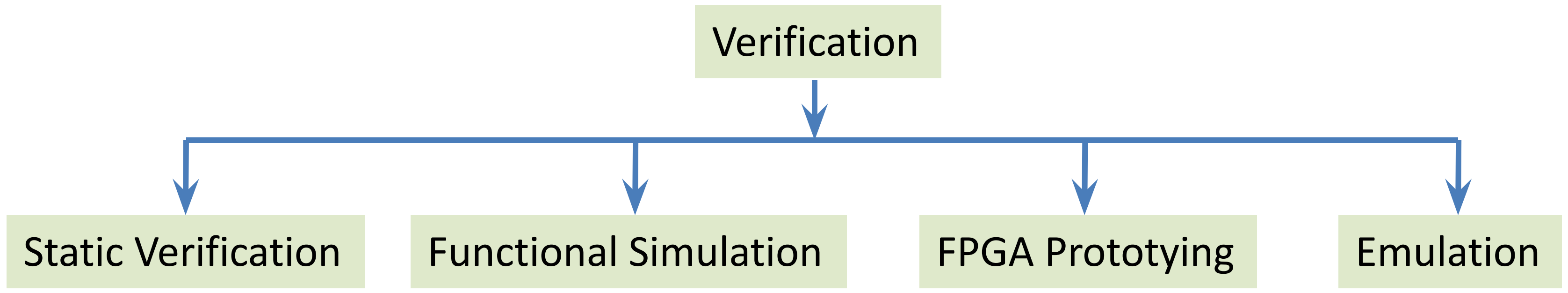
Verification techniques

- Verification is the process of demonstrating the functional correctness of a design with respect to the design specifications. Verification does not confirm the correctness of the design specification and assumes that the design specification is correct. It is one of the most challenging steps of the IC design cycle, and is the main reason for IC re-spin.

- **Purpose**
  - Functional correctness of individual IPs
  - Internal module communication
  - External module communication
  - End to end functional paths
  - Pad connectivity
  - Clock and reset circuits
  - Power up and down sequence
  - Complete integration of all IPs

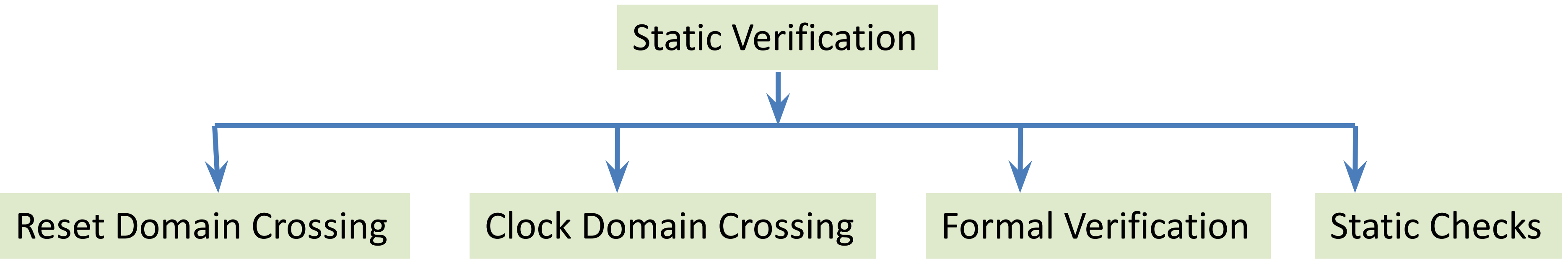
Verification techniques



Static verification

- Static verification is the process of verifying the design against some predefined rules without executing the design. It allows you to verify your design at an early stage, without any stimulus or setup, and hence is performed early in the IC design cycle, that is, as soon as the RTL code is available. It does not perform any timing checks. The sooner a bug is found, the easier it is to fix it.
- **Purpose**
  - The purpose of static verification is to reduce the verification effort at the RTL level.

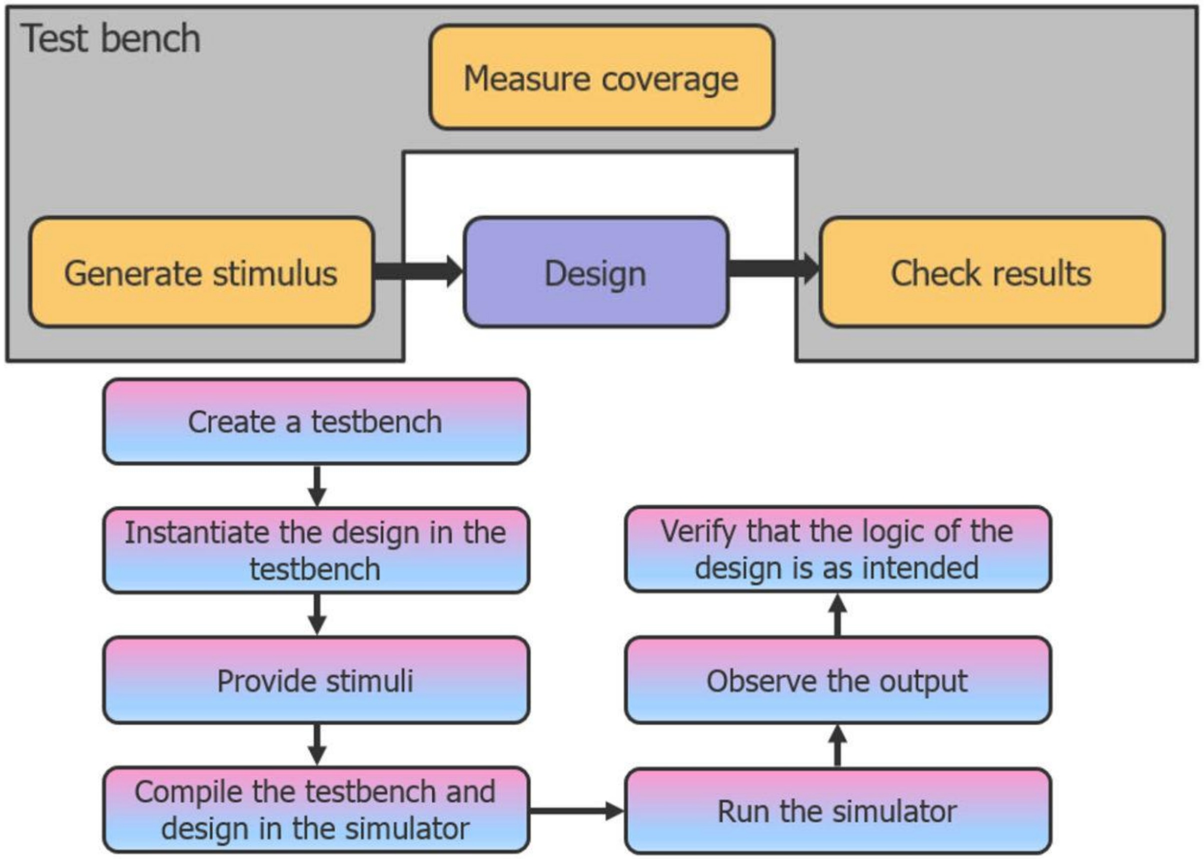
Static verification techniques



- **Advantages**
  - Detects bug early in the design cycle
  - Less time consuming
  - Reliable
  - Faster
  - Exhaustive
  - Large designs are easily handled
- Mentor, A Siemens Business, offers a broad spectrum of static and formal solutions and applications, including Questa® Formal Verification, Questa Clock-Domain Crossing (CDC), Questa Reset Domain Crossing (RDC), and the Questa Formal Verification Apps. These formal-based technologies complement simulation in a number of key areas.
- Synopsys: VC Formal, SpyGlass
- Cadence: LEC, Xcelium

Functional Simulation

- Functional simulation is the process of verifying the functional behavior of a design by simulating it in software. It does not consider the timing delays of the internal logic or interconnects, and is not helpful in software development.
- **Purpose**
  - The purpose of simulation is to verify the individual IPs or the individual blocks of the IC. System-level verification is not possible with functional simulation.

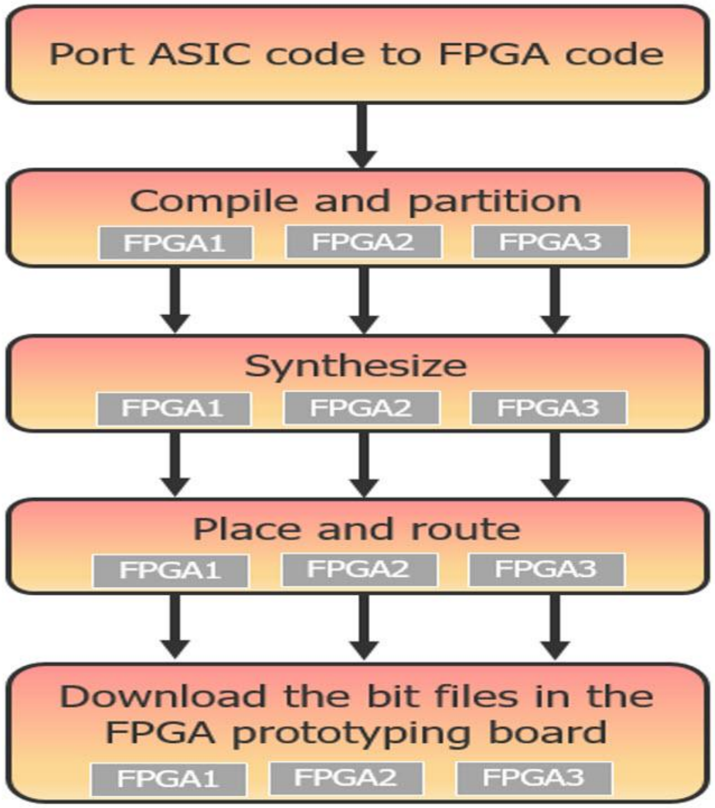

Test bench
Measure coverage
Generate stimulus
Design
Check results
Create a testbench
Instantiate the design in the testbench
Provide stimuli
Compile the testbench and design in the simulator
Run the simulator
Observe the output
Verify that the logic of the design is as intended

Functional Simulation (contd)

- **Advantages**
  - Quick to set up
  - High visibility in the design
  - Bugs are identified early in the design
  - All corner cases of the designs are verified
  - Not expensive
- **Limitations**
  - Very slow speed
  - Complex designs are not easily simulated
  - Verification of the entire IC is difficult
  - All possible scenarios and states are not covered
  - Verification of software is not possible
  - Timing-related issues are not identified

FPGA Prototyping

- FPGA prototyping is the process of verifying the functionality of the system (IC) on FPGAs. With the rise in the complexity of ICs and the increasing demand to shorten the time to market of ICs, FPGA prototyping remains a key solution.
- **Purpose**
  - The purpose of FPGA prototyping is to verify that the design operates as expected when it is driven with live data, and all its external interfaces are working correctly.



FPGA Prototyping (contd)

- **Advantages**
  - High speed
  - Early software development and validation
  - Useful for verifying critical IPs used in applications like aerospace, military, medical etc.
  - Complete IC design is verified
  - Reduces risk of re-spin of ICs
  - Reduces time to market
- **Limitations**
  - Slow to compile (than emulators)
  - Partitioning of designs is error-prone and tedious
  - ASIC to FPGA clock conversion is complex
  - FPGA to FPGA interconnections are limited
  - Very limited debug capability
  - Long bring up time as ASIC code has to be converted to FPGA code
  - Expensive and time consuming to build the FPGA board

Emulator

- Emulation, also called as pre-silicon validation, is the process of verifying the functionality of the system on a hardware device called as an emulator. An emulator can handle both system-level designs (in C, C++, or SystemC) and RTL designs (in Verilog or VHDL). Emulators are much faster than simulators. A design that takes days in simulation, will take only hours in emulation.
- **Purpose**
  - The purpose of emulation is to find issues in the system level design with live data, to verify the system integration, and for development of the embedded software.
- **Workflow**
  - The emulator workflow is similar to the FPGA prototyping flow, except that you use emulator tools in place of the FPGA prototyping tools.

Emulator (contd)

- **Advantages**
  - Higher speed (than simulators)
  - Higher design visibility (than FPGA prototyping) and hence higher debug capability
  - Faster to compile (than emulators)
  - Parallel verification of many designs possible
  - Just like simulators, emulators can be stopped and later started from the same point
  - Verification of embedded software is possible
- **Limitations**
  - Slower to compile (than simulators)
  - Very expensive
  - Long set up time
  - Not all functional paths are covered

Functional vs Formal Verifications

- Functional verification is usually exhaustive, checking all boundary conditions to get a golden RTL, it is prior to synthesis. It is generally done using a tool that does timing verification such as NCSIM/SimVision and ModelSim.

- Formal verification tools compare RTL golden netlist with mapped/gate level/synthesized netlist using formal techniques and algorithms such as bounded model checking, SMV, CTL, or temporal methods by representing the circuit as canonical binary decision diagrams, Kripke structures or even perhaps Petri nets. Bottomline is to determine whether two circuit structures are mathematically equal i.e. they would produce same boolean outputs. This is a rather quick

- In EDA industry a subset of formal verification known as logic equivalence check is used mostly for digital designs although formal techniques are being applied to analog designs and mixed signal circuits as well but BDDs and Kirpke graphs are treated as discrete automata having finite states and transitions/paths.

  0-In (Mentor Graphics)
  Formal Pro (Mentor Graphics)
  Conformal LEC (Cadence)
  VC Formal (Synopsys).