Introduction to Electronic Design Automation

Jie-Hong Roland Jiang
江介宏

Department of Electrical Engineering
National Taiwan University

Spring 2011

Design Automation?

Course Info (1/4)

Instructor
- Jie-Hong R. Jiang
- email: jhjiang@cc.ee.ntu.edu.tw
- office: 242, EE2 Building
- phone: (02)3366-3685
- office hour: 14:00-16:00 Thu, and other time by appointment

TA
- Yi-Ting Chuang
- email: q82465@gmail.com
- office: 526, Ming-Dar Hall
- phone: (02)3366-9753
- office hour: TBA

Email contact list
- NTU email addresses of enrolled students will be used for future contact

Course webpage
- http://cc.ee.ntu.edu.tw/~jhjiang/instruction/courses/spring11-eda/eda.html
- please look up the webpage frequently to keep updated

Course Info (2/4)

Grading rules
- Homework 40%
- Midterm or CAD contest 25% (exam date may differ from official schedule)
- Final exam or project (either option) 30%
- Course participation 5%

Homework
- discussions encouraged, but solutions should be written down individually and separately
- 4~5 assignments in total
- late homework (20% off per day)

Midterm/final exams
- in-class exam

Project
- Team or individual work on selected topics (paper reading / implementation / problem solving, etc.)

Academic integrity: no plagiarism allowed

Course Info (3/4)

- Prerequisite
  - Switching circuits and logic design, or by instructor's consent
- Main lecture basis
  - Lecture slides and/or handouts
- Textbook
  - Y.-W. Chang, K.-T. Cheng, and L.-T. Wang (Editors). *Electronic Design Automation: Synthesis, Verification, and Test*. Elsevier, 2009.
- Reference
  - S. H. Gerez. *Algorithms for VLSI Design Automation*. John Wiley & Sons, 1999.

Course Info (4/4)

- Objectives:
  - Peep into EDA
  - Motivate interests
  - Learn problem formulation and solving
  - Have fun!

FYI

- 九十九學年度大學校院積體電路電腦輔助設計(CAD)軟體製作競賽
  - http://cad_contest.cs.nctu.edu.tw/cad11/
    - Registration deadline 01/04/2011
    - Submission deadline 25/04/2011

FAQ

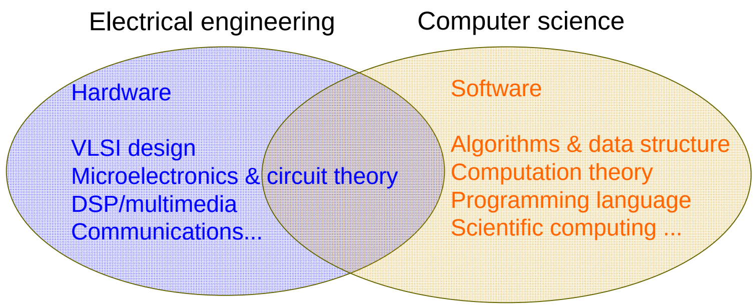
- What's EDA?
  - What are we concerned about?
  - What's unique in EDA compared to other EE/CS disciplines?
- What time is good to take *Intro to EDA*?
  - Am I qualified? Do I have enough backgrounds?
- How's the loading?
  - Program to death!?
- What kind of skills and domain knowledge can I learn? Other applications?
- What are the career opportunities?
- Yet another question?

Course Outline

- Introduction
- Computation in a nutshell
- High-level synthesis
- Logic synthesis
- Formal verification
- Physical design
- Testing
- Simulation
- Advanced topics

Introduction

- EDA, where HW and SW meet each other

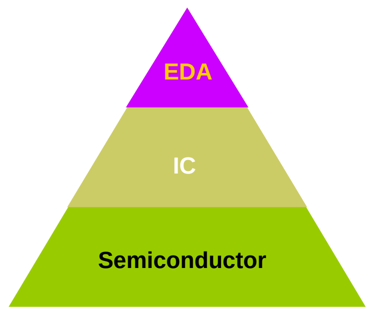


Introduction

- EDA is concerned about HW/SW design in terms of
  - Correctness
  - Productivity
  - Optimality
  - Scalability

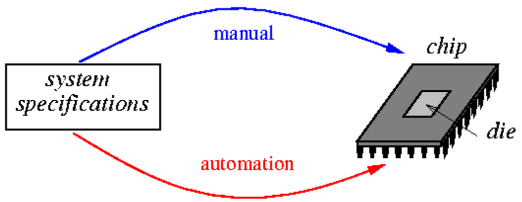
Introduction

- EDA (in a strict sense) and industries
  - Impact - solving a problem may benefit vast electronic designs



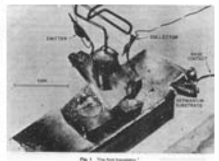
Introduction

- Today's contents:
  - Introduction to VLSI design flow, methodologies, and styles
  - Introduction to VLSI design automation tools
  - Semiconductor technology roadmap
  - CMOS technology
- Reading:
  - Chapters 1, 2



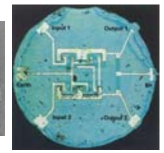
Milestones of IC Industry

- **1947:** Bardeen, Brattain & Shockly invented the transistor, foundation of the IC industry.
- **1952:** SONY introduced the first transistor-based radio.
- **1958:** Kilby invented integrated circuits (ICs).
- **1965:** Moore's law.
- **1968:** Noyce and Moore founded Intel.
- **1970:** Intel introduced 1 K DRAM.



First transistor

First IC by Kilby

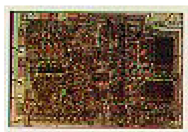


First IC by Noyce

Milestones of IC Industry

- **1971:** Intel announced 4-bit 4004 microprocessors (2250 transistors).
- **1976/81:** Apple II/IBM PC.
- **1985:** Intel began focusing on microprocessor products.
- **1987:** TSMC was founded (fabless IC design).
- **1991:** ARM introduced its first embeddable RISC IP core (chipless IC design).

Intel founders

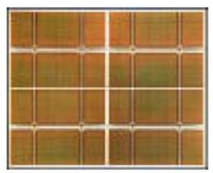


4004

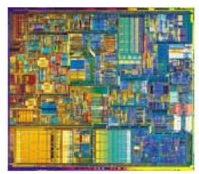
IBM PC

Milestones of IC Industry

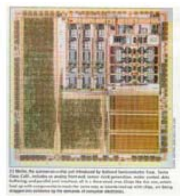
- **1996:** Samsung introduced 1G DRAM.
- **1998:** IBM announces 1GHz experimental microprocessor.
- **1999/earlier: System-on-Chip (SoC)** methodology applications.
- **2002/earlier: System-in-Package (SiP)** technology
- An Intel P4 processor contains 42 million transistors (1 billion by 2005)
- Today, we produce > 30 million transistors per person (1 billion/person by 2008).



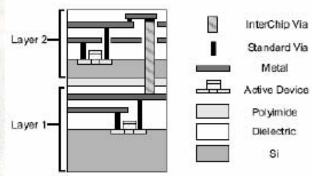
4GB DRAM (2001)



Pentium 4

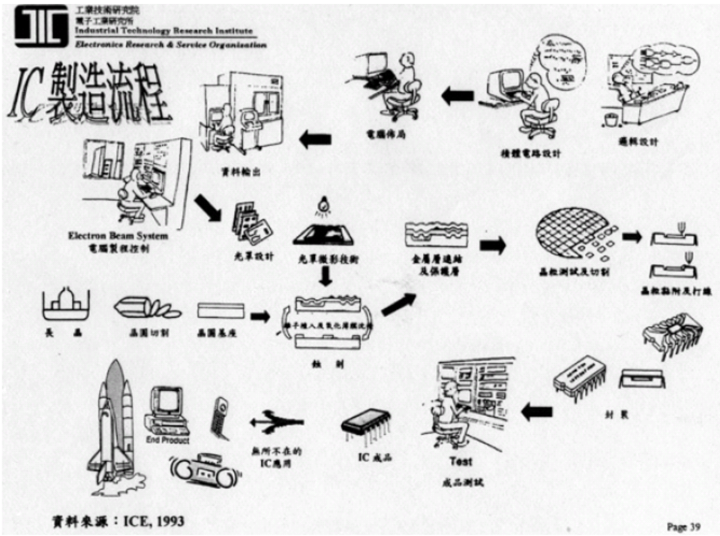


Scanner-on-chip

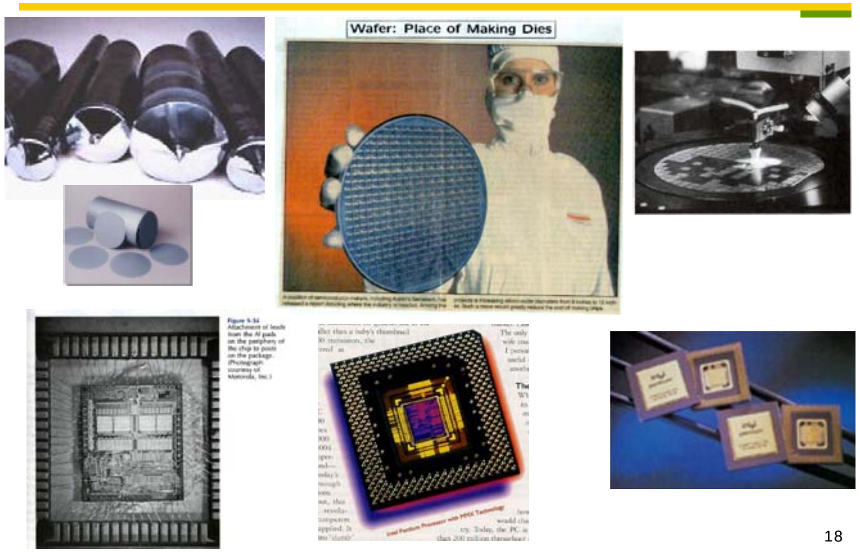


System in Package (SiP)

IC Design & Manufacturing Process



From Wafer to Chip

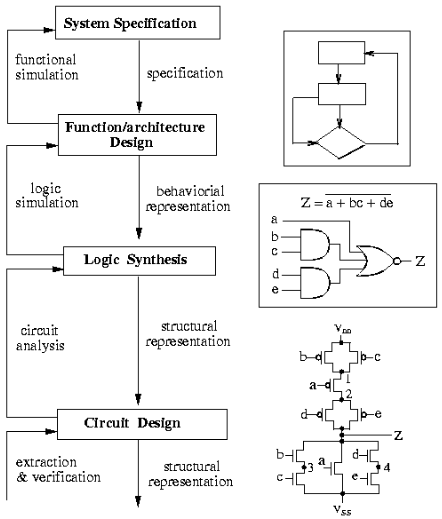


Standard VLSI Design Cycles

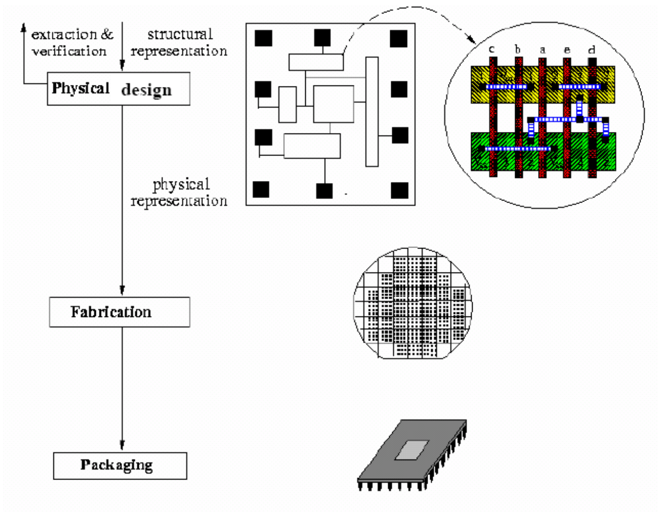
1. System specification
2. Functional design
3. Logic synthesis
4. Circuit design
5. Physical design and verification
6. Fabrication
7. Packaging

- Other tasks involved: testing, simulation, etc.
- Design metrics: area, speed, power dissipation, noise, design time, testability, etc.
- Design revolution: interconnect (not gate) delay dominates circuit performance in deep submicron era.
  - Interconnects are determined in physical design.
  - Shall consider interconnections in early design stages.

VLSI Design Flow



VLSI Design Flow



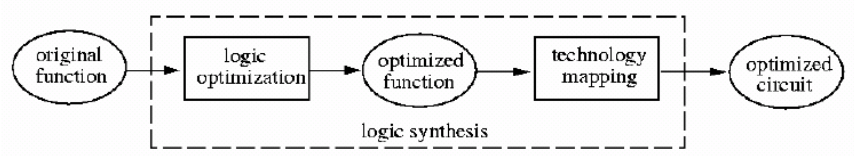
Design Actions

- **Synthesis:** increasing information about the design by providing more detail (e.g., logic synthesis, physical synthesis).
- **Analysis:** collecting information on the quality of the design (e.g., timing analysis).
- **Verification:** checking whether a synthesis step has left the specification intact (e.g., function, layout verification).
- **Optimization:** increasing the quality of the design by rearrangements in a given description (e.g., logic optimizer, timing optimizer).
- **Design management:** storage of design data, cooperation between tools, design flow, etc. (e.g., database).

Design Issues and Tools

- System-level design
  - Partitioning into hardware and software, co-design/simulation etc.
  - Cost estimation, design-space exploration
- Algorithmic-level design
  - Behavioral descriptions (e.g. in Verilog, VHDL)
  - High-level simulation
- From algorithms to hardware modules
  - High-level (or architectural) synthesis
- Logic design:
  - Register-transfer level and logic synthesis
  - Gate-level simulation (functionality, power, etc)
  - Timing analysis
  - Formal verification

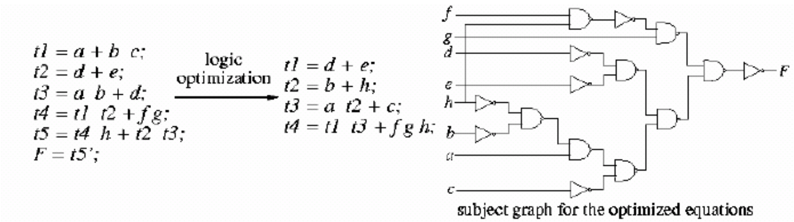
Logic Design/Synthesis



- **Logic synthesis** programs transform Boolean expressions into logic gate networks in a particular library.
- Optimization goals: minimize area, delay, power, etc
- **Technology-independent** optimization: logic optimization
  - Optimizes Boolean expression equivalent.
- **Technology-dependent** optimization: **technology mapping/library binding**
  - Maps Boolean expressions into a particular cell library.

Logic Optimization Examples

- **Two-level:** minimize the # of product terms.
  - $F = \bar{x}_1\bar{x}_2\bar{x}_3 + \bar{x}_1\bar{x}_2x_3 + x_1\bar{x}_2\bar{x}_3 + x_1\bar{x}_2x_3 + x_1x_2\bar{x}_3 \Rightarrow F = \bar{x}_2 + x_1\bar{x}_3$.
- **Multi-level:** minimize the #'s of literals, variables.
  - E.g., equations are optimized using a smaller number of literals.



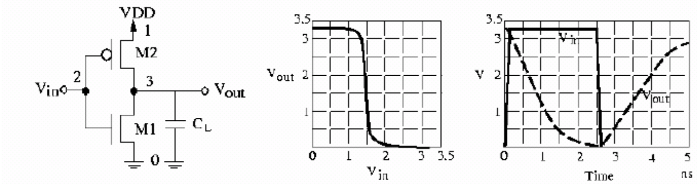
- Methods/CAD tools: Quine-McCluskey method (exponential-time exact algorithm), Espresso (heuristics for two-level logic), SIS (heuristics for multi-level logic), ABC, etc.

Design Issues and Tools

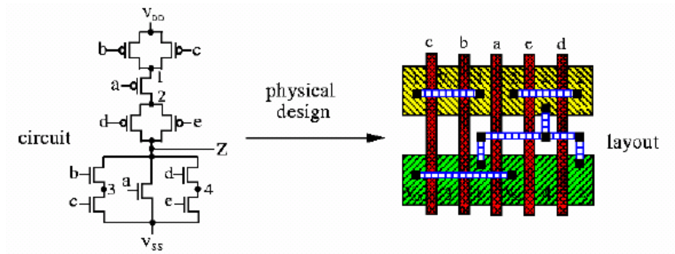
- Transistor-level design
  - Switch-level simulation
  - Circuit simulation
- Physical (layout) design:
  - Partitioning
  - Floorplanning and placement
  - Routing
  - Layout editing and compaction
  - Design-rule checking
  - Layout extraction
- Design management
  - Data bases, frameworks, etc.
- Silicon compilation: *from algorithm to mask patterns*
  - The *idea* is approached more and more, but still far away from a single *push-button* operation

Circuit Simulation

```
M1 3 2 0 0 nch W=1.2u L=0.6u AS=2.16p PS=4.8u AD=2.16p PD=4.8u
M2 3 2 1 1 pch W=1.8u L=0.6u AS=3.24p PS=5.4u AD=3.24p PD=5.4u
CL 3 0 0.2pF

VDD 1 0 3.3
VIN 2 0 DC 0 PULSE (0 3.3 0ns 100ps 100ps 2.4ns 5ns)

.LIB '../mod_06' typical

.OPTION NOMOD POST INGOLD=2 NUMDGT=6 BRIEF
.DC VIN 0V 3.3V 0.001V
.PRINT DC V(3)
.TRAN 0.001N 5N
.PRINT TRAN V(2) V(3)
.END
```

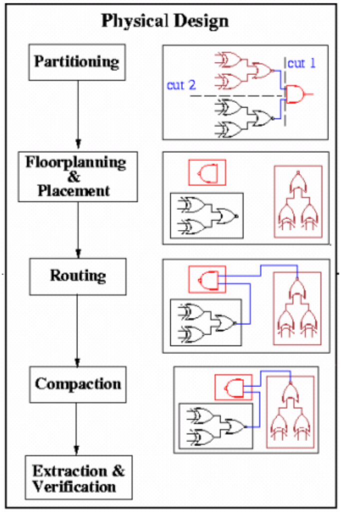


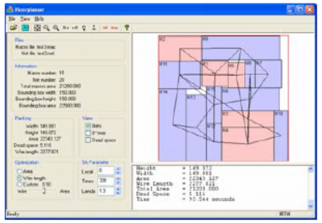
Physical Design



- Physical design converts a circuit description into a geometric description.
- The description is used to manufacture a chip.
- Physical design cycle:
  1. Logic partitioning
  2. Floorplanning and placement
  3. Routing
  4. Compaction
- Others: circuit extraction, timing verification and design rule checking

Physical Design Flow



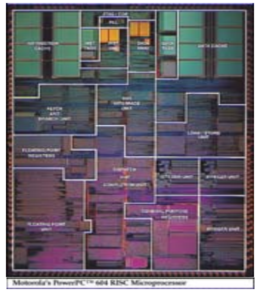


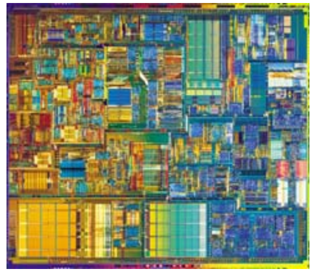
B*-tree based floorplanning system

A routing system

Floorplan Examples

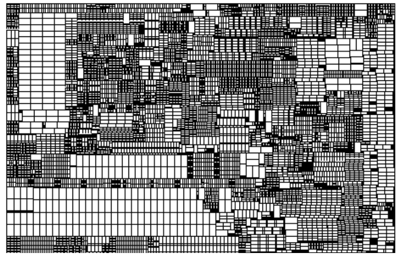
PowerPC 604





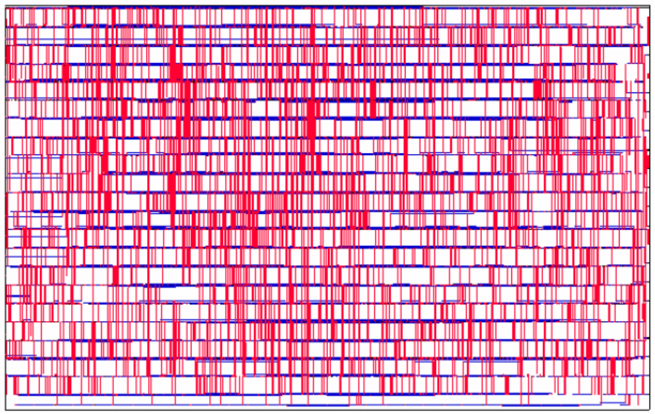
Pentium 4

A floorplan with 9800 blocks

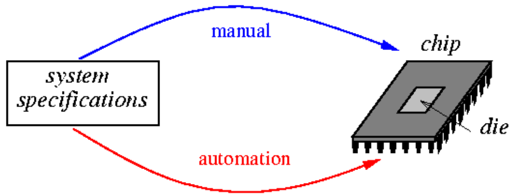


Routing Example

- 0.18um technology, two layers, pitch = 1 um, 8109 nets



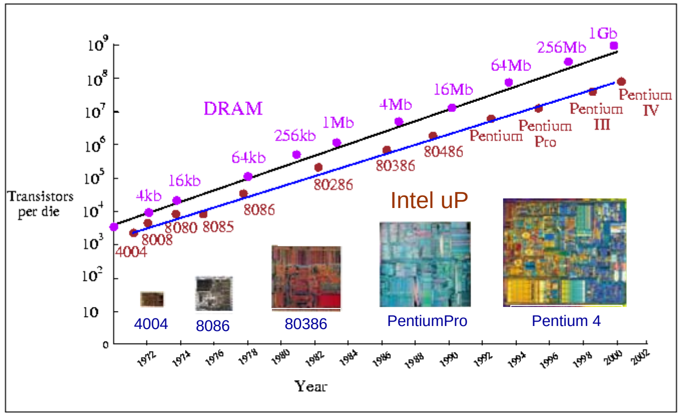
IC Design Considerations



- Several conflicting considerations:
  - **Design complexity:** large number of devices/transistors
  - **Performance:** optimization requirements for high performance
  - **Time-to-market:** about a 15% gain for early birds
  - **Cost:** die **area**, packaging, testing, etc.
  - Others: power, signal integrity (noise, etc), testability, reliability, manufacturability, etc.

Moore's Law: Driving Technology Advances

- Logic capacity doubles per IC at a regular interval
  - Moore: Logic capacity doubles per IC every two years (1975)
  - D. House: Computer performance doubles every 18 months (1975)



Technology Roadmap for Semiconductors

Year	1997	1999	2002	2005	2008	2011	2014
Technology node (nm)	250	180	130	100	70	50	35
On-chip local clock (GHz)	0.75	1.25	2.1	3.5	6.0	10	16.9
Microprocessor chip size (mm^2)	300	340	430	520	620	750	901
Microprocessor transistors/chip	11M	21M	76M	200M	520M	1.40B	3.62B
Microprocessor cost/transistor ($\times 10^{-8}$ USD)	3000	1735	580	255	110	49	22
DRAM bits per chip	256M	1G	4G	16G	64G	256G	1T
Wiring level	6	6–7	7	7–8	8–9	9	10
Supply voltage (V)	1.8–2.5	1.5–1.8	1.2–1.5	0.9–1.2	0.6–0.9	0.5–0.6	0.37–0.42
Power (W)	70	90	130	160	170	175	183

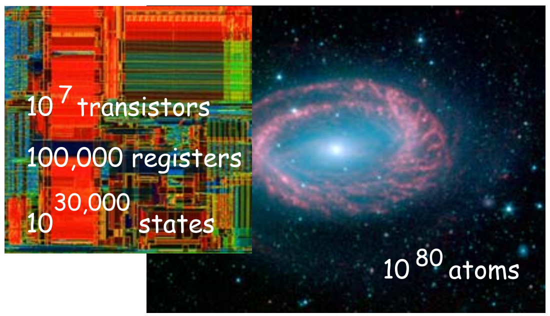
- Source: International Technology Roadmap for Semiconductors, Nov, 2002. http://www.itrs.net/ntrs/publntrs.nsf
- Deep submicron technology: node (**feature size**) < 0.25 μm
- Nanometer Technology: node < 0.1 μm

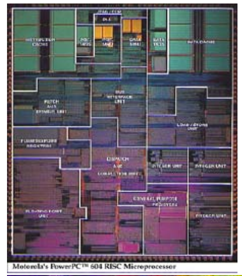
Nanometer Design Challenges

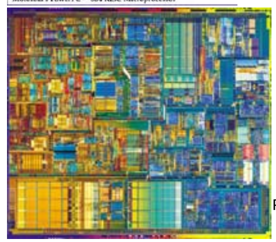
- In 2005, feature size ≈ 0.1 μm, μP frequency ≈ 3.5 GHz, die size ≈ 520 mm^2, μP transistor count per chip ≈ 200M, wiring level ≈ 8 layers, supply voltage ≈ 1 V, power consumption ≈ 160 W.
  - **Chip complexity**
    - **effective design and verification methodology? more efficient optimization algorithms? time-to-market?**
  - **Power consumption**
    - **power & thermal issues?**
  - **Supply voltage**
    - **signal integrity (noise, IR drop, etc)?**
  - **Feature size, dimension**
    - **sub-wavelength lithography (impacts of process variation)? noise? wire coupling? reliability? manufacturability? 3D layout?**
  - **Frequency**
    - **interconnect delay? electromagnetic field effects? timing closure?**

Design Complexity Challenges

- Design issues
  - Design space exploration
  - More efficient optimization algorithms
- Verification issues
  - State explosion problem
  - For modern designs, about 60%-80% of the overall design time was spent on verification; 3-to-1 head count ratio between verification engineers and logic designers

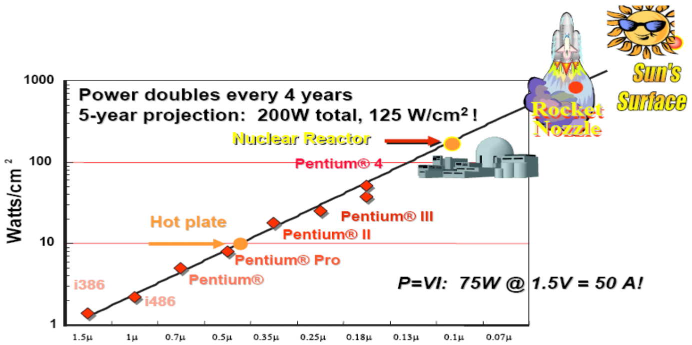



PowerPC 604


Pentium 4

Power Dissipation Challenges

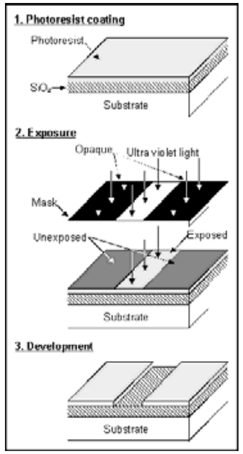
- Power density increases exponentially!

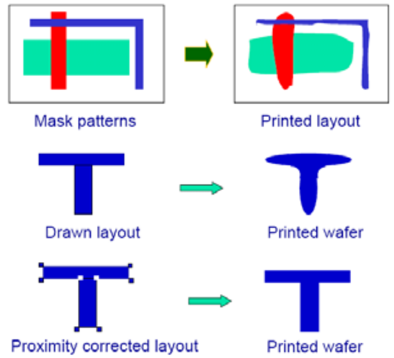


Fred Pollack, "New Microarchitecture Challenges in the Coming Generations of CMOS Process Technologies," 1999 Micro32 Conference keynote. Courtesy Avi Mendelson, Intel.

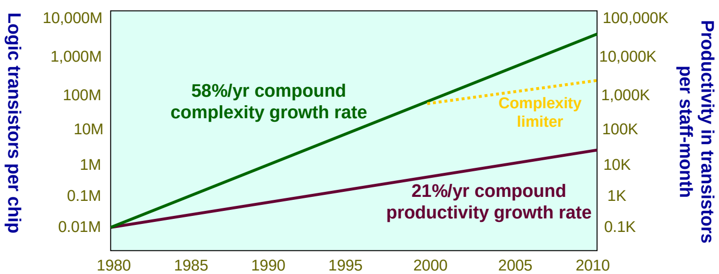
Semiconductor Fabrication Challenges

- Feature-size shrinking approaches physical limitation





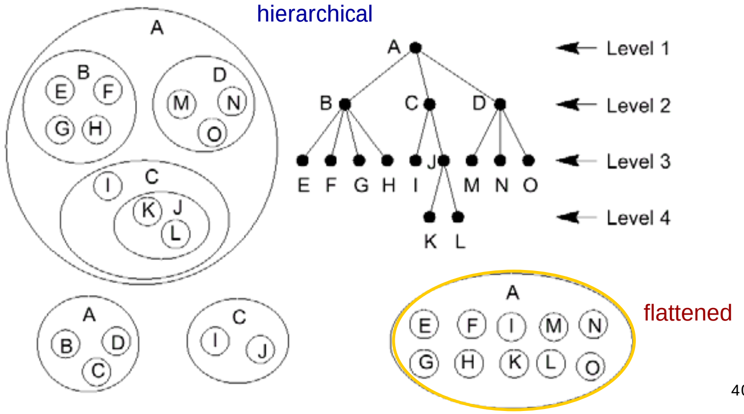
Design Productivity Challenges



- Human factors may limit design more than technology
- Keys to solve the productivity crisis: hierarchical design, abstraction, **CAD (tool & methodology)**, IP reuse, etc.

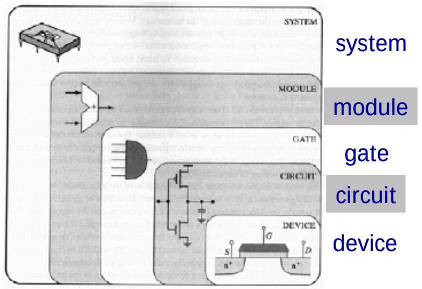
Cope with Complexity

- Hierarchical design
  - Design cannot be done in one step ⇒ partition the design hierarchically
  - *Hierarchy:* something is composed of simpler things

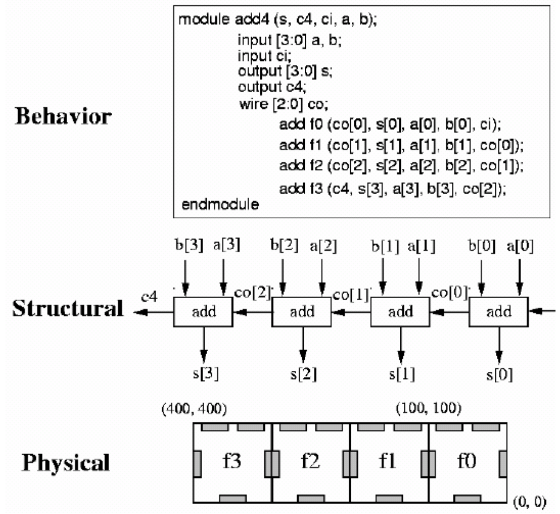


Cope with Complexity

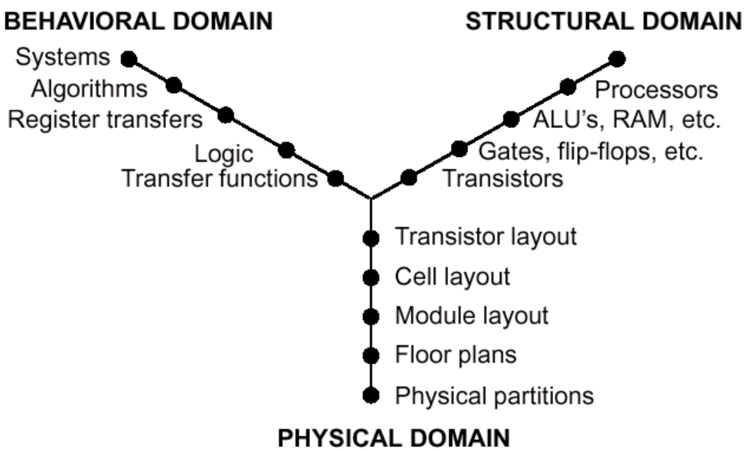
- Abstraction
  - Trim away unnecessarily detailed info at proper abstract levels
  - Design domains:
    - *Behavioral:* black box view
    - *Structural:* interconnection of subblocks
    - *Physical:* layout properties
    - Each design domain has its own hierarchy



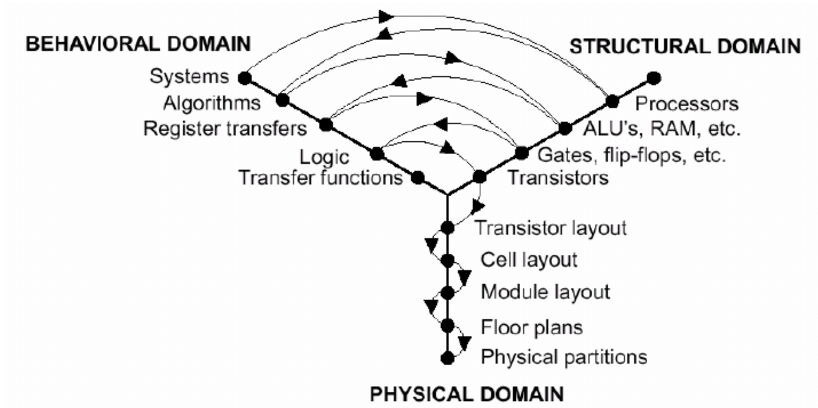
Three Design Views



Gajski's Y-Chart



Top-Down Structural Design

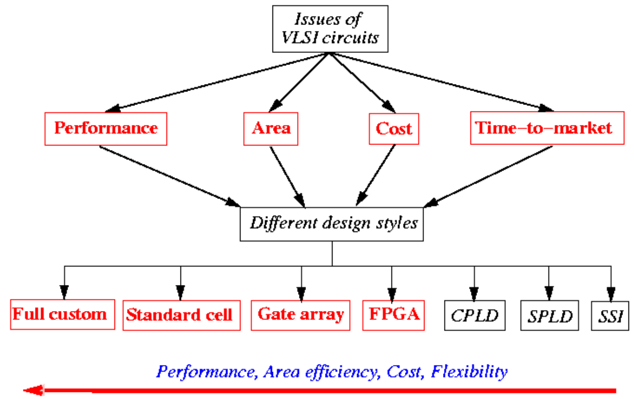


Design Styles

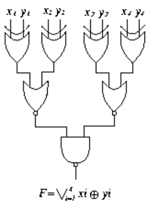
- There are various design styles:
  - Full custom, standard cell, sea of gates, FPGA, etc.
- Why having different design styles?

Design Styles

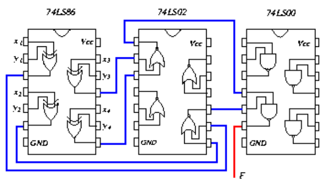
- Specific design styles shall require specific CAD tools



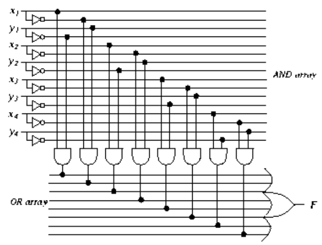
SSI/SPLD Design Style



(a) 4–bit comparator.



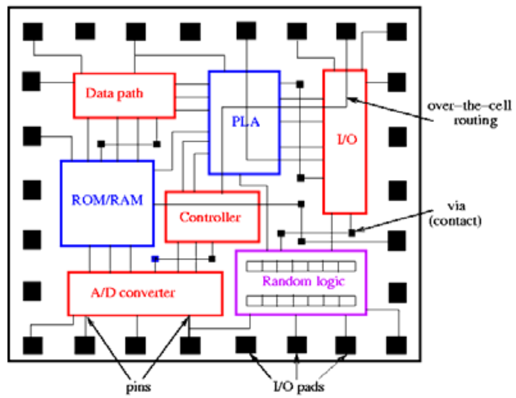
(b) SSI implementation.



(c) SPLD (PLA) implementation.

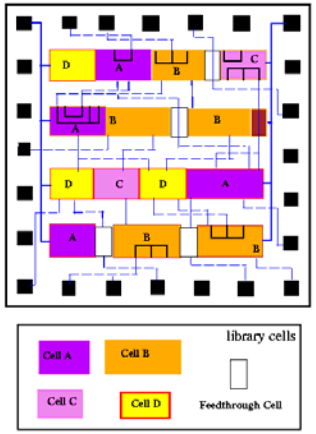
Full Custom Design Style

- Designers can control the shape of all mask patterns
- Designers can specify the design up to the level of individual transistors

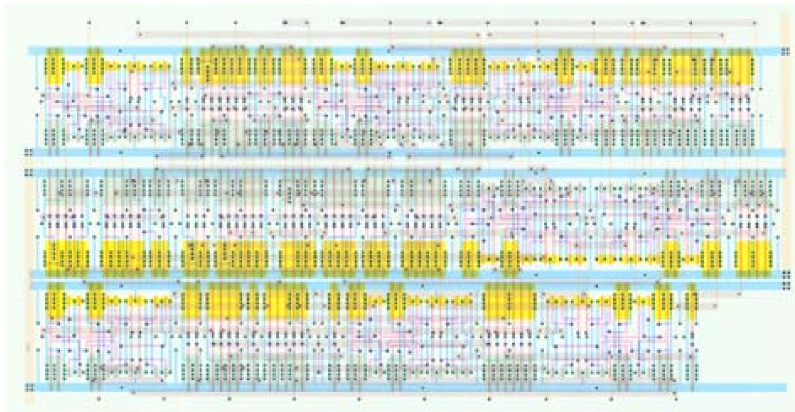


Standard Cell Design Style

- Selects pre-designed cells (of same height) to implement logic

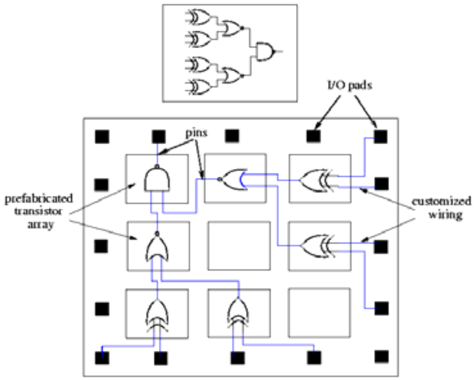


Standard Cell Example



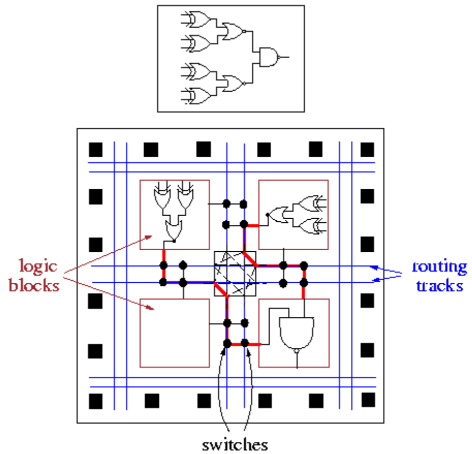
Gate Array Design Style

- Prefabricates a transistor array
- Needs wiring customization to implement logic



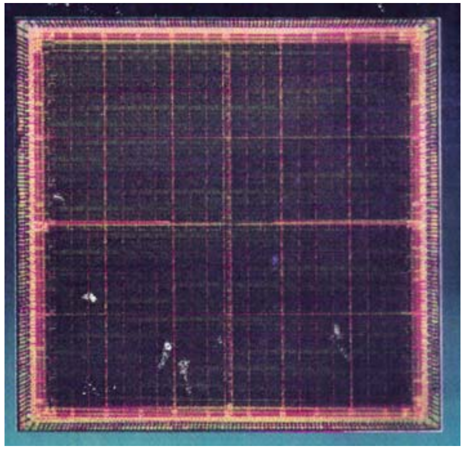
FPGA Design Style

- Logic and interconnects are both prefabricated
- Illustrated by a symmetri array-based FPGA



Prefabricated all chip components

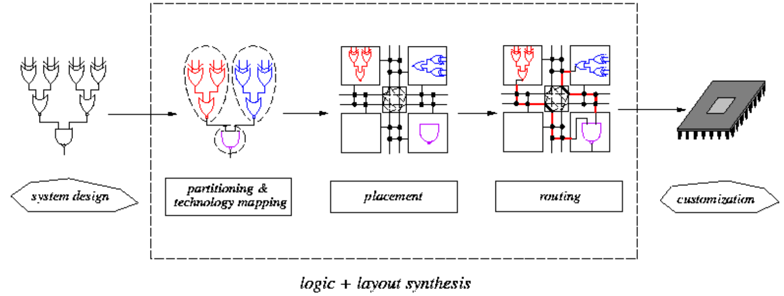
Array-Based FPGA Example



Lucent 15K ORCA FPGA
- 0.5 um 3LM CMOS
- 2.45 M Transistors
- 1600 Flip-flops
- 25K bit user RAM
- 320 I/Os

FPGA Design Process

- Illustrated by a symmetric array-based FPGA
- No fabrication is needed



Comparisons of Design Styles

	Full custom	Standard cell	Gate array	FPGA	SPLD
Cell size	variable	fixed height*	fixed	fixed	fixed
Cell type	variable	variable	fixed	programmable	programmable
Cell placement	variable	in row	fixed	fixed	fixed
Interconnections	variable	variable	variable	programmable	programmable

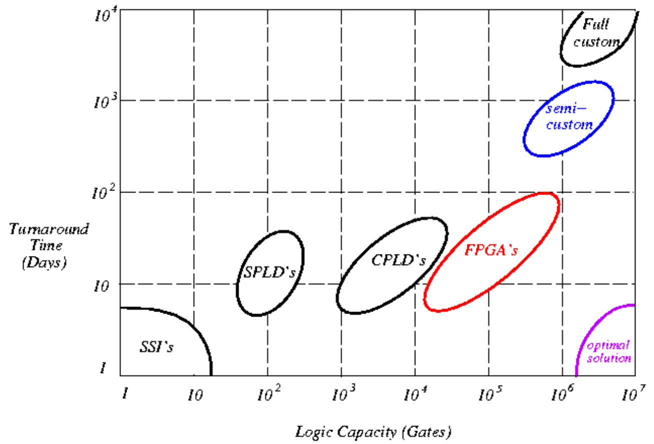
* Uneven height cells are also used.

Comparisons of Design Styles

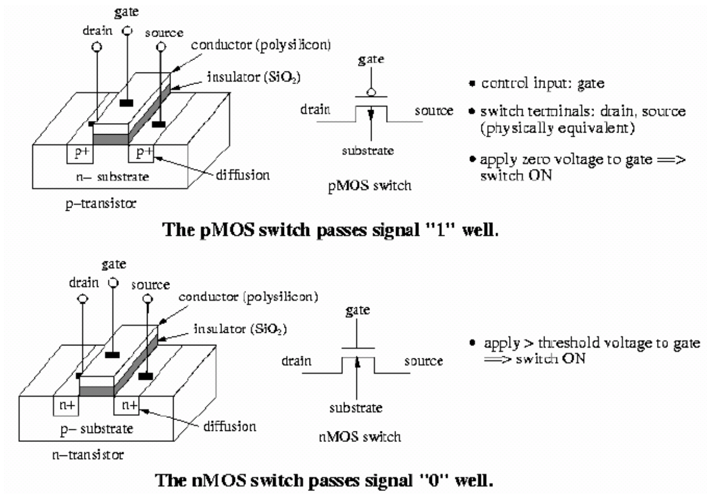
	Full custom	Standard cell	Gate array	FPGA	SPLD
Fabrication time	– – –	– –	+	+++	++
Packing density	+++	++	+	– –	– – –
Unit cost in large quantity	+++	++	+	– –	–
Unit cost in small quantity	– – –	– –	+	+++	++
Easy design and simulation	– – –	– –	–	++	+
Easy design change	– – –	– –	–	++	++
Accuracy of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

+ desirable; – not desirable

Design Style Trade-offs

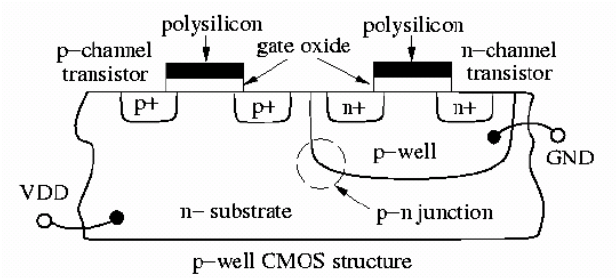


MOS Transistors

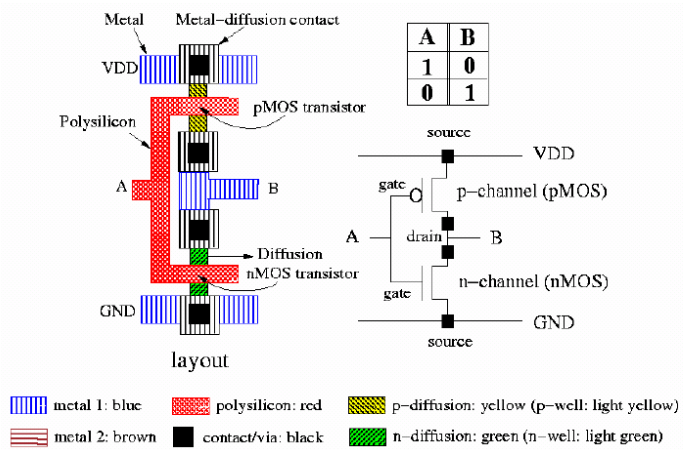


Complementary MOS (CMOS)

- The most popular VLSI technology (v.s. BiCMOS, nMOS)
- CMOS uses both *n*-channel and *p*-channel transistors
- Advantages: lower power dissipation, higher regularity, more reliable performance, higher noise margin, larger fanout, etc.
- Each type of transistor must sit in a material of the complementary type (the reverse-biased diodes prevent unwanted current flow)

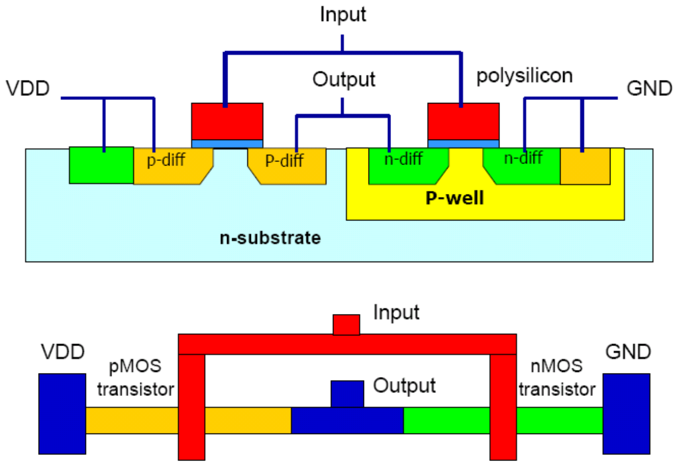


CMOS Inverter

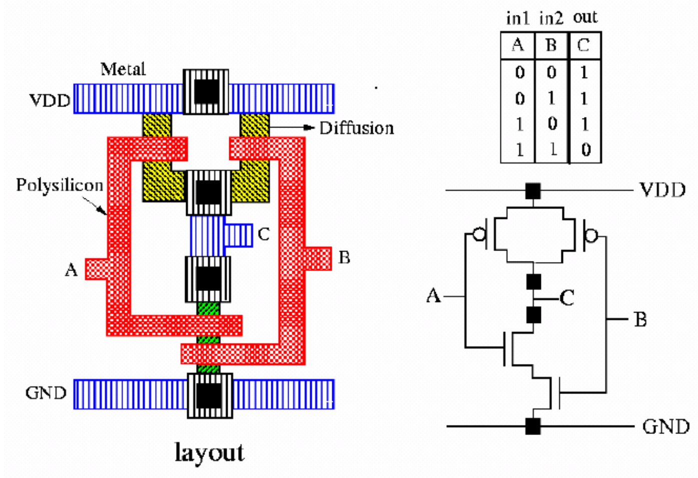


A	B
1	0
0	1

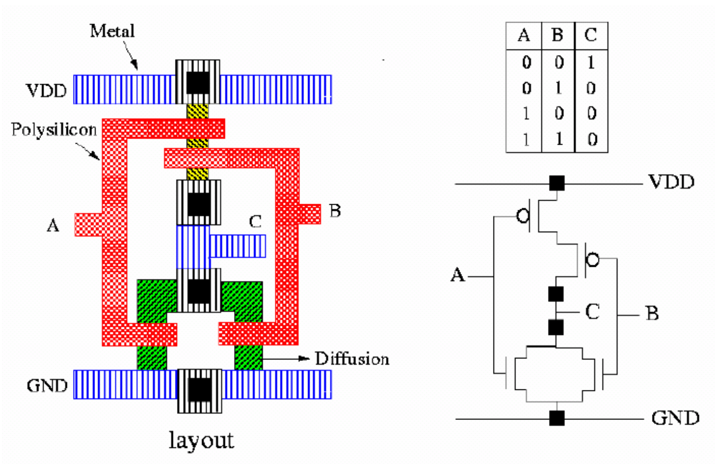
CMOS Inverter Cross Section



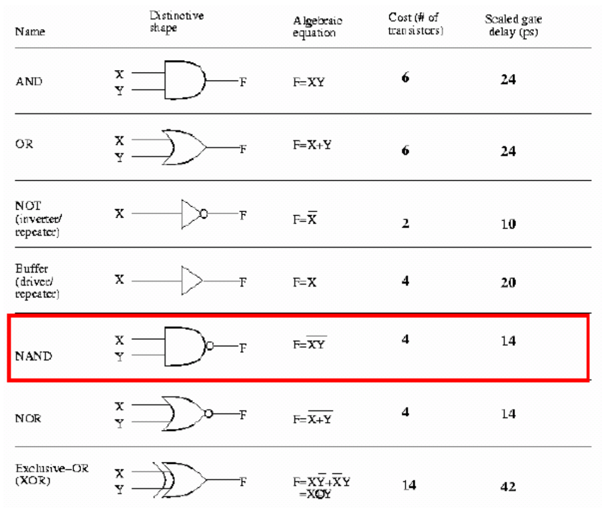
CMOS NAND Gate



CMOS NOR Gate



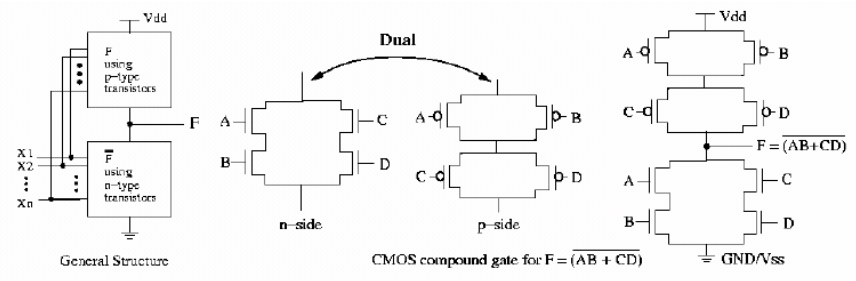
Basic CMOS Logic Library



Name	Distinctive shape	Algebraic equation	Cost (# of transistors)	Scaled gate delay (ps)
AND		$F = XY$	6	24
OR		$F = X + Y$	6	24
NOT (inverter/repeater)		$F = \overline{X}$	2	10
Buffer (driver/repeater)		$F = X$	4	20
NAND		$F = \overline{XY}$	4	14
NOR		$F = \overline{X+Y}$	4	14
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y = X \oplus Y$	14	42

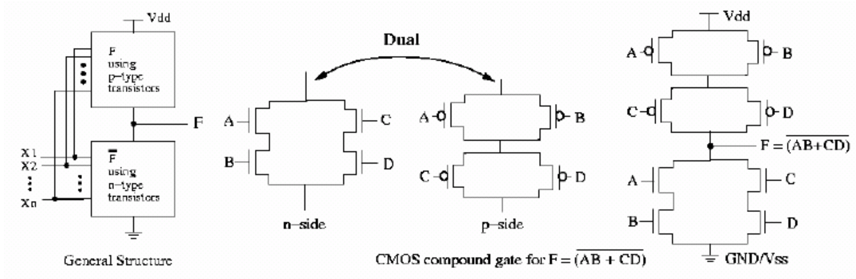
Construction of Compound Gates (1/2)

- Example: $F=\overline{A\cdot B+C\cdot D}$
- Step 1 (**n**-network): **Invert** *F* to derive *n*-network
  - $(\overline{F}=A\cdot B+C\cdot D)$
- Step 2 (**n**-network): Make connections of transistors:
  - AND ⇔ Series connection
  - OR ⇔ Parallel connection



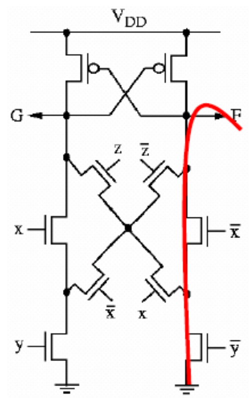
Construction of Compound Gates (2/2)

- Step 3 (**p**-network): Expand *F* to derive *p*-network
  - $(F=\overline{AB+CD}=\overline{AB}\cdot\overline{CD}=(\overline{A}+\overline{B})\cdot(\overline{C}+\overline{D}))$
  - **each input is inverted**
- Step 4 (**p**-network): Make connections of transistors (same as Step 2).
- Step 5: Connect the *n*-network to GND (typically, 0V) and the *p*-network to VDD (5V, 3.3V, or 2.5V, etc).



Complex CMOS Gate

- The functions realized by the *n* and *p* networks must be complementary, and one of the networks must conduct for every input combination
- Duality is not necessary



x	y	z	F	G
0	0	0	0	1
0	0	1	0	1
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

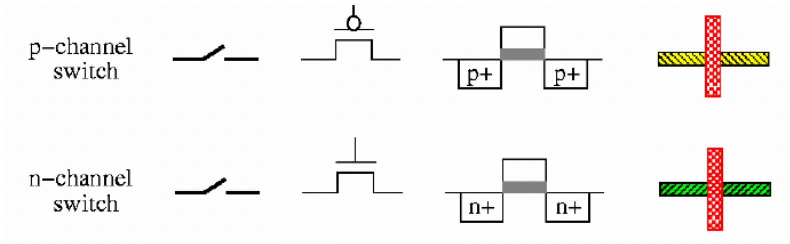
F = ?
G = ?

CMOS Properties

- There is always a path from one supply (VDD or GND) to the output.
- There is never a path from one supply to the other. (This is the basis for the low power dissipation in CMOS--virtually no static power dissipation.)
- There is a momentary drain of current (and thus power consumption) when the gate switches from one state to another.
  - Thus, CMOS circuits have dynamic power dissipation.
  - The amount of power depends on the switching frequency.

Stick Diagram

- Intermediate representation between the transistor level and the mask (layout) level.
- Gives topological information (identifies different layers and their relationship)
- Assumes that wires have no width.
- Possible to translate stick diagram automatically to layout with correct **design rules**.

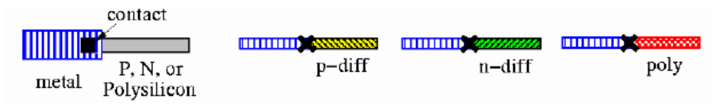


Stick Diagram

- When the same material (on the same layer) touch or cross, they are connected and belong to the same electrical node.

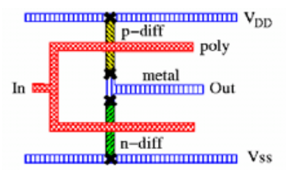
- When **polysilicon** crosses N or P **diffusion**, an N or P transistor is formed.
  - Polysilicon is drawn on top of diffusion.
  - Diffusion must be drawn connecting the source and the drain.
  - Gate is automatically self-aligned during fabrication.

- When a metal line needs to be connected to one of the other three conductors, a **contact** cut (**via**) is required.

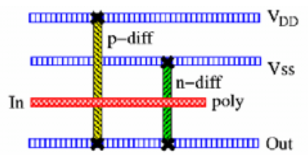


CMOS Inverter Stick Diagram

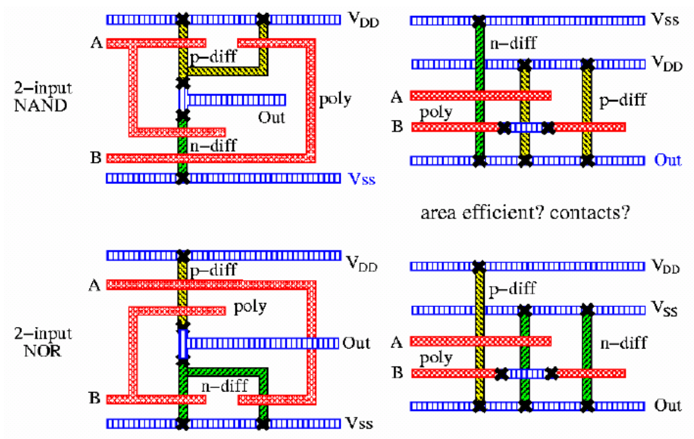
- Basic layout



- More area efficient layout



CMOS NAND/NOR Stick Diagram



Design Rules

- Layout rules are used for preparing the masks for fabrication.
- Fabrication processes have inherent limitations in accuracy.
- Design rules specify geometry of masks to optimize yield and reliability (trade-offs: area, yield, reliability).
- Three major rules:
  - **Wire width:** Minimum dimension associated with a given feature.
  - **Wire separation:** Allowable separation.
  - **Contact:** overlap rules.
- Two major approaches:
  - **"Micron" rules:** stated at micron resolution.
  - λ **rules:** simplified micron rules with limited **scaling** attributes.
- λ may be viewed as the size of minimum feature.
- Design rules represents a tolerance which insures very high probability of correct fabrication (not a hard boundary between correct and incorrect fabrication).
- Design rules are determined by experience.

MOSIS Layout Design Rules

- MOSIS design rules (SCMOS rules) are available at http://www.mosis.org
- 3 basic design rules: Wire width, wire separation, contact rule.
- MOSIS design rule examples

R1	Min active area width	3 λ
R3	Min poly width	2 λ
R4	Min poly spacing	2 λ
R5	Min gate extension of poly over active	2 λ
R8	Min metal width	3 λ
R9	Min metal spacing	3 λ
R10	Poly contact size	2 λ
R11	Min poly contact spacing	2 λ

SCMOS Design Rules