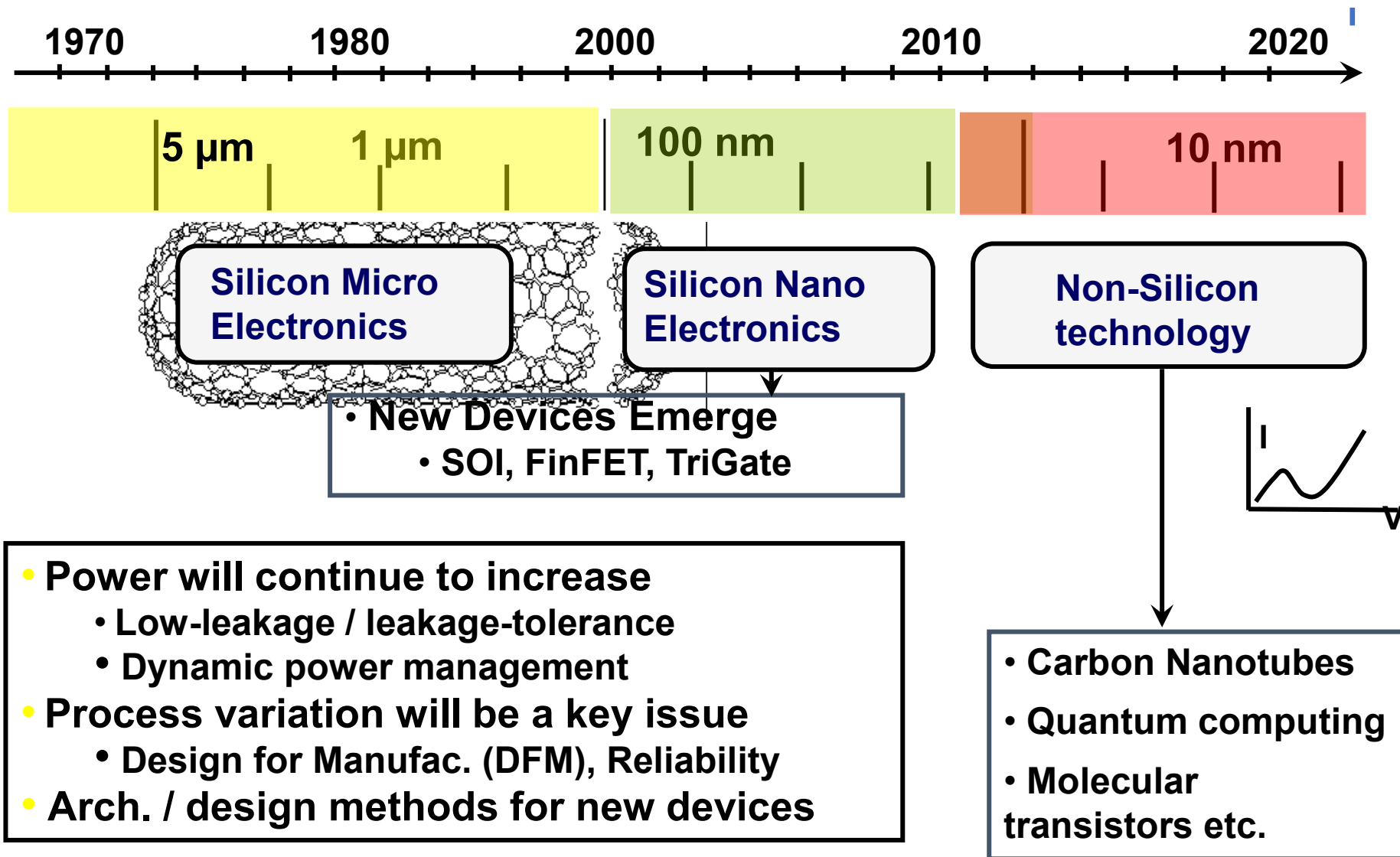


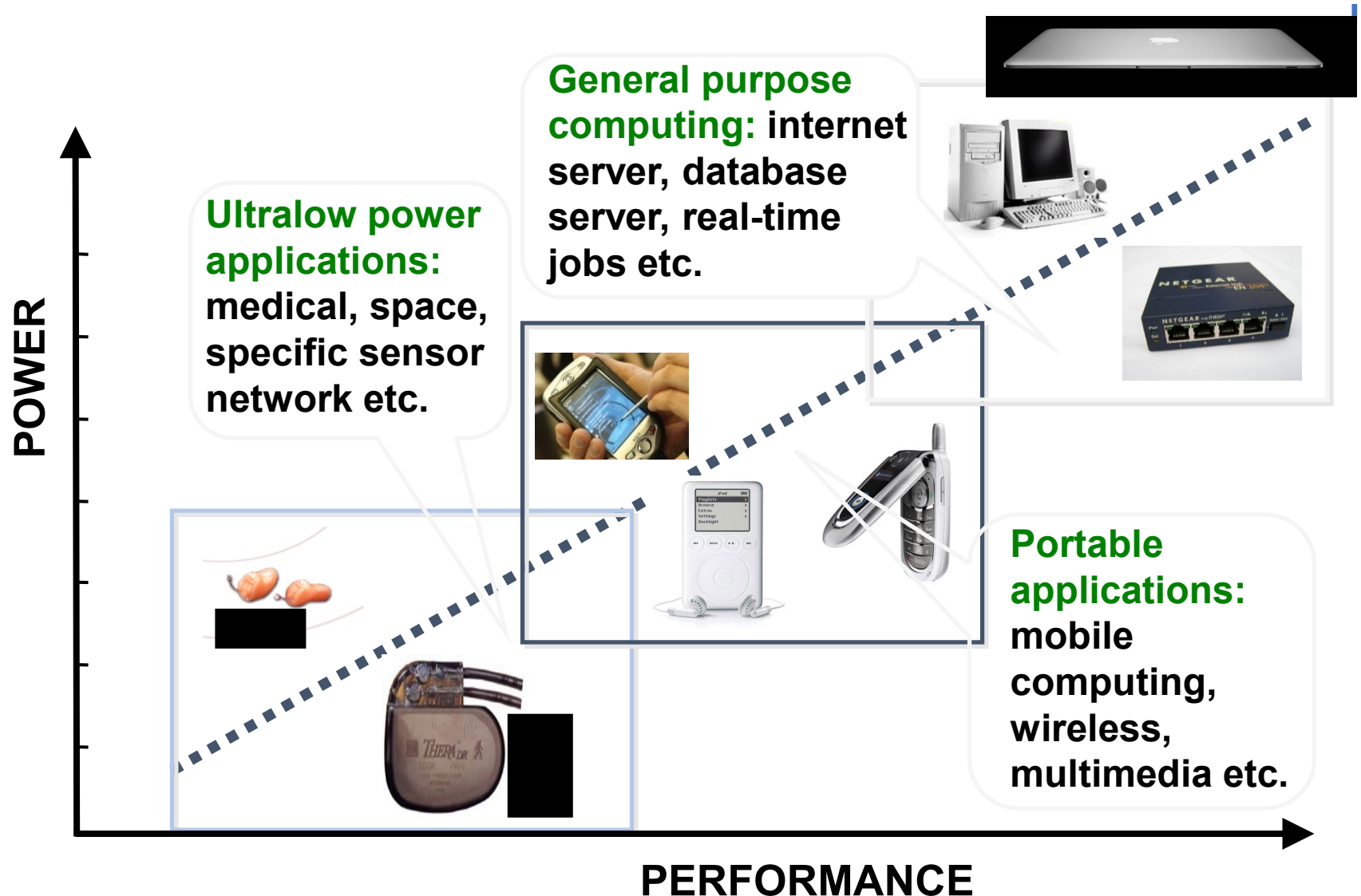
Chapters 2 & 3

Overview of VLSI Design and Test

Nanoelectronics

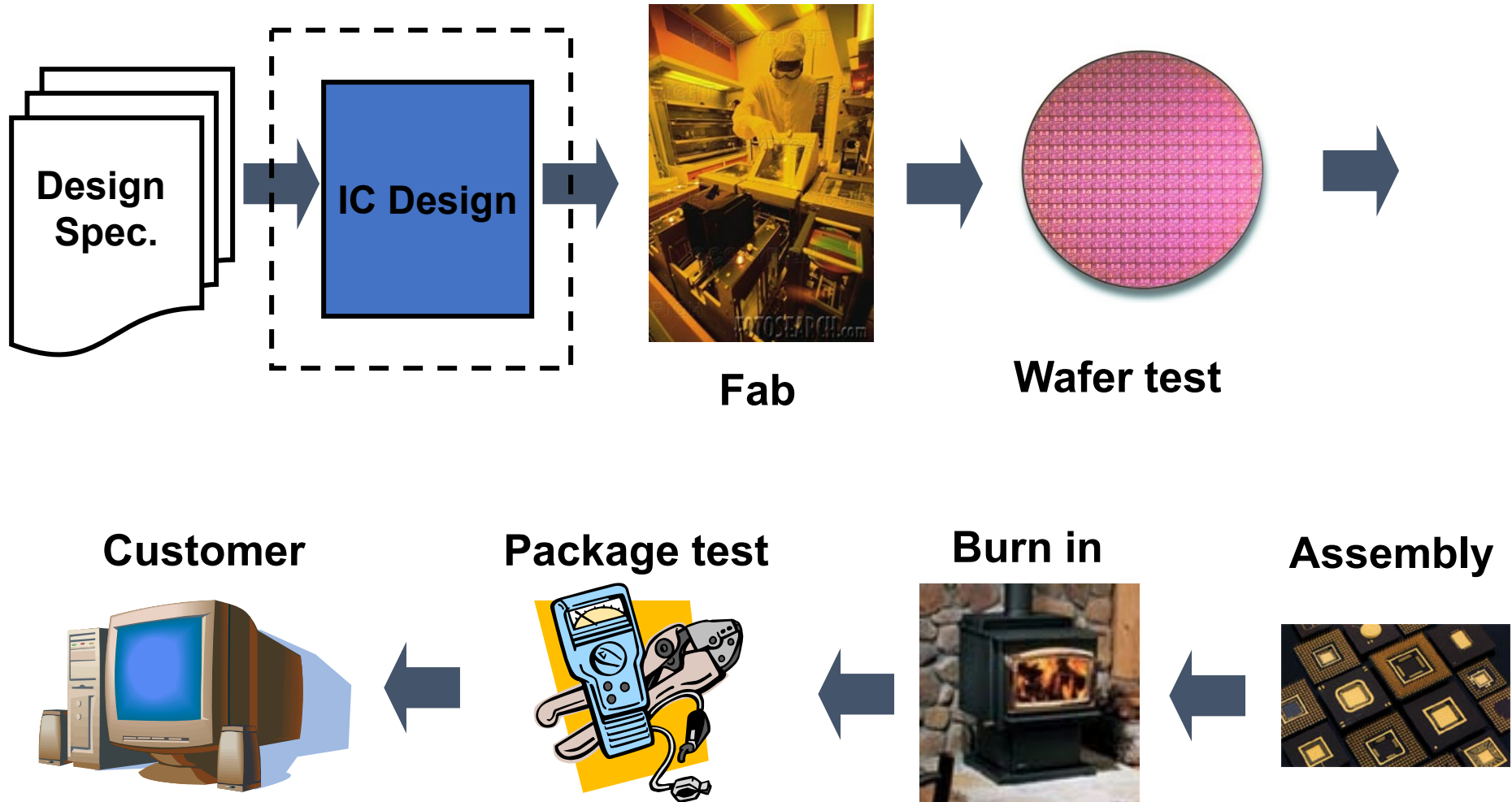


VLSI Applications

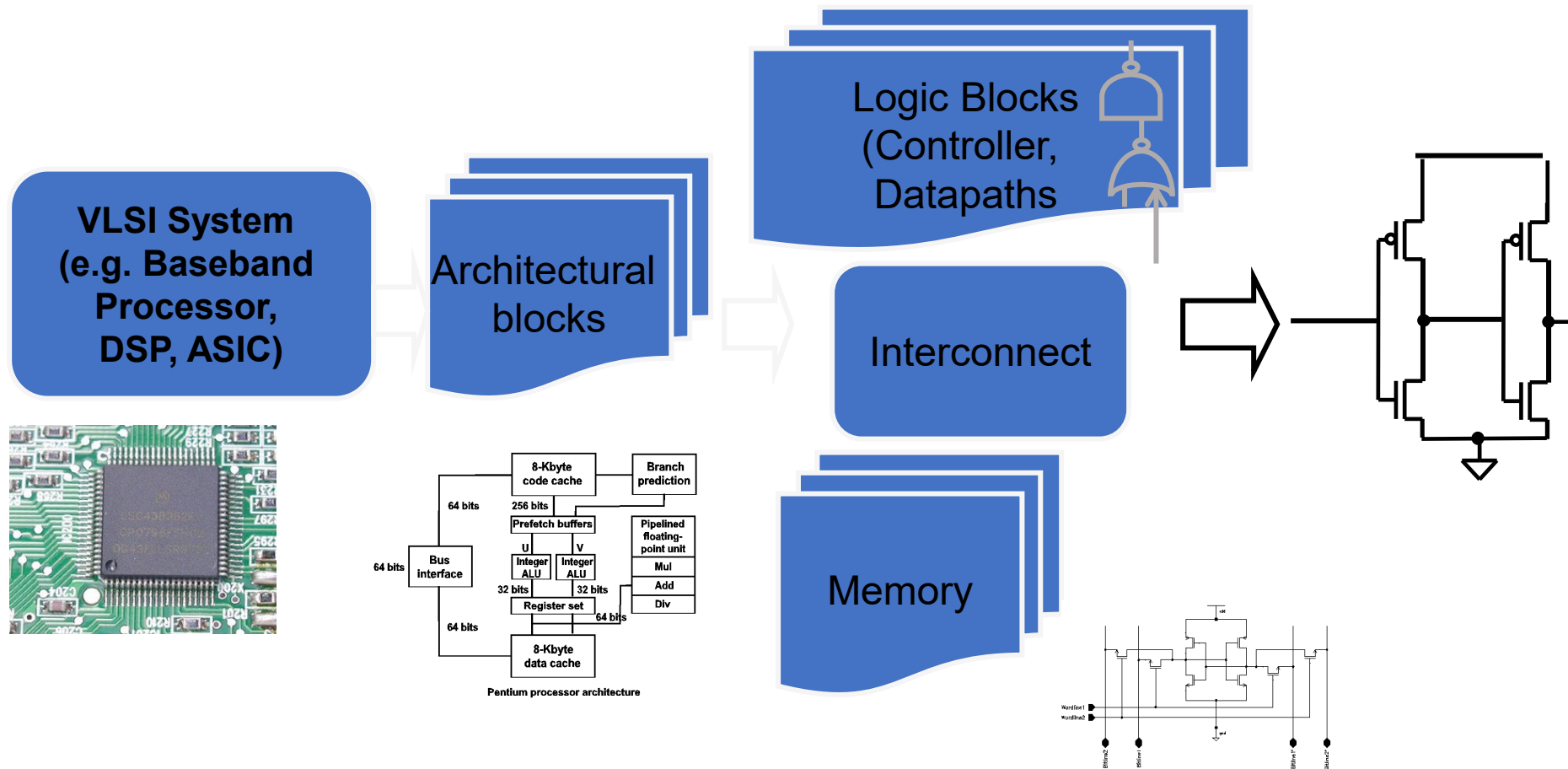


- Different applications have different power-performance demands

IC Design and Test Flow

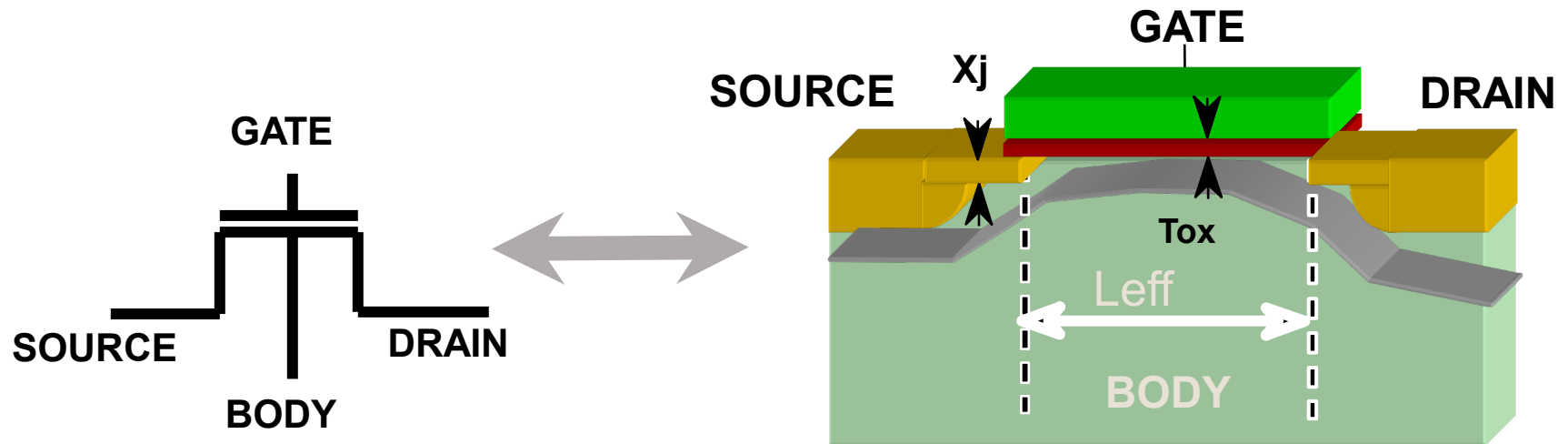


Nanoscale VLSI System



System -> Architecture -> Logic -> Transistor

Nanoscale Transistor and Technology Scaling



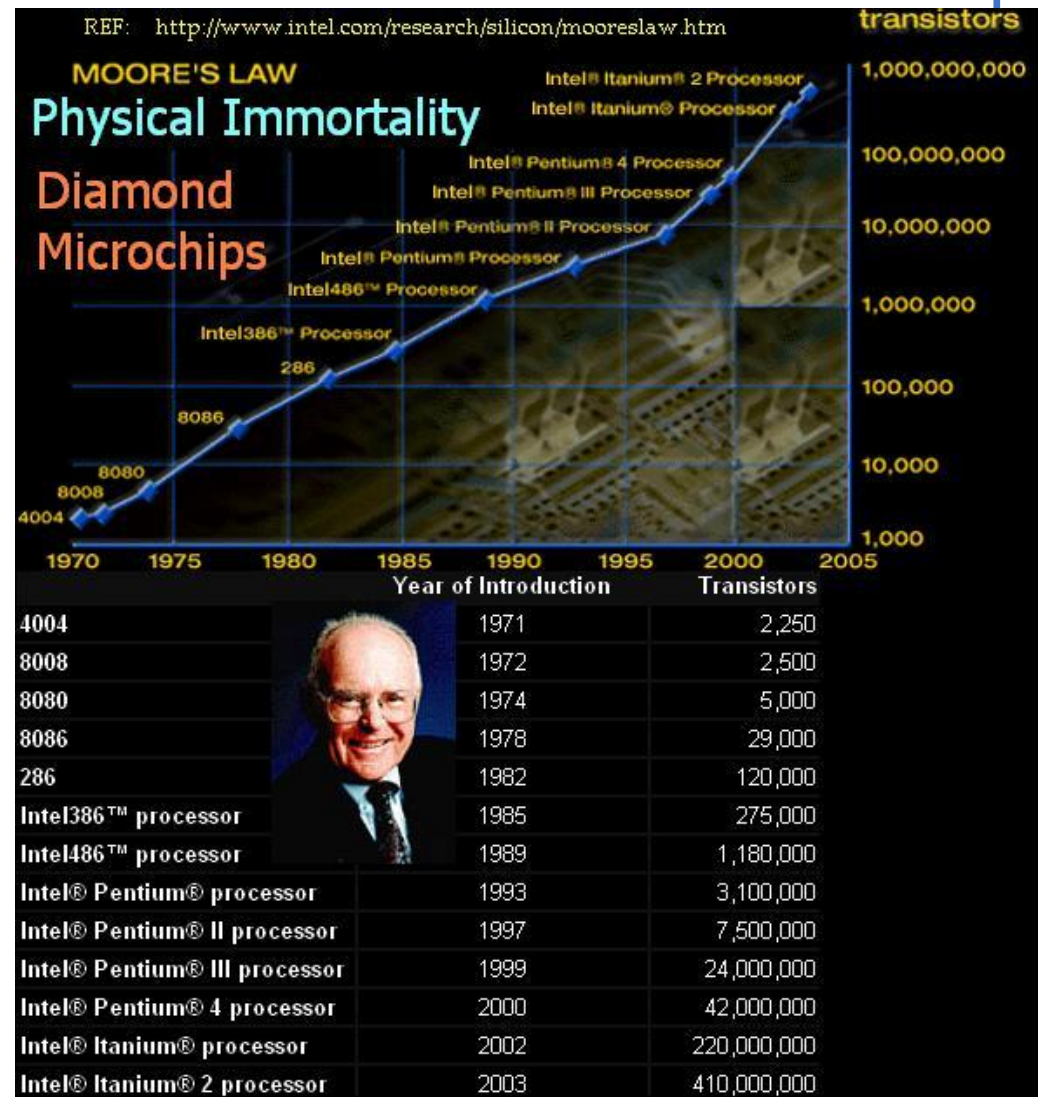
Dimensions scale down by 30%	Doubles transistor density
Oxide thickness scales down	Faster transistor, higher performance
Vdd & Vt scaling	Lower active power

- **Basic building block for the integrated circuit: both logic and memory**

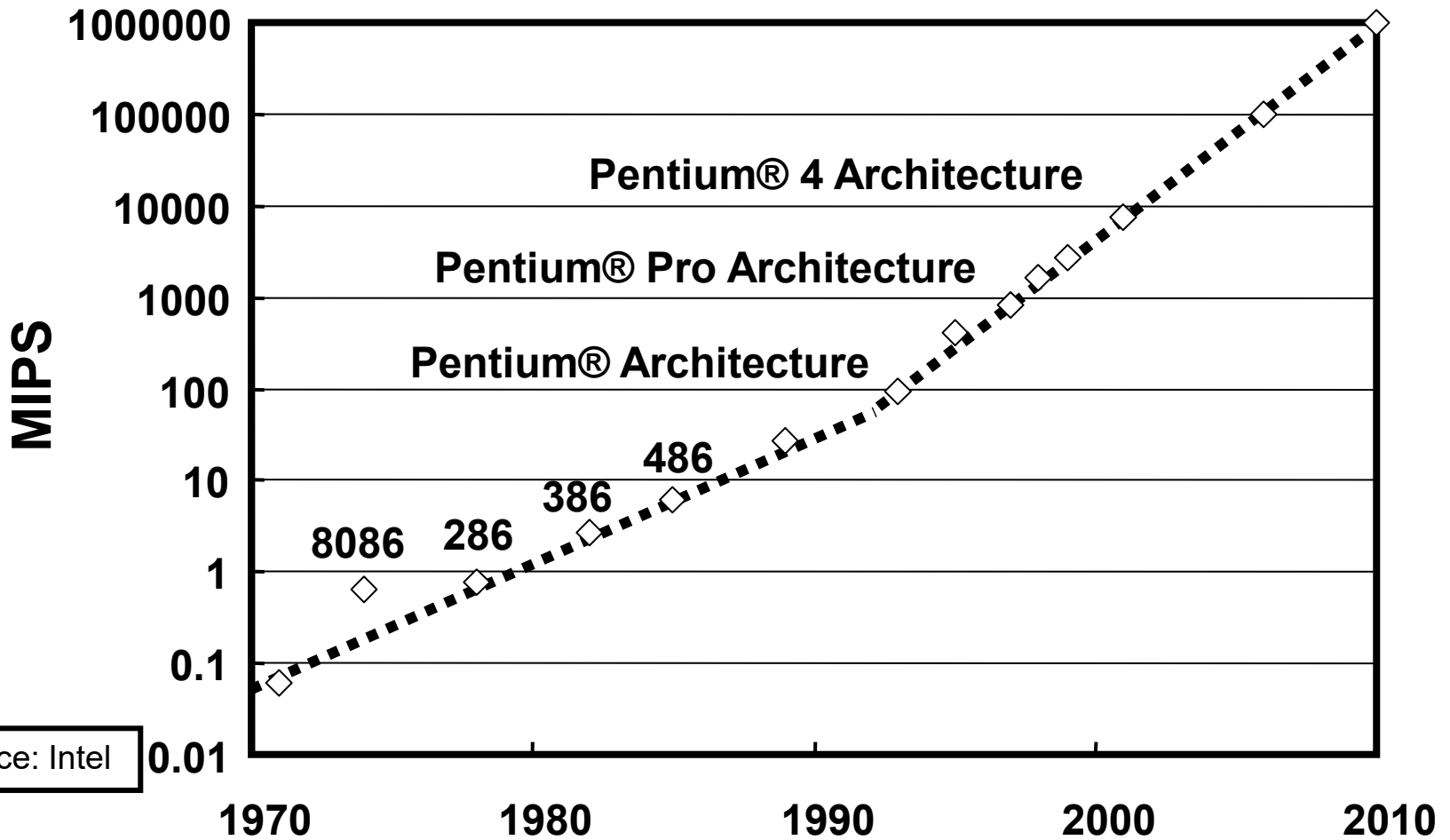
Moore's Law

In 1965, George Moore posited that roughly every two years, the number of transistors on microchips will double. Commonly referred to as Moore's Law, this phenomenon suggests that computational progress will become significantly faster, smaller, and more efficient over time.

Electronics Magazine, 19 April 1965



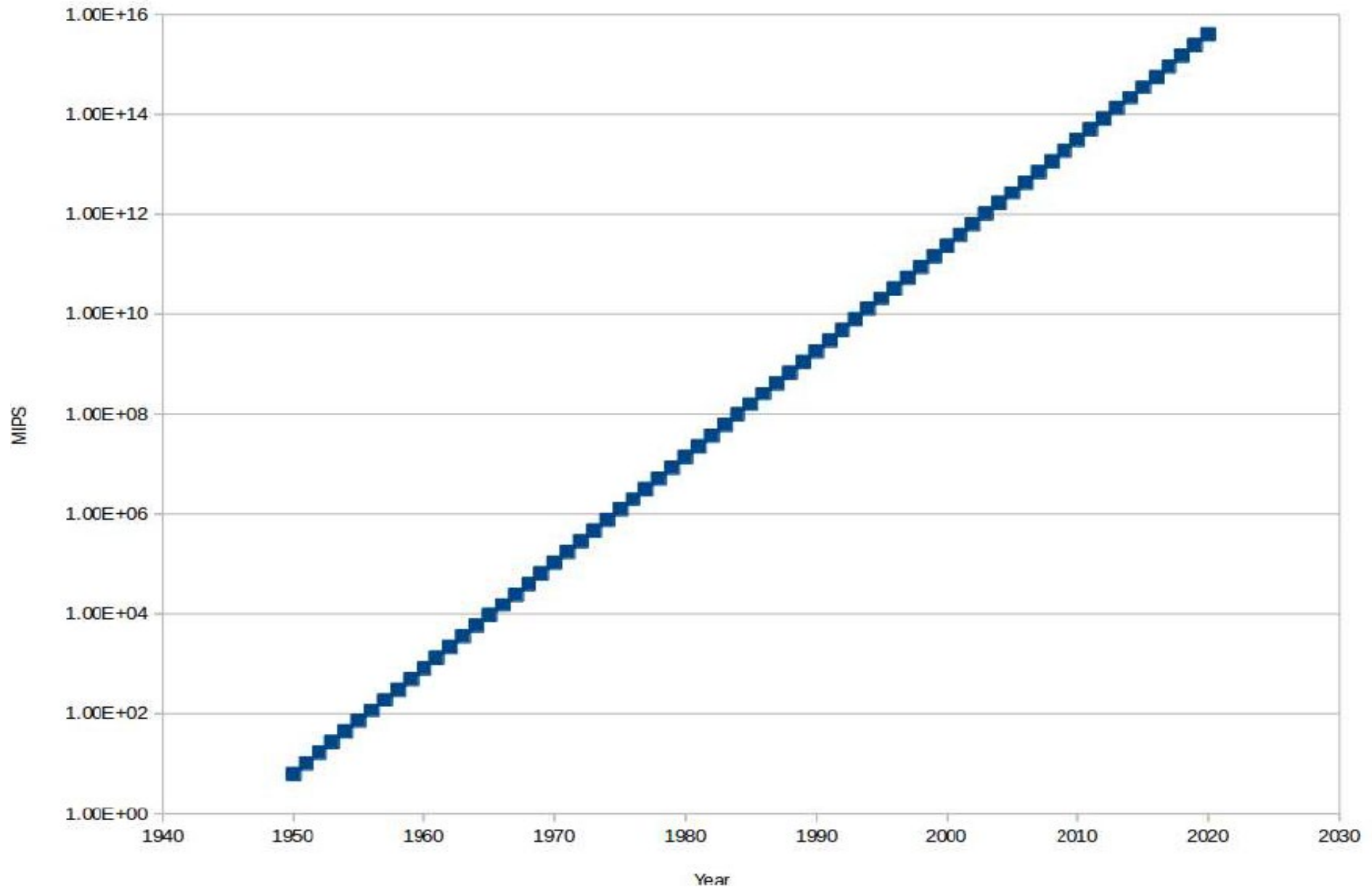
Exponential Growth in Computing Power



Source: Intel

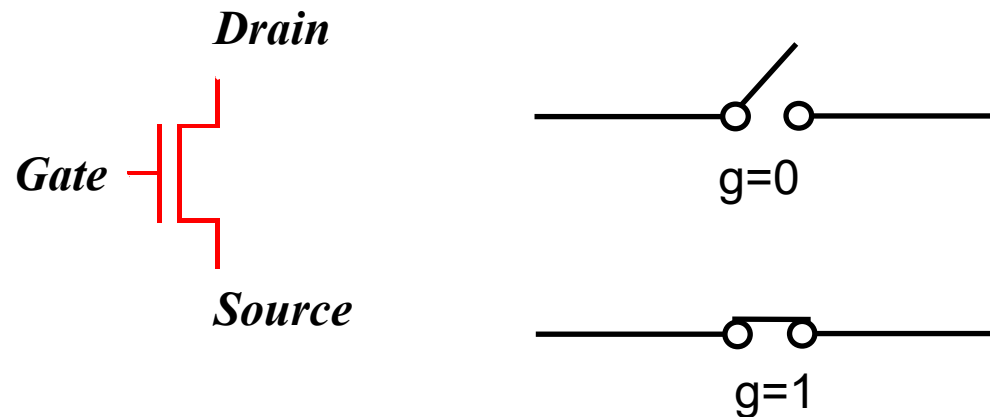
What is the key to the growth in computing power?

Quantum Computing



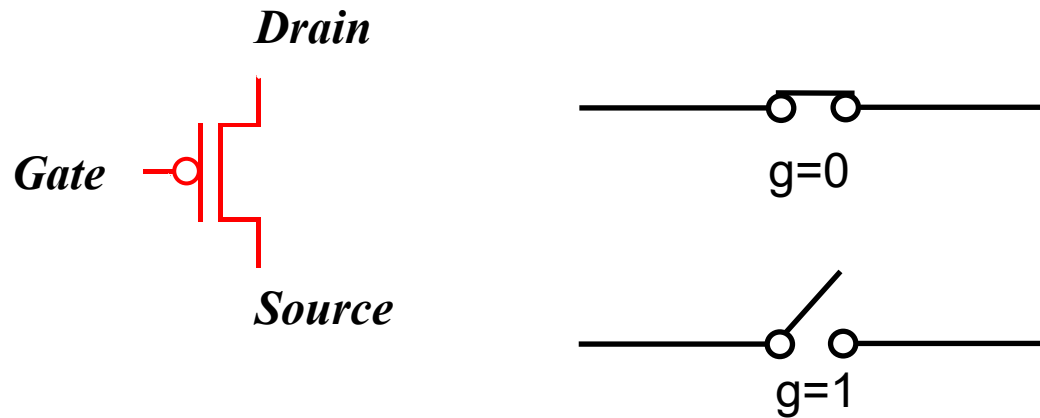
nMOS Transistor Switch Model

- If the gate is “high”, the switch is on
- If the gate is “low”, the switch is off



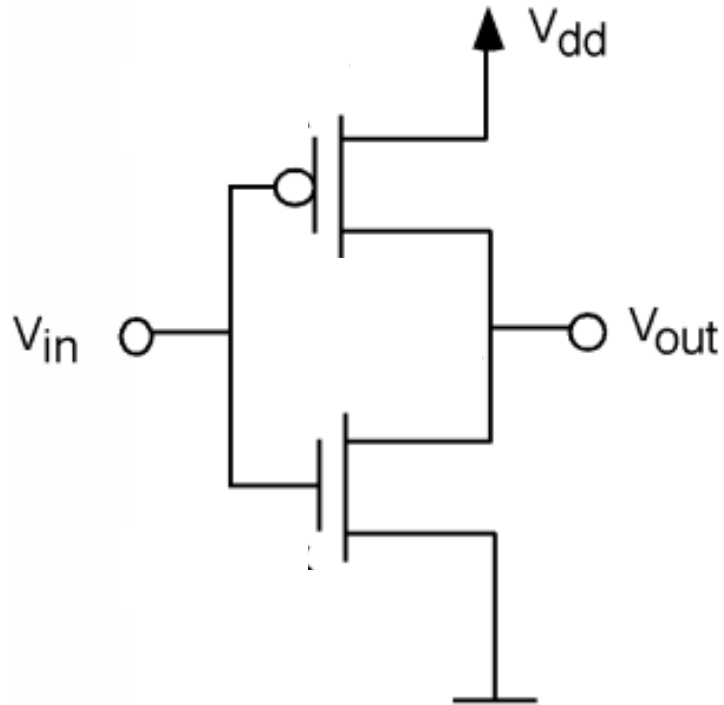
pMOS Transistor Switch Model

- If the gate is “low”, the switch is on
- If the gate is “high”, the switch is off



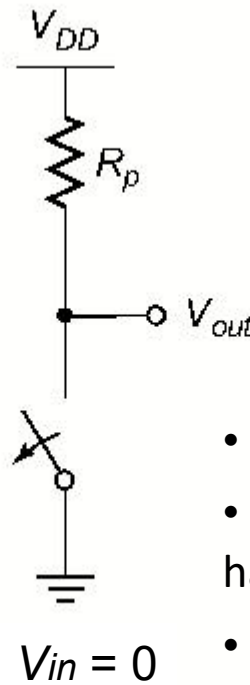
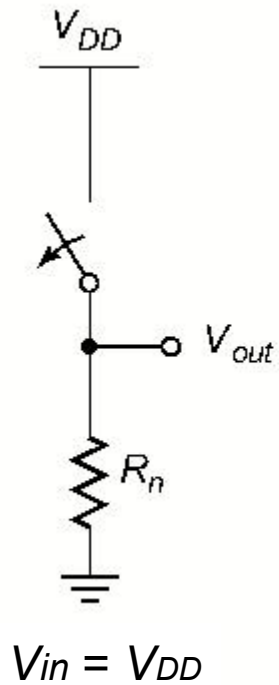
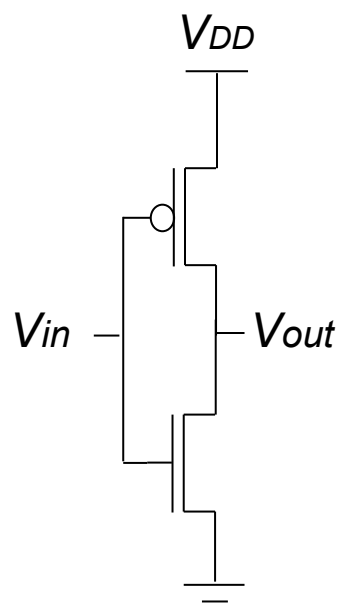
Solution: CMOS

I



- No static current flow ideally
- Less current means less power

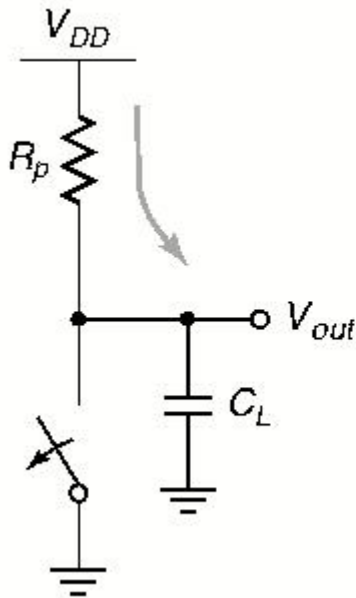
CMOS Inverter First-Order DC Analysis



$$\begin{aligned} V_{OL} &= 0 \\ V_{OH} &= V_{DD} \end{aligned}$$

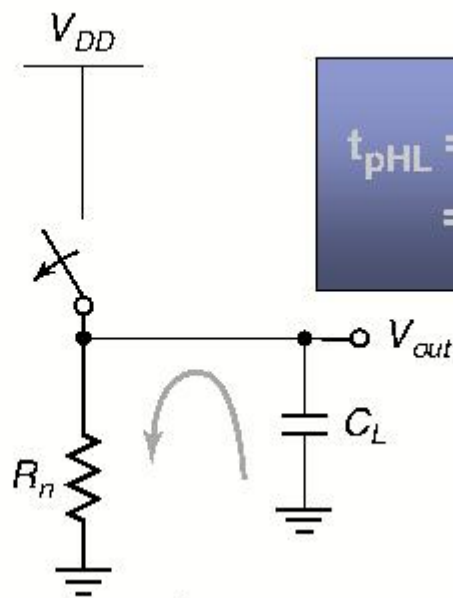
- High noise margin
- Ratioless: Transistors don't need to have big size different
- Low output impedance
- Extremely high input impedance
- Static power—leakage current*Vdd

CMOS Inverter: Transient Response (RC delay model)



$V_{in} = V_{DD} \rightarrow 0$

Output: Low-to-High



$V_{in} = 0 \rightarrow V_{DD}$

High-to-Low

$$t_{pHL} = f(R_{on} \cdot C_L) \\ = 0.69 R_{on} C_L$$

To reduce delay:

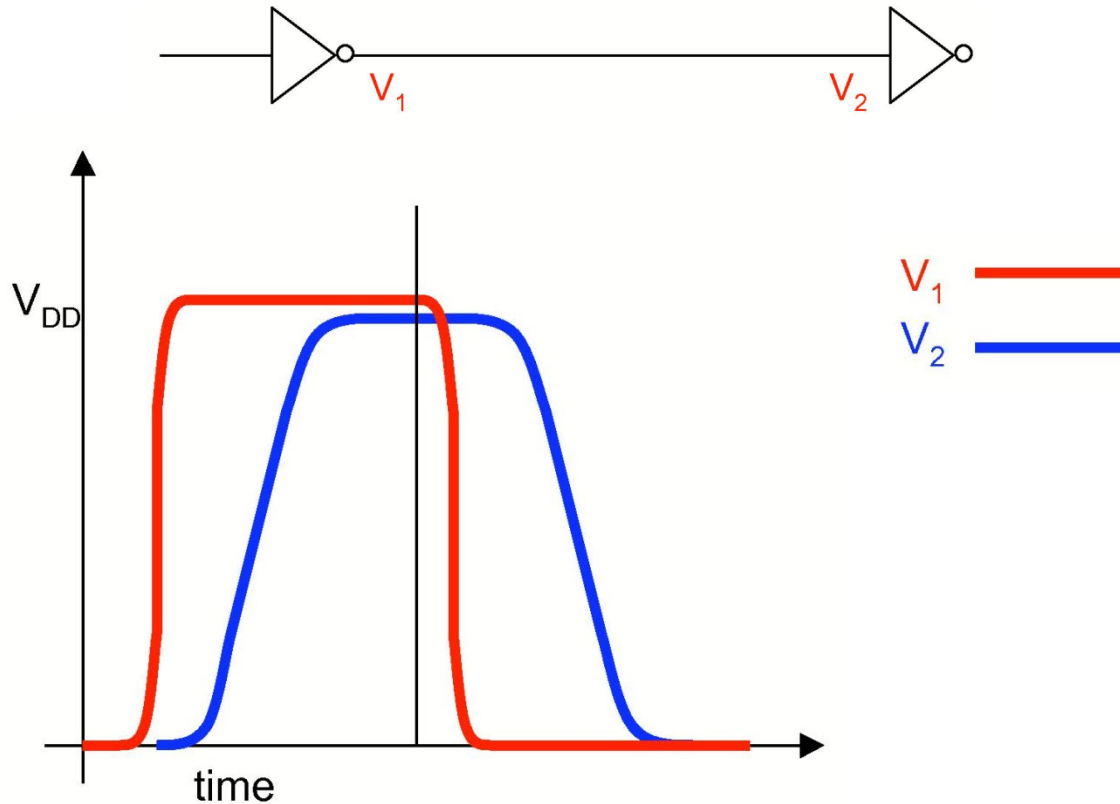
- Reduce C_L
- Reduce $R_{p,n}$
- Increase W/L ratio

**Load changing will
change propagation
delay**

C_L is composed of the drain diffusion capacitances of the NMOS and PMOS transistors, the capacitance of connecting wires, and the input capacitance of the fan-out gates

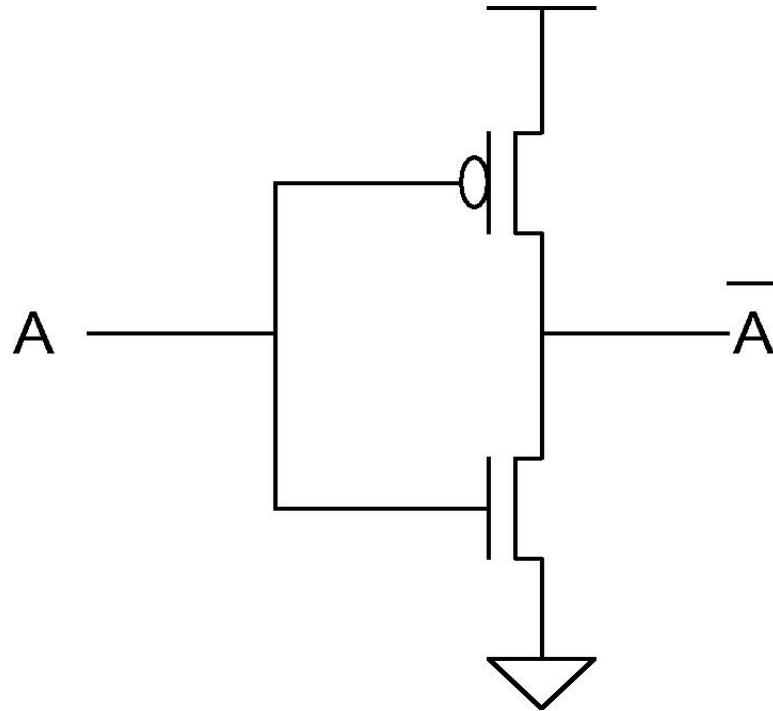
Performance Characterization

- Interconnect delay



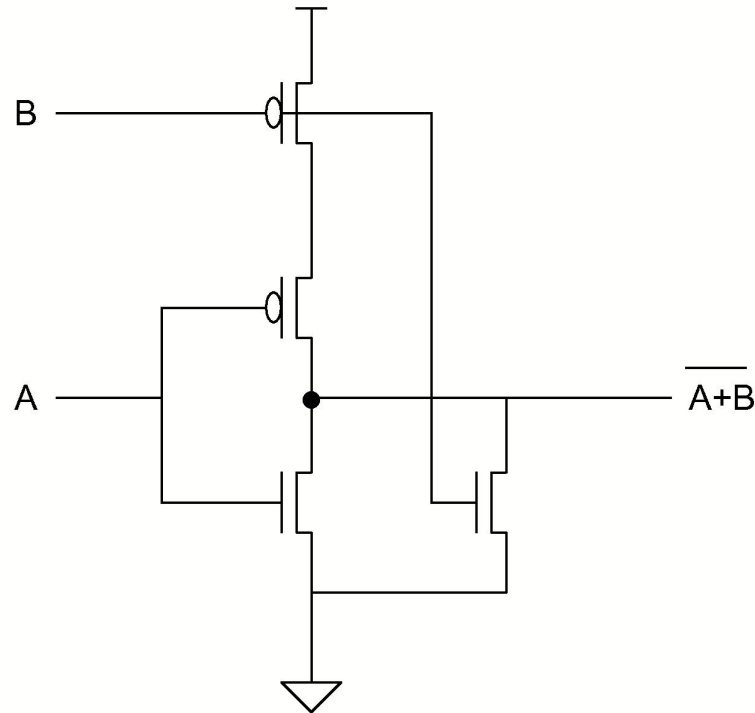
CMOS Logic Implementations

- Inverter



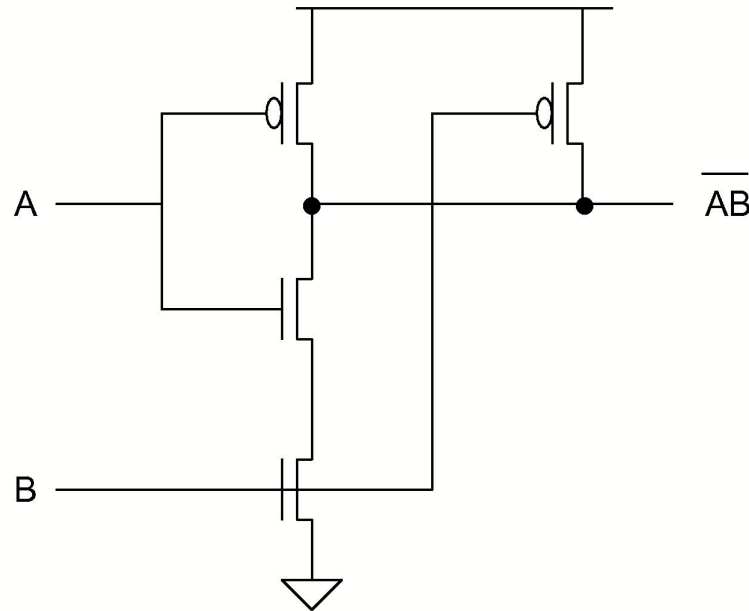
CMOS Logic Implementations

- NOR



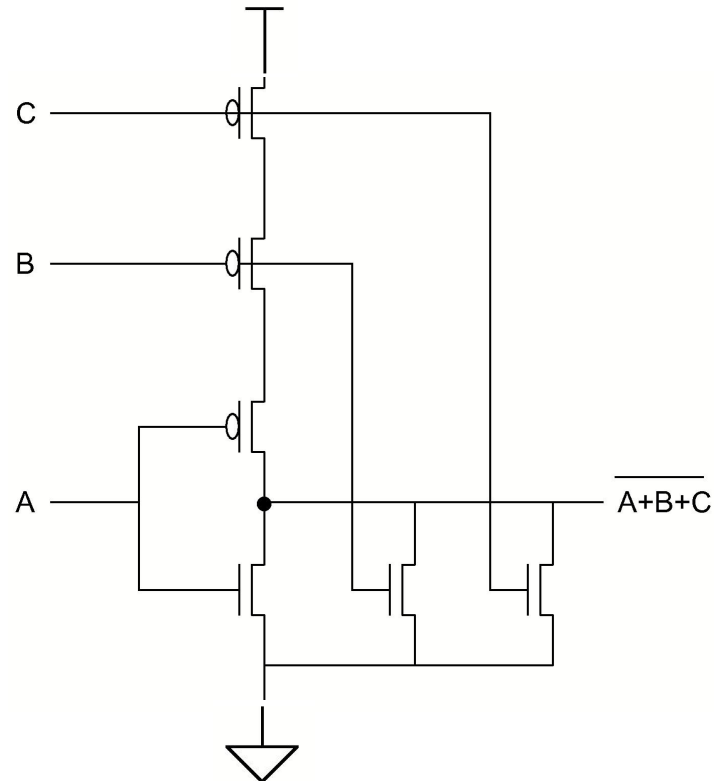
CMOS Logic Implementations

- NAND



CMOS Logic Implementations

- Multi-input NOR



What is VLSI design?

- The process of creating an integrated circuit from specifications to fabrication

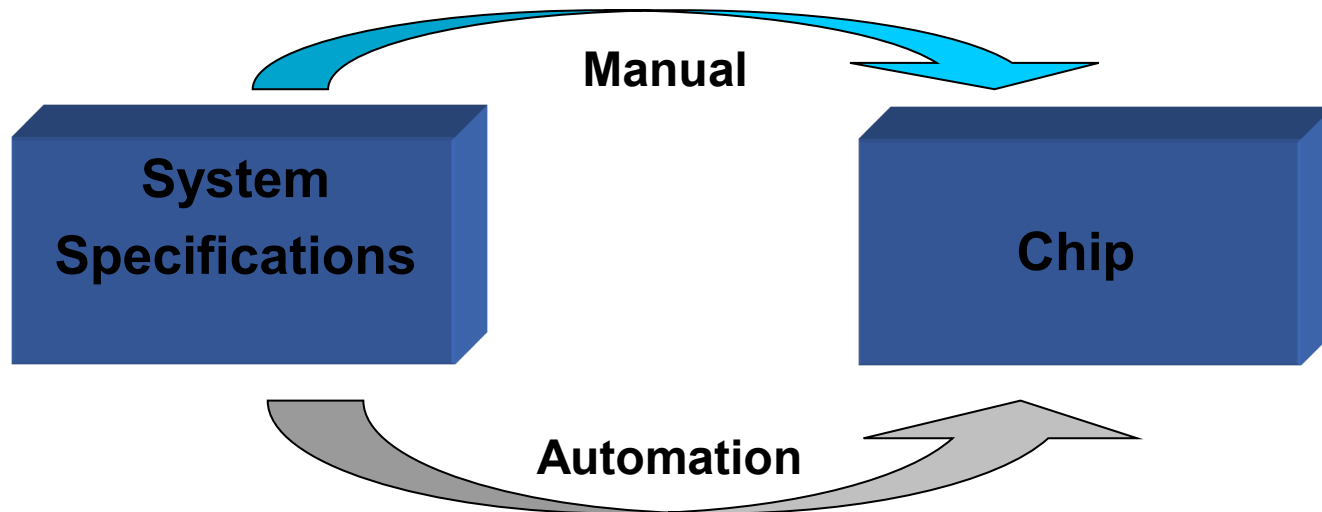
What is an integrated circuit?

- A single integrated component that contains the primary elements of an electrical circuit: transistors, wiring, resistors, capacitors, etc.

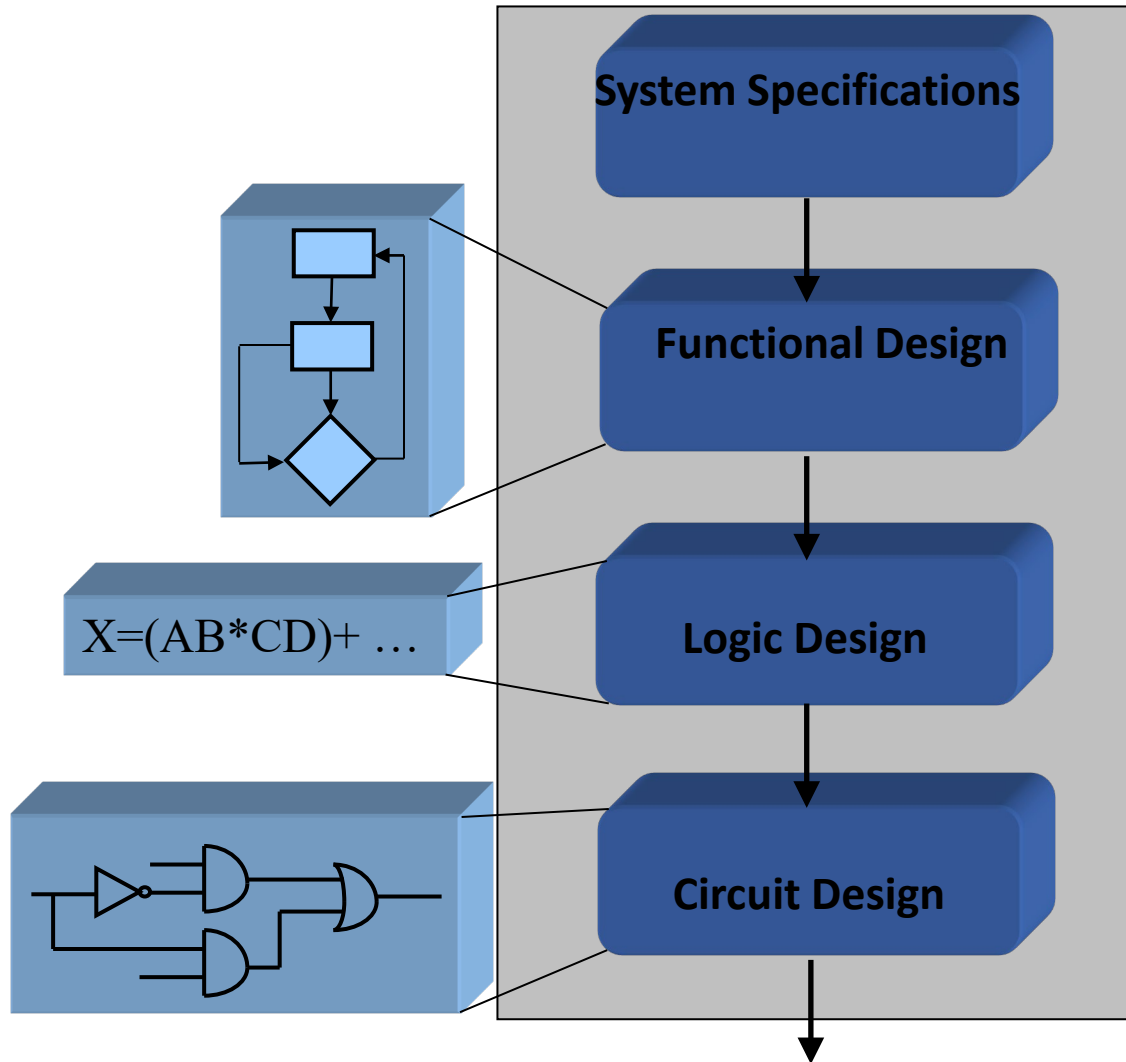


VLSI Design Automation

- Large number of components
- Optimize requirements for higher performance
 - Performance relates to speed, power and size.
- Time to market competition
- Cost
 - Using computer makes it cheaper by reducing time-to-market.

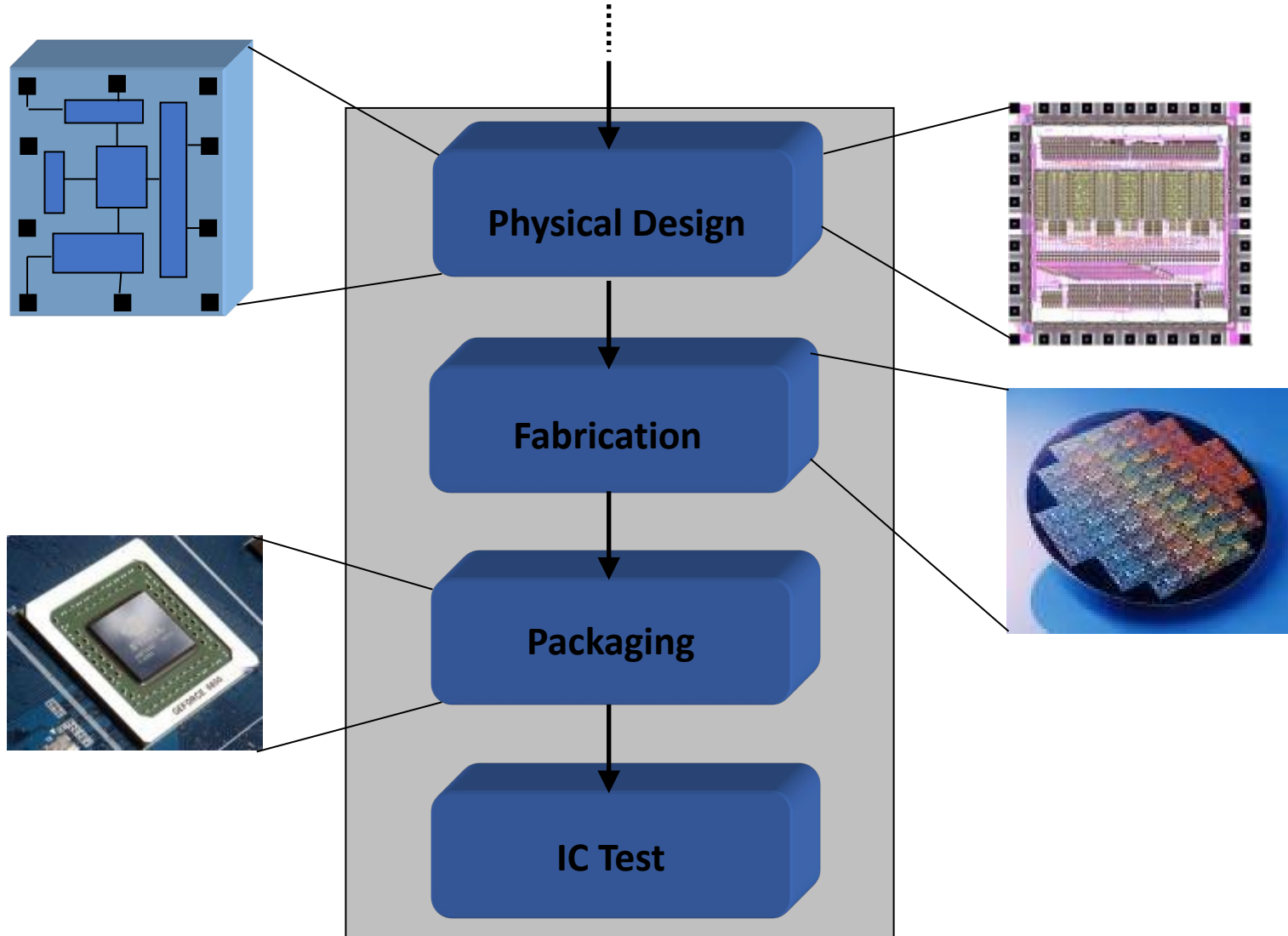


VLSI Design Cycle



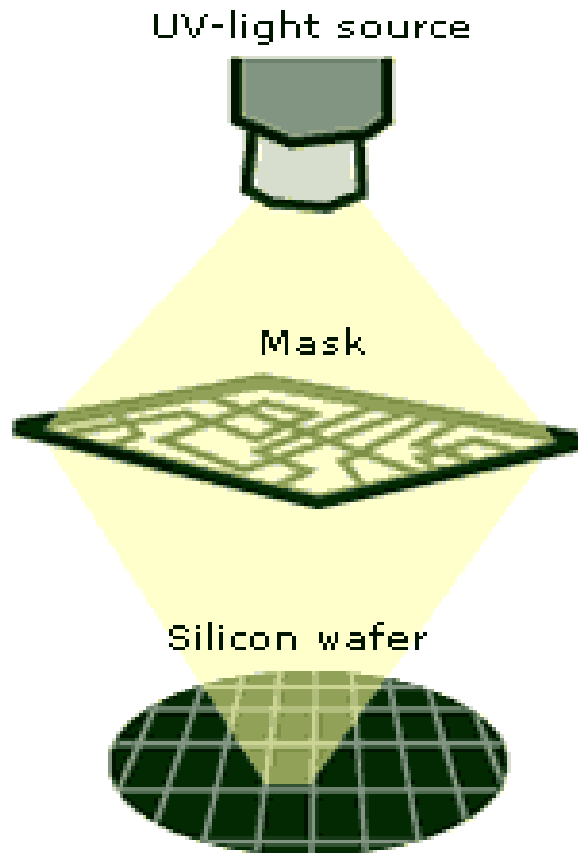
VLSI Design Cycle (Cont)

I



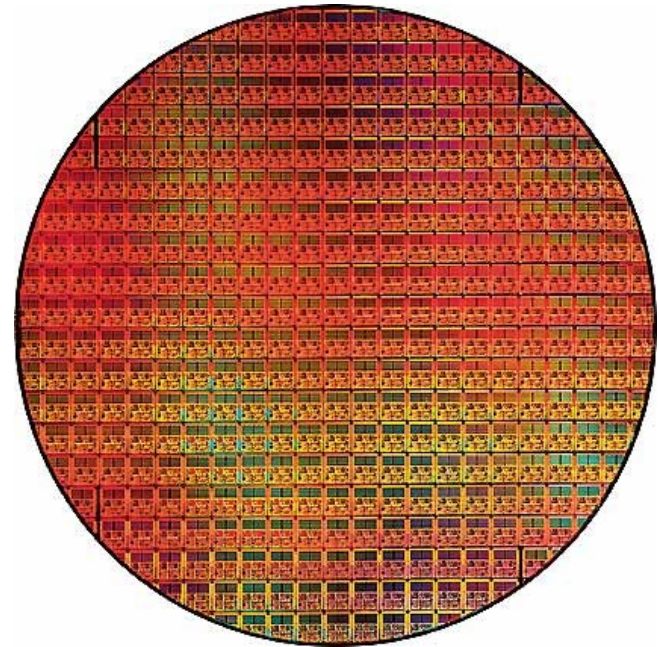
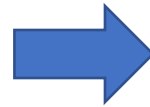
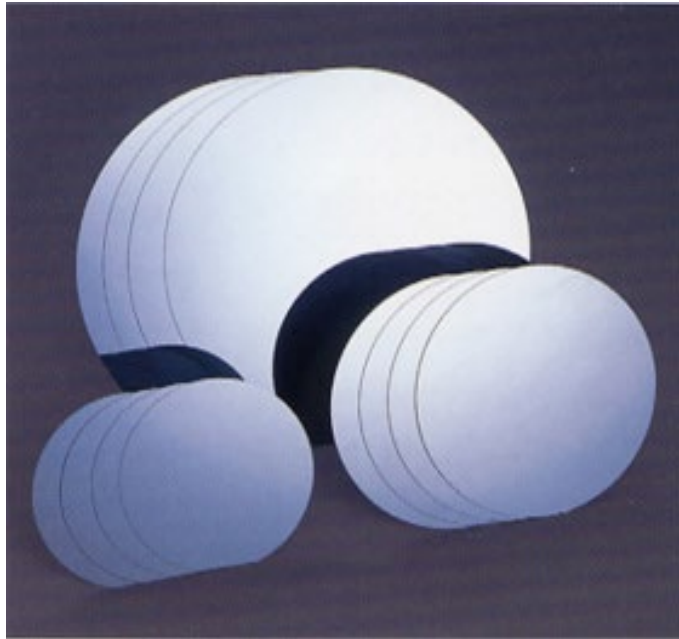
Semiconductor Processing

- How do we make a transistor?



- How do you control where the features get placed?
 - Photo lithography masks

Wafer Processing



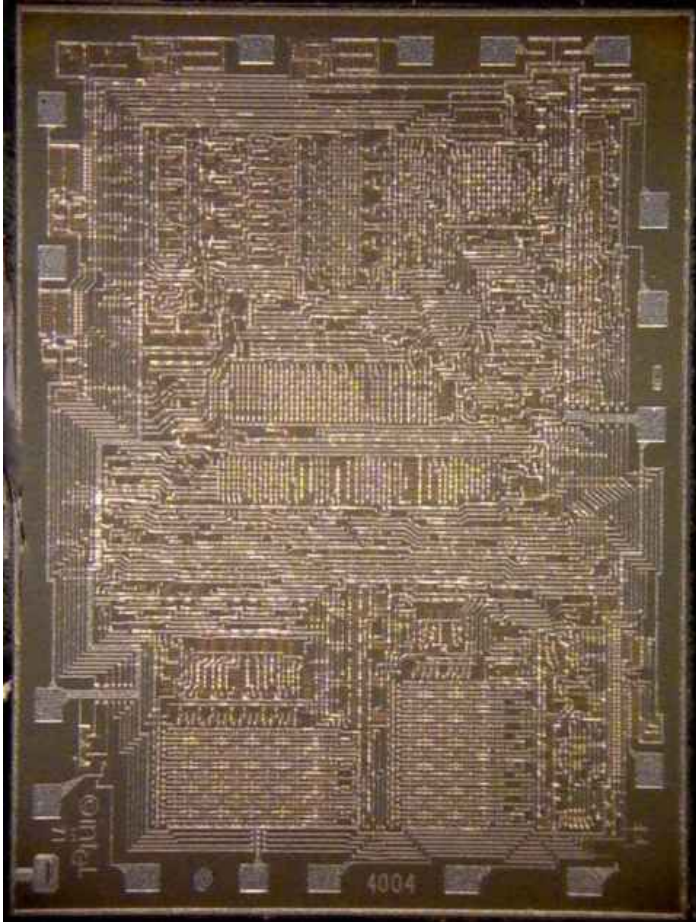
D: 11.8inch/300mm; T: 775um

D: 17.7inch/450mm; T: 925um

D: 26.6inch/675mm; ?

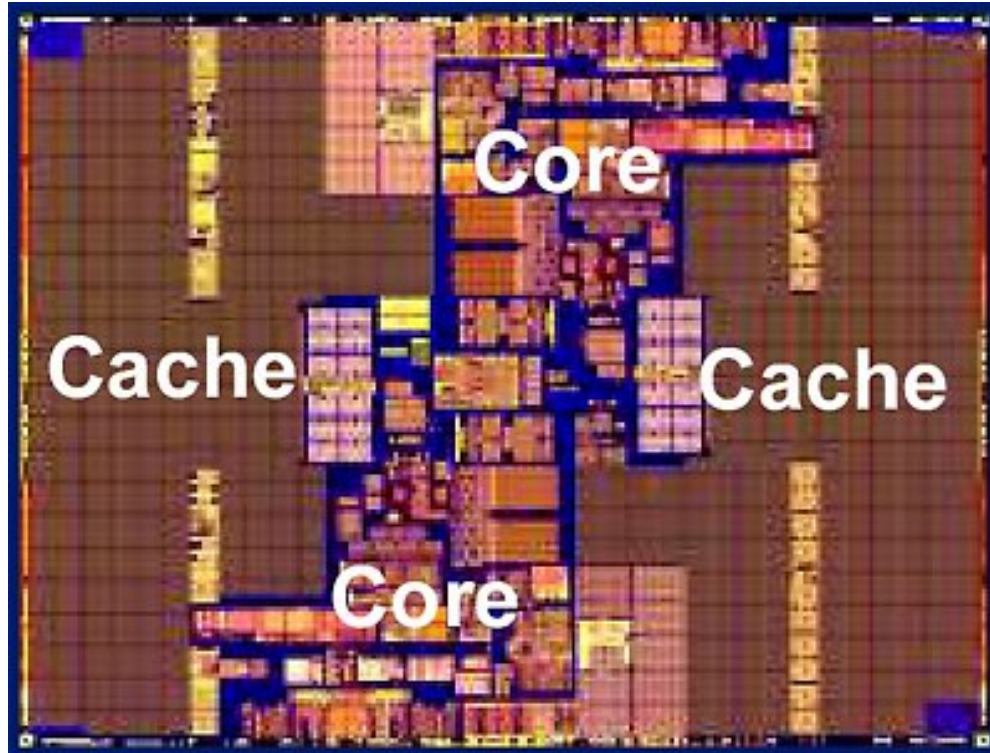
Processor Performance History

Intel 4004



- First microprocessor
- Designed in 1971
- 2300 transistors
- 10-um process
- ~100 KHz

Intel Processor



- **Intel Itanium**
released in 2005
- 1.72 Billion transistors
- 90-nm process
- 2 GHz
- Intel Core i9-13900K
released in 2022
- 2.95B transistors
- 7-nm process
- 5.2 GHz

Design Methodology

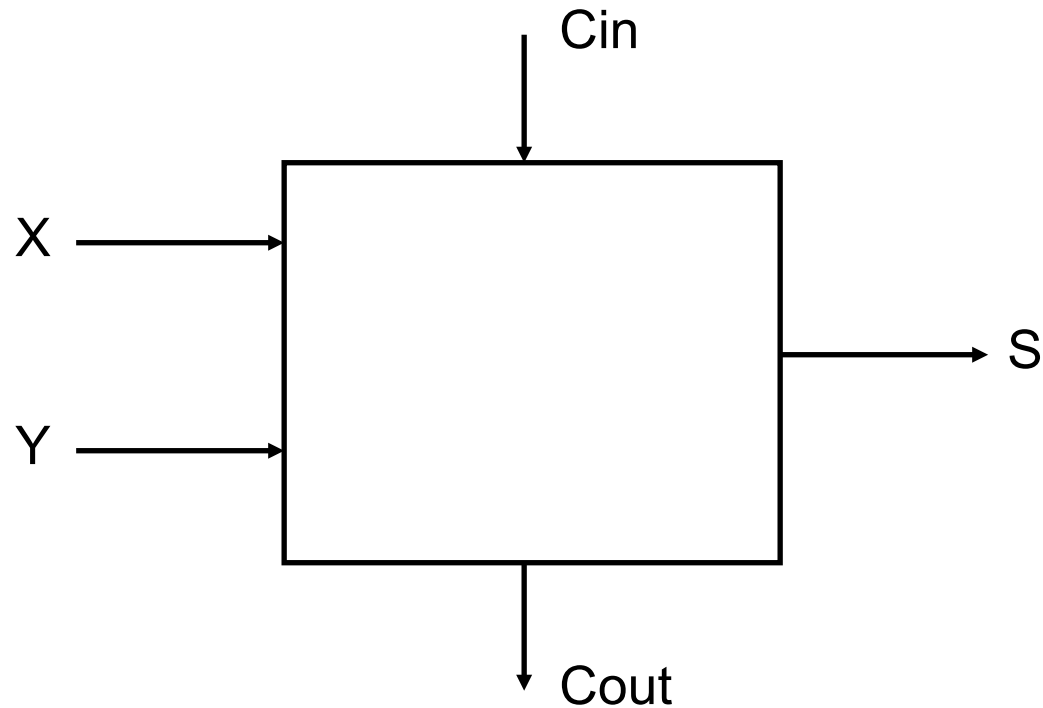
- Functional specification
 - What does the chip do?
- Behavioral specification
 - How does it do it? (abstractly)
- Logic design
 - How does it do it? (logically)
- Layout
 - How does it do it? (physically)

Design Constraints

- Budget
 - Total cost
- Silicon area
- Power requirements
 - Dynamic
 - Static
- Speed
 - Performance
- Schedule
 - Time to market

Functional Specification

- Full adder



Behavioral Specification

- VHDL
- Verilog

entity adder is

-- *i0*, *i1* and the carry-in *ci* are inputs of the adder.

-- *s* is the sum output, *co* is the carry-out.

port (*i0*, *i1* : in bit; *ci* : in bit; *s* : out bit; *co* : out bit);

end adder;

architecture rtl of adder is

begin -- This full-adder architecture contains two concurrent assignment.

-- Compute the sum. $s \leq i0 \text{ xor } i1 \text{ xor } ci$;

-- Compute the carry. $co \leq (i0 \text{ and } i1) \text{ or } (i0 \text{ and } ci) \text{ or } (i1 \text{ and } ci)$;

end rtl;

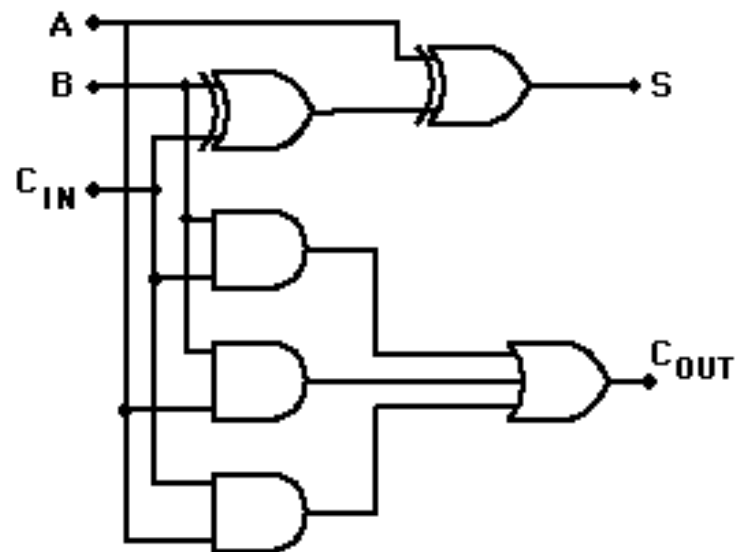
Behavioral Specification

```
module fulladder (a,b,cin,sum,cout);  
    input a,b,cin;  
    output sum,cout;  
  
    reg sum,cout;  
    always @ (a or b or cin)  
    begin  
        sum <= a ^ b ^ cin;  
        cout <= (a & b) | (a & cin) | (b & cin);  
    end  
endmodule
```

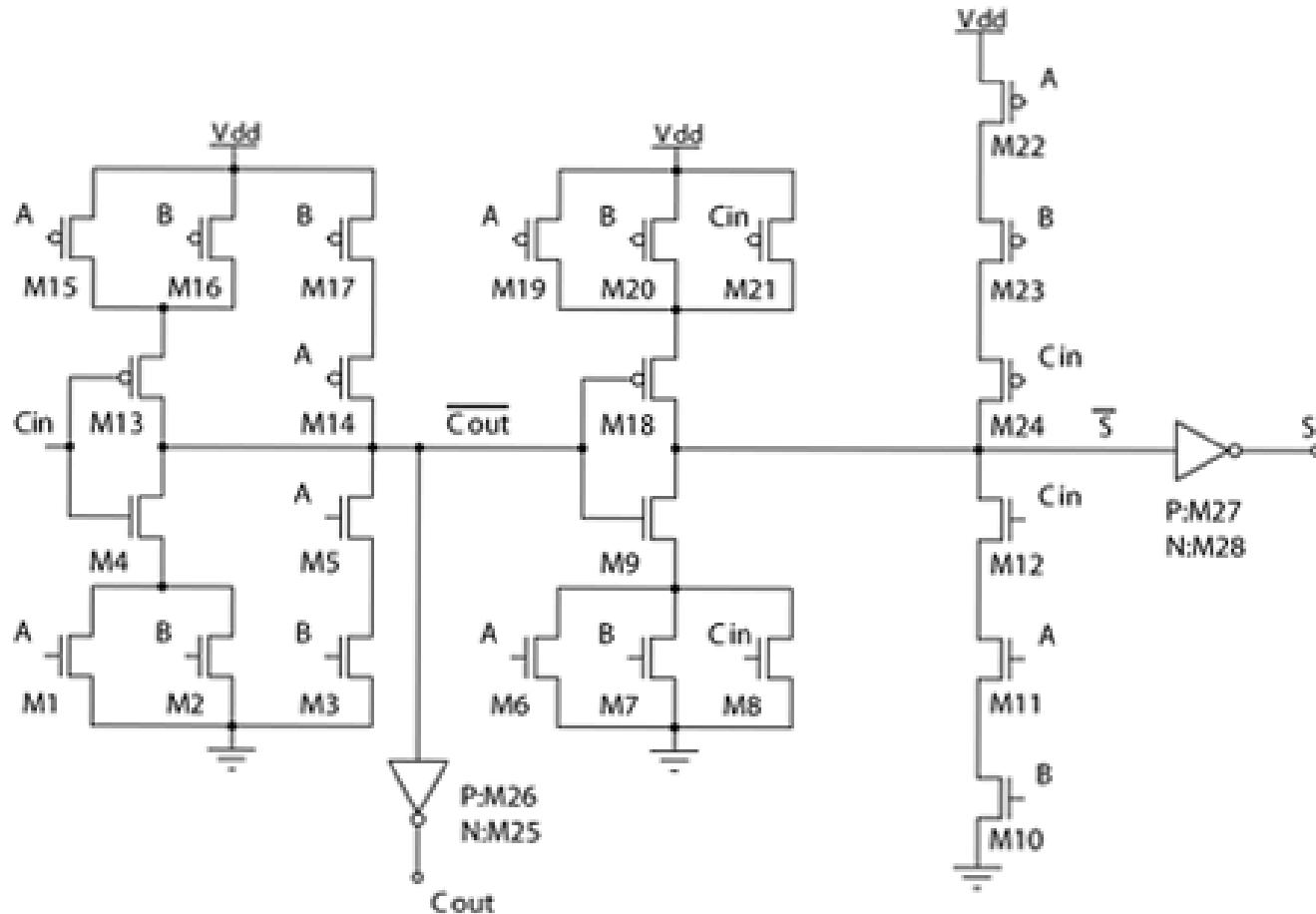

Logic Design

Full Adder Truth Table

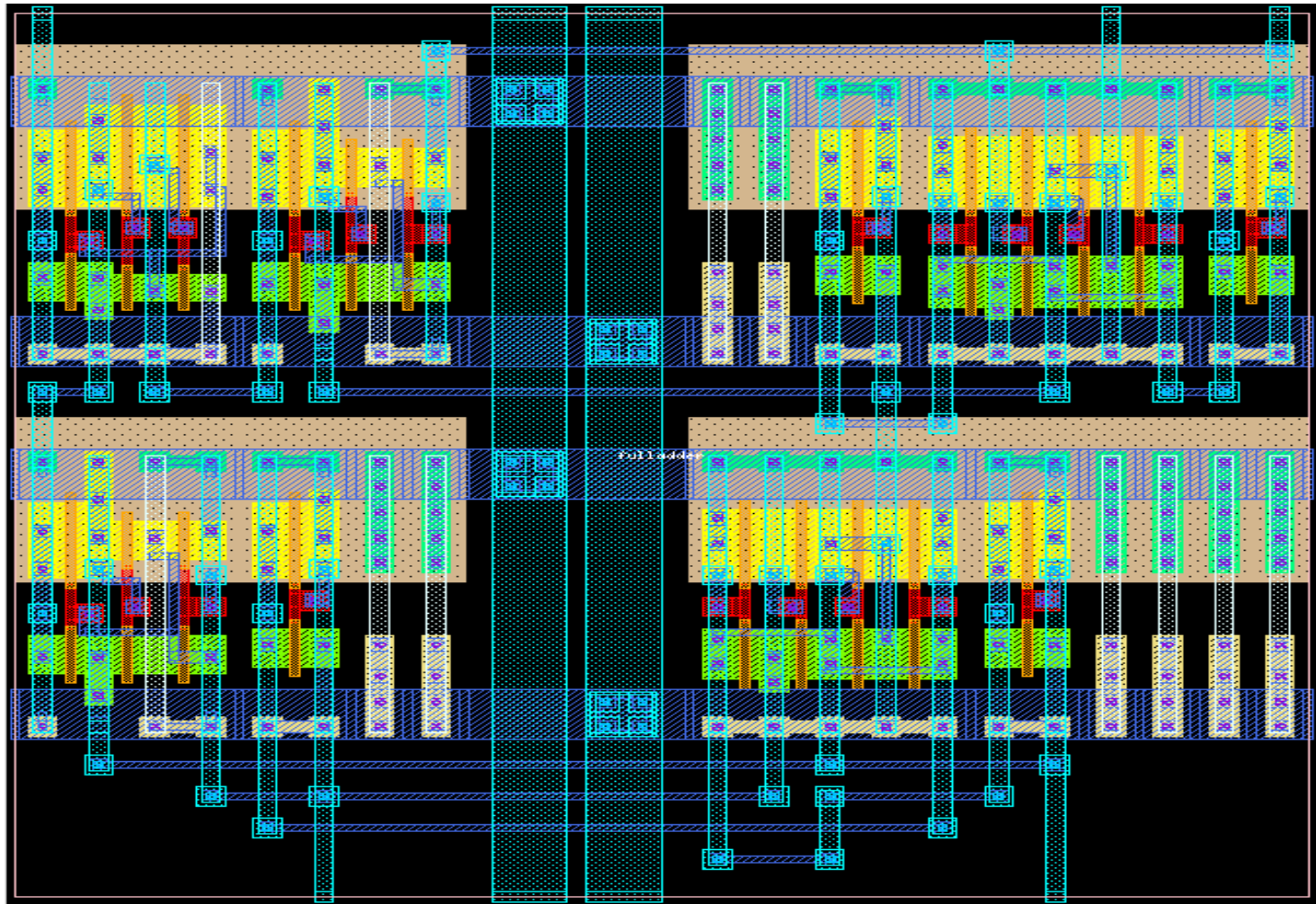
<i>CARRY IN</i>	<i>input B</i>	<i>input A</i>	<i>CARRY OUT</i>	<i>SUM digit</i>
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



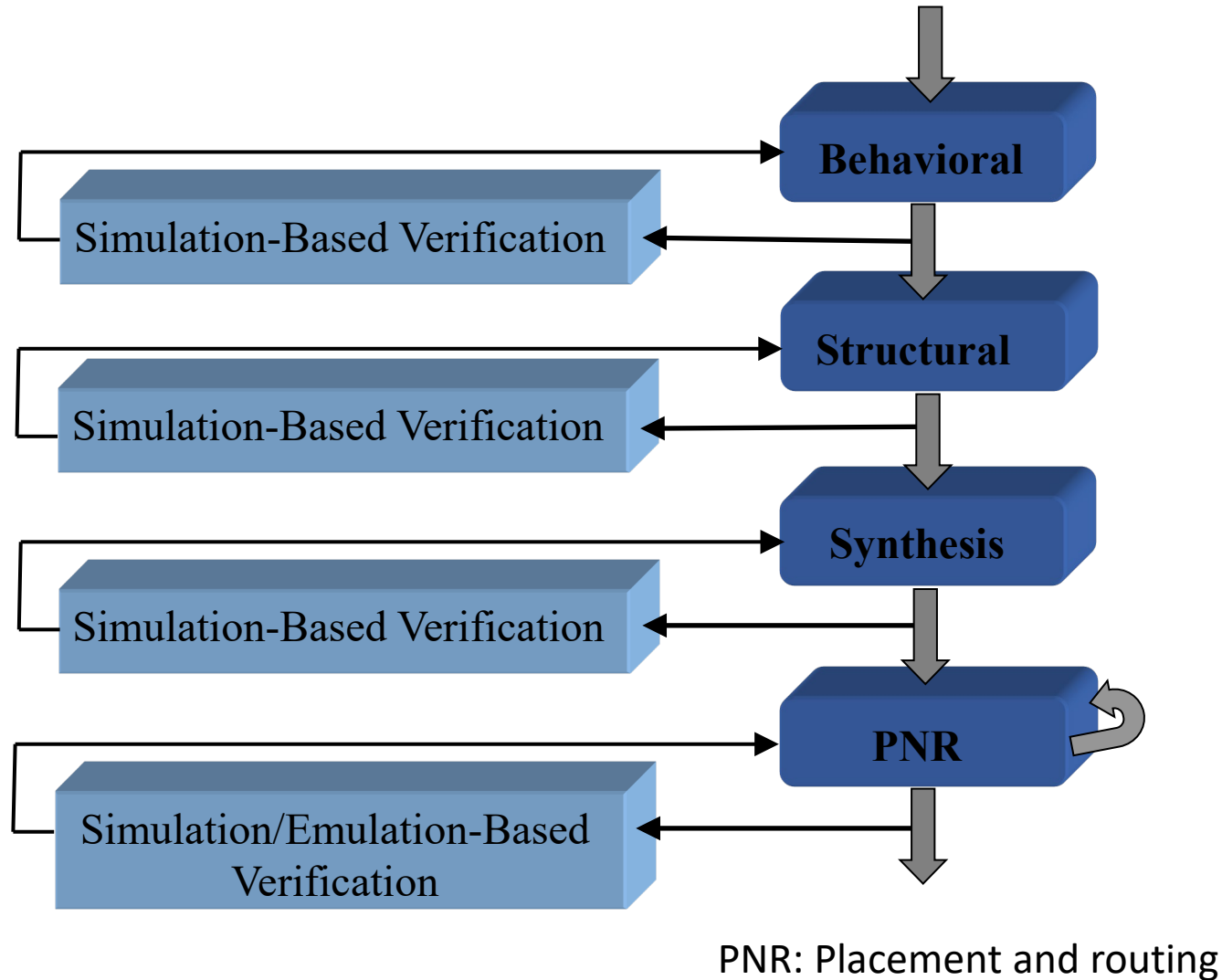
Transistor Schematic



Layout



Design Process is Iterative





VLSI Design Methodologies

- Full custom
 - Design for performance-critical cells
 - Very expensive
- Standard cell
 - Faster
 - Performance is not as good as full custom
- Gate array
- Field Programmable Gate Array

Comparison of Design Styles

	Full Custom	Standard Cell	Gate Array	FPGA
Area	Compact	Moderate	Moderate	Large
Performance	High	Moderate	Moderate	Low

Production Volume:

**Mass
Production
Volume**

**Medium
Production
Volume**

**Medium
Production
Volume**

**Low
Production
Volume**

Complexity:

High

Low

VLSI Chip Yield

- A manufacturing defect in fabrication process causes electrically malfunctioning circuitry.
- A chip with no manufacturing defect is called a good chip.
 - The defective ones are called bad chips.
- Percentage of good chips produced in a manufacturing process is called the *yield*.
- Yield is denoted by symbol Y .

- How to
$$Y = \frac{\text{\# of good die}}{\text{\# total manufactured die}}$$
 nes?
TEST ALL CHIPS

Why Does Test Matter ?

I

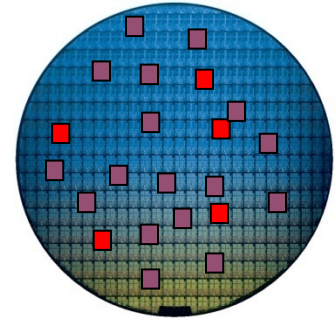
- In simple terms, TEST identifies the defective chips
- Some bad chips () are easy to find



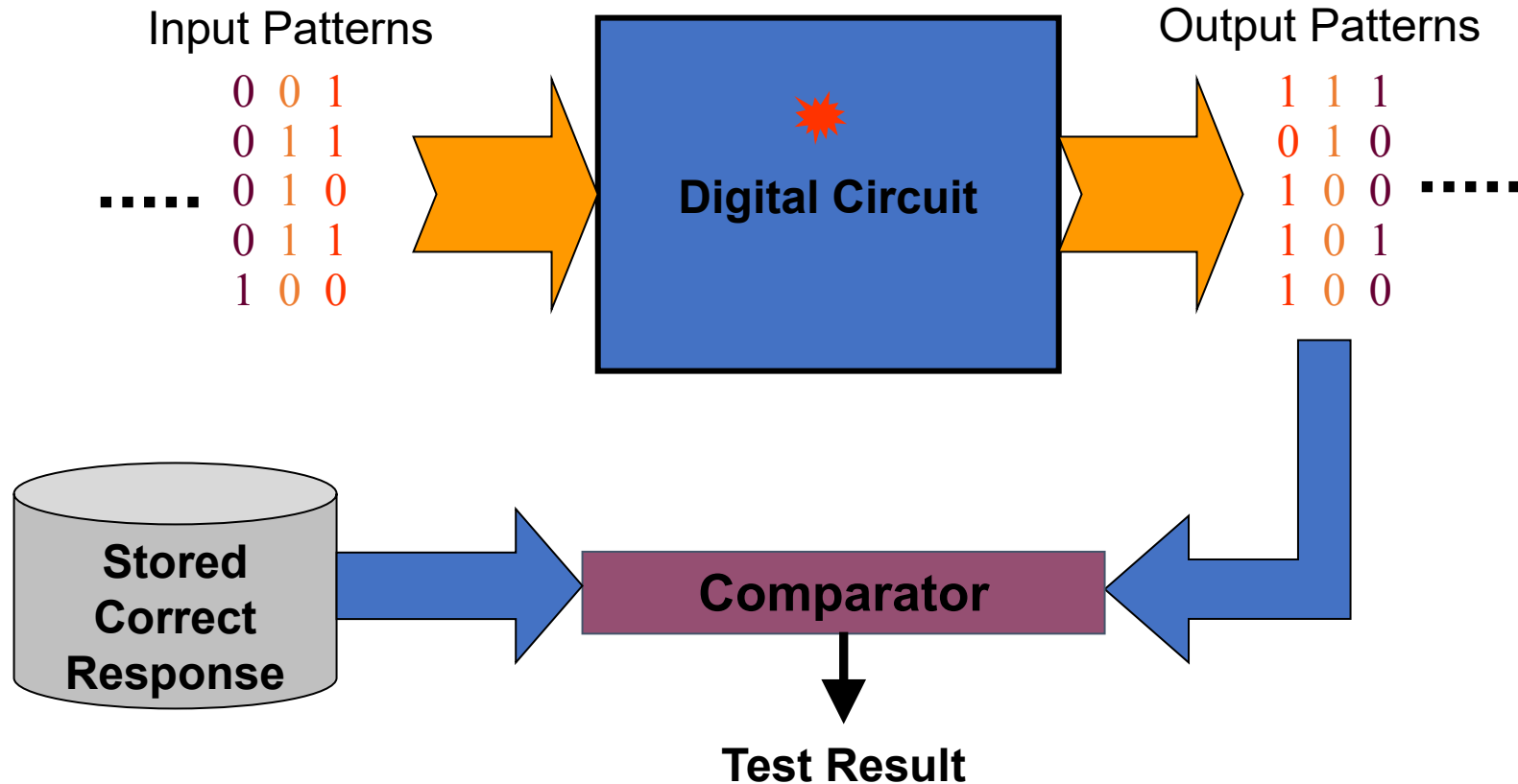
- Some others are difficult ()
- Test is associated with
 - Cost
 - Return On Investment (ROI)
 - ¥ € \$ - Money



Wafer



Testing Principle



Functional Test Method – Not very efficient

Ideal Tests

- Ideal tests detect **all** defects produced in the manufacturing process.
- Ideal tests pass all functionally good devices.
- Very large numbers and varieties of possible defects need to be tested.
- Difficult to generate tests for some real defects. *Defect-oriented testing is an open problem.*

Real Tests

- Based on analyzable fault models, which may not map on real defects.
- Incomplete coverage of modeled faults due to high complexity.
- Some good chips are rejected. The fraction (or percentage) of such chips is called the **yield loss**.
- Some bad chips pass tests. The fraction (or percentage) of bad chips among all passing chips is called the **defect level**.

Level of testing (1)

- Levels

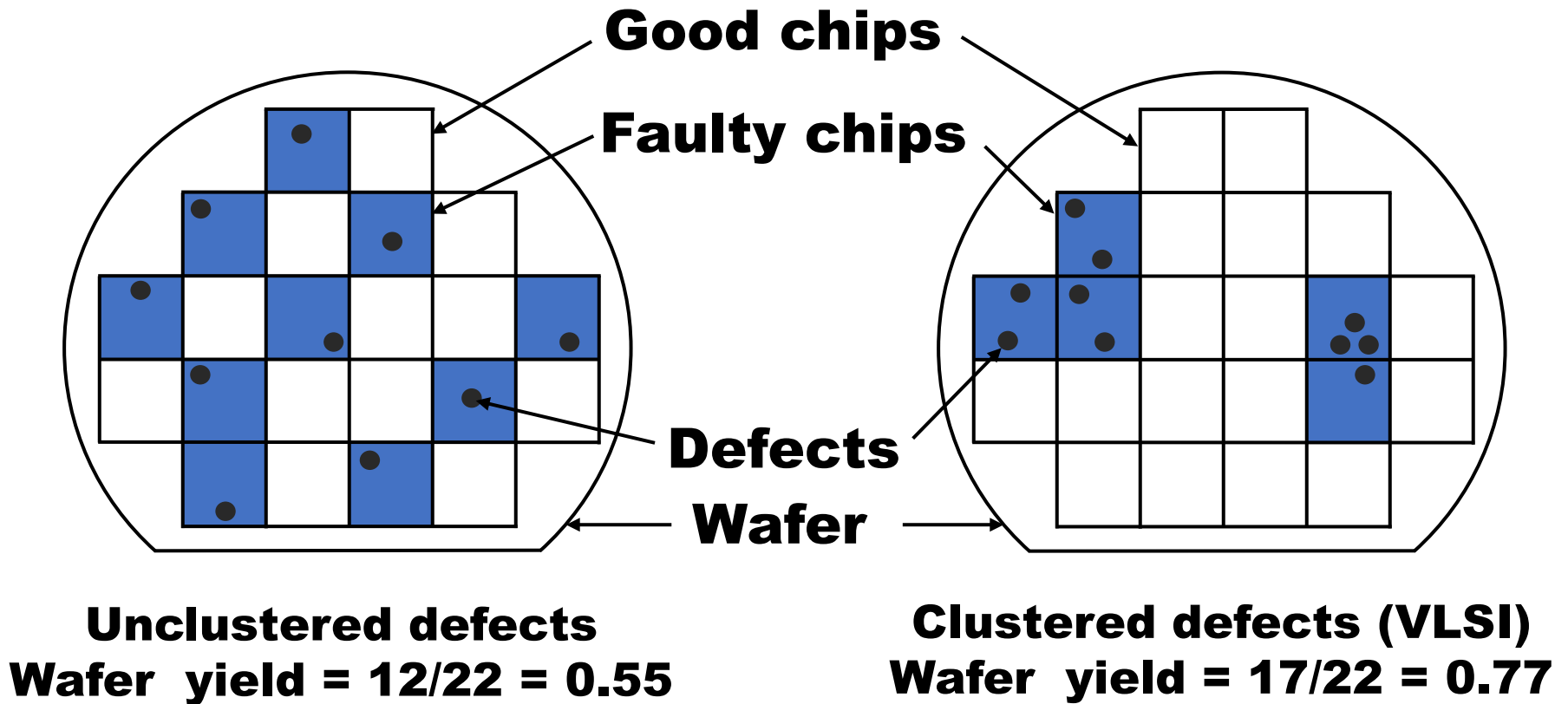
- Chip
- Board
- System
 - Boards put together
 - System-on-Chip (SoC)
- System in field

- Cost – Rule of 10

- It costs 10 times more to test a device as we move to higher level in the product manufacturing process



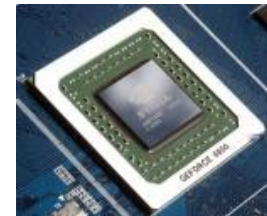
VLSI Defects



ADVANTEST Model T6682 ATE



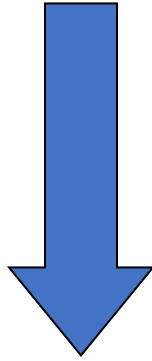
Test Head



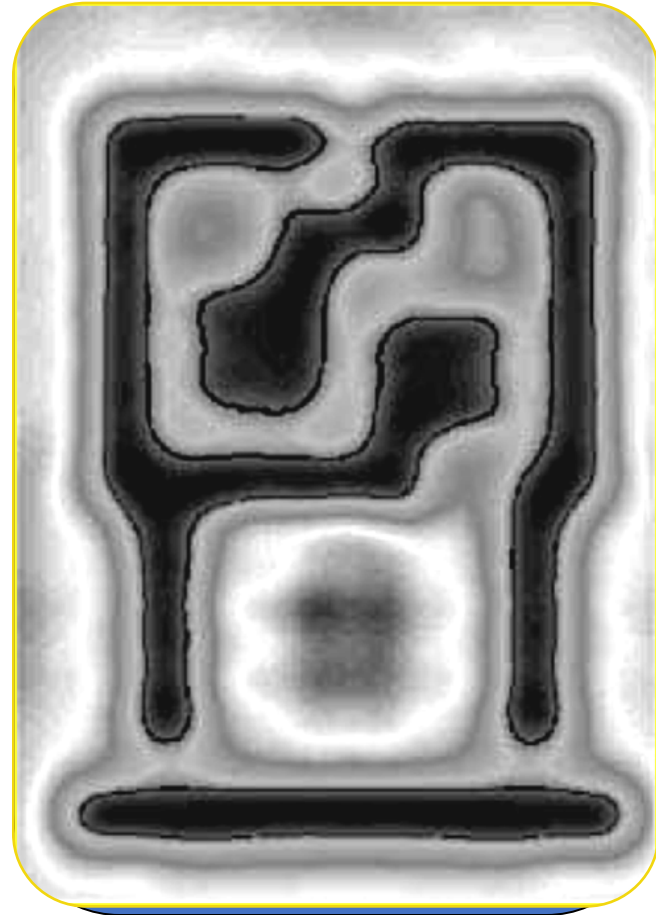
Testers are very expensive (\$150K – \$20M)

WYSINWYG

Sub-Wavelength WYSINWYG



What You See Is Not What You Get

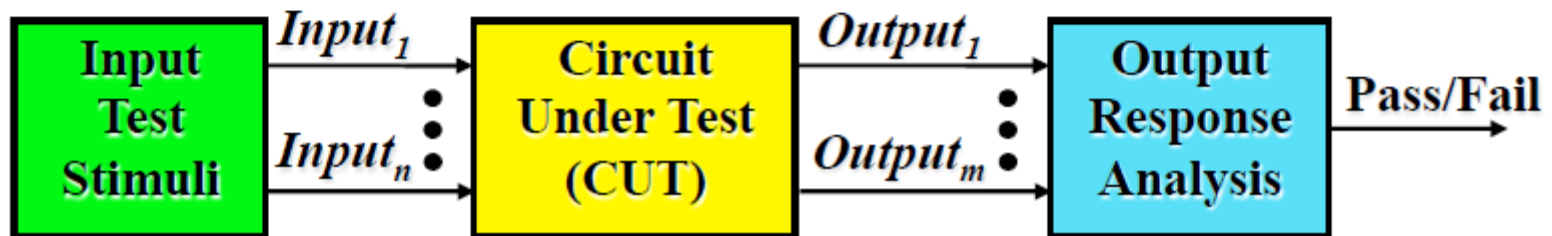


Process variations

No two transistors have the same parameters

Testing Theory

- Testing typically consists of
 - Applying set of test stimuli to
 - Inputs of *circuit under test* (CUT), and
 - Analyzing output responses
 - If incorrect (fail), CUT assumed to be faulty
 - If correct (pass), CUT assumed to be fault-free



Test Generation

- A test is a sequence of test patterns, called test vectors, applied to the CUT whose outputs are monitored and analyzed for the correct response
 - Exhaustive testing – applying all possible test patterns to CUT
 - Functional testing – testing every truth table entry for a combinational logic CUT
 - Neither of these are practical for large CUTs

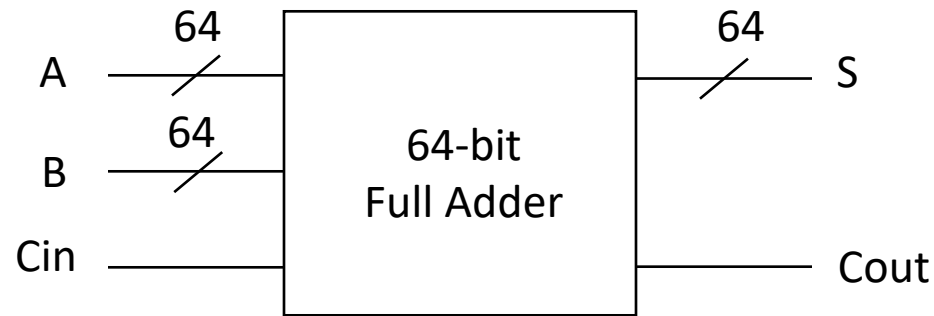
Test Generation



1-bit Binary Adder

$2^3 = 8$ input combinations,
verify $2^2 = 4$ output response.

Test Generation



64-bit Binary Adder

$2^{129} = 6.8 \times 10^{38}$ input combinations,

verify $2^{65} = 3.69 \times 10^{19}$ output response.

Test Vector applied at 1GHz rate, take 2.15×10^{22} years.

CMOS Testing

- ❖ Testing
 - ❖ Logic Verification
 - ❖ Silicon Debug
 - ❖ Manufacturing Test
- ❖ Fault Models
- ❖ Observability and Controllability
- ❖ Design for Test
 - ❖ Scan
 - ❖ BIST

Testing

- Testing is one of the most expensive parts of chips
 - Logic verification accounts for $> 50\%$ of design effort for many chips
 - Debug time after fabrication has enormous cost
 - Shipping defective parts can sink a company
- Example: Intel FDIIV bug
 - Logic error not caught until $> 1\text{M}$ units shipped
 - Recall cost \$450M (!!!)

Logic Verification

- Does the chip simulate correctly?
 - Usually done at HDL level
 - Verification engineers write test bench for HDL
 - Can't test all cases
 - Look for corner cases
 - Try to break logic design
- Ex: 32-bit adder
 - Test all combinations of corner cases as inputs:
 - 0, 1, 2, $2^{31}-1$, -1, -2^{31} , a few random numbers
- Good tests require ingenuity

Silicon Debug

- Test the first chips back from fabrication
 - If you are lucky, they work the first time
 - If not...
- Logic bugs vs. electrical failures
 - Most chip failures are logic bugs from inadequate simulation
 - Some are electrical failures
 - Crosstalk
 - Dynamic nodes: leakage, charge sharing
 - Ratio failures
 - A few are tool or methodology failures (e.g. DRC)
- Fix the bugs and fabricate a corrected chip

Testing Techniques

- How to diagnose failures?
 - Hard to access chips
 - Picoprobes
 - Electron beam
 - Laser voltage probing
 - Built-in self-test
- Shmoo plots
 - Vary voltage, frequency
 - Look for cause of electrical failures

Manufacturing Testing

Goal: Detect manufacturing defects

Defects: layer-to-layer shorts
discontinuous wires
thin-oxide shorts to substrate or well

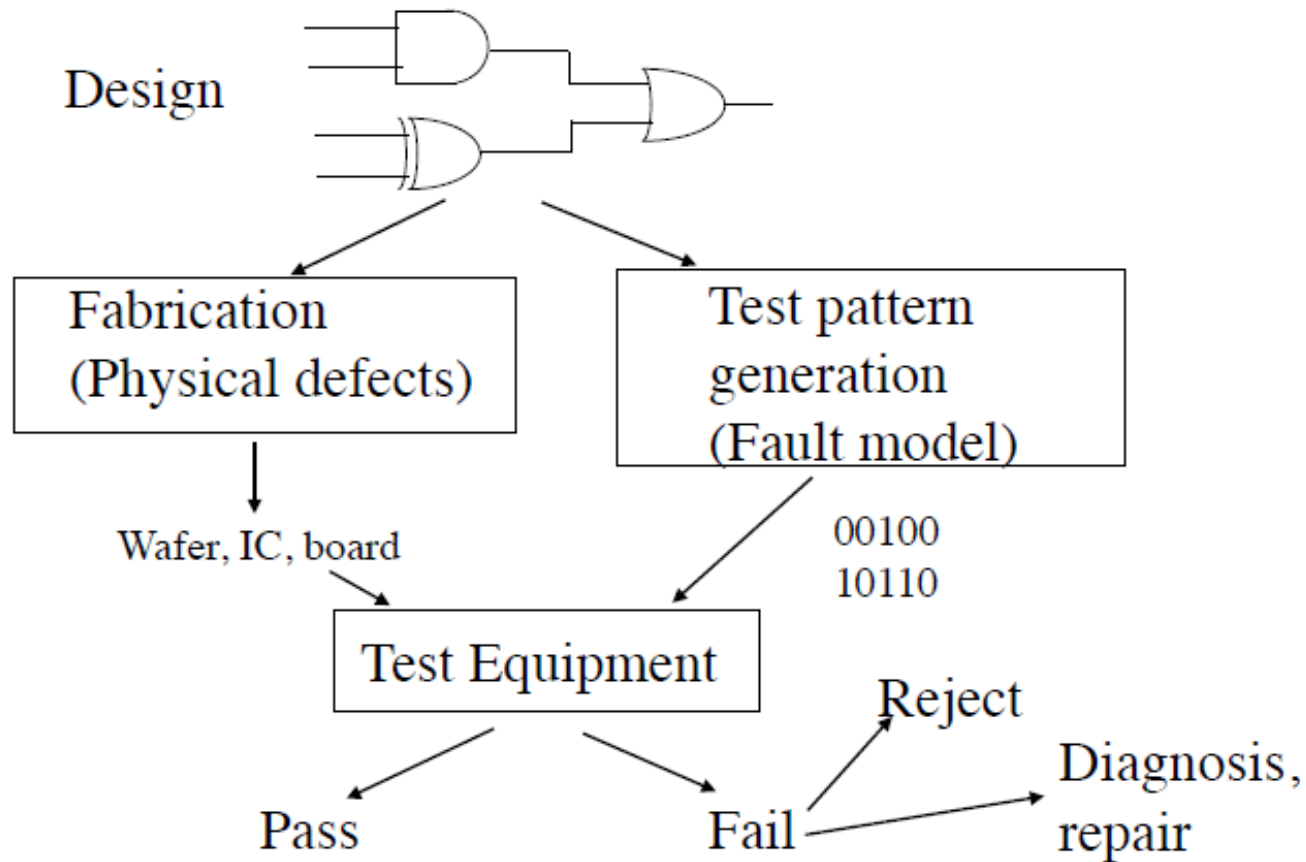


Faults: nodes shorted to power or ground (stuck-at)
nodes shorted to each other (bridging)
inputs floating, outputs disconnected (stuck-open)

Testing Terms

- Errors
 - Permanent
 - Intermittent
 - Transient
- Faults
 - Physical
 - Logical
- Test Evaluation
 - Fault coverage
 - Fault simulation
- Types of testing
 - Off-line, on-line
 - Self-test vs external test
 - DC (static) vs AC (at-speed)
 - Edge-pin, guided-probe, bed-of-nails, E-beam, in-circuit

Testing and Diagnosis



Testing Acronyms

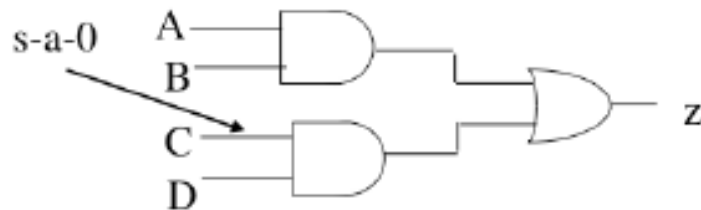
- System under test
 - UUT: Unit Under Test
 - CUT: Circuit Under Test
 - DUT: Device Under Test
 - The tester
 - ATE: Automatic Test Equipment
 - Test generation
 - ATPG: Automatic Test Pattern Generation
 - Fault Models
 - SSL: Single Stuck-Line
 - MSL: Multiple Stuck-Line
 - BF: Bridging Fault
 - DFT: Design for Testability
 - BIST: Built-in self-test
 - LFSR: Linear-Feedback Shift-Register
- Equivalent faults
- One or more single faults that have identical behavior for all possible input patterns
 - Only one fault from a set of equivalent faults needs to be simulated

Fault Model--Stuck At Faults

- How does a chip fail?
 - Usually failures are shorts between two conductors or opens in a conductor
 - This can cause very complicated behavior
- A simpler model: *Stuck-At*
 - Assume all failures cause nodes to be “stuck-at” 0 or 1, i.e. shorted to GND or V_{DD}
 - Not quite true, but works well in practice

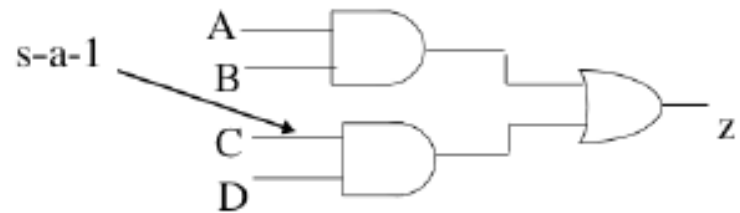
Single Stuck Line (SSL) Model

- A single node in the circuit is stuck-at 1 (s-a-1) or 0 (s-a-0).



Fault-free function $z = AB + CD$

Faulty function $z^f = AB$



Fault-free function $z = AB + CD$

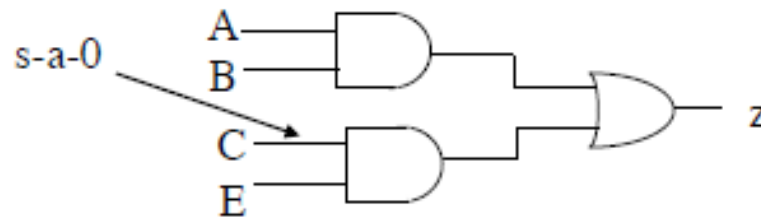
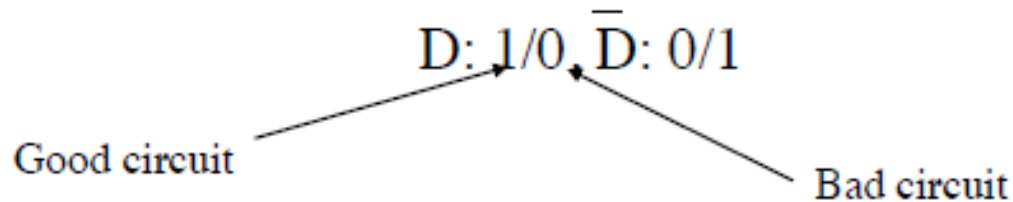
Faulty function $z^f = AB + D$

Number of possible stuck-at faults in a circuit with n lines?

Number of faults reduced by finding equivalent classes

SSL Fault Detection

- A test pattern for fault x s-a- \bar{d} is an input combination that
1) places \bar{d} on x (activation), 2) propagates fault effect (D or \bar{D}) to primary output

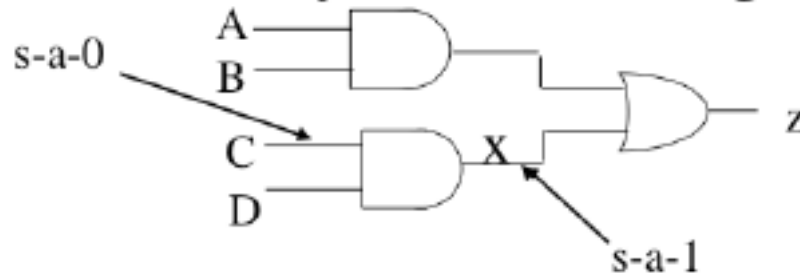


$ABCE = 0011$ is a test pattern for C s-a-0

Test pattern: Find input vector to have Z (XOR) $Z_f = 1$

Multiple Stuck Line (MSL) Faults

- More than one line may be stuck at a logic value



Fault: {C s-a-0, x s-a-1}

How many MSL fault can there be in a circuit with n nodes?

How to get test patterns for MSL faults?

Fault universe is too large, MSL fault model seldom used, especially since tests for SSL faults cover many MSL faults

Controllability and Observability



- Controllability and observability are two important attributes related to testability.
- Controllability of a signal is the ability to control the value of a signal to '0' and '1'. If both '0' and '1' can be propagated with an output, it is said to be controllable.
- Observability on the other hand is the ability to observe the signal state. If it is possible to observe the changes at input is said to be observable.

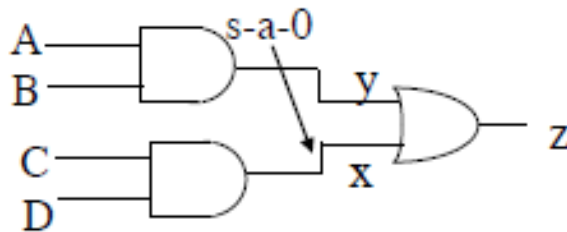
Test Pattern Generation

- Manufacturing test ideally would check every node in the circuit to prove it is not stuck.
- Apply the smallest sequence of test vectors necessary to prove each node is not stuck.
- Good observability and controllability reduces number of test vectors required for manufacturing test.
 - Reduces the cost of testing
 - Motivates design-for-test

Test Pattern Generation

- Exhaustive testing: Apply 2^n pattern to n -input circuit
- Not practical for large n
- Advantage: Fault-model independent

Fault-Oriented Test Generation Algorithm:



- 1) Set x to 1: activate fault
- 2) Justify D on x , propagate D

to z

Set C and D to 1

Set y to 0

Set either A or B to 0

Example test pattern: $ABCD = 0011$

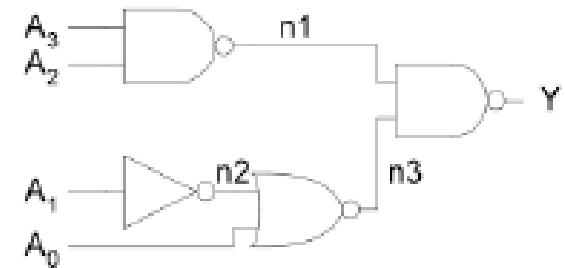
- Backtracking may be necessary
- Test generation is NP-complete

Test pattern: Find input vector to have Z (XOR) $Z_f=1$

NP-complete: nondeterministic polynomial-time complete

Test Example

	SA1	SA0
• A_3	{0110}	{1110}
• A_2	{1010}	{1110}
• A_1	{0100}	{0110}
• A_0	{0110}	{0111}
• n1	{1110}	{0110}
• n2	{0110}	{0100}
• n3	{0101}	{0110}
• Y	{0110}	{1110}

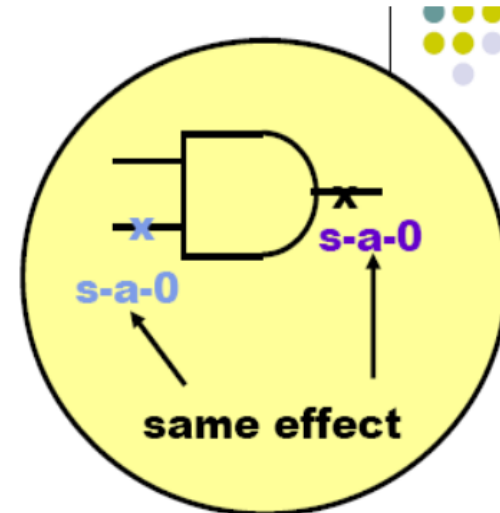


- Minimum set: {0100, 0101, 0110, 0111, 1010, 1110}

Down from original $2n$ stuck-at faults. N is number of nodes

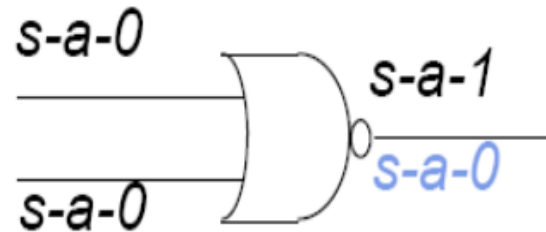
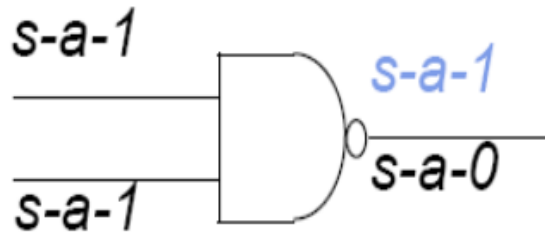
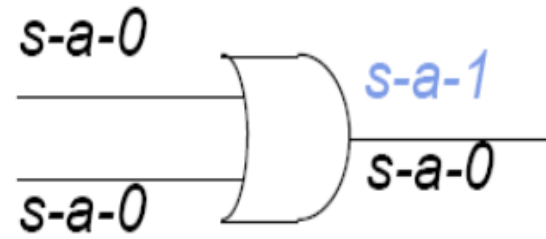
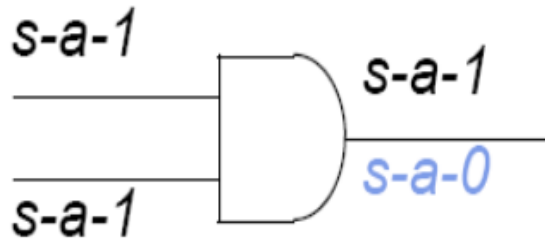
Fault Equivalence

- AND gate:
 - all s-a-0 faults are equivalent
- OR gate:
 - all s-a-1 faults are equivalent
- NAND gate:
 - all the input s-a-0 faults and the output s-a-1 faults are equivalent
- NOR gate:
 - all input s-a-1 faults and the output s-a-0 faults are equivalent
- Inverter:
 - input s-a-1 and output s-a-0 are equivalent
 - input s-a-0 and output s-a-1 are equivalent

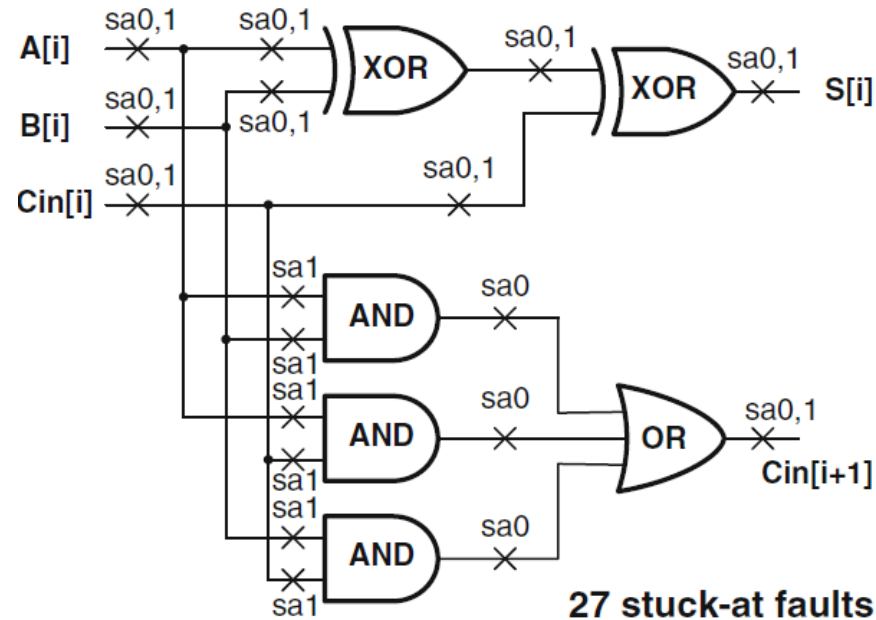


Equivalence Fault Collapsing

- $n+2$ instead of $2(n+1)$ faults need to be considered for n -input gates (n+1) nodes



Test Generation—Stuck-At-Fault (1)



27 stuck-at faults for 1-bit adder after removing the equivalent faults

$27 \times 64 = 1728$ possible fault for 64-bit binary adder

1728 **T_s** @ 1GHz = 1.72us.

Test Generation—Stuck-At-Fault (2)

- High levels have few implementation details needed for effective test generation
 - Fault models based on gate & physical levels
- Example: two circuits for same specification
 - Ckt B test vectors do not detect 4 faults in Ckt A

$$f(a,b,c) = \sum_m(1,7) + d(3) = \bar{a}\bar{b}c + abc + Xabc$$

Circuit A

	<i>ab</i>	00	01	11	10
<i>c</i>	0		1	X	
	1			1	

$$f = abc + \bar{a}\bar{b}c$$

Test Vectors

{111,110,101,011,010,000}

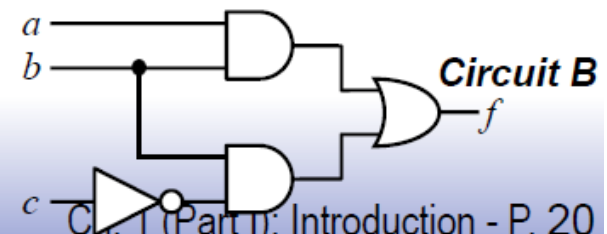
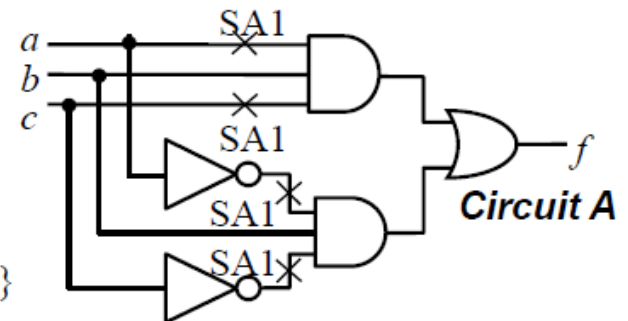
Circuit B

	<i>ab</i>	00	01	11	10
<i>c</i>	0		1	X	
	1			1	

$$f = ab + b\bar{c}$$

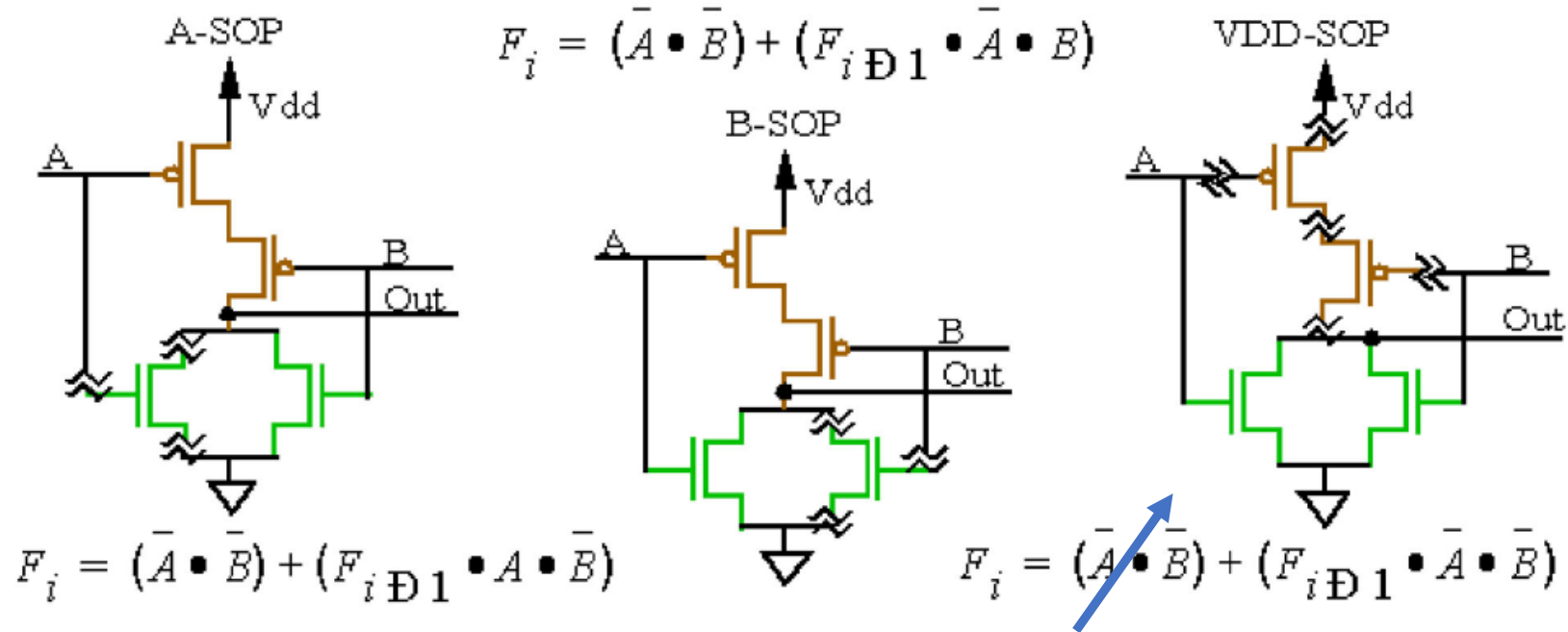
Test Vectors

{111,101,010,000}



Stuck-open Faults (SOP)

- Defect creates an unintended high impedance states on the node



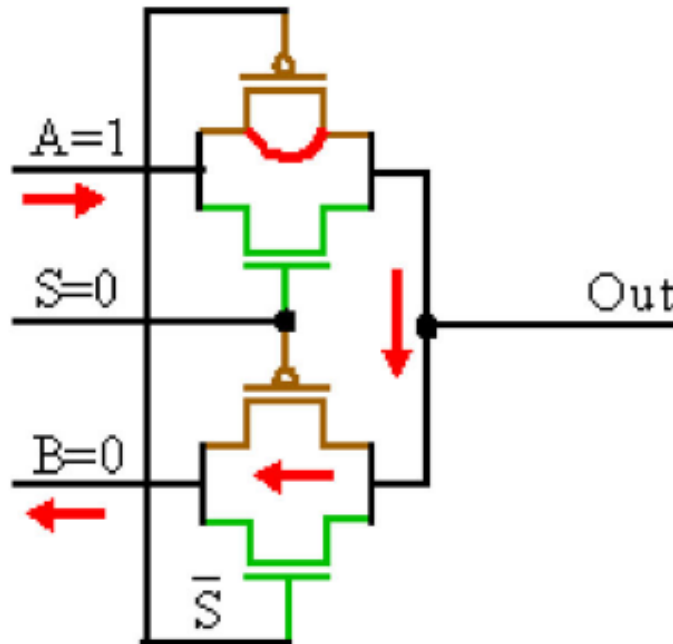
- Exhaustive test $AB = (00, 01, 10, 11)$ does not detect all faults.
- For SOP, vector order is important.
 - For example, the sequence $AB = (00, 01, 00, 10)$ is able to detect all faults on the NOR gate, including the SOP faults.

Stuck-short (Stuck-on) Faults

- Complement of the Stuck-open fault model

Stuck-On faulty gate output is difficult to predict.

- A transistor that is permanently stuck-on will, for some input combination(s), compete with its complementary transistors for control of the output.

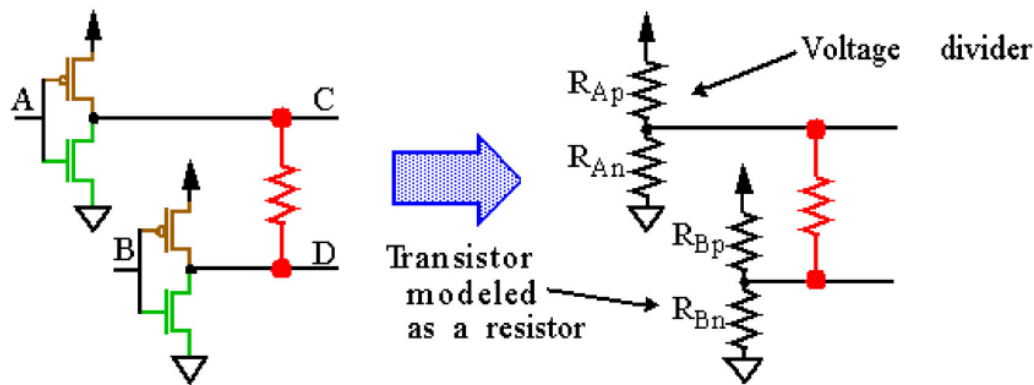


Bridging Faults

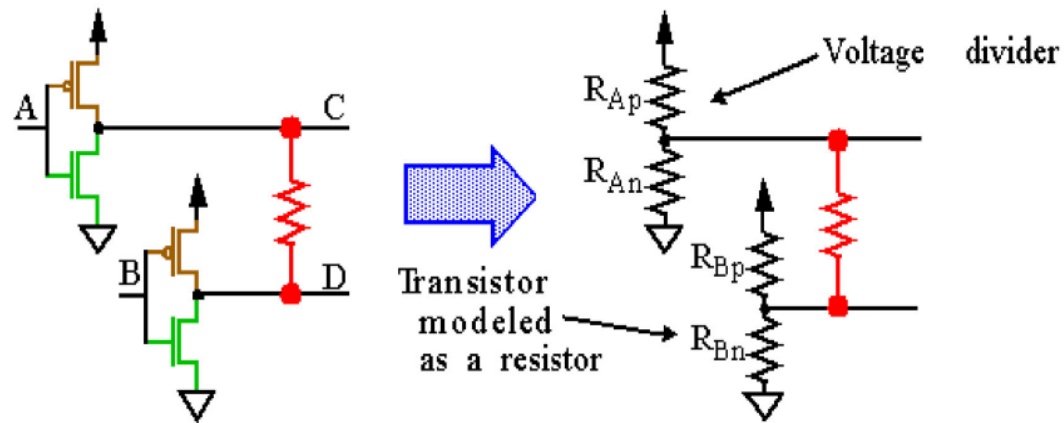
- Modeled at the gate or transistor level as a short between 2 (simple) or more of signal lines.
- *Non-feedback versus feedback (memory) versions.*



- Fault is usually modeled using **wired logic** : AND and OR.
- For CMOS, it **depends** on the type of gates driving the shorted lines and their input values



Bridging Faults



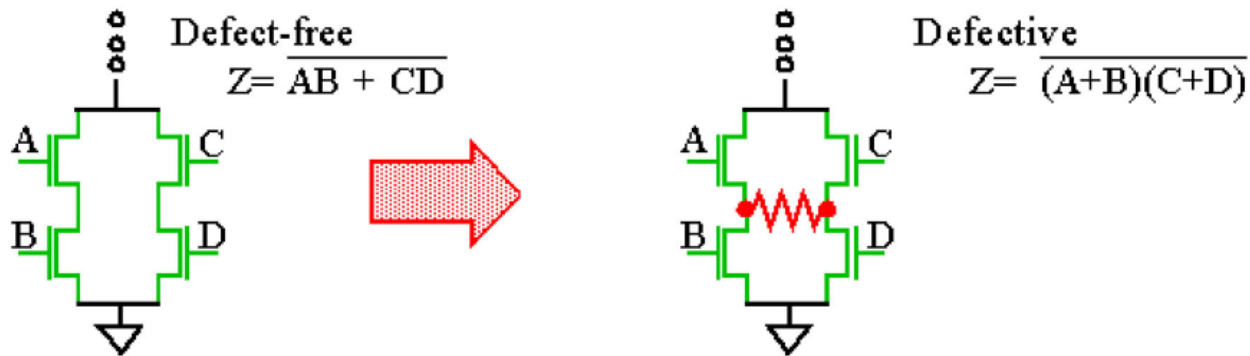
Bridging Faults

- *The transistor **resistances** determine the appropriate model:*

Input values	Resistance relationships	Resulting output value	Wired logic model.
A=B	Any ratio	$C = D$	AND, OR
A=0, B=1	$R_{Ap} > R_{Bn}$	$C = D = 0$	AND
	$R_{Ap} < R_{Bn}$	$C = D = 1$	OR

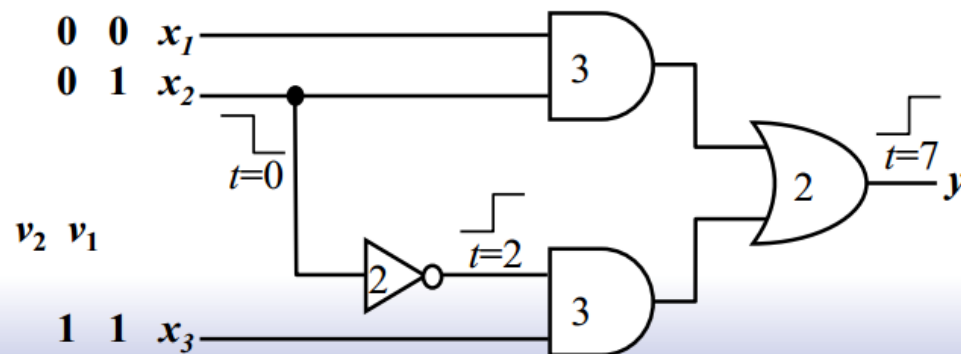
Bridging Faults

- *Bridging faults that can not be represented by a **known** fault model.*



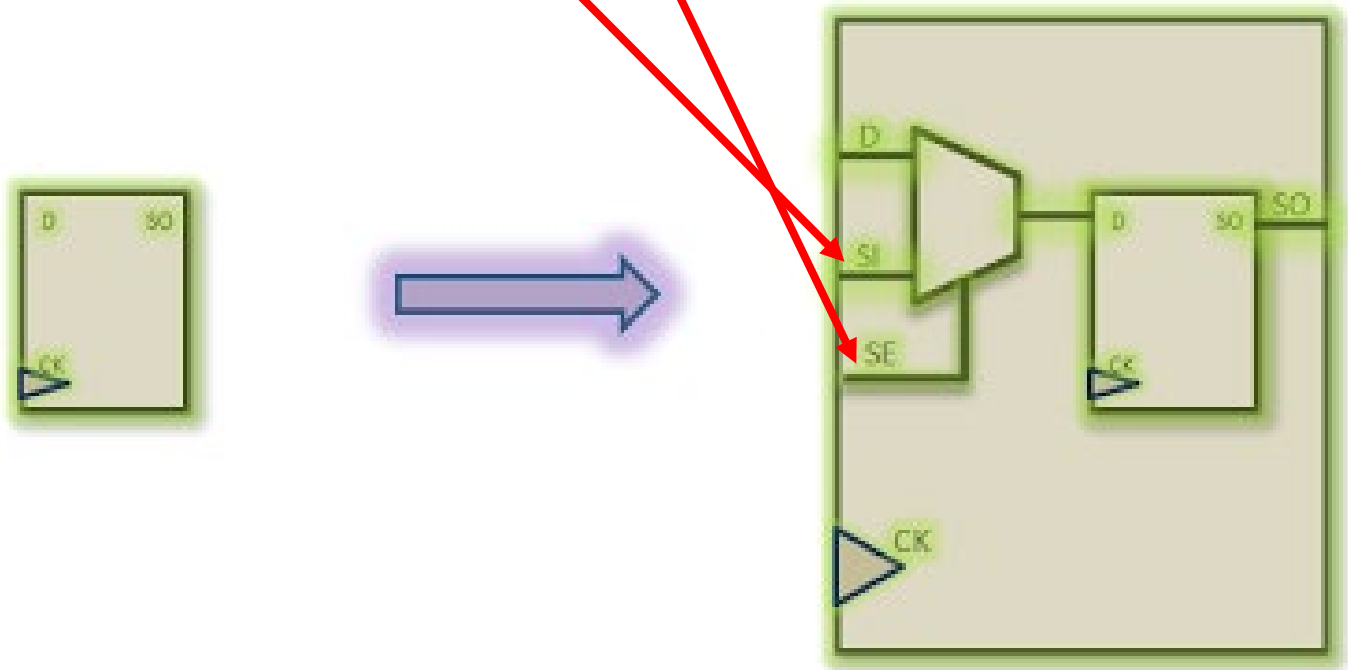
Delay Faults and Crosstalk

- Path-delay fault model considers cumulative propagation delay through CUT
 - 2 test vectors create transition along path
 - Faulty circuit has excessive delay
- Delays and glitches can be caused by crosstalk between interconnect
 - due to inductance and capacitive coupling



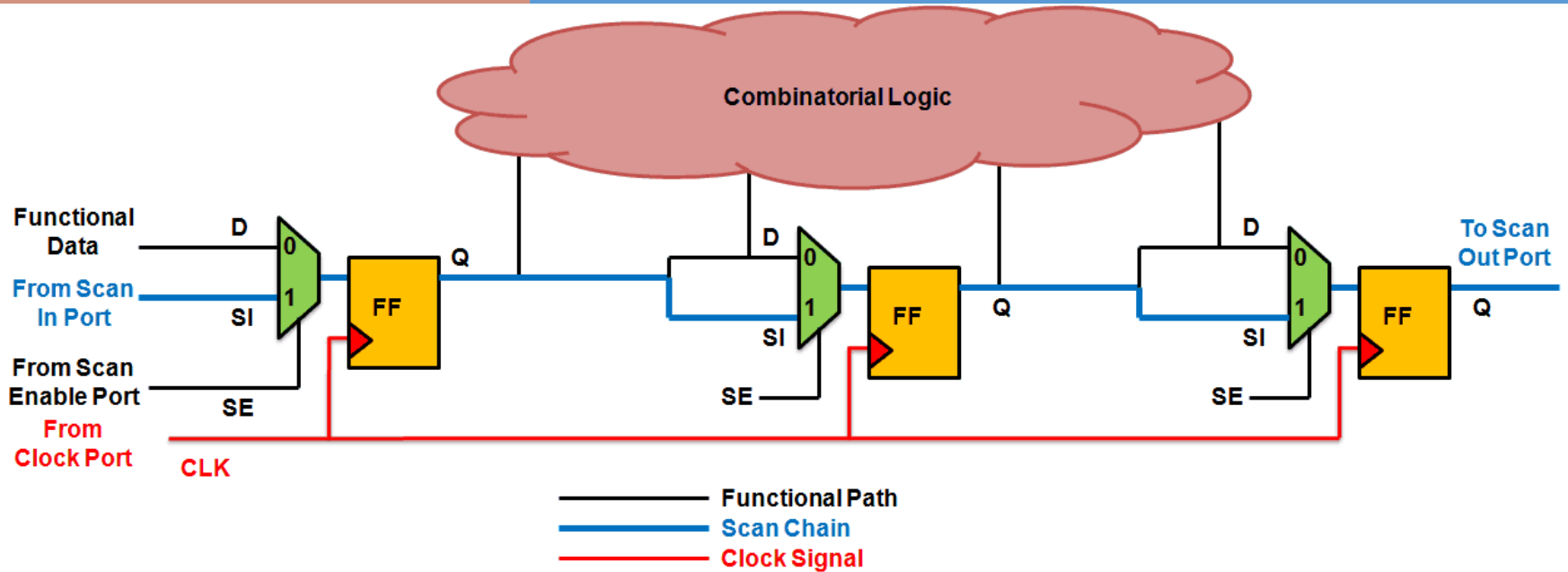
Built-in-test--Scan Flip-Flop

- ❖ Essentially a scan flip-flop is a d flip-flop with an additional input “Scan In” added that is able to be selected with a “Scan Enable” signal.



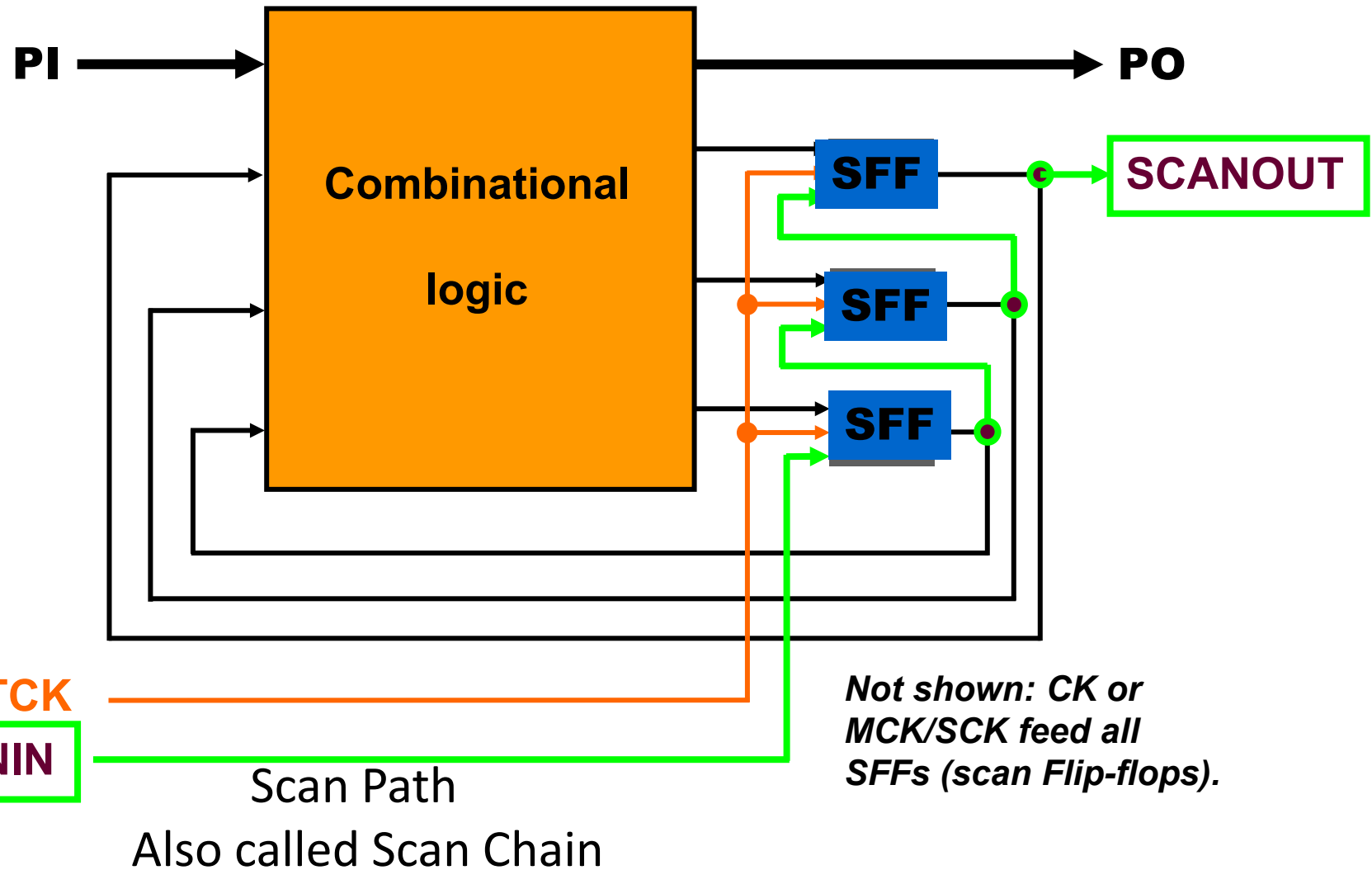
PURPOSE OF SCAN FLIP-FLOP AND SCAN CHAINS

- A scan flip-flop is an element that is implemented in chains.
- The chains of scan flip-flops are called “scan chains”.
- Scan chains are used to allow for a data to be sent -> shifted -> detected in next scan flip-flop



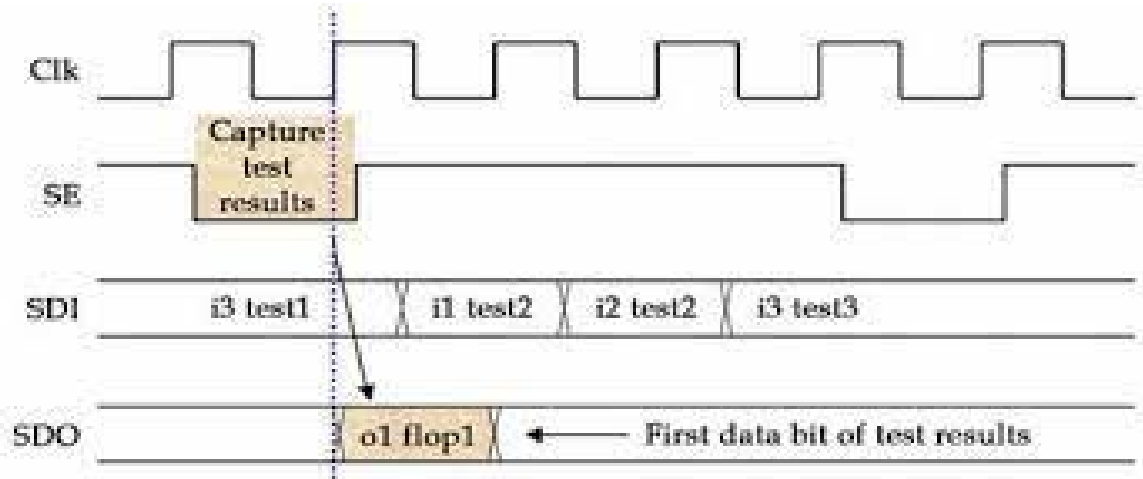
Adding Scan Structure to Original Logic

I



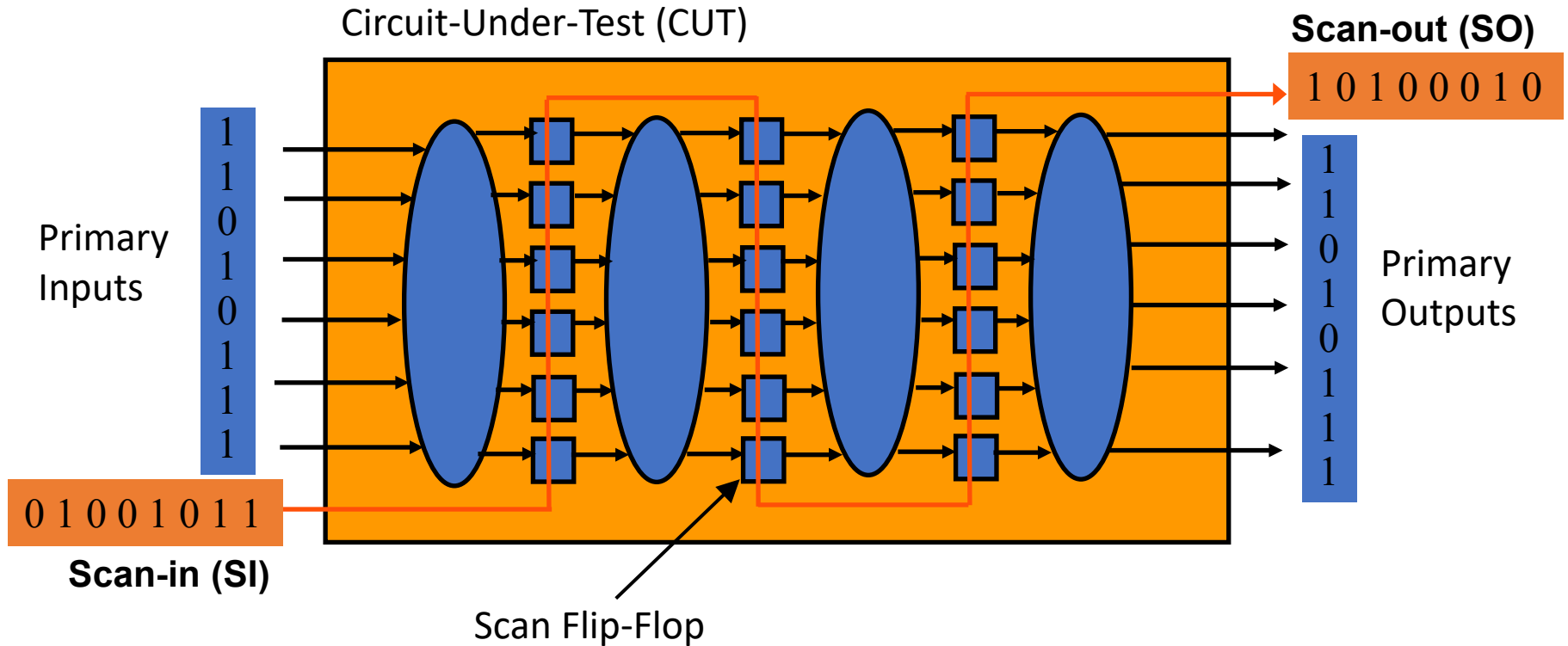
Scan Chain Operation

□ 3 stages operations



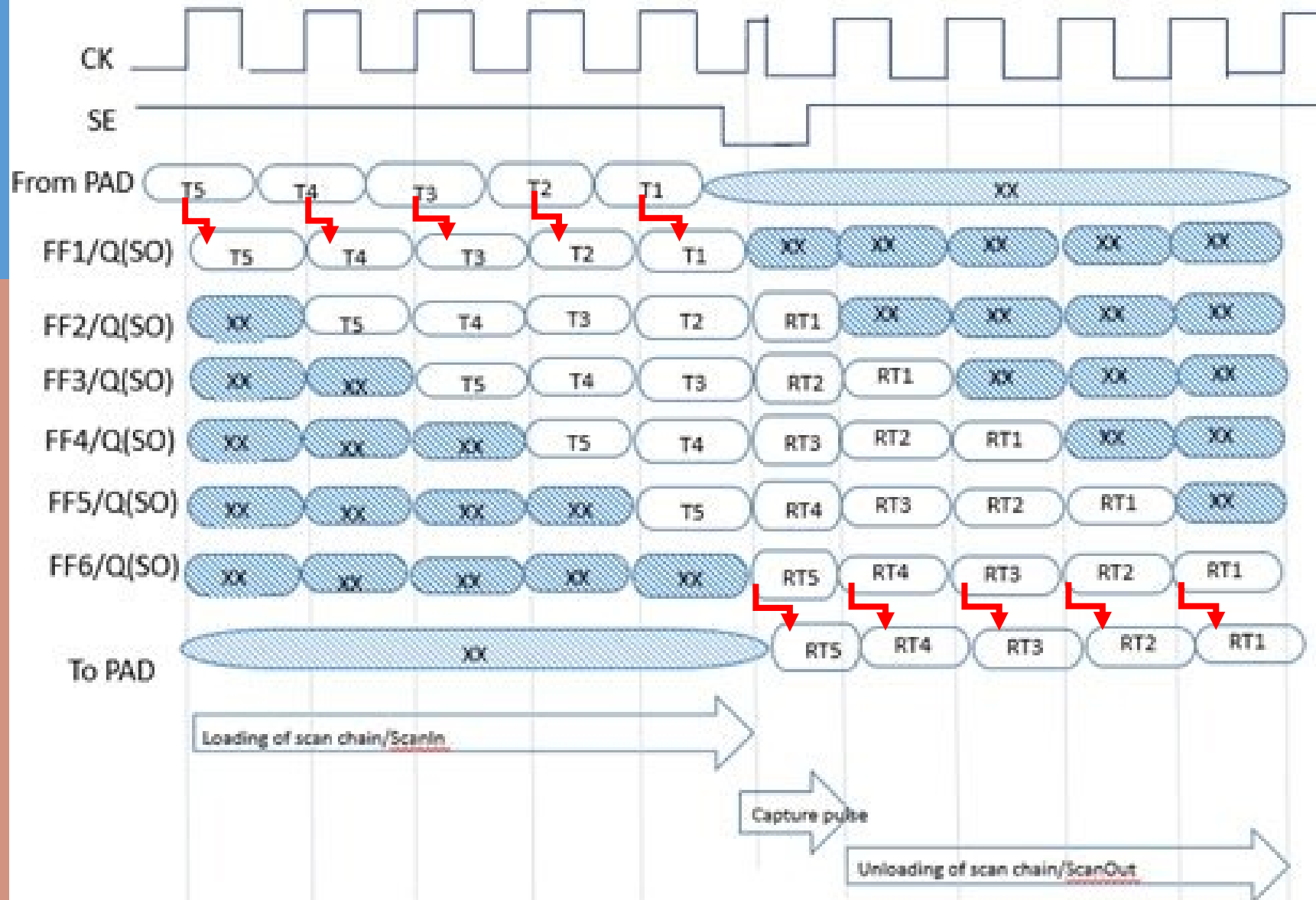
- **Scan In**--Test signals are sent into the first Scan In input of the chain. Each clock cycle the test signals shift from the input of one scan flip-flop to the input of the next. This can be seen as “loading” the test vectors. This is done while Scan Enable is set to high.
- **Capture**--In this stage the Scan Enable is set to low. This allows for all the data that was “loaded” into the scan chain to be sent into the combinational logic portion of the design and the functions of the circuit are preformed on the test vectors. This stage lasts for only one cycle.
- **Scan Out**--In this stage the Scan Enable signal is set back to high after the “processed” test signals have been captured in the Capture stage. The test signals are then “unloaded” in the same manner that they were “loaded” in the Scan In stage. These signals are then recorded as they flow through the final Scan Out pin in the design and compared with the expected results.

Scan Design



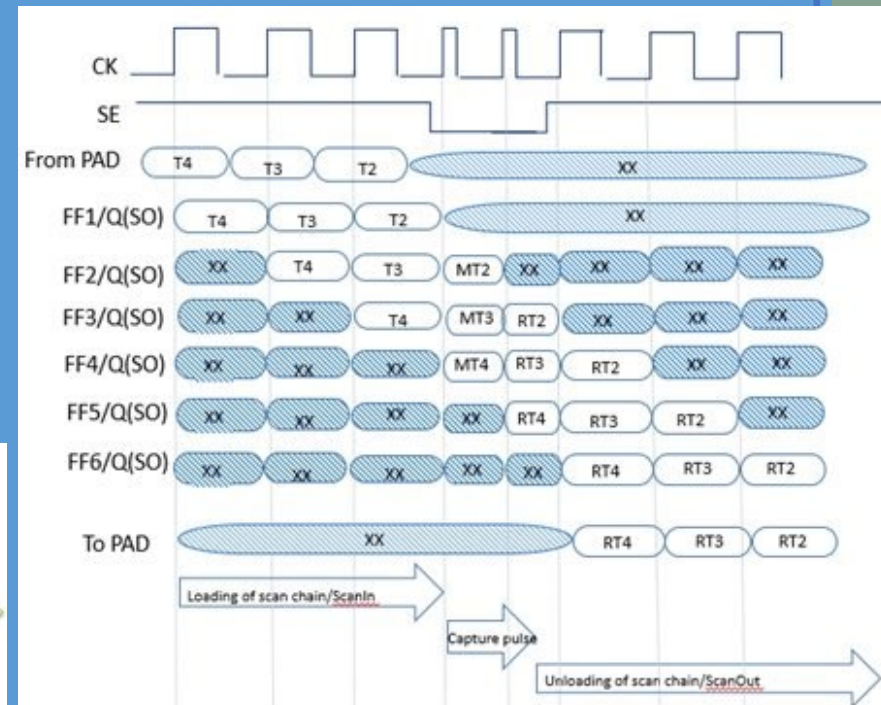
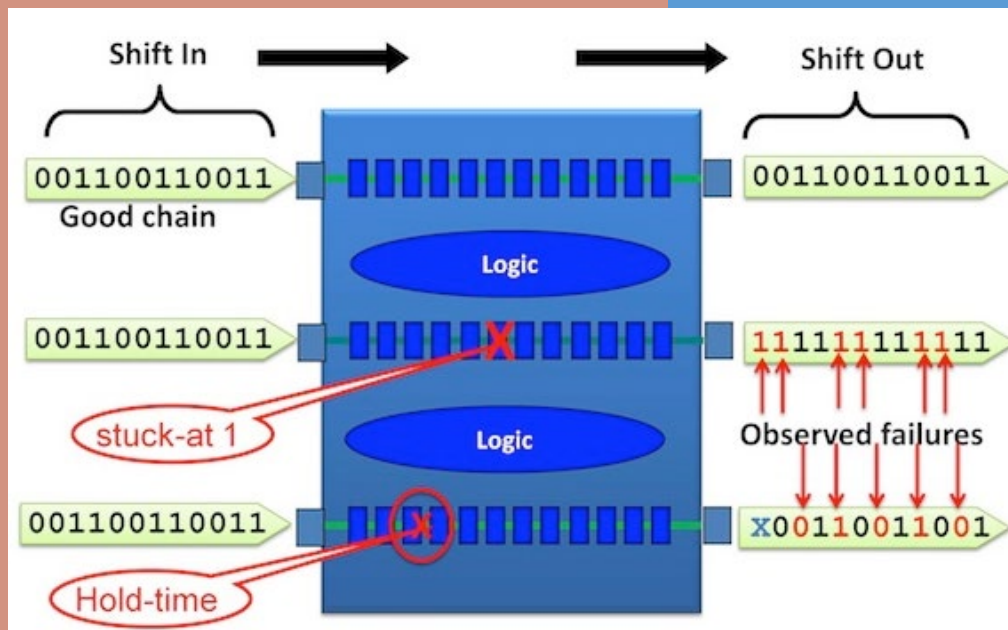
Structural Test Method – Extremely efficient

OPERATION VISUALIZED



OPERATION VISUALIZED

Errors in operation represent faults in the physical implementation of the design



OPERATION VISUALIZED

fault types for input

"0 0 1 1 0 0 1 1 0 0 1 1"

Shift input

001100110011

Shift out

Fault type

001000100010

Slow rise

011101110111

Slow fall

011001100110

Slow

101110111011

Fast rise

000100010001

Fast fall

100110011001

Fast (hold-time)

00000000000

Stuck-at-0

11111111111

Stuck-at-1

Overview of VLSI Test Technology

□ Automatic Test Equipment (ATE) consists of

- Computer – for central control and flexible test & measurement for different products
- Pin electronics & fixtures – to apply test patterns to pins & sample responses
- Test program – controls timing of test patterns & compares response to known good responses

Overview of VLSI Test Technology

□ Automatic Test Pattern Generation (ATPG)

- Algorithms generating sequence of test vectors for a given circuit based on specific fault models

□ Fault simulation

- Emulates fault models in CUT and applies test vectors to determine fault coverage
- Simulation time (significant due to large number of faults to emulate) can be reduced by
 - Parallel, deductive, and concurrent fault simulation

Overview of VLSI Test Technology

□ Design for Testability (DFT)

- Generally incorporated in design
- Goal: improve controllability and/or observability of internal nodes of a chip or PCB

□ Three basic approaches

- Ad-hoc techniques
 - Scan design
 - Boundary Scan
 - Built-In Self-Test (BIST)
-