ECE 587 – Hardware/Software Co-Design
Lecture 13 Verification II

Professor Jia Wang
Department of Electrical and Computer Engineering
Illinois Institute of Technology

February 26, 2025

Reading Assignment

- This lecture: 7.2, 7.3
- Next lecture: 4, 5

Outline

Equivalence Checking

Model Checking

Equivalence Checking

- Make sure two designs are the same
  - in term of functionality
- Ensure the correctness of implementations and optimizations
  - Designers could make mistakes when implement a system manually.
  - Automated software tools for synthesis and optimization may have bugs.
- What does it mean by the *same* functionality?
  - Same output given same input?

Combinational Equivalence Checking

- What does it mean by the *same* functionality?
- Combinational circuits $C1$ and $C2$
  - At least they should have the same inputs and outputs
  - For any input x, the output $C1(x)$ should be the same as $C2(x)$
- One can build a larger combinational circuit $E(x) = (C1(x) \neq C2(x))$.
  - XOR each pair of corresponding bits of $C1(x)$ and $C2(x)$
  - Then OR the results together
- To prove $C1$ and $C2$ are equivalent is the same as to prove there is no x such that $E(x) = 1$.
  - If such x exists, you can use it to debug your circuits.

Satisfiability (SAT)

- Find x such that $E(x) = 1$ for a given combinational circuit E or prove no such x exists.
  - This is a well-studied problem: satisfiability (SAT)
- In theory, we *don't know* if there is a better way to solve it than to inspect the truth table of E.
  - i.e. to try all possible inputs, 2^N for N inputs
- In practice, efficient solutions have been developed during the last two decades.
  - That's the reason formal verification becomes more and more popular now.

FSM Equivalence Checking

- Concerning synchronous sequential circuits
  - i.e. RTL designs
- For any input trace, the two FSM should produce the same output trace.
  - From certain initial states
- What about states and state encodings?
  - State encodings could be different.
  - States could also be different.
- In general a very difficult problem that requires further research.
  - In practice, if you could provide hints to relate the states of the two circuits, tools may be able to provide a proof.

FSM Equivalence Checking Examples

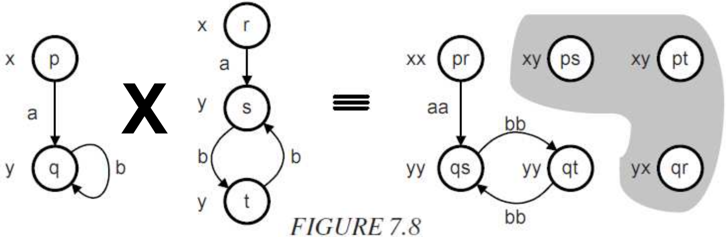


FIGURE 7.8

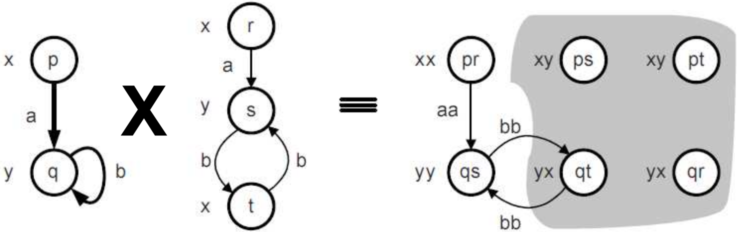


FIGURE 7.9

(Gajski et al.)

- Consider reachable states in product FSM

A Special Case of FSM Equivalence Checking

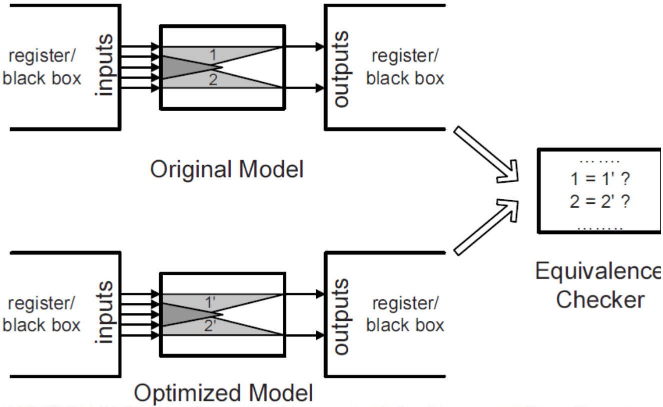


FIGURE 7.6 Logic equivalence checking by matching of cones.

(Gajski et al.)

- With the same states, state encodings, and initial states, one just need to prove the next state and the output functions are equivalent.
- That's combinational equivalence checking. Solved!

Limitations of Combinational and FSM Equivalence Checking

- What if the functional equivalence goes beyond inputs/outputs per cycle?
  - e.g. two kinds of processors with the same ISA but requiring different number of cycles to complete the same instructions?
- Although we perceive them as “equivalent”, it is difficult to define equivalence in mathematical sense.
- Equivalence checkings cannot be applied as of now.
  - We have to apply a more expensive method called model checking.

Outline

Equivalence Checking

Model Checking

Model

- Model refers to desired system properties (in mathematics)
  - Safety: bad things never happen
  - Liveness: good things eventually happen
- Examples for safety
  - Two circuits always produce the same outputs with the same inputs (i.e. equivalence checking)
  - The microwave oven will not start when the door is open.
- Examples for liveness
  - A processor eventually executes an instruction.
  - The brake is eventually applied after you hit the pedal (within a deadline)

Model Checking

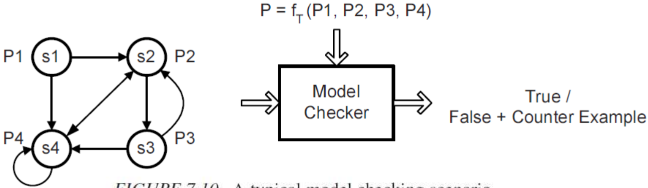


FIGURE 7.10 A typical model checking scenario.

(Gajski et al.)

- Model checking requires to represent the system and the model to be checked in a mathematically unambiguous way.
  - The system: FSM
  - The model: temporal logic, a function that maps all traces of state transitions to 0 and 1
- The system FSM could be the RTL implementation of the system but is usually its abstraction.
  - Enable one to check a RTL system with more than 10^{100} states
- Model checker will generate a counter example when the property doesn't hold, helping to identify corner cases for simulation-based verification.

Computation as a Tree

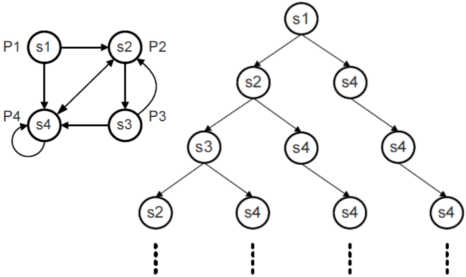
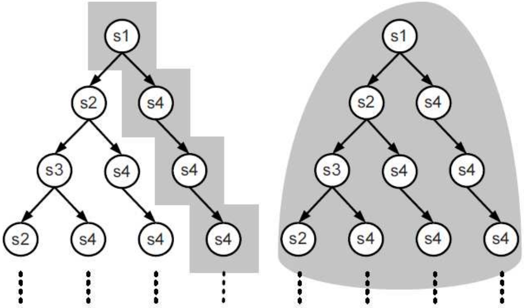


FIGURE 7.11 A computation tree derived from a state transition diagram.

(Gajski et al.)

- We can expand the FSM into a tree that captures all possible traces of state transitions to 0 and 1.
- Temporal logics are limited to certain traces on the tree.

Temporal Logics I



(a) EG p

(b) AG p

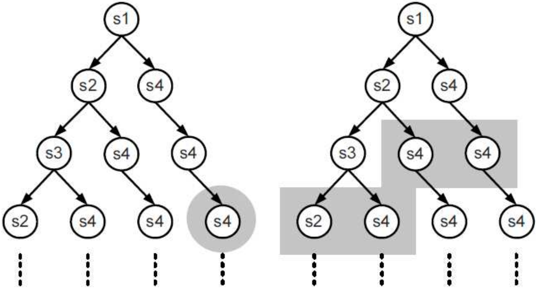
FIGURE 7.12

(Gajski et al.)

(a) p always (G) holds in one future (E): liveness

(b) p always (G) holds in all futures (A): safety

Temporal Logics II



(c) EF p FIGURE 7.12 (d) AF p

(Gajski et al.)

(c) p holds eventually (F) in one future (E): safety
  - $EF p = \neg AG \neg p$

(d) p holds eventually (F) in all futures (A): liveness
  - $AF p = \neg EG \neg p$

Example

- Consider a game that the player wins if he/sha could obtain exactly 4 gallons of water using a 5 gallon jug, a 3 gallon jug, and a water faucet.
- All jugs start empty.
- Each step the player could either
  - Empty a jug to ground.
  - Pour water from a jug to another, until one of them is empty or full.
  - Fill a jug full with the faucet.
- How to model the game and how to reason with the model?

Example (Cont.)

- Use a state machine.
  - State: (a, b), where a is the amount of water in the 5 gallon jug, and b is the amount of water in the 3 gallon jug.
  - Transition from (a, b): $(0, b)$, $(a, 0)$, $(5, b)$, $(a, 3)$, $(a + b - 3, 3)$ or $(0, a + b)$, $(5, a + b - 5)$ or $(a + b, 0)$
- EF: from $(0, 0)$, can we reach $(4, 0)$?
- $(0, 0) \to (5, 0) \to (2, 3) \to (2, 0) \to (0, 2) \to (5, 2) \to (4, 3) \to (4, 0)$
  - A harder question: is this the shortest path?

Other Formal Verification Techniques

- Theorem proving via deductive reasoning
  - Proofs are usually obtained *interactively*, i.e. designers need provide additional deduction rules for the prover if it cannot proceed further automatically.
- Bounded model checking
  - Simplify model checking by bounding the lengths of traces
- Symbolic simulation
  - Use symbols to increase coverage in simulation-based verification while utilizing equivalence checking for monitor

Summary of Verification Techniques

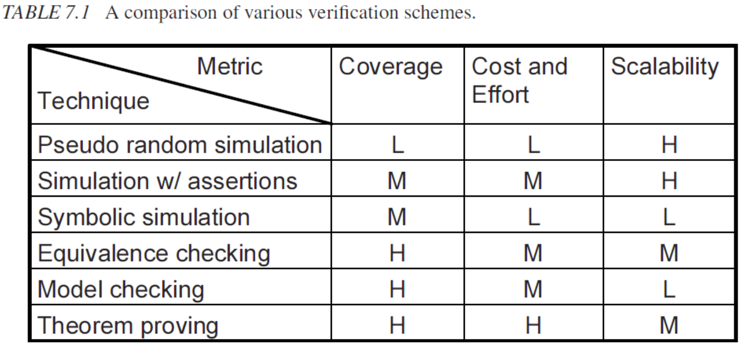


TABLE 7.1 A comparison of various verification schemes.

Metric Technique	Coverage	Cost and Effort	Scalability
Pseudo random simulation	L	L	H
Simulation w/ assertions	M	M	H
Symbolic simulation	M	L	L
Equivalence checking	H	M	M
Model checking	H	M	L
Theorem proving	H	H	M

(Gajski et al.)