CSE 313s
Selected Topics in Computer Engineering
Sheet 2

1. Static Analysis involves simulating a model.
    a. True [✓ b. False]

Static Verification utilizes search and analysis to find all targeted failures without running a testbench

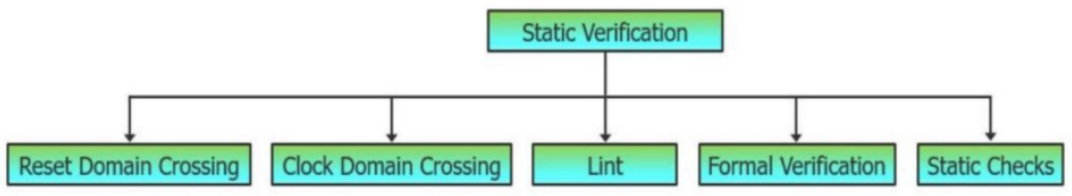
---

2. Which of the following is a technique covered in Static Analysis?
    a. Formal Verification
    b. Model checking
    c. Equivalence checking
    [✓ d. All of the above]

Formal Verification is considered as Static Verification and can be classifies as: **(1) Equivalence checking**
(2) Model checking

---

3. Describe three common techniques used for static verification in hardware design



(1) Reset Domain Crossing: This refers to verifying the proper handling of reset signals as they cross from one clock domain to another.

(2) Clock Domain Crossing: This involves ensuring correct data transfer between different clock domains within a design. When data crosses clock domains, issues such as metastability and data loss can occur if not handled properly.

(3) Lint: Linting is a static analysis technique used to identify potential issues in HDL code. It checks for coding style violations, potential bugs, and non-synthesizable constructs. Lint tools analyze the code without executing it and provide feedback on coding practices and potential problems.

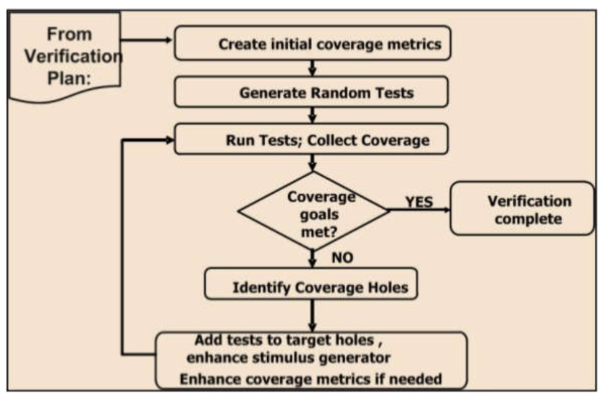
(4) Formal Verification: Formal verification involves mathematically proving that a design meets its specified requirements or properties. It uses formal methods such as model checking and theorem proving to exhaustively analyze all possible behaviors of a design and verify its correctness with respect to a given specification.

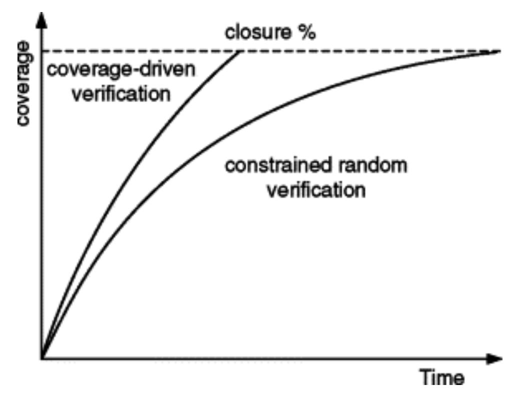
(5) Static Checks: This is used to verify the timing performance of a digital design without actually simulating its behavior. It ensures that the design meets its timing requirements, such as setup time, hold time, and clock-to-q delays, across all possible operating conditions.

---

4. What is a coverage driven verification?

الcoverage driven verification هو approach مبني على الfunctional و الcode coverages من خلاله أقدر أقيس هل قدرت أ-verify كل features الdesign ولا لا.
بنبدأ من الverification plan اللي اتكلمنا عنها الsheet اللي فات ومنها بنحط الcoverage metrics اللي هي الfeatures اللي الdesign بيقدمها, وبعدين نعمل random tests كتير وفي نفس الوقت بكون بـmonitor الcoverage, ونقعد بقى في loop كده, افضل اروح أ-run random tests واشوف الcoverage وصل كام انهاردة, لقيته لسة قليل, اروح اعدل في الconstraints بحيث أغطي الcoverage holes و هكذا لحد ما أغطي كل الfeatures


From Verification Plan:
Create initial coverage metrics
Generate Random Tests
Run Tests; Collect Coverage
Coverage goals met?
YES
Verification complete
NO
Identify Coverage Holes
Add tests to target holes , enhance stimulus generator Enhance coverage metrics if needed



5. Name three of the coverage metrics that we discussed in lectures.

- **Functional coverage:** It covers the functionality of the design. It is derived from the design specification. Tools can't not generate an automatic functional coverage.
- **Code coverage:** It refers to the measurement of how much of the design's code constructs have been exercised by the test cases.

(1) Statement coverage: It is a straightforward metric which tells you how much of the statements in the source code are executed during your test

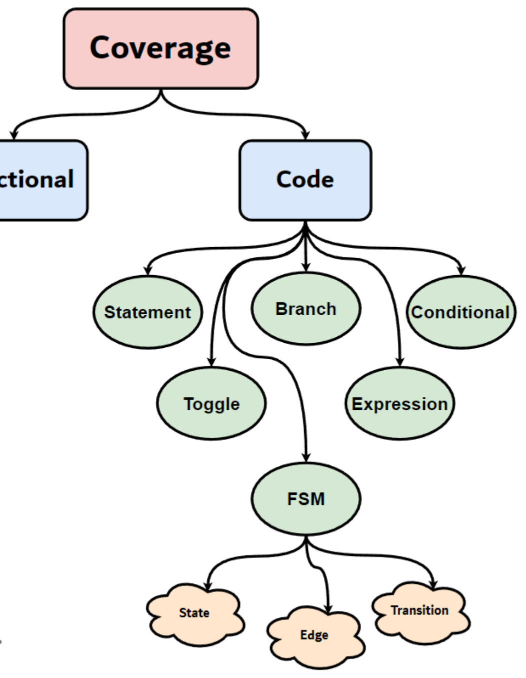
(2) Toggle coverage: It tracks the percentage of flip-flops or registers that have changed state during simulation, indicating whether these elements have been exercised.

(3) FSM coverage: It gives a coverage about the state transitions, and the different arc coverages.

(4) Branch coverage: It evaluates the percentage of decision points (branches) in the code that have been taken during testing. It ensures that both the true and false branches of conditional statements have been exercised.

(5) Expression coverage: It considers the RHS of expressions.

(6) Conditional coverage: It gives you a measure of whether every Boolean subexpression is evaluated to true or false.



6.

Below is a Verilog code for a simple 2-to-1 multiplexer: Compute statement coverage for this Verilog module with the given input values.

ازاي كنت بحصي الtotal number of statements؟

- كل assign statement بتعتبر one statement
- بالنسبة للconditional statements --> • الif-else statement نفسها بتتحسب كstatement
  • الcondition مش بيتحسب معانا
  • بنعد عدد الstatements اللي جوة الtrue part والfalse part

- بالنسبة للLoop statements --> • الfor-loop statement نفسها بتتحسب بstatement واحدة
  • جزء الinitialization والcondition والincrementation كل ده مش بيتعد
  • بنجمع عدد الstatements اللي جوة الloop body

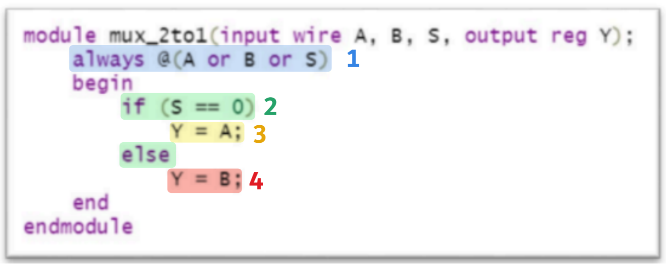
- بالنسبة للbreak, continue, return --> كل واحدة منهم بتتحسب بone statement

- بالنسبة للModule Instantiation --> • كل سطور الinstantiation مهما طالب بتتحسب بone statement

- بالنسبة للProcedural Blocks --> • الalways block نفسها بتتحسب بone statement
  • بنجمع عدد الstatements اللي جوة الblock body

- بالنسبة للFunction and Task Calls --> • كل function او task call بتتحسب بone statement

واخيرًا بالنسبة للmodule declaration بقى شخصيًا والbegin/end --> لا يتم حسابهم

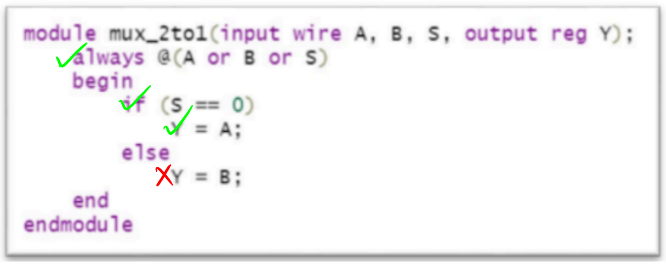


```
module mux_2to1(input wire A, B, S, output reg Y);
    always @(A or B or S) 1
    begin
        if (S == 0) 2
            Y = A; 3
        else
            Y = B; 4
    end
endmodule
```

كده انا حسبت الtotal number of statements وهو 4

بعد ما أ-apply الstimulus, هنروح نشوف مين فيهم اللي اتنفذ ومين لا

Input values:
- A = 1
- B = 0
- S = 0



```
module mux_2to1(input wire A, B, S, output reg Y);
   ✓always @(A or B or S)
    begin
       ✓if (S == 0)
          ✓Y = A;
        else
          ✗Y = B;
    end
endmodule
```

بالinputs دي, انا كده عديت على 3 statements من ال4

يعني الstatement coverage ب75%

7.

For the given code and input values compute the toggle coverage

ازاي كنا بنحسب الـtoggle coverage؟

الأمر في غاية البساطة, محتاج اشوف كل signal عندي إنها طلعت من 0 لـ1, ونزلت من 1 لـ0

ممكن كمان نحسب إذا كان كل signal مرت بالقيم المتاحة ليها, يعني مثلًا تكون الـsignal أصبحت بـ0 وأمست بـ1

Input values at consecutive clock cycles:
- clk = 1, rst = 0 (initial values)
- clk = 0, rst = 0
- clk = 0, rst = 0
- clk = 0, rst = 0

	1	0	1->0	0->1
clk	✓	✓	✓	✗
rst	✗	✓	✗	✗

بالـinputs دي, انا كده 4 cases فقط من الـ8 المنتظرين

يبقى كده الـtoggle coverage بـ50%

---

8. Assume in a coverage report, you found the functional coverage = 60% while the code coverage = 100%. What could be the reasons for that? In another experiment you found functional coverage = 100% while the code coverage =60%. What could be the reasons in that case?

خلينا قبل ما نحل السؤال نفتكر إن الـfunctional coverage هو عبارة إني بـlist الـfeatures بتاعة الـdesign بتاعي ومن ثم ابدا اعملها test بالـconstrainted randomization

المهم إننا لازم نكون عارفين إننا احنا اللي بنـlist الـfeatures دي

ثانيًا نفتكر إن الـcode coverage هو فقط بيبص على الـcode بتاع الـdesign هل مريت عليه كله ولا لا, بصرف النظر عن الـfunctionality.

وجب التذكير أنه لا غنى عن الأثنين, أنا محتاجهم صورة لرسم الصورة الشاملة للـtest بتاعي

الحالة الأولى:

لو أنا حققت 60% فقط من الـfunctional coverage ومع ذلك حققت 100% من الـcode coverage, ده ممكن يرجع لسبب من أتنين:

مشكلة من طرف الـRTL ==> إن الـdesigner باشا نسي يكمل شغله ويـdesign كل الـfeatures المطلوبة

مشكلة من طرف الـVerification Environment ==> إن الـtest مش كافي ومحتاج ألعب اكتر بقى في الـconstraints وكده زي ما اتكلمنا قبل كده ازود شوية scenarios او ناقص specific stimuli

سؤال منطقي, أزاي انا عديت على كل سطور الـcode وفي نفس الوقت محققتش كل الـfeatures؟

هديك مثال بسيط جدًا, ممكن يكون عندي مثلًا مثًلا مثلًا FSM, انا ممكن كـcode coverage أعدى على كل الـstates واحقق 100% coverage

أنما من جهة الـfunctionality, أنا محتاج flow محدد بين الـstates, مرة مثلا أروح من A لـB لـC, مرة تاني اروح من B لـA لـC ... وهكذا

لو لحظت في حالة إني عندي 3 states, فأنا عندي حوالي 8 scenarios محتاج أعدي عليهم على الرغم إن لو عملت scenario واحد بس فيهم هيحققلي الـ100% code coverage

تمام؟

الحالة الثانية:

لو انا حققت الـ100% من الـfunctional coverage ولكن 60% فقط من الـcode coverage, ده ممكن يعني حاجة من أتنين:

مشكلة من طرف الـRTL ==> إن الـdesigner كاتب code زائد عن الحاجة وملهوش لازمة, عرفت أحقق كل الـfeatures اللي محتاجها وفي الوقت محتجتش اعدي على كل الـcode أو ممكن يكون عندي ما يسمى بالـdead code واللي هو unreachable code مستحيل اوصله خالص, زي مثال الـstates اللي في صفحة 5

مشكلة من طرف الـVerification Environment ==> إن الـverification engineer نسي يحط كل الـfeatures اللي متفقين عليها في الـverification plan, فهو أه مكتوبله إنه خلص 100% من الـfunctional coverage, لكن الـlist بتاعت الـfeatures اللي اتفقنا عليها اصلًا ناقصة

EXTRA QUESTIONS

3. Select the disadvantage of using Model Checking
   a. Concurrent systems cannot be analyzed using this method.
   b. Producing a mathematical specification requires a detailed analysis of the requirements.
   c. They require the use of specialized notations that can only be understood by domain experts.
   d. All of the above

Complexity disadvantage of Model checking states that: "Formal verification techniques often require expertise in formal methods, mathematical logic, and model checking algorithms."

---

4. Which of the following is incorrect with respect to Model Checking?
   a. ✓ Model checking is particularly valuable for verifying concurrent systems
   b. Model checking is computationally very expensive
   c. ✓ The model checker explores all possible paths through the model
   d. All of the above

- Fairness properties of Model checking are often applied to concurrent or distributed systems to ensure that all processes or components of the system are given a fair chance to progress or access shared resources.

- Scalability disadvantage of Model checking states that: "Formal verification becomes computationally expensive for large designs" → in case of large designs only

- Completeness advantage of Model checking states that: "Formal verification aims to exhaustively explore all possible behaviors and corner cases of a hardware design"

---

5. What is a BDD?
   a. Boolean Decision Diagram
   b. Binary Decision Diagram
   c. Binary Decision Device
   d. Binary Device Diagram

BDDs are extensively used in CAD software to synthesize circuits (logic synthesis) and in formal verification.

---

6. What are the two inputs to the model checker?
   a. Implementation and specification
   b. BDD and FSM
   c. FSM of the design and properties
   d. Verification and Validation

The model checker takes two inputs: (1) the finite state machine representation of the design.
(2) the formal representation of some properties representing the specification (or the properties to be checked).

---

7. What is meant by the "counter example" generated by the Model Checker?

A counter example is an input sequence that shows how the system is put in the violating state. Counterexamples are very useful for debugging purposes.

الModel Checker بيسعى أنه يعدي على كل الpossible states للfinite-state model اللي انا عامله للsystem ويقارن بين الmodel والdesign الحقيقي.
لما يحصلي violation في state معينة, الModel Checker بيgenerate حاجة زي test pattern مثلًا أو input sequence اللي وصل بيه لحل الstate اللي فيها violation
بحيث يجي الdebugger بعد كده يستخدم الsequence ده ويمشي معاه واحدة واحدة ويشوف المشكلة حصلت فين بظبط ويعالجها.. هو ده الcounter example

---

8. An escaped bug is a design bug that is not detected during pre-silicon verification, and it is only caught during post-silicon verification.
   a. True
   b. False

الbug escaped دي هي الbug اللي مظهرتش في الconventional verification flow زي الtestbench والUVM وكده, ولكن ظهرت في الpre-silicon formal verification

---

9. Formal verification encompasses all techniques that leverage mathematical reasoning and proofs to determine the correctness of a silicon design.
   a. True
   b. False

الformal verification زي ما اتفقنا هو قائم على تحويل الdesign لmathematical model او finite-state model, وبعدين نقارن بين الmodel والdesign من غير testbench او حاجة

11. Which of the following techniques can be used to prove a general SAFETY PROPERTY?
    a. Equivalence checking
    b. Simulation
    c. Model Checking
    d. Emulation

Model checking is a formal verification technique used to verify whether a system meets a certain property or specification.

One of the properties is "Safety": it means an undesirable state would never happen.(something bad would never happen)

Consider the function: F = c.b + b.(a + d) + a'.d' For the questions below, use variable order (from top to bottom): a,b,c,d.

12. What are the cofactors of F w.r.t. a (i.e $F_{a=0}$, $F_{a=1}$)? Provide a simplified expression with the minimum number of literals

عشان أعرف الcofactors بتوع الF بالنسبة للa
وكل اللي بعمله إني بعوض عن الa مرة بـ0 ومرة بـ1, وأشوف بقى الF هتreduce لأيه, وهو ده هيكون الcofactors بتوعي

Given function F:

$$F = c \cdot b + b \cdot (a + d) + a' \cdot d'$$

Replace 'a' with 0

$$F_{a=0} = c \cdot b + b \cdot (0 + d) + 1 \cdot d'$$

$$F_{a=0} = c \cdot b + b \cdot d + d'$$

Replace 'a' with 1

$$F_{a=1} = c \cdot b + b \cdot (1 + d) + 0 \cdot d'$$

$$F_{a=1} = c \cdot b + b$$

$$F_{a=1} = (c + 1) \cdot b$$

$$F_{a=1} = b$$

ممكن نكمل الباقي عشان نشرح المبدأ ونعرف نرسم

in case of a = 0

Replace 'b' with 0

$$F_{b=0} = c \cdot 0 + 0 \cdot d + d'$$

$$F_{b=0} = d'$$

Replace 'b' with 1

$$F_{b=1} = c \cdot 1 + 1 \cdot d + d'$$

$$F_{b=1} = c + d + d'$$

$$F_{b=1} = 1$$

in case of a = 1

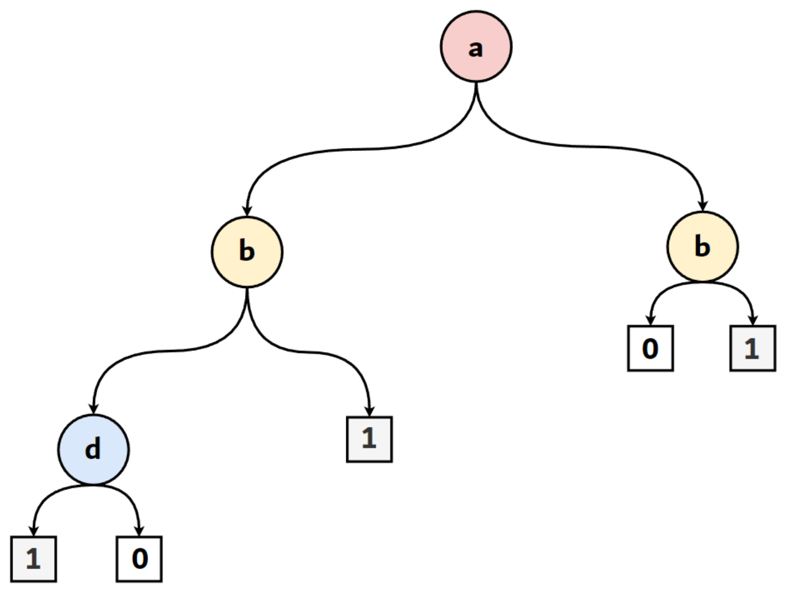
Replace 'b' with 0

$$F_{b=0} = 0$$

Replace 'b' with 1

$$F_{b=1} = 1$$

13. Draw the BDD for the function F. Remember to mark the edge with '0' or '1' to indicate 0 and 1 cofactors. Please draw the '0' edges on the left and the '1' edges on the right.



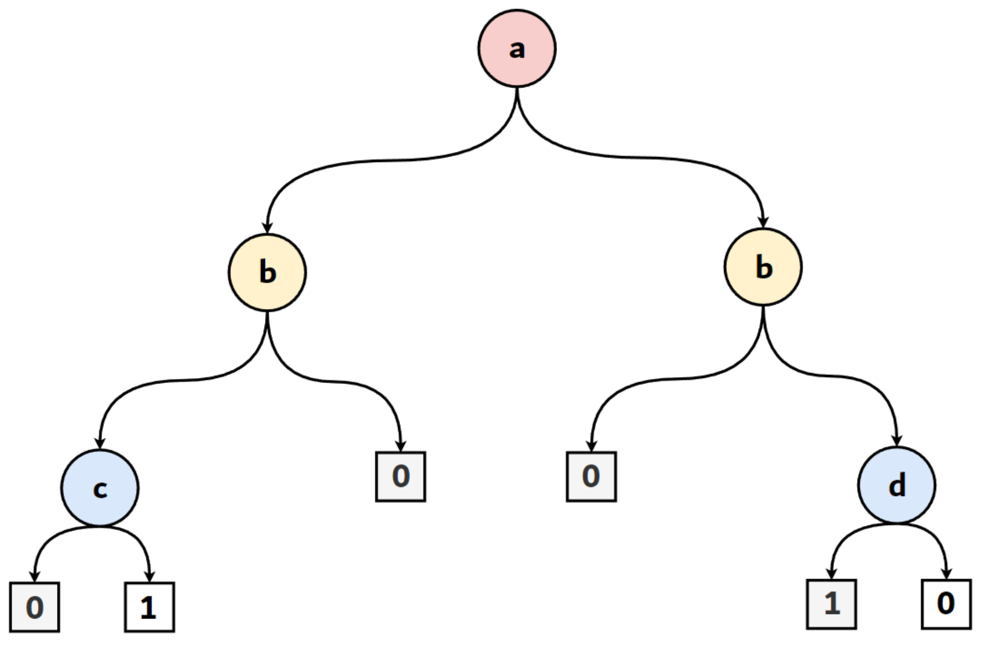
---

14. Consider the following function: F (a,b,c,d) = $\Sigma(2,3,12,14)$. Draw the BDD to represent this function. Remember to mark the edge with '0' or '1' to indicate 0 and 1 cofactors. Please draw the '0' edges on the left and the '1' edges on the right.

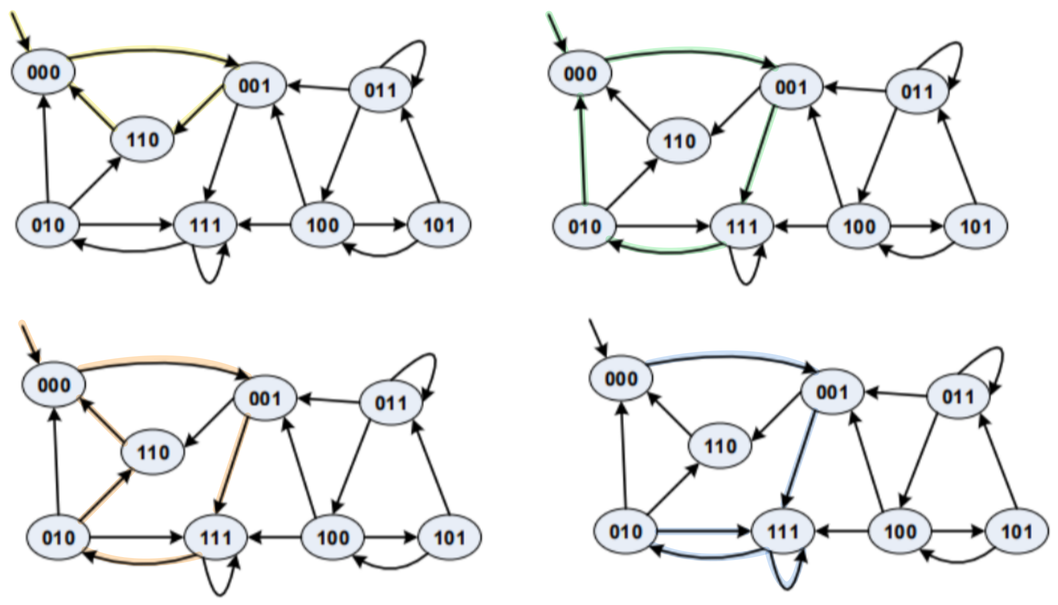
we need to represent it as a logical sum of products where each product term corresponds to one of the given minterms.

$c \cdot d$ \ $a \cdot b$	00	01	11	10
00			1	
01				
11	1			
10	1		1	

$$\therefore F = a' \cdot b' \cdot c + a \cdot b \cdot d'$$



15. For the following figure, perform a reachability analysis. Are there unreachable states?



Explored States:

000 ✓

001 ✓

110 ✓

010 ✓

111 ✓

011 ✗

101 ✗

100 ✗

} Unreachable States

16. For the previous problem, is the following property true?
"State 101 can eventually be reached from the initial state"

No, starting from 000, state 101 can't be reached.

17. What are the two types of assertions, and what are the differences between them?

(1) Immediate Assertion: - these assertions are evaluated continuously during simulation.
- Immediate assertions are typically used to verify conditions, such as checking the value of a signal or the relationship between multiple signals.

(2) Concurrent Assertions: these assertions are used to specify temporal properties that must hold true at specific points of time.

الـimmediate assertion انا بـcheck على حاجة لازم تكون متحققة طول الوقت, لازم!
انما الـconcurrent assertions انا بـcheck على حاجة مربوطة بحدث معين وخلاص, بتكون متحققة فترة من الزمن مش طول الوقت

18. What is the type of the following assertion. Explain its meaning.

```
property hash_delay_prop;
  @(posedge prop_clk) req ##5 gnt;
endproperty

hash_delay_check: assert property (hash_delay_prop);
```

It's a Concurrent Assertion

المقصود هنا إني بـcheck كل posedge للـclock, أنا منتظر إن الـsignal اللي اسمها req لما تكون بـ1, لازم الـsignal اللي اسمها gnt تكون هي كمان بـ1 ولكن بعد 5 clock cycles

---

19. What is the type of the following assertion. Explain its meaning.

```
assert (grant && request) begin
   $display ("Seems to be working as expected");
 end
else begin
   current_time = $time;
   #1  $error("assert failed at time %0t", current_time);
 end
```

It's a Immediate Assertion

المقصود هنا إني بـcheck طول الوقت, أنا منتظر إن الـsignal اللي اسمها request واللي اسمها grant, يكونوا الأتنين asserted طول الوقت, لو ده متحققش يطلع error

Juba