IC Lab Formal Verification Bonus Quick Test 2023 Fall

312510152/iclab059 許澤群

(a) What is Formal verification?

Formal verification 是透過數學或邏輯方法來驗證系統的正確性。Formal verification 不做時序的檢查，而是窮舉出每個 clock 可能的輸入和狀態，來驗證系統的正確性。

What's the difference between Formal and Pattern based verification? And list the pros and cons for each.

Formal verification 和 pattern based verification 的主要差異為驗證方法，如上所述 formal verification 透過數學和邏輯方法證明系統的正確性，pattern based verification 則是透過測試 pattern 檢查常見的問題或錯誤。

Formal verification 的優點如下，

(1) 精確度高

使用數學方法來證明在所有可能的輸入情況，且方法的隨機性幾乎為零，因此具更高的確定性。

(2) 前期除錯

可以檢測到其他驗證方法（例如 Pattern based verification）可能忽略的潛在問題或是邊界情況，有助於在設計前期找到錯誤，並即時修復。

(3) 獨立於 testbench

Formal verification 不依賴 testbench，相較使用 testbench 模擬，formal 透過列舉 property 的方式便可以測試。且因為獨立於 testbench，formal 提供另一種檢查設計的角度，有助於提高驗證品質。

(4) 具重複使用性

Formal verification 的 assertion 和 cover 通常可以在設計的不同階段中重複使用，提供額外價值。

Formal verification 的缺點如下，

(1) 資源消耗高

驗證每個 clock 可能的輸入和狀態需要大量的計算時間和運算資源，因此不適合應用在複雜的電路設計。

(2) 可偵測的錯誤類型受限

Formal verification 無法驗證某些類型的錯誤，例如時序上的功能正確性或效能等。

Pattern Based Verification 的優點如下，

(1) 效率高

可以對常見的錯誤和問題設計對應的 pattern 進行驗證，快速分辨問題。且測試 pattern 通常較直觀，使用上比較容易。

(2) 互補性

Pattern based verification 可以與其他驗證方法（例如 simulation based verification）結合使用，提高整體驗證的覆蓋率。

Pattern Based Verification 的缺點如下，

(1) Pattern 受限

測試的 pattern 受限於已知的情況，雖然可以透過亂數方式提高覆蓋率，仍有可能遺漏某些未考慮到的狀況。

(b) What is glue logic?

Glue logic 是指將原本電路中的訊號拉出來，組合而成的一個新的邏輯訊號。

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

Glue logic 將複雜的訊號關係簡化成一個新的邏輯，因此原本需要對各個訊號撰寫 SVA 表達式，用 glue logic 的 SVA 表達式就可以達到相同的效果，簡化 SVA 的複雜度，提高可讀性，也有助於模組間的驗證。

(c) What is the difference between Functional coverage and Code coverage?

Functional coverage 主要檢查設計中的功能和規格是否被測試，確保設計在不同的情況下都能正常運作。Functional coverage 是由設計者進行規畫和設計，因此容易產生人為疏失或考量不足等問題。

Code coverage 主要檢查程式碼中的 branch、statement 和 expression 是否被執行，確保程式碼的每部分在測試過程中都有被執行。Code coverage 是由 EDA 工具產生，可以避免人為的錯誤，但有時會檢查到一些不需檢查的部分。

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

100%的 code coverage 代表程式碼各部分都有被執行，但因 code coverage 沒有進行功能上的驗證和測試，無法保證功能的正確性與覆蓋率，所以只有 100% code coverage 的驗證並不足夠。

(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.

(1) Meaning

COI coverage 會對 assertion 所檢查的邏輯，往前回朔所有影響該邏輯的 cover item，並對這些 cover item 進行驗證。proof coverage 則是將電路轉換成簡化後數學邏輯模型，對所檢查的邏輯，僅驗證直接影響該邏輯的 cover item，透過 formal engine 驗證每個 cycle 所有可能的情況。通常 proof coverage 是 COI 的一個分支。

(2) Relationship

COI coverage 通常和 simulation verification 相關，透過測試的 cases 來檢查 COI。Proof coverage 則是和 formal verification 相關，透過 formal engine 驗證所有可能的情況。

(3) Tool effort

COI coverage 主要對 assertion 和 assume 做覆蓋率分析，tool effort 較低。Proof coverage 需要透過 formal engine 驗證所有可能的情況，tool effort 較高。

(e) 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.

(1) Definition

ABVIP 是一種驗證的 IP，內容包含各式測試和檢查的 assertions，通常用於驗證設計的特定功能或協議。Scoreboard 是一個驗證環境的組件，其角色類似監視器，負責比對設計的實際輸出和期望輸出，確保設計的功能正確性。

(2) Objective

ABVIP 的目標是提供驗證的 solution，加速測試和驗證過程。Scoreboard 的目標則是檢查設計的功能是否正確。

(3) Benefit

對於定義明確且規格統一的功能或協定，ABVIP 讓使用者無需額外撰寫 checker，減少測試負擔，並加速驗證過程。且由專業驗證團隊開發的 ABVIP，驗證結果也較為嚴謹和可靠。Scoreboard 可以檢查錯誤，如資料遺失、資料重複等，提供錯誤訊息有助於修復問題，且 Scoreboard 自動化特性也可以降低人工操作的負擔。

(f) List four bugs in Lab Exercise. What is the answer of the Lab Exercise?

在 bridge.sv 中有四個錯誤，分別是（上圖為錯誤程式碼，下圖為正確答案的程式碼），

(1)

```
if(inf.AR_READY)  inf.AR_VALID <=  1'b1;
if(n_state == AXI_AR)  inf.AR_VALID <=  1'b1;
```

(2)

```
if(inf.AW_READY)    inf.AW_VALID <=  1'b1;
if(n_state == AXI_AW)   inf.AW_VALID <=  1'b1;
```

(3)

```
if(n_state == AXI_AW && c_state != AXI_AW) inf.AW_ADDR <= {8'h1000_0000, inf.C_addr, 2'b0};
if(n_state == AXI_AW && c_state != AXI_AW) inf.AW_ADDR <= {1'b1, 7'b0, inf.C_addr, 2'b0};
```

(4)

```
if(inf.C_in_valid && inf.C_r_wb)       inf.W_DATA  <= inf.C_data_w;
if(inf.C_in_valid && !inf.C_r_wb)       inf.W_DATA  <= inf.C_data_w;
```

(g) Among the JasperGold tools (Formal Verification, SuperLint, Jasper CDC, IMC Coverage), which one have you found to be the most effective in your verification process? Please describe a specific scenario where you applied this tool, detailing how it benefited your workflow and any challenge you encountered while using it.

依照我的使用經驗，我認為 Formal Verification 是最方便且實用的工具。使用 ABVIP 來驗證 AXI4 的設計，不僅不須額外撰寫 testbench 或 checker 進行驗證，節省非常多時間，其驗證結果也更加嚴謹和可靠，對驗證階段有非常大的幫助。