



# EECS 168 Introduction to VLSI Design

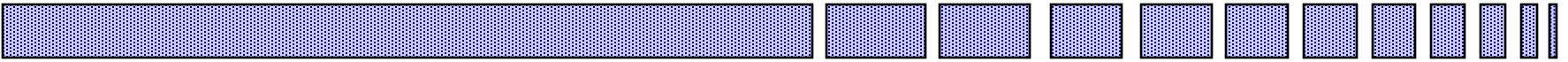
**Dr. Sheldon Tan**

[www.ece.ucr.edu/~stan](http://www.ece.ucr.edu/~stan)

Email: stan@ece.ucr.edu



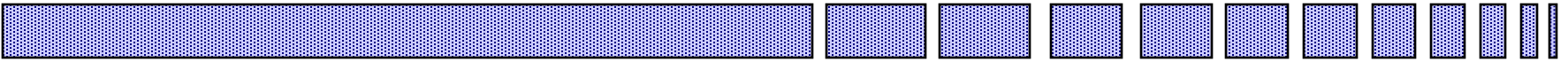
# Overview



- Why VLSI?
- Moore's Law.
- The VLSI design process.
- IP-based design.



# Why VLSI?

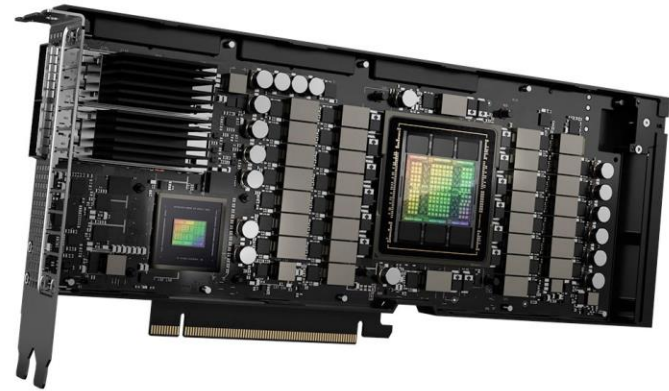


- Integration improves the design:
  - lower parasitics = higher speed;
  - lower power;
  - physically smaller for smaller form factor
- Integration reduces manufacturing cost-  
(almost) no manual assembly.



# VLSI and you

- Microprocessors:
  - personal computers;
  - microcontrollers.
- DRAM/SRAM.
- Special-purpose processors.
- Hardware for domain-specific computing and machine learning
  - Digital design is going through a renaissance
  - Machine learning, Image and video/vision computing, TPU, IPU, xPU
- GPU for emerging Generative AI...



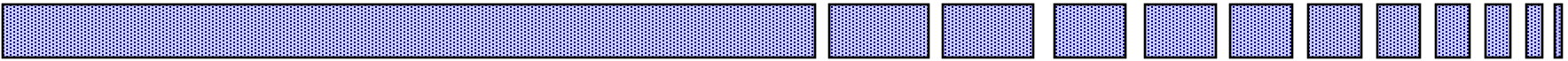
Nvidia H100 GPU 2023



Google TPU



# Semiconductor industry trend

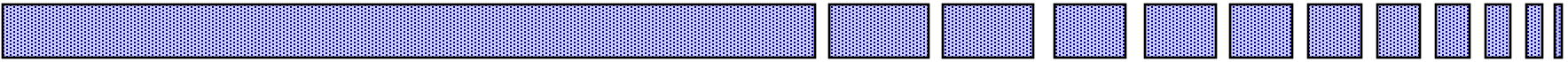


- Developing system-on-chips (SoCs) on advanced fabrication technologies is becoming more expensive.
- Many companies are opting to design custom SoCs despite the higher costs, differentiating themselves from rivals.
  - Companies like Google and Amazon are creating custom silicon to optimize performance, reduce costs, and gain differentiation.
- Companies in the data infrastructure space, including cloud and 5G, are increasingly adopting custom silicon
- Established players like AMD and Intel provide off-the-shelf products, while newcomers challenge industry giants.

A screenshot of an EE Times article. The header is red with 'EE Times' in white. Below the header is a navigation bar with links: HOME, NEWS, PERSPECTIVES, DESIGNLINES, PODCASTS, EDUCATION, STORE, and SP. The article title is 'The Golden Age of Custom Silicon Draws Near' in large black font. Below the title is 'EXCLUSIVE REPORT' and 'By Anton Shilov 07.26.2023'. There are social sharing buttons for Facebook, Twitter, and LinkedIn. Below the article is an advertisement placeholder with the text 'Ad served by Google' and buttons for 'Ad options', 'Send feedback', and 'Why this ad?'. Below the ad is a quote: '—First in a three-part series' and a paragraph: 'Several curious trends can be observed in the semiconductor industry today. On the one hand,'.



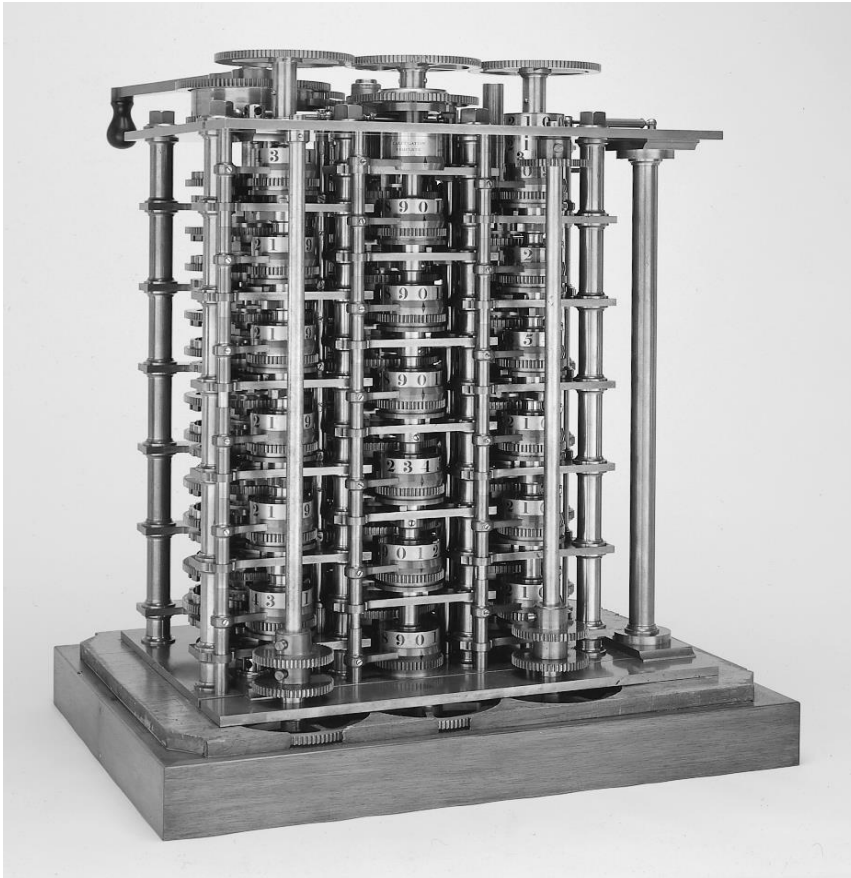
# Quote from Steven Job's introduction of iphone in 2007



- Why hardware design becomes so important, **again!**
- “People who are really serious about software should make their own hardware” – Alan Kay, 1982, pioneer on object-oriented programming
- Companies like Apple approves that hardware-software co-design can leads to much better products and company profits
- Microsoft's surface series, Google's Pixel smartphone, Samsung's Galaxy smartphone



# The First Computer



**The Babbage  
Difference Engine  
(1832)**

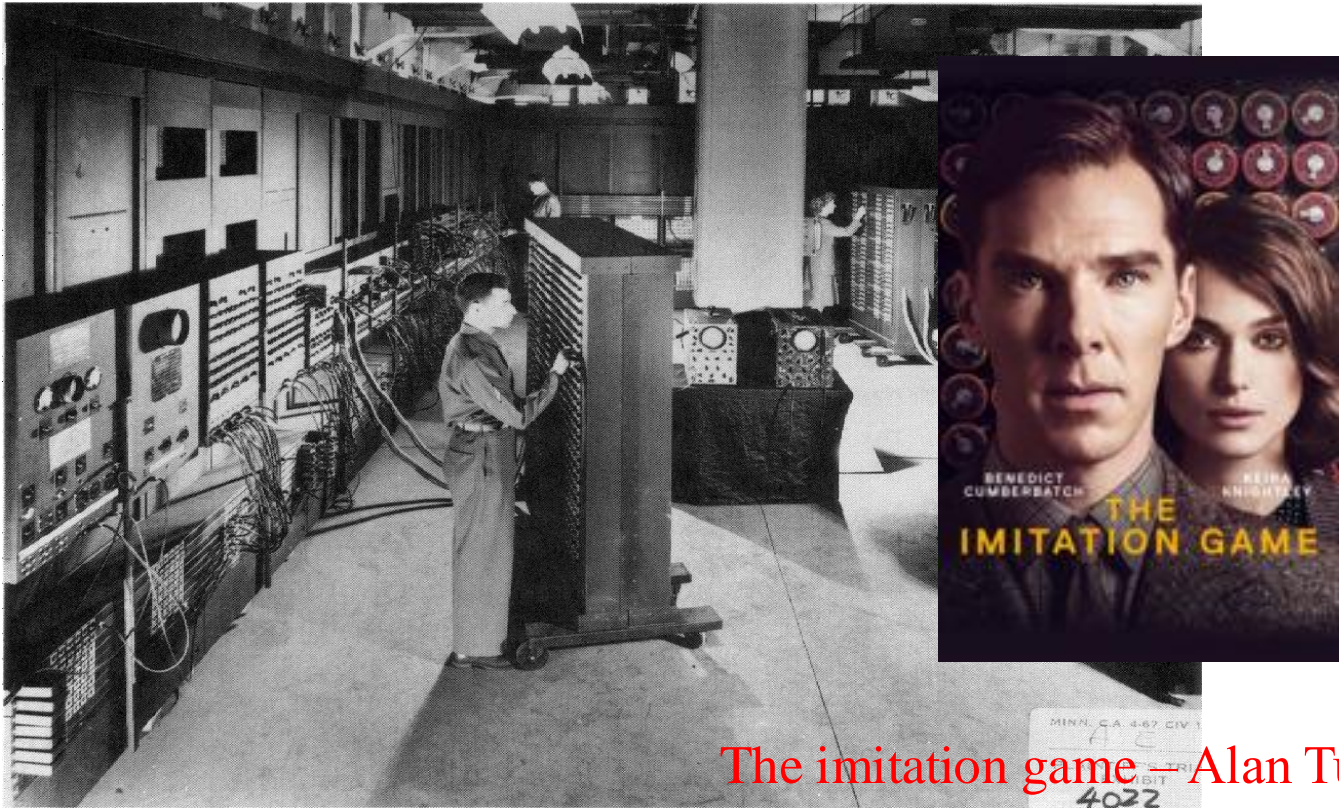
**25,000 parts**

**cost: £17,470**





# ENIAC - The first electronic computer (1946)

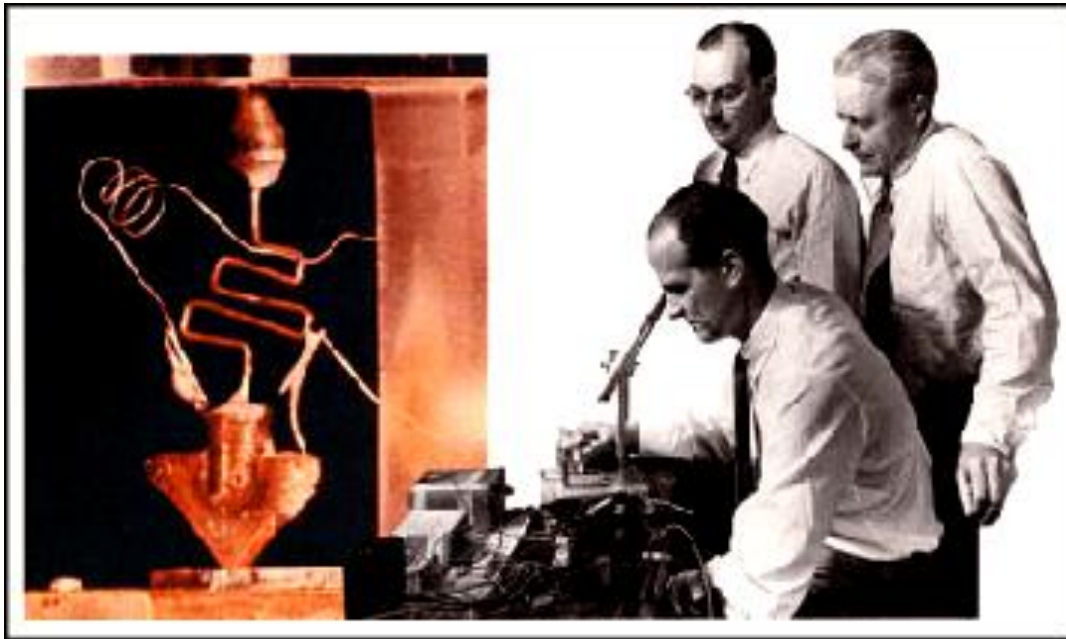


The imitation game — Alan Turing's machine for cracking Enigma code from Germany





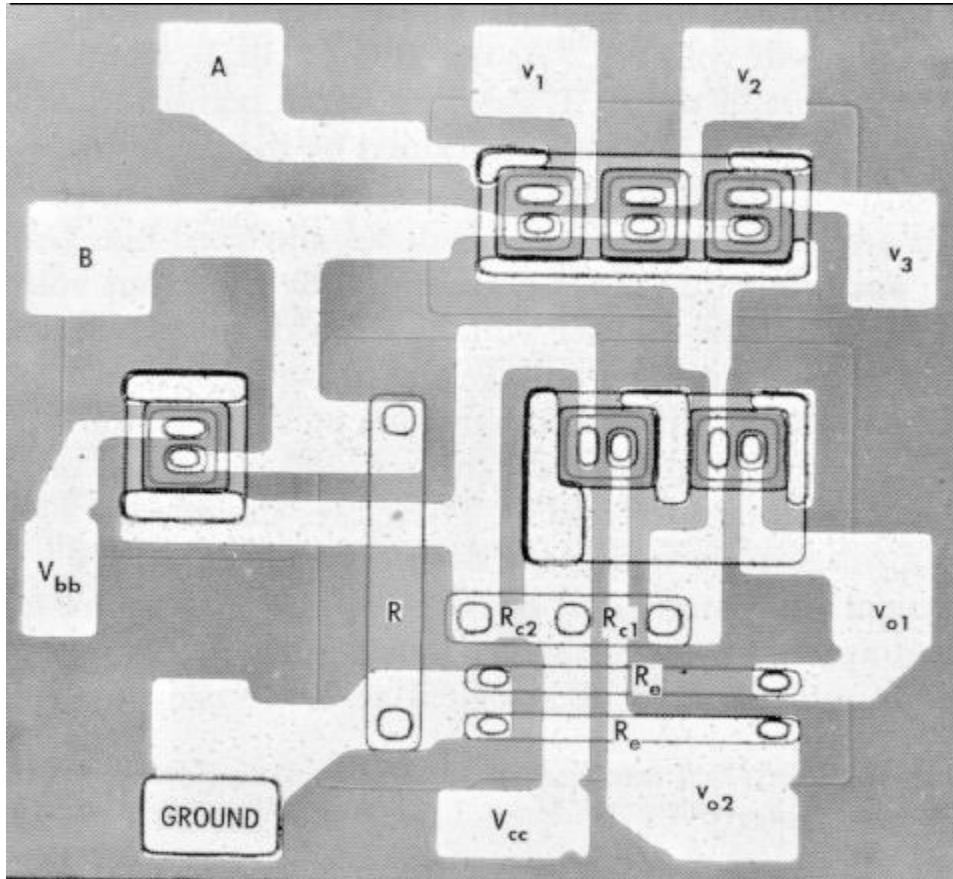
# The Transistor Revolution



First transistor invented by  
[Bardeen, Brattain & Shockley](#) in  
Bell Labs, 1948



# The First Integrated Circuits

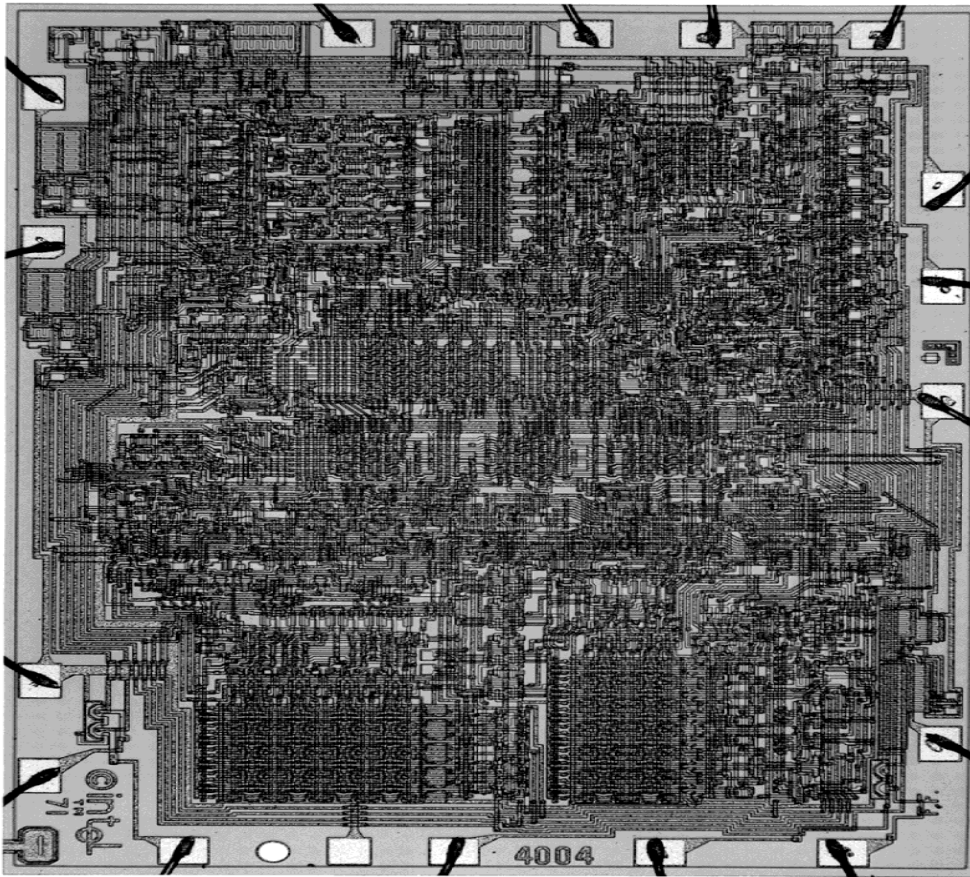


Bipolar logic  
1960's

ECL 3-input Gate  
Motorola 1966



# Intel 4004 Micro-Processor



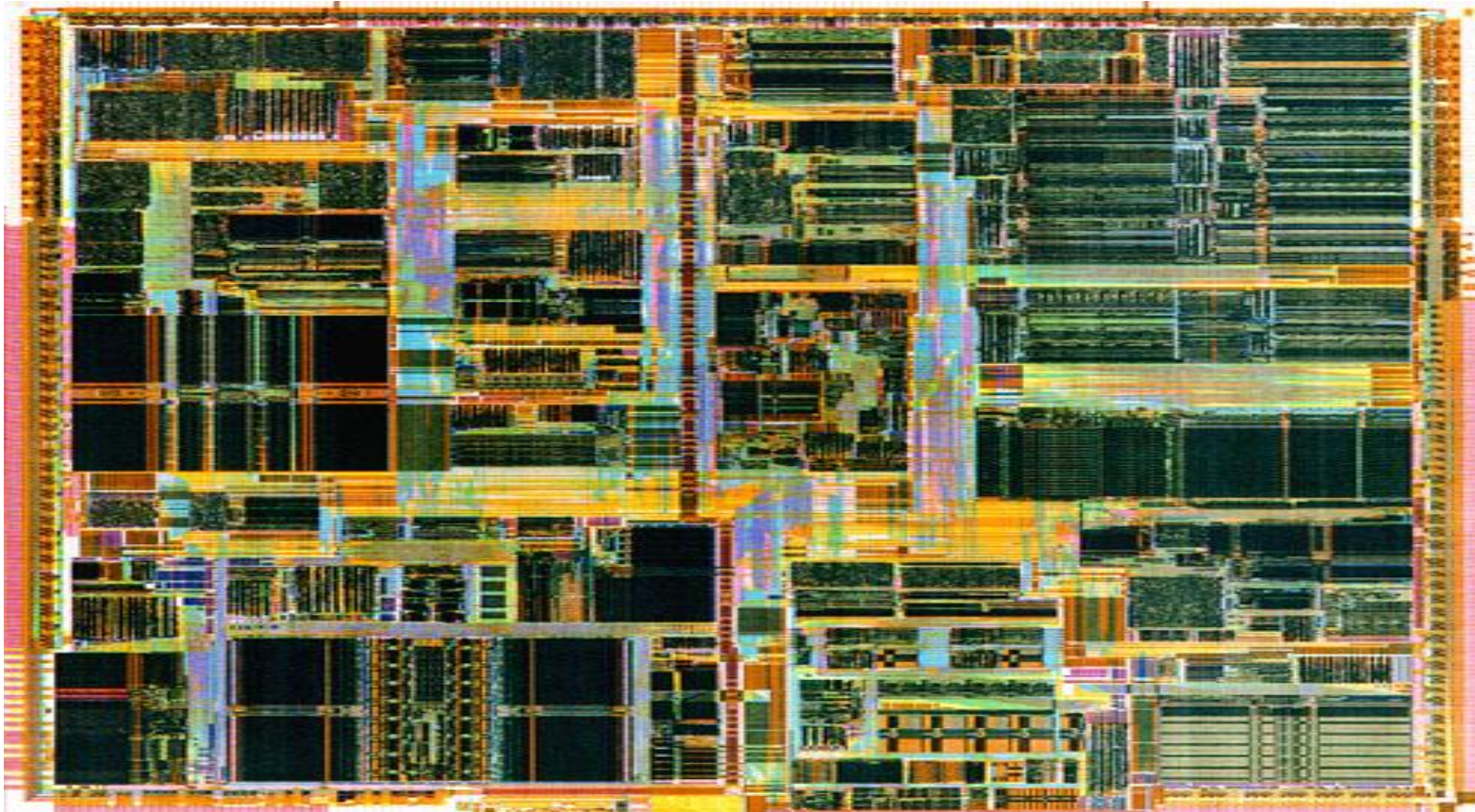
1971  
1000 transistors  
1 MHz operation



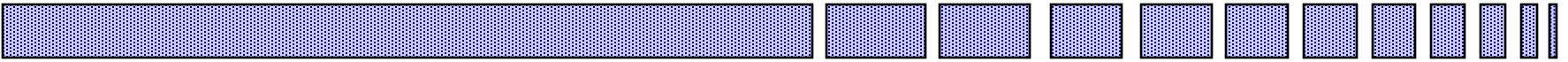
Intel's founders: Robert Noyce,  
Andy Grove, Golden M



# Intel Pentium (IV) microprocessor



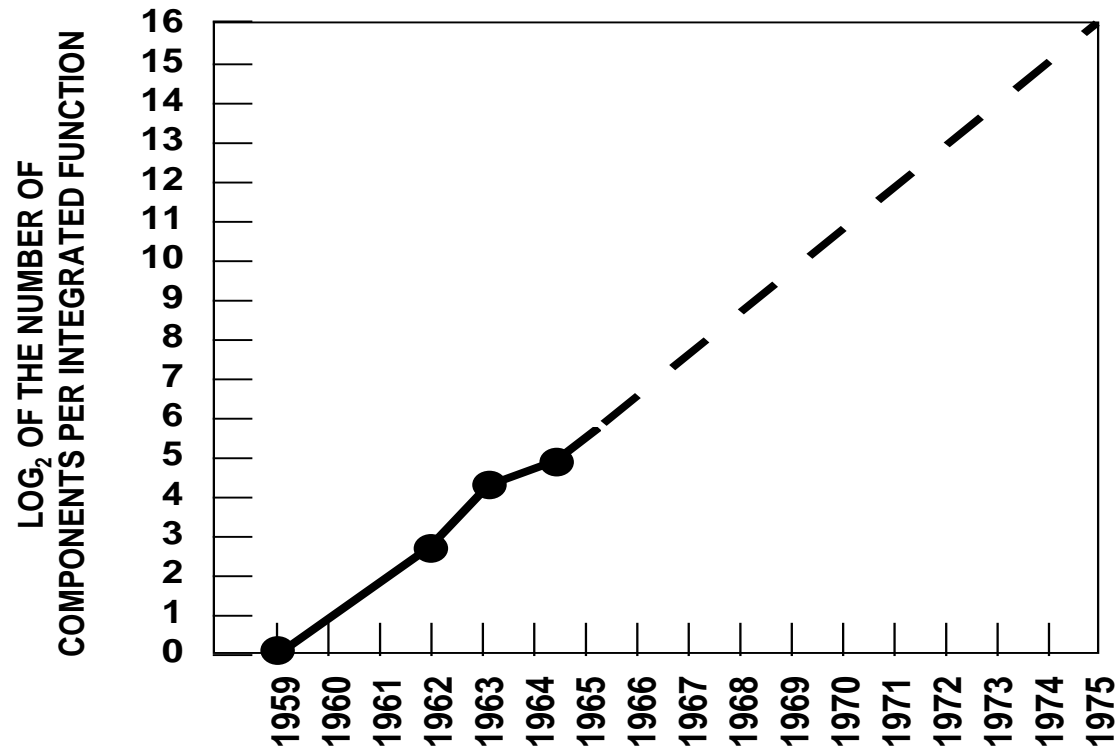
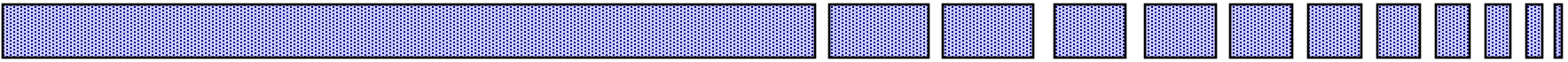
# Moore's Law



- In 1965, Gordon Moore noted that the number of transistors on a chip doubled every 18 to 24 months.
- He made a prediction that semiconductor technology will double its effectiveness every 18 months



# Moore's Law

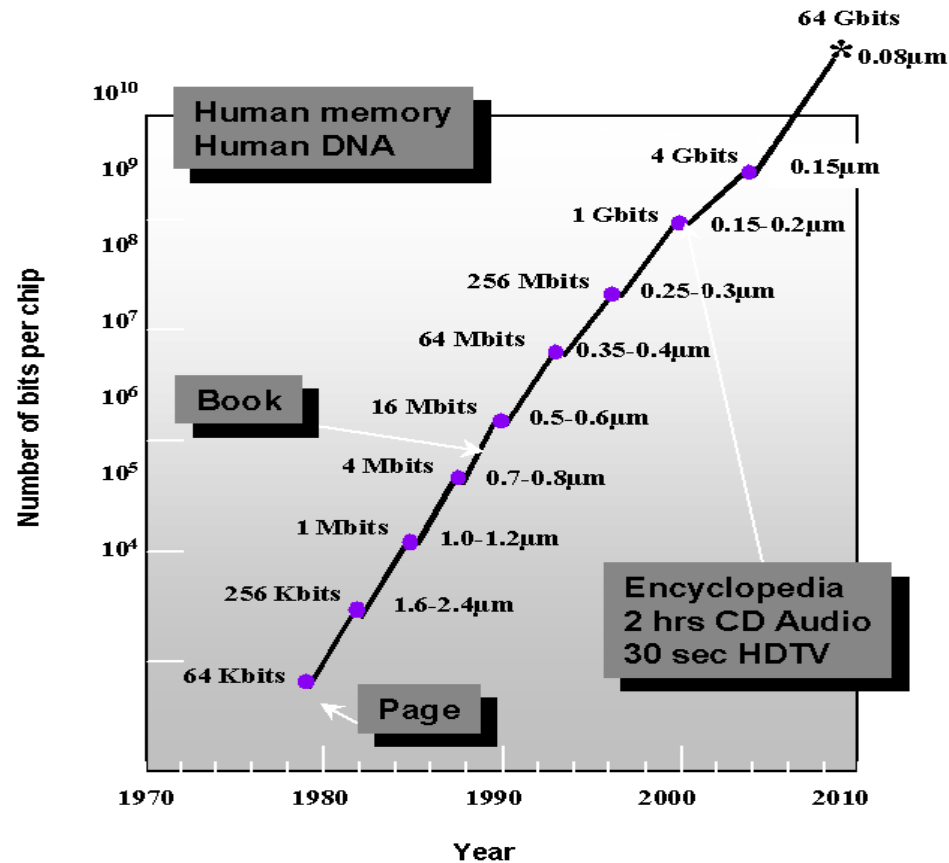
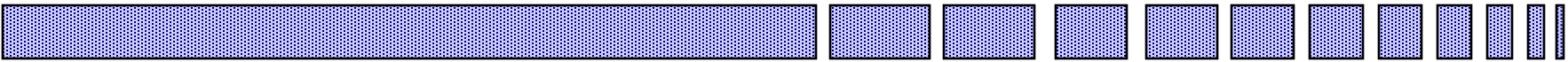


Electronics, April 19, 1965.

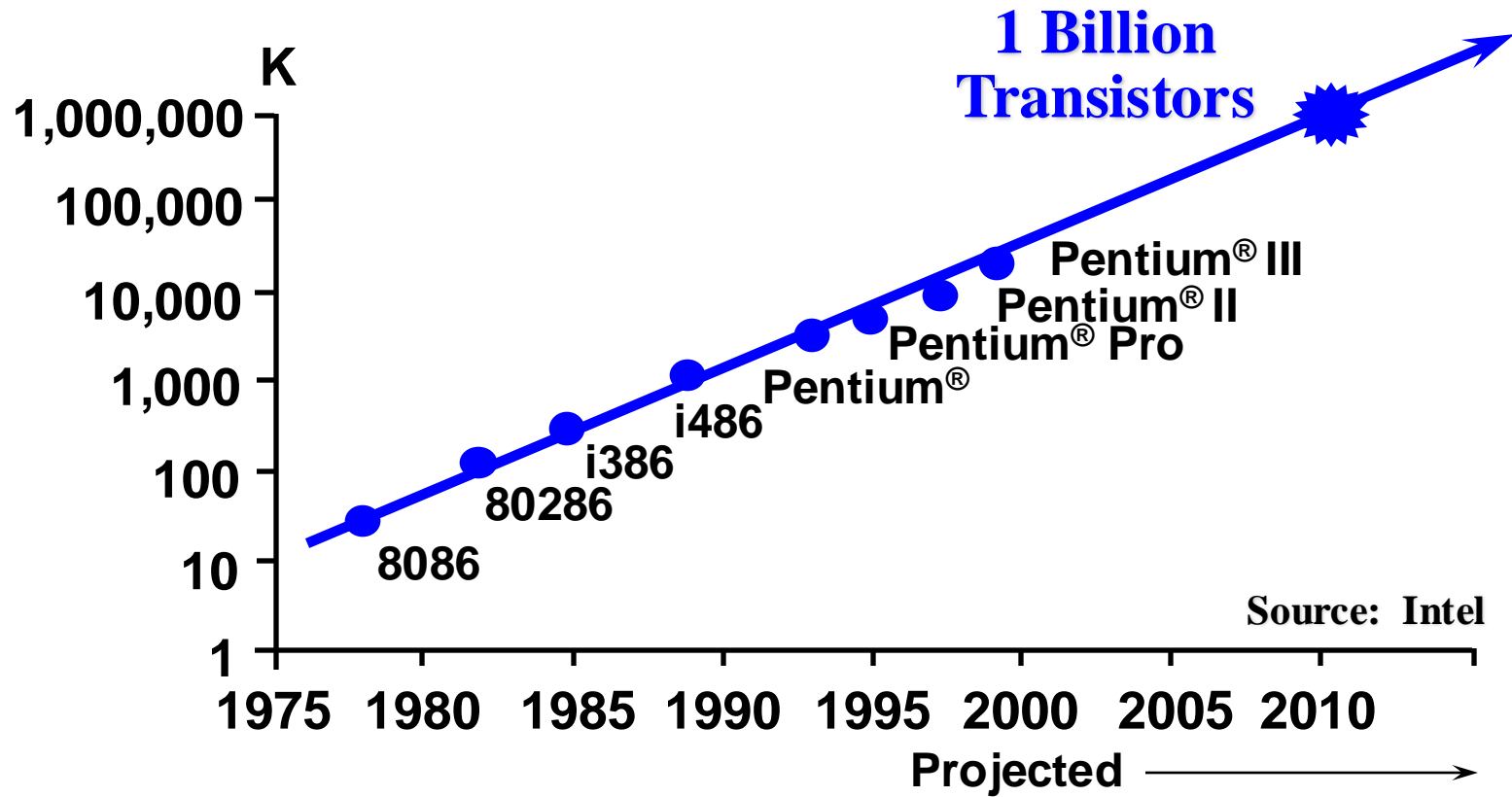
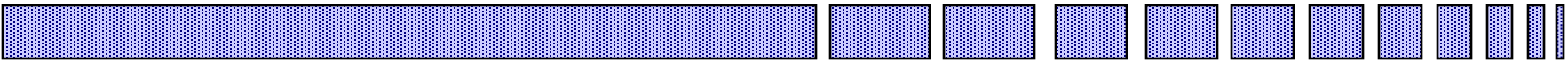




# Evolution in Complexity



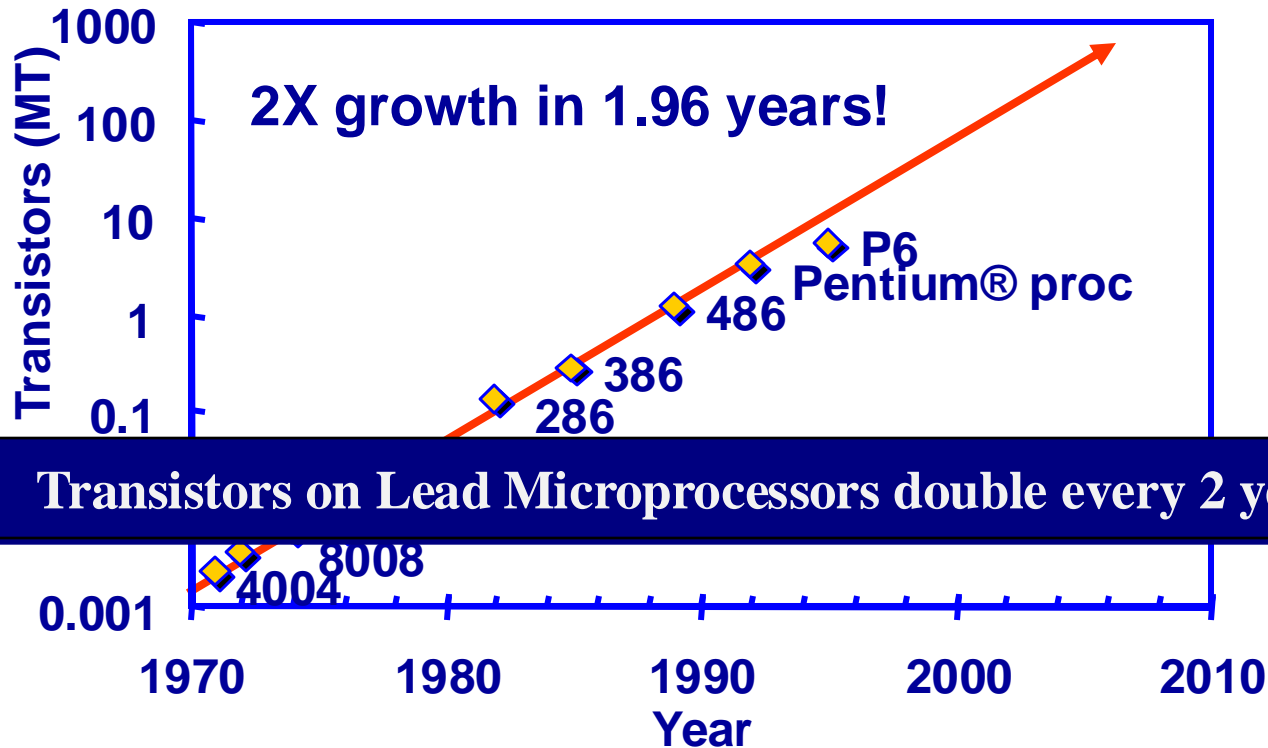
# Transistor Counts



Courtesy, Intel



# Moore's law in Microprocessors

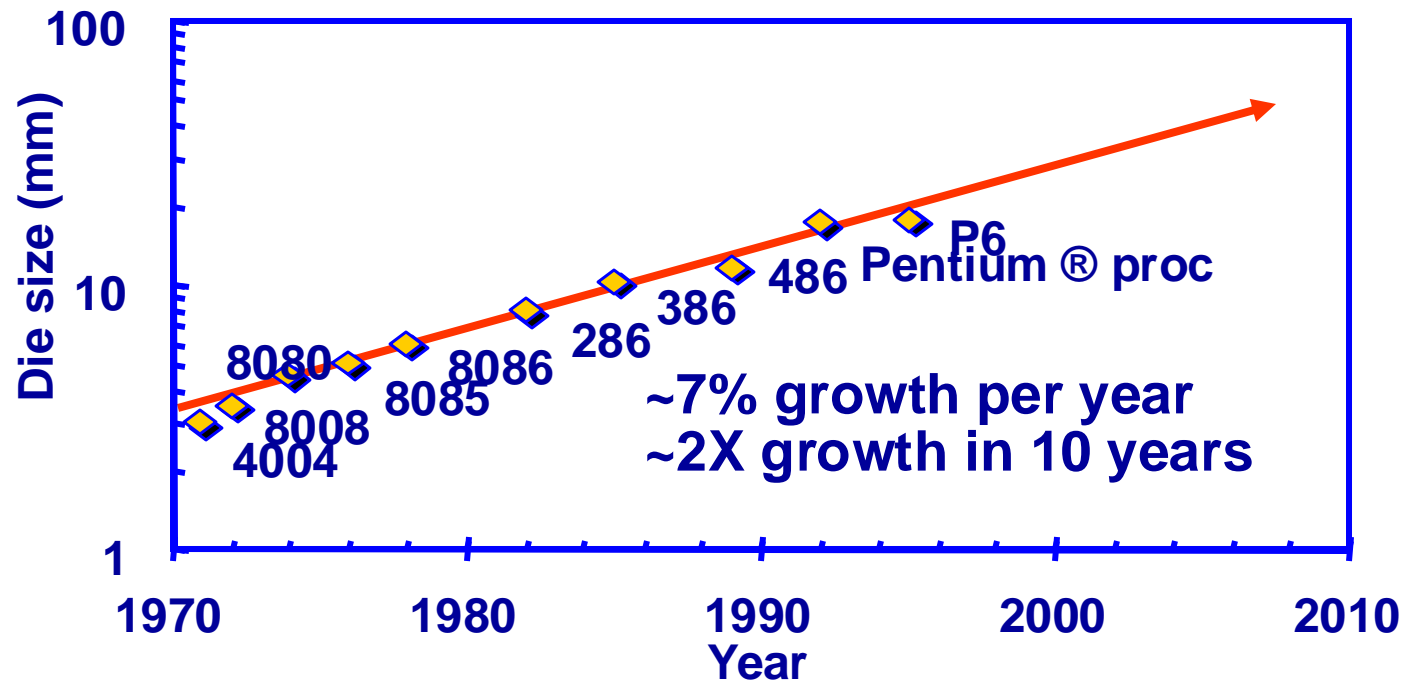


Transistors on Lead Microprocessors double every 2 years

Courtesy, Intel



# Die Size Growth

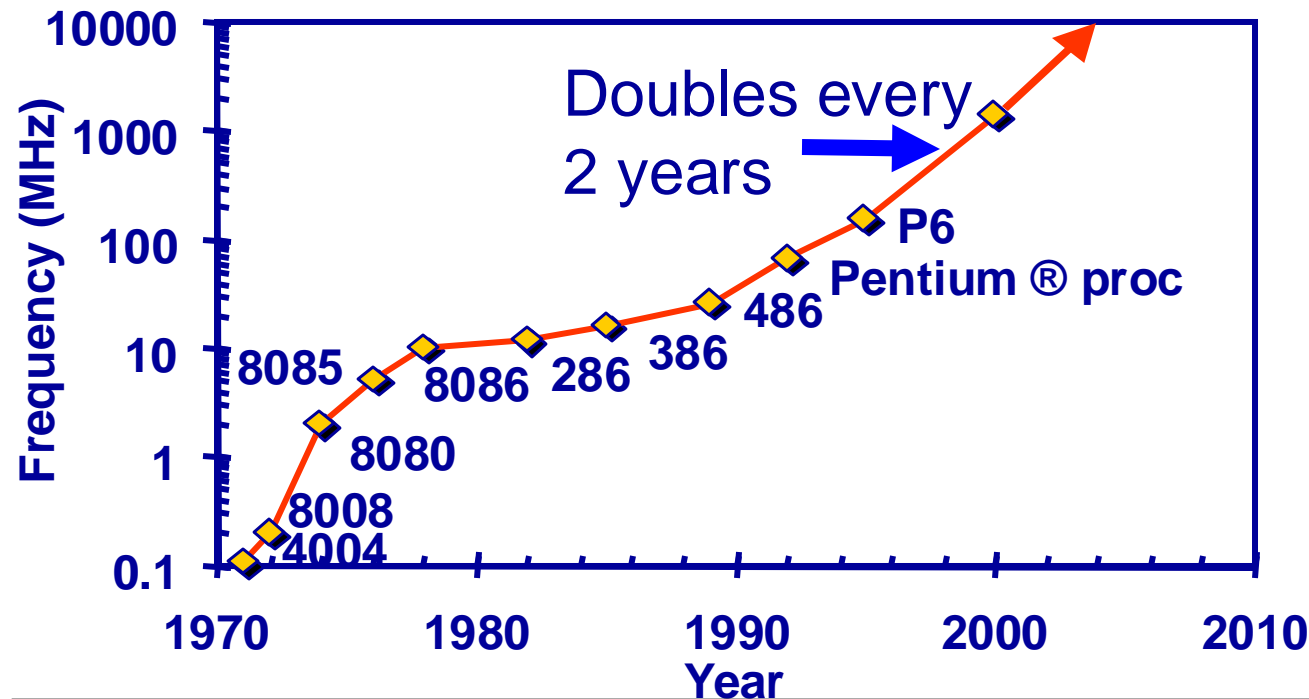


Die size grows by 14% to satisfy Moore's Law

Courtesy, Intel



# Frequency

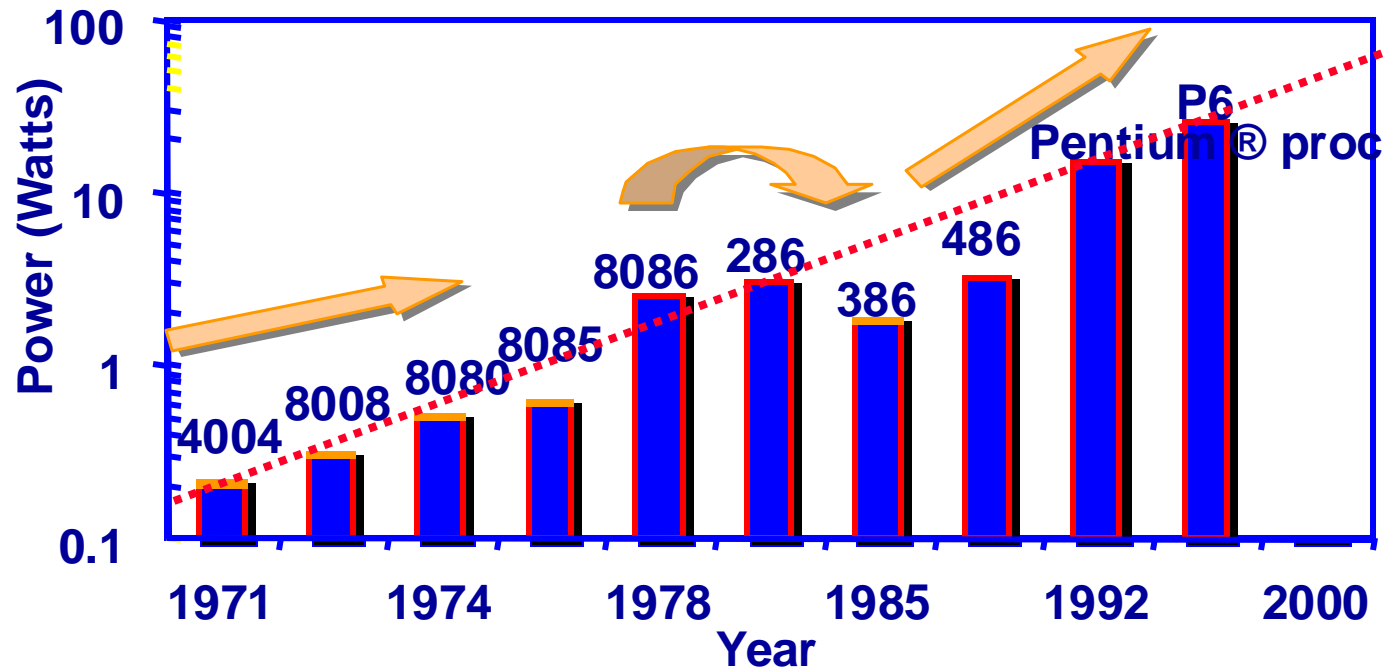


**Lead Microprocessors frequency doubles every 2 years**

**Courtesy, Intel**



# Power Dissipation



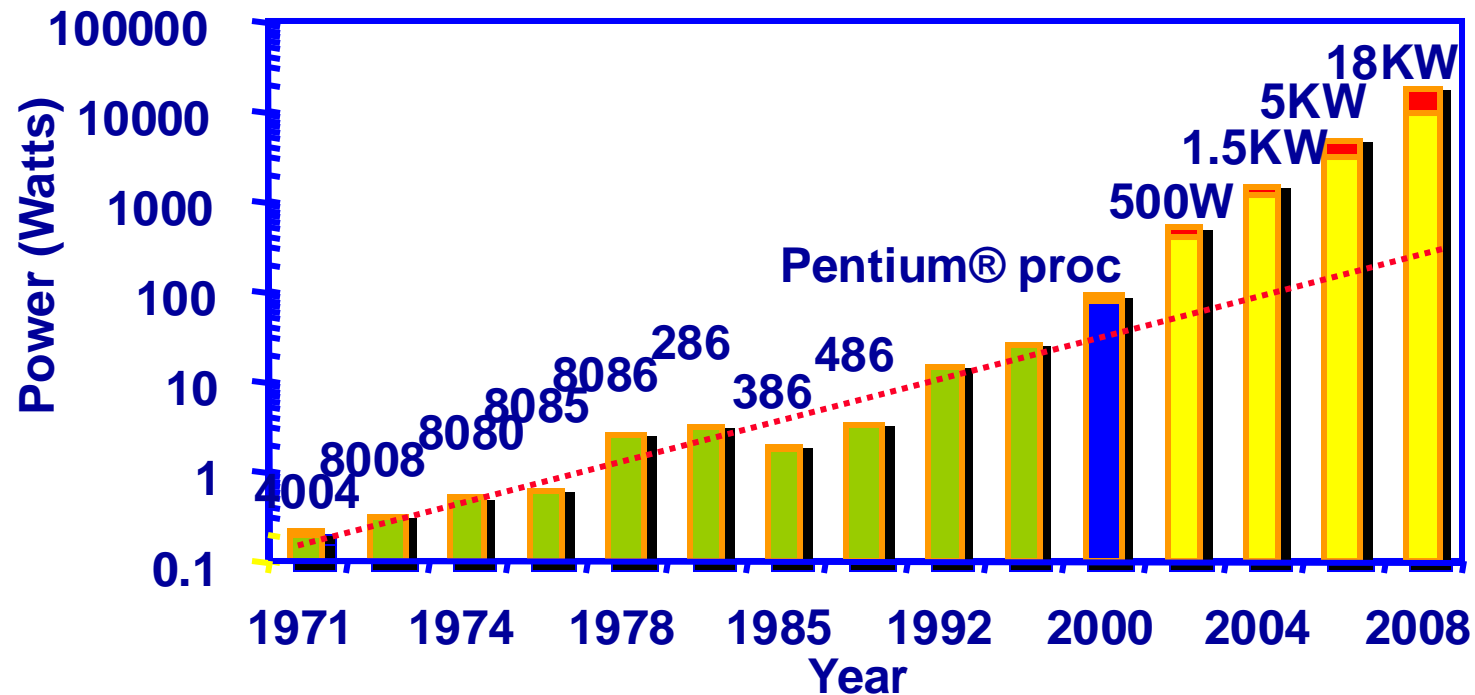
Lead Microprocessors power continues to increase

Courtesy, Intel





# Power will be a major problem

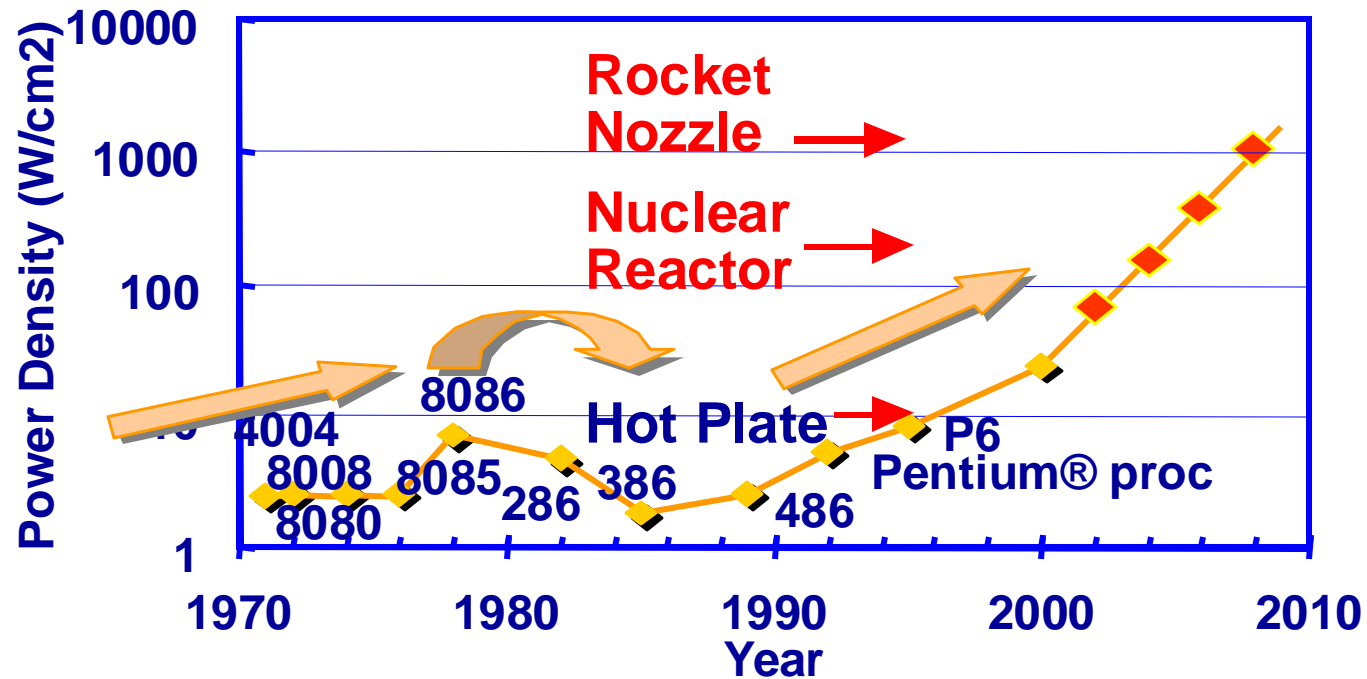


**Power delivery and dissipation will be prohibitive**

Courtesy, Intel



# Power Density

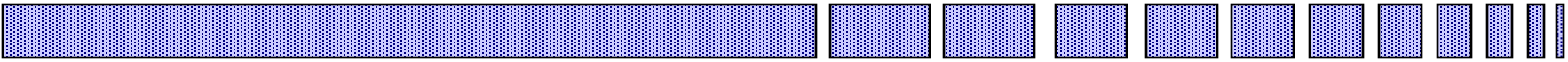


**Power density too high to keep junctions at low temp**

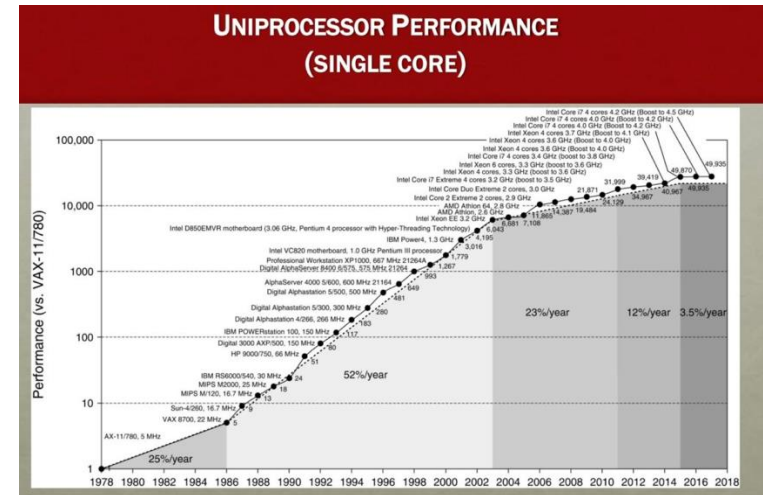
Courtesy, Intel



# End of Moore's law, Dennard Scaling and Power Wall

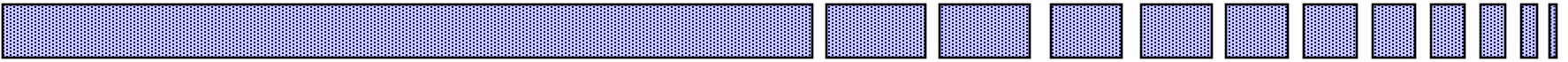


- ❑ Moore's law is claimed to end already
- ❑ Dennard scaling -- as transistors get smaller, their power density stays constant, so that the power use stays in proportion with area, ended also
- ❑ Power wall -- a barrier to clock speed—that has limited microprocessor frequency to around 4 GHz since 2005.



This issue is part science and part finance. As a result of the shrinking the size of microscopic transistors in a modern integrated circuit, these transistors are closer together. This causes two problems: heat and quantum effects. In order to achieve the desired high performance, these packed transistors generate a lot of heat. Too much heat can literally fry a chip, making it useless. Quantum effects cause their behavior to become unpredictable, not a desired trait in the way computers today.

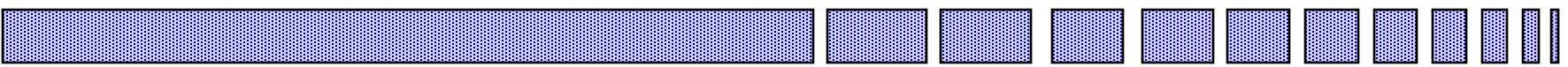
# End of Dennard Scaling is a Crisis



- Energy consumption has become more important to users
  - For mobile, IoT, and for large clouds
- Processors have reached their power limit
  - Thermal dissipation is maxed out (chips turn off to avoid overheating!)
  - Even with better packaging: heat and battery are limits.
- Architectural advances must increase energy efficiency
  - Reduce power or improve performance for same power
- The dominant architectural techniques have reached limits in energy efficiency!
  - 1982-2005: Instruction-level parallelism
  - Compiler and processor find parallelism
  - 2005-2021: Multicore
  - Programmer identifies parallelism
  - Caches: diminishing returns (small incremental improvements).



# What will happen next?



- ❑ Moore's Law—the doubling of every two years of how many transistors can fit on a chip—has ended
- ❑ Innovation will continue in new computer architectures, chip packaging, interconnects, and memory
- ❑ [5G networks](#) will move more high-performance consumer computing needs seamlessly to the cloud
- ❑ New applications and hardware other than CPU speed ([5G networks](#), displays, sensors) will now drive sales of consumer devices
- ❑ New winners and losers will emerge in consumer devices and chip suppliers

<https://medium.com/@sgblank/the-end-of-more-the-death-of-moores-law-5ddcfd8439dd>



# Not only Microprocessors

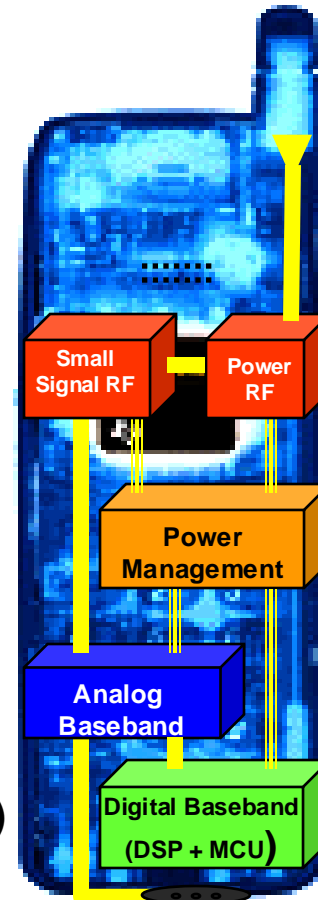
Cell  
Phone



**Digital Cellular Market  
(Phones Shipped)**

	<u>1996</u>	<u>1997</u>	<u>1998</u>	<u>1999</u>	<u>2000</u>
Units	48M	86M	162M	260M	435M

(data from Texas Instruments)



iphone 14, 2022





# Challenges in Digital Design

$\propto$  DSM

## “Microscopic Problems”

- Ultra-high speed design
- Interconnect
- Noise, Crosstalk
- Reliability, Manufacturability
- Power Dissipation
- Clock distribution.

Everything Looks a Little Different

$\propto$  1/DSM

## “Macroscopic Issues”

- Time-to-Market
- Millions of Gates
- High-Level Abstractions
- Reuse & IP: Portability
- Predictability
- etc.

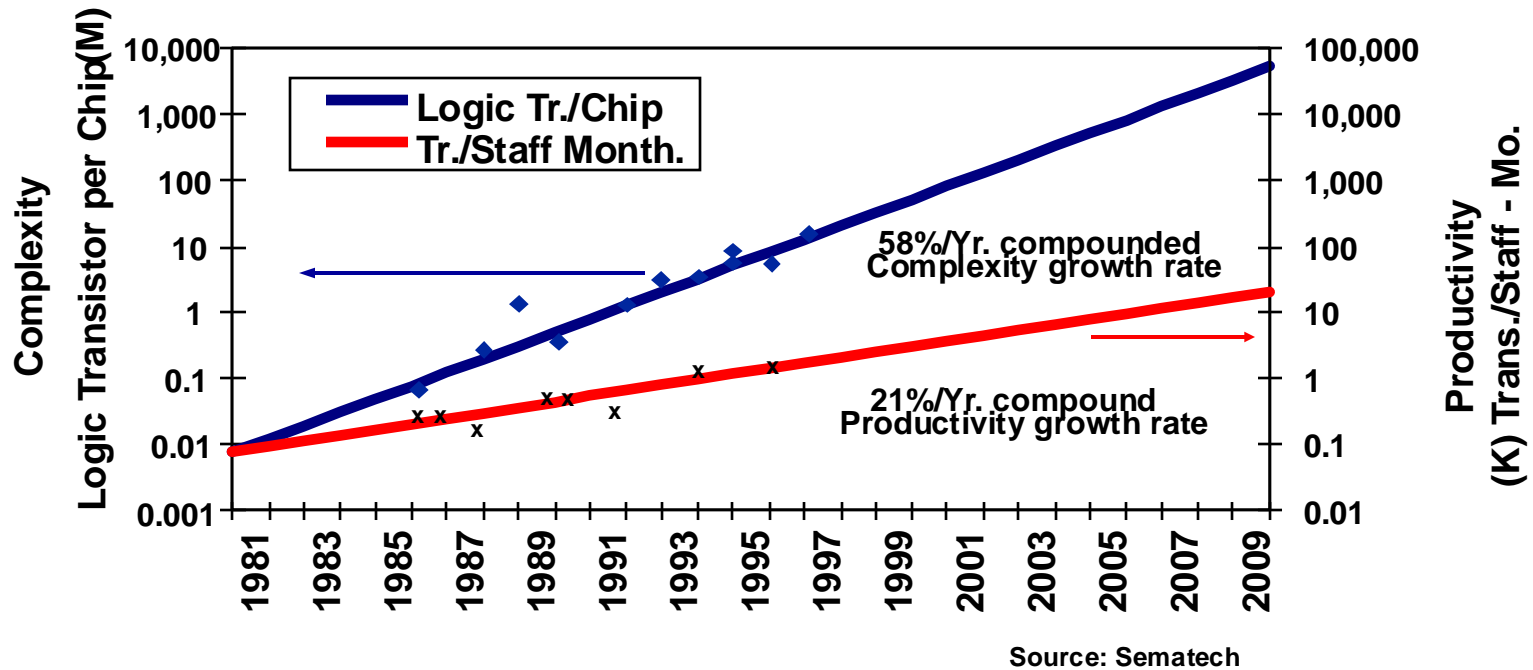
...and There's a Lot of Them!



?



# Productivity Trends

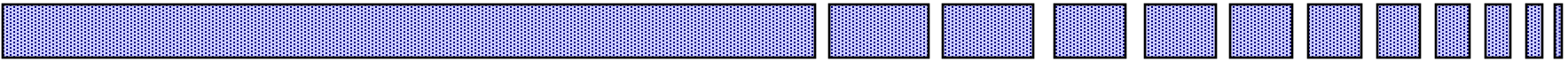


**Complexity outpaces design productivity**

Courtesy, ITRS Roadmap



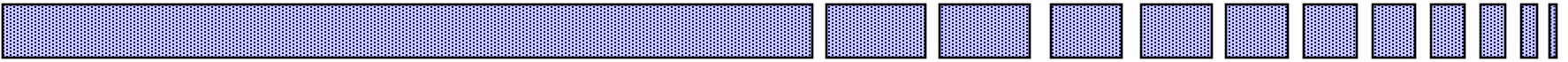
# Why Scaling?



- Technology shrinks by 0.7/generation ( $0.7 \times 0.7 = 0.49$ )
- With every generation can integrate 2x more functions per chip; chip cost does not increase significantly
- Cost of a function decreases by 2x
- But ...
  - How to design chips with more and more functions?
  - Design engineering population does not double every two years...
- Hence, a need for more efficient design methods
  - Exploit different levels of abstraction



# VLSI Design Problems



The most important characteristics

- Area
- Speed
- Power dissipation
- Design time
- Reliability
- Testability

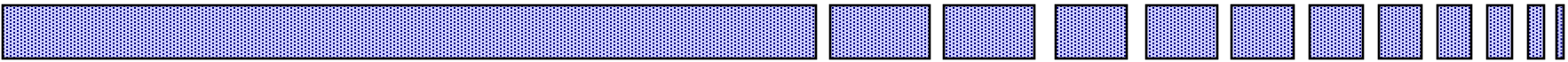
All these characteristics can be combined into a single cost function, the VLSI cost function.

It is impossible to try to design a VLSI circuit at one step while at the same time optimizing the cost function.

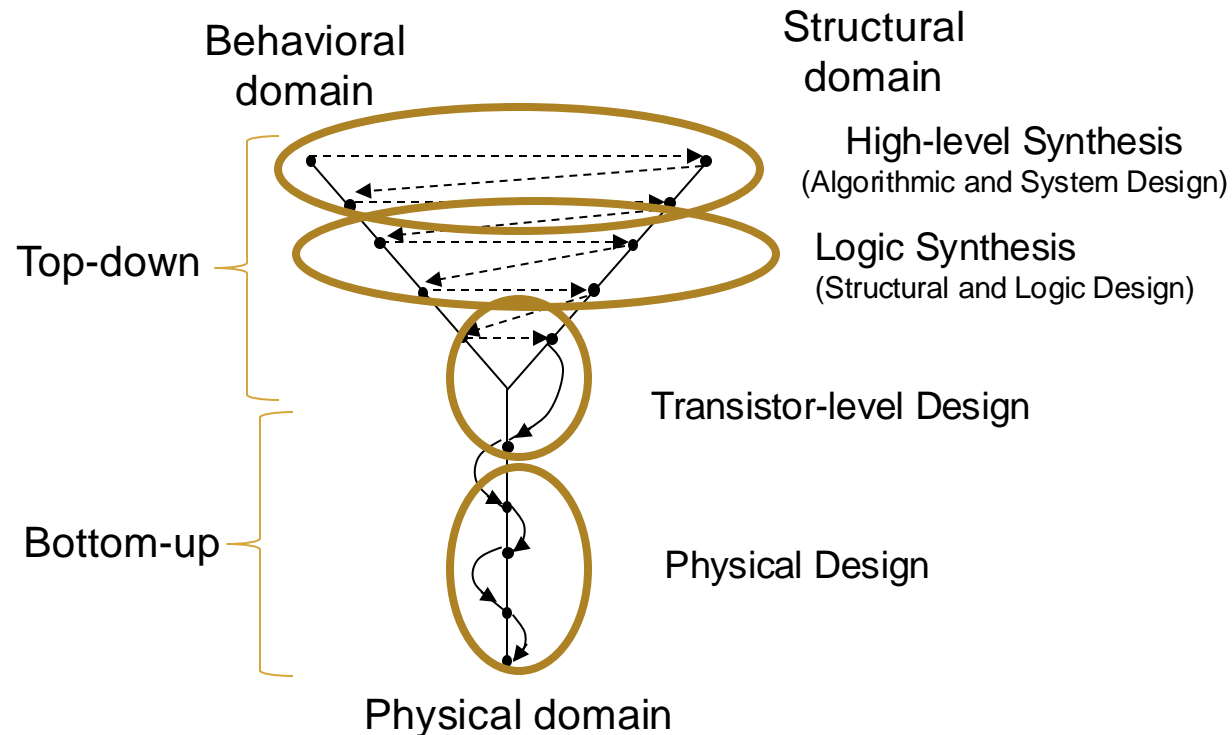
The complexity is simply too high. Two main concepts that are helpful to deal with this complexity are [hierarchy](#) and [abstraction](#).



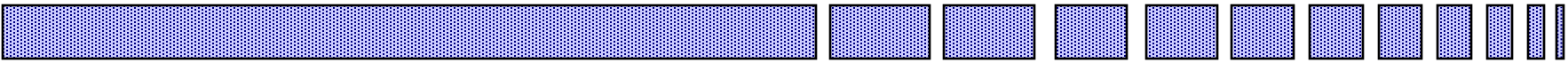
# Design Domains (1)



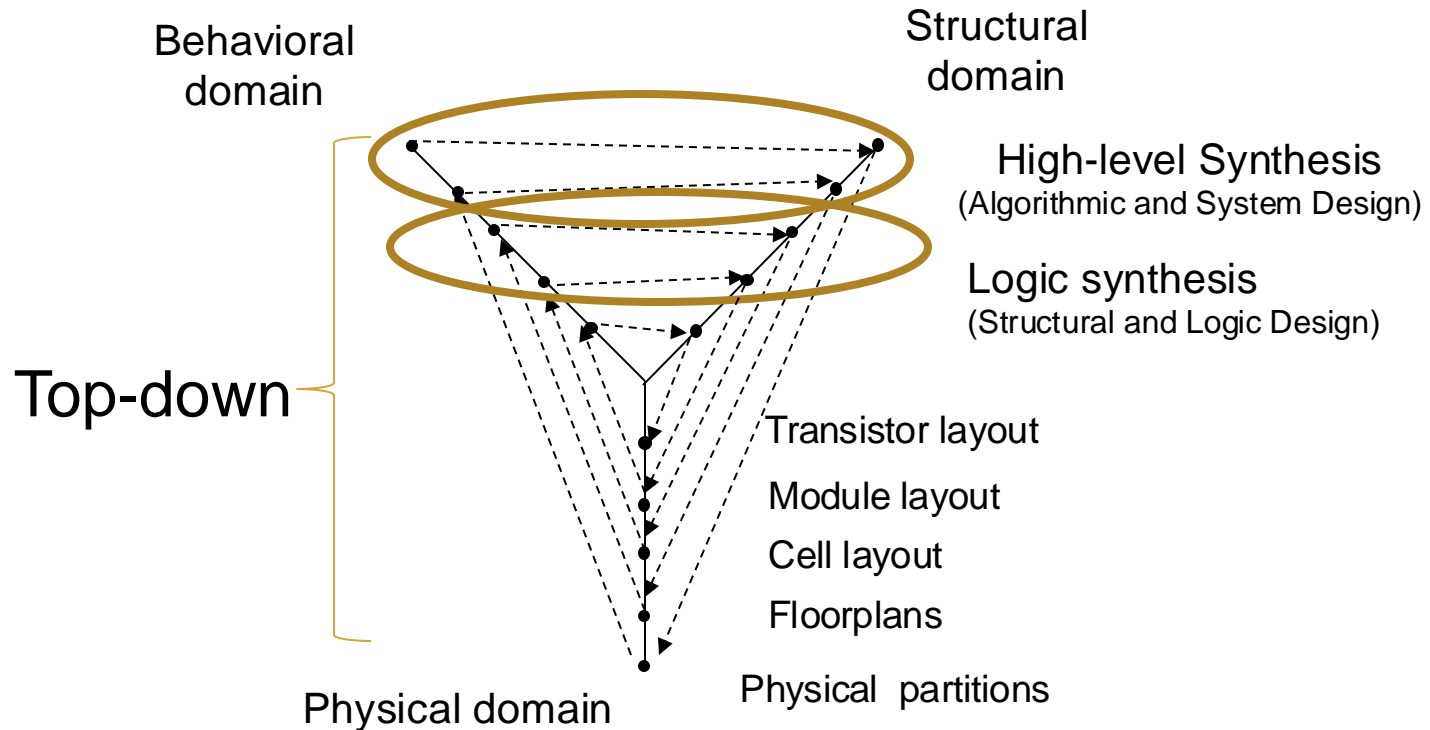
Top-down and bottom-up design methodology



# Design Domains (2)

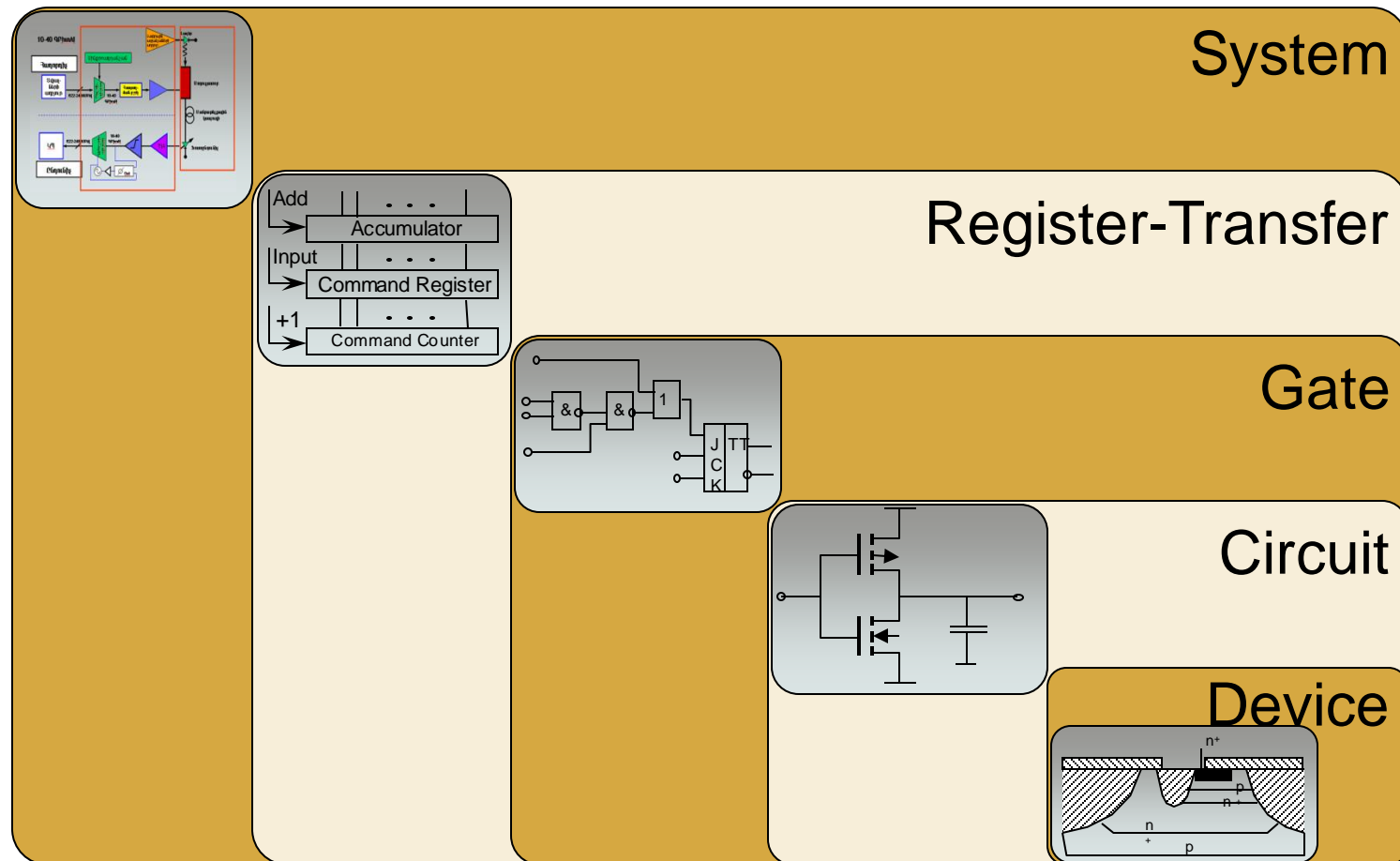
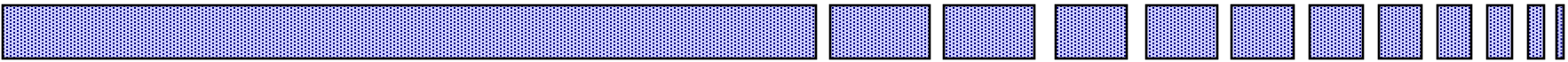


Only top-down design methodology

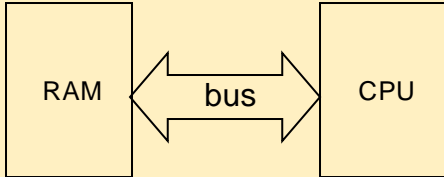
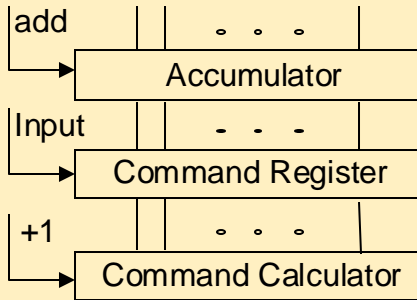
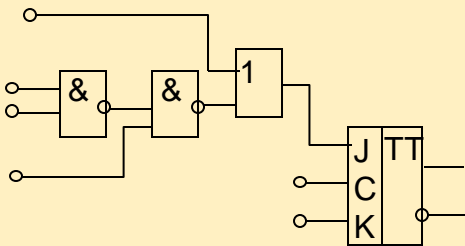




# VLSI Design Abstraction Levels (1)

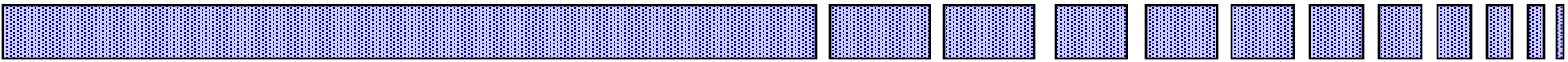


# VLSI Design Abstraction Levels (2)

N	Level	Modeling Object	Example of Modeling Object	Mathematical Apparatus
1	System	Structural Circuit		Theory of Massive Operation
2	Register-Transfer Level	Functional Circuits at the Register-level		Boolean Algebra
	Logic	Circuit at the gate and flip flop-level		



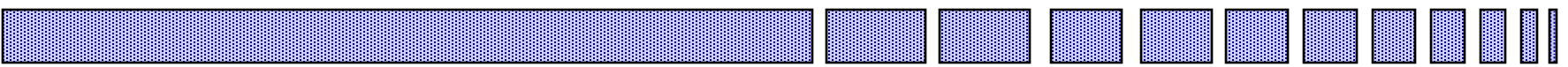
# VLSI Design Abstraction Levels (3)



N	Level	Modeling Object	Example of Modeling Object Level	Mathematical Apparatus
3	Circuit	Electrical Circuit		System of differential equations
4	Component	IC Components		System of differential equations involving partial arbitrary functions



# Intellectual property



- Intellectual property (IP): pre-designed components.
  - May come from outside vendors, internal sources.
- IP saves time, design cost.
- IP blocks must be designed to be reused.

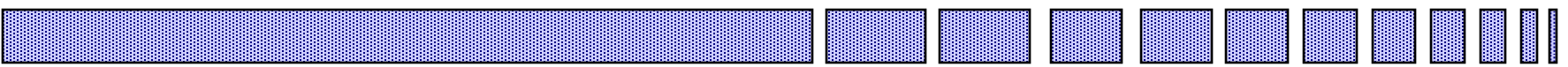


# Reliability

- Nanometer technologies require attention to reliability.
  - Chip reliability is the major challenges for 22nm and beyond (EM, SM, NBTI, hot carriers, TDDB)
- Design-for-manufacturing (DFM) and design-for-yield (DFY) techniques adjust the design to improve yield.
- Circuit and architecture techniques can compensate for unreliable components.



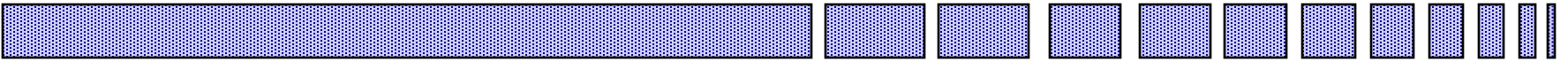
# The VLSI design process



- May be part of larger product design.
- Major levels of abstraction:
  - specification;
  - architecture;
  - logic design;
  - circuit design;
  - layout.



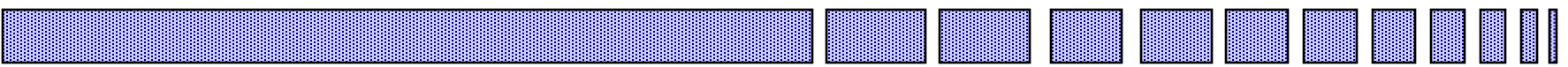
# Challenges in VLSI design



- Multiple levels of abstraction: transistors to CPUs.
- Multiple and conflicting constraints: low cost and high performance are often at odds.
- Short design time: Late products are often irrelevant.



# Dealing with complexity

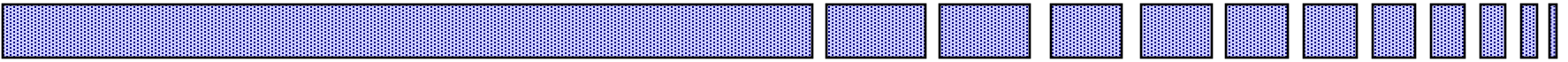


- Divide-and-conquer: limit the number of components you deal with at any one time.
- Group several components into larger components:
  - transistors form gates;
  - gates form functional units;
  - functional units form processing elements;
  - etc.

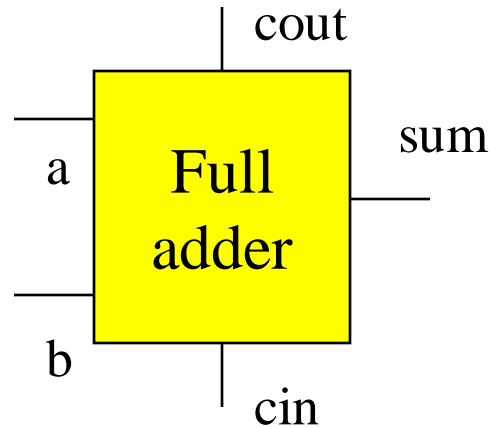




# Hierarchical name

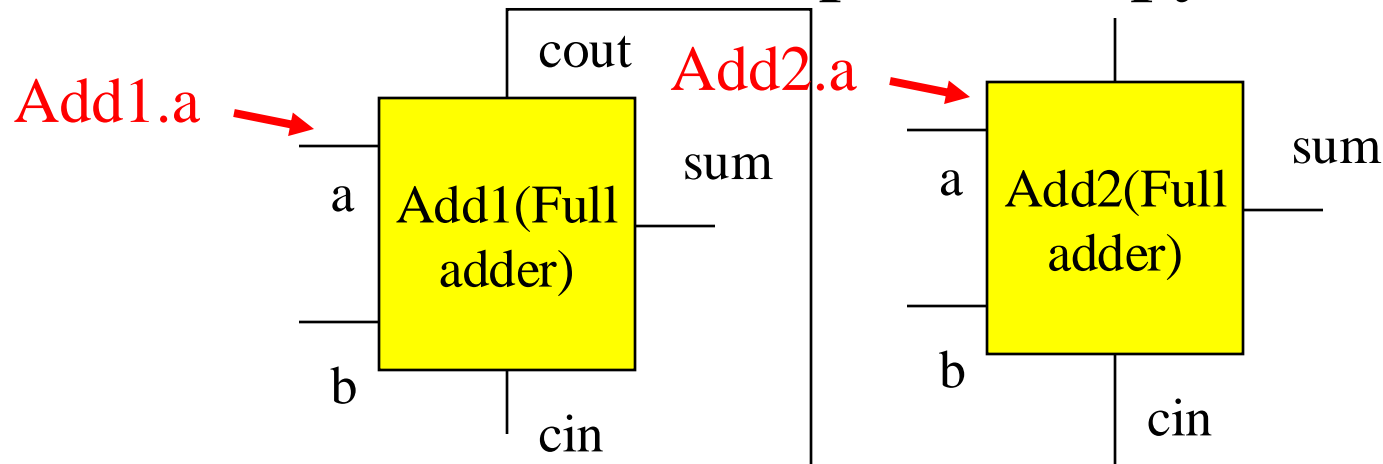


- Interior view of a component:
  - components and wires that make it up.
- Exterior view of a component = type:
  - body;
  - pins.

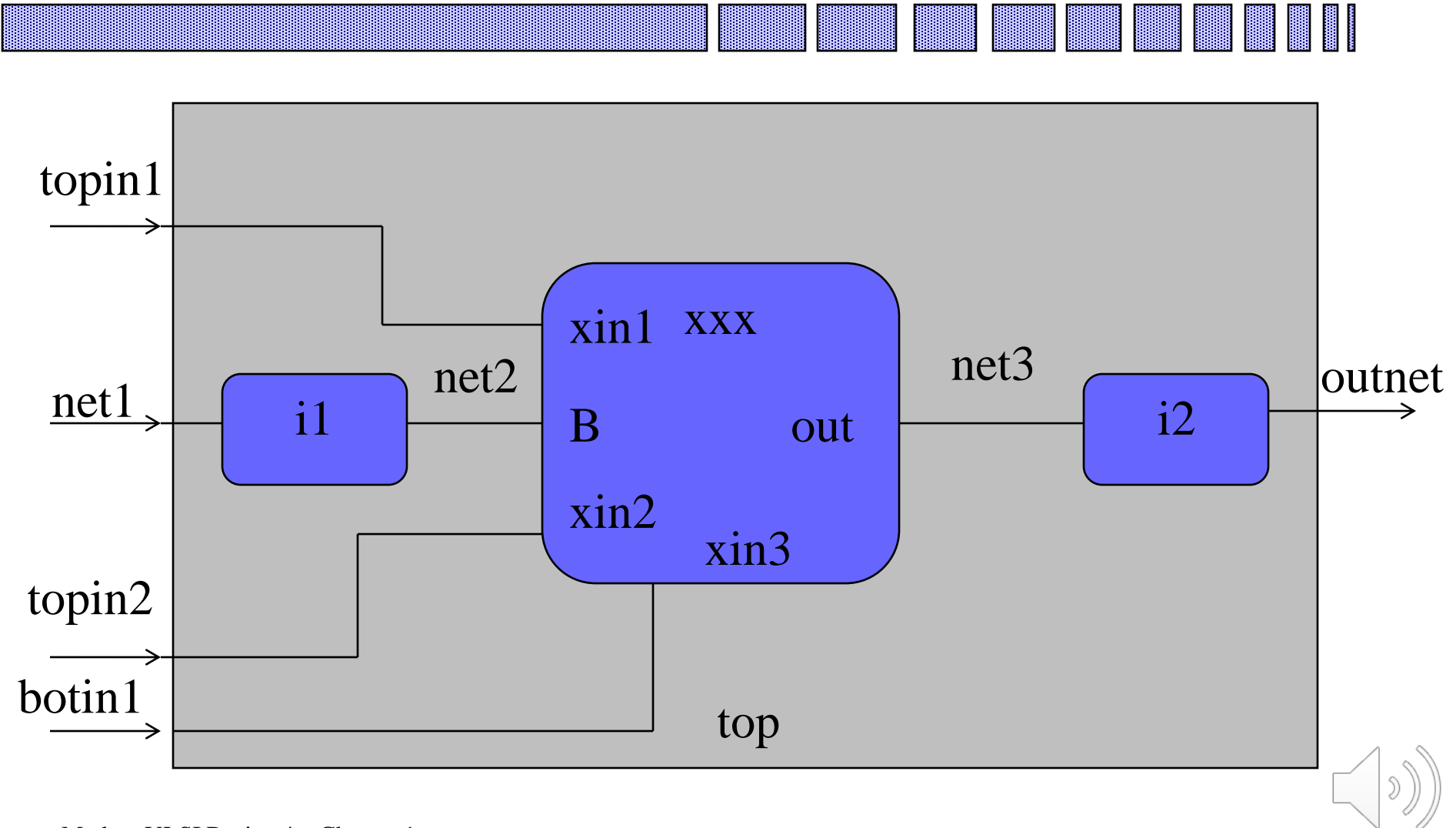


# Instantiating component types

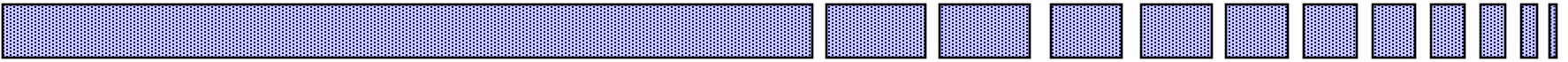
- Each instance has its own name:
  - add1 (type full adder)
  - add2 (type full adder).
- Each instance is a separate copy of the type:



# A hierarchical logic design



# Net lists and component lists



## □ Net list:

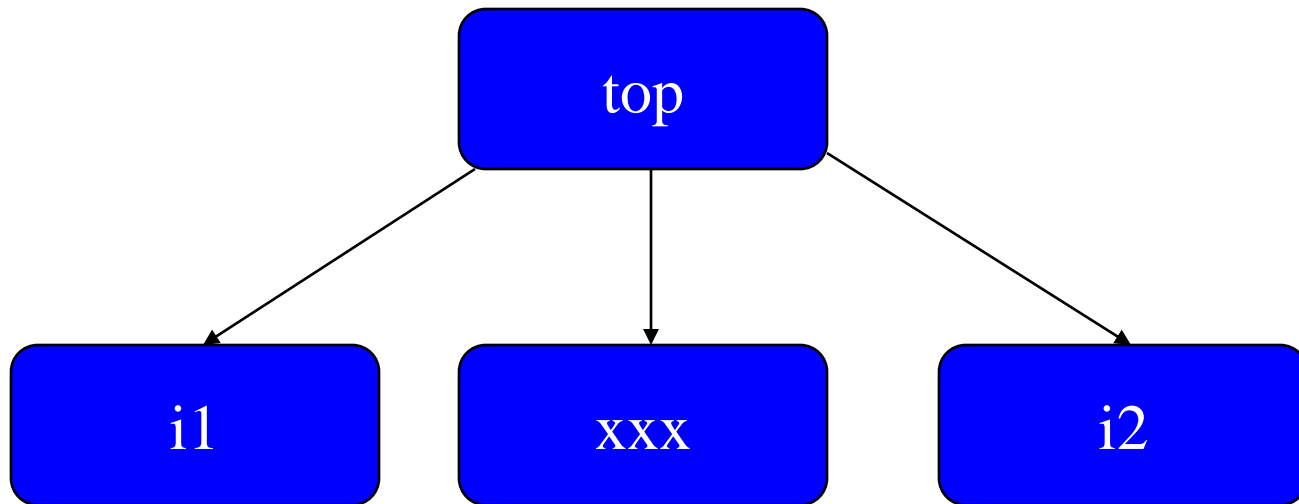
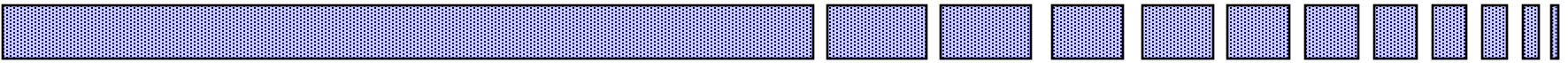
```
net1: top.in1 i1.in
net2: i1.out xxx.B
topin1: top.n1 xxx.xin1
topin2: top.n2 xxx.xin2
botin1: top.n3 xxx.xin3
net3: xxx.out i2.in
outnet: i2.out top.out
```

## □ Component list:

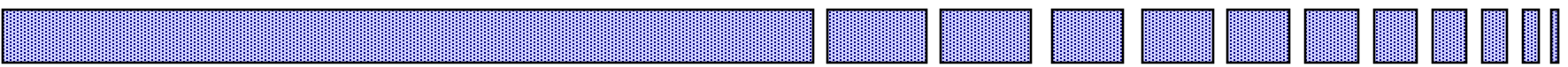
```
top: in1=net1 n1=topin1
    n2=topin2 n3=botin1
    out=outnet
i1: in=net1 out=net2
xxx: xin1=topin1
    xin2=topin2
    xin3=botin1 B=net2
    out=net3
i2: in=net3 out=outnet
```



# Component hierarchy

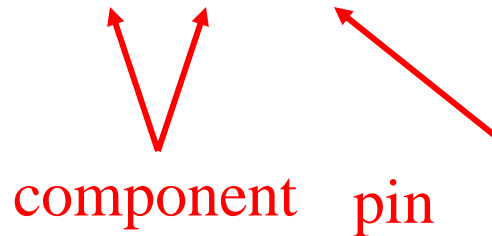


# Hierarchical names

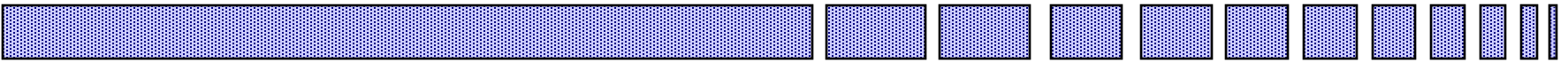


□ Typical hierarchical name:

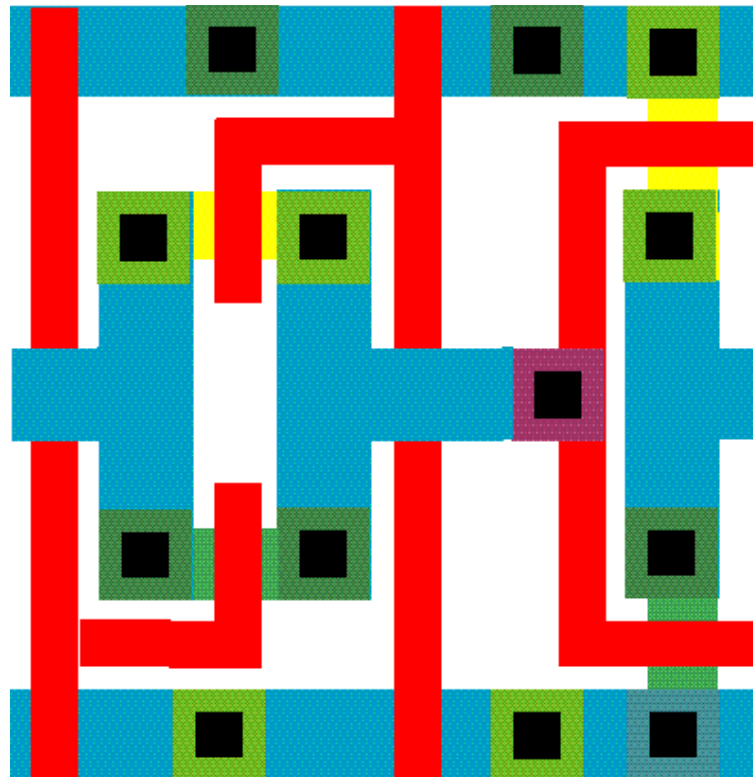
— top/i1.foo



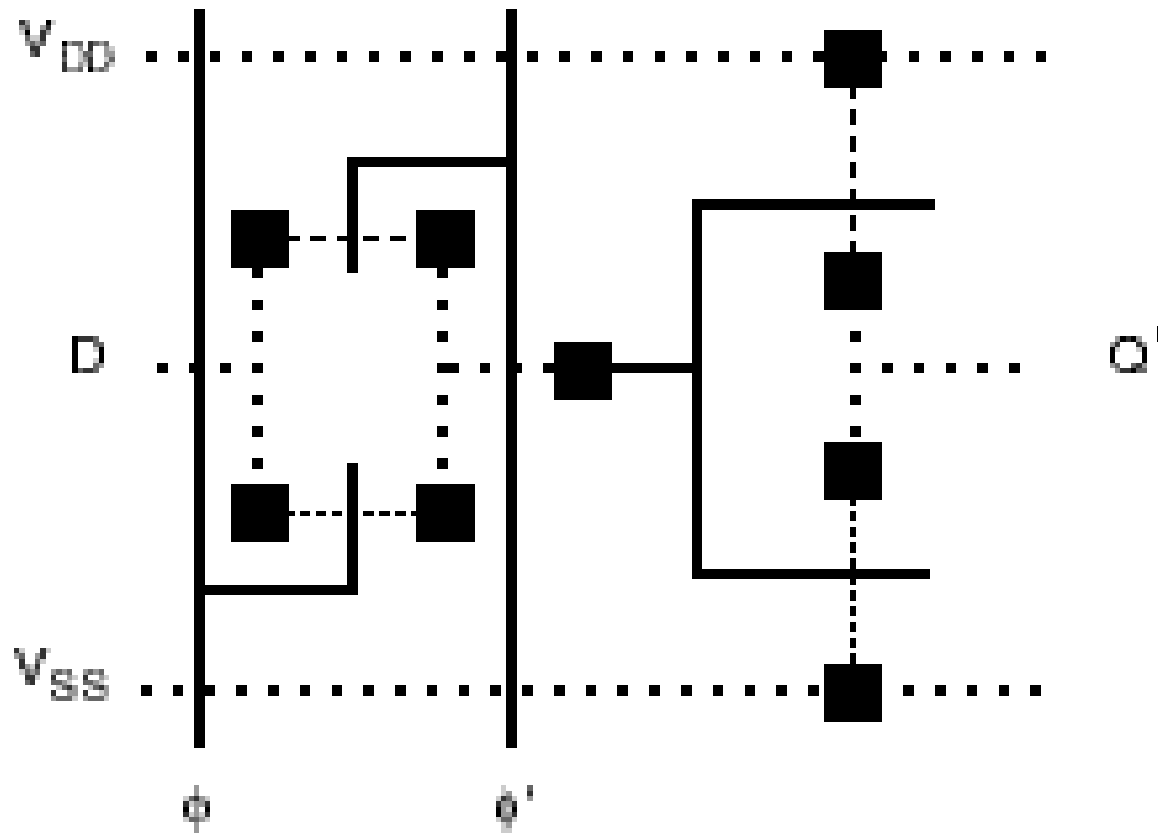
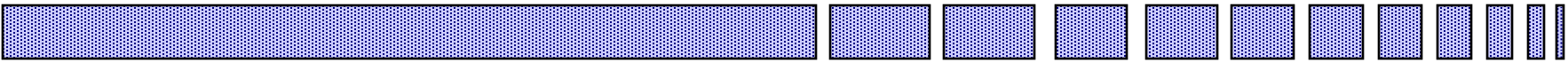
# Layout and its abstractions



- Layout for dynamic latch:

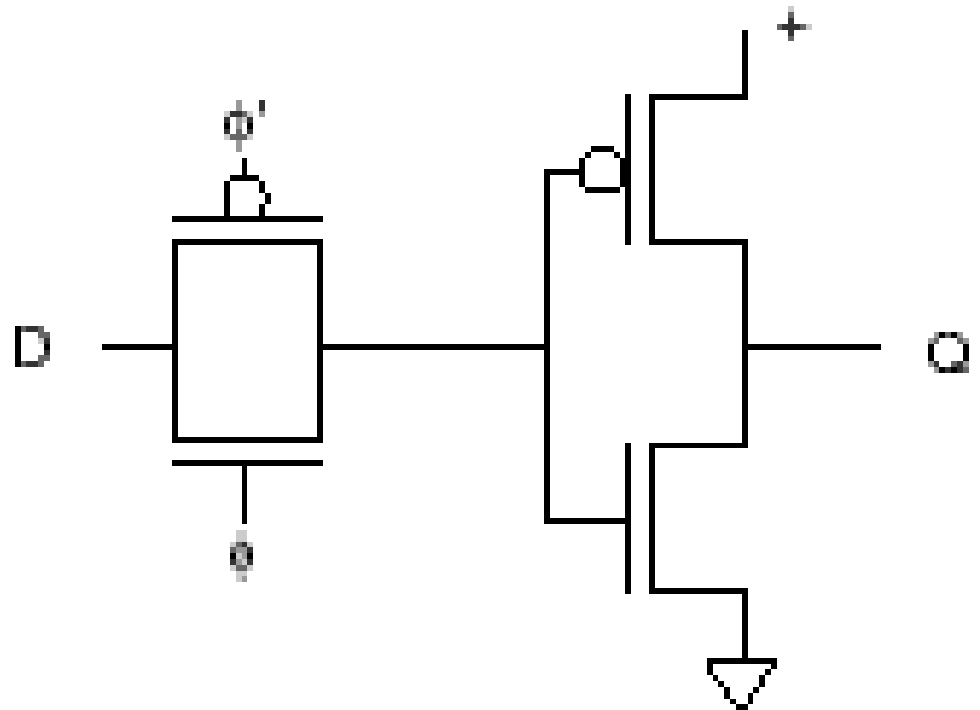


# Stick diagram

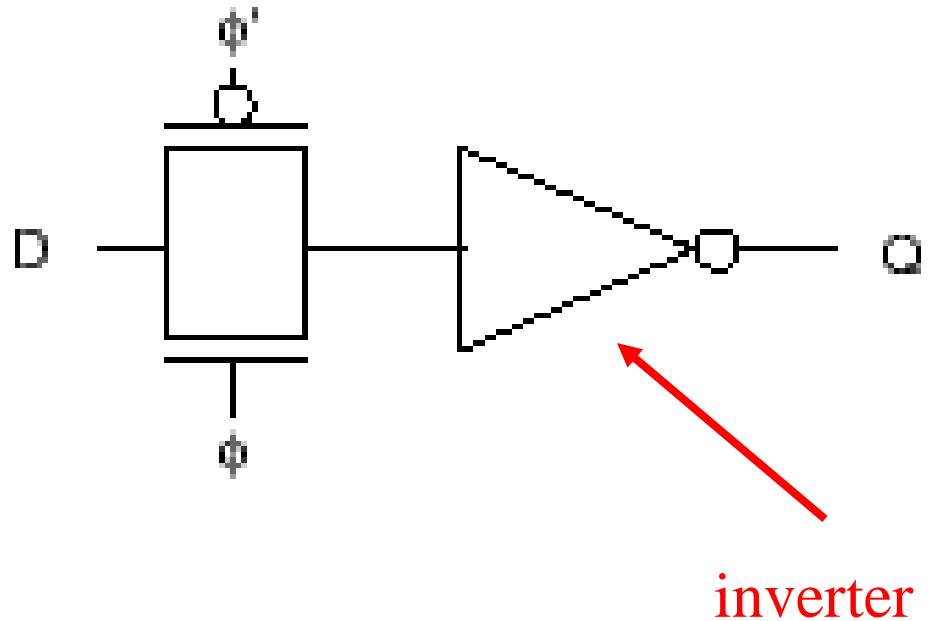
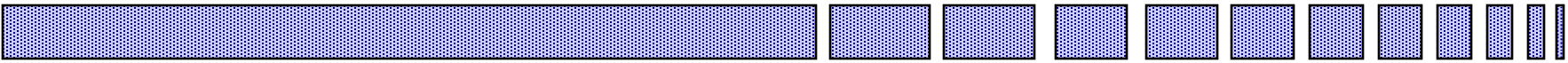




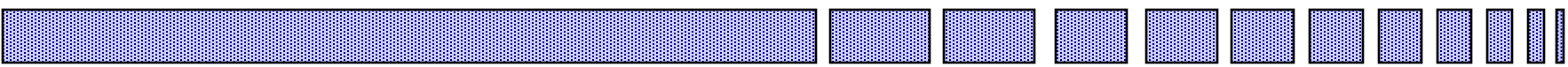
# Transistor schematic



# Mixed schematic



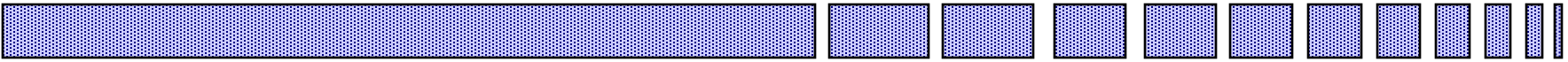
# Levels of abstraction



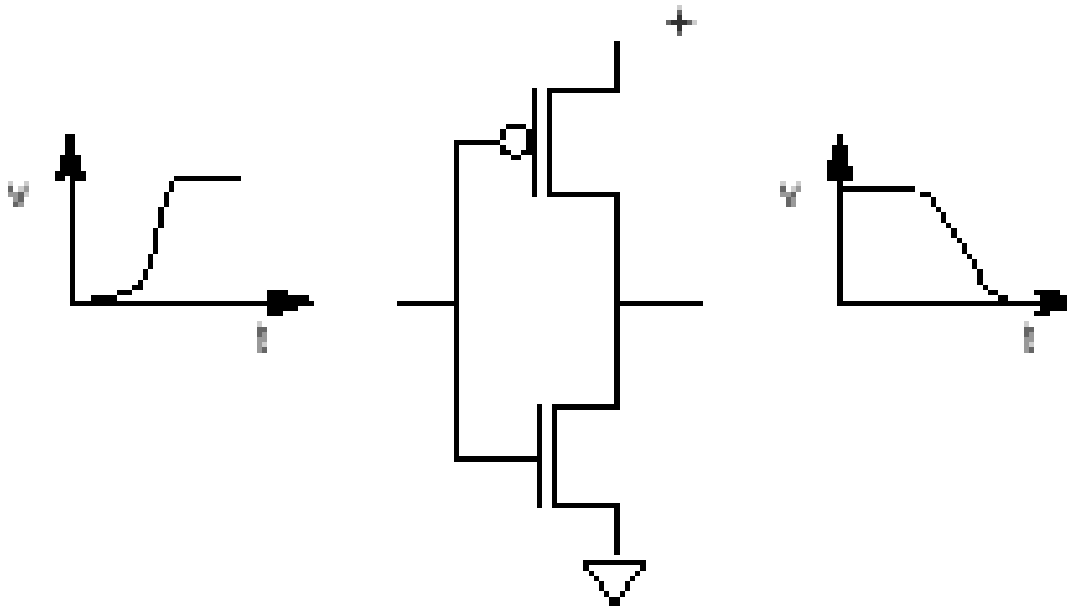
- Specification: function, cost, etc.
- Architecture: large blocks.
- Logic: gates + registers.
- Circuits: transistor sizes for speed, power.
- Layout: determines parasitics.



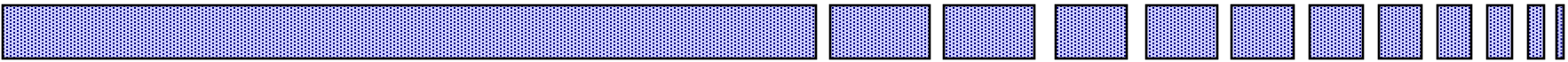
# Circuit abstraction



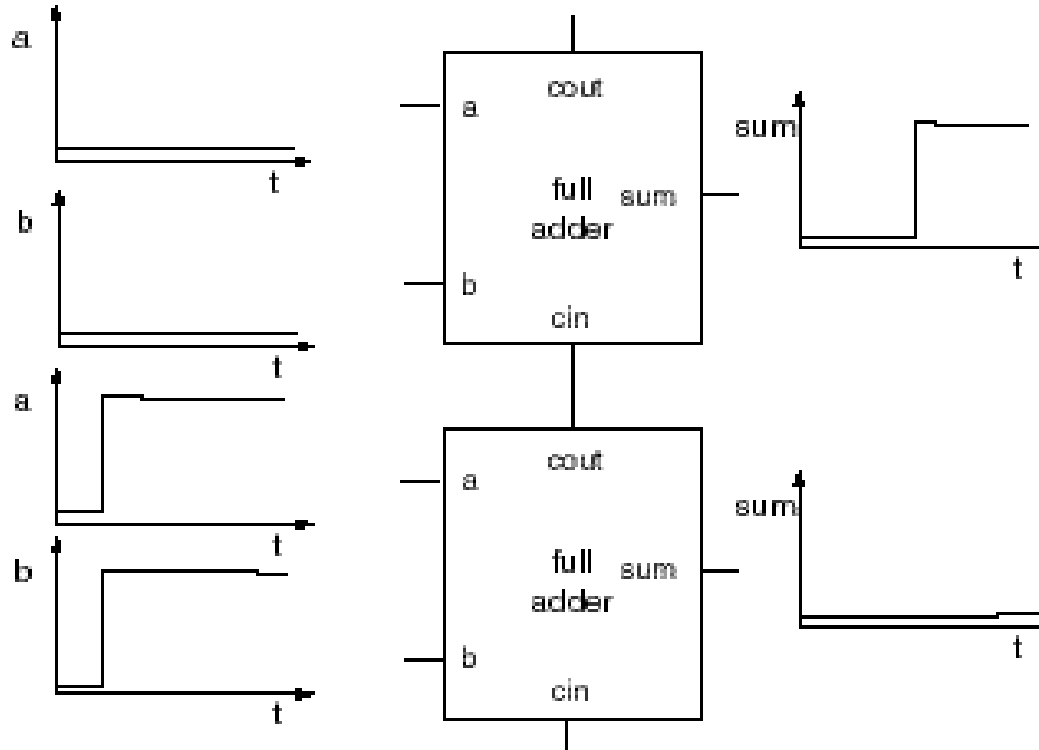
- Continuous voltages and time:



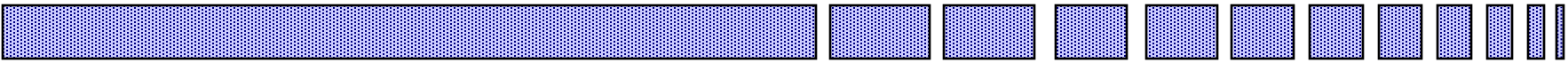
# Digital abstraction



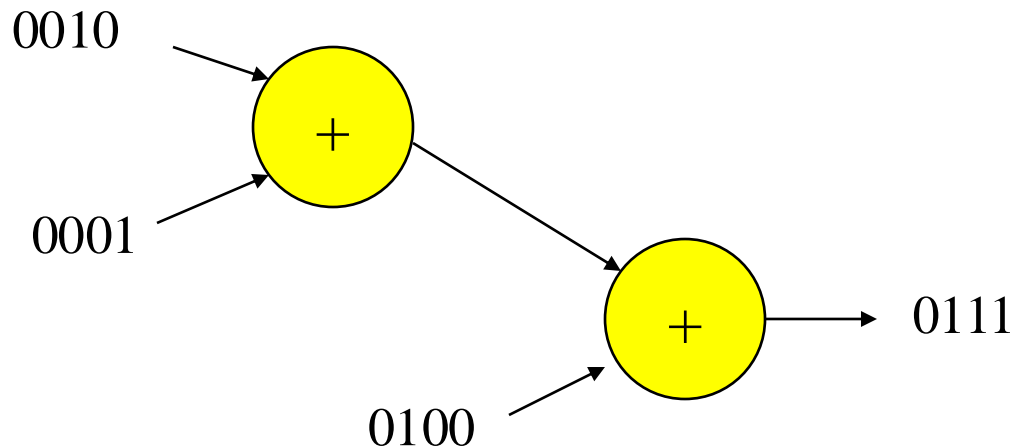
□ Discrete levels, discrete time:



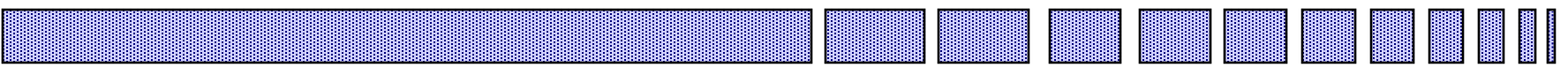
# Register-transfer abstraction



- Abstract components, abstract data types:



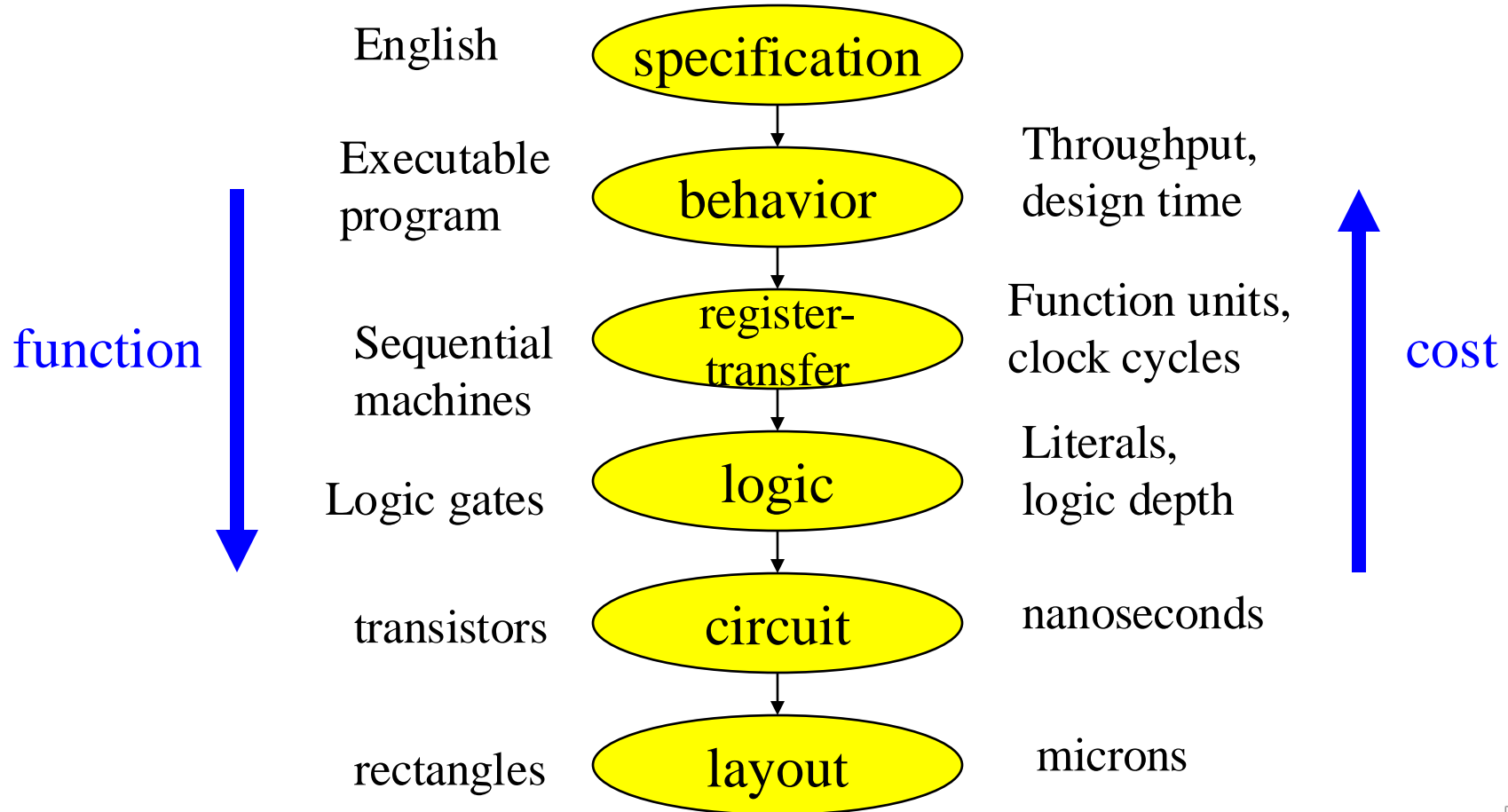
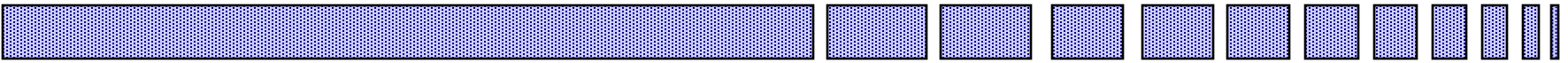
# Top-down vs. bottom-up design



- Top-down design adds functional detail.
  - Create lower levels of abstraction from upper levels.
- Bottom-up design creates abstractions from low-level behavior.
- Good design needs both top-down and bottom-up efforts.

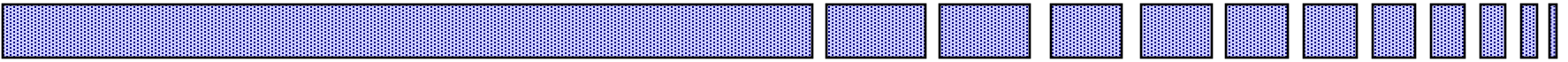


# Design abstractions





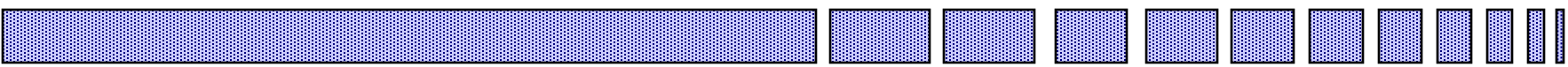
# Design validation



- Must check at every step that errors haven't been introduced-the longer an error remains, the more expensive it becomes to remove it.
- Forward checking: compare results of less- and more-abstract stages.
- Back annotation: copy performance numbers to earlier stages.



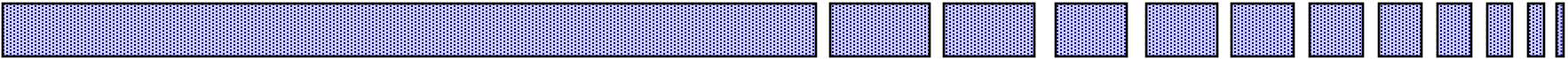
# Manufacturing test



- Not the same as design validation: just because the design is right doesn't mean that every chip coming off the line will be right.
- Must quickly check whether manufacturing defects destroy function of chip.
- Must also speed-grade.

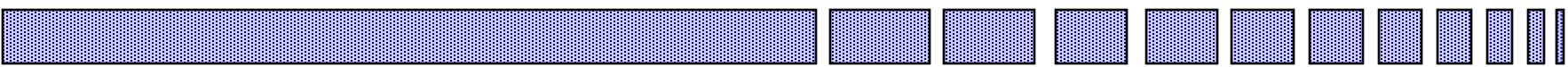


# IP-based design

- 
- Almost every chip uses some form of IP:
    - Standard cell libraries.
    - Memories.
    - IP blocks.
  - Designers must know how to:
    - Create IP.
    - Use IP.



# Types of IP



## □ Hard IP:

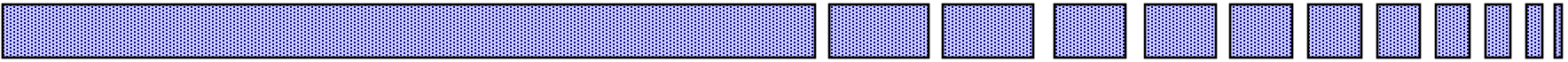
- Pre-designed layout.
- Allows more detailed characterization.

## □ Soft IP:

- No layout---logic synthesis, etc.
- IP layout is created by the IP user.



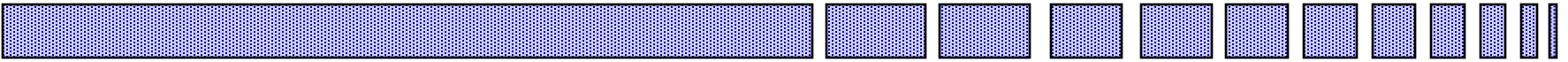
# Hard IP



- Must conform to many standards:
  - Layout pin placement.
  - Layer usage.
  - Transistor sizing.
- Hard IP blocks are usually qualified on a particular process.
  - Component is fabricated and tested to show that the IP works on that fab line.



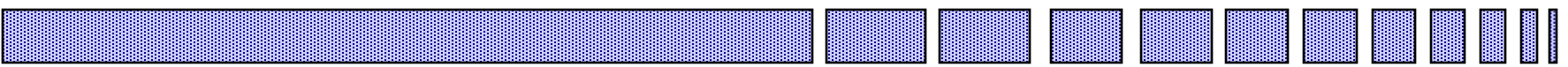
# Soft IP



- Conformance of layout to local standards is easier since it is created by the user.
- Timing can only be estimated until the layout is done.
- Must conform to interface standards.
  - A wrapper adapts a block to a new interface.



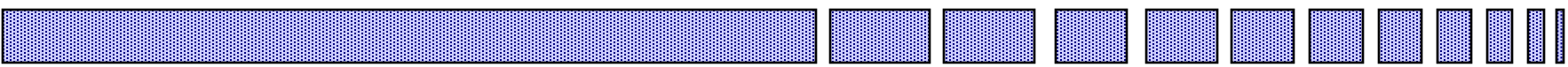
# IP across the design hierarchy



- Standard cells.
  - Pitch matched in rows, compatible drive.
- Register-transfer modules.
- Memories.
- CPUs.
- Busses.
- I/O devices.



# Specifying IP

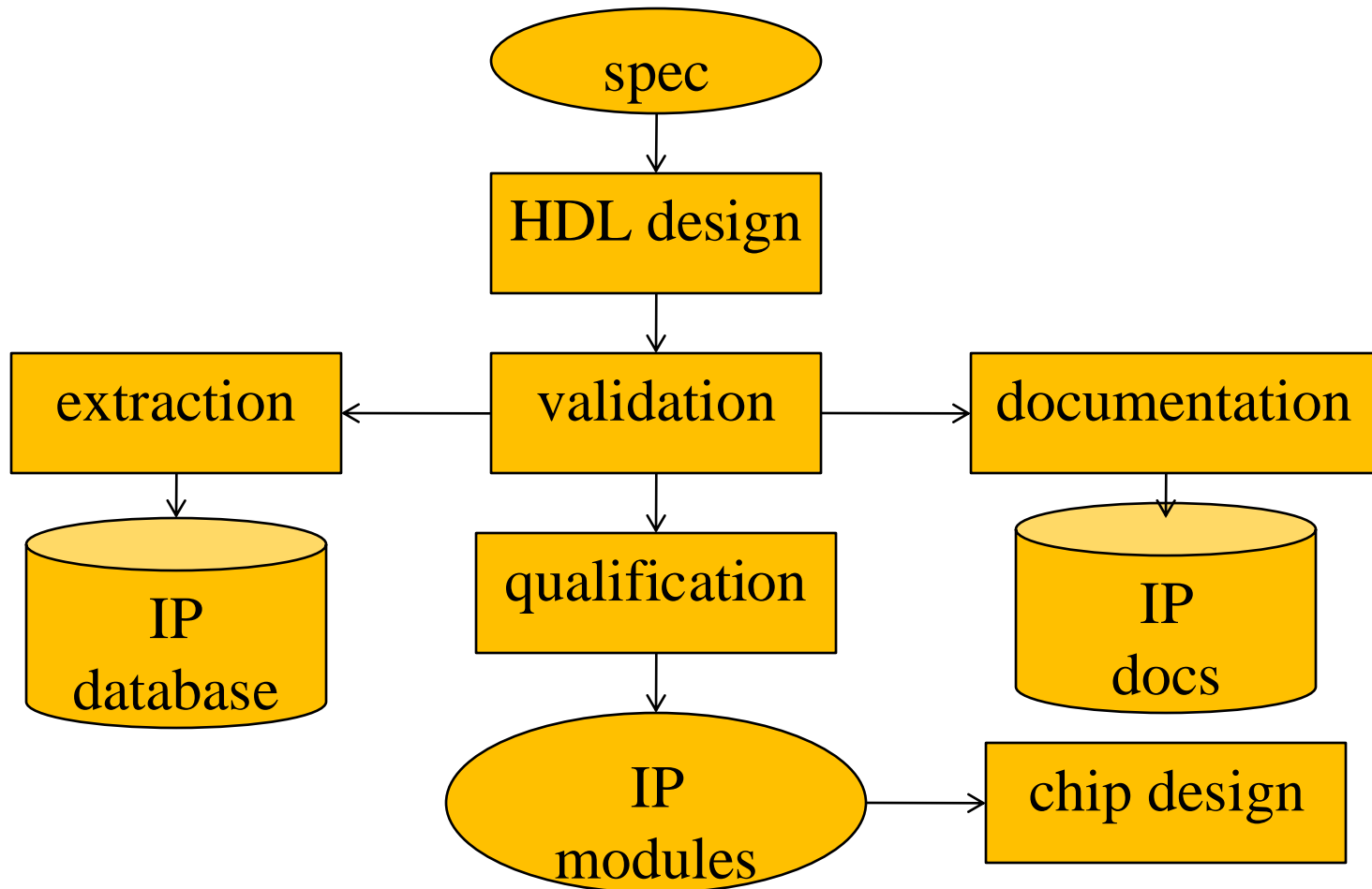


- Hard or soft?
- Functionality.
- Performance, including process corners.
- Power consumption.
- Special process features required.

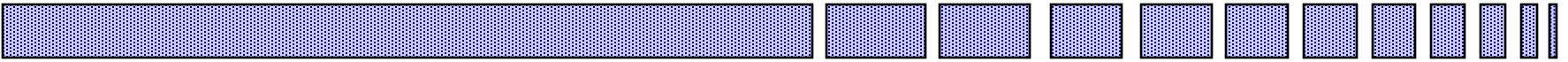




# The IP lifecycle



# Using IP



- May come from vendor, open source, or internal group.
- Must identify candidate IP, evaluate for suitability.
- May have to pay for IP.
- May want to qualify IP before use, particularly if it pushes analog characteristics.

