


Verification Plan

Reference:

- Chapter – 1-3, System on a chip verification methodology and techniques by Prakash Rashinkar
 - Chapter – 3: Writing Test Benches using SystemVerilog
-
-

Phases of Verification

1. Verification Plan
 2. Building Testbench
 3. Writing Tests
 4. Integrating Code Coverage
 5. Analyze code coverage
- 
- Two horizontal orange lines of different lengths are positioned at the bottom of the slide. The top line is longer and starts further to the left, while the bottom line is shorter and starts further to the right.

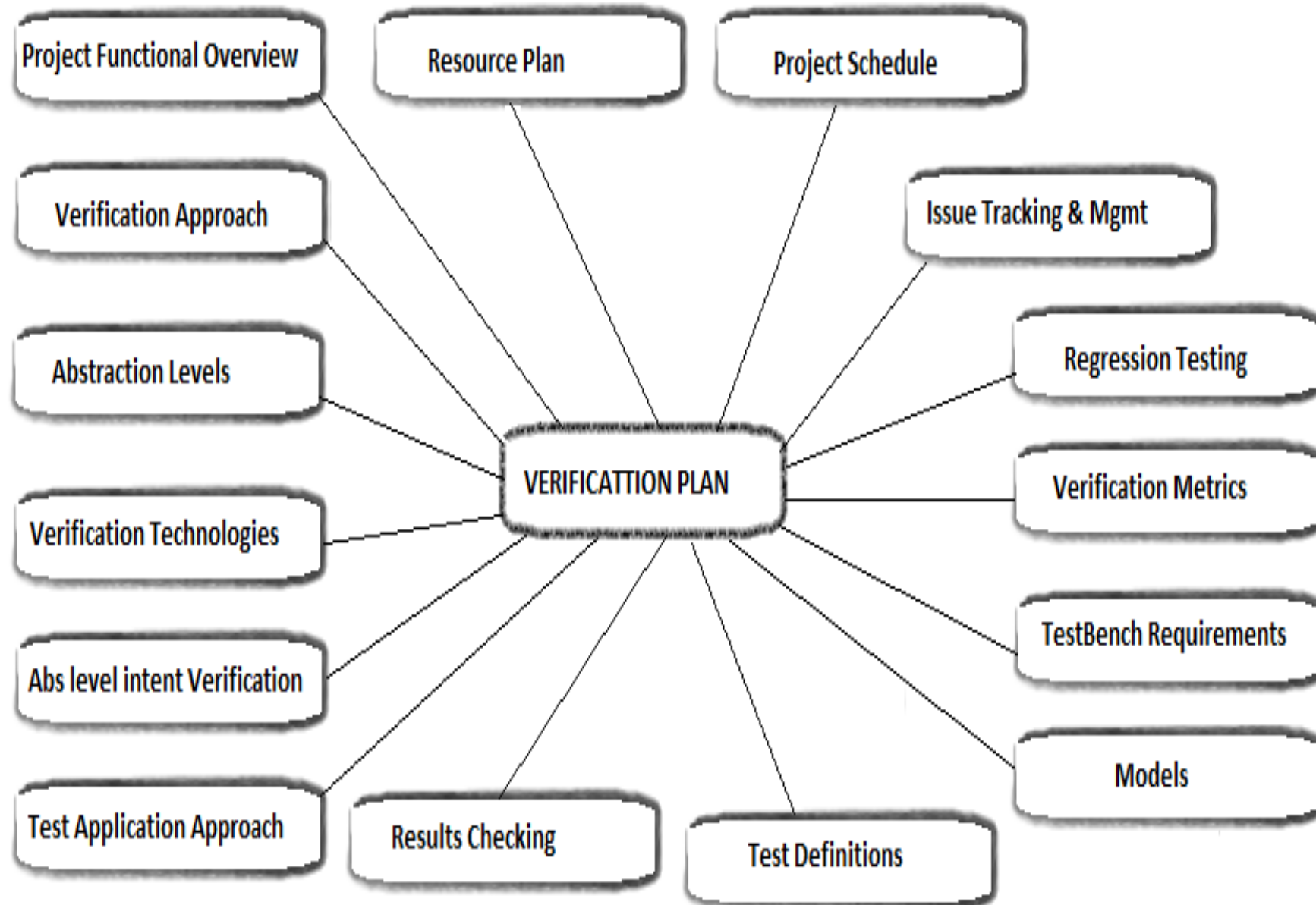
1. Verification Plan

- Develop and document a verification plan to serve as a map for all of the verification activities to be performed.
 - Must be prepared to modify the plan if unanticipated problems and limitations arise
 - If a plan is modified, the changes must still meet the overall goals of the plan
 - A test plan can be in many forms, such as a spreadsheet, a document or a simple text file
 - Test plan also contains the descriptions of testbench architecture and description of each component and its functionality
-
-

1. Verification Plan

- Verification plan should address the following areas
 - Project Functional Overview: Summarizes the design functionality, defines all external interfaces, and lists the major blocks to be used.
 - Verification Approach: Which verification approach fits the design requirements, for example
 - Top-down design and verification
 - Bottom-up verification
 - Platform-based verification
 - System interface-driven verification
-
-

1. Verification Plan



1. Verification Plan

Abstraction level for intent verification

- The abstraction level that the design functionality is verified against.
 - The abstraction levels to be verified include behavioural, functional, gate level, and switch level.
- Test Application Approach
- The approach for applying functional tests to the design.
 1. Either pre-loading tests into on-chip memory and executing the test on-chip processor
 2. Applying the test throughout the external interfaces to the device

1. *Verification Plan*

Verification Technologies

- Dynamic verification tools
 - Exercises a model of the design with specific stimulus, checking that the design functions as expected
 - Static verification tools
 - Confirms that the design exhibits specific behaviors by mathematical analysis of the design. Exhaustively checks the functionality for the state space identified
 - Co-verification
 - Rapid prototype
 - Emulation system
 - Static timing verification tools
 - Physical verification and analysis tools
 - Device test tools
-
-

1. *Verification Plan*

Test Definitions

- Defines the tests that are to be performed and the model abstraction levels to which the tests are to be applied.
 - In many instances, the same test will be applied to several model levels.
 - For each test, the verification technology or tool and the associated metrics should be included.
 - The metric indicates when the test is complete.
 - Metrics can be defined for combinations of tests.
 - For example, 100 percent statement coverage will be achieved when executing all of the simulation tests.
-
-

1. *Verification Plan*

Models

Models include behavioral, functional, RTL, logic level, gate level, switch level, and circuit level

- Model sources
 - Identify the source for all of the models required in the verification plan and when they will be available.
 - Existing models
 - Part of a standard library, be provided as a supporting model for a core supplied (either directly by the core provider or by a third-party supplier)
 - Model that was created for a previous design and is available for reuse.
 - A plan for acquiring the models and making them available to the verification team should be developed.
-
-

1. *Verification Plan*

Models

- Derived models
 - Derived models are created as part of the design flow.
 - For example, if an RTL model of an element is available, a gate-level model can be derived through synthesis.
 - The project plan must show that all derived models will be created before they are required for verification and that the development plan has explicit dependencies showing this requirement.
- Authored models
 - Models to be developed as part of the overall development plan.
 - A plan for developing the models and making them available to the verification team should be determined.

1. *Verification Plan*

Testbench Elements

- Checkers
 - Includes protocol, expected results, and golden model checkers.
 - Transaction verification modules
 - A collection of tasks, each of which executes a particular kind of transaction.
 - The module connects to the DUT at a design interface.
 - Because most designs have multiple interfaces, they also have multiple transaction verification modules to drive stimulus and check results.
-
-

1. Verification Plan

Testbench Elements

- Stimulus
 - A set of test vectors based on the design specification.
 - All boundary conditions are covered.
 - The stimulus is created to check whether the model covers exception cases in addition to the regular functional cases.
 - Test vectors are broken down into smaller sets of vectors, each testing a feature or a set of features of the model. These test vectors are built incrementally.
 - Various data patterns, for example, all ones and zeroes 0xff, 0x00, walking ones and zeroes 0xaa, 0x55
 - Deterministic boundary test vectors, like FIFO full and empty tests
 - Test vectors of asynchronous interactions, including clock/data margining
 - Bus conflicts, for example, bus arbitration tests in full SOC tests
 - Implicit test vectors that might not be mentioned in the data sheet, for example, action taken by a master on a bus retract, system boot, system multitasking and exception handling, and randomly generated full-system tests
-
-