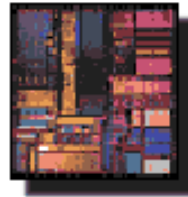




Digital Systems Design Automation

Unit 1: Course Introduction and Overview

Lecture 1.6: A Quick Tour of Logic Level Design Automation

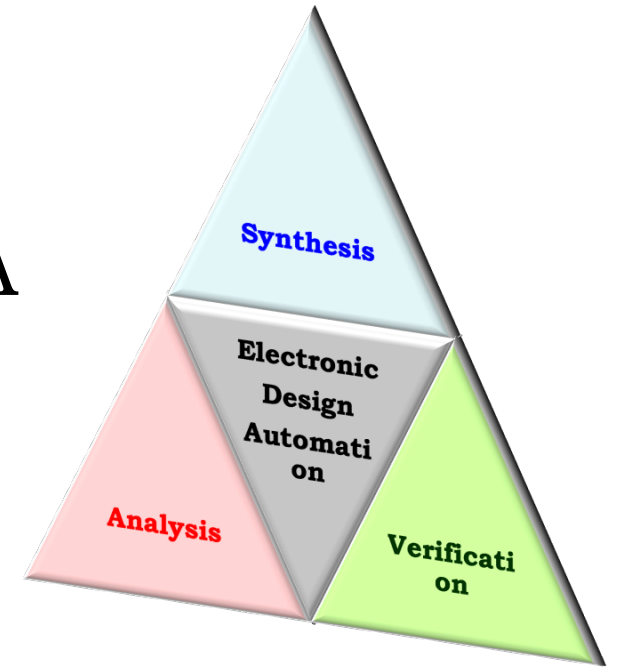


Anand Raghunathan

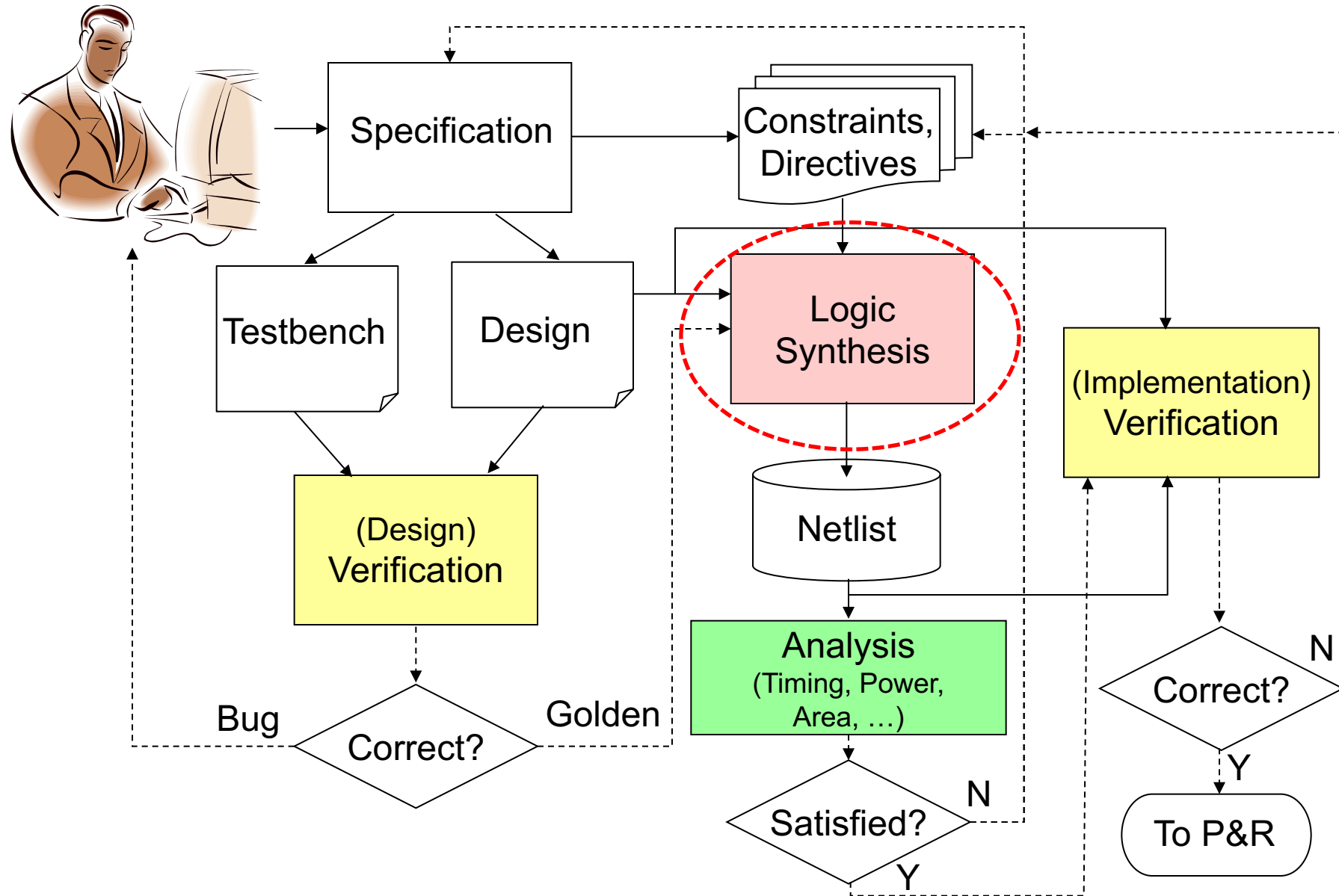
raghunathan@purdue.edu

Outline

- 1.1 Moore's Law
- 1.2 Design Complexity and need for EDA
- 1.3 Course Overview
- 1.4 Taxonomy of integrated circuits
- 1.5 Levels of abstraction in IC design
- 1.6 A quick tour of logic level design automation



A Closer Look at the Logic Design Flow



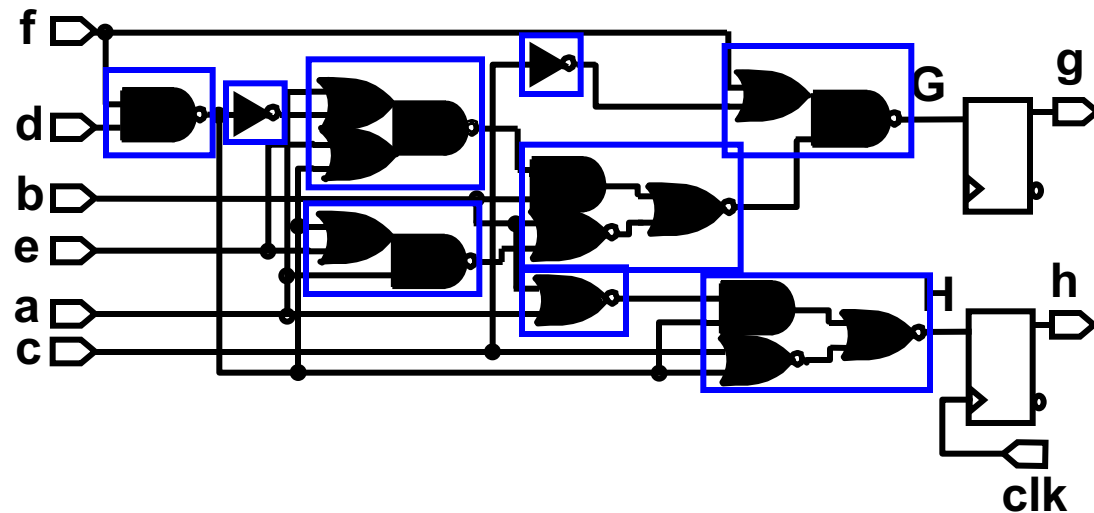
Logic Synthesis

Register-Transfer Level Specification

Logic Extraction (Elaboration)

Technology-Independent Optimization

Technology Mapping



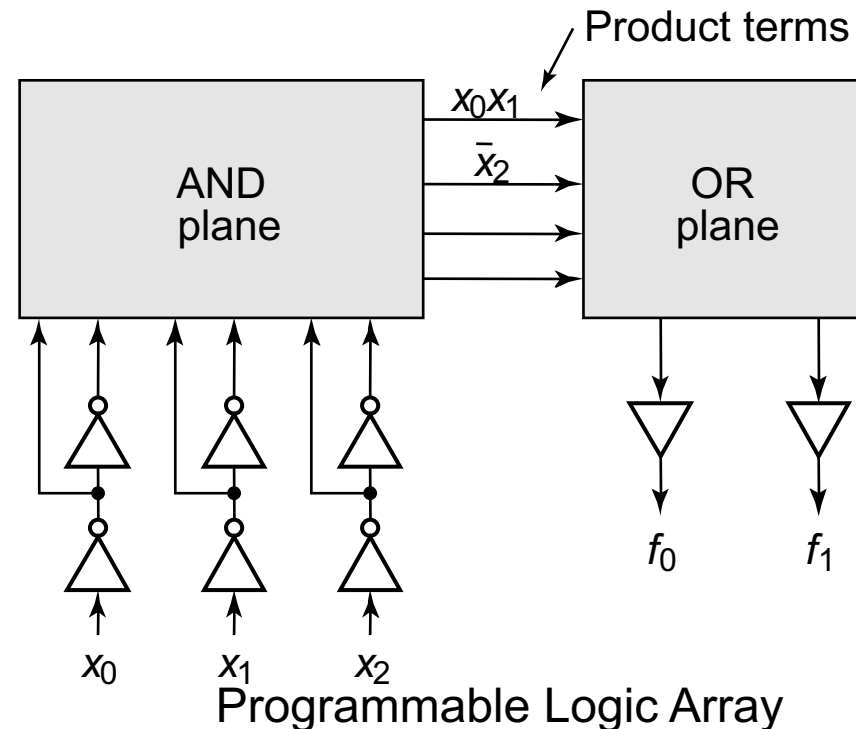
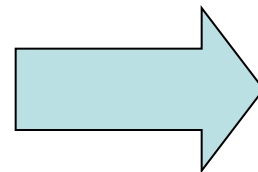
A Quick Tour Through Logic-Level Design Automation

- **Two-level combinational circuit synthesis**
- **Input:** Set of Boolean equations
- **Output:** Minimal two-level implementation
- Used for Programmable Logic Array (PLA) implementations
- Two-level implementations are not very scalable, but synthesis techniques are useful in multi-level context as well

$$f_0 = x_0x_1 + \bar{x}_2$$

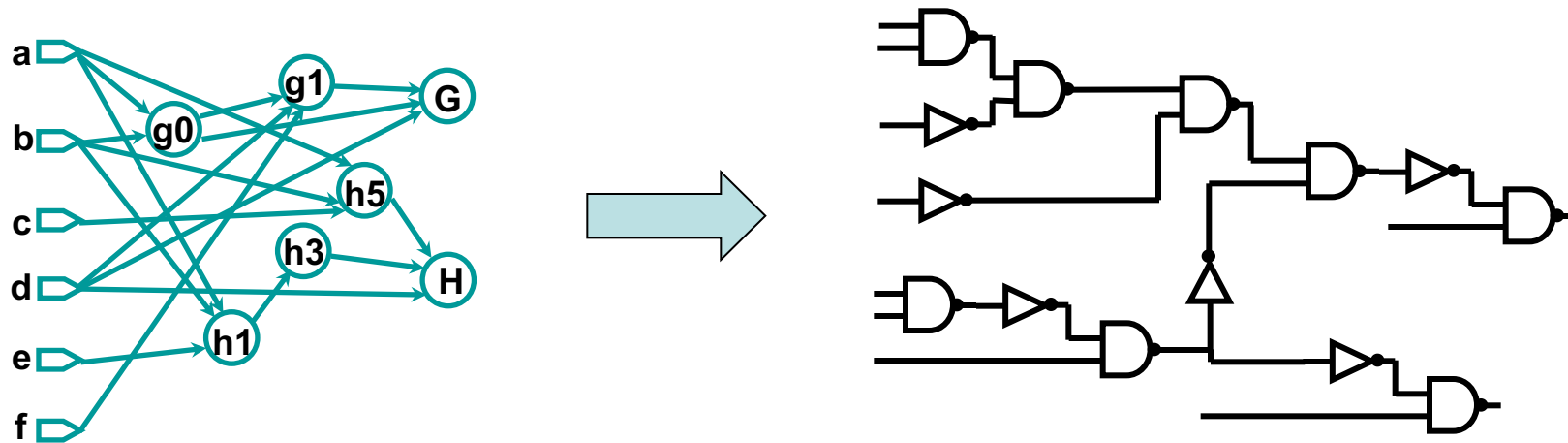
$$f_1 = x_0x_1x_2 + \bar{x}_2 + \bar{x}_0x_1$$

Any combinational logic function can be expressed in the form of Boolean equations



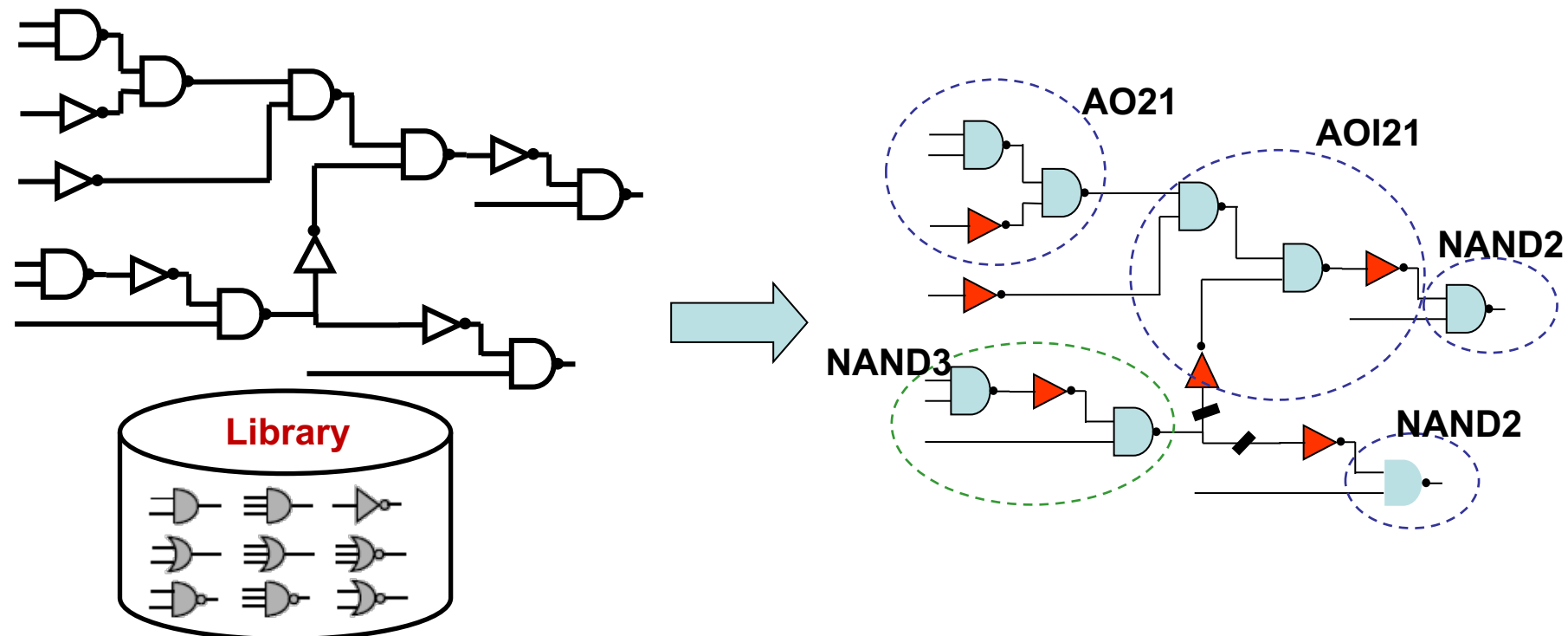
A Quick Tour Through Logic-Level Design Automation

- **Multi-level combinational circuit synthesis**
- **Input:** Set of Boolean equations OR un-optimized Boolean network elaborated from HDL
- **Output:** Optimized multi-level implementation (network of gates)



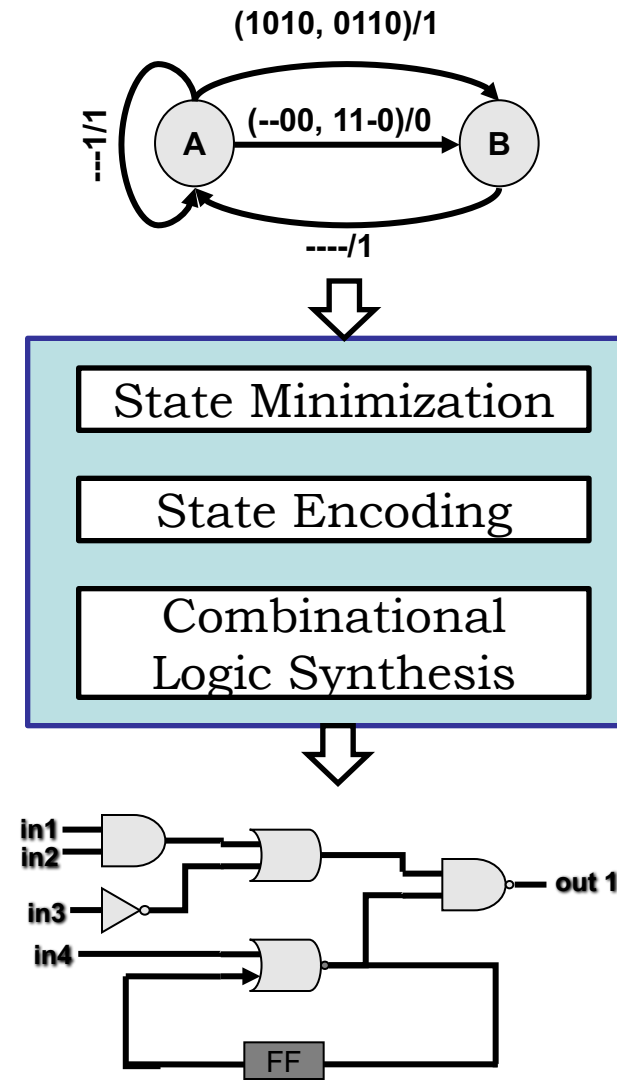
A Quick Tour Through Logic-Level Design Automation

- **Technology Mapping**
- **Input:** Technology-independent implementation, Cell library
- **Output:** Implementation mapped to cells in library



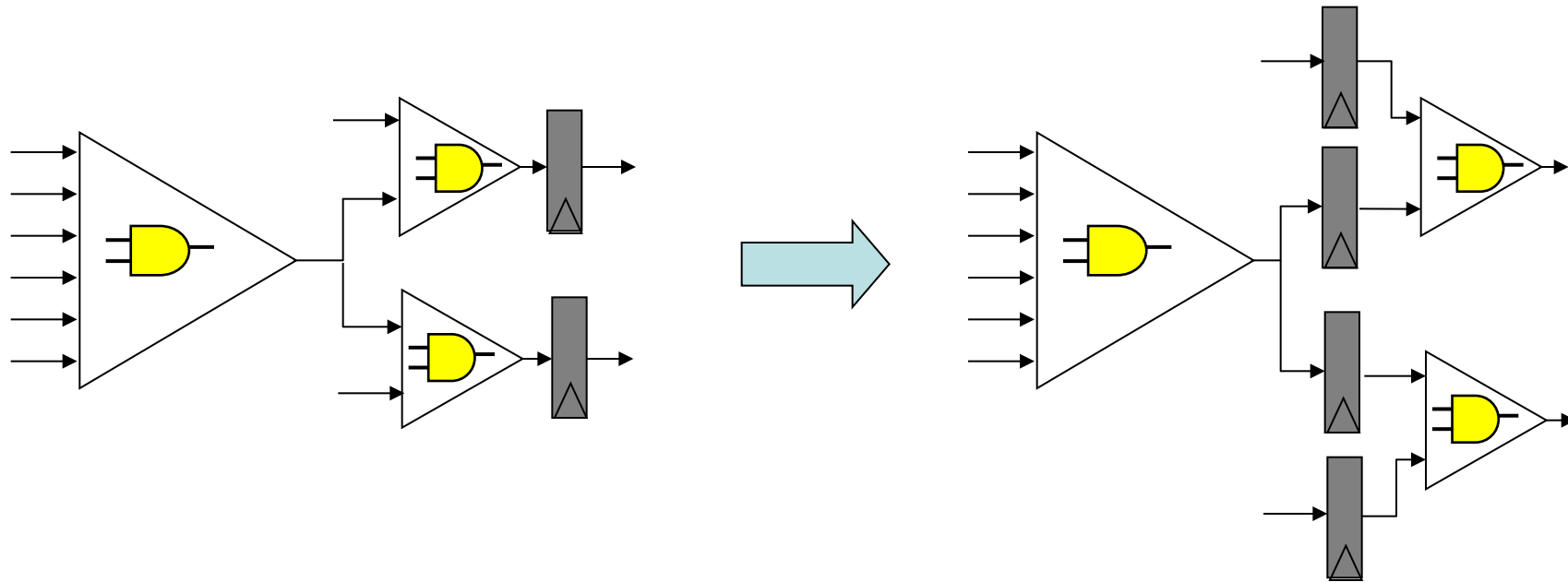
A Quick Tour Through Logic-Level Design Automation

- **Sequential Logic Optimization: FSM Synthesis**
- **Input:** Finite State Machine specification
- **Output:** Optimized implementation (circuit consisting of logic gates and storage elements)

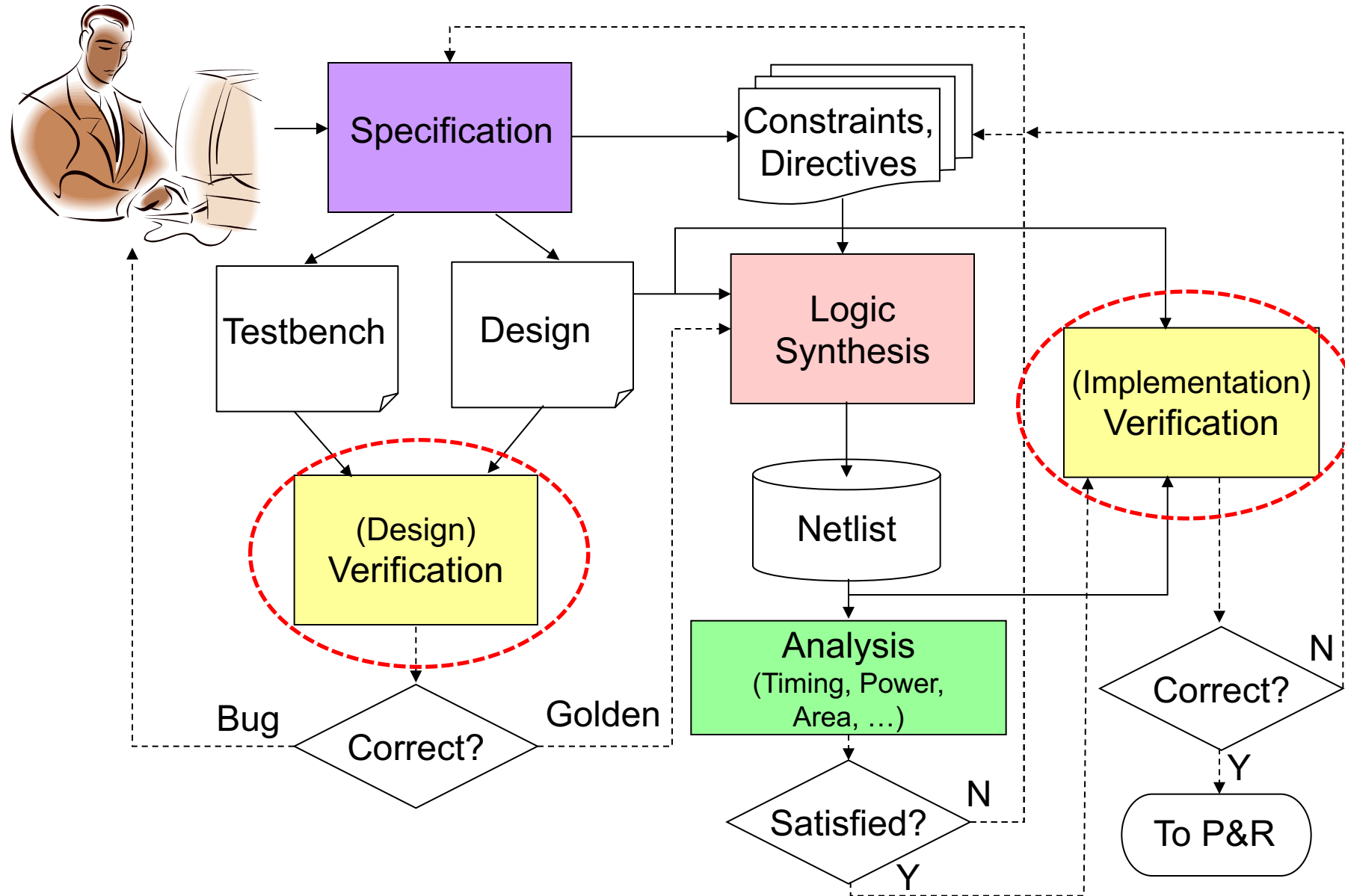


A Quick Tour Through Logic-Level Design Automation

- **Sequential Logic Optimization: Retiming**
- **Input:** Structural implementation consisting of gates and FFs
- **Output:** Optimized implementation with improved area / speed / power.

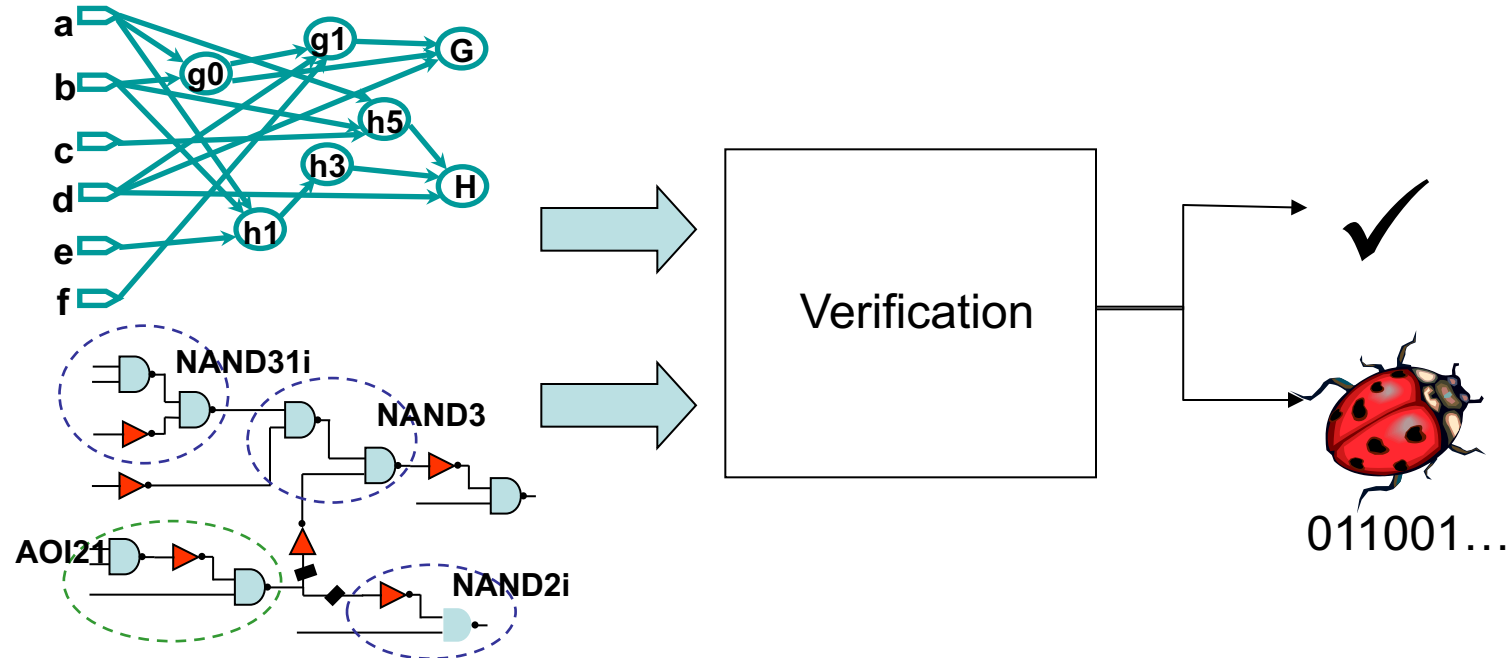


A Closer Look at the Logic Design Flow



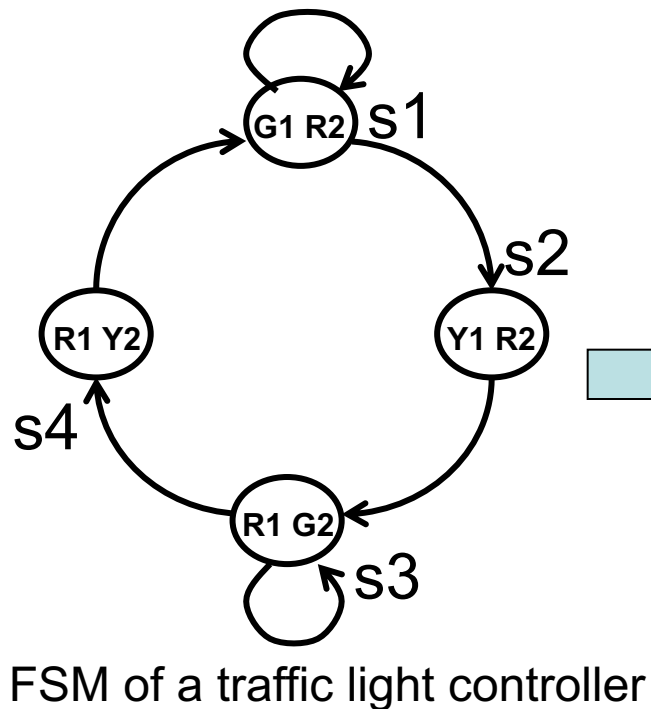
A Quick Tour Through Logic-Level Design Automation

- **Verification: Equivalence Checking**
- **Input:** Specification (RTL, Boolean equations), Optimized implementation (netlist)
- **Output:** Proof that specification == implementation OR counterexample demonstrating otherwise



A Quick Tour Through Logic-Level Design Automation

- **Verification: Property Checking**
- **Input:** Specification or implementation, properties that must hold (e.g., assertions)
- **Output:** Proof that property holds OR counterexample demonstrating otherwise



Property: Both lights never
green at the same time
 $AG \neg (G1 \wedge G2)$

