1. Verification Plan

Testbench Requirements

a) The model types

b) Abstraction levels

c) Model sources

d) Testbench elements (checkers, stimulus, and so on) need to be considered

- For formal verification, define design properties and constraints

1. Verification Plan

Verification Metrics

- Two classes of metrics should be addressed in the verification plan
  - Capacity metrics
    - Identifies tool capacity assumptions (run times, memory size, disk size, and so on) and verifies that the assumptions made holds true during the execution of that plan
  - Quality metrics
    - Establishes when a verification task is complete
    - Quality metrics include functional coverage and code coverage

1. Verification Plan

- **Regression Testing**
  a. The strategy for regression testing
  b. The test plan details when the regression test are to be run (overnight, continuously, triggered by change levels, and so on) AND
  c. Specifies the resources needed for the regression testing
- **Issue Tracking and Management**
  – Which tracking system to use to manage bugs and errors found in the design

1.Verification Plan

- **Resource Plan**
  - The resources required to execute the verification plan, such as human resources, machine resources, and software tool resources.
- **Project Schedule**
  - The tasks that must be performed to execute the verification plan as well as key benchmarks and completion dates.
  - The plan should show the interdependencies between tasks, resource allocation, and task durations.

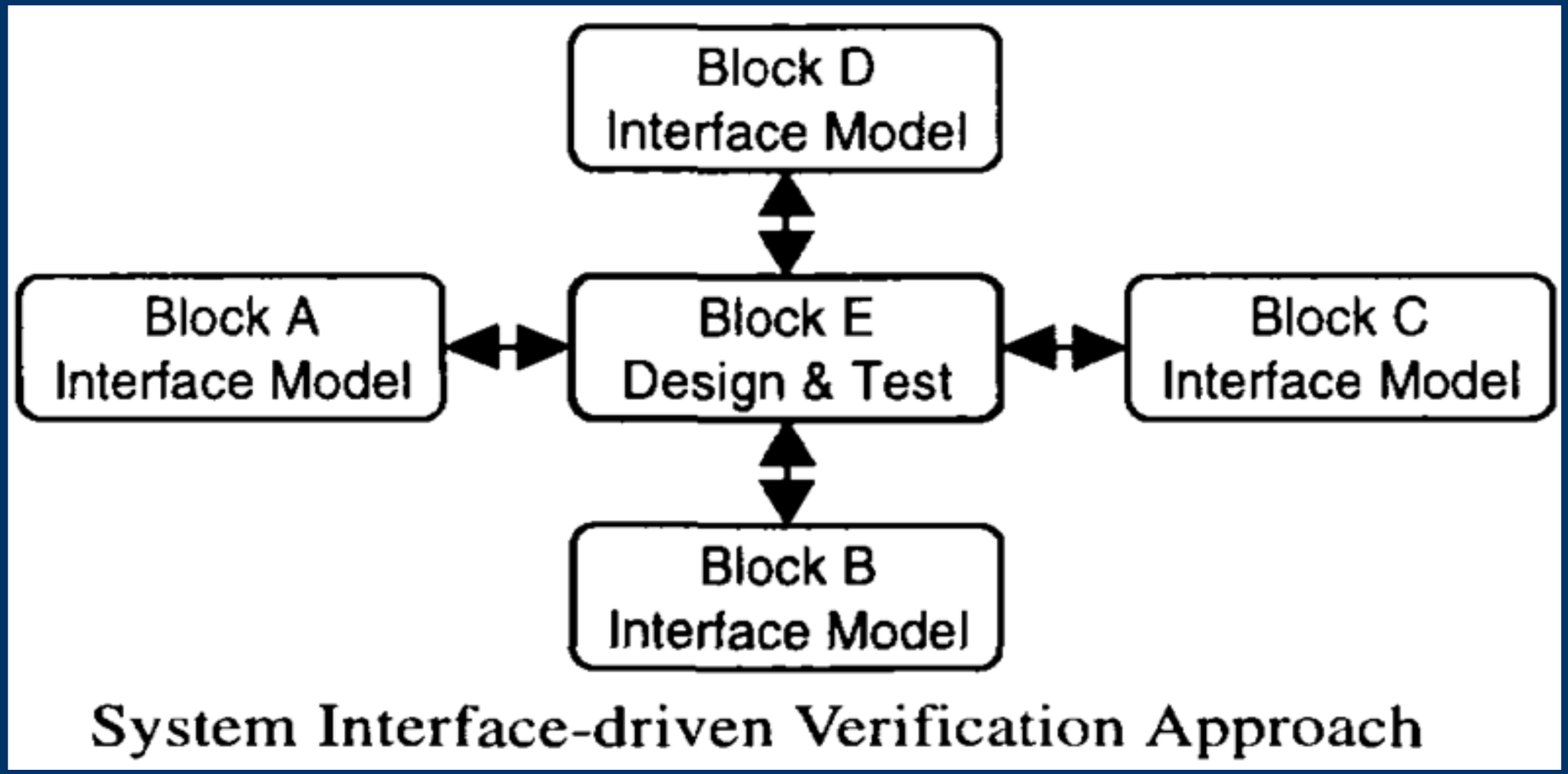
1.Verification Plan

Platform Based Verification

- Confined to particular platform like xilinx, altera.
- We can verify mutually interactive multiple IP's with very low cost.
- It is a standard C-based test methodology

1.Verification Plan

- System interface driven verification



System Interface-driven Verification Approach

2. Building Testbench

Building Testbench

- In this phase, the verification environment is developed.
- Each component can be developed one by one.
- It is preferred to write down the coverage module first as it gives some idea of the verification.

3. Writing Tests

Writing Tests

- After the Testbench is built and integrate to DUT
- Initially in CDV, the test are ran randomly till some 70% of coverage is reached (or) if there is no improvement
- After analyzing the coverage reports, new tests are written to cover the holes.
- Randomization is directed to cover the holes
- Corner cases have to be written in directed verification fashion

4. Integrating Code Coverage

Integrating Code Coverage

- Once after achieving certain level of functional coverage, integrate the code coverage.
- The code coverage tools have option to switch it on.

5. Analyze Code Coverage

Analyze Coverage

- Finally analyse both functional coverage and code coverage reports and take necessary steps to achieve coverage goals.