DAY-82
#100DAYSRTL

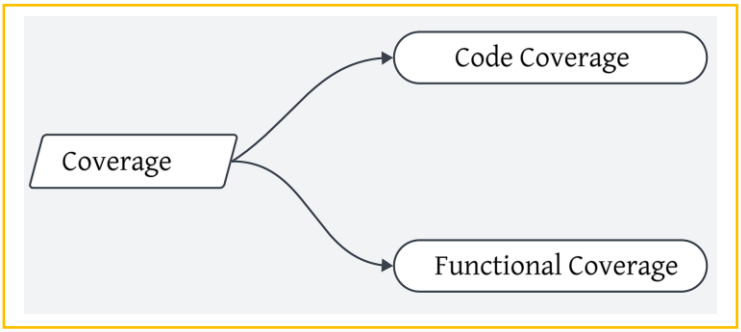
"System Verilog:-Coverage Intro"

"Introduction":-

We can generate an infinite number of random scenarios for the design under test, but we need to stop our verification at some point in time and consider the design to be verified. Coverage gives us that metric which can be used to say verification is complete. This article will be more focused on the concepts of coverage and thus will be the same for HDLs like System Verilog, VHDL, etc.

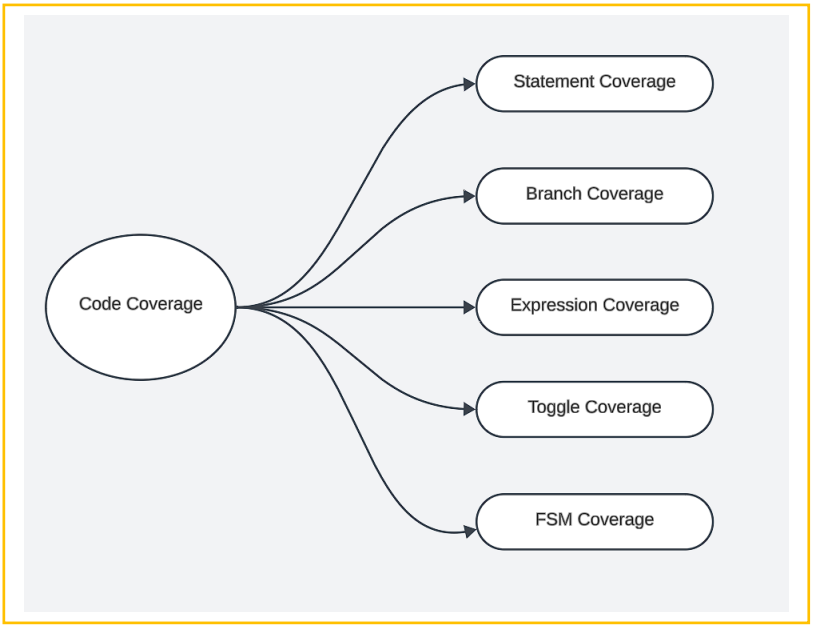
"Coverage in verification":-

- Coverage is a metric that measures how well the design under test (DUT) has been verified by the testbench.
- Coverage helps us identify the gaps in the verification process and ensure that the DUT meets the specifications and requirements
- Coverage is important because it provides feedback on the quality and completeness of the verification process.
- Without coverage, it is difficult to know if the testbench has covered all the features and scenarios of the DUT, or if there are any missing or redundant tests



“Code Coverage”:-

- Code coverage is a measure of how much of the code in the DUT has been executed by the testbench.
- It also helps us detect any dead or unreachable code in DUT. This helps optimize the design to remove any redundant code.
- Code coverage can also help us optimize the verification process by reducing the number of tests or increasing the efficiency of the tests.
- With code coverage we can know if some codes have been executed or not by our stimulus.
- If there are multiple stimuli executing the same code, we can remove some redundant tests, similarly, we can add new stimuli if some code is not getting executed.



- **Statement Coverage:-** The percentage of executable statements in the DUT that have been executed at least once by the testbench. This helps us understand if there is some unreachable code or if there is some redundant code in the design.
- **Branch Coverage:-** The percentage of branches (if-else, case, etc.) in the DUT that have been taken by the testbench. Basically, it covers whether all the branches in the if-else

construct are getting executed or not. Branch coverage implies statement coverage, but not vice versa.

- **Expression Coverage:-** The percentage of expressions (logical, arithmetic, etc.) in the DUT that have been evaluated to all possible values by the testbench. This basically covers whether the operands in an expression have taken all possible values or not.
- **Toggle Coverage:-** The percentage of bits or signals in the DUT that have changed their values from 0 to 1 or 1 to 0 by the testbench. Toggle coverage generally slows down the simulation time for a complex design and thus as a trade-off, it can sometimes be disabled.
- **FSM Coverage:-** The percentage of states and transitions in the FSMs in the DUT that have been visited by the testbench.

Note:-Code coverage can be collected using tools such as simulators or code analyzers. Thus, code coverage does not require any extra coding as needed in functional coverage. The tool automatically generates a code coverage report if code coverage commands are used in the simulation.

"Functional Coverage":-

- Functional coverage is a measure of how well the functionality of the DUT has been verified by the testbench. Functional coverage is based on the specifications and requirements of the DUT, not on its implementation.
- In System Verilog, functional coverage can be defined by using cover-points and bins, which specify what values or ranges of values of certain signals or variables are of interest for verification.

Note:- Functional coverage cannot be automatically generated by a tool. All the coverage related codes are added as part of the verification plan.