VLSI CAD Overview: Design, Flows, Algorithms and Tools

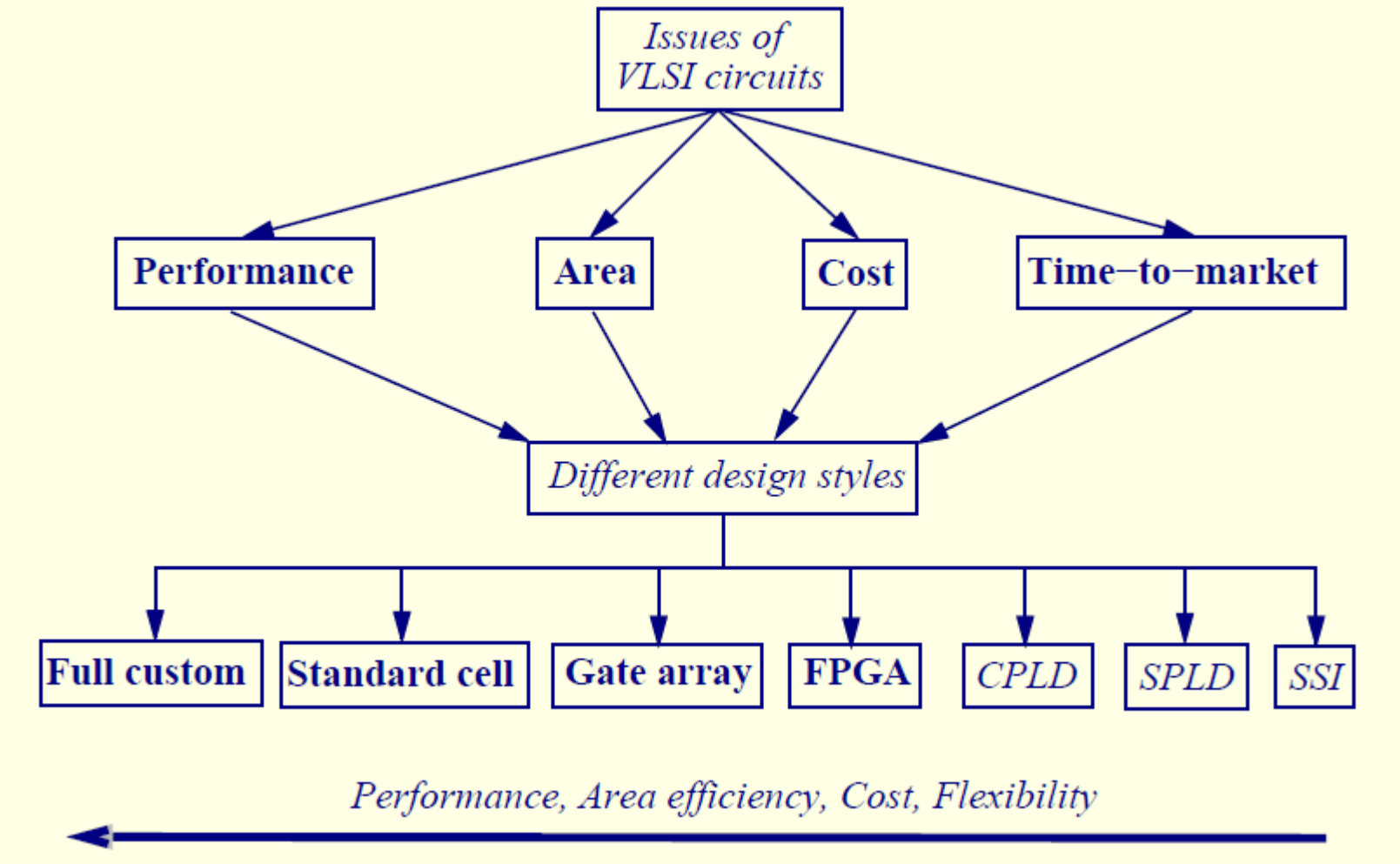
Konstantin Moiseev – Intel Corp. & Technion

Shmuel Wimer – Bar Ilan Univ. & Technion

Compiled from various presentation from the web.

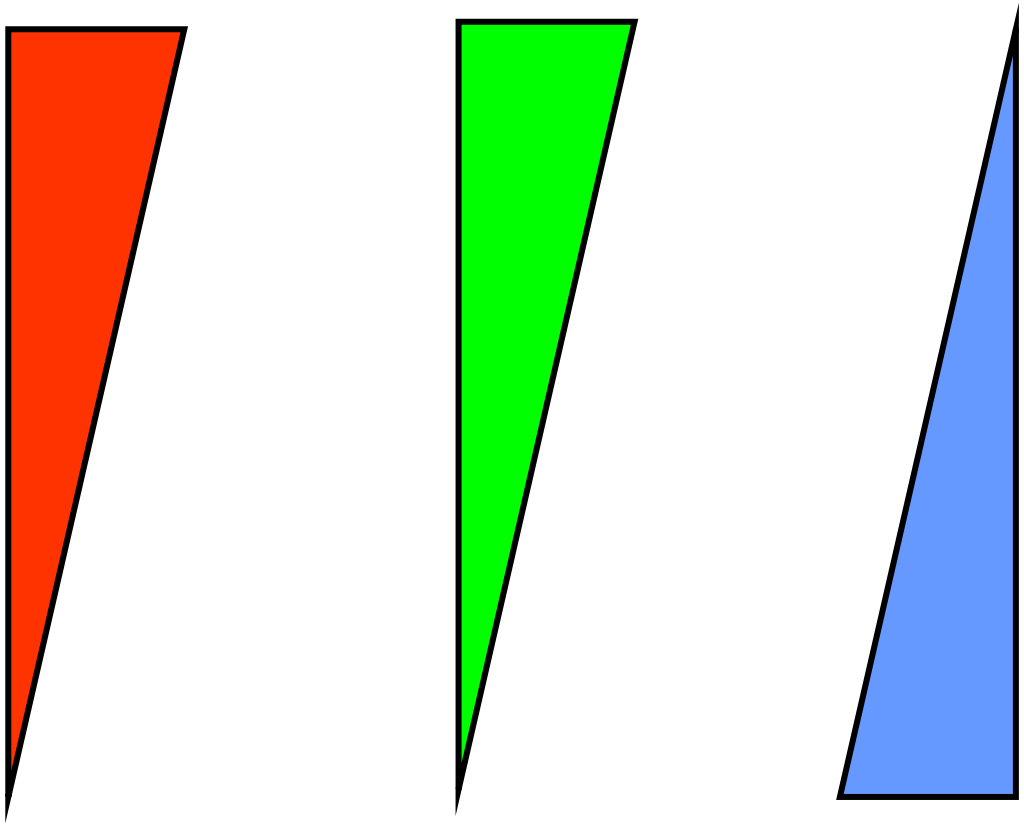
Credits:
David Pan – Univ. of Texas Austin
Maciej Ciesielski - UMASS
Andrew Kahng – UCSD
Hai Zhou – Northwestern Univ.
Kia Bazargan – Univ. of Minnesota
Avinoam Kolodny - Technion

Design Factors and Styles



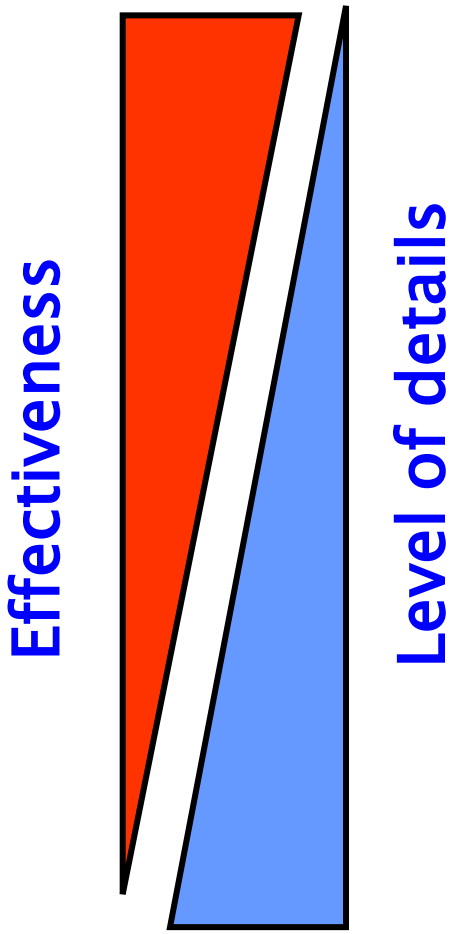
The Big Picture: IC Design Methods

Design Methods	Cost / Development Time	Quality	# Companies involved
Full Custom			
Standard Cell Library Design			
ASIC – Standard Cell Design			
RTL-Level Design			

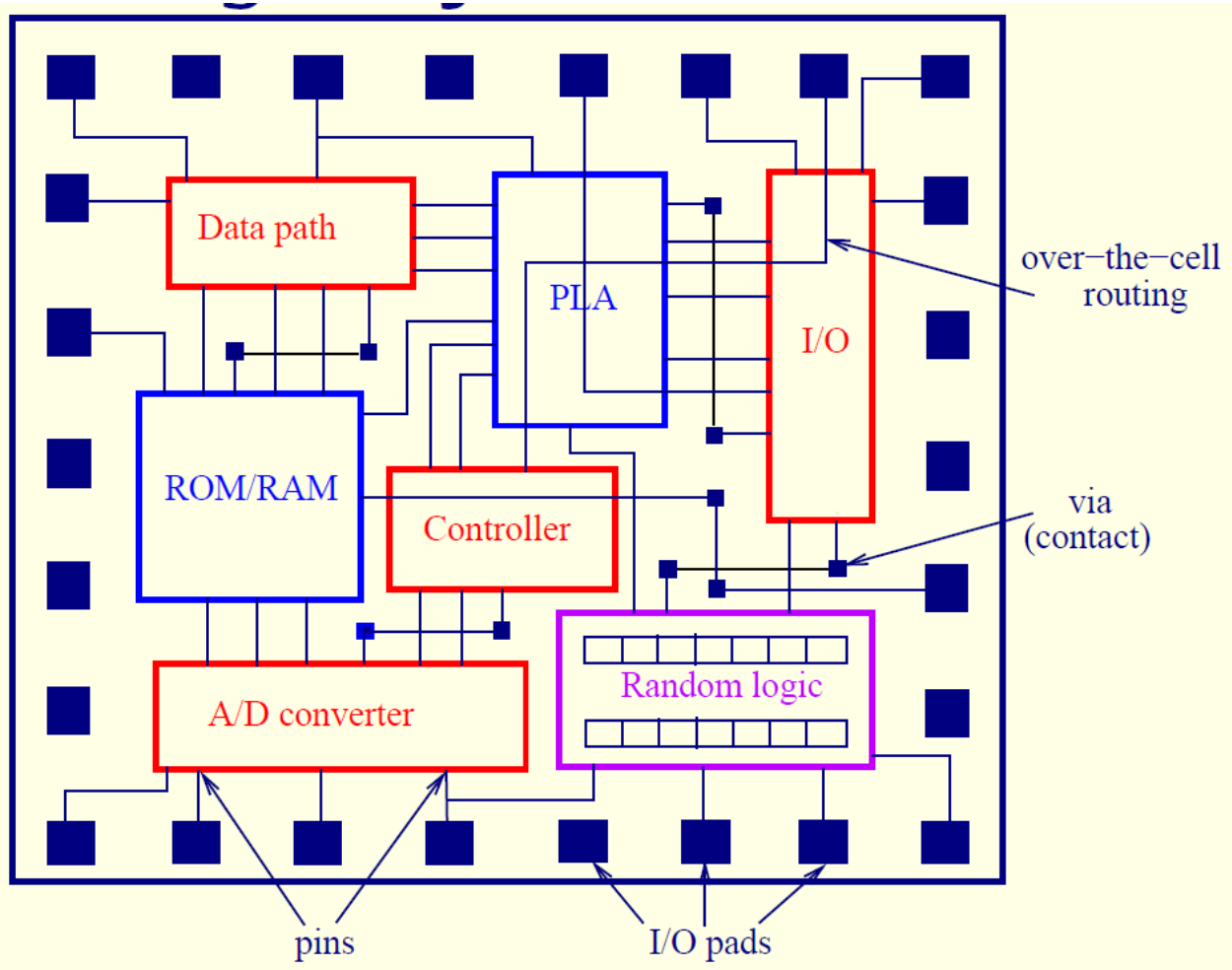


Optimization: Levels of Abstraction

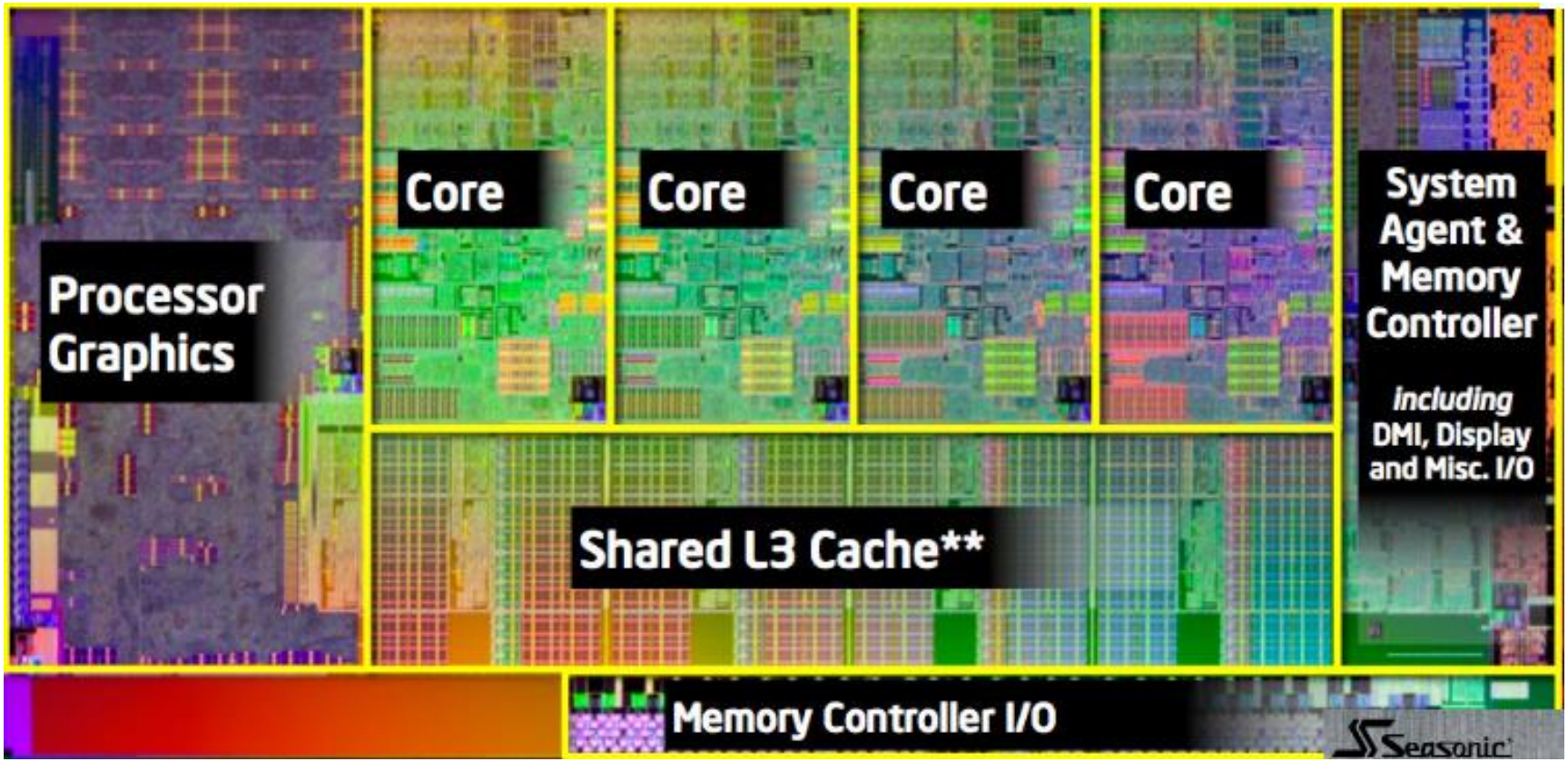
- Algorithmic
  - Encoding data, computation scheduling, balancing delays of components, etc.
- Gate-level
  - Reduce fan-out, capacitance
  - Gate duplication, buffer insertion
- Layout / Physical-Design
  - Move cells/gates around to shorten wires on critical paths
  - Abut rows to share power / ground lines



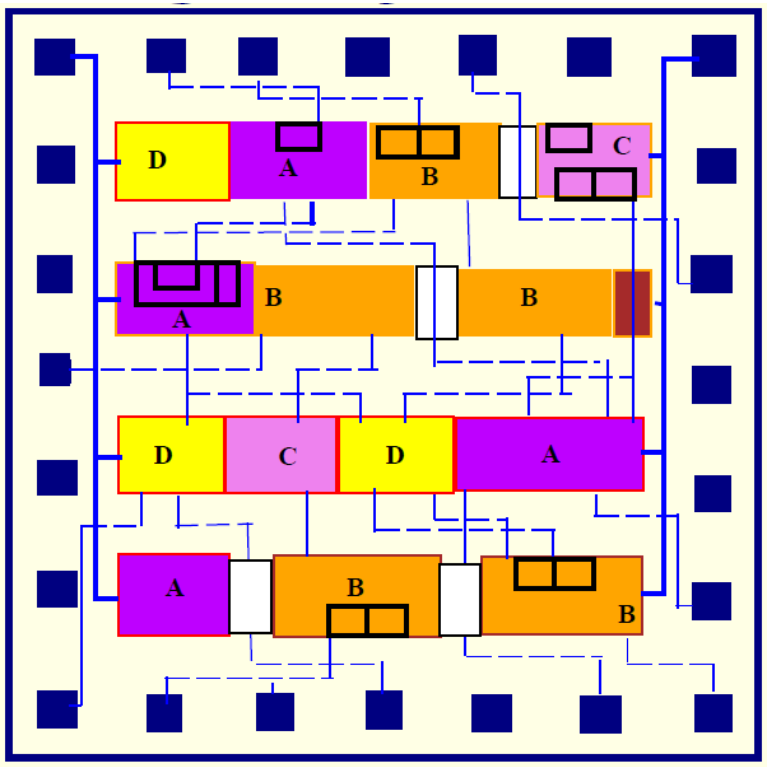
Full Custom

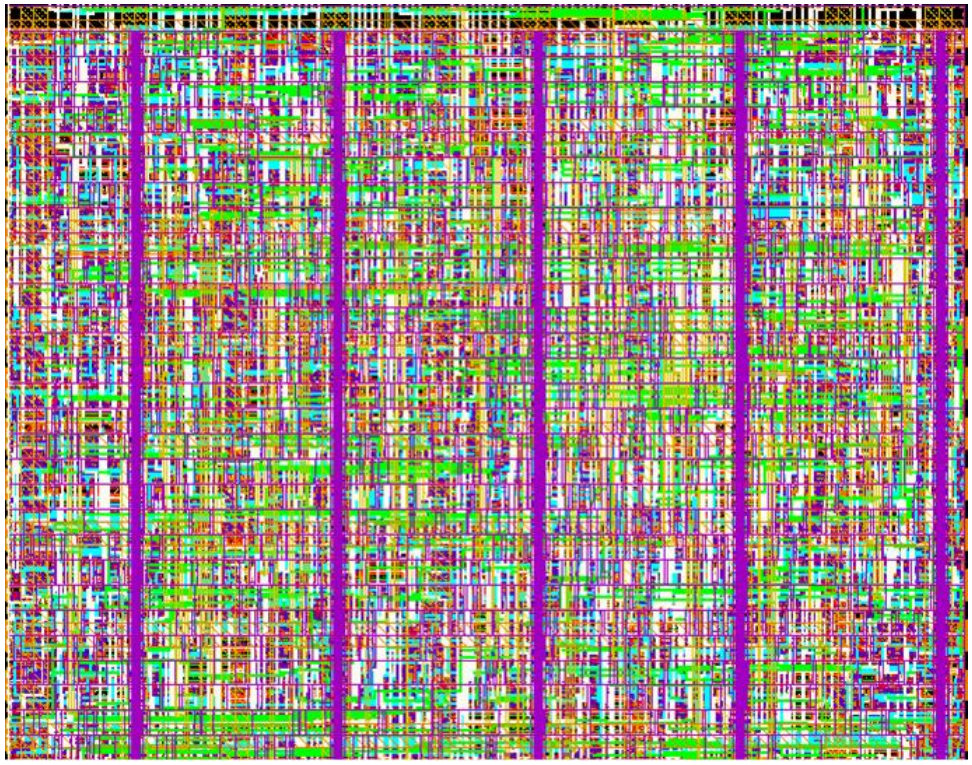

Data path
PLA
I/O
over−the−cell routing
ROM/RAM
Controller
via (contact)
A/D converter
Random logic
pins
I/O pads

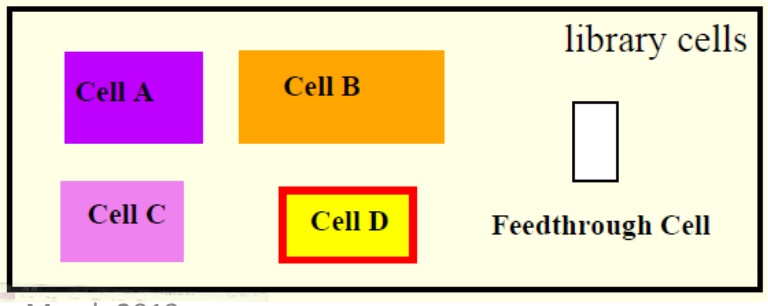
Full Custom


Processor Graphics
Core
Core
Core
Core
System Agent & Memory Controller
including DMI, Display and Misc. I/O
Shared L3 Cache**
Memory Controller I/O
Seasonic

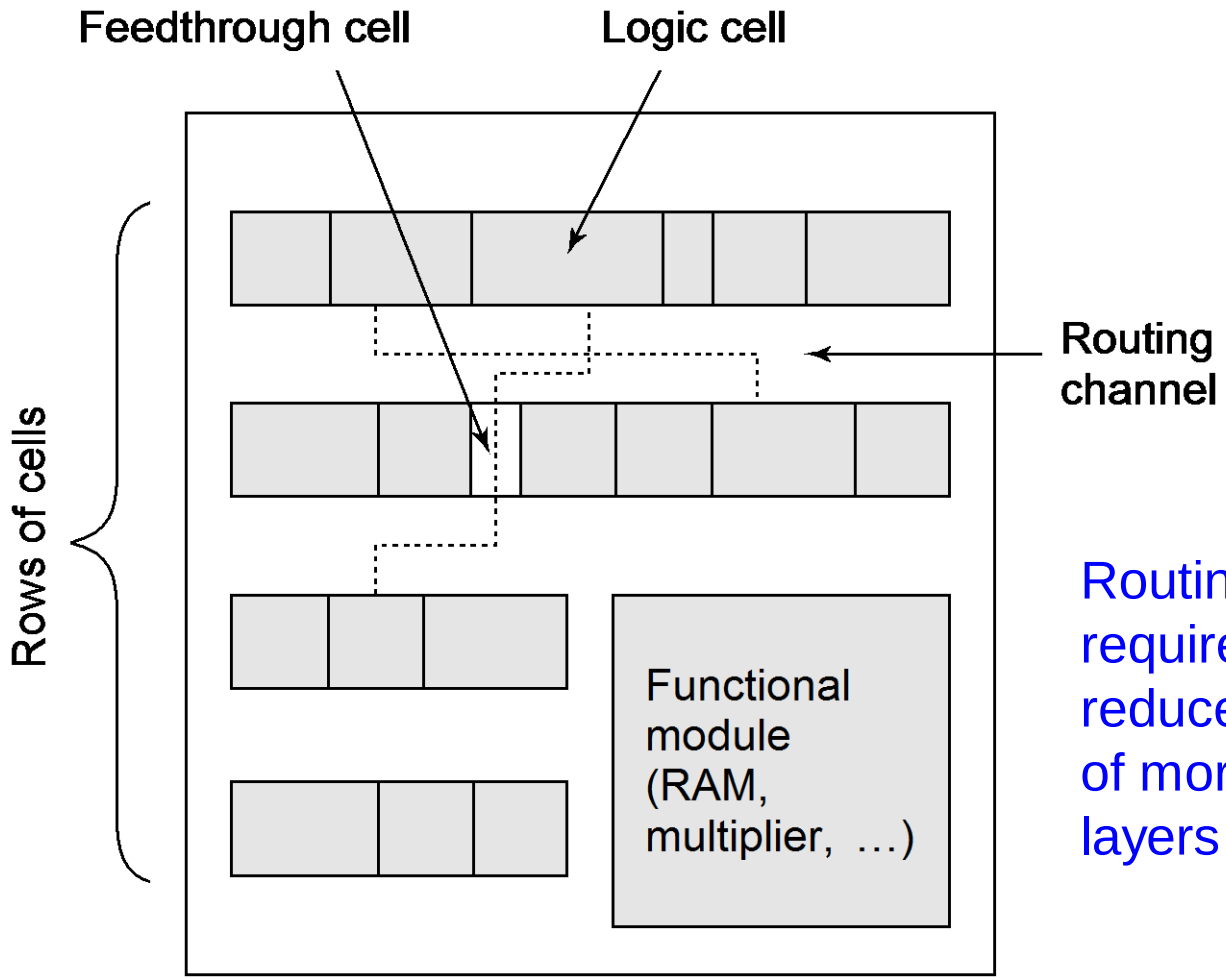
Standard Cell (Semi Custom)







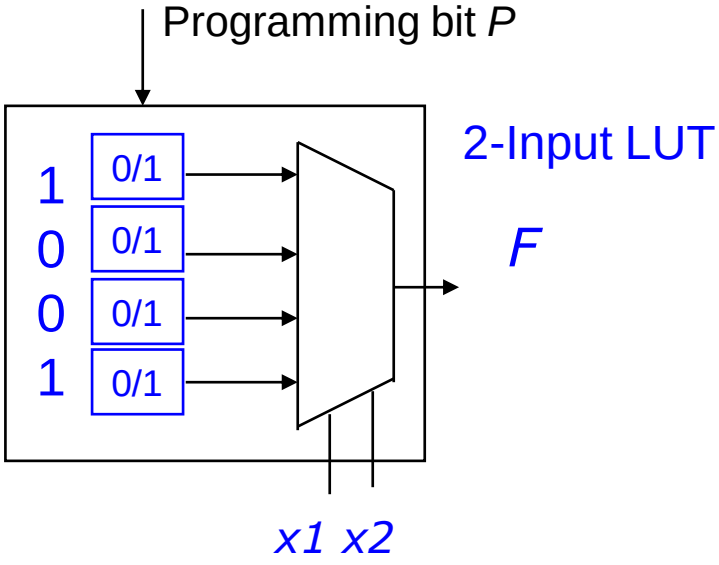
Cell-Based Design (Standard Cells)



Routing channel requirements are reduced by presence of more interconnect layers

FPGA: Lookup Table (LUT)

- Look-up Table
  - Truth table implemented in hardware
  - Can implement arbitrary function with fixed number of inputs (typically 4-5) by programming the storage bits (customizing the truth table)

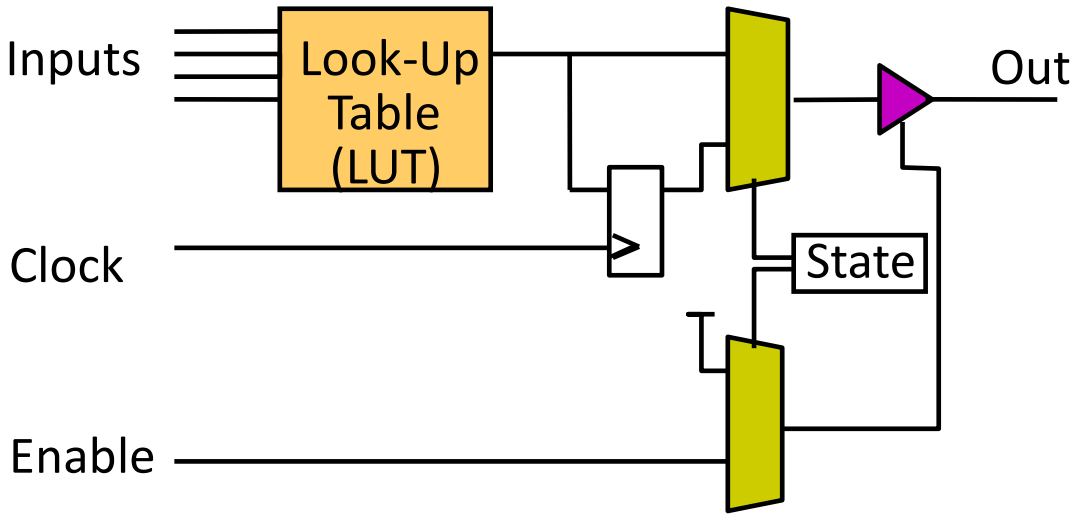


$F = x_1'x_2' + x_1x_2$

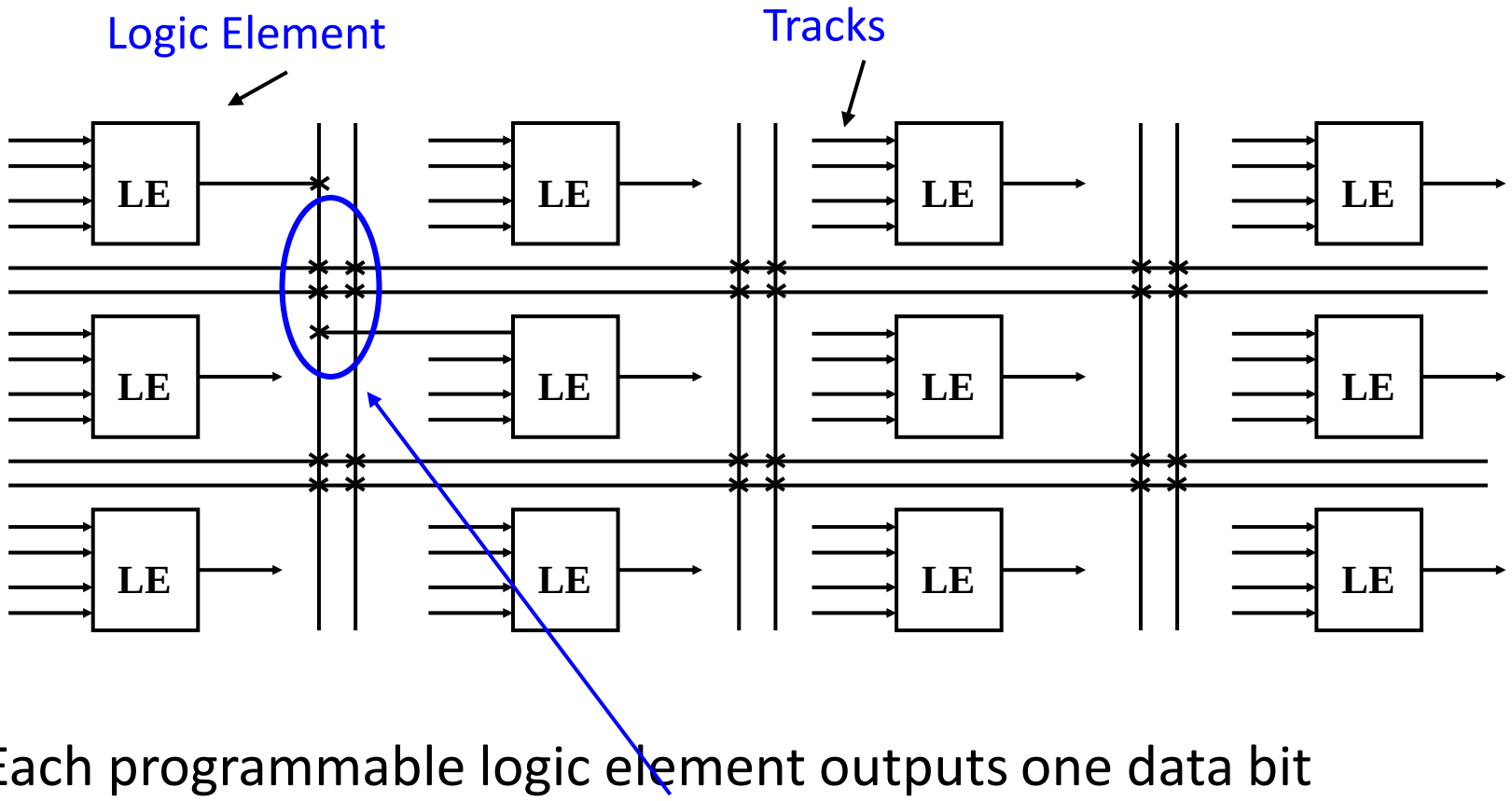
x1	x2	F
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

FPGA: Logic Element

- Logic Element: the basic programmable element of FPGA
  - Contains LUT
- Programming is a domain of specialized technology mapping onto device specific structure



FPGA: Architecture

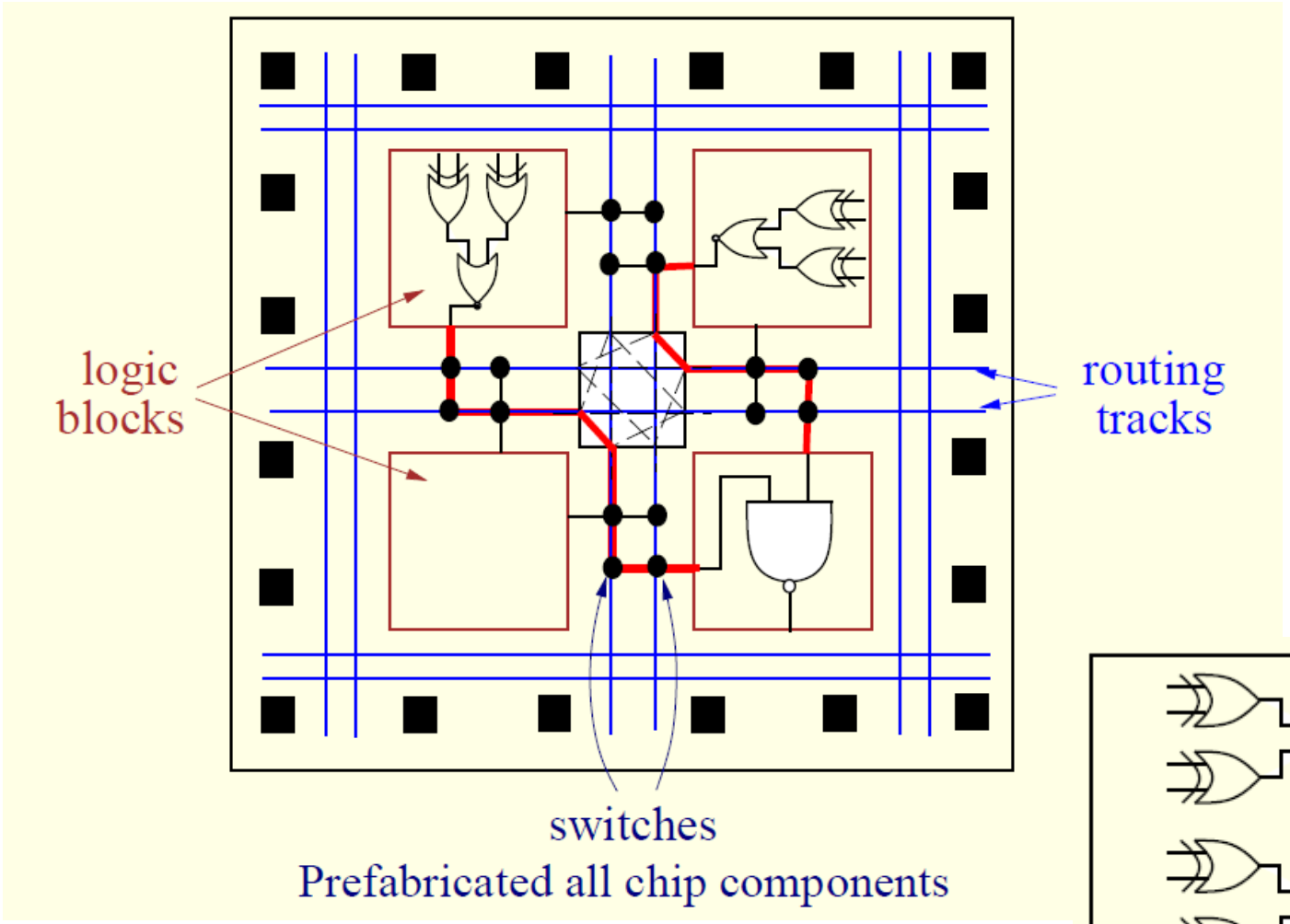


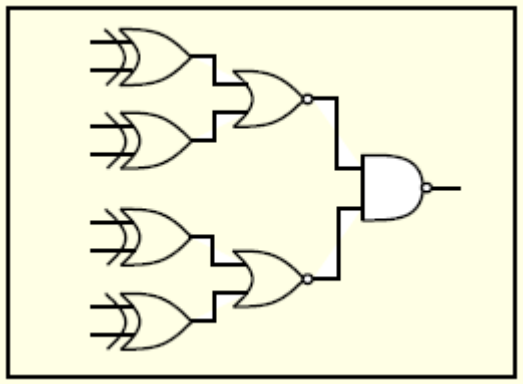
Each programmable logic element outputs one data bit

Interconnects are also programmable

A domain of **physical synthesis** (place and route)

FPGA: Architecture





Comparison of Design Styles

	style			
	full-custom	standard cell	gate array	FPGA
cell size	variable	fixed height *	fixed	fixed
cell type	variable	variable	fixed	programmable
cell placement	variable	in row	fixed	fixed
interconnections	variable	variable	variable	programmable

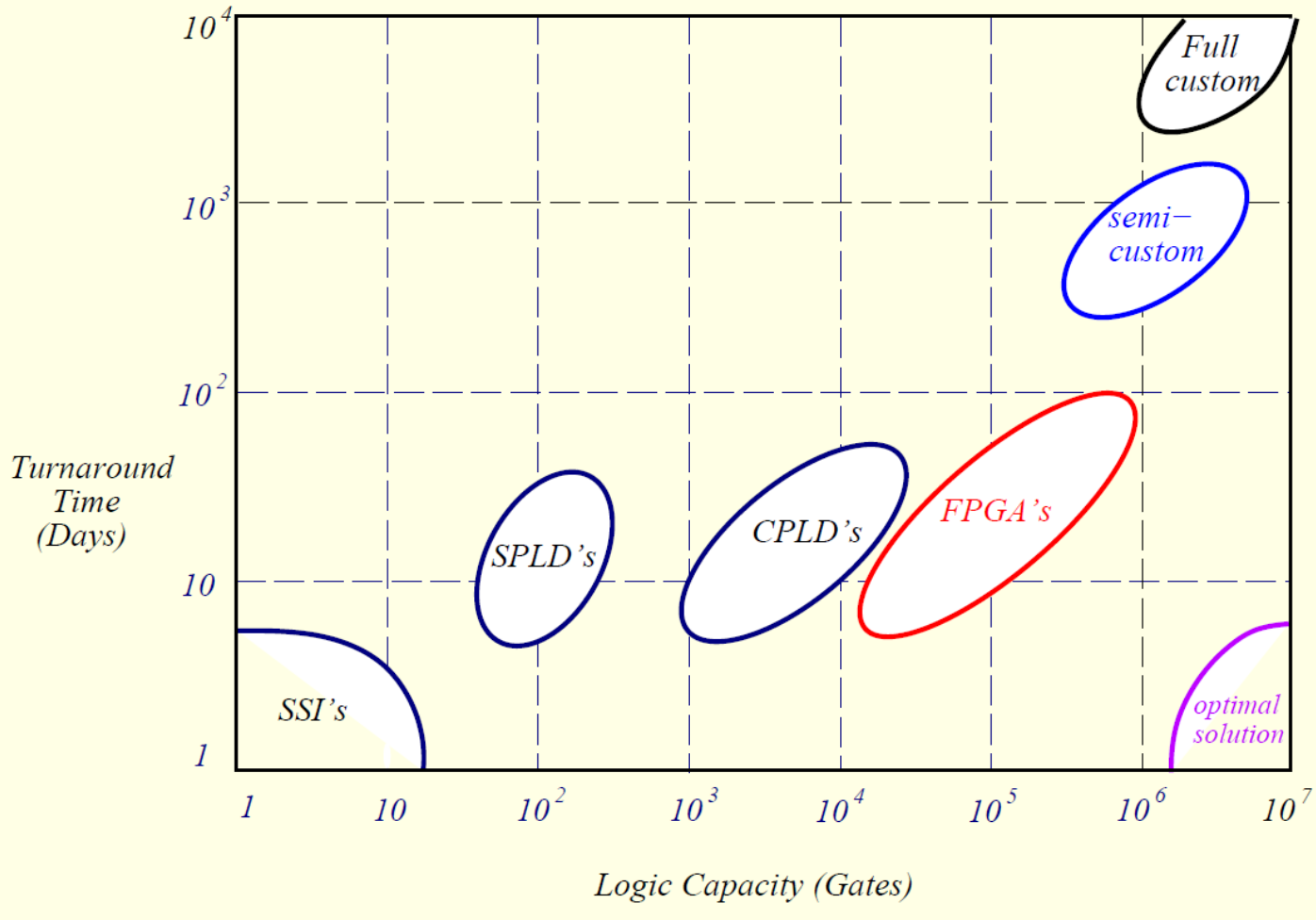
Comparison of Design Styles

	style			
	full-custom	standard cell	gate array	FPGA
Area	compact	compact to moderate	moderate	large
Performance	high	high to moderate	moderate	low
Fabrication layers	ALL	ALL	routing layers	none

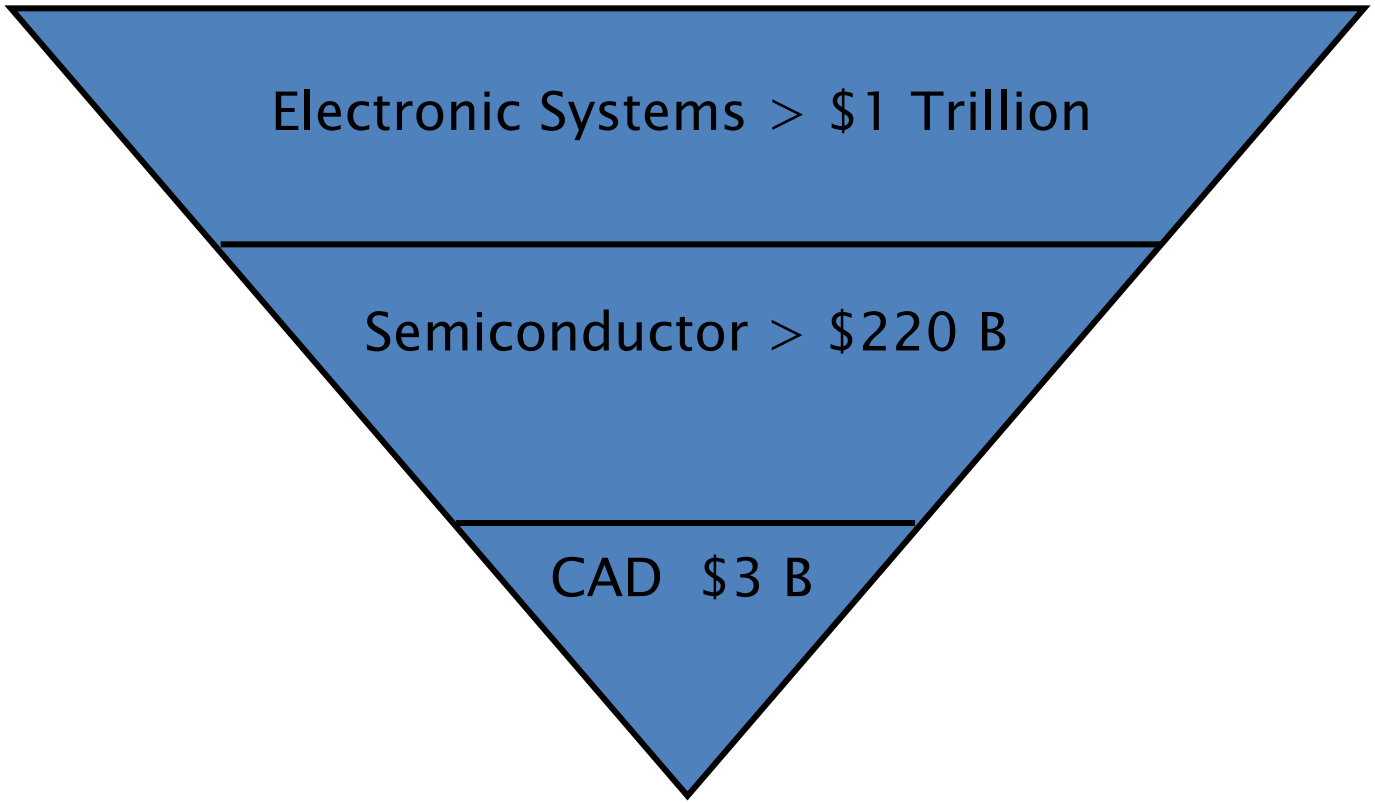
Comparison of Design Styles

	Full custom	Standard cell	Gate array	FPGA
Fabrication time	− − −	− −	+	+++
Packing density	+++	++	+	− −
Unit cost in large quantity	+++	++	+	− −
Unit cost in small quantity	− − −	− −	+	+++
Easy design and simulation	− − −	− −	−	++
Easy design change	− − −	− −	−	++
Accuracy of timing simulation	−	−	−	+
Chip speed	+++	++	+	−

Design Styles Tradeoffs

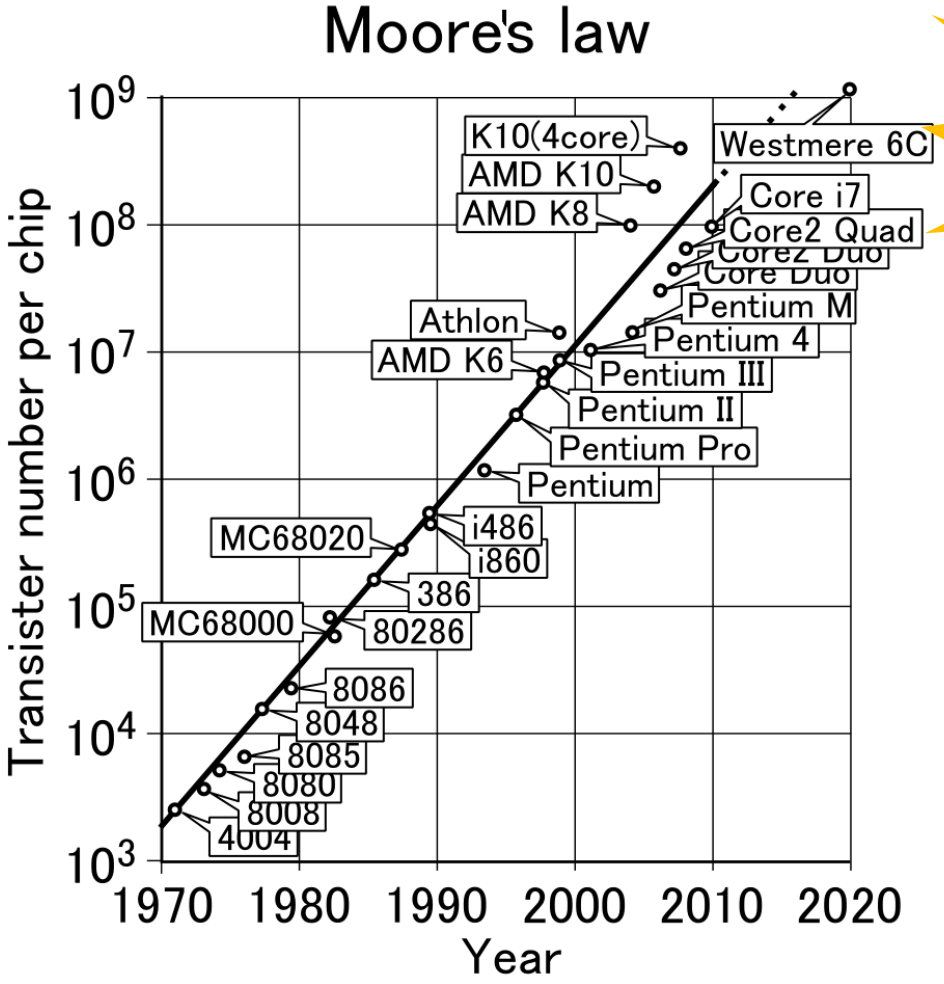


The Inverted Pyramid (~2000)



Moore’s law

- Moore’s law – exponential growth in complexity

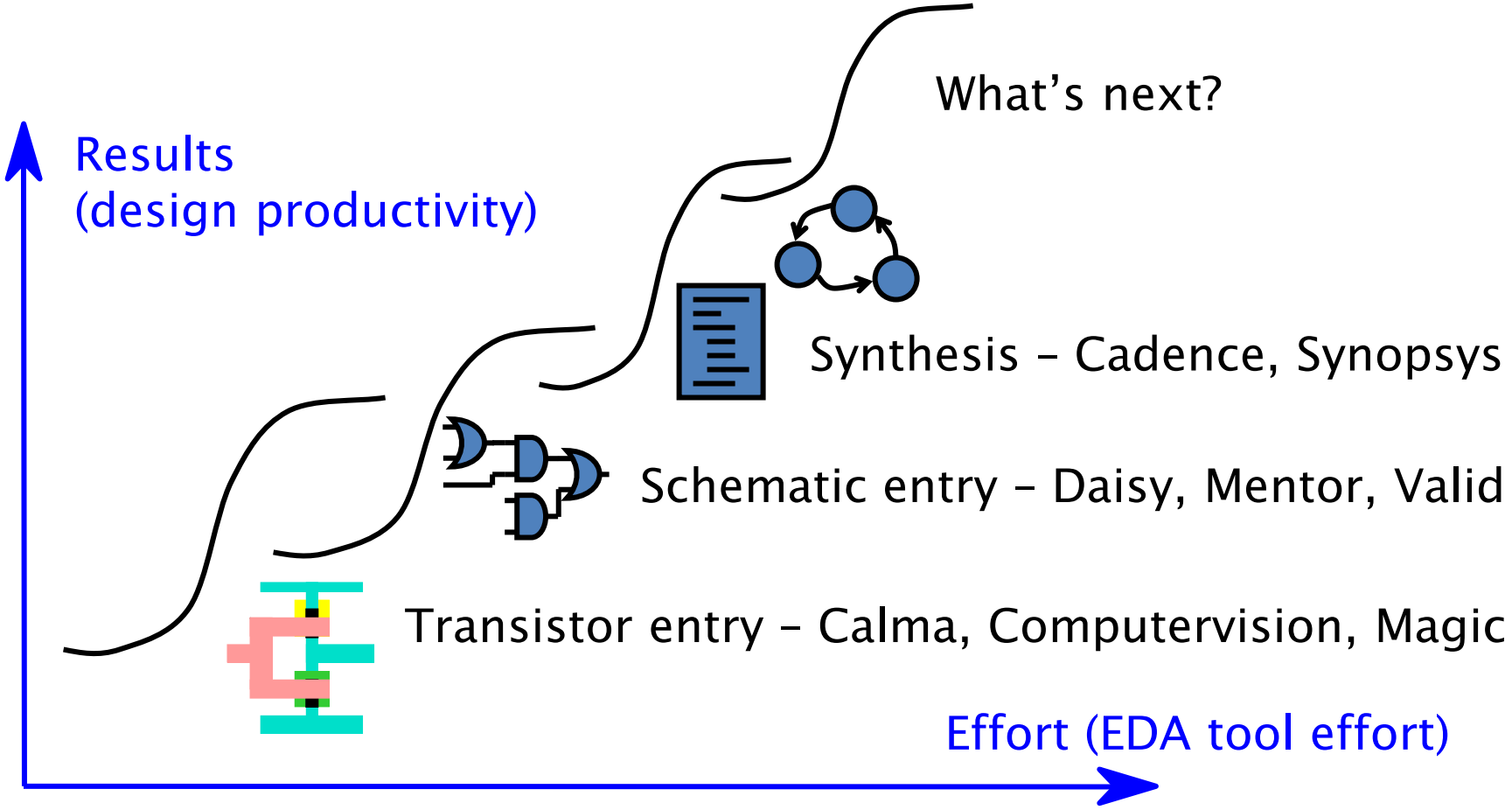


1 billion transistors

Data explosion and productivity

Representation of a uP		**Manual entry Productivity (in Gates/week/designer)**
"high level model"	(~50k lines)	~5k
RTL (register transfer level)	(~500k lines)	~1k
Gates	(~5M)	~100
Transistors	(~50M)	~10
Polygons	(~500M)	~1

Evolution of the EDA Industry



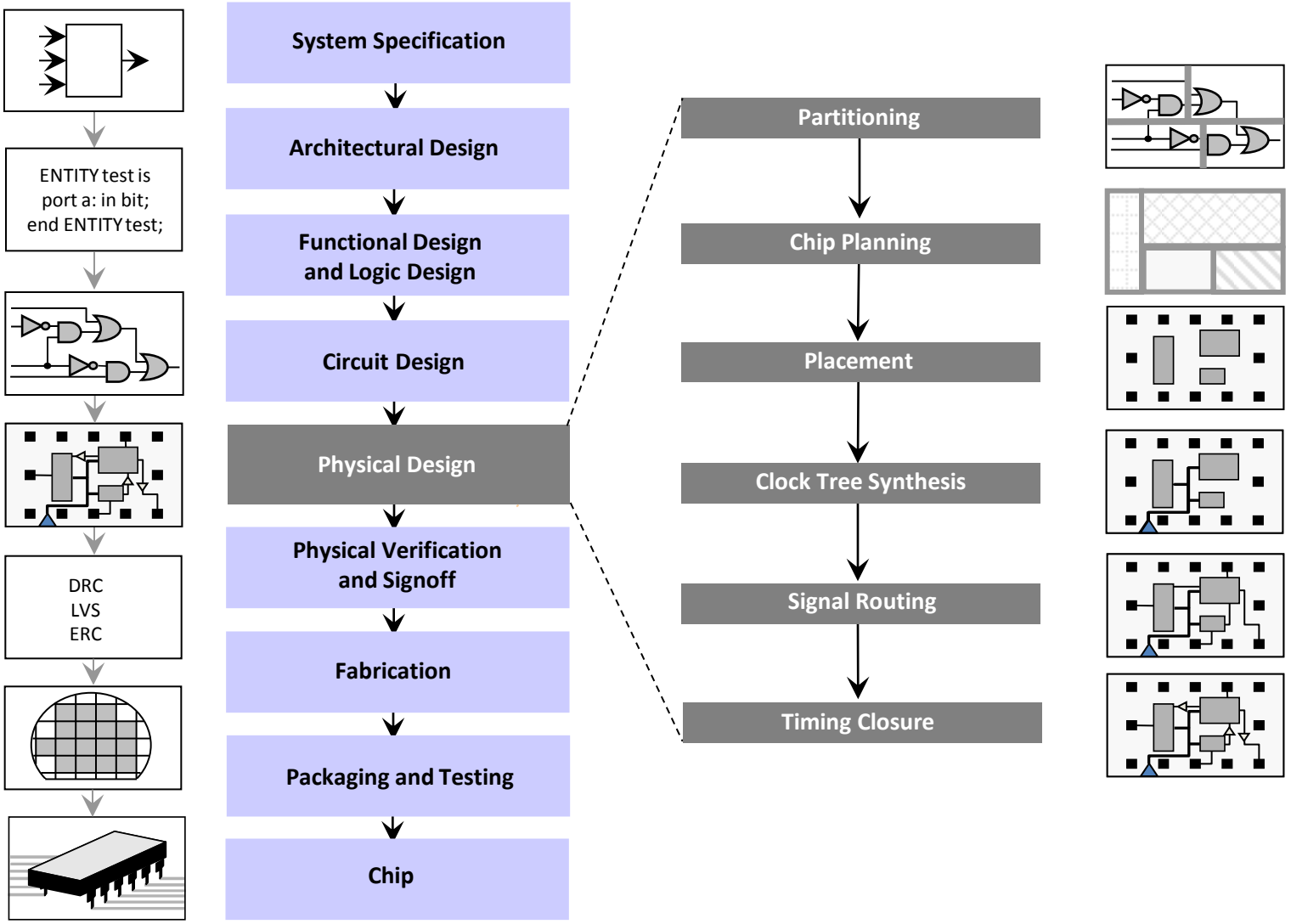
History of VLSI Layout Tools

Year	Design Tools
1950 - 1965	Manual Design
1965 - 1975	Layout editors Automatic routers(for PCB) Efficient partitioning algorithm
1975 - 1985	Automatic placement tools Well Defined phases of design of circuits Significant theoretical development in all phases
1985 – 1995	Performance driven placement and routing tools Parallel algorithms for physical design Significant development in underlying graph theory Combinatorial optimization problems for layout
1995 – 2002	Interconnect layout optimization, Interconnect-centric design, physical-logical codesign
2002 - present	Physical synthesis with more vertical integration for design closure (timing, noise, power, P/G/clock, manufacturability)

Synthesis and Design Process (High Level)

- Application (graphics, DSP, general processor)
- Algorithm (Z-buffer, FFT)
- Architecture (pipeline, cash sharing, parallelism)
- High level synthesis
- Logic and physical synthesis

VLSI Design Flow

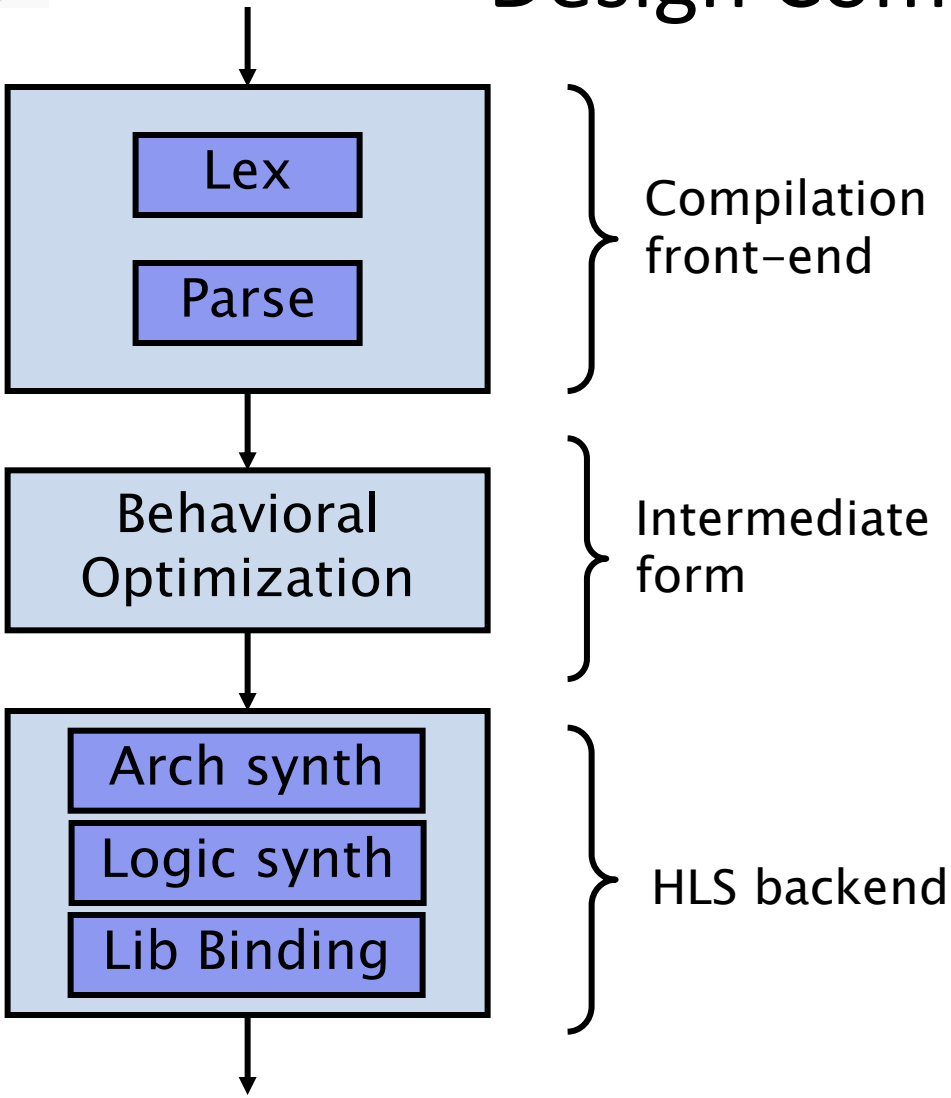


High Level Synthesis (HLS)

Converting high-level design description to RTL

- Input:
  - High-level languages (C, system C, system Verilog)
  - Hardware description languages (Verilog, VHDL)
  - State diagrams / logic networks
- Tools:
  - Parser, compiler
  - Library of modules
- Constraints:
  - Resource constraints (number of modules of a certain type)
  - Timing constraints (latency, delay, clock cycle)
- Output:
  - Operation scheduling (time) and binding (resource)
  - Control generation
  - RTL architecture

Design Compilation



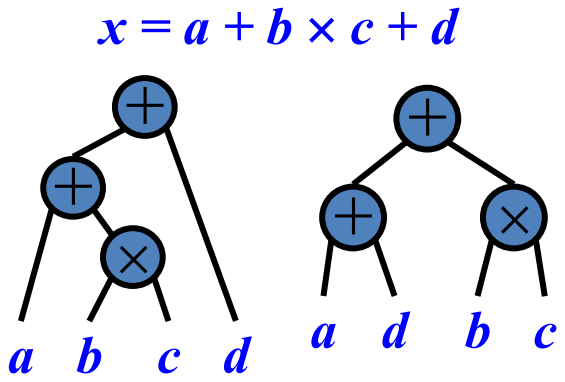
Separation into

- Data Path (arithmetic)
- Control (Boolean logic)

Behavioral Optimization

- Techniques used in software compilation
  - Expression tree height reduction
  - Constant and variable propagation
  - Common sub-expression elimination
  - Dead-code elimination
  - Operator strength reduction (e.g., *4 → << 2)
- Hardware transformations
  - Conditional expansion
    - If *c* then *x = A*

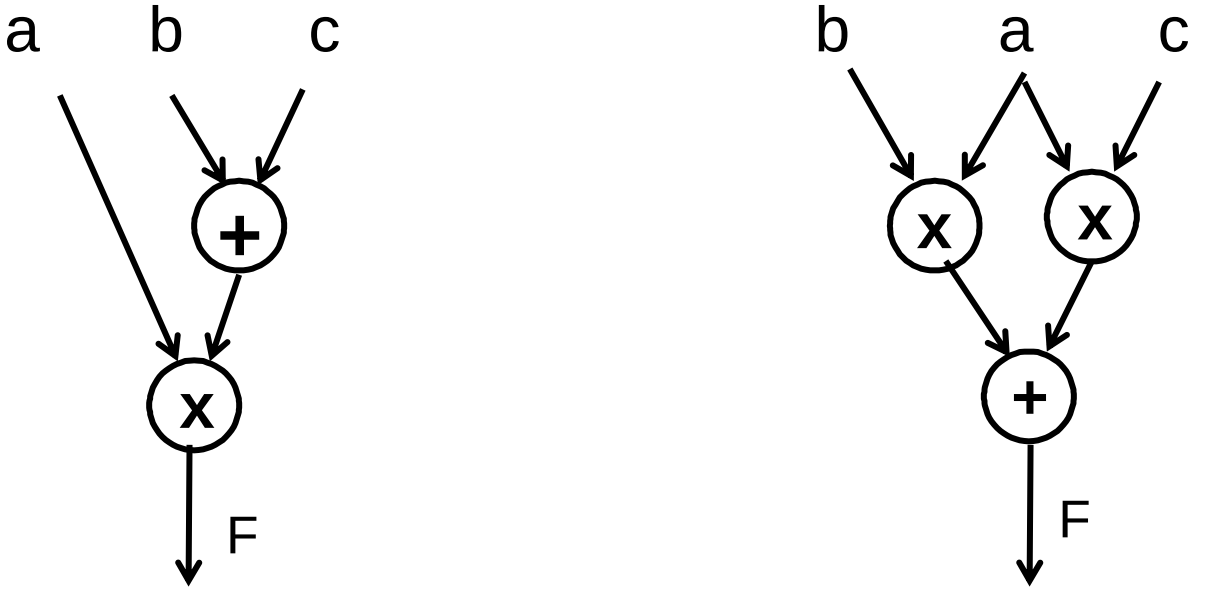
      else *x = B;*
    - Compute *A* and *B* in parallel: *x = C ? A : B* (MUX)
  - Loop unrolling
    - Replace *k* iterations of a loop by *k* instances of the loop body



Data Flow Graph Transformation

Transformation

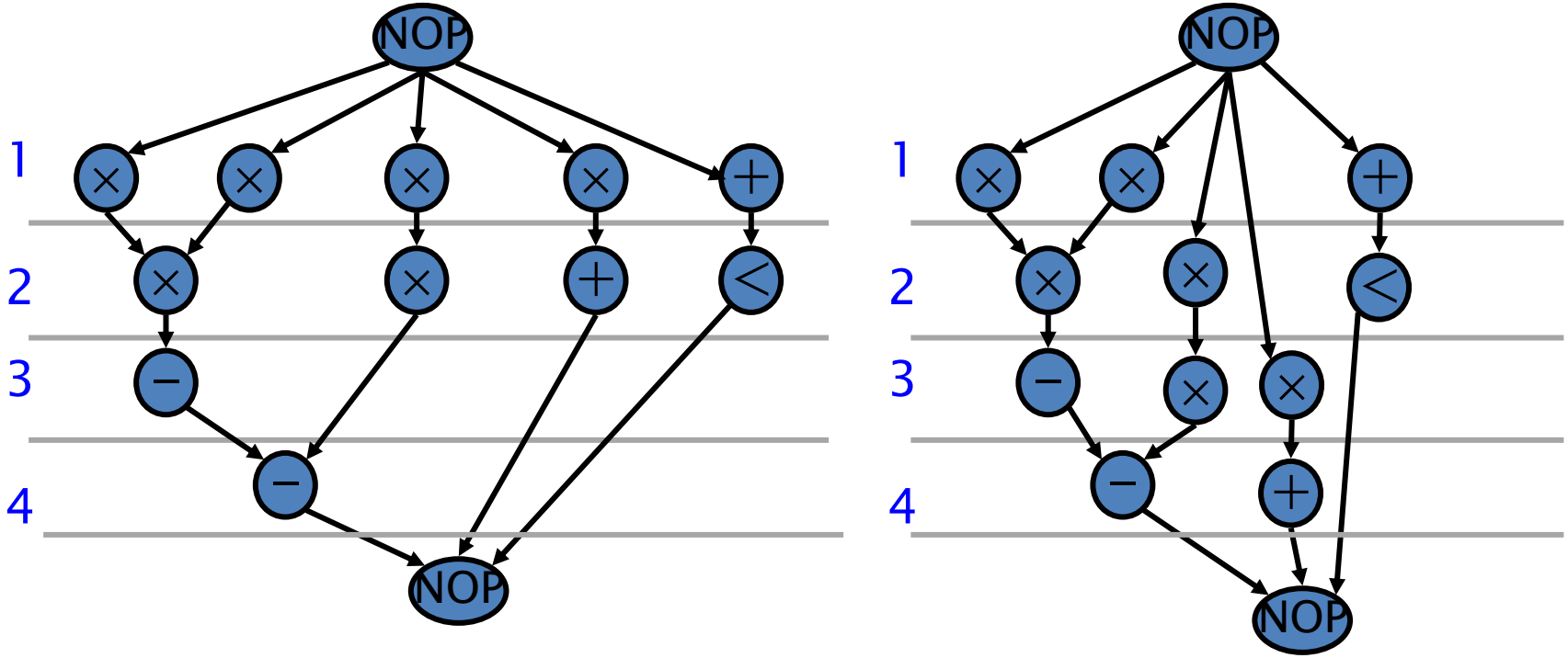
$F = a*(b + c) \longleftrightarrow F = a*b + a*c$



Optimization in Temporal Domain

Scheduling

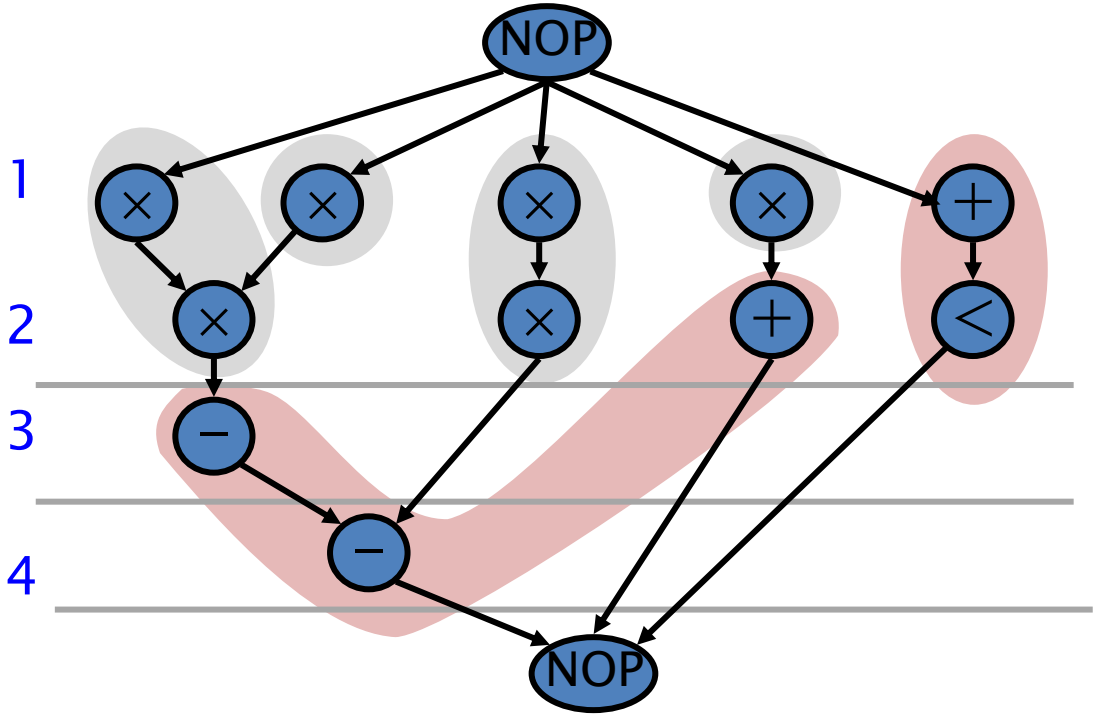
- Mapping of operations to time slots (cycles)
- Uses sequencing graph (data flow graph, DFG)
- Goal: minimize latency (s.t. resource constraints)



Optimization in Spatial Domain

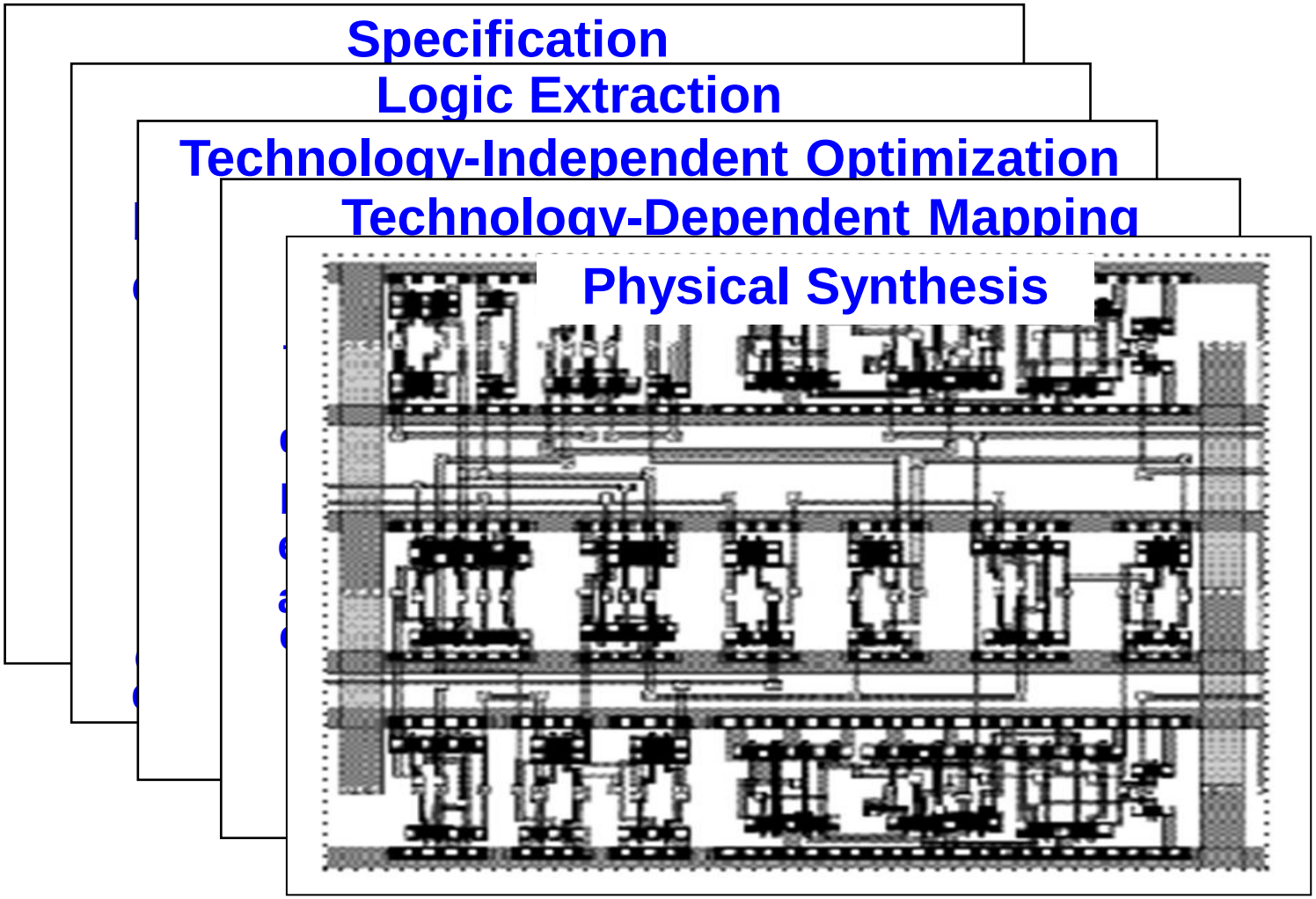
Resource allocation & binding

- Assigning operations to hardware units
- Allocating registers
- Binding operations to same resource
- Goal: minimize resource (s.t. latency constraints)



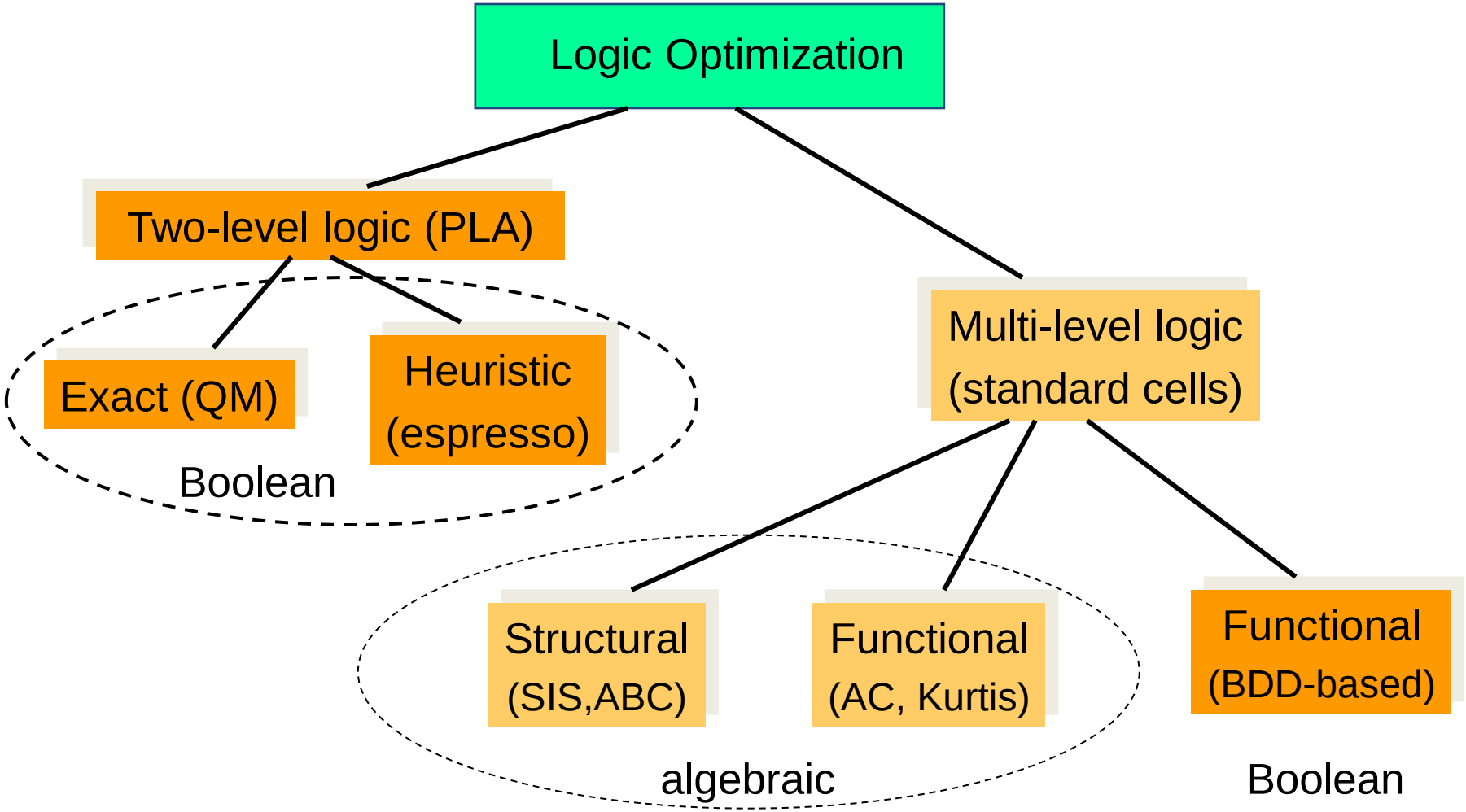
Synthesis Flow at Logic Level

a multi-stage process



Logic Optimization Methods

Depends on target technology



Optimization Criteria for Synthesis

- Area occupied by the logic gates and interconnect (approximated by literals = transistors in technology independent optimization)
- Critical path delay of the longest path through logic
- Degree of testability of the circuit
- Power consumed by the logic gates
- Placeability, Wireability

Transformation-Based Synthesis

sequence of transformations that change network topology and its characteristics

- All modern synthesis systems are built that way
  - work on uniform network representation
  - use scripts, lists of transformations forming a strategy
- Transformations are mostly algebraic
  - very little is based on Boolean factorization
- Representation
  - Cube notation, BDDs, AIGs
- The underlying algorithms
  - Algebraic transformations
  - Collapsing, decomposition
  - Factorization, substitution

Multi-Level Logic Minimization

- Objective
  - Minimize number of literals
  - Literals represent inputs to CMOS gates
- Representation
  - Factored form
  - Compatible with CMOS
- Optimization techniques
  - Algebraic factorization and decomposition (heuristic)
    - Technology independent
  - Requires mapping onto target architecture
    - Standard cells
    - FPGAs (LUT)

Two-Level Logic Minimization

Representation

- Truth tables
- Karnaugh maps
- Sum of Products (SOP) form
- Binary Decision Diagrams (BDD)

Objective

- Minimize number of product terms in SOP
- Challenge: multiple-output functions

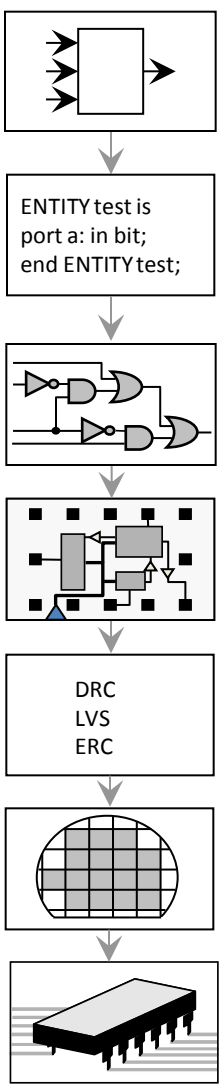
Optimization techniques

- Quine McCluskey (optimal)
- Espresso logic minimizer (heuristic)
- Ashenhust-Curtis functional decomposition (nearly optimal)
- BDD-based (heuristic)

Physical Design Steps

- Circuit partitioning
- Floorplanning
- Pin assignment
- Placement
- Routing
- Convergence

Partitioning



System Specification

Architectural Design

Functional Design and Logic Design

Circuit Design

Physical Design

Physical Verification and Signoff

Fabrication

Packaging and Testing

Chip

Partitioning

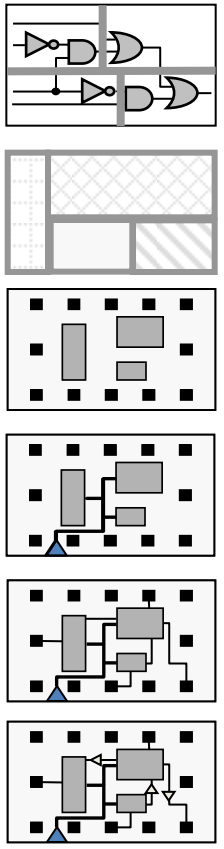
Chip Planning

Placement

Clock Tree Synthesis

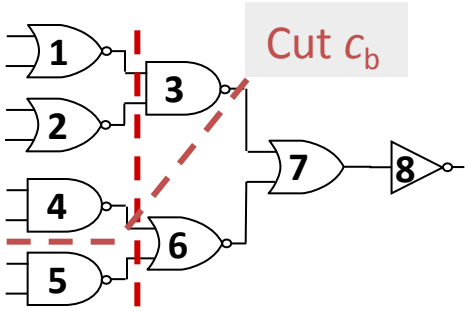
Signal Routing

Timing Closure

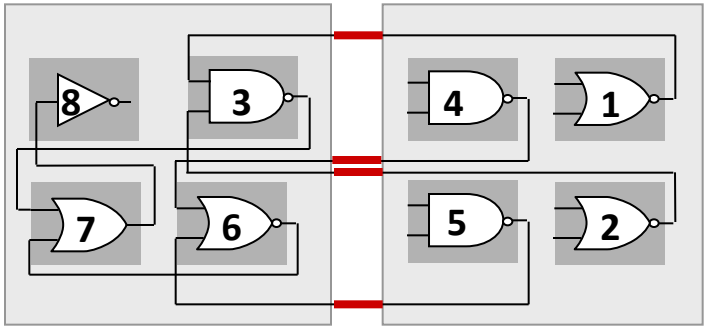


Partitioning

Circuit:

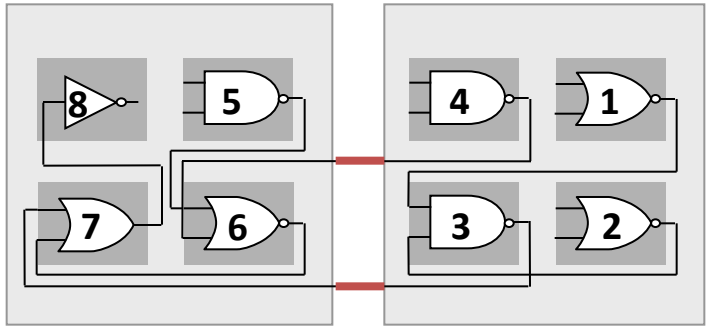


Block *A* Block *B*



Cut c_a: four external connections

Block *A* Block *B*



Cut c_b: two external connections

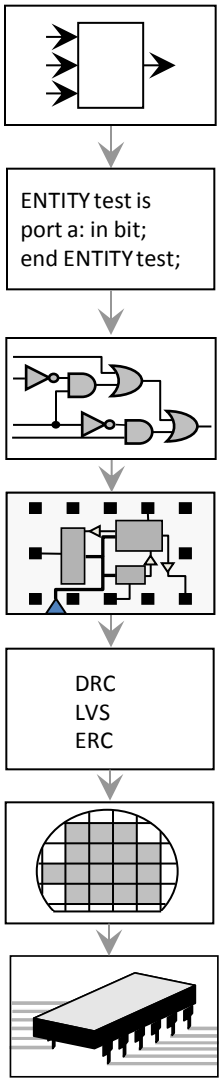
Partitioning - optimization Goals

- In detail, what are the optimization goals?
  - –Number of connections between partitions is minimized
  - –Each partition meets all design constraints (size, number of external connections..)
  - –Balance every partition as well as possible

- How can we meet those goals?
  - –Unfortunately, this problem is NP-hard
  - –Efficient heuristics developed in the 1970s and 1980s. High quality and low-order polynomial time.

39

Floorplanning



System Specification

Architectural Design

Functional Design and Logic Design

Circuit Design

Physical Design

Physical Verification and Signoff

Fabrication

Packaging and Testing

Chip

Partitioning

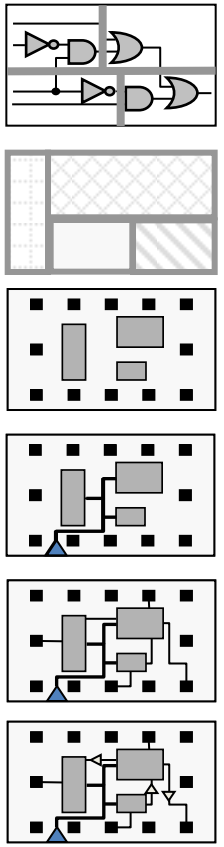
Chip Planning

Placement

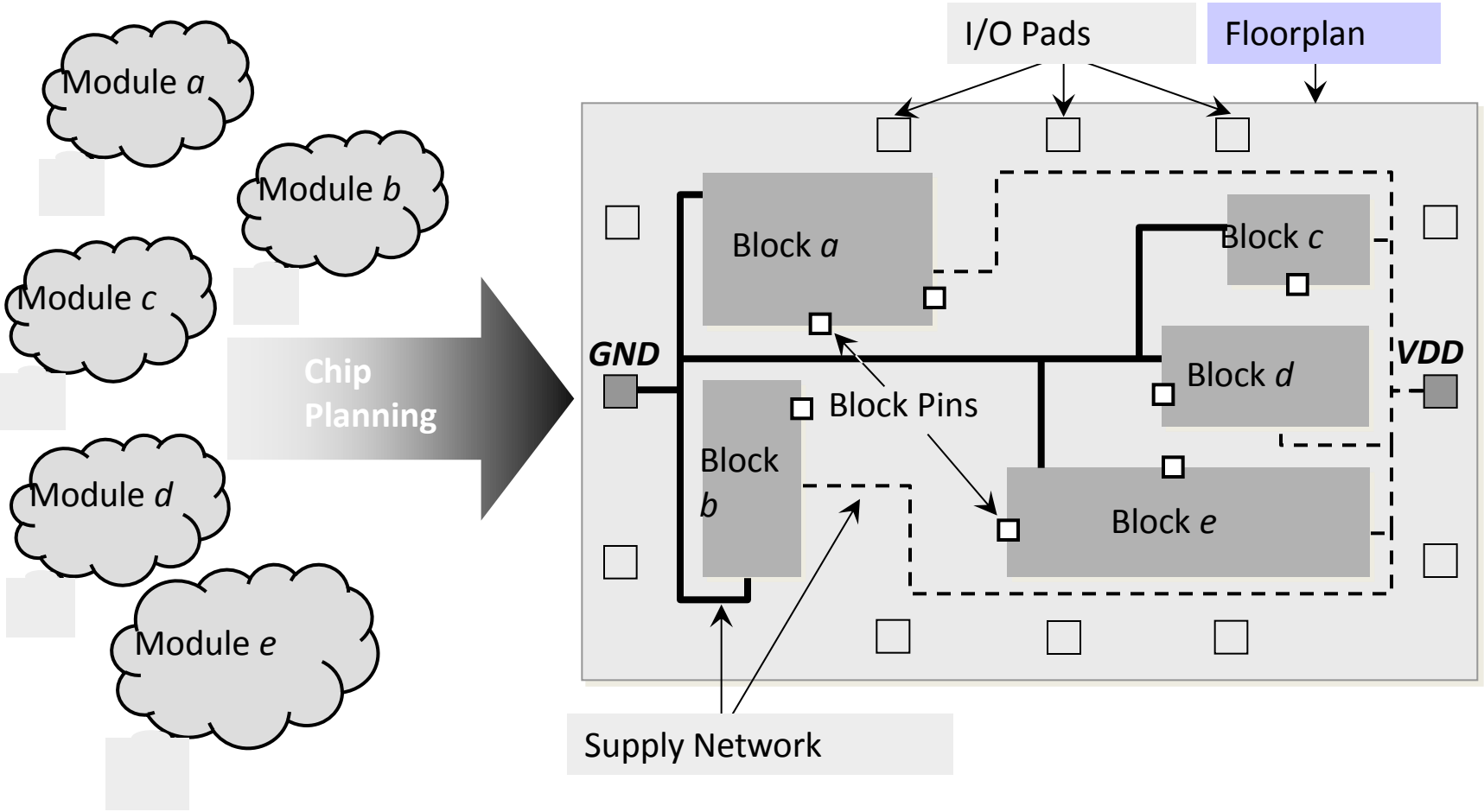
Clock Tree Synthesis

Signal Routing

Timing Closure



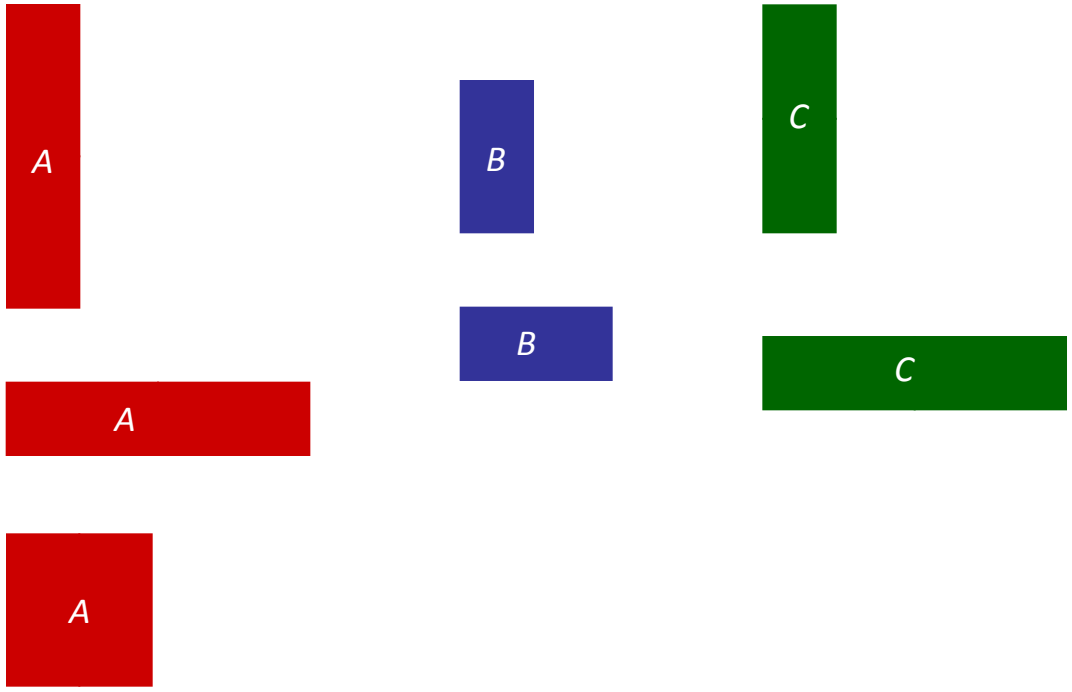
Floorplanning



Floorplanning

Example
Given: Three blocks with the following potential widths and heights
Block *A*: $w = 1, h = 4$ or $w = 4, h = 1$ or $w = 2, h = 2$
Block *B*: $w = 1, h = 2$ or $w = 2, h = 1$
Block *C*: $w = 1, h = 3$ or $w = 3, h = 1$

Task: Floorplan with minimum total area enclosed



Floorplanning

Example

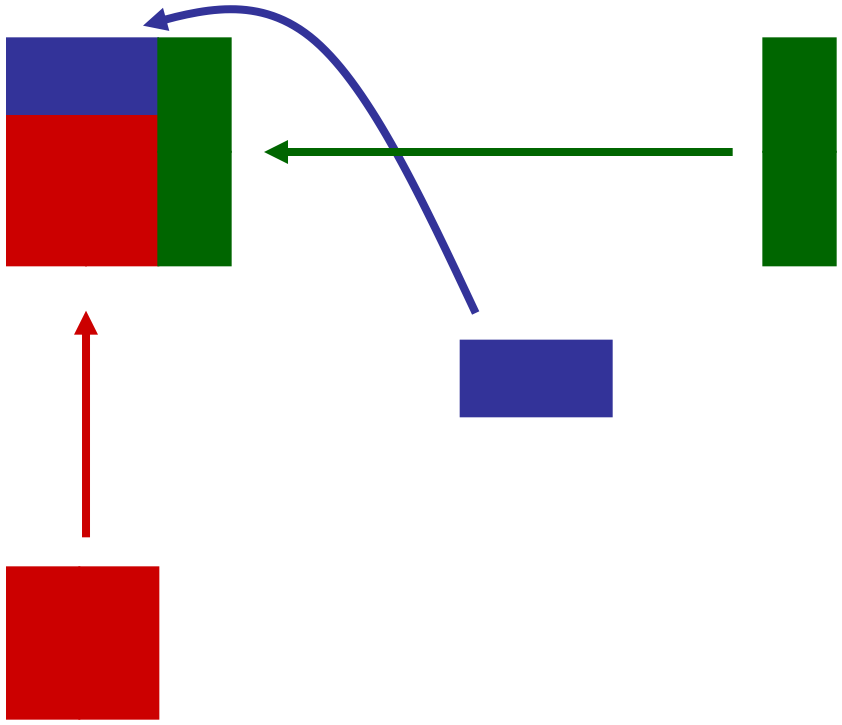
Given: Three blocks with the following potential widths and heights

Block *A*: $w = 1, h = 4$ or $w = 4, h = 1$ or $w = 2, h = 2$

Block *B*: $w = 1, h = 2$ or $w = 2, h = 1$

Block *C*: $w = 1, h = 3$ or $w = 3, h = 1$

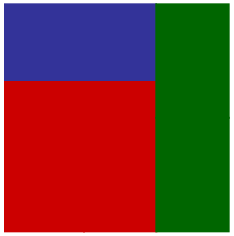
Task: Floorplan with minimum total area enclosed



Floorplanning

Example
Given: Three blocks with the following potential widths and heights
Block *A*: $w = 1, h = 4$ or $w = 4, h = 1$ or $w = 2, h = 2$
Block *B*: $w = 1, h = 2$ or $w = 2, h = 1$
Block *C*: $w = 1, h = 3$ or $w = 3, h = 1$

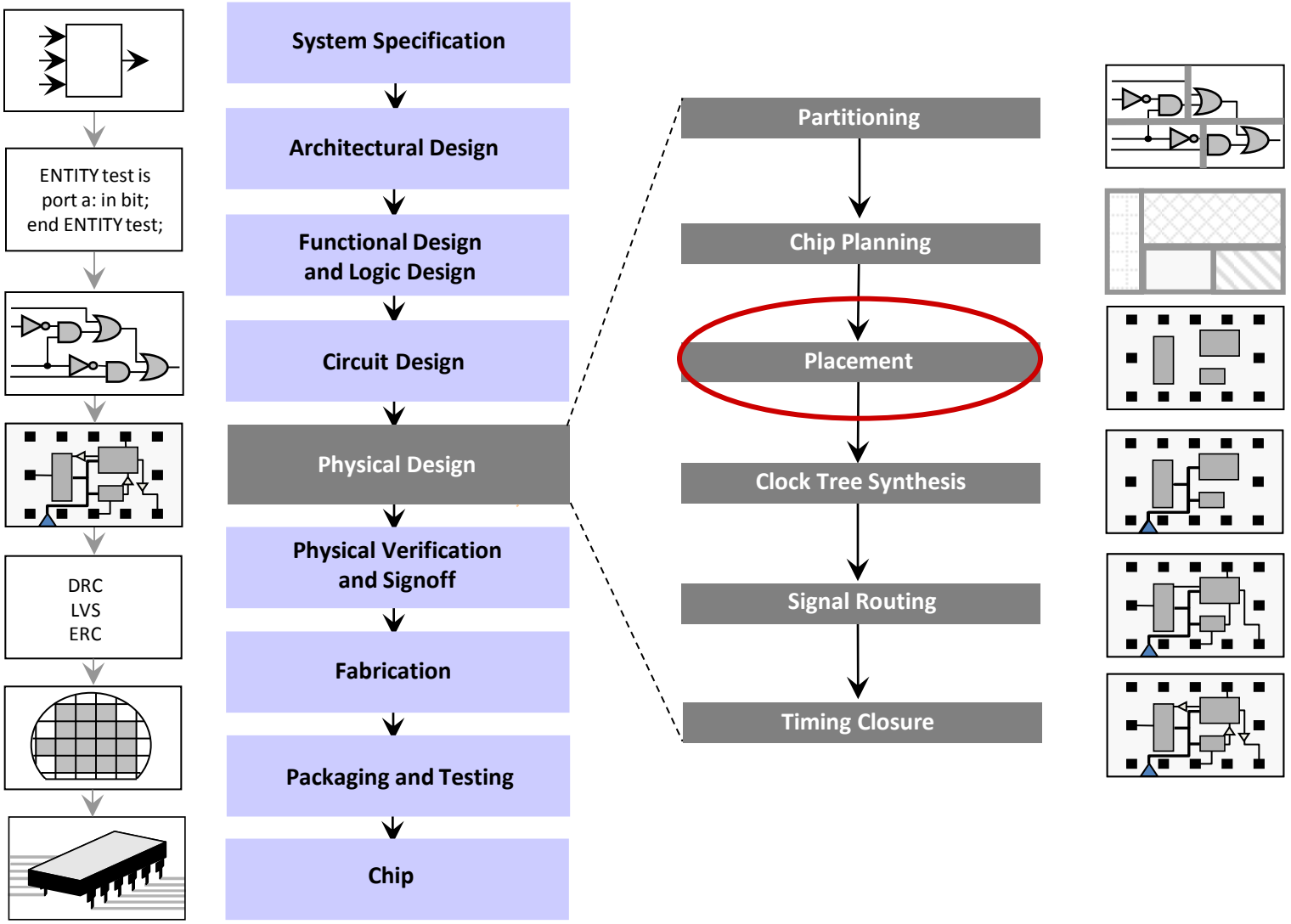
Task: Floorplan with minimum total area enclosed



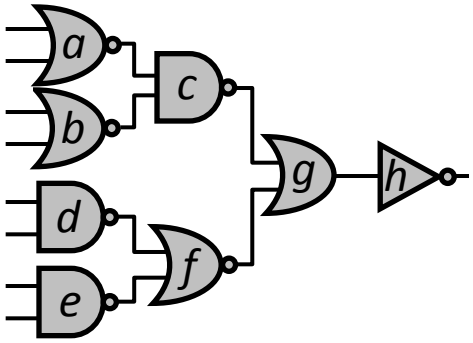
Solution:
Aspect ratios
Block *A* with $w = 2, h = 2$; Block *B* with $w = 2, h = 1$; Block *C* with $w = 1, h = 3$

This floorplan has a global bounding box with minimum possible area (9 square units).

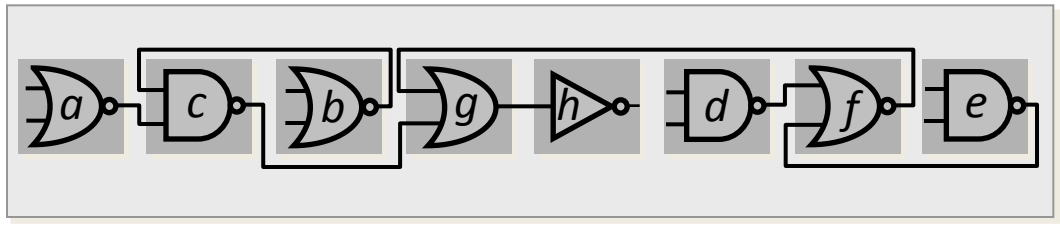
Placement

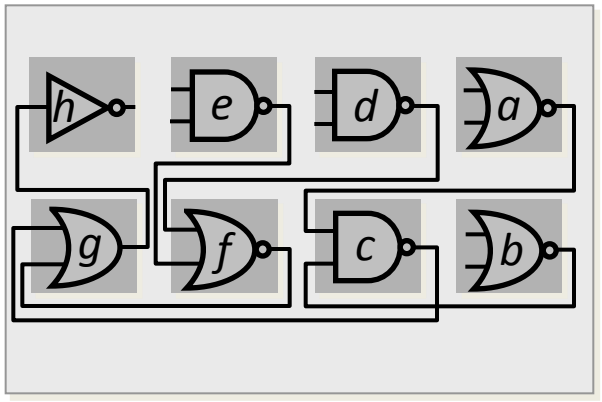

ENTITY test is
port a: in bit;
end ENTITY test;
DRC
LVS
ERC
System Specification
Architectural Design
Functional Design
and Logic Design
Circuit Design
Physical Design
Physical Verification
and Signoff
Fabrication
Packaging and Testing
Chip
Partitioning
Chip Planning
Placement
Clock Tree Synthesis
Signal Routing
Timing Closure

Placement

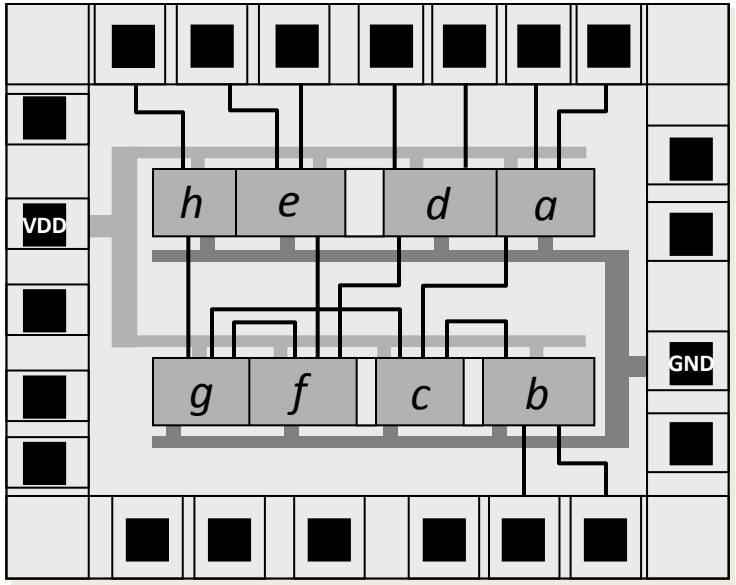


Linear Placement





2D Placement

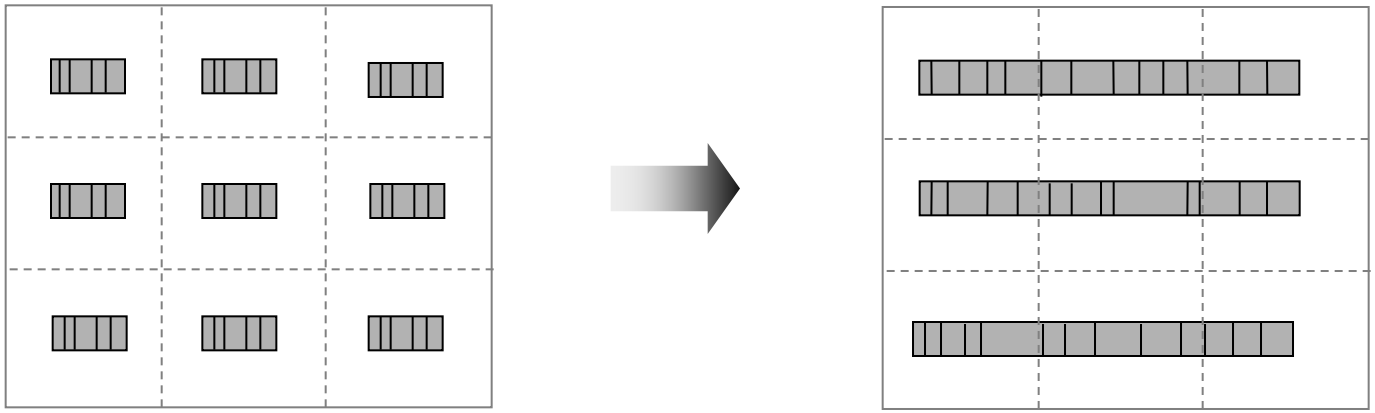


Placement and Routing with Standard Cells

Placement

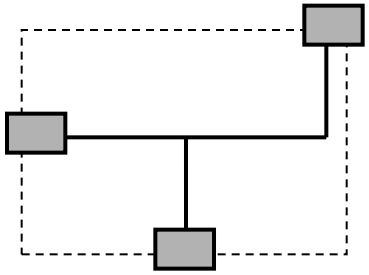
Global Placement

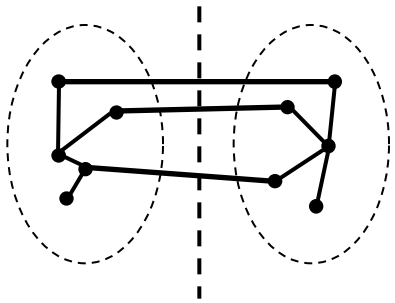
Detailed Placement

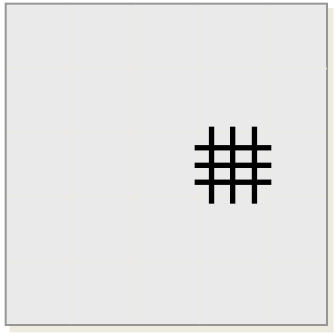


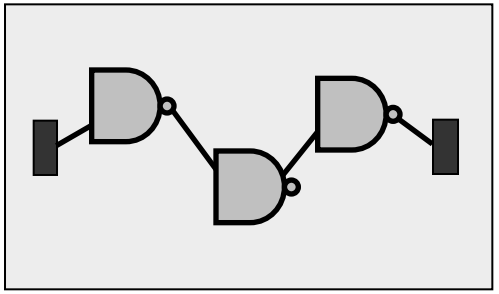
Placement Optimization Objectives

Total Wirelength	Number of Cut Nets	Wire Congestion	Signal Delay

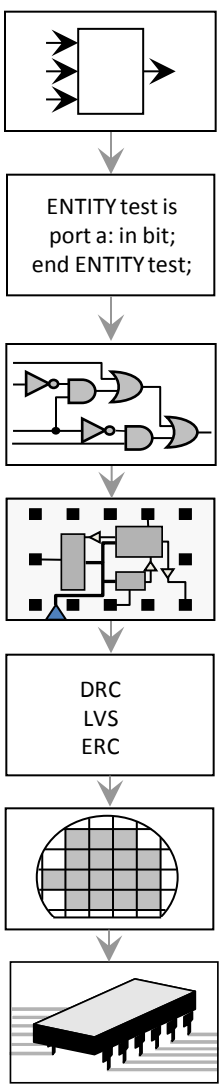








Routing



System Specification

Architectural Design

Functional Design and Logic Design

Circuit Design

Physical Design

Physical Verification and Signoff

Fabrication

Packaging and Testing

Chip

Partitioning

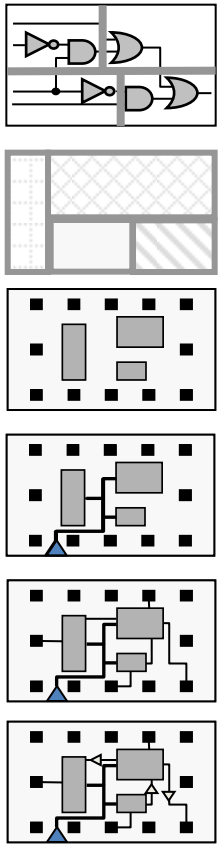
Chip Planning

Placement

Clock Tree Synthesis

Signal Routing

Timing Closure



Routing

Given a placement, a netlist and technology information,

- determine the necessary wiring, e.g., net topologies and specific routing segments, to connect the cells
- while respecting constraints, e.g., design rules and routing resource capacities, and
- optimizing routing objectives, e.g., minimizing total wirelength and maximizing timing slack.

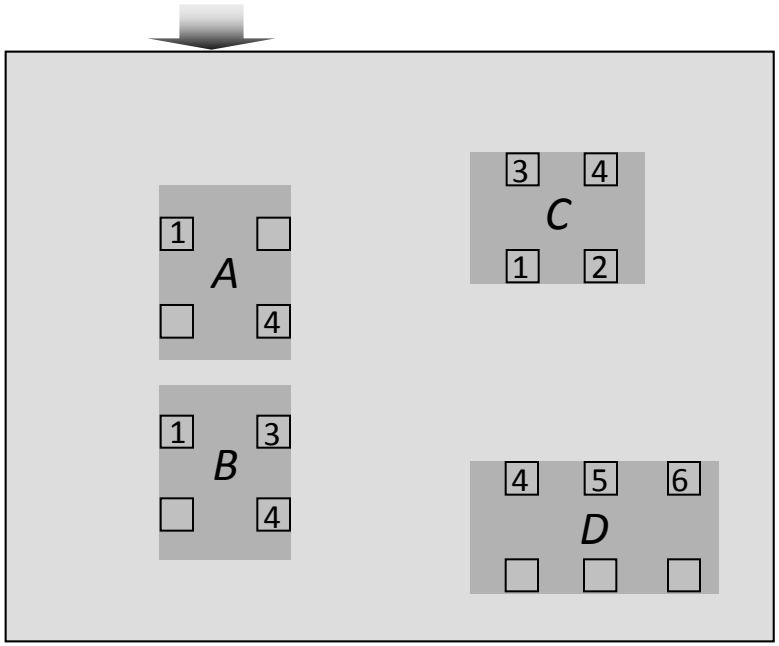
Routing

Netlist:

$N_1 = \{C_4, D_6, B_3\}$
$N_2 = \{D_4, B_4, C_1, A_4\}$
$N_3 = \{C_2, D_5\}$
$N_4 = \{B_1, A_1, C_3\}$

Technology Information (Design Rules)

Placement result

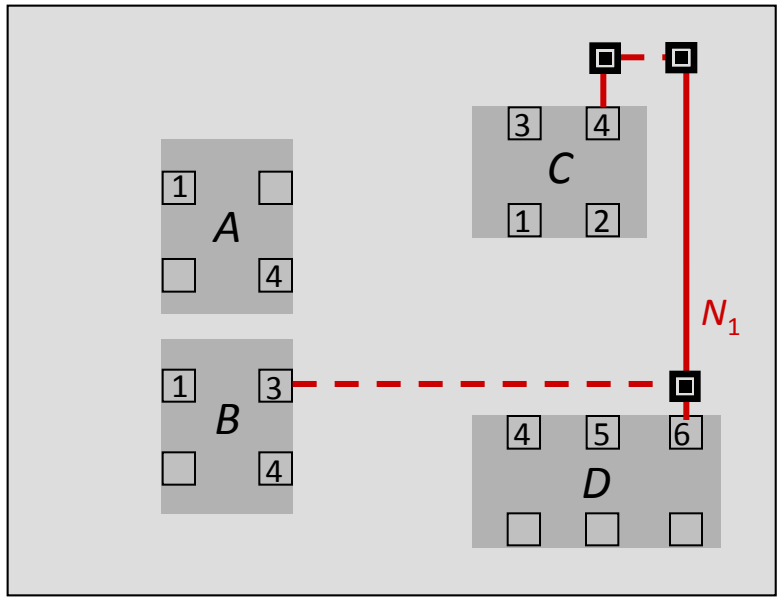


Routing

Netlist:

$N_1 = \{C_4, D_6, B_3\}$
$N_2 = \{D_4, B_4, C_1, A_4\}$
$N_3 = \{C_2, D_5\}$
$N_4 = \{B_1, A_1, C_3\}$

Technology Information
(Design Rules)

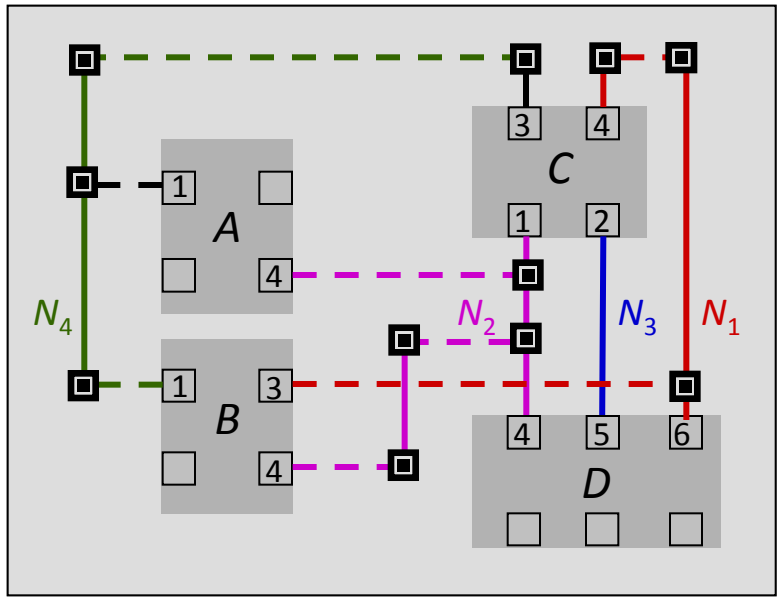


Routing

Netlist:

$N_1 = \{C_4, D_6, B_3\}$
$N_2 = \{D_4, B_4, C_1, A_4\}$
$N_3 = \{C_2, D_5\}$
$N_4 = \{B_1, A_1, C_3\}$

Technology Information
(Design Rules)

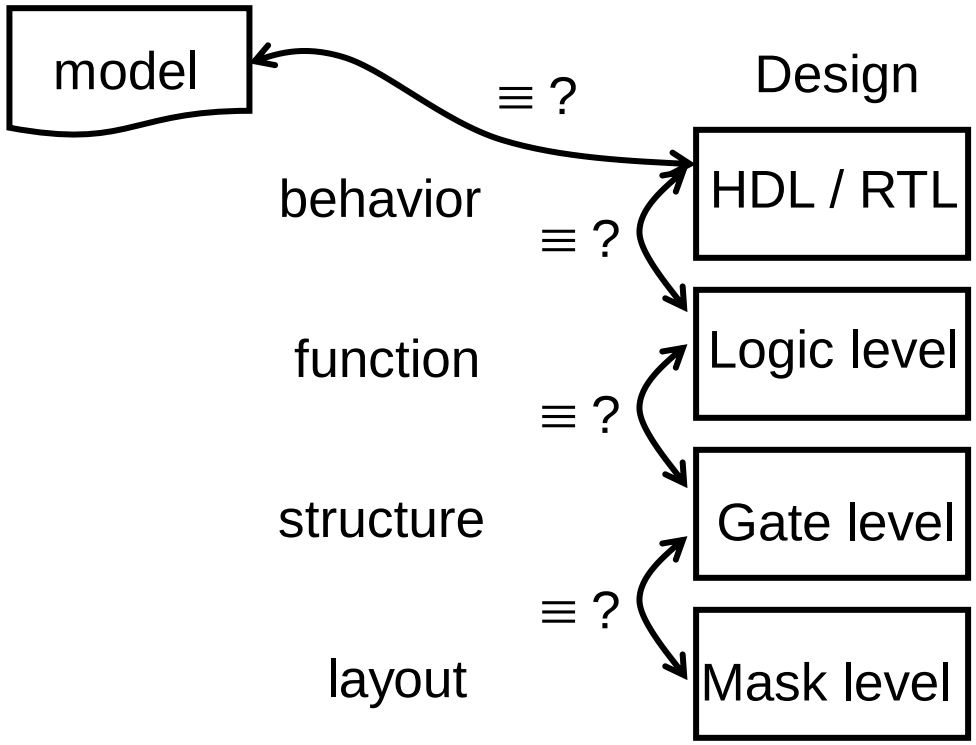


The Design Closure Problem

Iterative removal of timing violations (white lines)

Design Verification

Ensuring correctness of the design against its implementation (at different levels)



Algorithm Design Techniques

- Greedy
- Divide and Conquer
- Dynamic Programming
- Network Flow
- Mathematical Programming (e.g., linear programming, integer linear programming)

Reduction

- Idea: If I can solve problem A, and if problem B can be transformed into an instance of problem A, then I can solve problem B by reducing problem B to problem A and then solve the corresponding problem A.
- Example:
  - Problem A: Sorting
  - Problem B: Given n numbers, find the i-th largest numbers.

Analysis of Algorithm

- There can be many different algorithms to solve the same problem.
- Need some way to compare 2 algorithms.
- Usually run time is the most important criterion used
  - Space (memory) usage is of less concern now
- However, difficult to compare since algorithms may be implemented in different machines, use different languages, etc.
- Also, run time is input-dependent. Which input to use?
- Big-O notation is widely used for asymptotic analysis.