizes formal engines and full-and-bounded proof to perform measurement.

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

My answer:
ABVIP: Verification IP that can check the correctness of the circuit design with written assertions.
Scoreboard: Monitor IP which observe both the input data and the output data of DUV.

(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.
My answer:
I only tried two tools mentioned in the question, which are Jasper CDC and IMC Coverage.
Among these tools, I might think Formal Verification would be the most effective one.
In most of the scenarios, we are often asked to finish some small projects in the college. We have to design a circuit and meets the specification by our own hands, and then Formal Verification comes in handy under such circumstances.
Formal Verification is helpful in verifying the correctness of my design in early stages, allowing me to debug really fast.