Types of Verification

- Simulation based Pre-Si
- Formal Equivalence
- Post Silicon
- HW – SW Co sim

Verification Technology Options

- The goal of verification is to ensure that the design meets the functional requirements as defined in the functional specification.
- Verification of SOC devices takes anywhere from 40 to 70 percent of the total development effort for the design.

Classification:

1. **Simulation Technologies**
2. **Static Technologies**
3. **Formal Technologies**
4. **Physical Verification and Analysis**

Verification Technology Options

- Simulation Technologies
  - Event-based Simulators
  - Cycle-based Simulators
  - Transaction-based Verification
  - Code Coverage
  - HW/SW Co-verification
  - Emulation Systems
  - Rapid Prototyping Systems
  - Hardware Accelerators
  - AMS Simulation

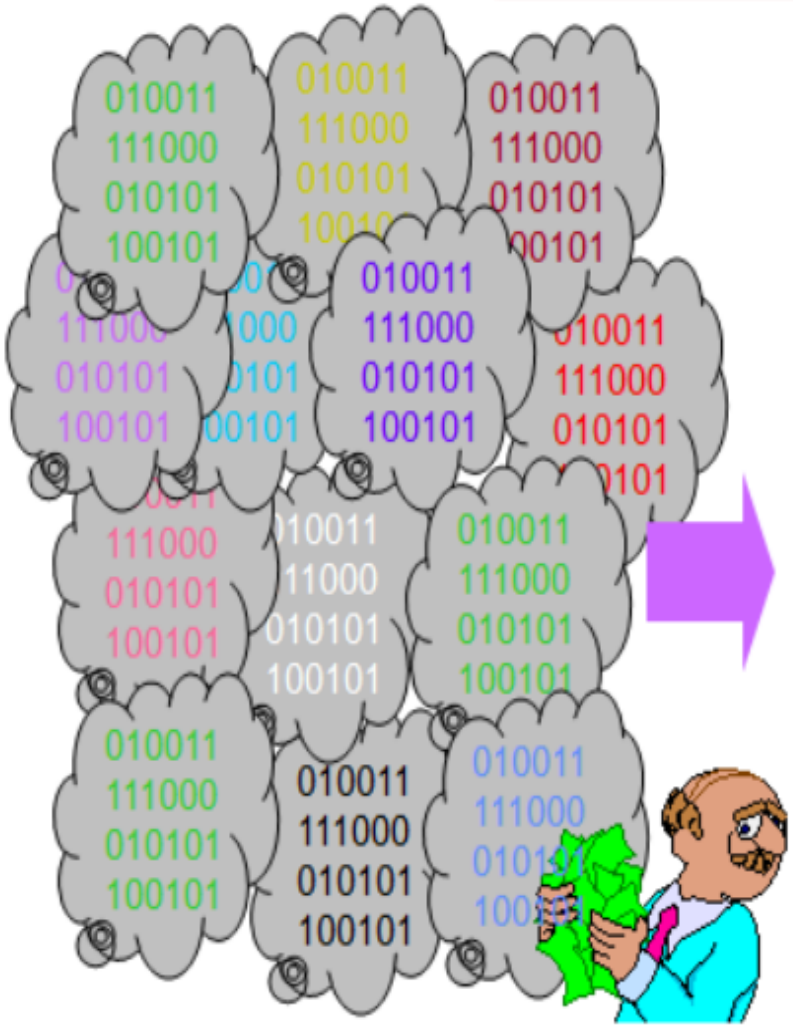
Verification Technology Options

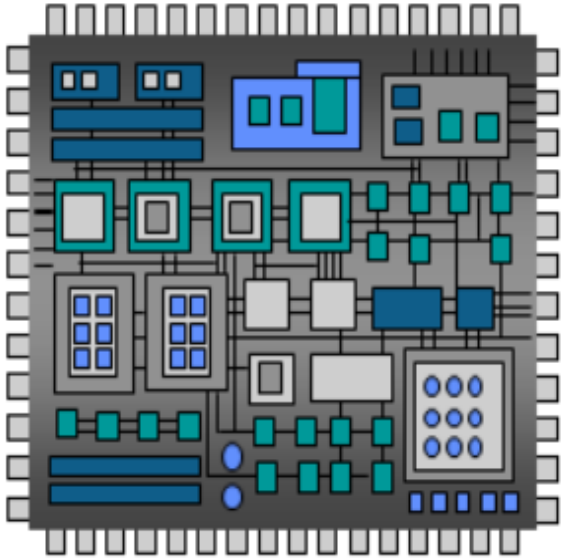
- **Static Technologies:** Does *not require* testbench or test vectors for carrying out the verification
  - **Lint Checking**
  - **Static Timing Verification**

Verification Technology Options

- **Formal Technologies:** *To detect bugs that depend on specific sequences of events.*
  - **Theorem Proving Technique**
  - **Formal Model Checking**
  - **Formal Equivalence Checking**

Formal Verification





impossible to be exhaustive ...

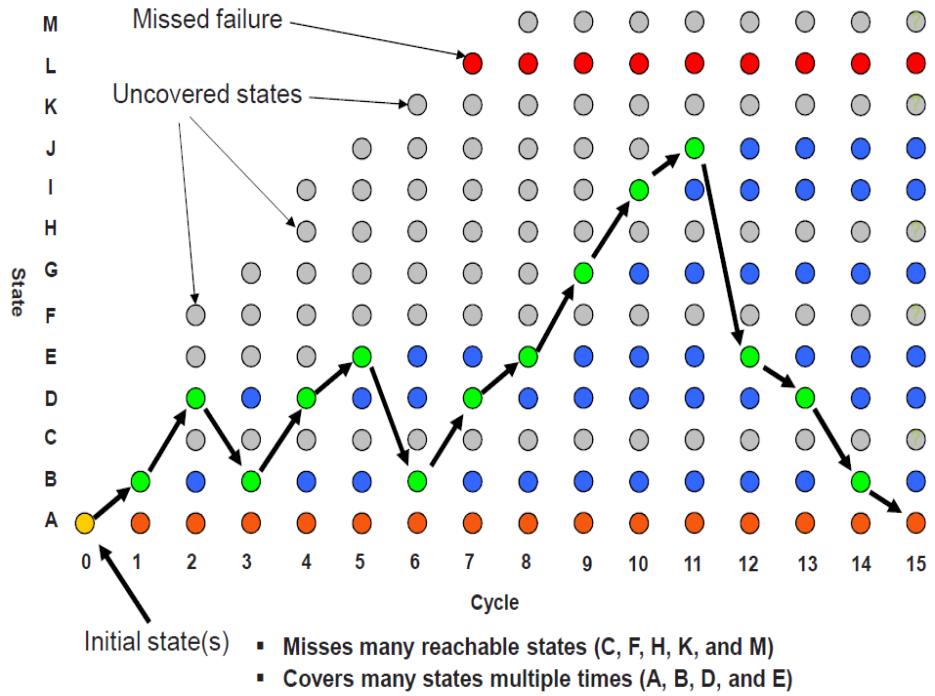
Formal Verification

- Formal Analysis exhaustively verifies assertions and reachability of covers in context of design and constraints
- **Why Formal ?**

  Consider comparing 2 input buses(32 bit wide) for equality. An exhaustive verification via simulation requires 2^64 bit vectors, and at the rate of 1 M vectors/s , it requires 500k + years to simulate. Formal can exhaustively verify it in 0.05 s (CPU time)
- Engineers build mathematical models of system so they can predict their properties through the power of calculation
- Formal Tools provide proprietary formal algorithm engines which attempt to prove each defined property as a mathematical proof rather than actually driving the stimulus.

Formal Flow

- Properties are statements about the **behavior** of a design and its environment.
  - Assertions express desired behavior of the **Design Under Verification (DUV)**
  - Constraints express behavior of the **environment**
  - Cover Statements express desired reachability of states of the **DUV** and the **environment**
- Proving a property is showing that it holds for all possible input combinations across all execution paths.
- Uses Formal Engines or algorithms to prove/disprove property.
  - IFV engines : Sword , Dagger , Bow , Saber , Hammer
  - Different Engines have wildly different performance for different properties.
  - Multiple Engines can run in parallel provided we have enough CPUs and memory.
- Proving runs until a time/system limit reached or Fail/Pass result found for ev assertion.
  - Fail : DUV violates the assertion (tool provides a failure trace).
  - Pass : DUV satisfies the assertion with the constraints.
  - Explored : DUV does not violate the assertion up to a particular depth and Time/System limit has been reached. Requires modifications to get it passed



Verification Technology Options

- **Physical Verification and Analysis:**

  All electrical issues and processes must be considered

  To understand and fix the effects of interconnect parasitics.

  The issues that are to be analyzed and fixed are timing, signal integrity, crosstalk, IR drop, electromigration, power analysis, process antenna effects, phase shift mask, and optical proximity correction.

Verification Technology Options

- **Which Is the Fastest/Best Option**
  - Event based: suited for asynchronous design (both function and timing), small designs
  - Cycle based: only function; no timing, medium designs
  - Formal: smaller designs
  - Emulation: higher speed, very large designs
  - Rapid prototyping: developing s/w for product