UF Herbert Wertheim College of Engineering
UNIVERSITY of FLORIDA

POWERING THE NEW ENGINEER TO TRANSFORM THE FUTURE

Department of Electrical & Computer Engineering

Digital Design

Chapter 06 – Verification

Dr. Christophe Bobda

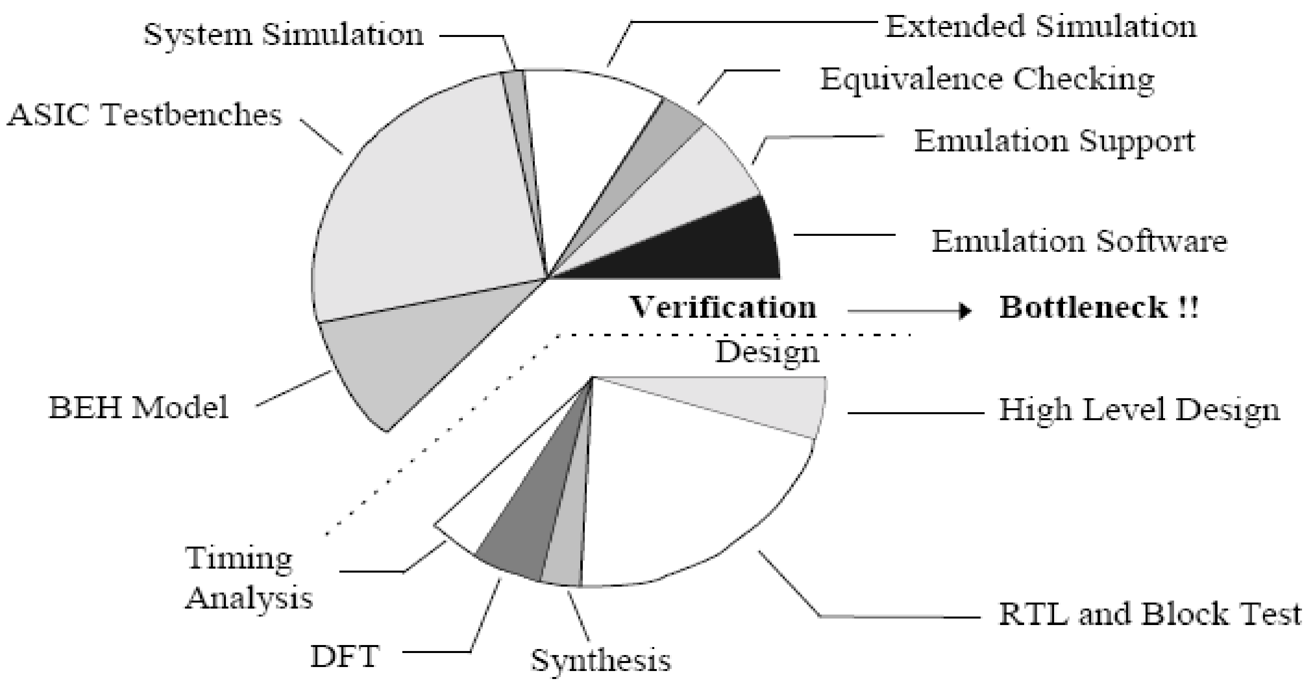
EEL4712C Spring 2024

Outline

1. Introduction
2. Overview of verification techniques
3. Verification flow
4. Verifcation tools
5. Case Studies

1. Introduction

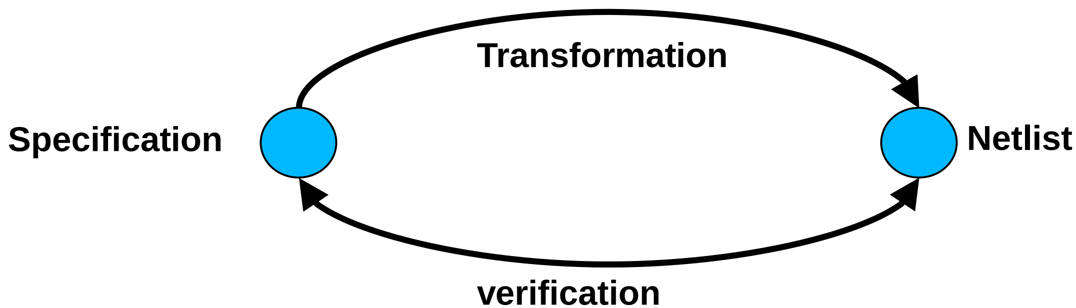
- 60 % to 80 % of design effort is now dedicated to verification
- Industrial example



Source : “Functional Verification on Large ASICs“ by Adrian Evans, etc., 35th DAC, June 1998

1. Introduction

- What is Verification ?
  - A process used to demonstrate the functional correctness of a design
  - To making sure that the implementation matches the specification
  - To ensure that the result of some transformation is as expected

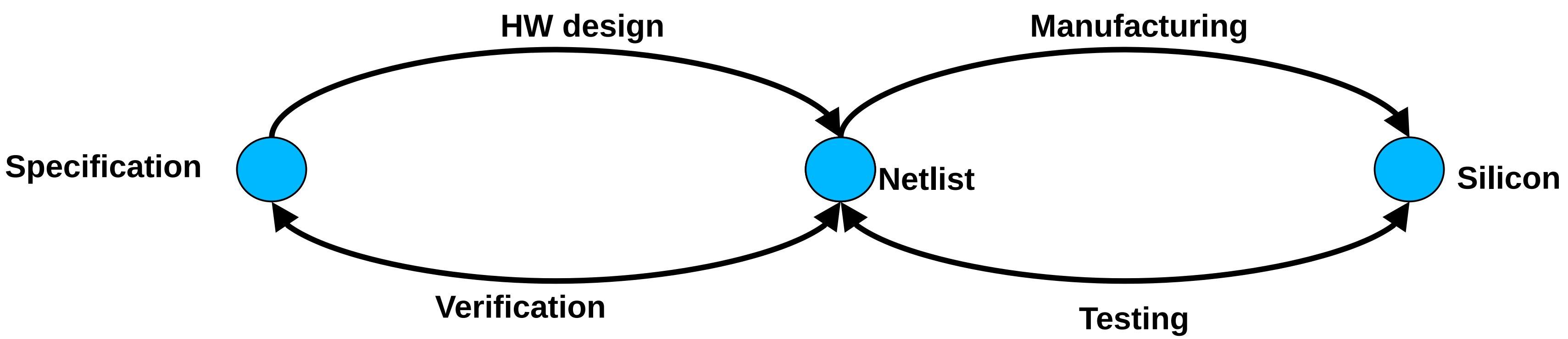


1. Introduction

- Verification Problems
  - Was the specification correct?
  - Did the design team understand the specification?
  - Were the blocks implemented correctly?
  - Were the interfaces between the blocks correct?
  - Does it implement the desired functionality?

1. Introduction

- Testing v.s. Verification
  - Testing verifies manufacturing
    - Verify that the design was manufactured correctly



1. Introduction

- Verification Complexity
  - For a single flip-flop:
    - Number of states = 2
    - Number of test patterns required = 4
  - For a Z80 microprocessor (~5K gates)
    - Has 208 register bits and 13 primary inputs
    - Possible state transitions = $2^{\text{bits+inputs}} = 2^{221}$
    - At 1M Instruction/sec would take 1053 years to simulate all transitions
  - For a chip with 20M gates
    - ??????

1. Introduction

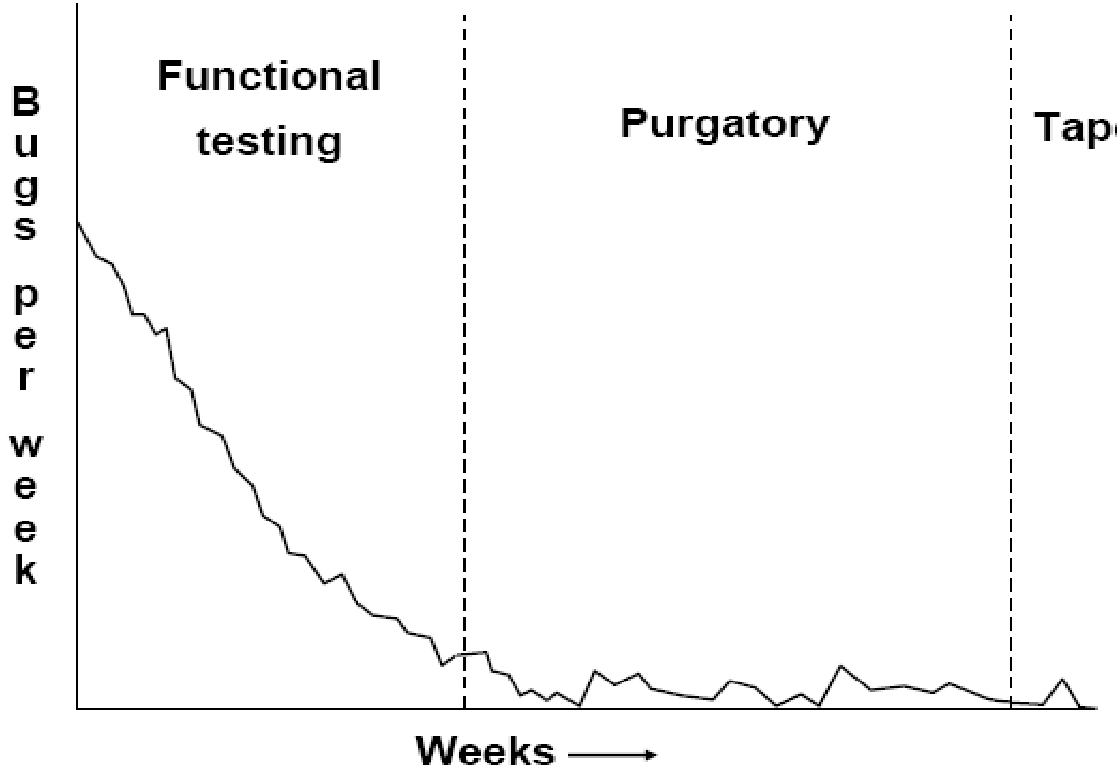
- Reduce verification complexity
  - Using pre-defined and pre-verified building block can effectively reduce the productivity gap
  - Block (IP)-based design approach
  - Platform-based design approach

1. Introduction

- When is Verification Complete ?
  - Some answers from real designers:
    - When we run out of time or money
    - When we need to ship the product
    - When we have exercised each line of the HDL code
    - When we have tested for a week and not found a new bug
    - We have no idea!!
- Designs are often too complex to ensure full functional coverage
  - The number of possible vectors greatly exceeds the time available for test

1. Introduction

- Typical Verification Experience



1. Introduction

- Error-Free Design ?
  - As the number of errors left to be found decreases, the time and cost to identify them increases
  - Verification can only show the presence of errors, not their absence
- Two important questions to be solved:
  - How much is enough?
  - When will it be done?

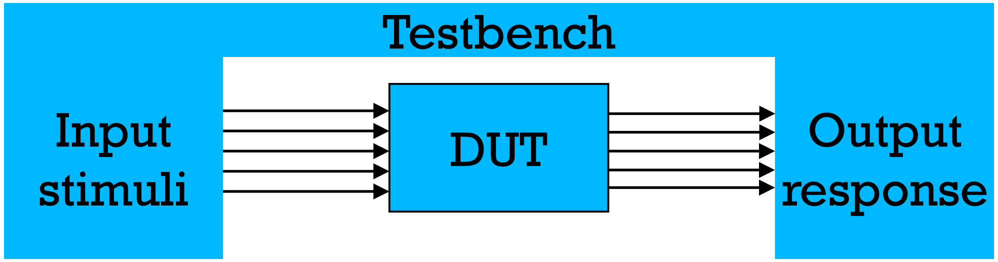
2. Verification technology options

- Simulation-based technologies
- Static technologies
- Formal technologies
- Physical verification and analysis

→ Systems are verified using combination of the afore listed technologies

2. Verification Approaches

- Verification environment



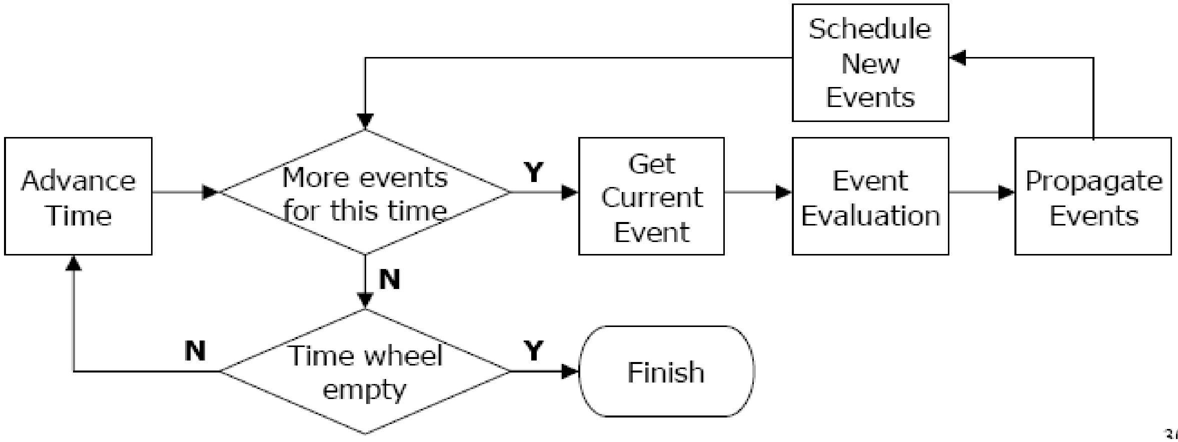
- Terminology
  - Verification environment
    - Commonly referred as testbench (environment)
  - Definition of a testbench
    - A verification environment containing
      - a set of components (bus functional models (BFMs), bus monitors, memory modules) and
      - the interconnect of such components with the design-under-test (DUT)
  - Verification (test) suites (stimuli, patterns, vectors)
      - Test signals and the expected response under given testbenches

2. Verification Approaches

- Testbench Design
  - Auto or semi-automatic stimulus generator is preferred
  - Automatic response checking is highly recommended
  - May be designed with the following techniques
    - Testbench in HDL
    - Tesebench in programming language interface (PLI)
    - Waveform-based
    - Transaction-based
    - Specification-based

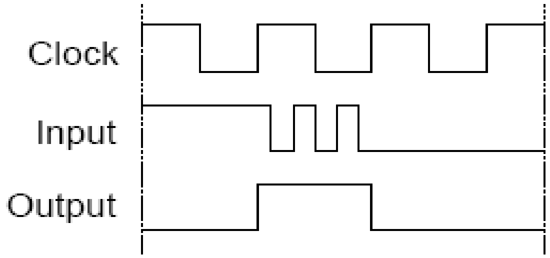
2. Verification technology options

- Simulation-based technologies
  - Event based and cycle-based simulators
    - Event = change in the input stimuli
    - Operate by propagating event through the design until a steady state is reached
    - A logic element may be evaluated several time until the steady state
    - Advantage: simulation accuracy
    - Drawback: slow → complex algorithms are used for event scheduling



2. Verification technology options

- Simulation-based technologies
  - Cycle based verification
    - No notion of time within a clock
    - Evaluate the logic between state/ports in a single shot
    - Each logic element is evaluated only once/cycle
    - Advantage: simulation speed. 5x to 100x faster than event based
    - Drawback:
      - Only work for synchronous design
      - cannot detect glitches in design because they respond only to clock signals
      - Require other tools (ex: Static timing analysis) to check timing problems



2. Verification technology options

- Simulation-based technologies
  - Transaction based verification
    - Allows simulation at the transaction level, in addition to the signal/pin level
    - Create and test all possible transactions between blocks
    - Does not require detailed testbenches
    - Features: Enhance the verification productivity by rising the abstraction level

2. Verification technology options

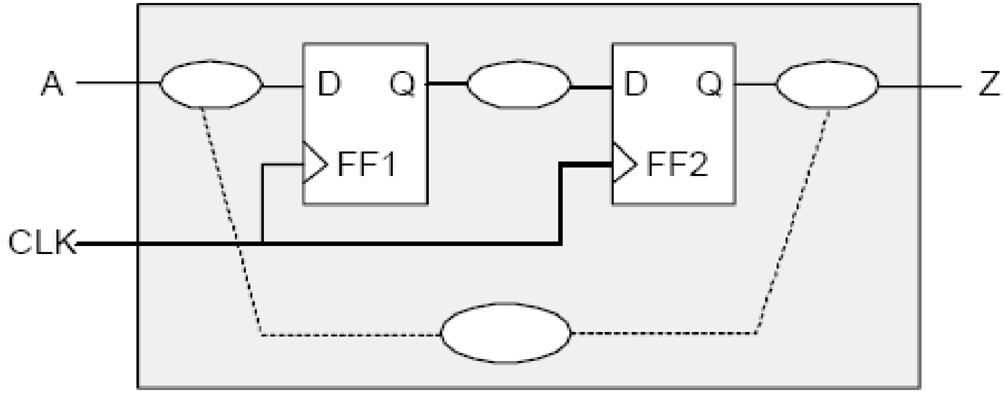
- Code coverage
  - Analysis quantifies the functional coverage of a particular test suite on a given design
  - Usually performs at the RTL-level
  - Coverage includes: statement, toggle, FSM-arc, visited state
  - Features:
    - provide an assessment on the quality of test suites
    - Identifies untested area of a design

2. Verification technology options

- Simulation-based technologies
  - AMS simulation
    - Analog Behavioral Modeling
      - A mathematical model written in **Hardware Description Language**
      - Emulate circuit block functionality by sensing and responding to circuit conditions
      - Available Analog/Mixed-Signal HDL:
        - Verilog-A, VHDL-A, Verilog-AMS, VHDL-AMS

2. Verification technology options

- Static verification technologies
  - Static timing verification (STA)
    - Determines (without simulation) whether the timing requirement will be met
    - Requires no input pattern
    - Very challenging for complex design
      - Each input can have multiple sources
      - Timing can vary depending on the circuit operating condition

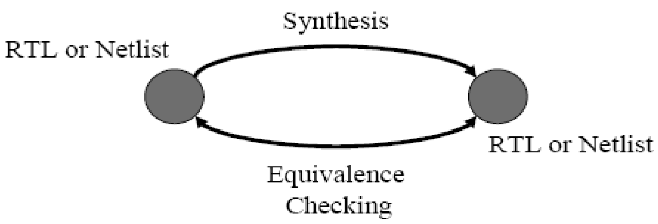


2. Verification technology options

- Formal technologies
  - Difficult to detect bugs that depend on specific sequences of events
  - Those bugs can have serious impact on the design if not detected earlier
  - Formal verification promise to solve this problem
    - Does not need any test bench
    - Theoretically promise a 100% coverage
  - Methods used are:
    - Theorem proving
    - Formal model checking
    - Formal equivalence checking

2. Verification technology options

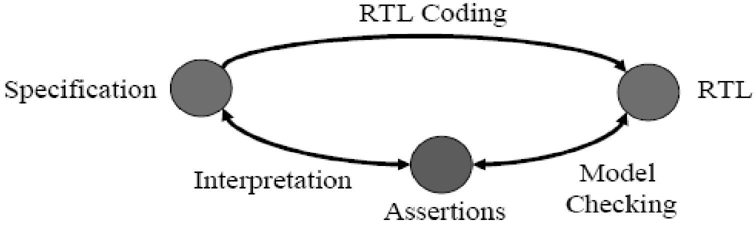
- Equivalence checking
  - Check for the equivalence of two different descriptions of the same design
  - Gaining acceptance in practice
    - Abstract, Avant!, Cadence, Synopsys, Verplex, Verysys
  - But the hard bugs are usually in both descriptions
  - Target *implementation* errors, not design errors
    - Similar to check C v.s. assembly language



  - Gate-level to gate-level
    - Ensure that the post-processing of a netlist did not change the functionality of the circuit
  - RTL to gate-level
    - Verify that the netlist correctly implements the original RTL code
  - RTL to RTL

2. Verification technology options

- Model checking
  - Formally prove or disprove some assertions (properties) of the design
  - The assertions of the design should be provided by users first
  - Described using a formal language
  - Enumerates all states in the STG of the design to check those assertions
  - Gaining acceptance, but not widely used
    - Abstract, Chrysalis, IBM, Lucent, Verplex, Verysys,
  - Drawback: low capacity (~100 register bits)
    - Require extraction (of FSM controllers) or abstraction (of the design)
    - Both tend to cause costly *false* errors



2. Verification technology options

- New approaches
  - The two primary verification methods both have their drawbacks and limitations
    - Simulation: time consuming
    - Formal verification: memory explosion
  - We need alternate approaches to fill the productivity gap
    - Verification bottleneck is getting worse
  - Semi-formal approaches may be the solution
    - Combine the two existing approaches
    - Completeness (formal) + lower complexity (simulation)

2. Verification technology options

- Physical verification and analysis
  - Test is performed to insure any defect free device fabrication
  - Test vectors focus on the device structure rather than on the functionality
  - SoC consist of processors, DSPs, memory, custom hardware

    →This level of complexity pose the following challenge in testing
    - Test vectors:
      - enormous set of possibilities
    - Core forms: cores are available in different forms (soft, hard, firm)
      - Developed by different suppliers
      - Different techniques applied for testing

2. Verification technology options

- Physical verification and analysis
  - Test strategies
    - Logic BIST (built-in self test): test logic circuit
      - Stimulus generators and response verifiers are embedded within the circuit
    - Memory BIST: test memory cores
      - Incorporates an on-chip address generator, data generator, and read/write controller applies a common memory test algorithm to test the memory

2. Verification technology options

- Physical verification and analysis
  - Mixed-signal BIST:
    - BIST used for AMS cores, such as ADC, DAC, and PLL
  - Scan chain: assesses timing and structural compliance
    - A set of scan cells are connected into a shift register
    - Connect scan cell's output port of one Flip-Flop to dedicated scan input port of the proceeding Flip-Flop
    - Scan cell = sequential element connected as part of a scan chain
    - Upon inserting scan chain into a design, ATPG (automatic test pattern generator) tools can be used to generate manufacturing test automatically

3. Verification Plan

- Verification plan is part of the design reports
- Contents
  - Test strategy for both blocks and top-level module
  - Testbench components: BFM, bus monitors, ...
  - Required verification tools and flows
  - Simulation environment including block diagram
  - **Key features** needed to be verified in both levels
  - Regression test environment and procedure
  - **Clear criteria** to determine whether the verification is successfully complete

3. Verification Plan

- Verification plan enables
  - Developing the testbench environment early
  - Developing the test suites early
  - Developing the verification environment in parallel with the design task by a separate team
  - Focusing the verification effort to meet the product shipment criteria
  - Forcing designers to think through the time-consuming activities before performing them