1. What are the advantages of using code coverage in **simulation-based functional** verification? If you got 100% statement coverage in the simulation, does it guarantee that the design is correct? Please explain the reasons of your answer.

* 速度快，可在大電路中使用，可測試驗證design是否全面，也可以減少不必要的code statement
* 無法保證design為100%正確，coverage 100%只代表我們所寫的code的每個部分都有被測試到，但是若有邏輯上疏漏未寫到的condition便無法被測試

1. What are the advantages and disadvantages of **formal verification**? What are the keys to improve the efficiency of formal verification?

* 優點: 能覆蓋完整的design狀態、速度快
* 缺點: 需要將ROBDD儲存於memory中，若design較大可能會導致state explosion
* Improvement: 使用partial或是bounded model checking 可減少狀態空間，亦可限制檢查的cycle數量降低memory占用

1. Using the CTL formulas, how can we verify the design functionality? What are the issues of CTL-based verification?

* CTL提供所有可能的路徑，藉由已logic operators, temporal operators, path quantifiers構成的CTL formulas可驗證特定條件屬性是否發生
* Issue: state explosion，CTL是使用tree紀錄各種可能路徑，若state過多，memory可能不夠用

1. Why do we need analog behavioral models for mixed-signal system verification? Compared to the original Verilog language, what are the major extensions of Verilog-AMS?

* 由於使用HSpice模擬真實電路花費時間很長，因此利用數學模型去模擬behavioral level會較省時
* 相較於普通Verilog，Verilog-AMS支援analog functions: 微分、積分、contribution operator <+，也包含新的單位宣告electrical等analog description

1. Given a 4-phase sin-wave generator, please show the analog section of the behavioral model for this circuit based on Verilog-A. Its output voltage swing is 1.0V with 0.4V DC shift, and the phase difference between consecutive outputs is 90. The signal frequency is 250MHz with maximum slew rate .

module sinwave (out1, out2, out3, out4);

`define PI 3.1415927

output out1, out2, out3, out4;

real phase;

parameter real fc = 2.5e6;

electrical t1, t2, t3, t4, out1, out2, out3, out4;

analog begin

phase = 2.0 \* `PI \* fc \* $realtime();

V(t1) = 0.5 \* sin(phase);

V(t2) = 0.5 \* sin(phase + `PI / 2);

V(t3) = 0.5 \* sin(phase + `PI);

V(t4) = 0.5 \* sin(phase + `PI \* 3 / 2);

V(out1) <+ slew(V(t1),8e7,-8e7) + 0.4;

V(out2) <+ slew(V(t2),8e7,-8e7) + 0.4;

V(out3) <+ slew(V(t3),8e7,-8e7) + 0.4;

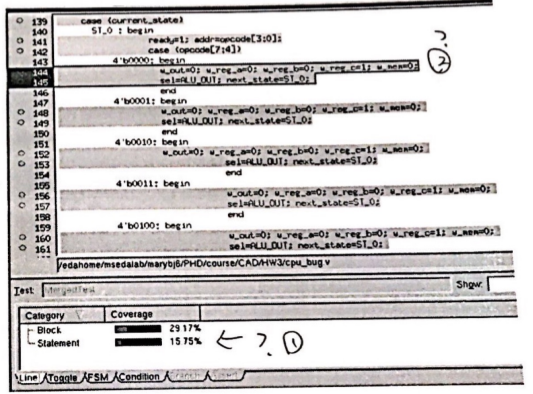
V(out4) <+ slew(V(t4),8e7,-8e7) + 0.4;

end

endmodule

1. What are the benefits and limitations of **simulation-based approach** and **equation-based approach** for analog design automation? Why does the difference between synthesis results and post-layout simulation often exist?

* Simulation-based: 可以應用於各種不同的design且正確率高，缺點為耗時且需要花費額外時間產生pattern
* Equation-based: 由於可以直接求解所需的design parameter，因此速度很快，但是由於model equation通常伴隨各種近似與假設，因此結果並沒有那麼準確
* Pareto-front-based: They can provide a tradeoff between different design objectives, allowing designers to choose a design point that best meets their needs / may not be able to find the optimal solution, as the optimal solution may lie outside of the search space, may require a large number of design simulations in order to generate the Pareto front, which can be computationally intensive.
* 通常在synthesis時並不會將parasitic effect, physics effect納入考量，因為這些參數必須有layout後才能確定，因此可能在synthesis時是符合spec，到了post-sim後就不符合了

1.  While you are running code coverage analysis, you got a coverage report as shown right. What are the meanings about the numbers in the bottom window and the red lines in the upper window? In your opinions, is the result good enough for functional verification? What can we do for the users who are not satisfied with the results?

* Upper: 沒有被執行到的statement
* Bottom: block, statement coverage 的比例
* 圖片所示並不是一個好的結果，statement coverage 未達100%代表code中仍有部分未被檢測。就算statement coverage達到100%也僅僅表示code中各個部分都有被檢測，但是code中未考慮到的部分仍然無法測試，因此至少需要達到100%，verification才具有基本的可信度
* Design 沒錯的前提下，增加random pattern數量或是設計corner case

1. What are the benefits and drawbacks of using the **simulation-based approach** and **formal-based approach** for functional verification? What are the keys to improve the efficiency of simulation-based verification?

* Simulation-based:
* 優點: 可以使用簡單方法驗證複雜design
* 缺點: 需要大量pattern 進行驗證，很難完全覆蓋各種可能，速度較慢
* Formal-based:
* 優點: 能夠完整覆蓋設計狀態，速度快
* 缺點: 需要memory儲存ROBDD，若design過大可能導致state explosion
* Improvement for simulation-based: faster simulator/testbench tools, better hardware, parallelize the verification process

1. What is “equivalent checking” verification technique? What are the benefits and limitations of this technique? Is it possible to have false alert in real cases with this techniques? Please explain your reasons.

* 檢查兩個design的功能是否完全相等，通常用於檢查兩gate level circuit 或HDL 與gate level design 有無mismatch
* 優點: 檢查所有可能的input組合(solve SAT problems)
* 缺點: 只可檢查implementation errors, 無法檢查design errors
* No method is foolproof, and there is always a chance for human error or other factors that can lead to a false result.

1. What are the difference of using **top-down** and **bottom-up** approaches for analog behavioral modeling in mixed-signal system designs? How can they help designers in the design flow?

* Top-down: verify whole system roughly before implemented, suitable for new design
* Bottom-up: more accurate, consider non-ideal effects, suitable for flattened transistor-level designs and IP-based designs
* The top-down approach helps develop a high-level understanding of the system behavior in early stage, while the bottom-up approach can be useful for verifying the accuracy of the model at a detailed level and identifying implementation-specific issues.

1. Given a 2-bit DAC circuit, please show the analog section of the behavioral model for this circuit based on Verilog-A. The maximum input voltage is 1.0V, and the signal frequency is 250MHz. The output signal has a 0.5ns rising time and a 0.4ns falling time, and its input-to-output delay time is 1ns.

以下是3-bit的範例，較為清楚

`timescale 1ns/1ps

module dac\_3bit (

input [2:0] digital\_in, // 3-bit digital input

output real analog\_out // Analog output

);

// 定義參數

parameter real V\_MAX = 1.0; // 最大輸出電壓

parameter real DELAY = 1.0; // 輸入到輸出的延遲時間 (ns)

parameter real RISE\_TIME = 0.5; // 上升時間 (ns)

parameter real FALL\_TIME = 0.4; // 下降時間 (ns)

// 中間變量

real v\_out;

// 行為模型，n-bit input就有2 種case

always @\* begin

case (digital\_in)

3'b000: v\_out = 0.0;

3'b001: v\_out = V\_MAX / 7.0;

3'b010: v\_out = 2 \* V\_MAX / 7.0;

3'b011: v\_out = 3 \* V\_MAX / 7.0;

3'b100: v\_out = 4 \* V\_MAX / 7.0;

3'b101: v\_out = 5 \* V\_MAX / 7.0;

3'b110: v\_out = 6 \* V\_MAX / 7.0;

3'b111: v\_out = V\_MAX;

default: v\_out = 0.0;

endcase

end

// 延遲和上升/下降時間處理

analog begin

analog\_out <+ transition(v\_out, DELAY, RISE\_TIME, FALL\_TIME);

end

endmodule

1. Why **simulation-based** functional verification often takes a lot of time for large designs? Is formal-based functional verification able to solve those issues? If so, why formal verification is not so popular in current design flow?

* 若要使用simulation-based verification驗證所有可能的case需要大量pattern，且產生testbench也需要額外的時間
* Formal 不需要input pattern, 使用理論推倒而可達100% coverage
* 現在的design通常很大，使用formal-based會有state explosion的問題

1. Compared to the original Verilog language, what are the major extensions of Verilog-AMS? If we use the original language to model analog circuits, what issues should be solved?

* 相較於普通Verilog，Verilog-AMS支援analog functions: 微分、積分、contribution operator <+，也包含新的單位宣告electrical等analog description
* 在普通的Verilog中我們無法使用正確的analog model進行驗證，因此只能夠使用近似的方式導致正確率下降

1. What are the major non-ideal effects in an OPA circuit? How can we model those effects accurately in behavioral models?

* Non-ideal effects: noise, OP gain, slew rate, setting response
* 我們可以從simulation result中extract出上述effects:
* Noise: obtained by Hspice command .noise
* Slew rate: measure the slew rate of two different inputs to find the slew rate equation
* Setting response: 把標準公式的變因換為可量測的，再代入求得值

由以上參數建立的behavioral model 較準確