

VLSI Design

Lecture 1: Digital Systems and VLSI

Shaahin Hessabi

Department of Computer Engineering

Sharif University of Technology

Adapted with modifications from lecture notes
prepared by author (from Prentice Hall PTR)

Overview

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

Features of *Better* Circuit

1. Lower cost (chip area, number of ICs, ...)
2. Better performance (speed)
3. Lower power
4. Better reliability
 - More integration → less intra-chip connections
→ better reliability
 - ❖ Better testability
5. Better repeatability
6. Less design and fabrication time

Components of an Electronic System

- Chip (usually a small part of the total cost, but can influence the cost of other parts)
- Power supply
- Fan
- PCB (Printed Circuit Board)
- Bus
- Box

Why VLSI?

- Integration improves the design:
 - lower parasitics = higher speed;
 - Shorter length of signal transfer is another reason for higher speed (3 cm wire → $3 \times 10^{-2} / 3 \times 10^8 = 0.1 \text{ nsec}$)
 - lower power (hence better reliability);
 - Power is a limiting factor for high integration.
 - physically smaller.
- Integration reduces manufacturing cost - (almost) no manual assembly.
 - Greatly reduces cost of parts other than chip (supply, fan, PCB, ...)
 - ASIC might be more expensive than standard IC, but system's cost will be lower.

Levels of Integration

- SSI → MSI → LSI → VLSI
- Criteria:
 - Gate count (2-20, 20-200, 200-2000, 2000 +); you may see different numbers in literature
 - Pin count
 - Feature size (line widths, line spacing, size)
 - Chip size
 - Function (gate & FF, module, subsystem, system)

Levels of Integration (cont'd)

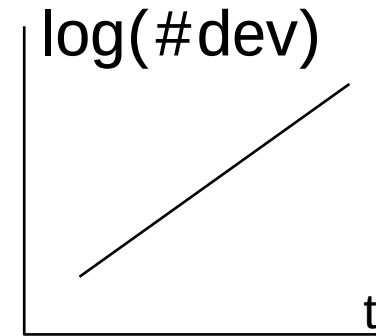
- Where to go after VLSI?
 - **ULSI** (Ultra Large Scale Integration - which is between 500,000 and 10,000,000 transistors),
 - **GSI** (Gigantic Scale Integration - which is over 10,000,000 transistors).

VLSI and you

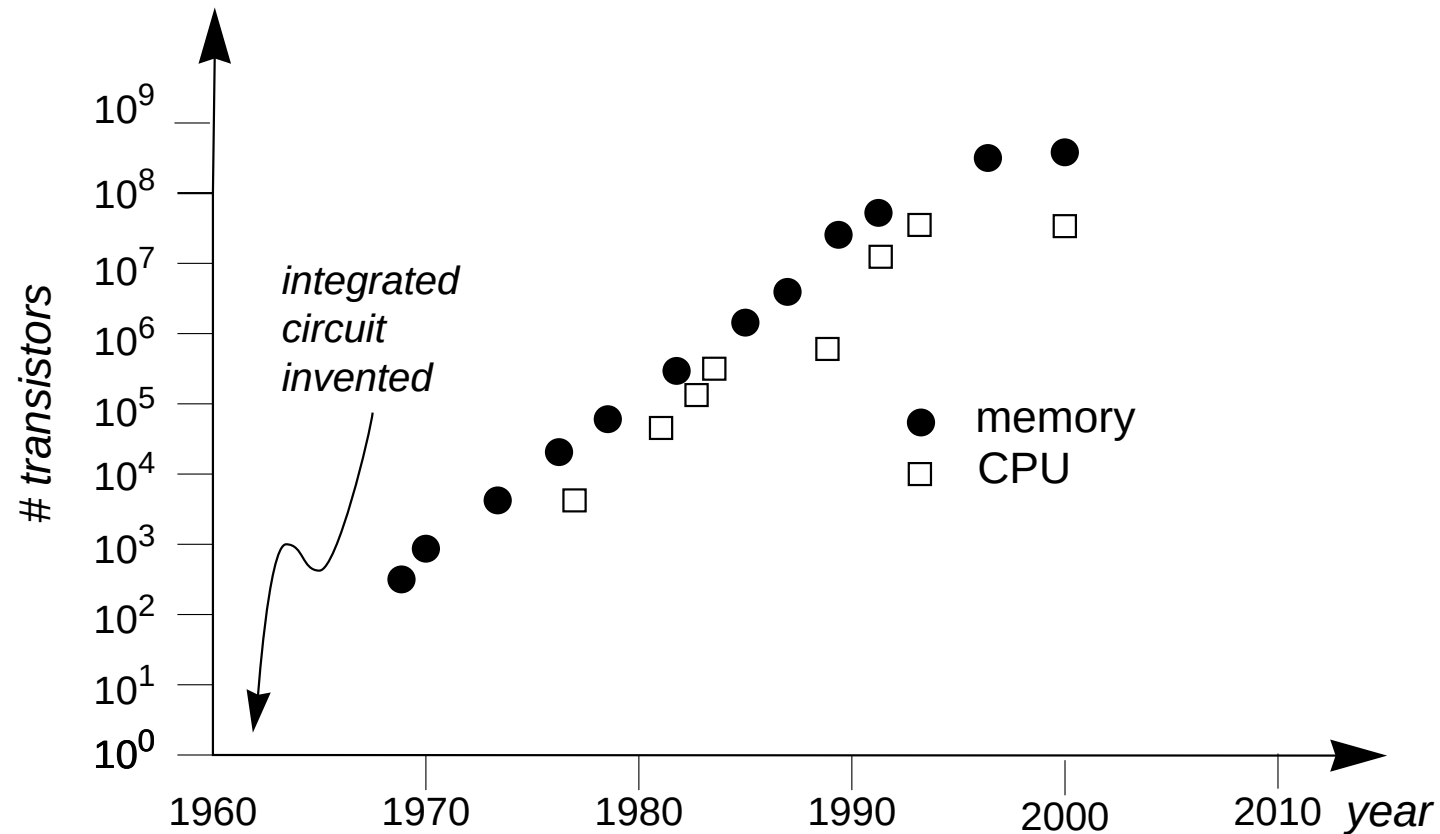
- Microprocessors:
 - personal computers;
 - microcontrollers.
- DRAM/SRAM.
- Special-purpose processors.

Moore's Law

- Gordon Moore (co-founder of Intel) predicted that number of transistors per chip would grow exponentially (doubles every 18 months).
- Exponential improvement in technology is a natural trend: steam engines, automobiles.
- Obstacles for Moore's law:
 1. Quantity and variety of products which use ICs has had less progress.
 2. Cost of design verification and test is large.
 3. Complexity of design makes it difficult to manage it among design and engineering groups.
 - Role of CAD tools.



Moore's Law plot



The cost of fabrication

- Current cost: about \$2-3 billion.
- Typical fab line occupies about 1 city block, employs a few hundred people.
- Most profitable period is the first 18 months - 2 years.

Cost factors in ICs

- For large-volume ICs:
 - packaging is the largest cost;
 - testing is second-largest cost.
- For low-volume ICs, design costs may swamp all manufacturing costs.
- IC manufacturing technology is remarkably versatile (change masks).
- Wafer size: 8 inch (12 inch)
- Chip size: $1.5 \times 1.5 \text{ cm}^2$

The VLSI design process

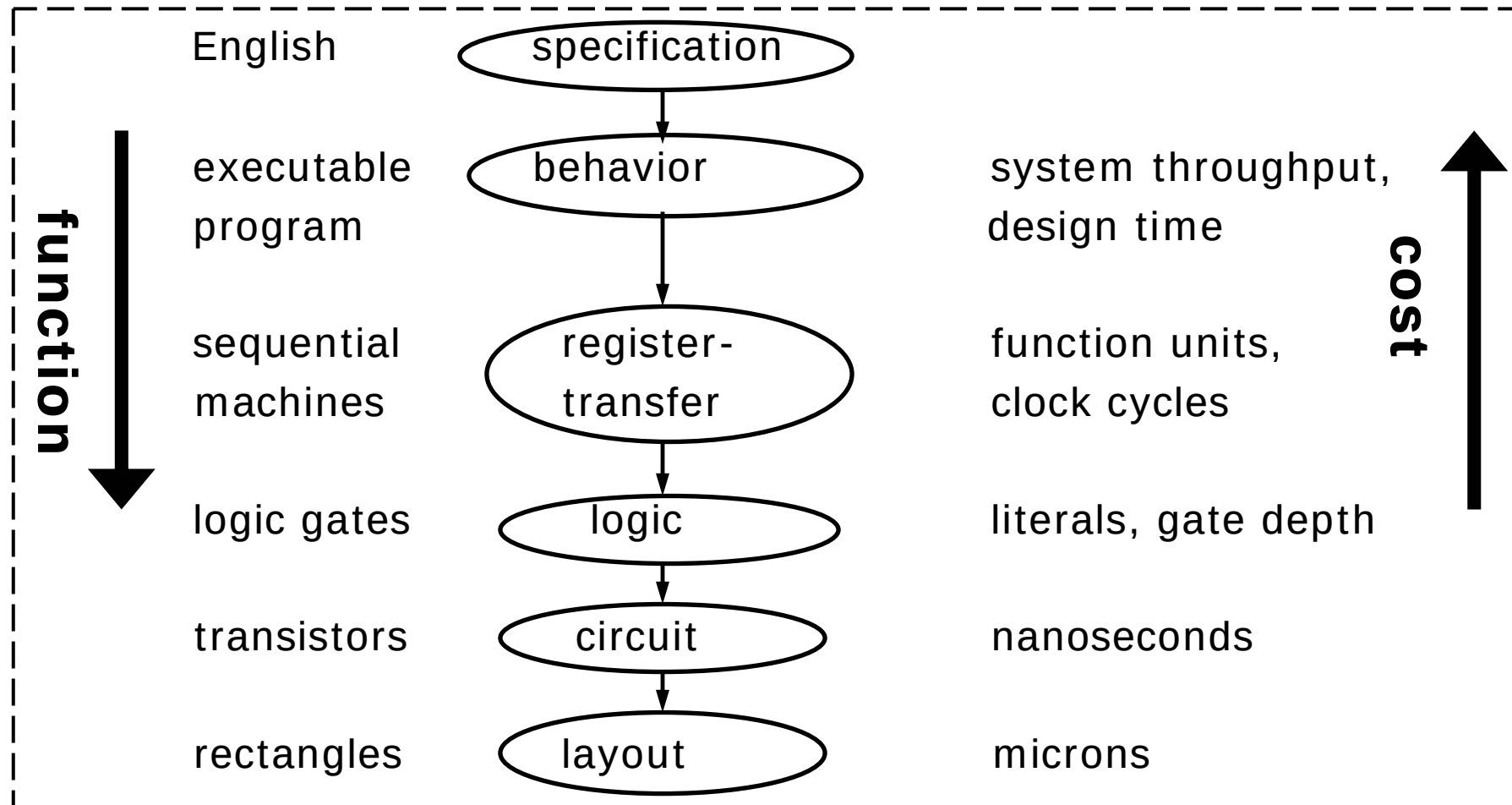
- May be part of larger product design.
- Major steps:
 - specification;
 - Algorithm design and transformation;
 - architecture;
 - logic design;
 - circuit design;
 - layout (physical design).

The steps

- Specification: function, cost, etc.
- Architecture: large blocks.
- Logic: gates + registers.
- Circuits: transistor sizes for speed, power.
- Layout:
 - Layout size determines fabrication cost.
 - Shapes determine parasitics; hence the circuit speed and power.

Challenges in VLSI design

1. Multiple levels of abstraction



Challenges in VLSI design(cont'd)

2. Multiple and conflicting costs

(speed, area, cost, power, ...)

3. Short design time

(6 months delay $\Rightarrow$ losing ~33% of the profit)

➤ Techniques to eliminate unnecessary detail:

1. Hierarchical design (divide and conquer, i.e.; breaking the chip into a hierarchy of components, where each consists of a body and a number of pins)

2. Design abstraction (use multiple levels of abstraction, as shown in previous slide)

3. Using CAD tools: tries to solve all the 3 mentioned problems;

1. dealing with multiple levels of abstraction is easier when you are not absorbed in the details,

2. computer programs can analyze cost trade-offs much better (because they are methodical)

3. computers are much faster than humans.

CAD Tools Categories

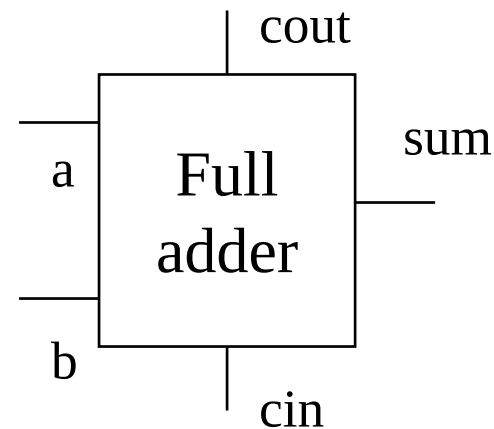
1. Design entry tools (e.g., schematic capture)
 - capture a design in machine-readable form for use by other programs, but don't do any real design work.
 2. Analysis and verification tools (e.g., spice)
 - ease the analysis task, but don't tell how to change the circuit for the desired function/spec.
 3. Synthesis tools (e.g., Leonardo)
 - create a design at a lower level of abstraction from a higher level description.
- Both hierarchical design and design abstraction are as important to CAD tools as they are to humans.

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.

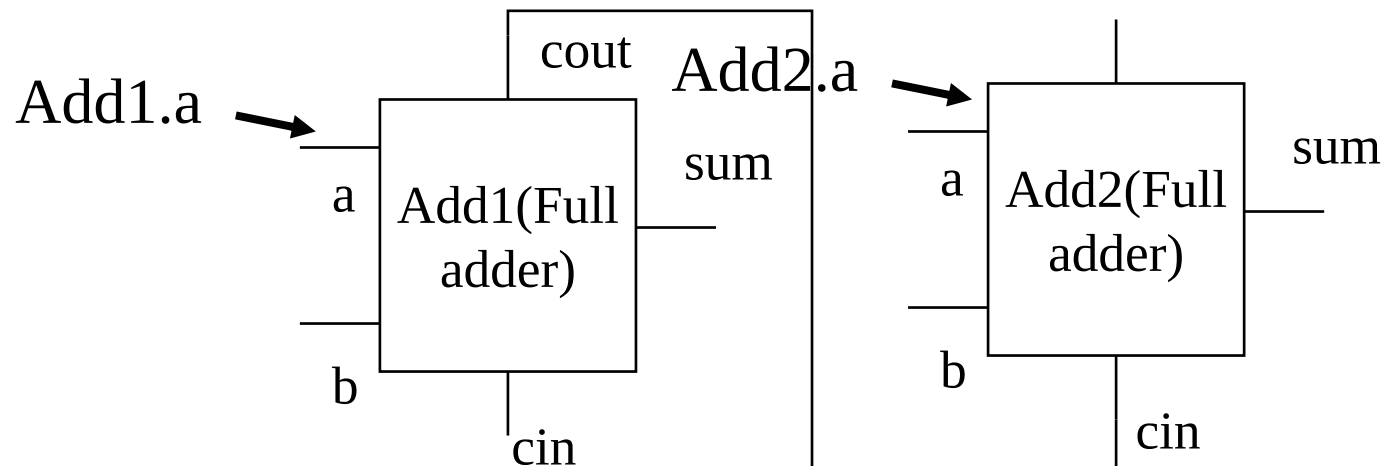
Hierarchical name

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

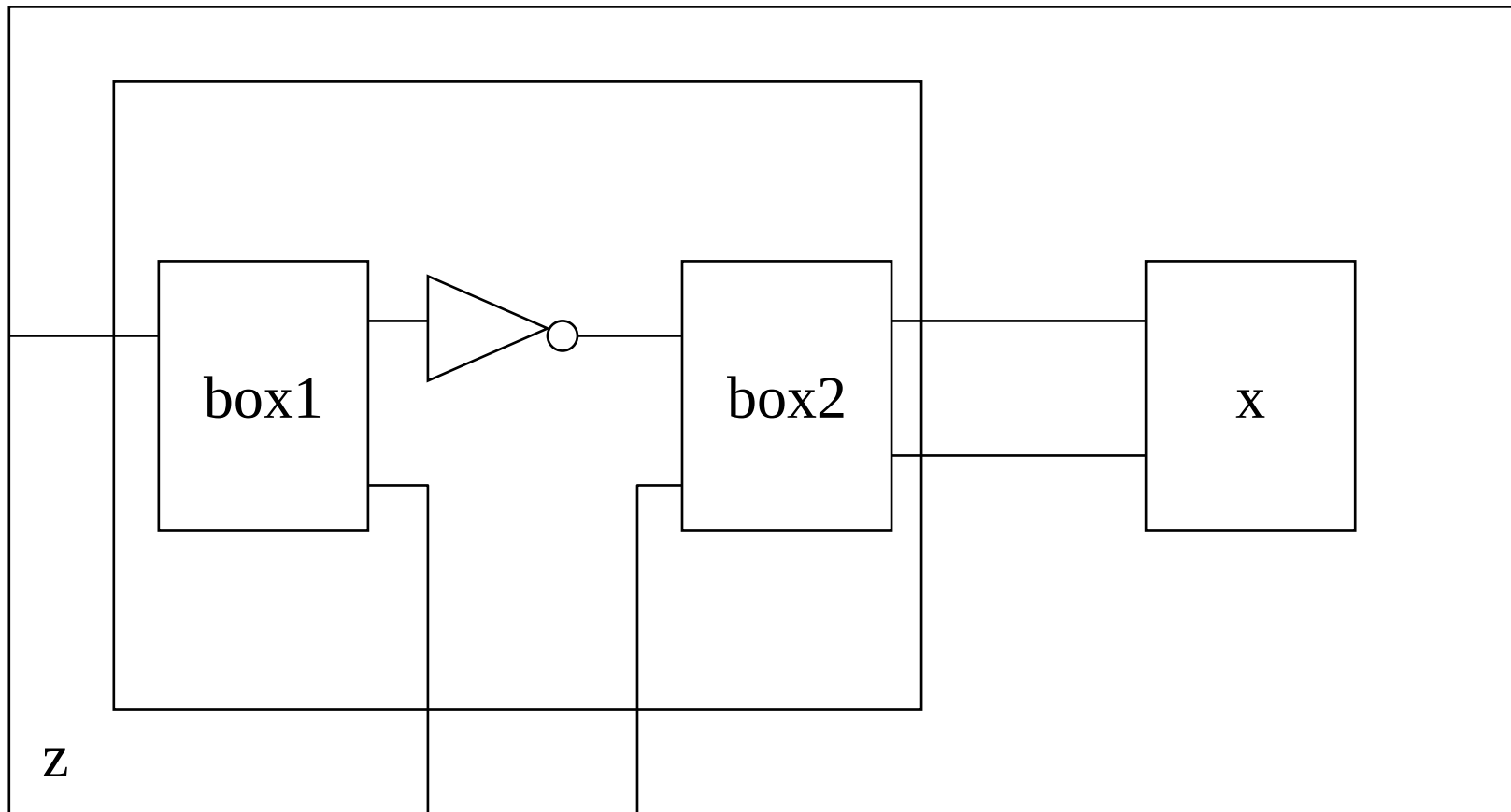


Instantiating component types

- 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

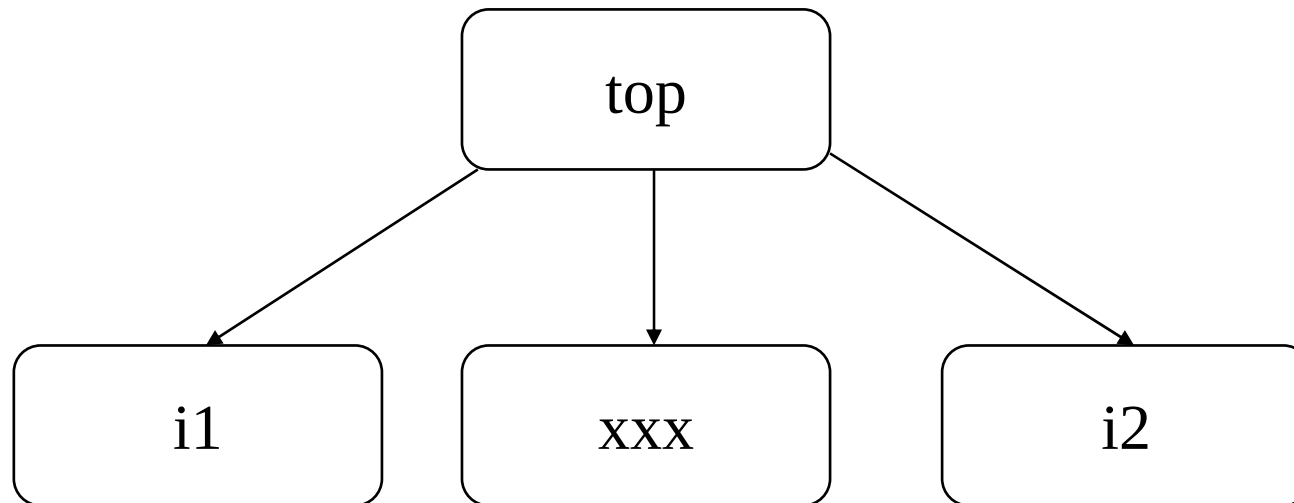
➤ Net list:

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

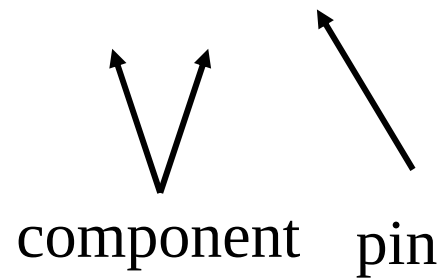
Component hierarchy



Hierarchical names

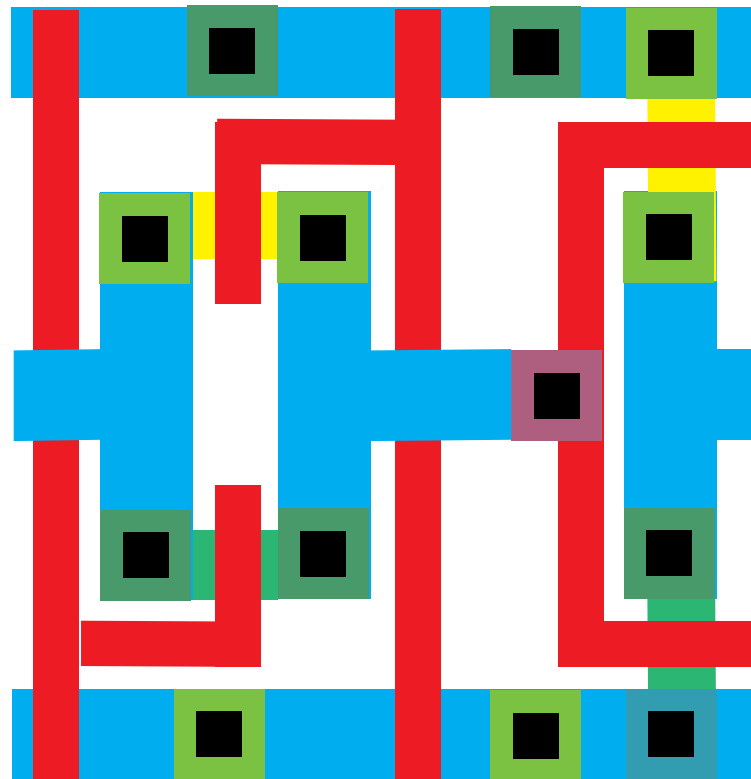
➤ Typical hierarchical name:

- top/i1.foo

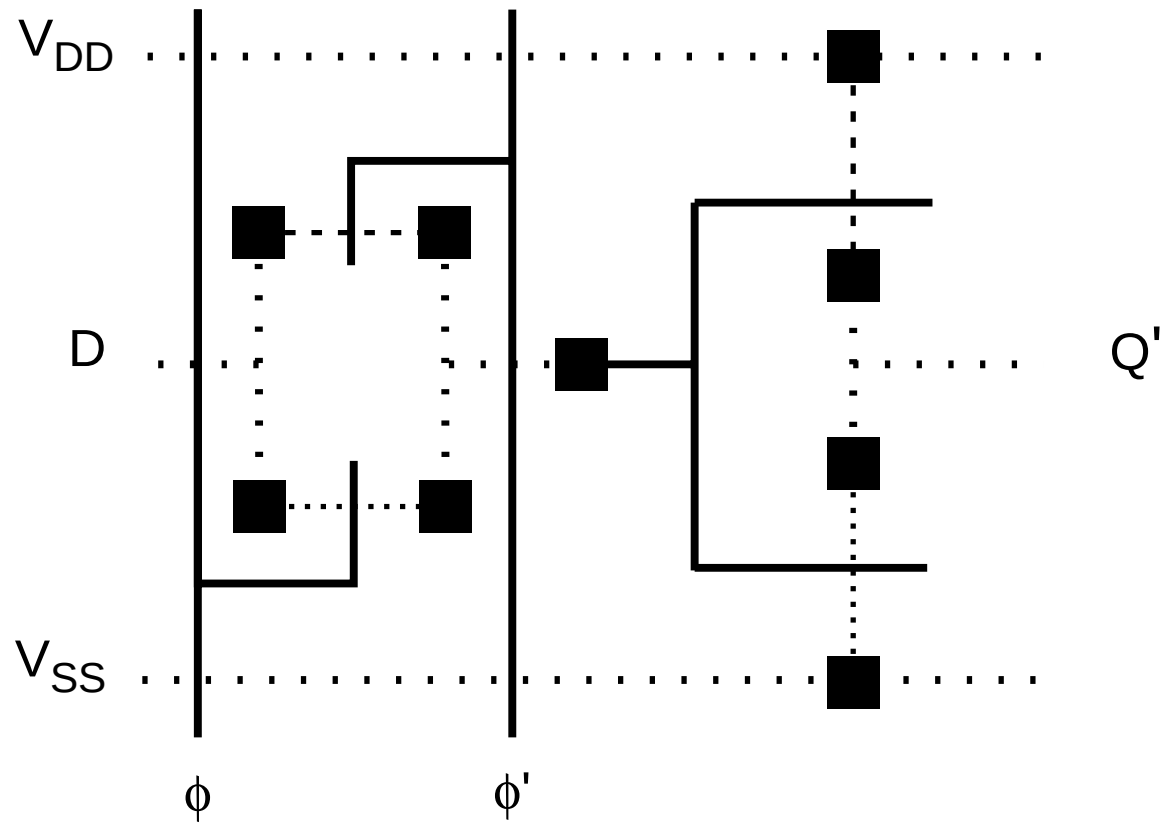


Layout and its abstractions

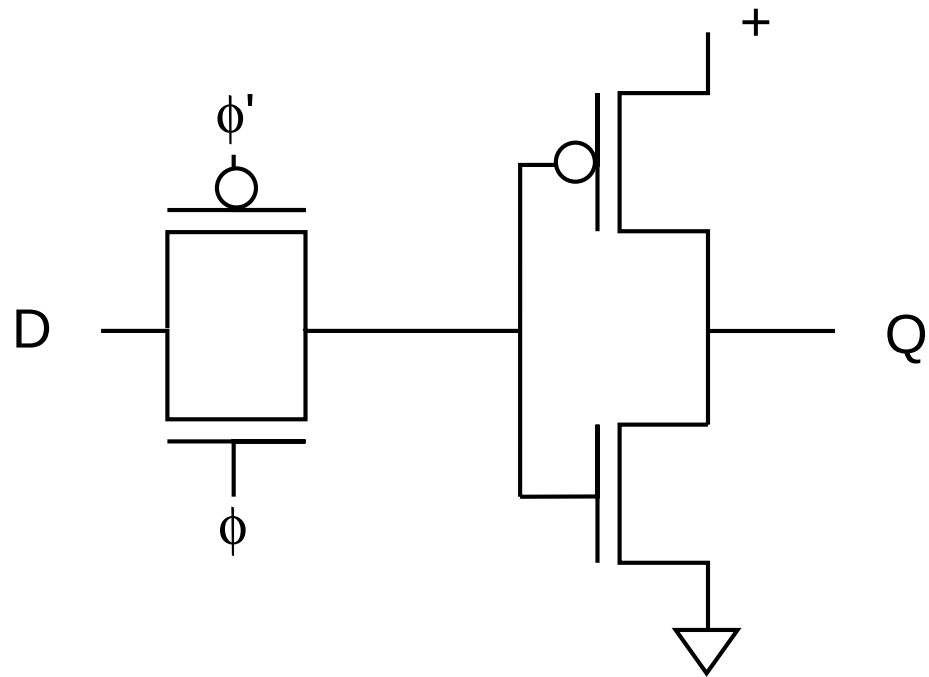
- Layout for dynamic latch:



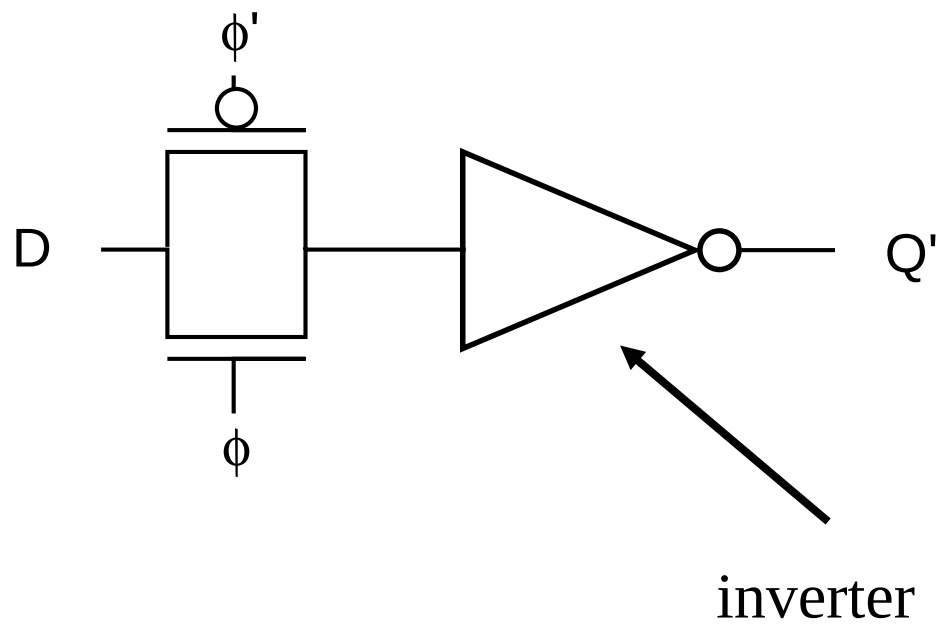
Stick diagram



Transistor schematic

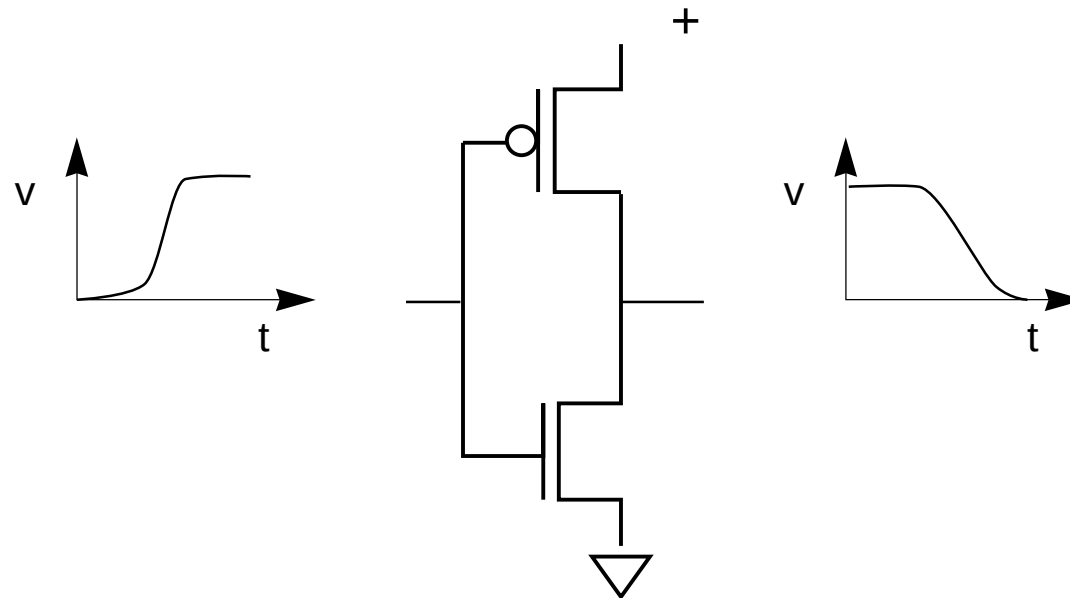


Mixed schematic



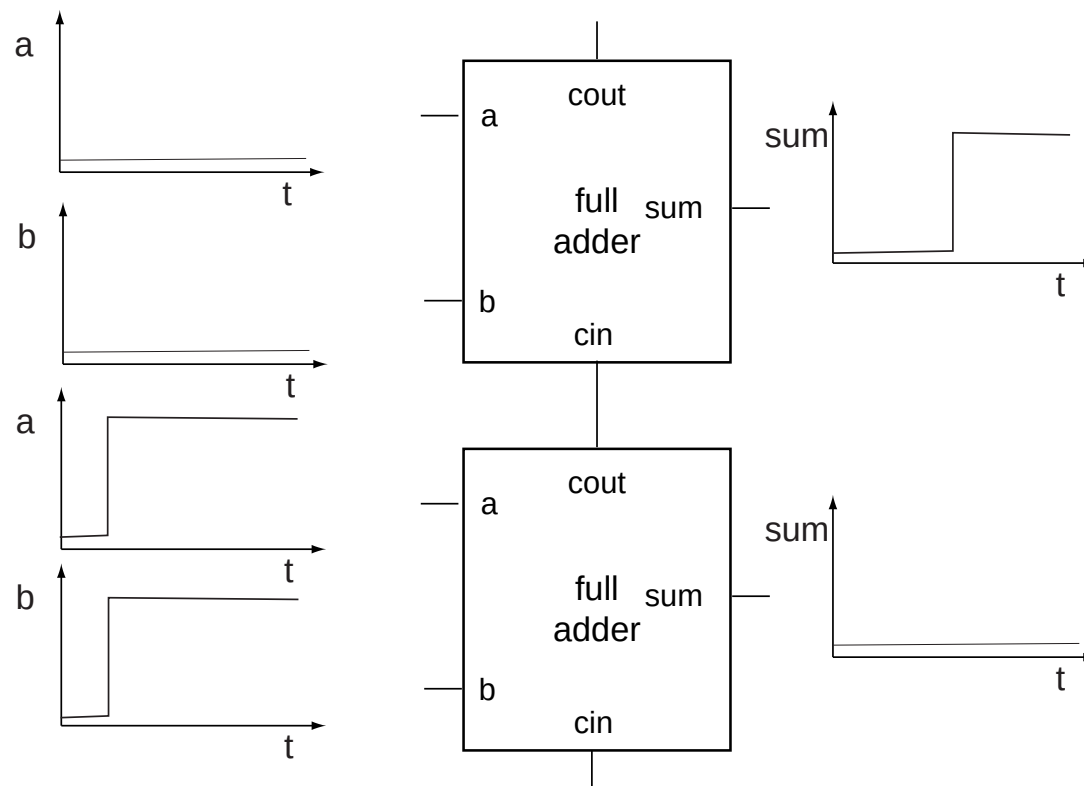
Circuit abstraction

- Continuous voltages and time:



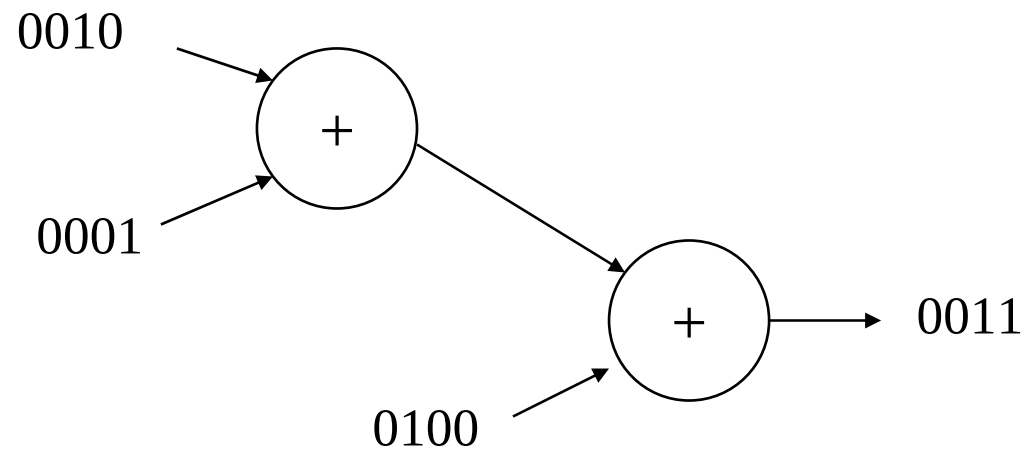
Digital abstraction

- 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 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.