Lecture 21

Verification of Digital Systems (Chapter 10)

Introduction

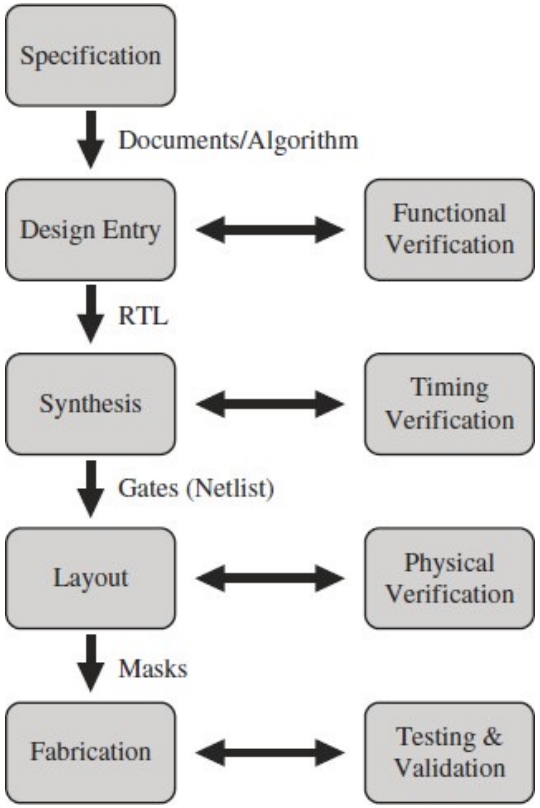
- Verification has gained a prominent place in chip design flow due to the need to deliver correctly functioning chips to the customer
- Verification in this context means pre-silicon functional testing.

Importance of Verification

- Verification is done in all stages of design.
- **Functional verification**: checking whether the system does what it is expected to do.
  - Happens early in the design process.
- **Timing verification**: checks whether there are any timing violations.
  - Happens after design is synthesized.

Importance of Verification (continued)

- Types of verification in different design stages:



Importance of Verification (continued)

- Objective of verification: make sure that the chip meets the product specification.
  - Find bugs before the customer does.
  - Best to catch errors early in the design stage, as the cost of a bug typically increases in later stages of product creation.
- Common error sources:

Source	Percentage
Functional/Logical	50
Timing	15
Signal Integrity	15
Power	5
Clocking	5
Other	10

Importance of Verification (continued)

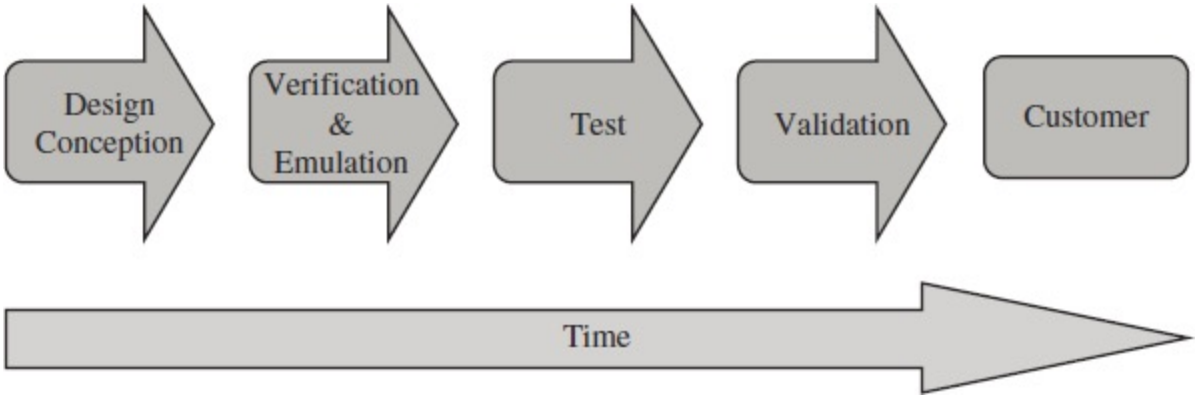
- Verification may take 50-75% of design time.
- The Intel Pentium Bug:
  - A bug in the floating-point unit of the 66MHz Pentium P5 chip.
  - Error showed up in the 4th decimal digit of certain division calculations, when most computers could return 15 digits correctly.
  - Cost to fix when discovered in July 1994 was several hundred thousand dollars.
  - However, Intel waited to respond and spent $475 million to replace flawed processors in January 1995.

Verification Terminology

- **Verification** specifically refers to the uncovering of functional bugs during the RTL coding stages. It is also called ***functional verification***.
- **Emulation** is the creation of specialized hardware prototypes that are then used for pre-silicon validation. The availability of a near-complete system allows to run software and helps to uncover bugs that could not be found in RTL verification stages.
- **Testing** refers to manufacturing test. It is performed on manufactured parts to uncover errors that creep in during manufacturing/fabrication.
- **Validation** refers to the testing done on a full-system with operating-system and real-system software using the first "alive" silicon on development boards.

Verification Terminology (continued)

- Exact terminology may differ between companies, but verification always precedes validation.
- Timeline:



Verification Terminology (continued)

Verification	Testing	Validation	Emulation
Done pre-silicon	Done post-silicon	Done on first "alive" silicon	Done pre-silicon
Also called functional verification	Also called manufacturing test	Also called post silicon validation	Also called pre-silicon validation
Targets design bugs (functional bugs)	Targets manufacturing defects	Targets bugs that might have escaped in pre-silicon verification. Also tests power and electrical signals.	Targets design bugs (functional bugs)
Goal is to check that the system implements the functional specification	Goal is to check that each manufactured part correctly implements design	Goal is to check that the system meets the user requirements	Goal is to check that the system implements the functional specification; an alternate goal is to prepare for validation
Done when the chip is being designed	Done after the chip is manufactured	Performed after the chip has passed "testing"	Performed during design, before first silicon
Performed once prior to manufacturing	Performed on each part after manufacturing	Performed on "first silicon" but before product release	Performed once after verification but before validation

Verification Terminology (continued)

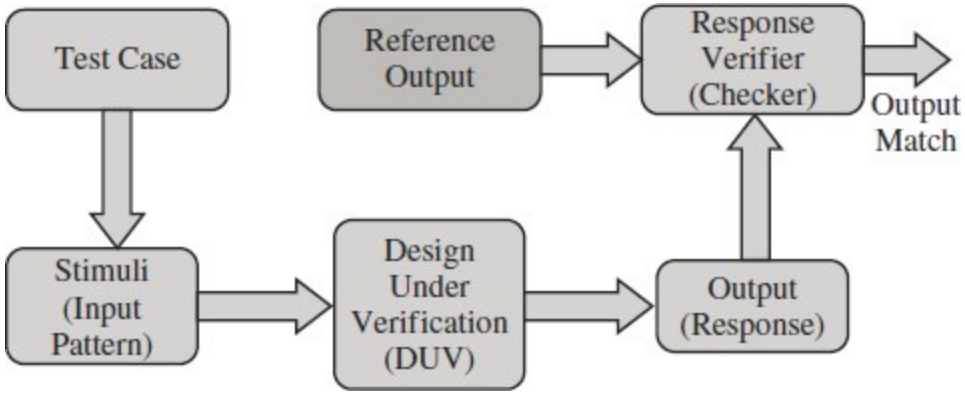
Verification	Testing	Validation	Emulation
Tests quality of design	Tests quality of shipped parts	Tests quality of design	Tests quality of design. Helps to bring up "alive" silicon quicker
Can control and observe all/many nodes	Can control and observe only limited nodes, i.e., only input-output pins	Can control and observe only limited nodes, i.e., only input-output pins	Can control and observe more nodes than actual product but generally less than a simulator model
Slow	At speed	At speed	Faster than simulation; possibly slower than final product
Performed on models built using VHDL, Verilog, System Verilog, etc.	Performed on silicon	Performed on first "alive" silicon, using validation boards. Operating system is booted, and real tests are done. Power and electrical measurements are done in addition to functional checks	Performed on specialized hardware built using FPGAs or other prototyping technologies, e.g., Palladium from Cadence, Veloce from Mentor Graphics
Limitation: Slow. Not at-speed; hence timing verification cannot be done; analog blocks missing; real clocking missing; software cannot be booted	Limitation: Very limited controllability and observability	Limitation: Limited controllability and observability	Limitation: Still not at-speed; hence timing verification cannot be done; analog blocks missing; real clocking missing

Functional Verification

- Consists of providing input test stimulus to test various aspects of functionality of the system.
- Self-Checking Test Benches:
  - Provide stimuli to the system and automatically monitors the circuit outputs and compares them against the expected outputs.
  - System under verification is called **design under verification** (**DUV**).

Functional Verification (continued)

- Self-Checking Test Benches (continued):
  - Overview:



Functional Verification (continued)

- Self-Checking Test Benches (continued):
  - **Test Benches with Golden Vectors**: A golden vector is a correct or expected output for a certain input stimuli.
  - **Test Benches with Golden Models or Reference Models**: Test benches can contain a golden model that calculates the expected outputs based on the input stimulus. The verification process checks whether the synthesizeable RTL model would give the same outputs as the golden model.

Functional Verification (continued)

- Verification Flow (continued):
  - **Directed testing**: a method for selecting a feature, writing a test to verify that feature, and repeating until all features are tested.
  - **Constrained random testing**: consists of generating random stimuli that are statistically derived to verify random parts of the design.
  - **Regression testing**: rerunning previously passed tests to ensure that modifications to the design remain consistent with previously verified functionality.

Functional Verification (continued)

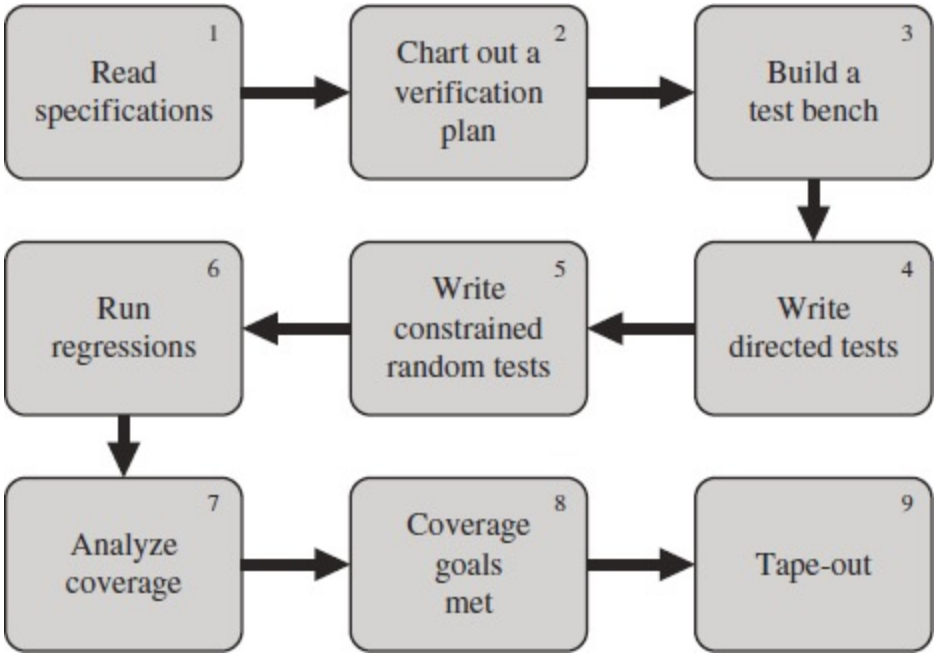
- Verification Flow (continued):
  - **Functional coverage**: denotes how much of the functionality has been verified.
  - **Code coverage**: quantifies verification process considering the RTL code. Examples:

Line coverage	Percentage of lines of the RTL code that got executed during simulation
Toggle coverage	Percentage of signals/bits that toggled from 0 to 1 and 1 to 0 at least once
FSM coverage	Percentage of finite state machine states reached and state transitions traversed during simulation
Condition/branch coverage	Percentage of conditions hit for each conditional statement in RTL code. Line coverage just tells that the particular line was executed. It does not tell whether it was executed in the multiple conditions it could possibly execute. For instance, going through a loop once did not test the other condition of the loop exit.

  - **Tape-out**: the point at which the photomask of an integrated circuit is sent for manufacture.

Functional Verification (continued)

- Verification Flow:
  - Illustration:



Functional Verification (continued)

- Verification Approaches:
  - **Black Box Approach**: verification engineers use only the external interfaces as defined by the specification to create the test bench.
  - **White Box Approach**: a complete understanding of the internal structures of the DUV is assumed. The verification engineer can observe the internal nodes of the design and can examine the behavior of internal structures.
  - **Gray Box Approach**: a combination of both black box and white box approaches. Some internal signals are available while the internals of the rest of the DUV cannot be observed.

Functional Verification (continued)

- Verification Languages:
  - Test benches in earlier chapters used VHDL.
  - Verification of industry-grade designs needs more software support.
  - Languages targeted for verification:
    - System Verilog
    - Vera
    - Specman e

Functional Verification (continued)

- Verification Languages (continued):
  - VHDL contains several useful constructs for test benches:
    - ASSERT, REPORT, and SEVERITY
    - Assert statement enables **assertion-based testing**: assertions capture specifications in an executable form. If errors exist, they produce error reports.
  - **Scripting languages** such as Perl and Python are used to create test inputs into files.
  - **Transaction level modeling** (**TLM**): a type of modeling at a higher abstraction level than RTL.
  - **Open Verification Methodology** (**OVM**): provides verification support in the form of building-blocks (libraries) which can be used by verification engineers.
  - **Formal verification**: mathematics is applied to solve the verification problem.

Timing Verification

- Once basic functionality of the design is checked using RTL models, the RTL model is synthesized.
- **Timing verification** is done at this stage in order to check that the circuit meets timing requirements.
- **Timing closure** is the process by which a design is modified to meet its timing requirements.
- EDA tools are capable of making these changes; timing verification is done before tape-out.

Timing Verification (continued)

- Sequential Circuit Timing Basics:
  - Major timing parameters of a flip-flop:

Propagation delay (Clock-to-Q delay)	The time that elapses between the active edge of the clock to the time the flip-flop output changes
Setup time	The time duration for which the flip-flip input must be stable before the active edge of the clock
Hold time	The time interval for which the flip-flop input must be stable after the active edge of the clock

- Static Timing Analysis:
  - Maximum clock frequency is determined by the worst-case delay of the longest path in the circuit (**critical path**).
  - **Static timing analysis** (**STA**): method of estimating the maximum frequency of operation and other timing properties of a design under worst-case assumptions.

Timing Verification (continued)

- Static Timing Analysis (continued):
  - All circuit paths are considered, but no specific input values are assumed.
  - Checks whether there are timing violations if each gate/flip-flop worked at its highest or lowest speed or anything in between.
  - Relevant only for circuits with a clock (synchronous).
  - Two possible types of errors:
    - **Setup time violation**: data arrives too late without providing enough setup time for the flip-flop.
    - **Hold time violation**: input data changes too soon after the clock's active transition without providing enough hold time for the flip-flop.
  - **Slack** is the amount of time left before a signal will violate a setup or hold-time constraint.