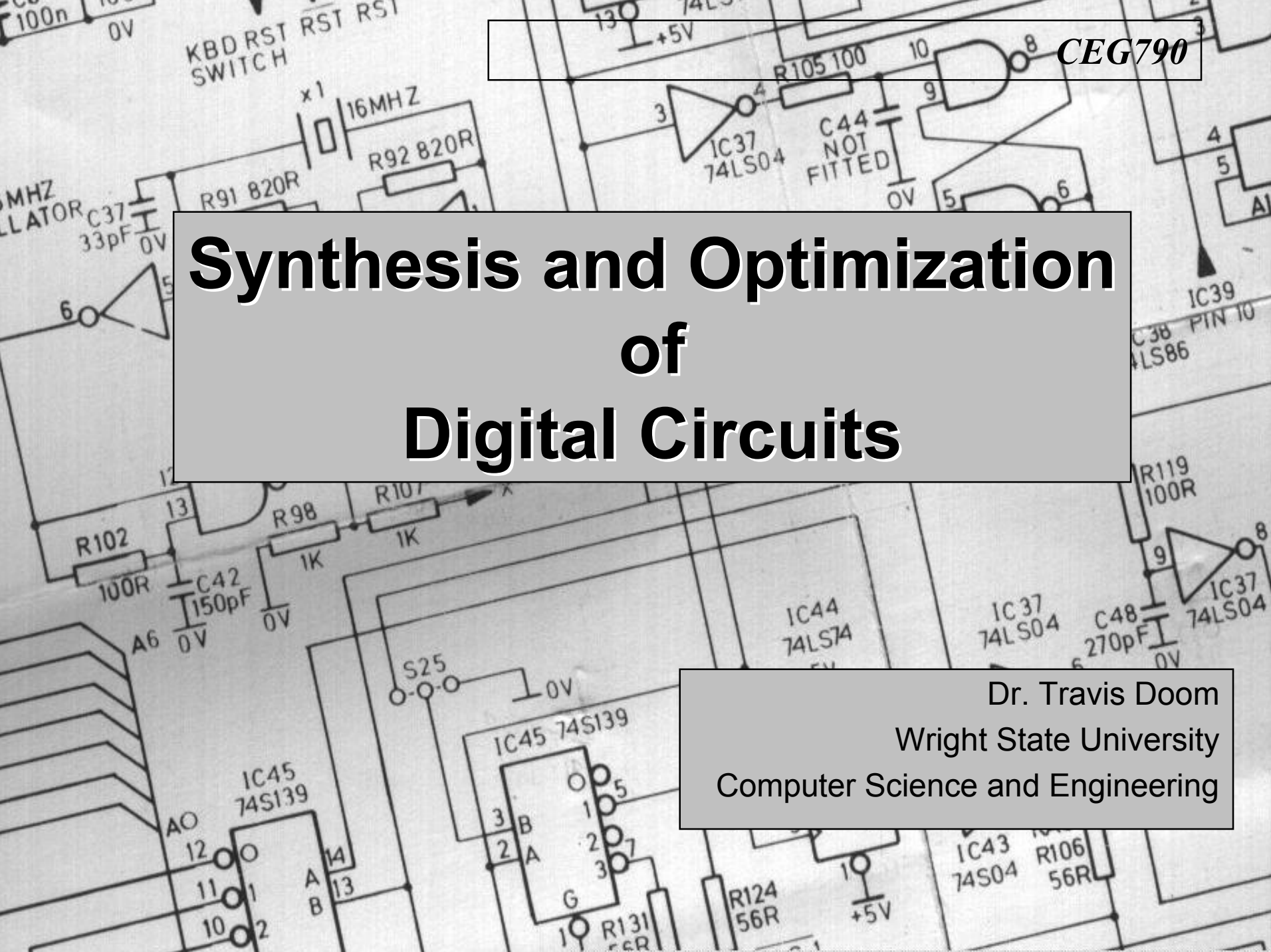


CEG790

Synthesis and Optimization of Digital Circuits

Dr. Travis Doom
Wright State University
Computer Science and Engineering

Outline

- Introduction
  - Microelectronics
  - Micro “economics”
  - What is “design”?
- Techniques for Digital Synthesis
  - Architectural-Level Synthesis
  - Logic-Level Synthesis
  - Geometric Synthesis
- Techniques for Formal Verification
  - Automated Theorem-Provers
  - Graph-based Algorithms
- Reengineering

Overview: Microelectronics

- What *is* a microelectronic component?
  - Devices which exploit the properties of semiconductor materials
  - Constructed by patterning a substrate and locally modifying its properties to shape “wires” and logical “devices”
  - Complex functions are “integrated” into one physical package
  - Fabrication is very complex
- Microelectronic components enable “smart” systems
  - Prevalent in modern systems
  - Failures are not taken well - most applications are “critical”

Overview: "Micro" Economics

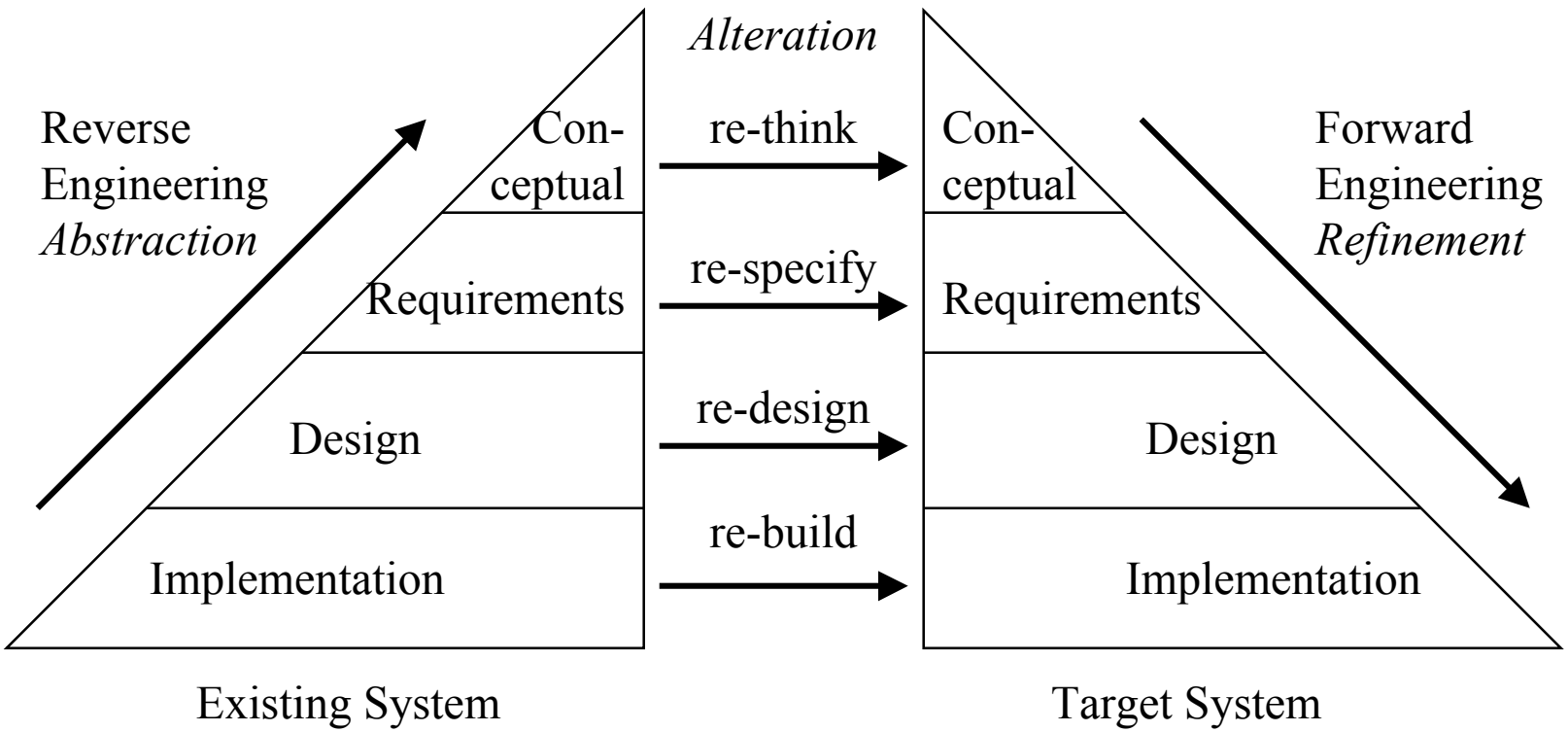
- IC technology has progressed tremendously over 40 yrs.
  - Moore's Law [SSI - '60, MSI - '70, VLSI - '90, ?? - '00]
- Costs have increased tremendously as well
  - Larger capital investment due to cost of refining precision
  - Larger scale increases effort to achieve zero-defect design
    - ICs are nearly impossible to repair
    - The design must be correct (and manufacturing defects limited)
  - Design and manufacturing costs must be recovered via sales
    - Few designs do enjoy a high volume of sales or long life
    - Many systems require specialized devices (ASICs) - few hold a significant market share individually
    - Improvement of technology causes immediate obsolescence

Overview: "Micro" Economics

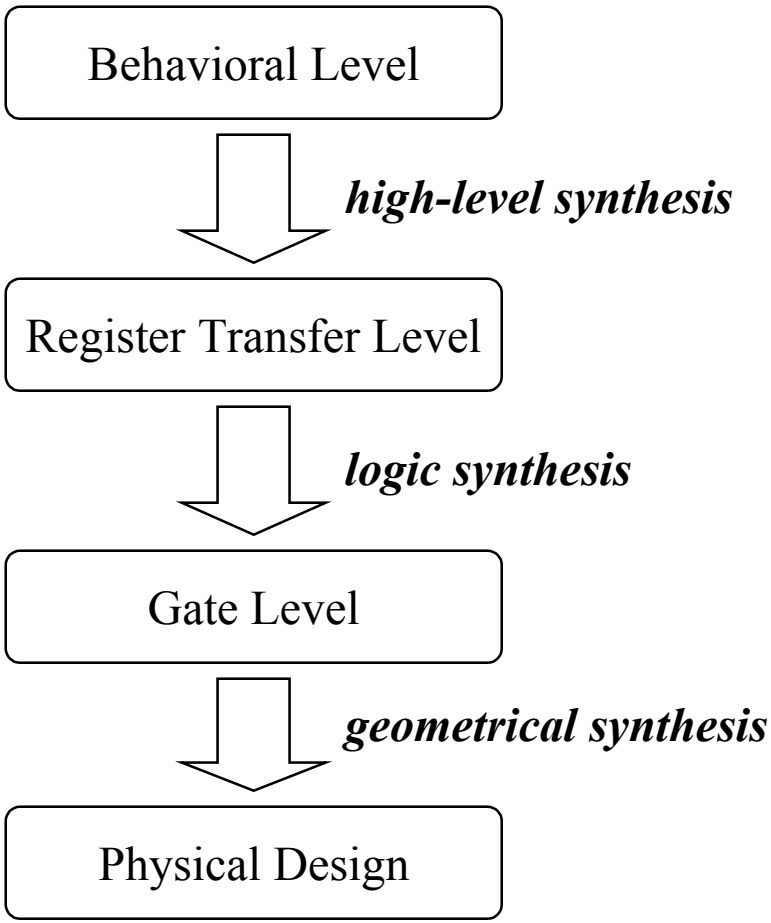
- How can costs be reduced and net profit increased?
  - Minimize Design (and test) time
    - Reduces both time-to-market and designers' salaries
  - Increase quality of design to increase fabrication yield and provide competitive performance
- Design automation techniques provide an effective means for designing economically viable products
  - Carrying out a full design w/o errors is increasingly difficult w/o systematic techniques to handle data
  - CAD techniques tend to focus on Digital Synchronous circuits as they represent the vast majority of circuits in the market

Overview: What is "Design"?

- General model for (Re-)Engineering (Byrne, 1992)



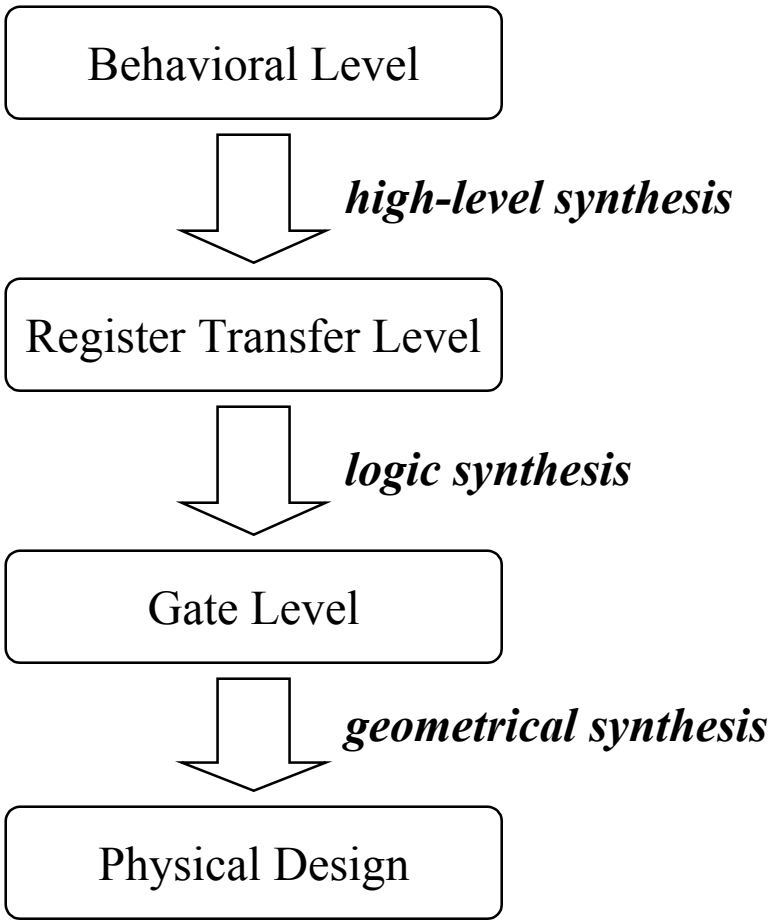
Automated Synthesis



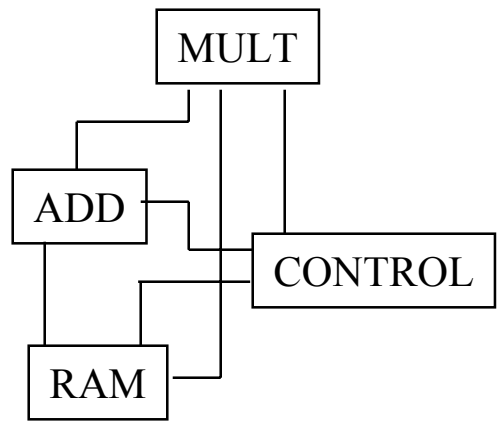
~ Requirements Spec.

~ Implementation Spec.

Automated Synthesis



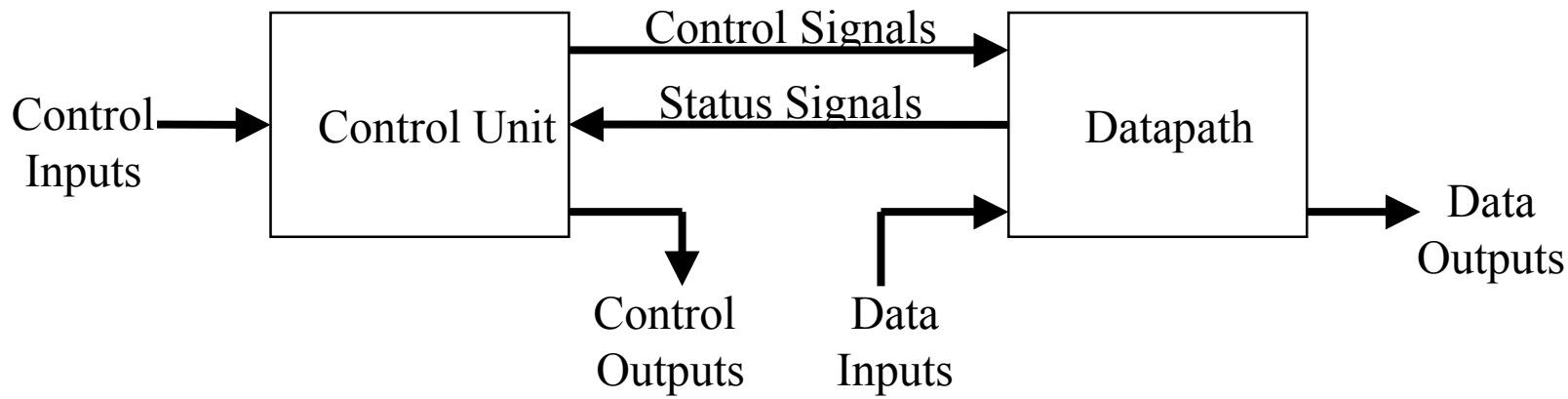

```
...
PC = PC + 1;
FETCH(PC);
DECODE(INST);
...
```



High-level Synthesis

- High-level (Architectural-level) synthesis deals with the transformation of an abstract model of behavior into a model consisting of standard functional units
  - Goal: to construct the macroscopic structure of a circuit
  - Input: an abstract model of behavior
    - Common Abstract Models: HDLs, State diagrams, ASM charts, Sequencing graphs or Control/Data-flow graphs.
  - Output: a structural view of the circuit, in particular of its datapath, and a logic-level specification of its control unit
    - often referred to as the register-transfer level or macro-module model

High-level Synthesis



- The data path is an interconnection of resources whose execution and I/O is determined by the control unit according to a schedule
  - **functional resources:**
    - Primitive resources: "stock" functions
    - Application-specific resources: requires model
  - **memory resources: registers** or **memory arrays** to store data
  - **interface resources:** steering logic circuits (e.g., muxes and buses) that send data to the appropriate destination at the appropriate time

High-level Synthesis

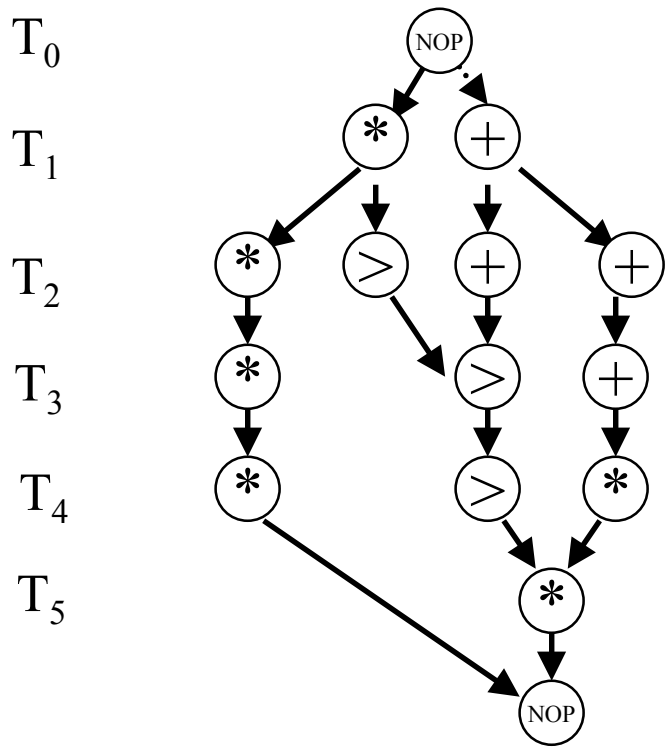
- Measuring cost
  - Evaluation Metrics: area, cycle-time (clock period), latency, and throughput (pipelines)
  - The objectives form a n-dimensional *design space*
    - *Architectural exploration* is the traversal of the design space to provide a spectrum of solutions for the designers selection
    - Generally only the resources are considered *(resource dominant)*
- The fundamental architectural synthesis problem
  - Explore the design space to minimize “cost” given:
    - A circuit model (behavioral)
    - A set of constraints (on cost)
    - A set of functional resources (characterized for area, delay, etc.)

Temporal Scheduling

- Automated approaches to the fundamental problem consist of two related constrained optimization problems: *Temporal Scheduling* and *Spatial Binding*

- Temporal Scheduling
  - Each architectural-level operation is reduced to resource operations and the time interval for the operation execution determined
    - A graph of resources must be created such that one path from start to end exists to perform each operation (in parallel)
    - The length of the path represents the operation latency
    - Constraints include maximum latency, bounds on the resource usage per type, etc.

Scheduled Sequencing Graph



Constraints:

Maximum Latency: 5

*: 3 max

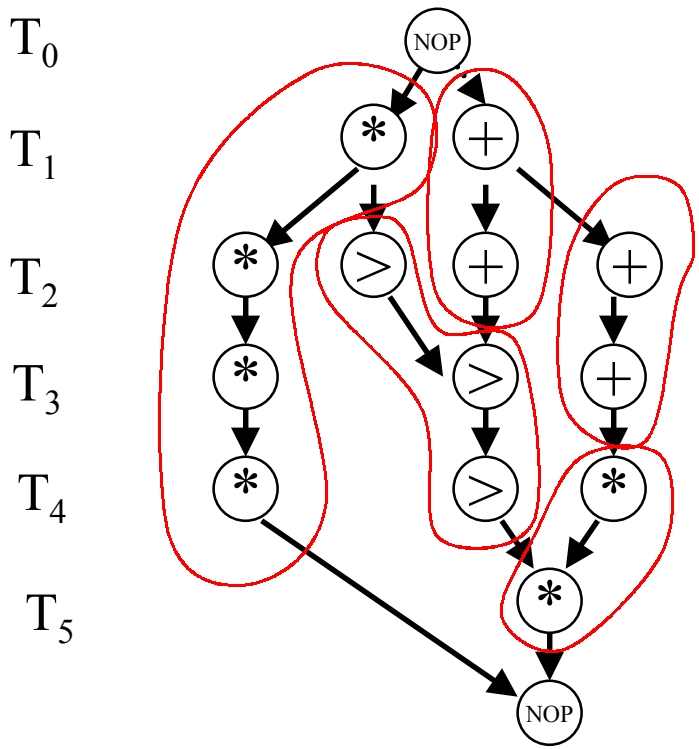
+: 3 max

>: 3 max

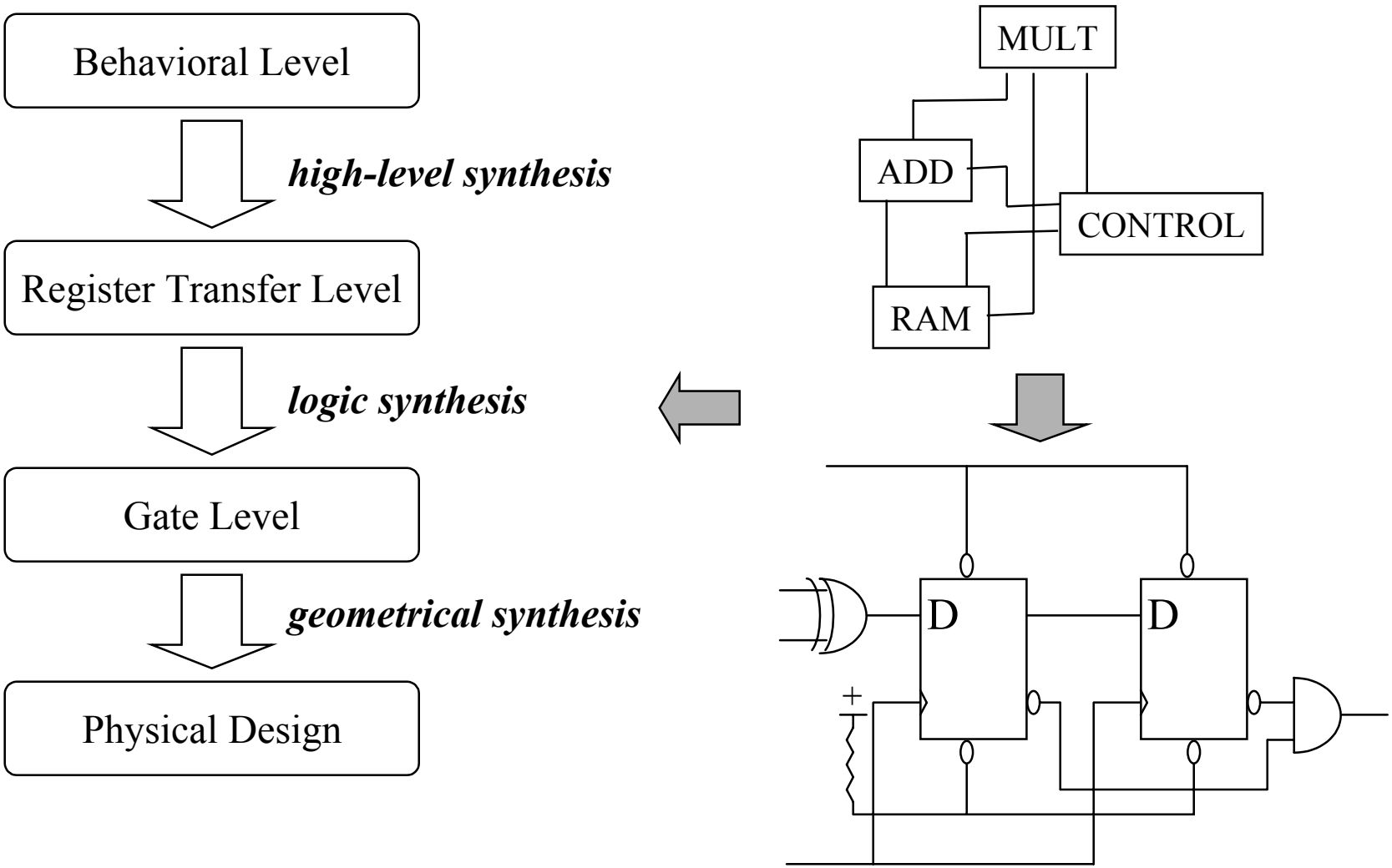
Spatial Binding

- Spatial Binding
  - Determining the detailed interconnections of the data path and the logic-level specifications of the control unit
    - The scheduled sequencing graph represents all necessary operations
    - Each resource may cover several operations (for example, an ALU covers addition, subtraction, comparison, etc.)
    - A simple case is dedicated resource binding - each operation is bound to one resource
    - In general, we wish to share a resources - we don't need to replicate beyond the maximum number of resources at any given temporal depth
    - In essence, this becomes a set-covering problem (NPC)
- Once a set of resources is identified, area and performance estimations can be calculated from model data

Scheduled Sequencing Graph w/Resource Binding



Automated Synthesis



Logic-level Synthesis

- Logic-level synthesis deals with the transformation of an macroscopic model to an interconnection of logic primitives
  - These primitives determine the microscopic (i.e., gate-level) structure of the circuit
- A basic approach is to replace “stock” modules with pre-optimized “stock” logic-level representations
  - Local optimizations of do not necessarily create an optimal result
  - Cost is increased (area, latency, power) / decreased (design time)
- Alternatively, the modules are partitioned into manageable designs (generally straightforward for the data path)
  - Several different types of finite-state machine decompositions exist

Logic-level Synthesis Tasks

- Optimize finite-state machines by state minimization
  - Stated as a bi-partite covering problem
- Select a state encoding (for control unit)
  - Heuristics include one-hot, almost one-hot, minimal-bit change, prioritized-adjacency, etc.
- Minimize the related combinational component
  - Two-level (SOP) minimization (Quine-McCluksey, Rudell-Sangiovanni, and McGeer algorithms)
  - Multi-level minimization (Decomposition is non-trivial)
- Cell-library binding
  - Implement minimized combinational functions as an interconnection of devices that are available in a given technology library (a bound network)

Automated Synthesis

Behavioral Level

high-level synthesis

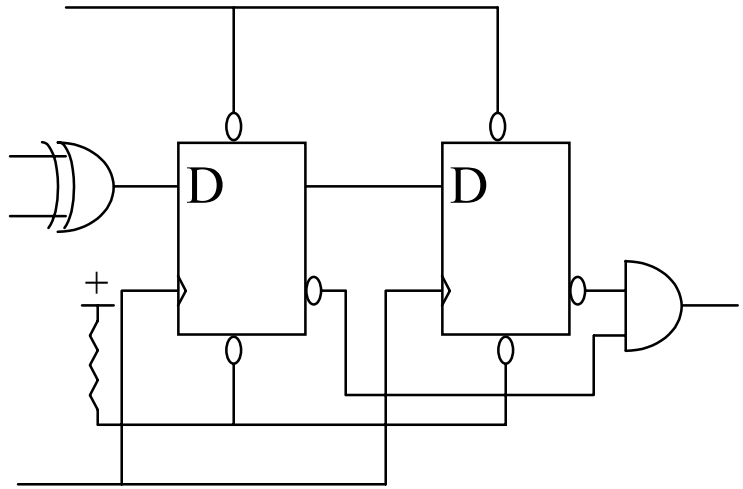
Register Transfer Level

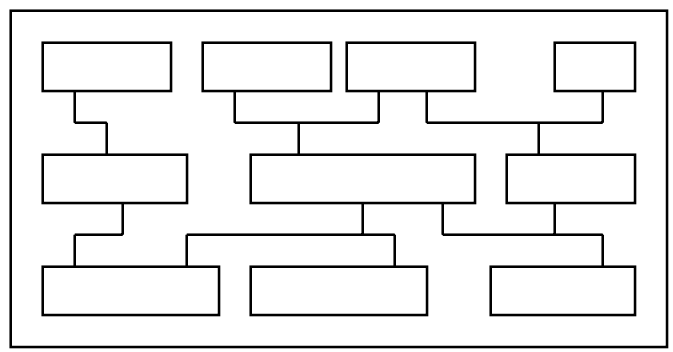
logic synthesis

Gate Level

geometrical synthesis

Physical Design





Geometrical-level Synthesis

- Geometrical-level synthesis (physical design) consists of creating a physical view at the geometric level
  - It entails the specification of all geometric patterns defining the physical layout of the chip, as well as their position

Validation and Verification

Behavioral Level

high-level synthesis

Register Transfer Level

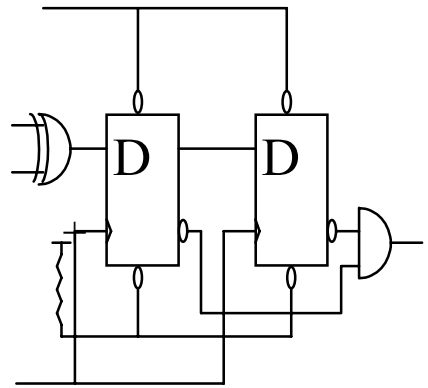
logic synthesis

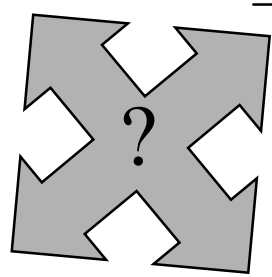
Gate Level

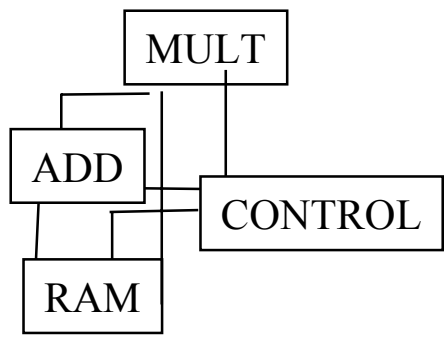
geometrical synthesis

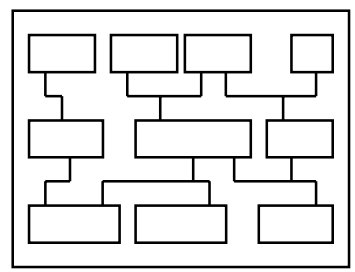
Physical Design

```
...
PC = PC + 1;
FETCH(PC);
DECODE(INST);
...
```

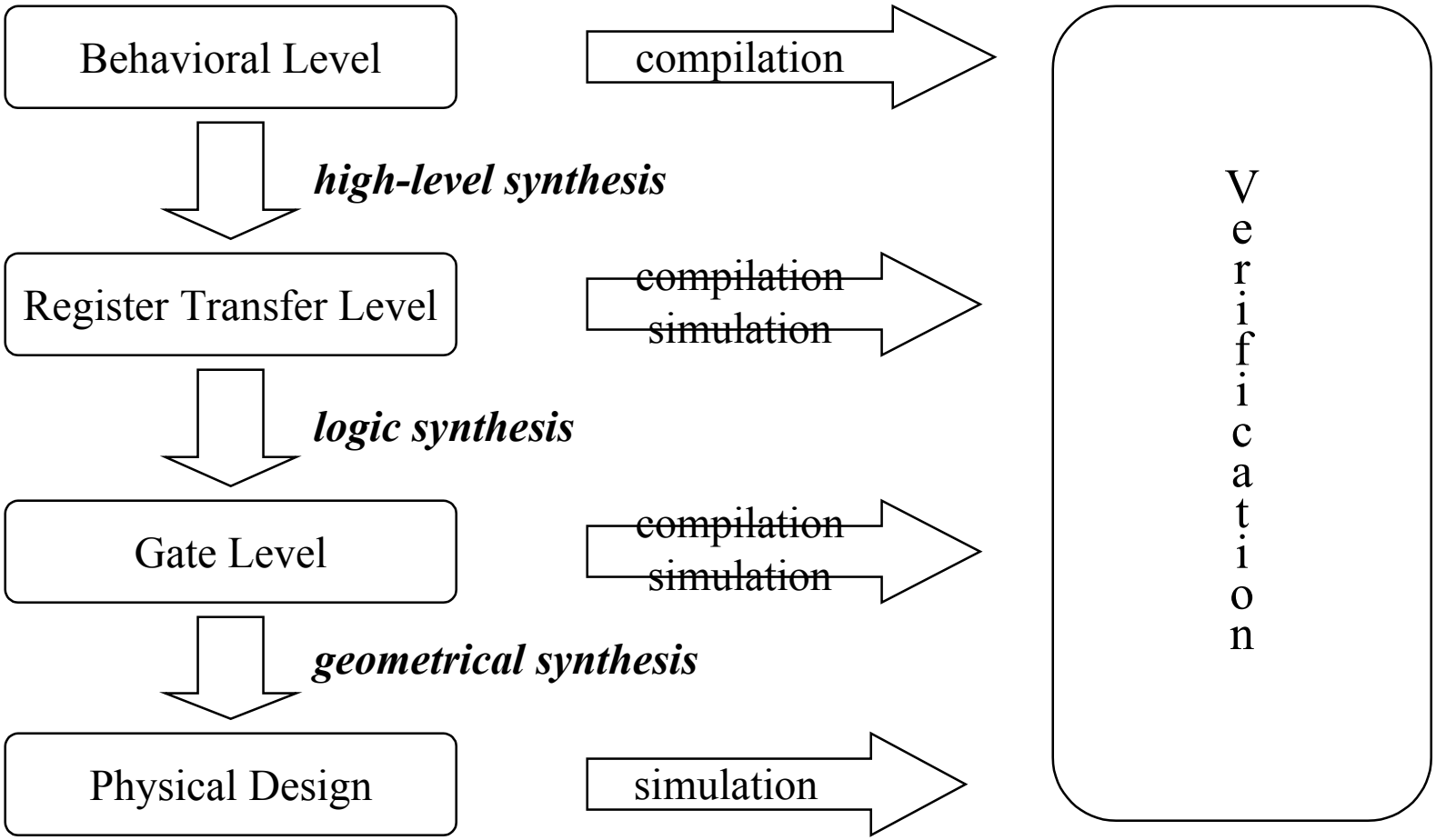








Validation and Verification



Validation and Verification

- Circuit validation consists of acquiring reasonable certainty that a circuit will function correctly
  - Assume no manufacturing fault is present
  - Can be performed via simulation or via verification
- Simulation (Traditional Validation)
  - Traditional verification consists of analyzing circuit variables (at different levels) over an interval of time
    - Unless exhaustive, simulation does not provide full coverage
- Formal Verification (Design Verification)
  - Verification methods mathematically prove or disprove the consistency between two models, or a model and some set of circuit model properties
    - Requires a suitable representation system
    - Proofs must be mechanizable

Formal Verification

- Property Testing (Testing via partial specification)
  - Safety properties: verify "bad things will never occur"
    - ex: for every path in the future, at every node on the path, if the Request signal is low, it remain lows until Acknowledge goes low
  - Liveness properties: verify "good things will occur"
    - ex: for every path in the future, if there has been a Request signal, then eventually there will be an Acknowledge signal in response to the request on at least one node on the path
- Popular FV approaches include:
  - Theorem Proving
  - Symbolic Model Checking
  - Recursive Learning
  - many graph-based approaches (BDDs, etc.)

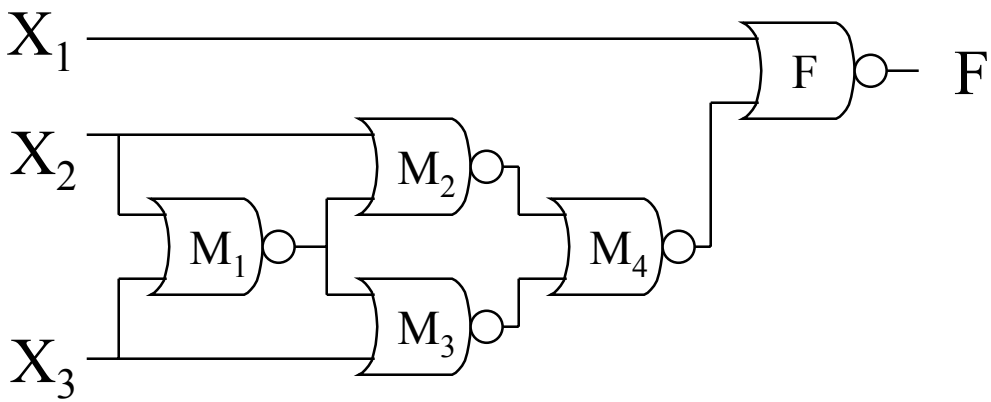
Automated Theorem-Provers

- Automated Theorem-Proving techniques require:
  - A representation of the model and/or properties as a series of formulas (axioms) in a High-Order Language
  - A finite collection of rules of inference
- By means of a rule of inference a new formula can be derived from a given finite set of formulas
  - A formal proof is a finite sequence of formulas, each member of which is either an axiom or the outcome of apply a rule of inference to previous members of the sequence
  - The last formal proof is the theorem
- Allows exhaustive (heuristic directed) search for proof
  - Theorem-provers presently require extensive user intervention

Equivalence Checking

- Equivalence checking (complete functional testing)
  - The function of an model is *equivalent* to the function of another if input, state, and output correspondences exist under which the functions are equivalent
  - Many techniques require factorial exploration of the input and state correspondence search space (in the worst-case)
  - FV *equivalence checking* of designs is known to be intractable
    - co-NP complete
    - heuristic techniques to achieve efficient performance

Representations of External Function



Schematic of simplecircuit

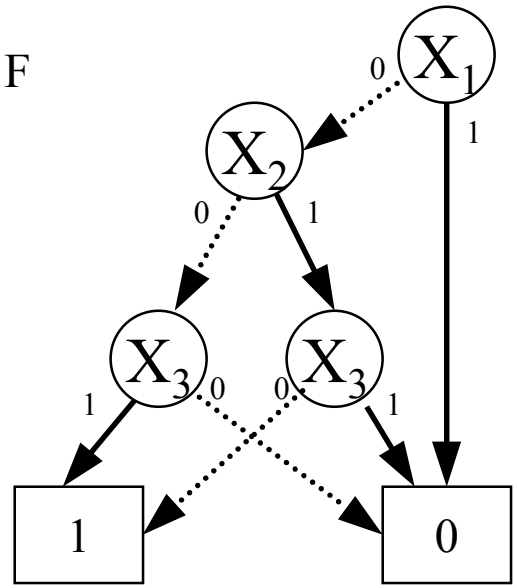
X_1	X_2	X_3	F
1	*d*	*d*	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

$$F \leftrightarrow \neg X_1 \wedge ((\neg X_2 \wedge X_3) \vee (X_2 \wedge \neg X_3))$$

```
ARCHITECTURE behavioral OF simplecircuit IS
BEGIN
  F <= (not X1) and ((not X2) and X3) or (X2 and (not X3))) after 10 ns;
END behavioral
```

BDD Representation

BDD for F

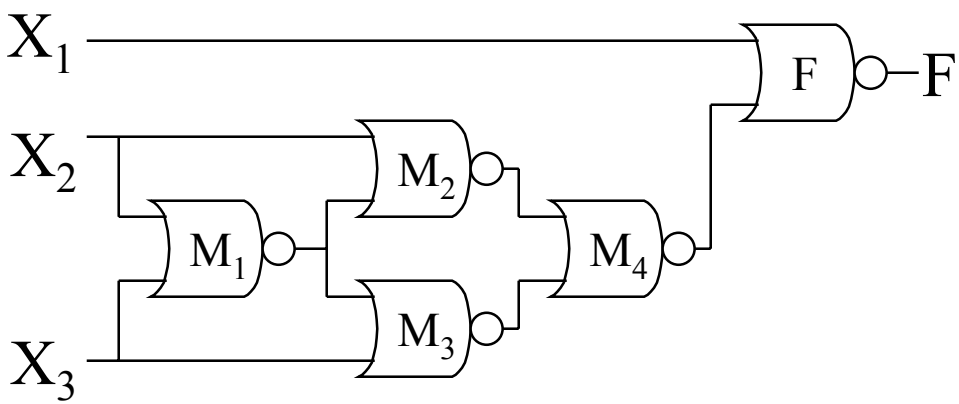


X_1	X_2	X_3	F
1	d	d	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

$F \leftrightarrow \neg X_1 \wedge ((\neg X_2 \wedge X_3) \vee (X_2 \wedge \neg X_3))$

```
ARCHITECTURE behavioral OF simplecircuit IS
BEGIN
  F <= (not X1) and ((not X2) and X3) or (X2 and (not X3))) after 10 ns;
END behavioral
```

BDD Representation

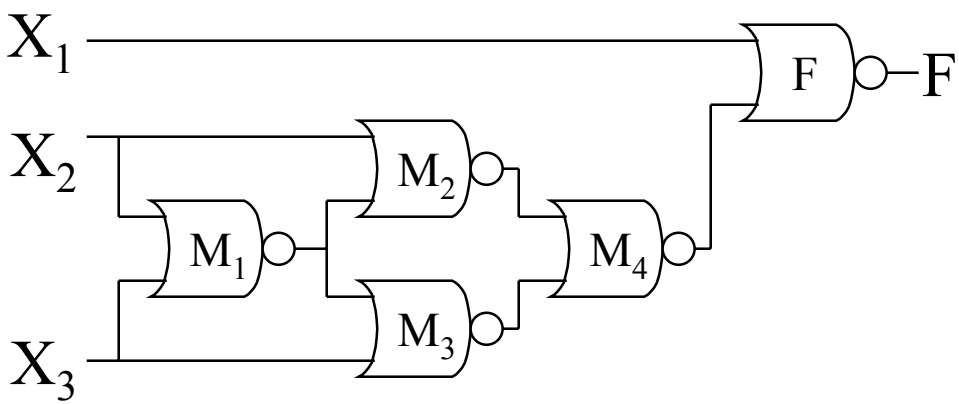


Schematic of simplecircuit

X_1	X_2	X_3	M_1	M_2	M_3	M_4	F
1	1	1	0	0	0	1	0
1	1	0	0	0	1	0	0
1	0	1	0	1	0	0	0
1	0	0	1	0	0	1	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	1
0	0	1	0	1	0	0	1
0	0	0	1	0	0	1	0

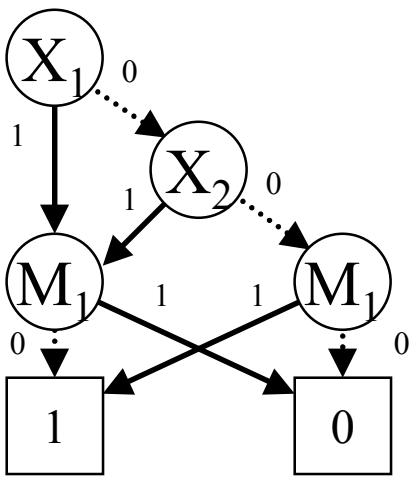
```
ARCHITECTURE structural OF simplecircuit IS
  SIGNAL M1, M2, M3, M4: bit ;
BEGIN
 gate0: nor2 PORT MAP ( O => M1, a=> X2, b => X3 );
 gate1: nor2 PORT MAP ( O => M2, a=> X2, b => M1 );
 gate2: nor2 PORT MAP ( O => M3, a=> M1, b => X3 );
 gate3: nor2 PORT MAP ( O => M4, a=> M1, b => M3 );
 gate4: nor2 PORT MAP ( O => F,    a=> X1,  b => M4 );
 output: probe PORTMAP ( F );
END structural
```

BDD Representation



Schematic of simplecircuit

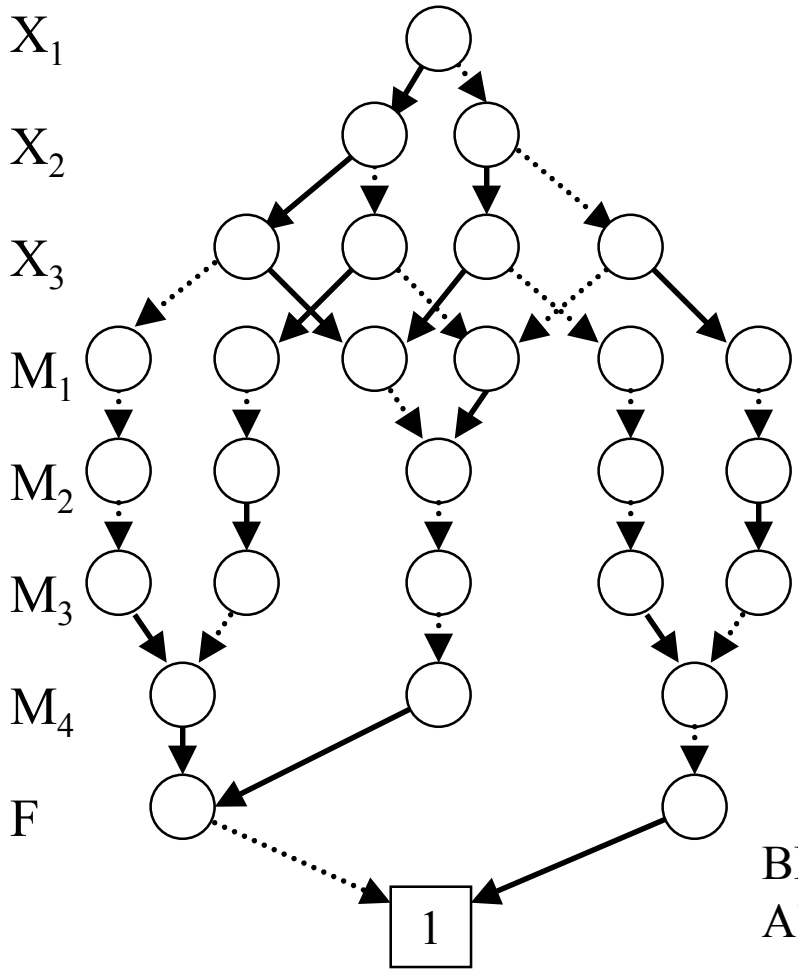
X_1	X_2	X_3	M_1	M_2	M_3	M_4	F
1	1	1	0	0	0	1	0
1	1	0	0	0	1	0	0
1	0	1	0	1	0	0	0
1	0	0	1	0	0	1	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	1
0	0	1	0	1	0	0	1
0	0	0	1	0	0	1	0



BDD representing the characteristic function of NOR gate M_1:
$(M_1 \leftrightarrow \neg(X_2 \wedge X_3))$

X_2	X_3	M_1	BDD
1	d	0	1
0	1	0	1
0	0	1	1
otherwise			0

BDD Representation



X_1	X_2	X_3	M_1	M_2	M_3	M_4	F
1	1	1	0	0	0	1	0
1	1	0	0	0	1	0	0
1	0	1	0	1	0	0	0
1	0	0	1	0	0	1	0
0	1	1	0	0	1	0	0
0	1	0	0	0	1	0	1
0	0	1	0	1	0	0	1
0	0	0	1	0	0	1	0

$$(M_1 \leftrightarrow \neg(X_2 \wedge X_3)) \wedge$$
$$(M_2 \leftrightarrow \neg(X_2 \wedge M_1)) \wedge$$
$$(M_3 \leftrightarrow \neg(X_3 \wedge M_1)) \wedge$$
$$(M_4 \leftrightarrow \neg(M_2 \wedge M_3)) \wedge$$
$$(F \leftrightarrow \neg(X_1 \wedge M_4))$$

BDD representing structural relationships
All edges not shown lead to the 0-terminal

Binary Decision Diagrams

- BDDs have been shown to be efficient under a mild assumption on the *order* of the variables
- OBDDS have more practical applications than most graphical representations:
  - OBDDs can be transformed into canonical forms to uniquely characterize their function
  - Operations on OBDDs can be done in O(|G|) time
- OBDDs are the basis for many new FV approaches
  - Unfortunately, the size of the BDD is based upon the variable ordering and can have exponential size in the worst case
  - FV problems are far from solved!

Modern Design Approaches

- The most common contemporary design approaches are:
  - Custom Approach: Designed primarily by hand (so to speak)
    - Full Custom Vs. Standard Cell - Using standard cell designs (same height, variable width) and routing channels simplifies design process
    - Highest Density, Highest Manufacturing Cost
  - Semi-custom Approach: Design process focuses on CAD tools
    - **Gate array**: a partially prefabricated IC that incorporates a large number of identical devices (ex: 3-input NAND or NOR gates) that are laid out in a regular two-dimensional array
    - **Technology mapping**: The process of designing a logic function as a network of the available devices (a.k.a cell-library binding)
    - Lower Density (110-125% devices of equivalent custom design)
    - Inexpensive: Requires only metal deposition (to define device interconnections), economies of scale

Modern Design Approaches

- The most common contemporary design approaches are:
  - PLD Approach: Often dependent on CAD tools
    - ex: Field Programmable Gate Arrays (FPGAs)
    - VLSI modules that can be programmed to implement a digital system consisting of tens of thousands of gates.
    - LSI PLDs implement two-level combinational and sequential networks
    - FPGAs allow the realization of “reprogrammable” multilevel networks and complex systems on a single chip
    - Low cost
    - May produce slower network
    - May require a larger silicon area

Reengineering

Behavioral Level

high-level synthesis

Register Transfer Level

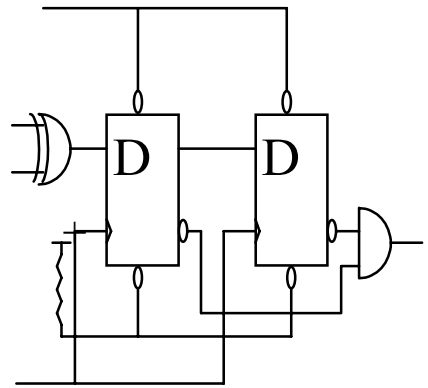
logic synthesis

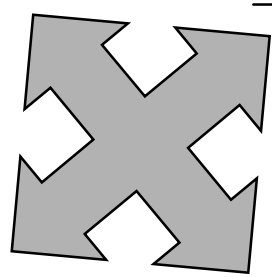
Gate Level

geometrical synthesis

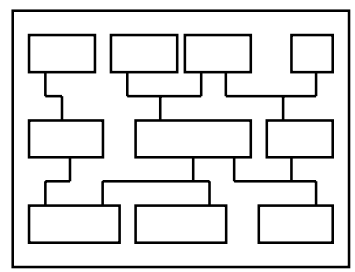
Physical Design

?





?



CAD tools for Design Recovery