N26F300
VLSI SYSTEM DESIGN
(GRADUATE LEVEL)

Verification using SystemVerilog (I)

Outline

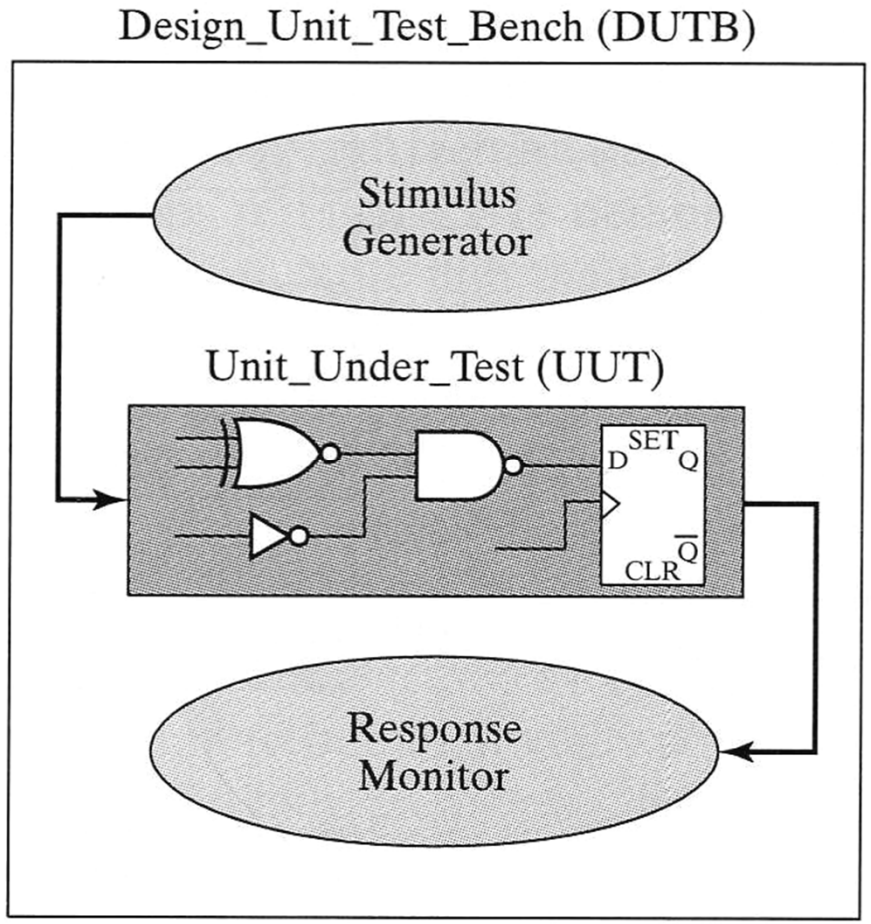
- Testbench Functionality
- Approaches for Writing Testbenches
  - Directed Test
  - Constrained-Random Test
  - Quick View on Assertions
- Connect Testbench and Design
  - Interface
  - Stimulus Timing

References: Chris Spear & Greg Tumbush, “SystemVerilog for Verification: a Guide to Learning the Testbench Language Features,“ 3rd ed., 2012, Springer.

3

Testbench Functionality

Testbench



- Generate test patterns
  - Waveforms to inputs
- Timing of applying patters
  - Delay in signals
  - Clock generation
- Start/end time for simulation
  - $initial
  - $finish

Basic Testbench Functionality

- Generate stimulus
- Apply stimulus to the DUT
- Capture the response
- Check for correctness
- Measure progress against the overall verification goals

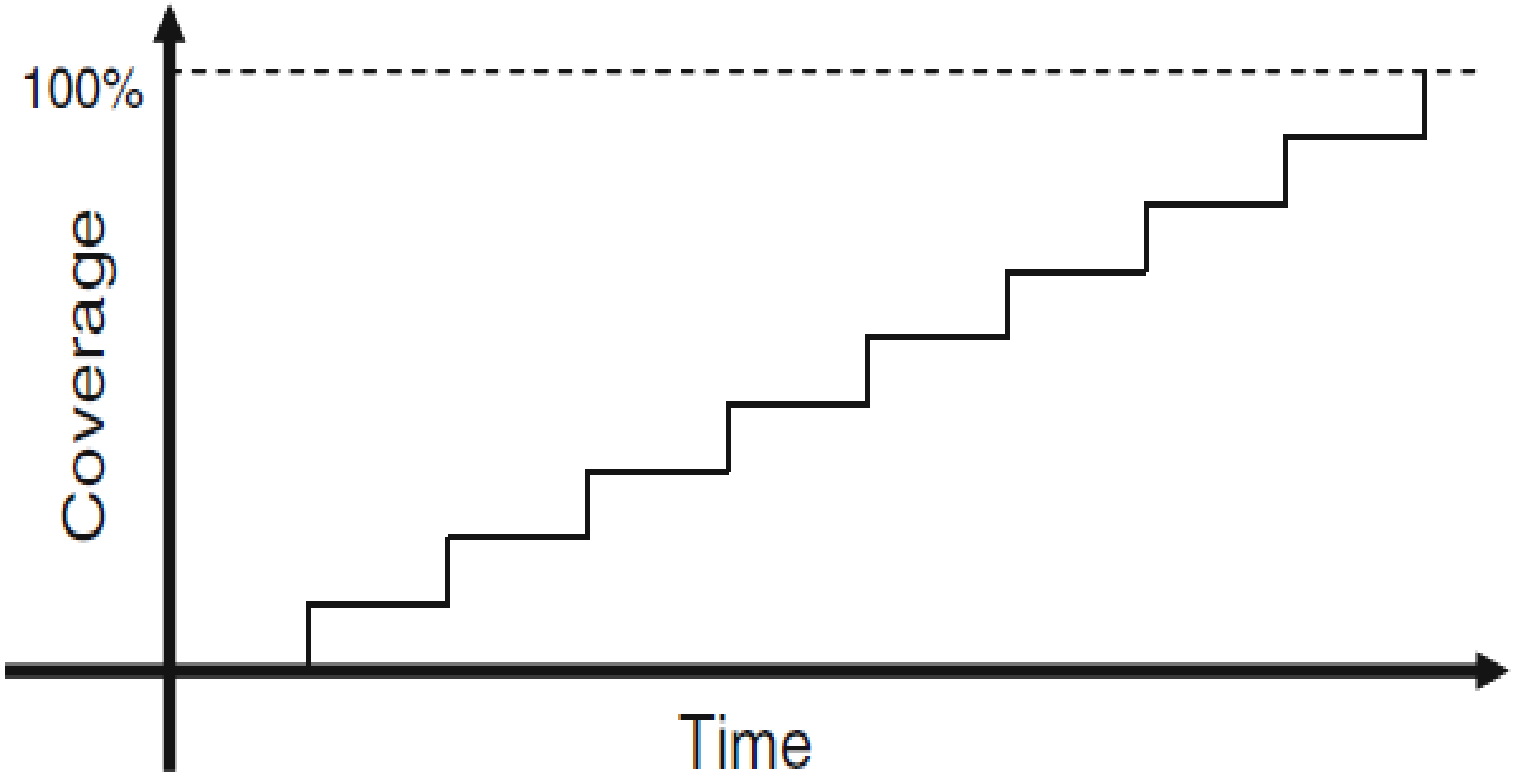
6

Approaches for Writing Testbenches

Directed Testing Approach

- Write a verification plan with a list of tests based on hardware specification
- Prepare stimulus vectors for a test and manually review the resulting log files and waveforms
- Check off the test and move to the next one
- Goods
  - Produce immediate results because little infrastructure is needed
  - Given ample time and staffing, sufficient to verification many designs

Directed Test Progress over Time



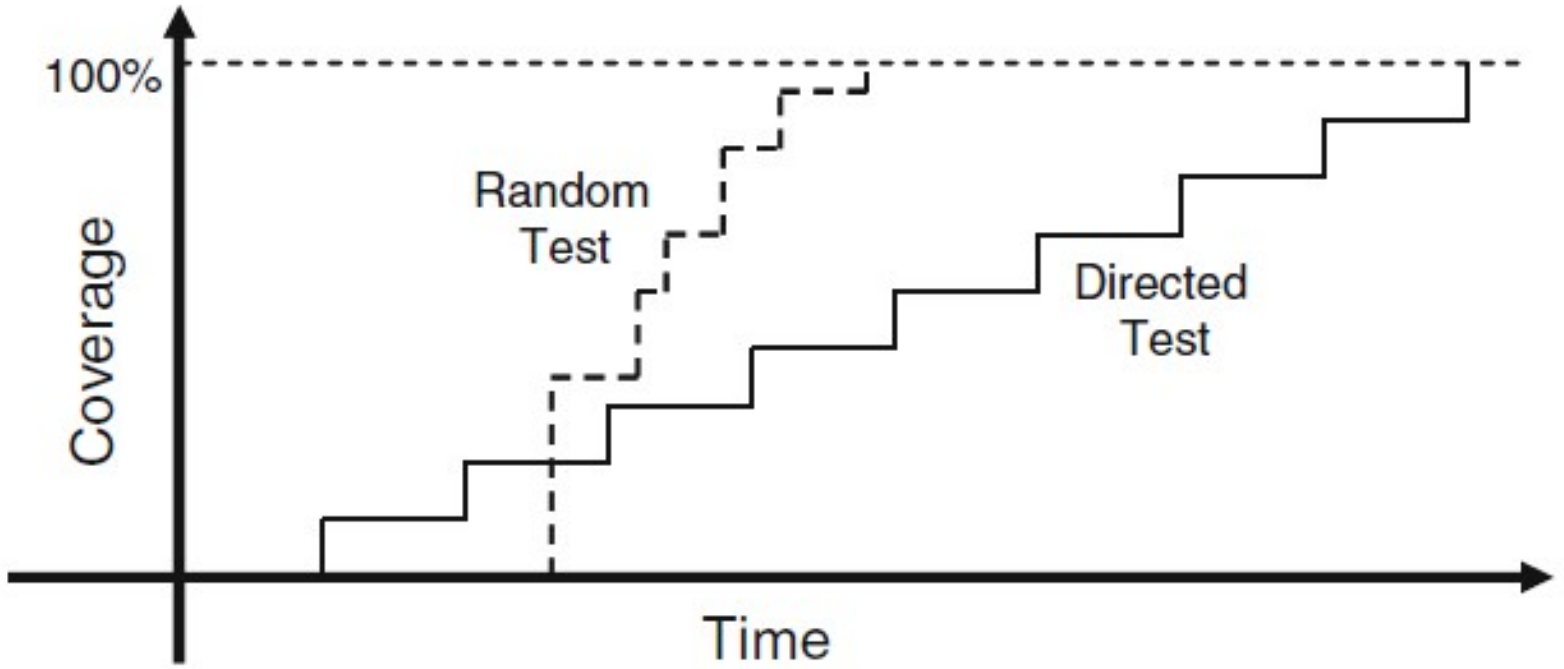
What if time and resources are not available?

- The slope may remain the same ….
- As the design complexity doubles, one needs twice as long to complete or require twice as many people to implement it
- Brute force does not work, because of complexity

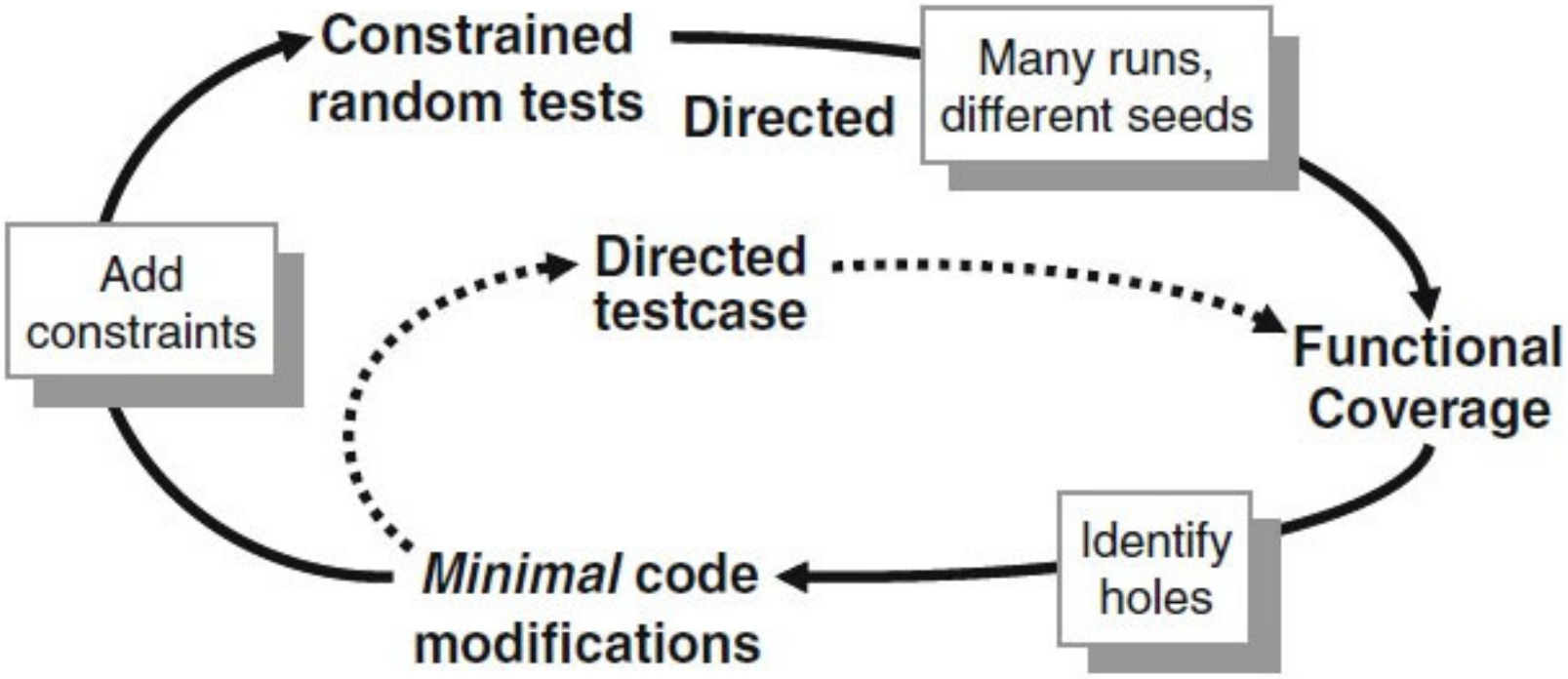
Better Methodology

- Constrained-random stimulus
- Functional coverage
- Layered testbench using transactors
- Common testbench for all tests
- Test case-specific code kept separate from testbench

Constrained-random Test vs. Directed Test



Coverage Convergence



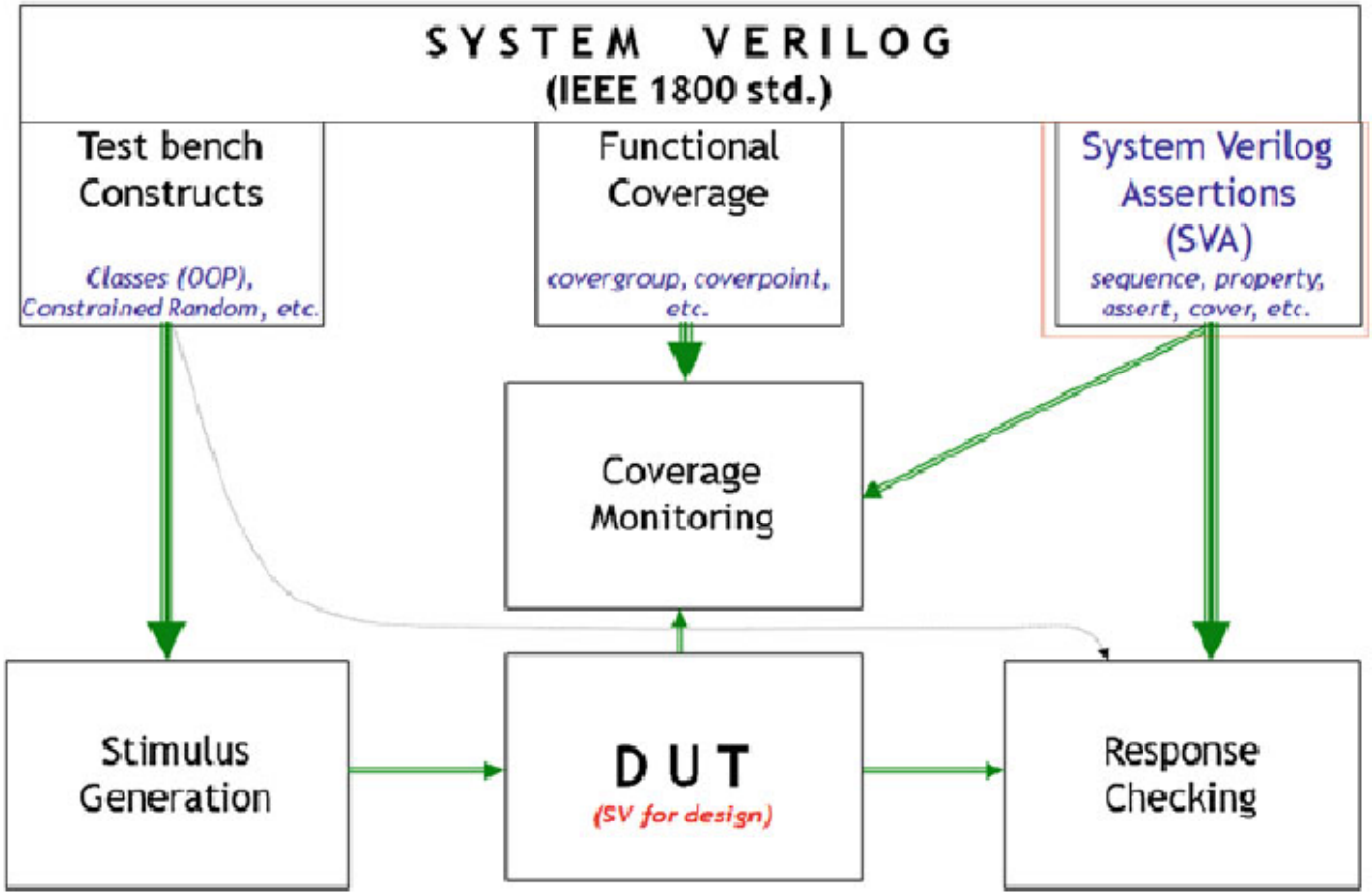
Constrained-Random Stimulus

- Totally random?
  - Not really
  - Need to follow certain format
    - Address is 32-bits
    - Opcode is ADD, SUB, LOAD or STORE
    - Cache line is 128 bits, i.e., 4 x 64 bits
- What to randomize
  - Device configuration, Environment configuration
  - Input data, Protocol exceptions, violations, or delays

Functional Coverage

- Which areas have been visited more often than others?
- Are there any unreachable states?
- What has been verified in the verification plan.
- Typical process
  - Add monitors to the stimulus going to the device, and its reaction and response
  - Run several simulations, each with a different seed
  - Merge the simulation results into a report
  - Analyze and determine how to create new stimulus

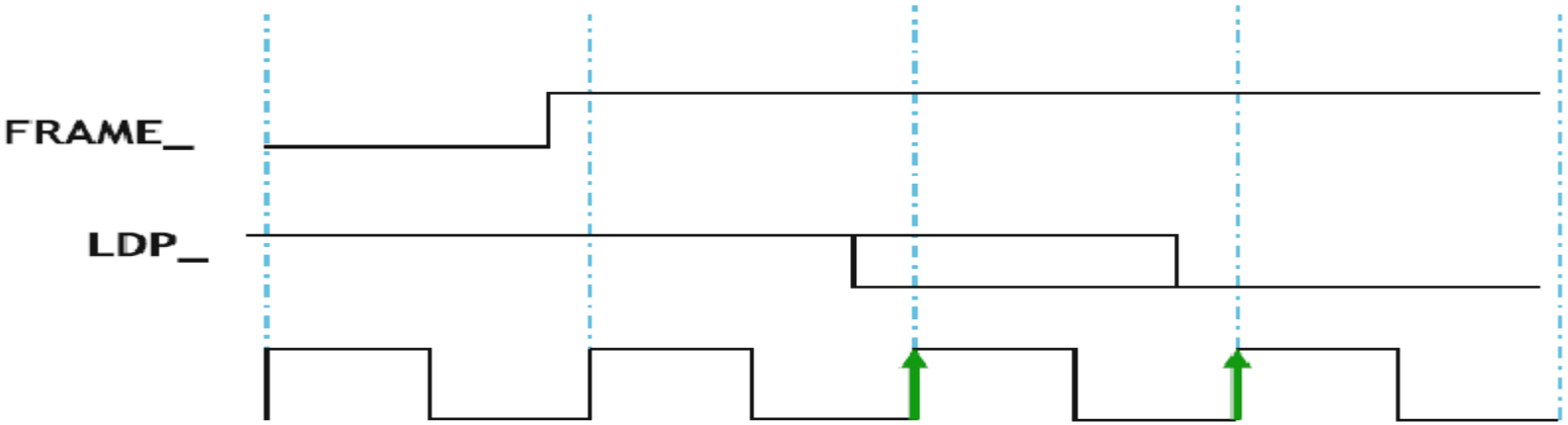
Assertions and Functional Coverage



Source: A. Metha, "SystermVerilog Assertions and Functional Coverage"

Quick View on Assertion (AST)

- An assertion: a check against the specification of your design

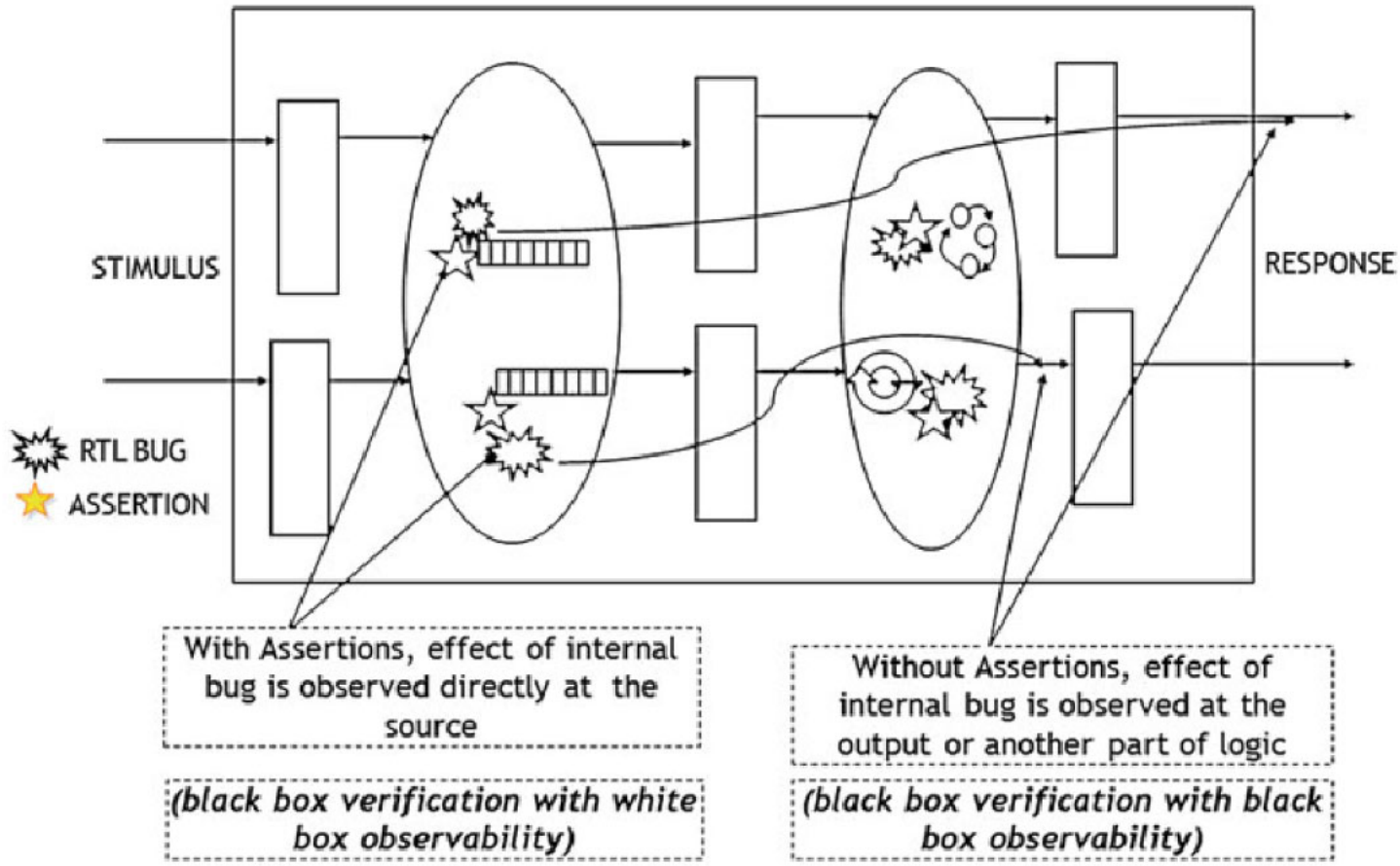


When FRAME_ is de-asserted (i.e. goes High), Last Data Phase (LDP_) must be asserted (i.e. goes Low) within the next 2 clocks.

Capability of temporal domain scenarios!!

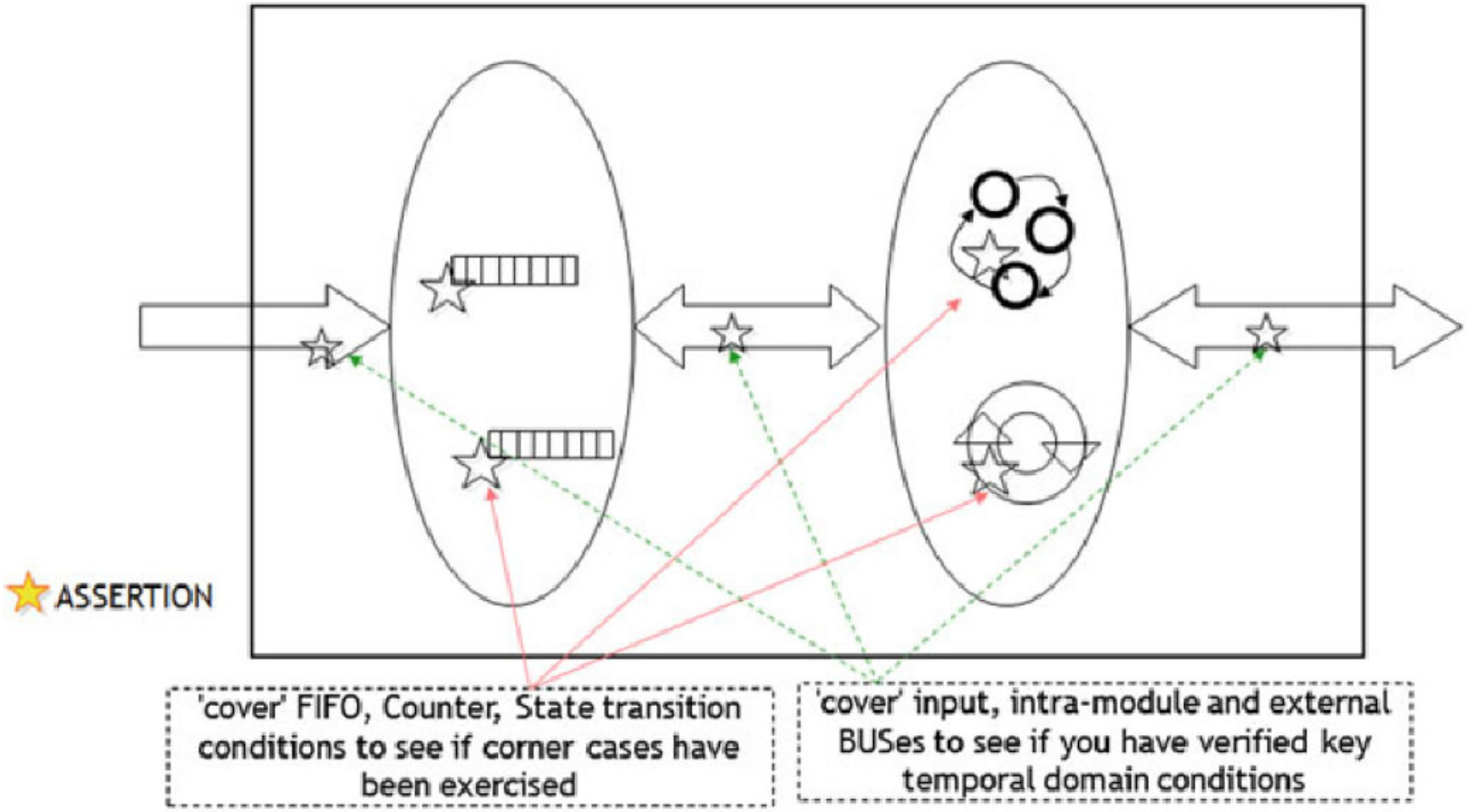
Source: A. Metha, "SystermVerilog Assertions and Functional Coverage"

ASTs Improve Observability



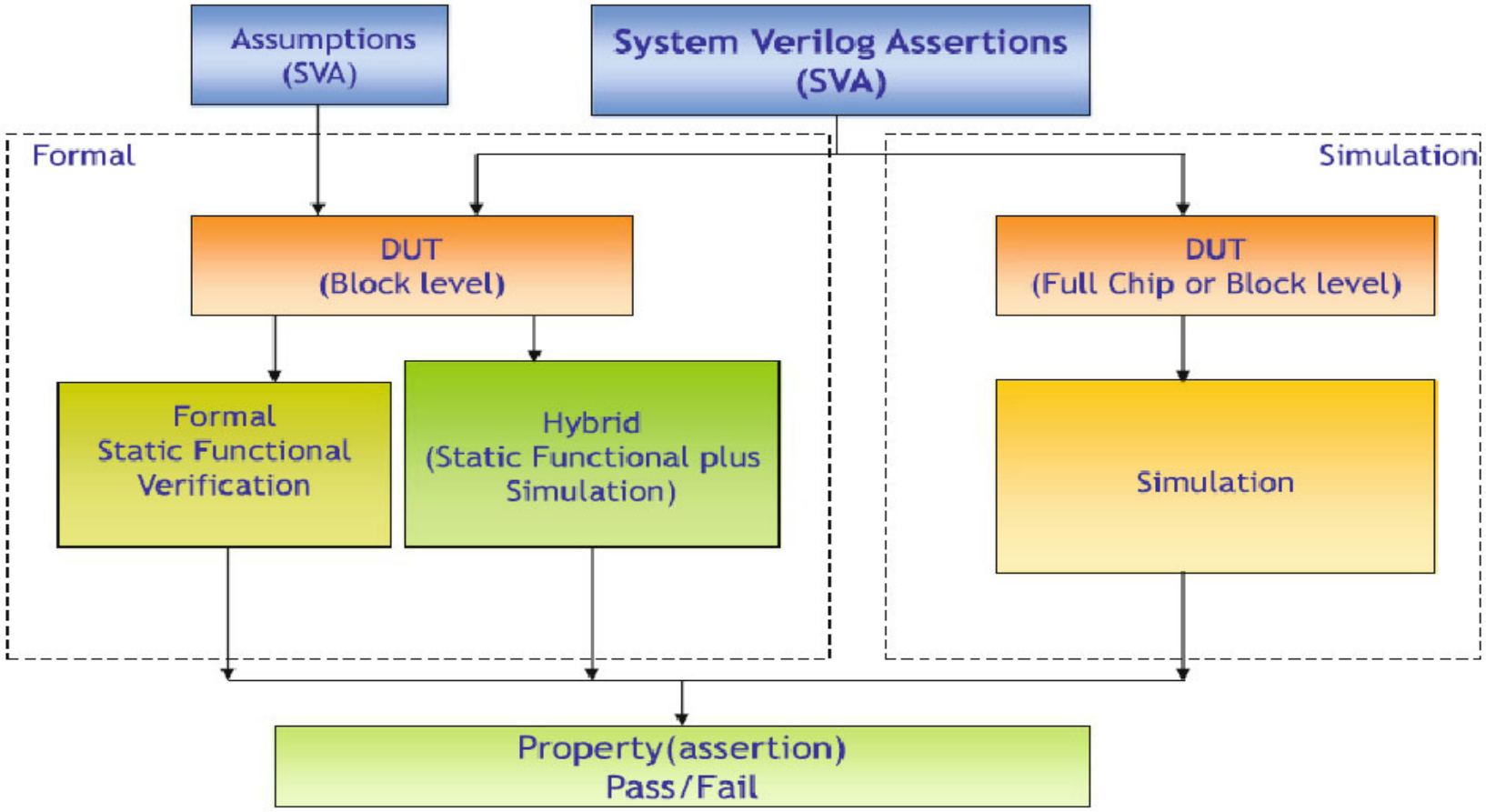
Source: A. Metha, "SystermVerilog Assertions and Functional Coverage"

Temporal Domain Functional Coverage



Source: A. Metha, "SystermVerilog Assertions and Functional Coverage"

Assertions and Assumptions



Source: A. Metha, "SystermVerilog Assertions and Functional Coverage"

What types of ASTs are added? (1/2)

- RTL ASTs (design intent)
  - Intra Module: illegal state transitions; deadlocks; livelocks; FIFOs, onehot, etc.
- Module interface ASTs (design interface intent)
  - Inter-module protocol verification; illegal combinations (ack cannot be '1' if req is '0'); steady state requirements (when slave asserts write_queue_full, master cannot assert write_req)
- Chip functionality ASTs (chip/SoC functional intent)
  - A PCI transaction that results in Target Retry will indeed end up in Retry Queue

Source: A. Metha, "SystermVerilog Assertions and Functional Coverage"

What types of ASTs are added? (2/2)

- Chip interface ASTs (chip interface intent)
  - Commercially available standard bus assertion VIPs, such as PCIXm AXI, etc.
  - Every design assumption on IO functionality is an assertion
- Performance Implication ASTs (performance intent)
  - Cache latency for read; packet processing latency; etc. to catch performance issues before it's too late. For example, if the 'Read Cache Latency' is greater than 2 clocks, fire the assertion. Easy to write and very useful!!

Source: A. Metha, "SystermVerilog Assertions and Functional Coverage"