ECE 587 – Hardware/Software Co-Design
Lecture 12 Verification I

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

February 24, 2025

Reading Assignment

- This lecture: 7.1
- Next lecture: 7.2, 7.3

Outline

Functional Verification

- Analysis and reasoning on a computer model of the system
  - Before manufacturing
- Establish confidence in functional correctness before the product is shipped
  - Critical for systems where safety is the first concern
  - Prevent costly recall for non-critical systems as well
- Designers need to make sure the model at each step of design does reflect the original intent.
  - Catch bugs as early as they are introduced so one can locate them effectively

Simulation Based Verification vs. Formal Verification

- Both verification methods need a golden reference as part of the specification.
  - Simulation based methods: stimuli (inputs) and monitors (expected outputs)
  - Formal methods: mathematical model of desired properties
- Simulation-based methods are effective to catch bugs at early design stages, though only as good as the stimuli.
- Formal methods can provide a proof of correctness, but require more effort at design time.
- Simulation is still predominant while formal methods are catching up.
  - More formal verification tools become available.
  - More designers are trained to use these tools.
  - e.g. most gate-level verifications are based on formal methods nowadays

Outline

Simulation Based Methods

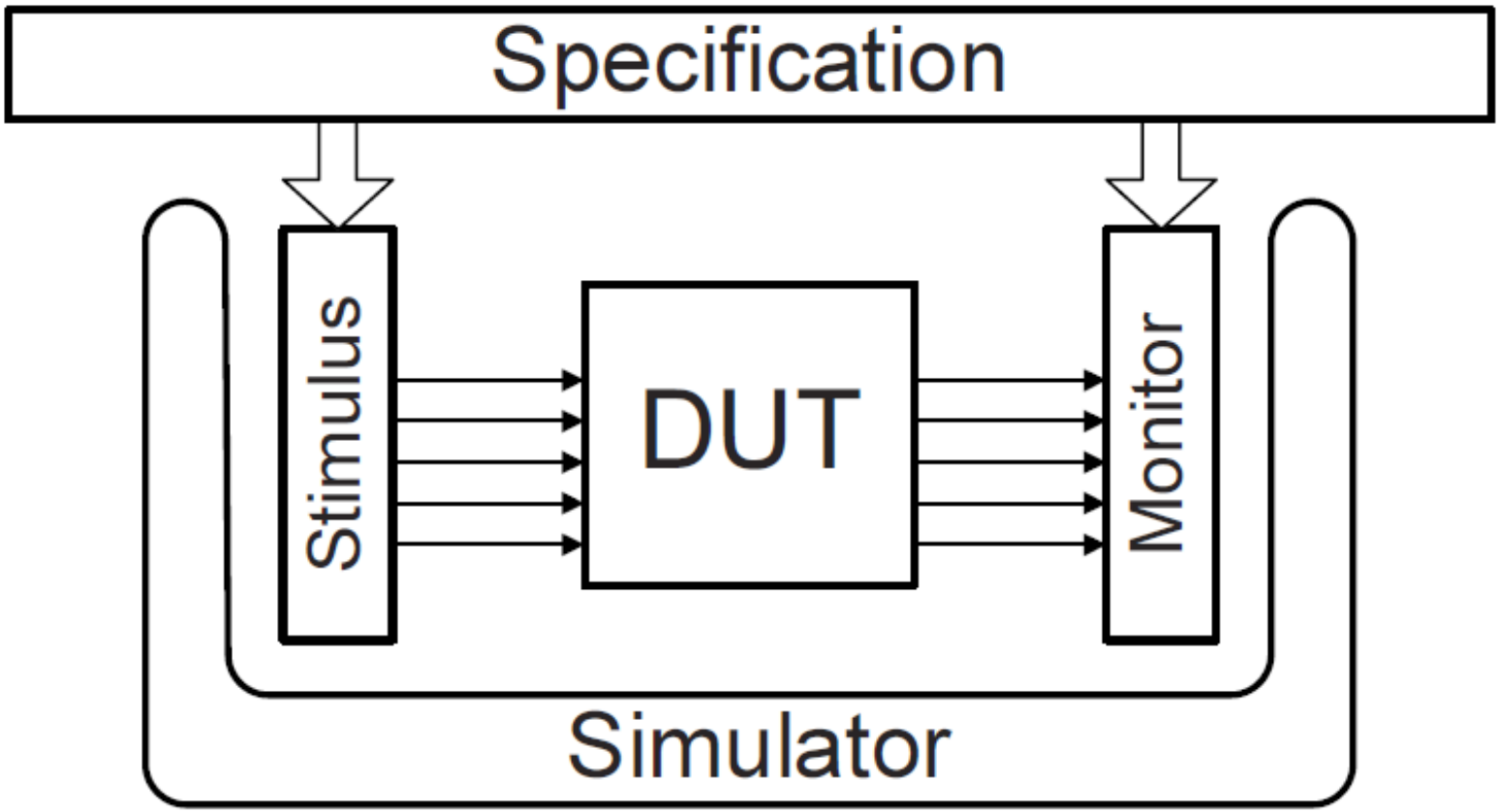

FIGURE 7.1 A typical simulation environment

(Gajski et al.)

Test-Bench

- A test-case consists of a stimulus and the corresponding monitor.
  - Monitors are usually generated from stimuli using a higher level model (golden reference) that is more likely to be correct.
- A collection of test-cases forms the test-bench.
  - To achieve maximum productivity, each test-case should uncover some bug that has not been uncovered by a previous one – waste of time to test part of DUT (device under test) that have already been tested.
- How to measure the part being tested under a test-case?

Coverage

- Could be based on the percentage of DUT that has been checked
  - e.g. in terms of # lines of source code
  - or # states in a state diagram modeling the system
- However, that's different from the entire behavior of the design.
  - A single statement may involve many variables and a test-case covers it may miss some important corners.
- All corner cases may be modeled explicitly in the state diagram for coverage.
  - But then the size of the diagram will become a big concern.

Performance Considerations

- Coverage may increase as you simulate with more test-cases.
  - However, simulation takes time.
  - Need to trade-off verification performance with quality
- Stimulus optimization: simulate less cases
  - Use coverage feedback mechanism to improve test-cases that are otherwise generated randomly
  - Not all test-cases are valid – only simulate with valid ones
- Monitor optimization: discover bugs faster
  - White box testing – also monitor internal variables
  - Use assertions instead of golden for internal variables
  - Communicate more effectively with designers via visualization
- Speed-up techniques: faster simulation
  - Faster algorithms
  - Hardware assistance, e.g. on FPGA
  - Simulate at higher abstraction levels, if certain details can be omitted, e.g. to use an instruction set simulator instead of a cycle-accurate simulator.

Coverage Feedback I

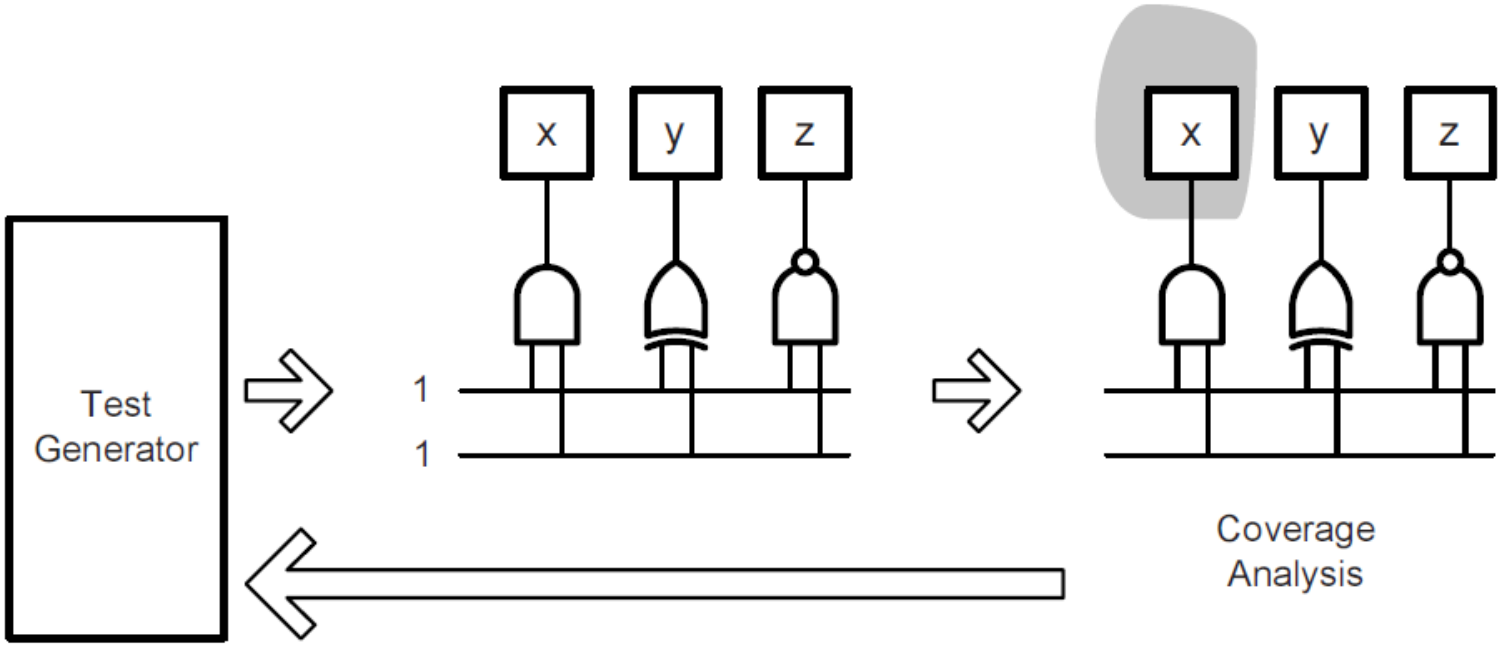

FIGURE 7.2 A test case that covers only part of the design.

(Gajski et al.)

- The logic gates could be a simplification of branch conditions.
- It could be impossible to generate a single test-case to cover all branches.

Coverage Feedback II

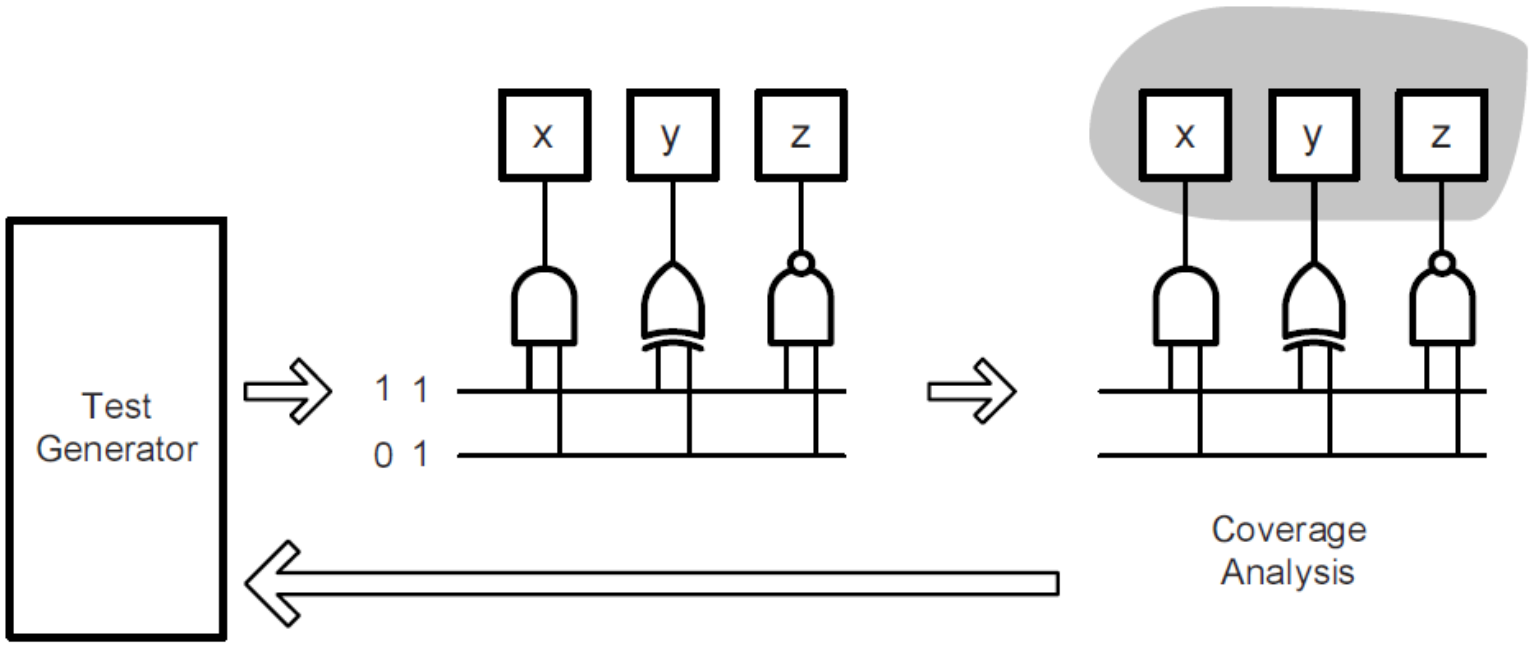

FIGURE 7.3 Coverage analysis results in a more useful test case.

(Gajski et al.)

- The additional test-cases can be found manually or automatically (by formal methods).

Hardware Emulation

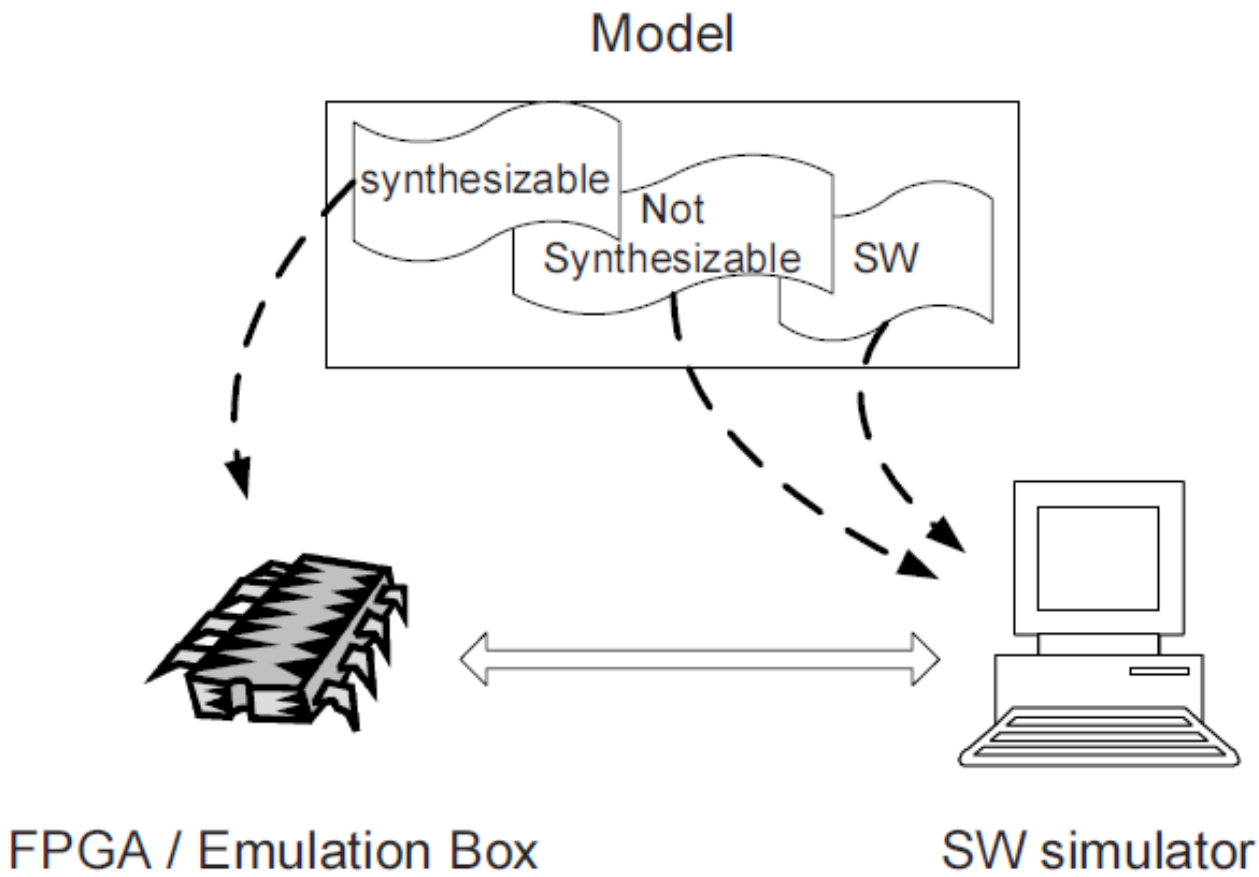

FIGURE 7.5 A typical emulation setup.

(Gajski et al.)

- Speed-up simulations that cannot be speeded-up otherwise.
  - Usually when cycle-accurate behavior has to be simulated