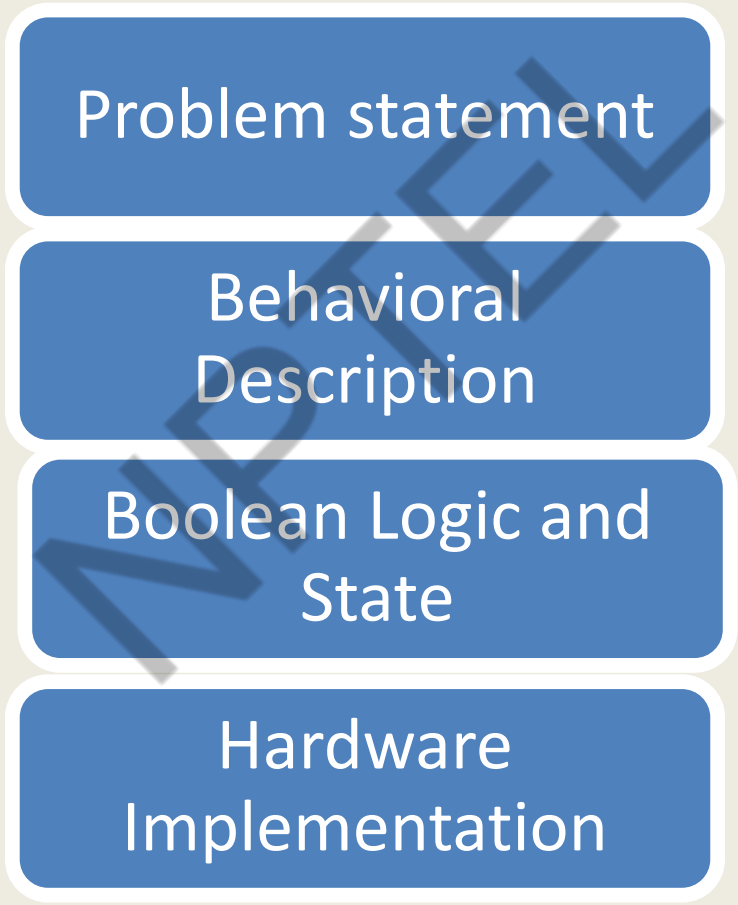
HDL INTRODUCTION

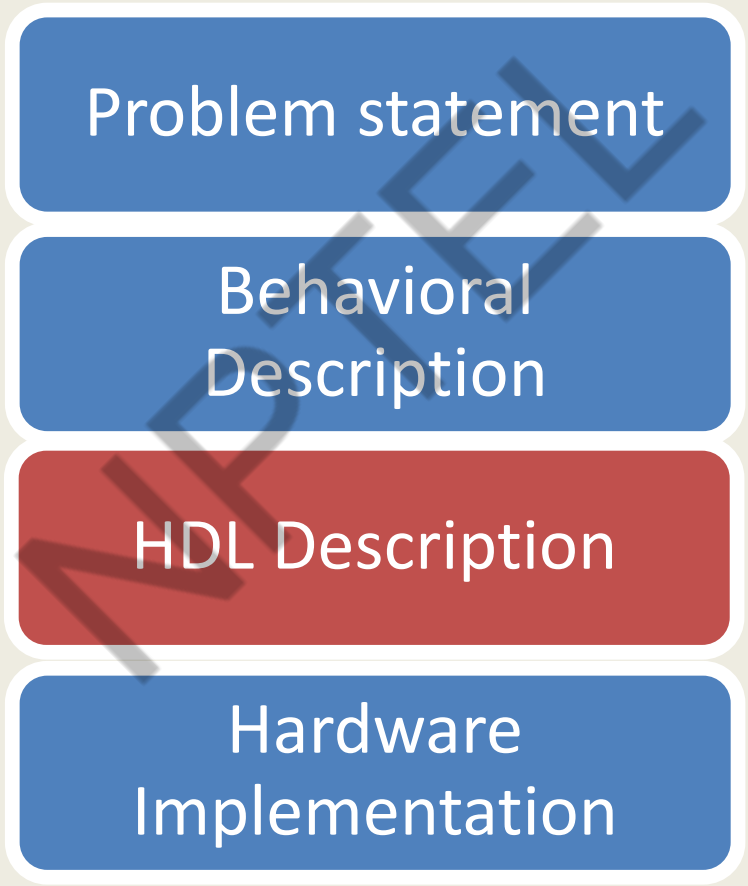
Course: Embedded systems (CS60087)

Prepared By: Vidya Govindan (TA)

Building Digital Systems



Building Digital Systems with HDLs



Motivation for HDLs

- A specification is an engineering contract that lists all the goals for a project:
  - *Goals include area, power, throughput latency, functionality, test coverage, costs(NREs costs) etc. Helps you figure out when you're done and how to make engineering tradeoffs. Later on goals help remind everyone (especially management) what was agreed to at the outset!*
  - *Top down design: partition the project into modules with well defined interfaces so that each module can be worked on by a separate team.*

    for eg. well defined ISA
- A behavioral model serves as an executable functional specification that documents the exact behavior of all the individual modules and their interfaces. Since one can run tests , this model can be refined and finally verified through simulation.
- We need a way to talk about what hardware should do without actually designing the hardware itself, i.e., we need to separate behavior from implementation. We need a

Hardware Description Language

- If we were then able to synthesize an implementation directly from the behavioral model, we'd be in good shape!

Advantages of using HDLs

- Designs can be described at various levels of abstractions
  - *Addresses the current digital system complexity and* makes design of larger systems easier.
  - *Describe what you need the hardware to do, tools then design the hardware for you.*
- Functional Simulation Early in the Design Flow
  - *Allow modeling and simulating the functional behavior and timing of digital hardware.*
  - *Simplified & faster design process.*
- Automatic Conversion of HDL Code to Gates
  - *Synthesis tools take an HDL description and generate a technology-specific netlist.*
  - *Lessen the time spent debugging the design*
  - *Design errors still possible, but in fewer places generally easier to find and fix.*
- Early Testing of Various Design Implementations
  - *Due to fast synthesis, there is a scope for trying different implementations.*
- Design Reuse
  - *Don't manually change all transistors for rule change*.
  - *Technology independence, standardization, portability, ease of maintenance.*

A Tale of Two HDLs

VHDL

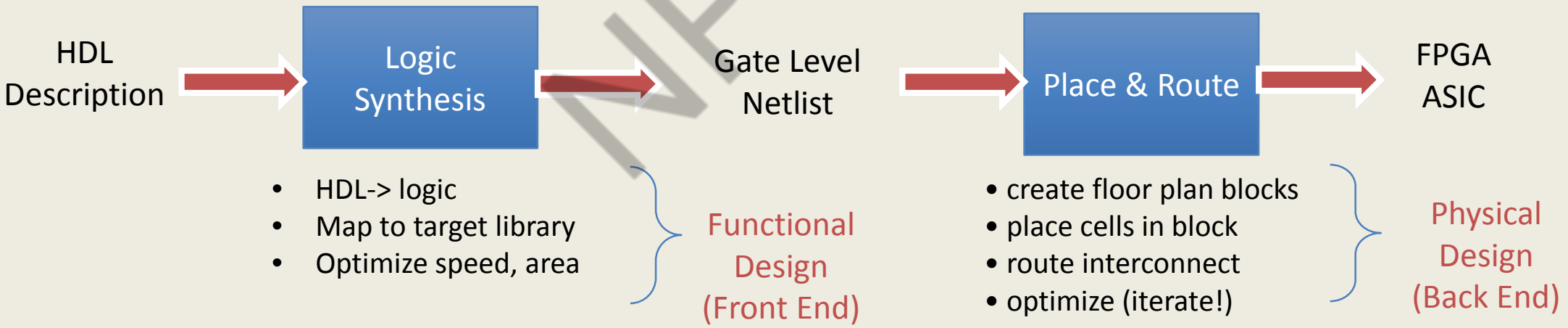
- Initially created for ASIC synthesis
- ADA-like verbose syntax, strongly and richly typed language
- Strong support for package management and large designs.
- Design is composed of entities each of which can have multiple architectures. A configuration chooses what architecture is used for a given instance of an entity.
- Behavioral, structural, logic-level modeling
- Synthesizable subsets
- Harder to learn and use, lacks language defined simulation commands.

Verilog

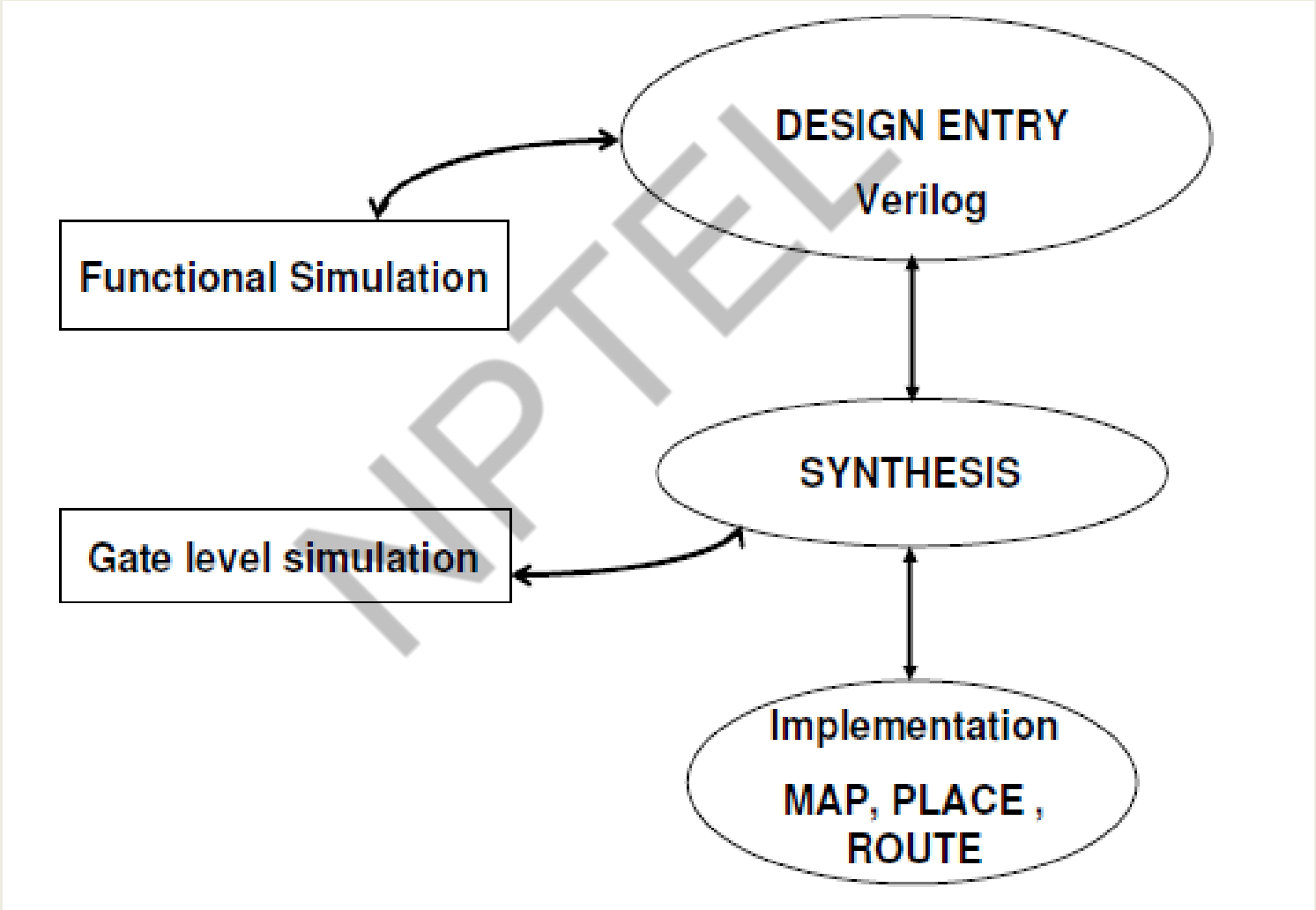
- Initially an interpreted language for gate-level simulation.
- C-like concise syntax, weakly and limited typed language.
- No special extensions for large designs.
- Design is composed of modules.
- Behavioral, structural, logic-level modeling
- Synthesizable subsets
- Easy to learn and use, fast simulation.

Synthesis of HDLs

- So, we have an executable functional specification that
  - *documents exact behavior of all the modules and their interfaces*
  - *can be tested & refined until it does what we want.*
- An HDL description is the first step in a mostly automated process to build an implementation directly from the behavioral model.



HDL Implementation Cycle



References

- Slide source
- Comparision of HDLs