Introduction to Functional Verification
Overview

- Verification issues
- Verification technologies
- Verification approaches
- Universal Verification Methodology
- Conclusion
Functional Verification issues

- Hardware Designs get more and more complex
- Hand-written stimulus (directed test) is difficult to write and maintain
- Corner cases are difficult to catch
- Visual inspection of waveforms in order to trace a bug is a tedious task
- “The amount of time spent on verification now exceeds the amount of time spent on design, comprising up to 70 percent of the total development effort.”

Goal: Find bugs early and fast !!
Verification Requirements

- Automation
- Progress measurement
- Reusability
- Easy to write and to maintain
- Find all bugs
Verification Technologies

- Formal Verification
  - Equivalence Checking
  - Formal Property (Assertion) Checking
- Simulation Based Verification
Simulation Based Verification

- Monitor: samples interface activity
- Scoreboard: checks DUT behavior
Formal Property Checking

- Proves the correct behavior for all operation states
- Design under test (DUT) gets translated into boolean expressions
- No input stimulus is needed – the stimulus is created by the tool
- Properties describe the behavior of the design
  - Assertions for DUT behavior
  - Constraints for environment behavior
Equivalence Checking

- Proves the exactly same behavior of different design representations
- e.g. RTL vs. synthesis net list
- Several tools in the design process change the design
  - Design for Test (DFT) inserts additional logic
  - Logic optimizations
  - Engineering Change Orders (ECOs)
Verification Approaches

- Transaction Based Verification
- Coverage Driven Verification
- Constrained Random Testing
- Assertion Based Verification
Transaction Based Verification

- Abstraction from low-level signals
- Contains abstract tasks that hide the implementation from the engineer
- Enhances reusability
- “task” in Verilog
Coverage Driven Verification

- Coverage metrics are used to ascertain whether a test verified a given feature
- Uncovers holes in the verification process
- Adjusts stimulus to check cases that have not yet been covered
- Defines a metric to measure verification progress
- Functional coverage, code coverage, assertion coverage, test coverage
Constrained Random Testing

- Focuses on input stimulus generation
- Randomized stimulus is generated automatically
- Stimulus is filtered by constraints in order to achieve only valid test patterns
Assertion Based Verification

- Assertions check the state of a DUT
- Run concurrently and ensure correct functional behavior
- Increase observability
- Can be used for formal verification

```vhdl
property legal_data_valid;
 @(posedge clk) disable iff(!res_n)
   (data_valid && $rose(sop)) -> ##[1:32] ##1 eop;
endproperty : legal_data_valid

data_valid : assert property(legal_data_valid);
```
Verification Plan

- Defines the functionality of the DUT
  - Checks
  - Coverage
- Does NOT define how the DUT has to be verified
- Coverage results are mapped to the verification plan for analysis
Verification Plan

5.1.1.1 - Network Port to Host Port Interface

- Grade: 88% (OK)
- Checks: (no checks)
- Name: Packet Length
- C: 100%

Details:
- Name: Network Port to Host Port Interface
- This vplan holds information about the coverage that is collected by the verification environment for the EXTOLL (2) XPath.
- Verification Scope: MODULE_NETWORK_PORT
- Subsections: 0
Universal Verification Methodology (UVM)

- A standard for building simulation based verification environments
- Class library based on SystemVerilog
- Key features
  - Data design and stimulus generation
  - Building and running a verification environment
  - Coverage modeling and checking
- Focuses on re-usability
- Maintained by Accellera
Universal Verification Component (UVC)

- Combines all components for a single interface into a reusable package

- Sequencer: creates transactions
- Driver: transforms transactions into stimulus
- Monitor: samples stimulus – turns into transactions
- Agent: environment for sequencer, driver and monitor
Conclusion

- Traditional design simulation has many drawbacks
- New verification methodologies improve the task of finding bugs
- Advanced verification methods have a high learning curve
- But: Once established, functional verification increases the productivity