Fundamentals of Computer Design

Technology, Cost, Performance, Power

Readings: H&P Chapter 1

This Unit

- What is a computer and what is computer architecture
- Forces that shape computer architecture
  - Applications (already covered)
  - Semiconductor technology
- Evaluation metrics: parameters and technology basis
  - Cost
  - Performance
  - Power
  - Reliability

What is Computer Architecture? (review)

- Design of interfaces and implementations…
- Under constantly changing set of external forces…
  - **Applications**: change from above (discussed last time)
  - **Technology**: changes transistor characteristics from below
  - **Inertia**: resists changing all levels of system at once

- To satisfy different constraints
  - This course mostly about **performance**
  - Cost
  - Power
  - Reliability

- Iterative process driven by **empirical evaluation**
- The art/science of tradeoffs

Abstraction and Layering

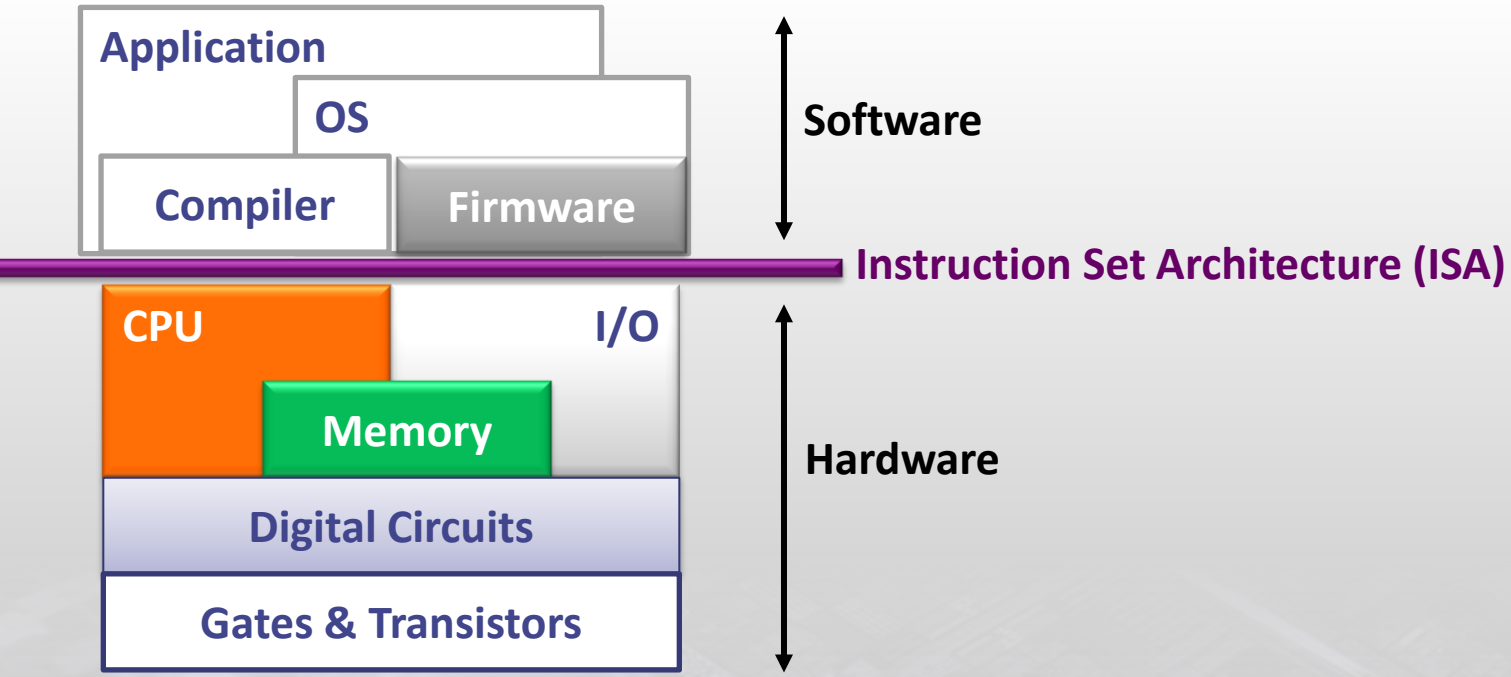
- **Abstraction**: only way of dealing with complex systems
  - Divide world into objects, each with an…
    - **Interface**: knobs, behaviors, knobs → behaviors
    - **Implementation**: "black box"
  - Specialists deal with implementation; others interface
  - Example: car drivers vs. mechanics

- **Layering**: abstraction discipline makes life even simpler
  - Removes need to even know interfaces of most objects
  - Divide objects in system into layers
  - Layer X objects
    - Implemented in terms of interfaces of layer X-1 objects
    - Don't even need to know interfaces of layer X-2 objects
  - Example: cab driver vs. mechanics

Abstraction, Layering, and Computers

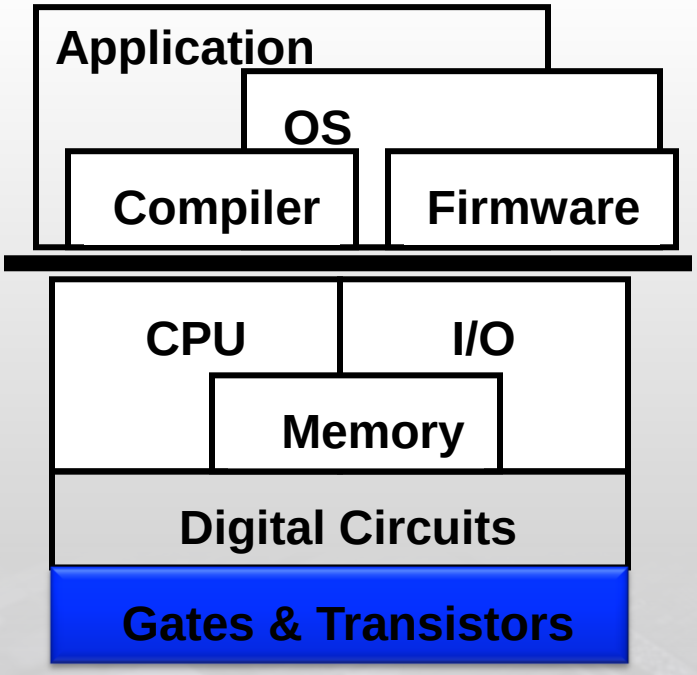
- Computers are complex systems, built in layers
  - Applications
  - O/S, compiler
  - Firmware, device drivers
  - Processor, memory, raw I/O devices
  - Digital circuits, digital/analog converters
  - Gates
  - Transistors
- 99% of users don't know hardware layers implementation
- 90% of users don't know implementation of any layer
- That's OK, world still works just fine
  - But unfortunately, the layers sometimes breakdown
  - Someone needs to understand what's "under the hood"

A Computer Architecture Picture

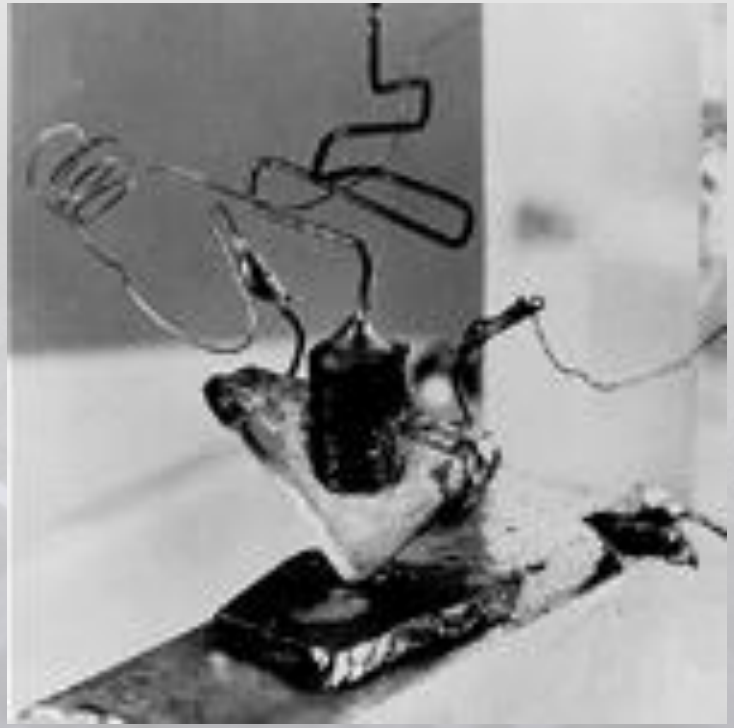


- Computer architecture
  - Definition of **ISA** to facilitate implementation of software layers
- This course mostly on **computer micro-architecture**
  - Design **CPU**, **Memory**, **I/O** to implement **ISA** …

Aside: Semiconductor Technology Background

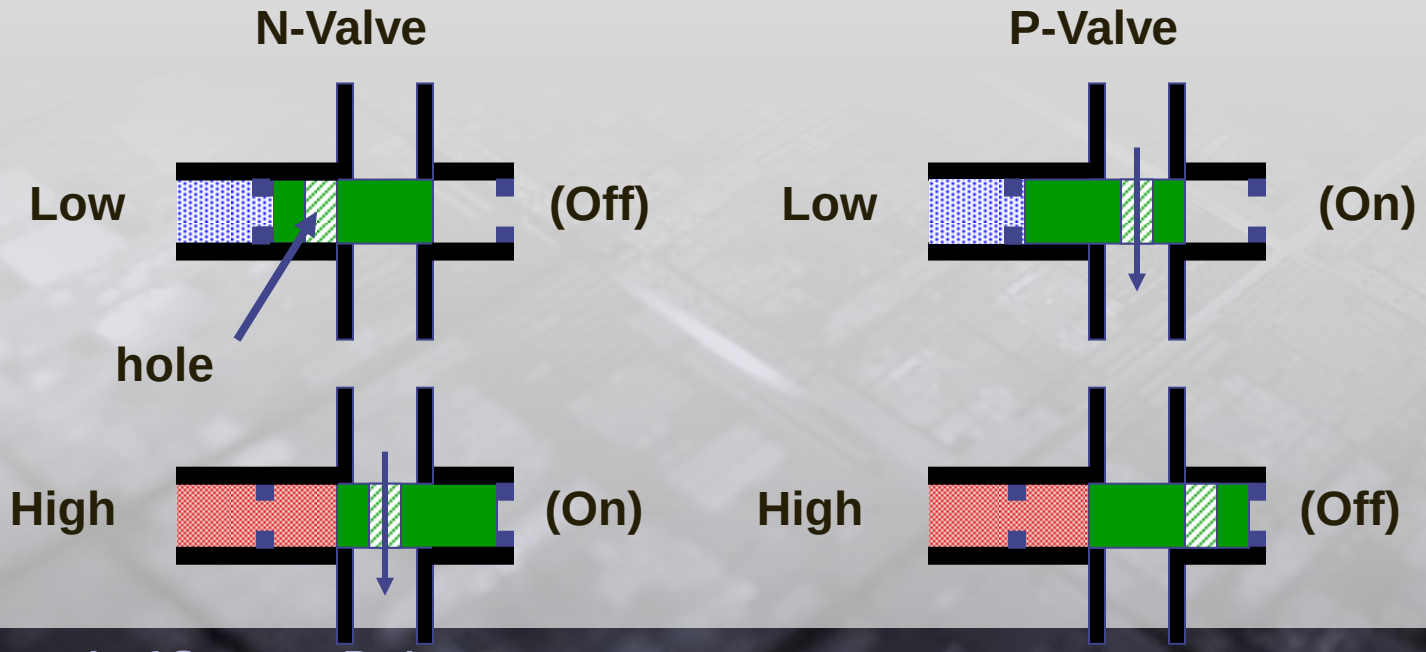


- Transistor (1947)
  - A key invention of 20th century
- Fabrication

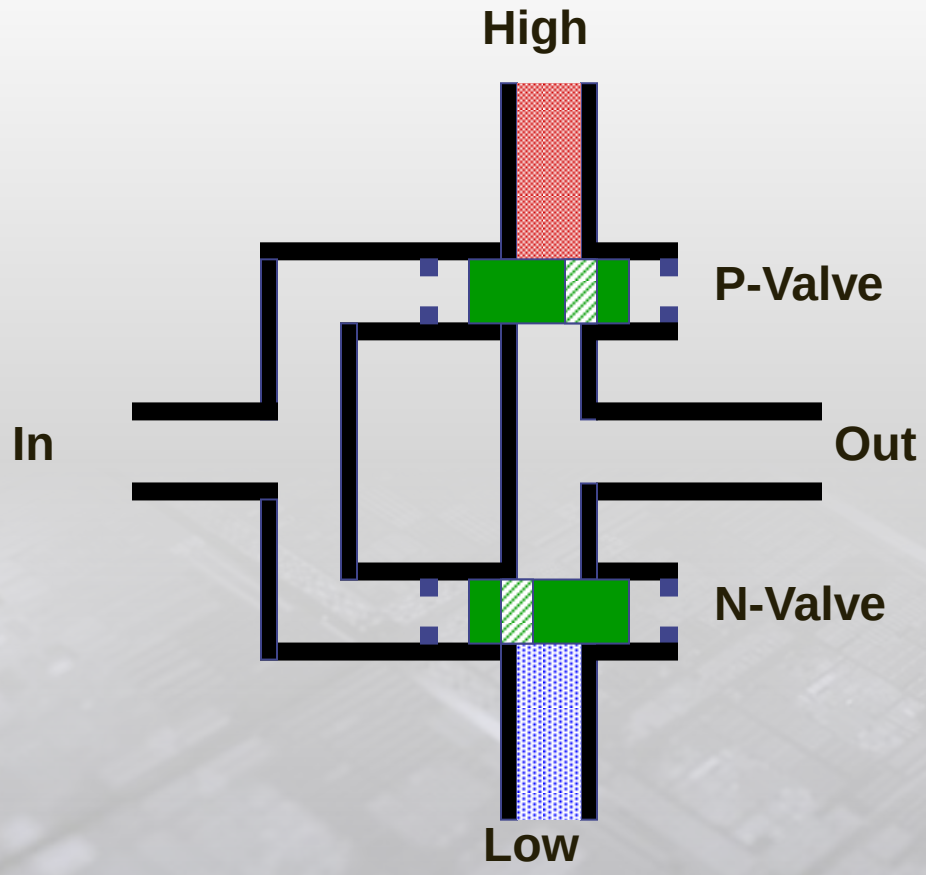


A Transistor Analogy: Computing with Air

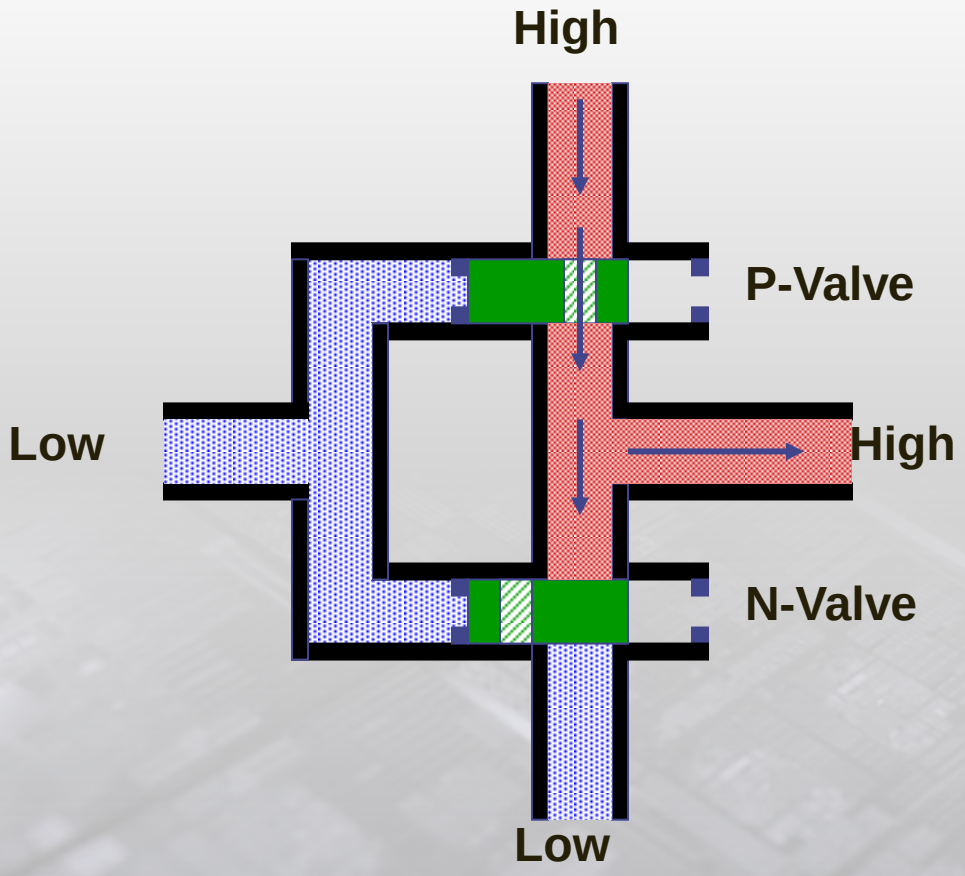
- Use air pressure to encode values
  - High pressure represents a “1” (blow)
  - Low pressure represents a “0” (suck)
- Valve can allow or disallow the flow of air
  - Two types of valves



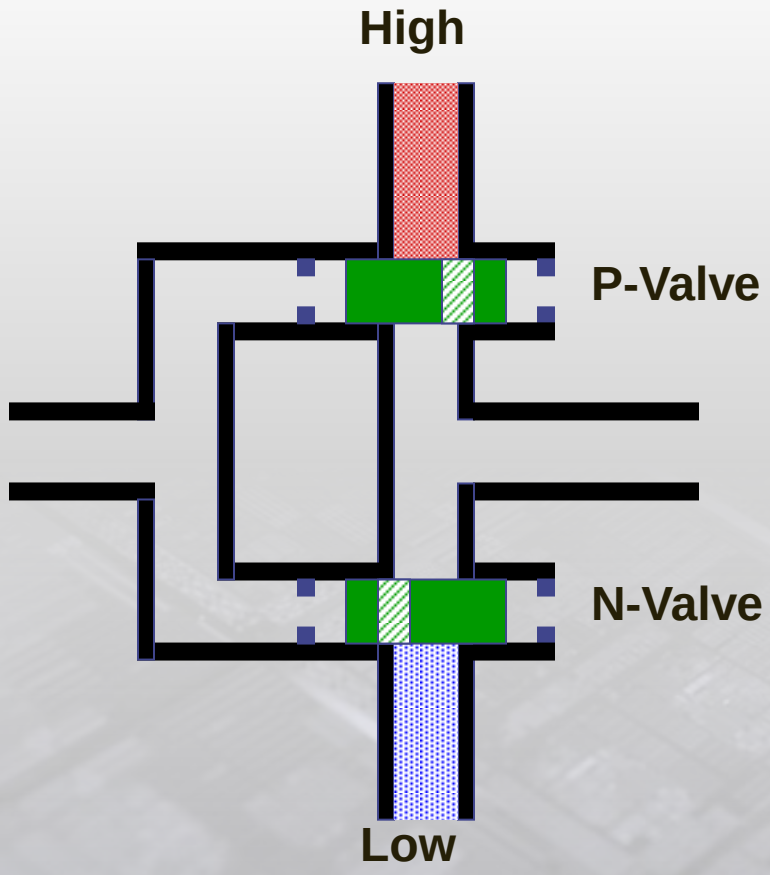
Pressure Inverter



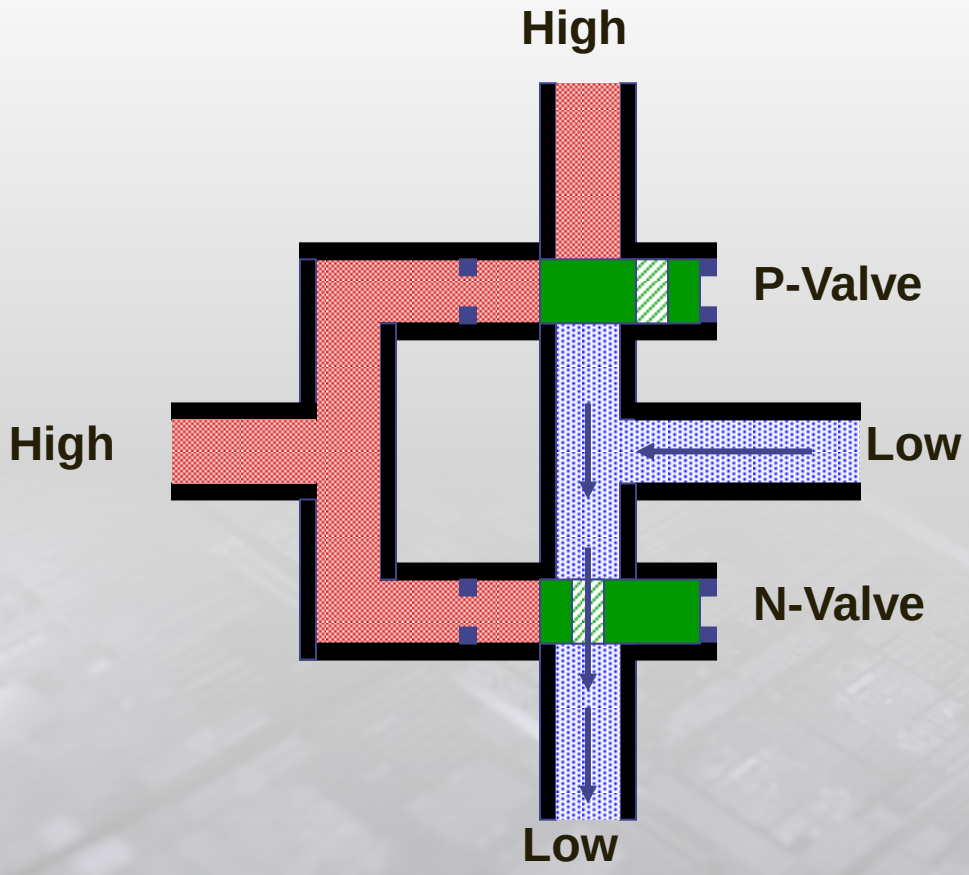
Pressure Inverter (Low to High)



Pressure Inverter



Pressure Inverter (High to Low)

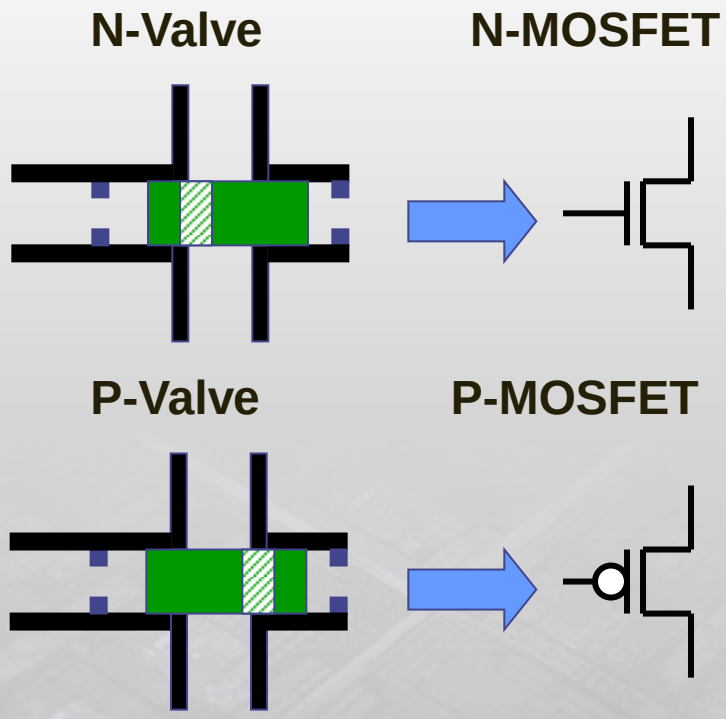


Analogy Explained

- Pressure differential → electrical potential (voltage)
  - Air molecules → electrons
  - High pressure → high voltage
  - Low pressure → low voltage
- Air flow → electrical current
  - Pipes → wires
  - Air only flows from high to low pressure
  - Electrons only flow from high to low voltage
  - Flow only occurs when changing from 1 to 0 or 0 to 1 (valve→valve)
- Valve → transistor
  - The transistor: one of the century's most important inventions

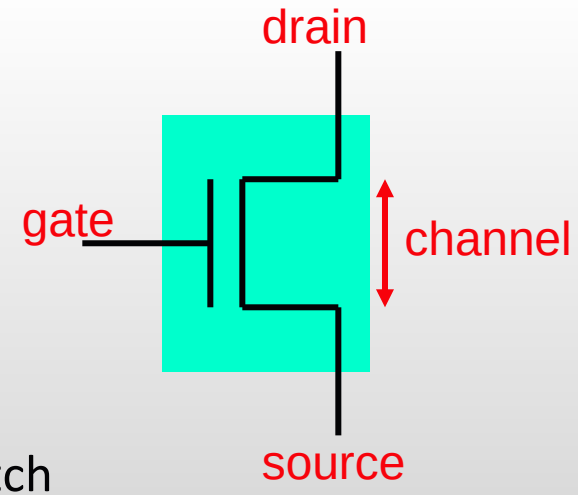
Transistors as Switches

- Two types
  - N-type
  - P-type
- Properties
  - Solid state (no moving parts)
  - Reliable (low failure rate)
  - Small (32nm channel length)
  - Fast (<0.1ns switch latency)



Shaping Force: Technology

- Basic technology element: **MOSFET**



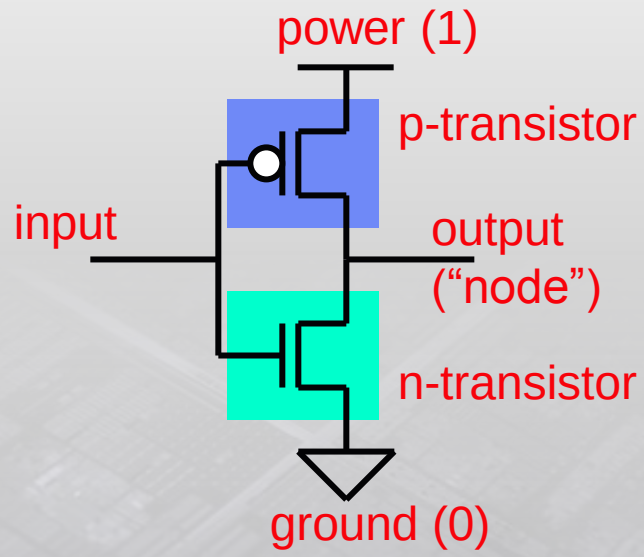
  - **MOS**: metal-oxide-semiconductor
    - Conductor, insulator, semi-conductor
  - **FET**: field-effect transistor
    - Solid-state component acts like electrical switch
    - Channel conducts source→drain when voltage applied to gate

- **Channel length**: characteristic parameter (short → fast)
  - Aka “feature size” or “technology”
  - Currently: 32 nm (0.032 micron) i.e. in 1 cm → ~300.000 trans.
  - Continued miniaturization (scaling) known as “**Moore’s Law**”
    - Won’t last forever, physical limits approaching (or are they?)

Complementary MOS (CMOS)

- Voltages as values
  - Power (V_{DD}) = 1, Ground = 0

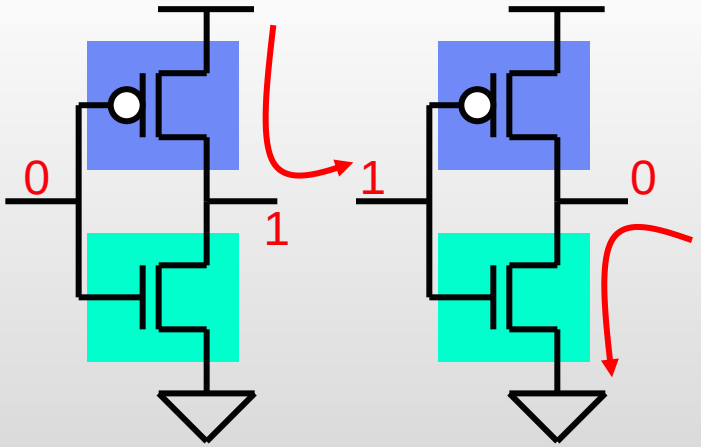
- Two kinds of MOSFETs
  - **N-transistors**
    - Conduct when gate voltage is 1
    - Good at passing 0s
  - **P-transistors**
    - Conduct when gate voltage is 0
    - Good at passing 1s



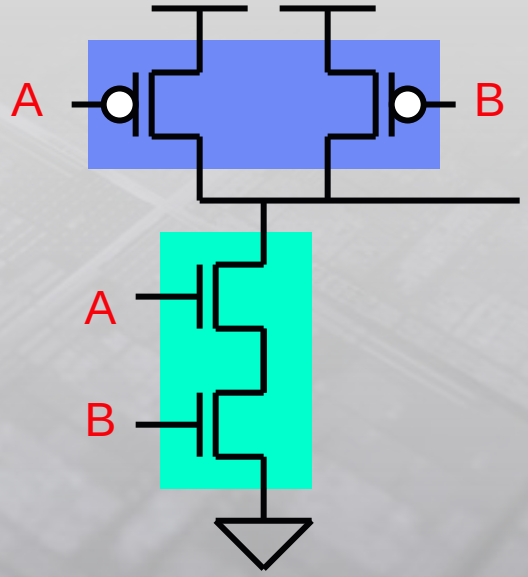
- **CMOS**: complementary n-/p- networks form boolean logic

CMOS Examples

- Example I: inverter
  - Case I: input = 0
    - P-transistor closed, n-transistor open
    - Power charges output (1)
  - Case II: input = 1
    - P-transistor open, n-transistor closed
    - Output discharges to ground (0)



- Example II: look at **truth table**
  - 0, 0 → 1 0, 1 → 1
  - 1, 0 → 1 1, 1 → 0
  - Result: this is a **NAND** (NOT AND)
  - NAND is universal (can build any logic function)



More About CMOS and Technology

- Two different CMOS families

- **SRAM (logic)**: used to make processors
  - Storage implemented as inverter pairs
  - Optimized for speed

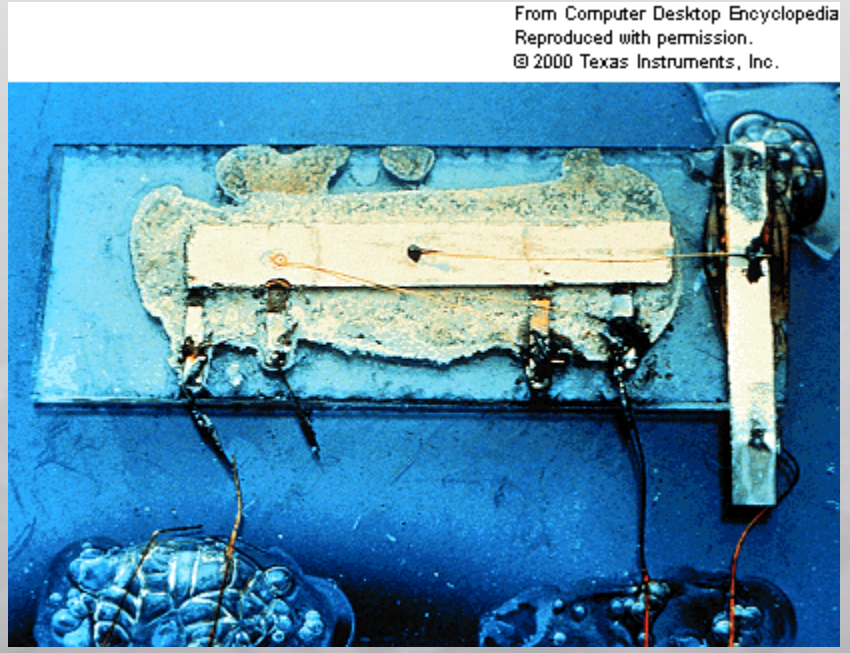
- **DRAM (memory)**: used to make memory
  - Storage implemented as capacitors
  - Optimized for density, cost, power

- Conventional Disk is also a “technology”, but isn’t transistor-based (except Solid State Disks-SSD or other non volatile tech Phase-RAM or **PRAM** , MMRAM, etc…)

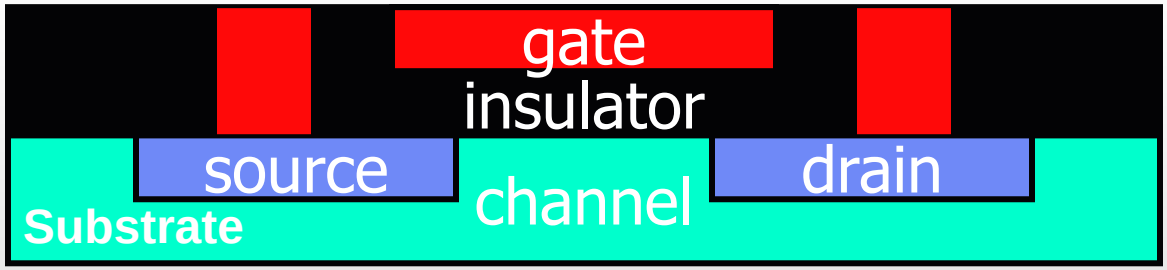
Aside: VLSI + Manufacturing

- **VLSI (very large scale integration)**
  - Transistor manufacturing process
  - Integrated Circuit (1958) as important as transistor itself
  - Multi-step photochemical and electrochemical process
  - Fixed cost per step
  - Cost per transistor shrinks with transistor size

- Other production costs
  - Packaging
  - Test
  - Mask set
  - Design



MOSFET Side View

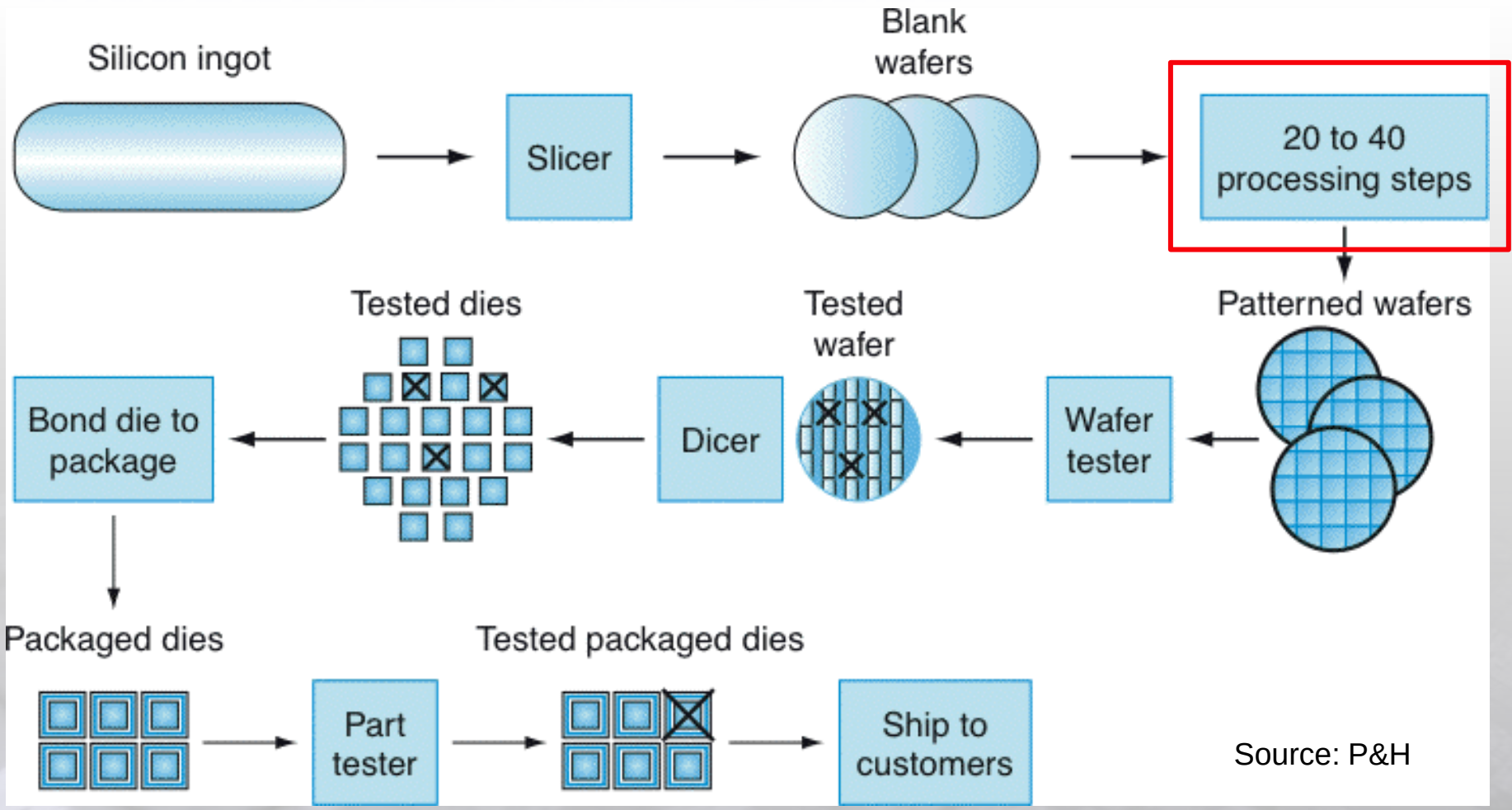


- **MOS**: three materials needed to make a transistor
  - **Metal** – Poli-silicon, Aluminum, Tungsten, Copper: conductor
  - **Oxide** -- Silicon Dioxide (SiO_2): insulator
  - **Semiconductor** - doped Si: conducts under certain conditions
- **FET**: field effect (the mechanism) transistor
  - Voltage on gate: current flows source to drain (transistor on)
  - No voltage on gate: no current (transistor off)

This Unit

- What is a computer and what is computer architecture
- Forces that shape computer architecture
  - Applications
  - Semiconductor technology
- Evaluation metrics: parameters and technology basis
  - Cost
  - Performance
  - Power
  - Reliability

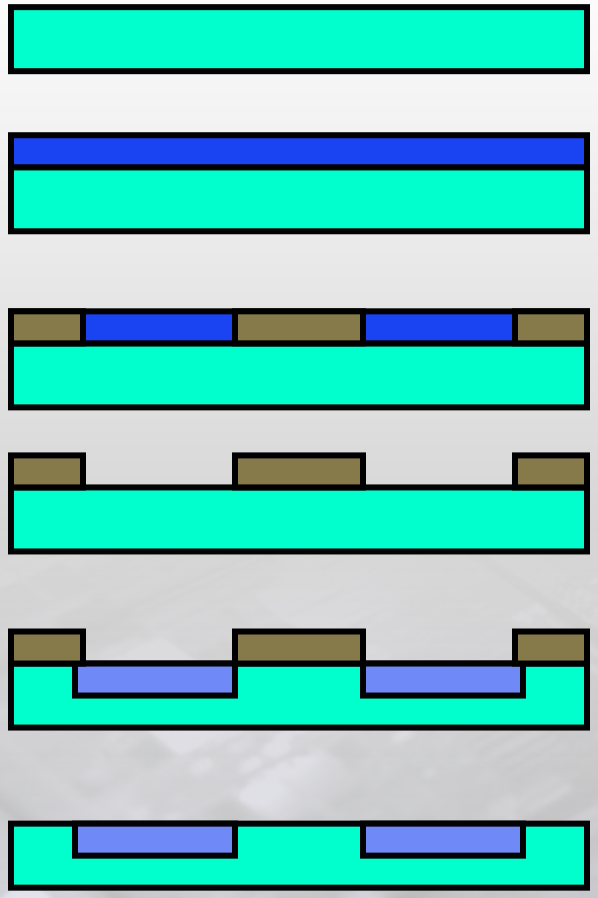
A:Manufacturing Steps



Source: P&H

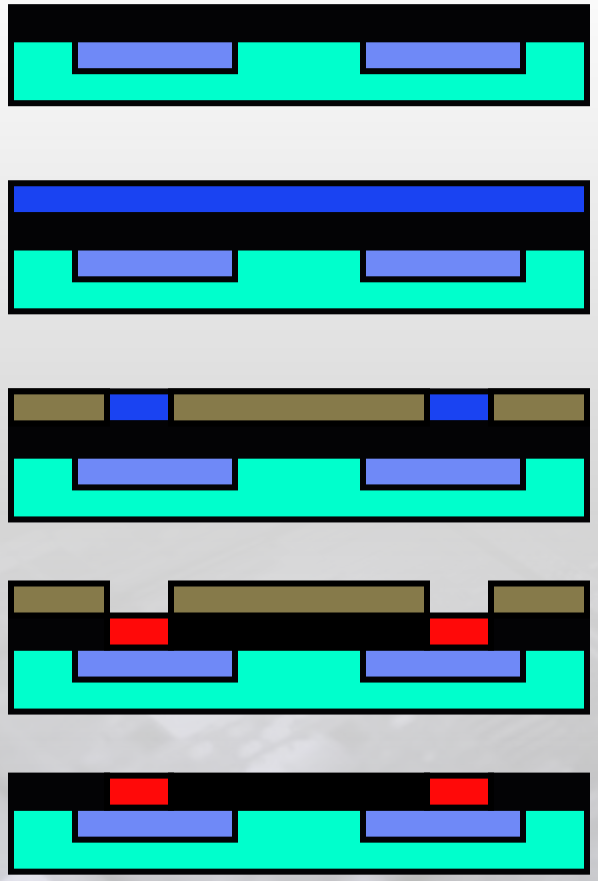
http://www.intel.com/pressroom/kits/chipmaking/index.htm

A:Manufacturing Process



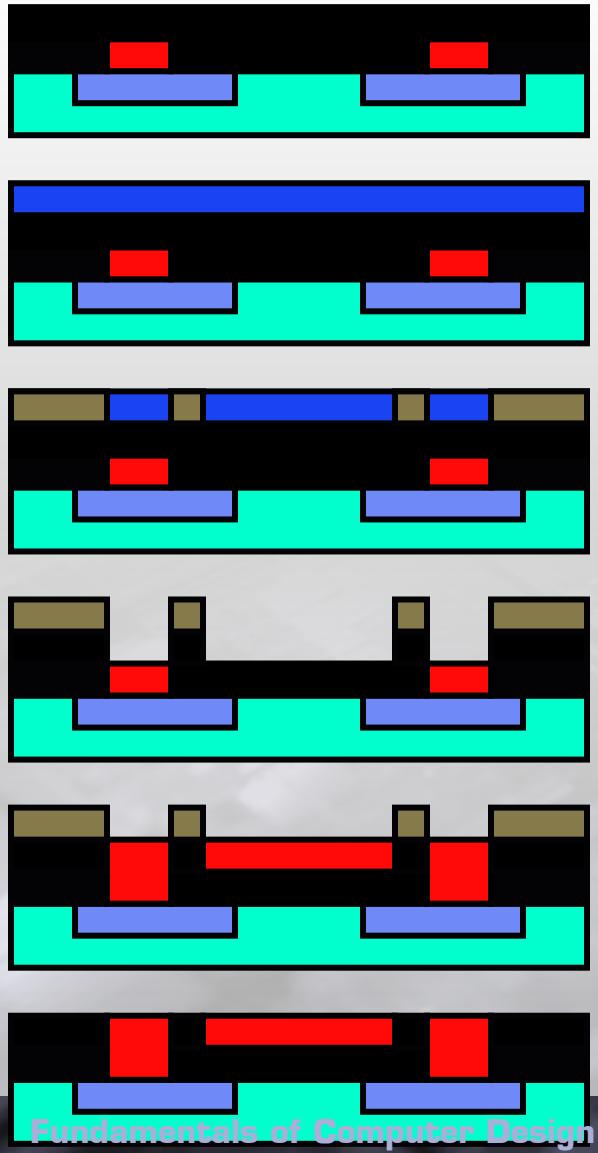
- Start with silicon wafer
- “Grow” photo-resist
  - Molecular beam epitaxy
- Burn positive bias mask
  - Ultraviolet light lithography
- Dissolve unburned photo-resist
  - Chemically
- Bomb wafer with negative ions (P)
  - Doping
- Dissolve remaining photo-resist
  - Chemically
- Continue with next layer

A:Manufacturing Process



- Grow SiO_2
- Grow photo-resist
- Burn “via-level-1” mask
- Dissolve unburned photo-resist
  - And underlying SiO_2
- Grow tungsten “vias”
- Dissolve remaining photo-resist
- Continue with next layer

A:Manufacturing Process

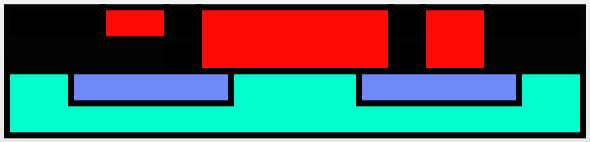


- Grow SiO_2
- Grow photo-resist
- Burn “wire-level-1” mask
- Dissolve unburned photo-resist
  - And underlying SiO_2
- Grow copper “wires”
- Dissolve remaining photo-resist
- Continue with next wire layer…

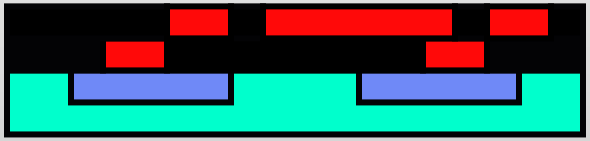
- Typical number of wire layers: 3-11

A:Defects

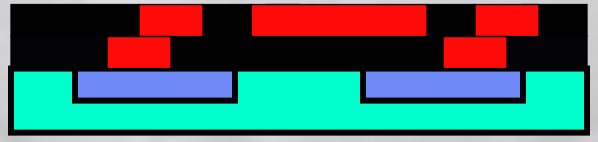
Defective:



Defective:



Slow:



- Defects can arise
  - Under-/over-doping
  - Over-/under-dissolved insulator
  - Mask mis-alignment
  - Particle contaminants
- Try to minimize defects
  - Process margins
  - Design rules
    - Minimal transistor size, separation
- Or, tolerate defects
  - Redundant or “spare” memory cells

Empirical Evaluation

- Metrics
  - Cost
  - Performance
  - Power
  - Reliability
- Often more important in combination than individually
  - Performance/cost (MIPS/$)
  - Performance/power (MIPS/W)
- Basis for
  - Design decisions
  - Purchasing decisions

Cost

- Metric: **$**

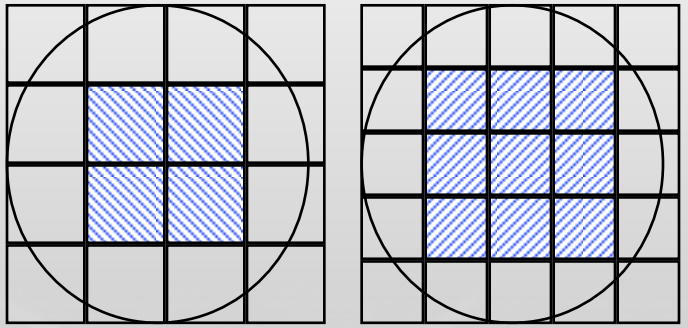
- In grand scheme: CPU accounts for fraction of cost
  - Some of that is profit (Intel's, Dell's)

	Desktop	Laptop	PDA	Phone
$	$100–$300	$150-$350	$50–$100	$10–$20
% of total	10–30%	10–20%	20–30%	20-30%
Other costs	Memory, display, power supply/battery, disk, packaging, **software**			

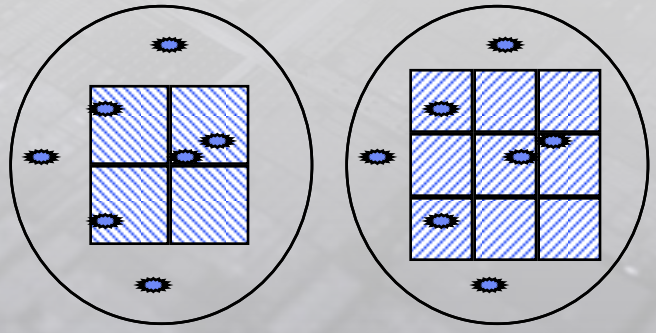
- We are concerned about chip cost
  - Unit cost: costs to manufacture individual chips
  - Startup cost: cost to design chip, build the fab line, marketing

Unit Cost: Integrated Circuit (IC)

- Chips built in multi-step chemical processes on **wafers**
  - Cost / wafer is constant, f(wafer size, number of steps)
- Chip (die) cost is proportional to **area**
  - Larger chips means fewer of them
  - Larger chips means fewer working ones
  - Why? Uniform defect density



  - Chip cost ~ chip area$^{\alpha}$
  - $\alpha = 2–3$



- **Wafer yield**: % wafer that is chips
- **Die yield**: % chips that work
- Yield is increasingly non-binary - fast vs slow chips

Yield/Cost Examples

- Parameters
  - wafer yield = 90%, α = 2, defect density = 2/cm^2

Die size (mm^2)	100	144	196	256	324	400
Die yield	23%	19%	16%	12%	11%	10%
6” Wafer	139(31)	90(16)	62(9)	44(5)	32(3)	23(2)
8” Wafer	256(59)	177(32)	124(19)	90(11)	68(7)	52(5)
10” Wafer	431(96)	290(53)	206(32)	153(20)	116(13)	90(9)

	Wafer Cost	Defect (/cm^2)	Area (mm^2)	Dies	Yield	Die Cost	Package Cost (pins)	Test Cost	Total
Intel 486DX2	$1200	1.0	81	181	54%	$12	$11(168)	$12	$35
IBM PPC601	$1700	1.3	196	66	27%	$95	$3(304)	$21	$119
DEC Alpha	$1500	1.2	234	53	19%	$149	$30(431)	$23	$202
Intel Pentium	$1500	1.5	296	40	9%	$417	$19(273)	$37	$473

Startup Costs

- **Startup costs**: must be amortized over chips sold
  - Research and development: ~$100M per chip (optimistic)
    - 500 person-years @ $200K per
  - Fabrication facilities: ~$2000 M per new line (optimistic)
    - Clean rooms (bunny suits), lithography, testing equipment

- If you sell 10M chips, startup adds ~$200 to cost of each
  - Companies (e.g., Intel) don't make money on new chips
  - They make money on proliferations (shrinks and frequency)
  - No startup cost for these

Moore's Effect on Cost

- Scaling has opposite effects on unit and startup costs
  - + Reduces unit integrated circuit cost
    - Either lower cost for same functionality…
    - Or same cost for more functionality
  - – Increases startup cost
    - More expensive fabrication equipment
    - Takes longer to design, verify, and test chips

This Unit

- What is a computer and what is computer architecture
- Forces that shape computer architecture
  - Applications
  - Semiconductor technology
- Evaluation metrics: parameters and technology basis
  - Cost
  - Performance
  - Power
  - Reliability

Performance

- Two definitions
  - **Latency (execution time)**: time to finish a fixed task
  - **Throughput (bandwidth)**: number of tasks in fixed time
  - Very different: throughput can exploit parallelism, latency cannot
  - Often contradictory (latency **vs.** throughput)
    - Will see many examples of this
  - Choose definition that matches goals (most frequently throughput)
    - Scientific program: latency; web server: throughput?

- Example: move people from A to B, 10 miles
  - Car: capacity = 5, speed = 60 miles/hour
  - Bus: capacity = 60, speed = 20 miles/hour
  - Latency: **car = 10 min**, bus = 30 min
  - Throughput: car = 15 PPH (count return trip), **bus = 60 PPH**

Performance Improvement

- Processor A is X times faster than processor B if
  - Latency(P,A) = Latency(P,B) / X
  - Throughput(P,A) = Throughput(P,B) * X
- Processor A is X% faster than processor B if
  - Latency(P,A) = Latency(P,B) / (1+X/100)
  - Throughput(P,A) = Throughput(P,B) * (1+X/100)

- Car/bus example
  - Latency? Car is 3 times (and 200%) faster than bus
  - Throughput? Bus is 4 times (and 300%) faster than car

What Is 'P' in Latency(P,A)?

- Program
  - Latency(A) makes no sense, processor executes **some program**
  - But which one?
- Actual target workload?
  - \+ Accurate
  - – Not portable/repeatable, overly specific, hard to pinpoint problems
- **Some representative benchmark program(s)?**
  - \+ Portable/repeatable, pretty accurate
  - – Hard to pinpoint problems, may not be exactly what you run
- Some small kernel benchmarks (micro-benchmarks)
  - \+ Portable/repeatable, easy to run, easy to pinpoint problems
  - – Not representative of complex behaviors of real programs

SPEC Benchmarks

- SPEC (Standard Performance Evaluation Corporation)
  - http://www.spec.org/
  - Consortium of companies that collects, standardizes, and distributes benchmark programs
  - Post **SPECmark** results for different processors
    - 1 number that represents performance for entire suite
  - Benchmark suites for CPU, Java, I/O, Web, Mail, etc.
  - Updated every few years: so companies don't target benchmarks

- SPEC CPU 2006
  - 12 "integer": bzip2, gcc, perl, xalancbmk (xml), h264ref (VC), etc.
  - 17 "floating point": povray(openGL), namd (biology), weather, etc.
  - Written in C, C++ and Fortran

Other Benchmarks

- Parallel benchmarks
  - SPLASH2/ PARSEC – Shared memory numerical multithread benchmarks
  - NAS – More general numerical multithread benchmarks
  - SPEC's – OpenMP benchmarks
  - SPECjbb – Java multithreaded database-like workload
- Transaction Processing Council (TPC)
  - TPC-C: On-line transaction processing (OLTP)
  - TPC-H/R: Decision support systems (DSS)
  - TPC-W: E-commerce database backend workload
  - Have parallelism (intra-query and inter-query)
  - Heavy I/O and memory components
- Graphics, Network, Disk, etc…

SPECmark

- Reference machine: Sun Ultra Enterprise II (1997/ 297 Mhz USII)
- Latency SPECmark (SPECint & SPECfp)
  - For each benchmark
    - Take odd number of samples: on both machines
    - Choose median
    - Take latency ratio (Sun Ultra Enterprise II / your machine)
  - Take GMEAN of ratios over all benchmarks
- Throughput SPECmark (SPECint_rate & SPECfp_rate)
  - Run multiple benchmarks in parallel on multiple-processor system
- Recent (latency) leaders
  - SPECint: Intel Core 7i EE 3.2GHz (33.6)
  - SPECfp: Intel Core 7i EE 3.2GHz (33.6)

Example Report

SPEC® CFP2006 Result

ASUSTeK Computer Inc.

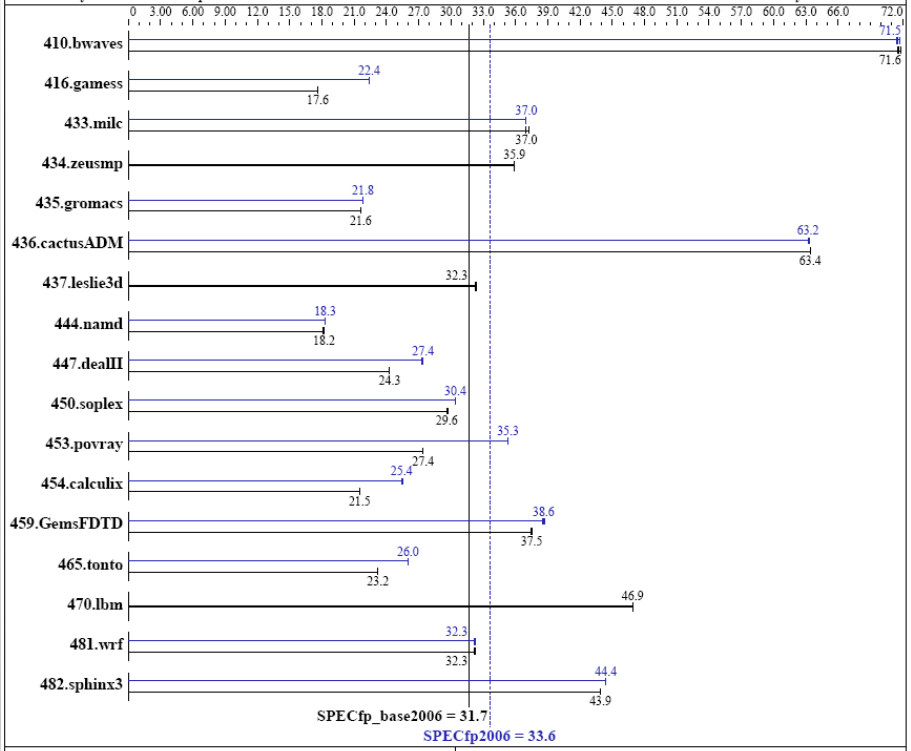
Asus P6T Deluxe (Intel Core i7-965 Extreme Edition)

SPECfp®2006 = 33.6

SPECfp_base2006 = 31.7

CPU2006 license: 13
Test sponsor: Intel Corporation
Tested by: Intel Corporation

Test date: Oct-2008
Hardware Availability: Nov-2008
Software Availability: Nov-2008



Hardware

CPU Name:	Intel Core i7-965 Extreme Edition
CPU Characteristics:	
CPU MHz:	3200
FPU:	Integrated
CPU(s) enabled:	4 cores, 1 chip, 4 cores/chip, 2 threads/core
CPU(s) orderable:	1 chip
Primary Cache:	32 KB I + 32 KB D on chip per core
Secondary Cache:	256 KB I+D on chip per core

Software

Operating System:	Windows Vista Ultimate w/ SP1 (64-bit)
Compiler:	Intel C++ Compiler Professional 11.0 for IA32 Build 20080930 Package ID: w_cproc_p_11.0.054 Intel Visual Fortran Compiler Professional 11.0 for IA32 Build 20080930 Package ID: w_cprof_p_11.0.054 Microsoft Visual Studio 2008 (for libraries)
Auto Parallel:	Yes
File System:	NTFS

Adding/Averaging Performance Numbers

- You can add latencies, but not throughput
  - Latency(P1+P2, A) = Latency(P1,A) + Latency(P2,A)
  - Throughput(P1+P2,A) != Throughput(P1,A) + Throughput(P2,A)
    - 1 km@ 30 km+ 1 mile @ 90 km/hour
      - Average is **not** 60 km/hour
    - 0.033 hours at 30 km/hour + 0.01 hours at 90 km/hour
      - Average is only 47 km/hour! (2 km/ (0.033 + 0.01 hours))
  - Throughput(P1+P2,A) =

    1 / [(1/ Throughput(P1,A)) + (1/ Throughput(P2,A))]

- Same goes for means (averages)
  - **Arithmetic**: $(1/N) * \sum_{P=1..N}$ Latency(P)
    - For units that are proportional to time (e.g., latency)
  - **Harmonic**: $N / \sum_{P=1..N}$ 1/Throughput(P)
    - For units that are inversely proportional to time (e.g., throughput)
  - **Geometric**: $\sqrt[N]{\prod_{P=1..N}}$ Speedup(P)
    - For unitless quantities (e.g., speedups, normalized performance)

CPU Performance Equation

- Multiple aspects to performance: helps to isolate them

- Latency(P,A) = seconds / program =
  - (instructions / program) * (cycles / instruction) * (seconds / cycle)
- **Instructions / program**: dynamic instruction count
  - Function of program, compiler, instruction set architecture (ISA)
- **Cycles / instruction**: CPI
  - Function of program, compiler, ISA, micro-architecture
- **Seconds / cycle**: clock period
  - Function of micro-architecture, technology parameters

- For low latency (better performance) minimize all three
  - Hard: often pull against the other

Danger: Partial Performance Metrics

- Micro-architects often ignore dynamic instruction count
  - Typically work in one ISA/one compiler → treat it as fixed

- CPU performance equation becomes
  - seconds / instruction = (cycles / instruction) * (seconds / cycle)
  - This is a latency measure, if we care about throughput …
  - **Instructions / second** = (instructions / cycle) * (cycles / second)

- **MIPS** (millions of instructions per second)
  - Instructions / second * 10^{-6}
  - **Cycles / second**: clock frequency (in MHz)
  - Example: CPI = 2, clock = 500 MHz, what is MIPS?
    - 0.5 * 500 MHz * 10^{-6} = 250 MIPS
  - Example problem situation:
    - compiler removes instructions, program faster
    - However, “MIPS” goes down (misleading)

MIPS and MFLOPS (MegaFLOPS)

- Problem: MIPS may vary inversely with performance
  - Some optimizations actually add instructions
  - Work per instruction varies (e.g., FP mult vs. integer add)
  - ISAs are not equivalent

- **MFLOPS**: like MIPS, but counts only FP ops, because…
  + FP ops can't be optimized away
  + FP ops have longest latencies anyway
  + FP ops are same across machines
- May have been valid in 1980, but today…
  - Most programs are "integer", i.e., light on FP
  - Loads from memory take much longer than FP divide
  - Even FP instructions sets are not equivalent
- Upshot: MIPS not perfect, but (usually) more useful than MFLOPS

Danger: Partial Performance Metrics II

- Micro-architects often ignore dynamic instruction count…
- … but general **public** (mostly) also **ignores CPI**
  - Equates clock frequency with performance!!

- Which processor would you buy?
  - Processor A: CPI = 2, clock = 500 MHz
  - Processor B: CPI = 1, clock = 300 MHz
  - Probably A, but B is faster (assuming **same** ISA/compiler)

- Classic example
  - 800 MHz PentiumIII **faster** than 1 GHz Pentium4
  - Same ISA and compiler

Cycles per Instruction (CPI)

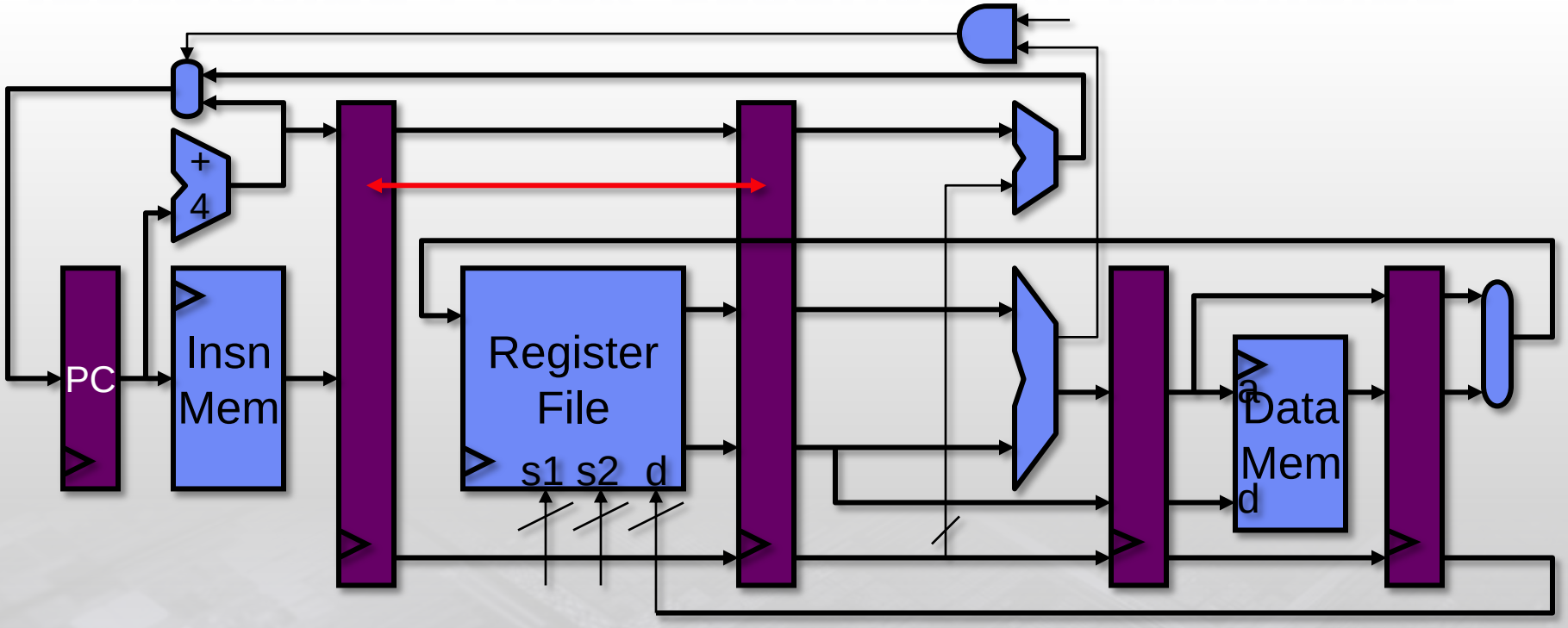
- This course is mostly about improving **CPI**
  - Cycle/instruction for **average instruction**
  - **IPC** = 1/CPI
    - Used more frequently than CPI, but harder to compute with
  - Different instructions have different cycle costs
    - E.g., integer add typically takes 1 cycle, FP divide takes > 10
  - Assumes you know something about instruction frequencies

- CPI example
  - A program executes equal integer, FP, and memory operations
  - Cycles per instruction type: integer = 1, memory = 2, FP = 3
  - What is the CPI? (0.33 * 1) + (0.33 * 2) + (0.33 * 3) = 2
  - **Caveat**: this sort of calculation ignores dependences completely
    - Back-of-the-envelope arguments only

Another CPI Example

- Assume a processor with instruction frequencies and costs
  - Integer ALU: 50%, 1 cycle
  - Load: 20%, 5 cycle
  - Store: 10%, 1 cycle
  - Branch: 20%, 2 cycle
- Which change would improve performance more?
  - A. Branch prediction to reduce branch cost to 1 cycle?
  - B. A bigger data cache to reduce load cost to 3 cycles?
- Compute CPI
  - Base = 0.5*1 + 0.2*5 + 0.1*1 + 0.2*2 = 2
  - A = 0.5*1 + 0.2*5 + 0.1*1 + 0.2*1 = 1.8
  - B = 0.5*1 + 0.2*3 + 0.1*1 + 0.2*2 = 1.6 (**winner**)

Increasing Clock Frequency: Pipelining



- CPU is a pipeline: **compute** stages separated by **latches**

- **Clock period**: maximum delay of any stage
  - Number of gate levels in stage
  - Delay of individual gates (these days, wire delay more important)

Increasing Clock Frequency: Pipelining

- Reduce pipeline stage delay
  - Reduce logic levels and wire lengths (better design)
  - Complementary to technology efforts (described later)
  - Increase number of pipeline stages (multi-stage operations)
  - Often causes CPI to increase
  - At some point, actually causes performance to decrease
  - "Optimal" pipeline depth is program and technology specific

- Remember example
  - PentiumIII: 12 stage pipeline, 800 MHz

    faster than
  - Pentium4: 22 stage pipeline, 1 GHz
  - Actual Intel designs: more like PentiumIII

CPI and Clock Frequency

- System components "clocked" independently
  - E.g., Increasing processor clock frequency doesn't improve memory performance

- Example
  - Processor A: $CPI_{CPU} = 1$, $CPI_{MEM} = 1$, clock = 500 MHz
  - What is the speedup if we double clock frequency?
  - Base: CPI = 2 $\rightarrow$ IPC = 0.5 $\rightarrow$ MIPS = 250
  - New: CPI = **3** $\rightarrow$ IPC = 0.33 $\rightarrow$ MIPS = 333
    - Clock *= 2 $\rightarrow$ CPI_{MEM} *= 2
  - Speedup = 333/250 = 1.33 << 2

- What about an infinite clock frequency?
  - Only a x2 speedup

Measuring CPI

- How are CPI and execution-time actually measured?
  - Execution time: time (Unix): wall clock + CPU + system
  - CPI = CPU time / (clock frequency * dynamic insn count)
  - How is dynamic instruction count measured?
  - Want CPI breakdowns (CPI_{CPU}, CPI_{MEM}, etc.) to see what to fix
- CPI breakdowns
  - Hardware event counters
    - Calculate CPI using counter frequencies/event costs
  - Cycle-level micro-architecture simulation (e.g., **Simics**)
    - \+ Measures breakdown "exactly" provided
      - \+ Models micro-architecture faithfully
      - \+ Ran realistic workload
  - Method of choice for many micro-architects (and you)

Improving CPI

- This course is more about improving CPI than frequency
  - Historically, clock accounts for 70%+ of performance improvement (some exceptions)
    - Achieved via deeper pipelines
  - That will (have to) change
    - Deep pipelining is not power efficient
    - Physical speed limits are approaching
    - 1GHz: 1999, 2GHz: 2001, 3GHz: 2002, 3.2GHz: 2008
  - Techniques we will look at
    - Caching, speculation, multiple issue, out-of-order issue
    - Multiprocessing, Vectors more…
- Moore helps because CPI reduction requires transistors
  - The definition of parallelism is “more transistors”
  - But best example is caches

Performance Trends

	386	486	Pentium	PentiumII	Pentium4	Core2
Year	1985	1989	1993	1998	2001	2006
Technode (nm)	1500	800	350	180	130	65
Transistors (M)	0.3	1.2	3.1	5.5	42	291
Clock (MHz)	**16**	**25**	**66**	**200**	**1500**	3000
Pipe stages	"1"	5	5	10	22	~15
(Peak) IPC	0.4	1	2	3	**3**	**"8"**
(Peak) MIPS	6	25	132	600	4500	24000

Performance Rules of Thumb

- Make common case fast
  - Sometimes called "**Amdahl's Law**"
  - $Speedup_{overall} = 1 / ((1 - fraction_x) + fraction_x/Speedup_x)$
  - Corollary: don't optimize 5% to the detriment of other 95%
  - $Speedup_{overall} = 1 / ((1 - 5\%) + 5\%/infinity) = 1.05$
- Build a balanced system
  - Don't optimize 1% to the detriment of other 99%
  - Don't over-engineer capabilities that cannot be utilized
- Design for actual, not peak, performance
  - Peak performance: "Performance you are guaranteed not to exceed"
  - Greater than "actual" or "average" or "sustained" performance
    - Why? Caches misses, branch mispredictions, limited ILP, etc.
  - For actual performance X, machine capability must be > X

This Unit

- What is a computer and what is computer architecture
- Forces that shape computer architecture
  - Applications
  - Semiconductor technology
- Evaluation metrics: parameters and technology basis
  - Cost
  - Performance
  - Power & speed
  - Reliability

Transistor Speed, Power, and Reliability

- Transistor characteristics and scaling impact:
  - Switching speed
  - Power
  - Reliability

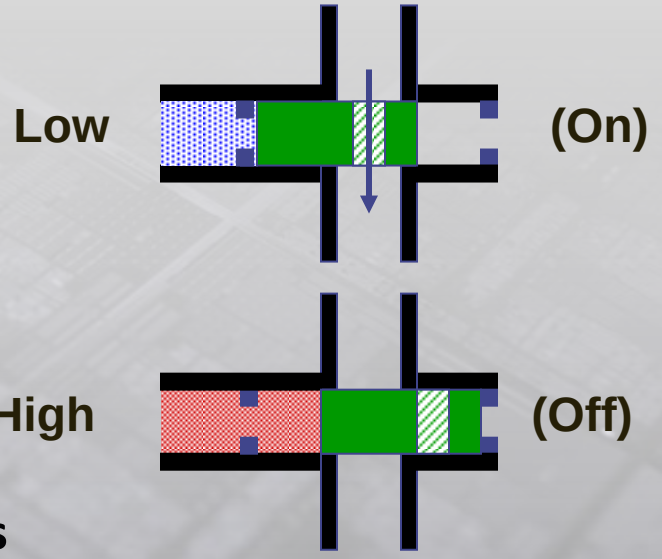
- Simplistic gate delay model for **architecture**
  - Each Not, NAND, NOR, AND, OR gate has delay of “1”
  - Reality is not so simple

Technology Basis of Transistor Speed

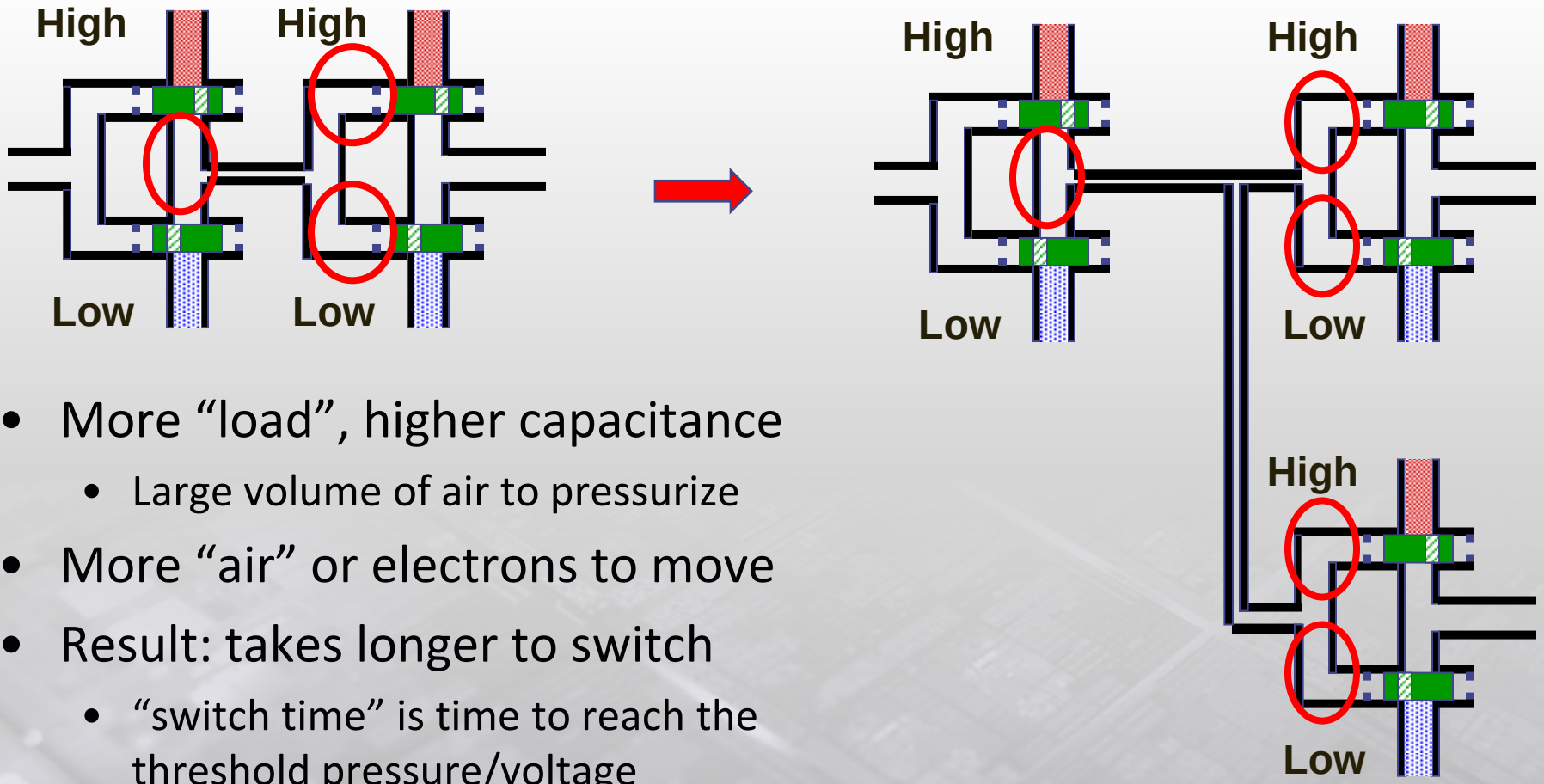
- Physics 101: delay through an electrical component ~ **RC**
  - **Resistance (R)** 
    - Slows rate of charge flow
    - ~ length / cross-section area
  - **Capacitance (C)** 
    - Stores charge
    - ~ surface-area / distance-to-other-plate
  - **Voltage (V)**
    - Electrical pressure
  - **Threshold Voltage (V_t)**
    - Voltage at which a transistor turns “on”

Analogy Extended

- Physics 101: delay through an electrical component ~ **RC**
  - **Resistance (R)** 
    - Slows rate of charge flow
    - ~ length / cross-section area
    - **Analogy: the friction of air flowing through a tube**
  - **Capacitance (C)** 
    - Stores charge
    - ~ surface-area / distance-to-other-plate
    - **Analogy: volume of tubes**
  - **Voltage (V)**
    - Electrical pressure
    - **Analogy: compressed air pressure**
  - **Threshold Voltage (V_t)**
    - Voltage at which a transistor turns “on”
    - **Analogy: pressure at which valve switches**



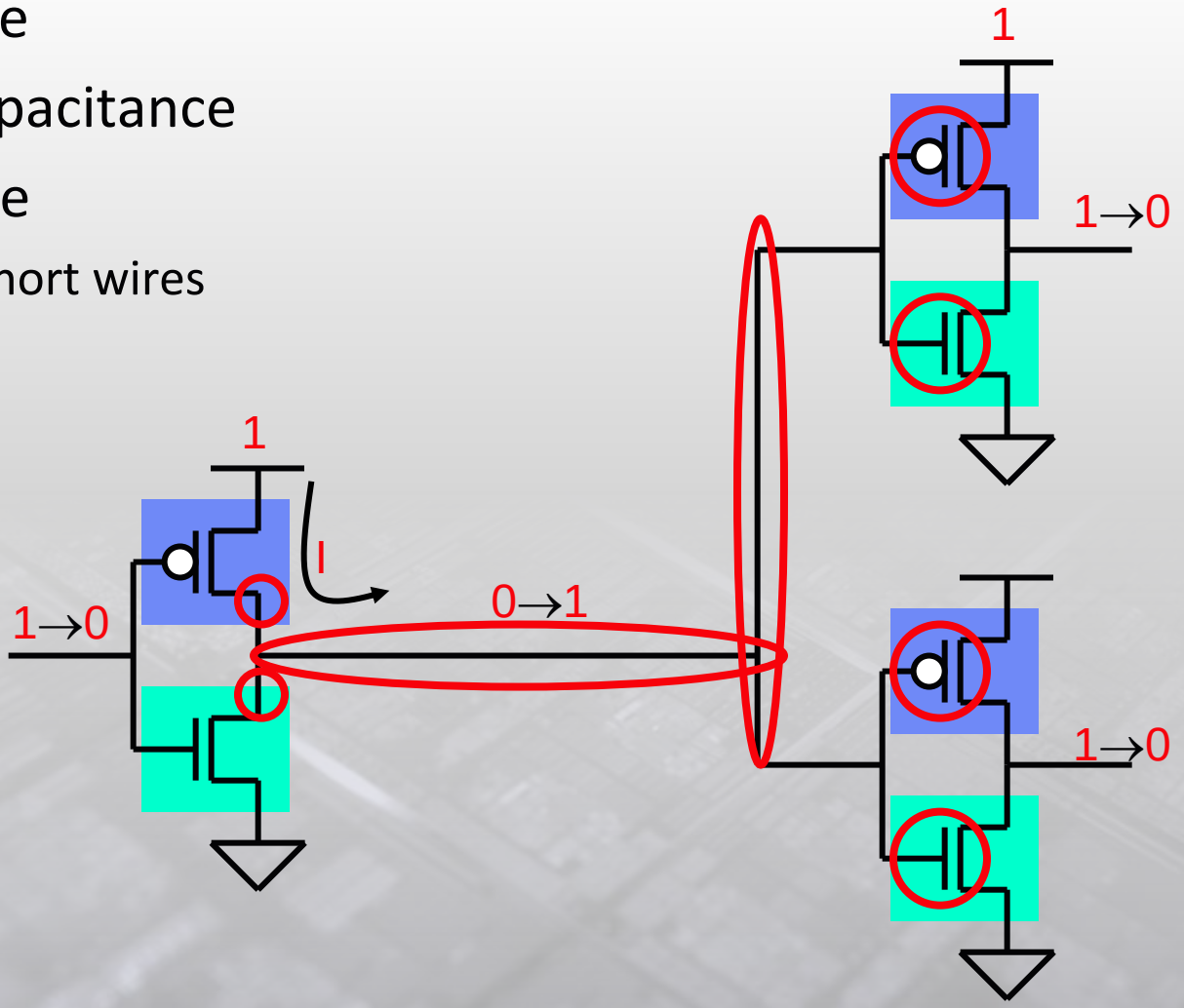
Capacitance Analogy: Air Capacity



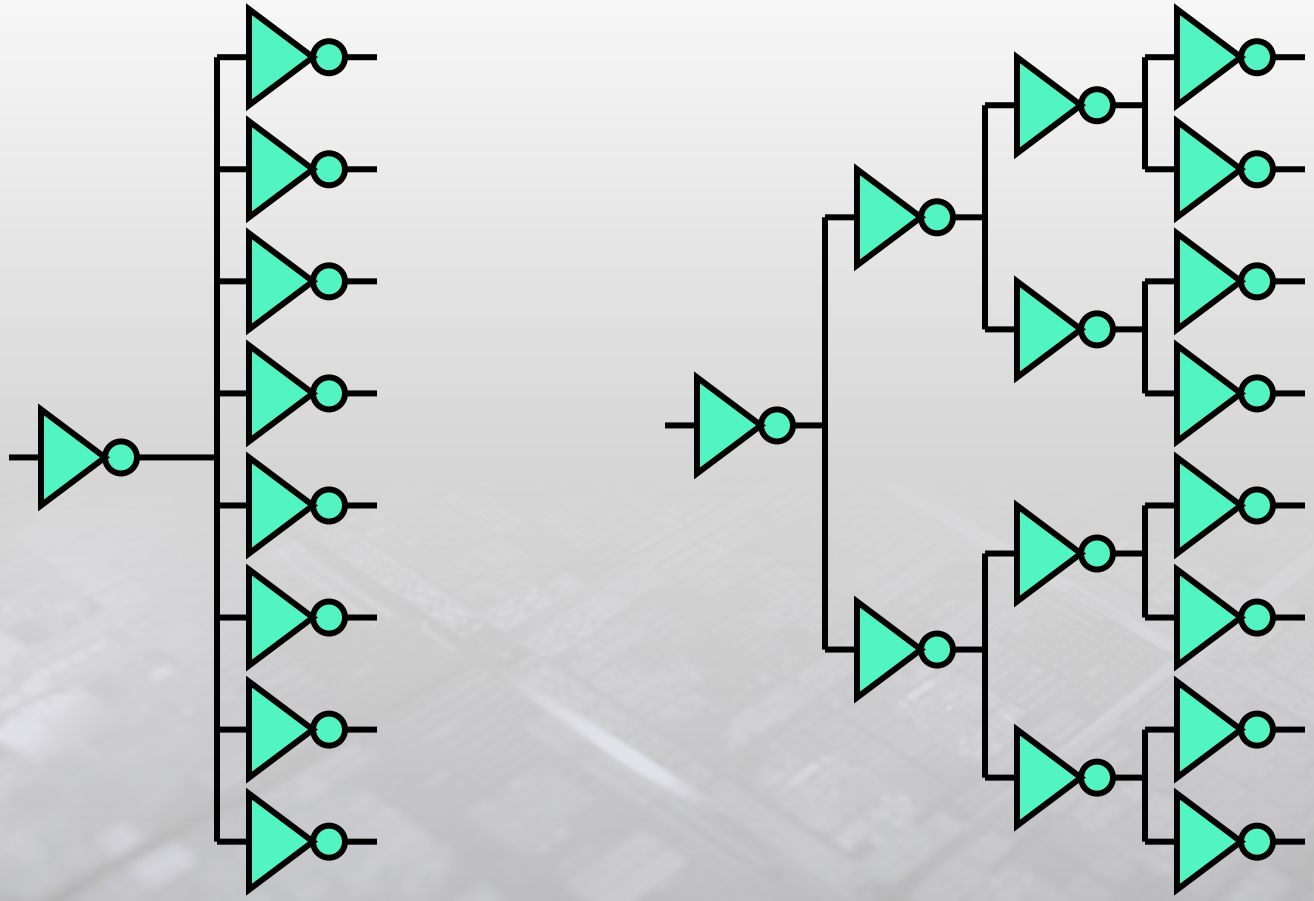
- More “load”, higher capacitance
  - Large volume of air to pressurize
- More “air” or electrons to move
- Result: takes longer to switch
  - “switch time” is time to reach the threshold pressure/voltage
- The “fan-out” of the device impacts its switching speed

Capacitance

- Gate capacitance
- Source/drain capacitance
- Wire capacitance
  - Negligible for short wires

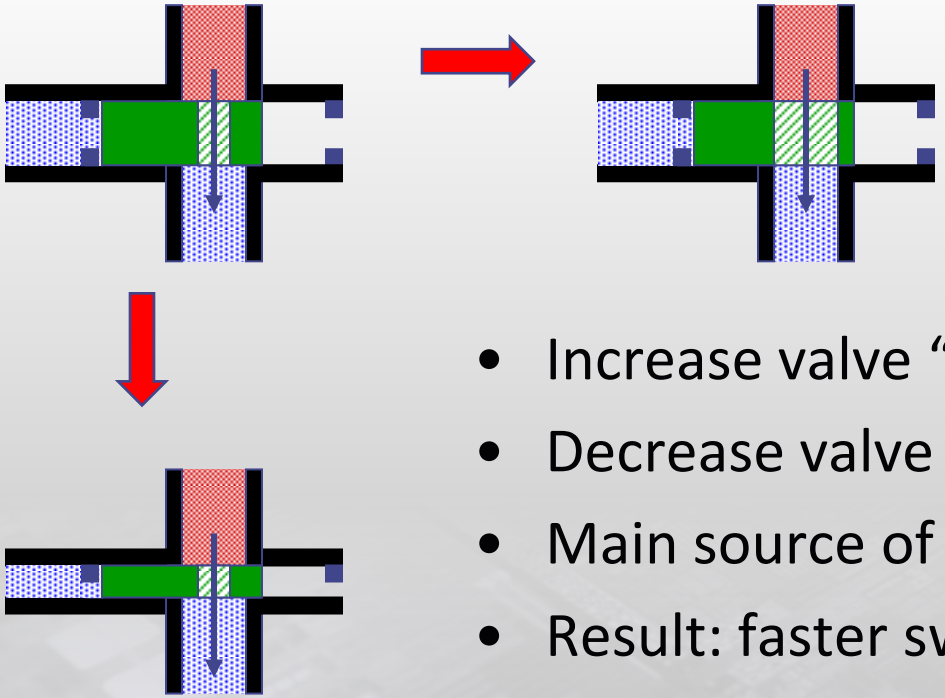


Which is faster? Why?

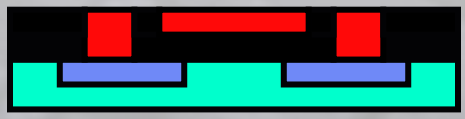


(Assume wires are short enough to have negligible resistance/capacitance)

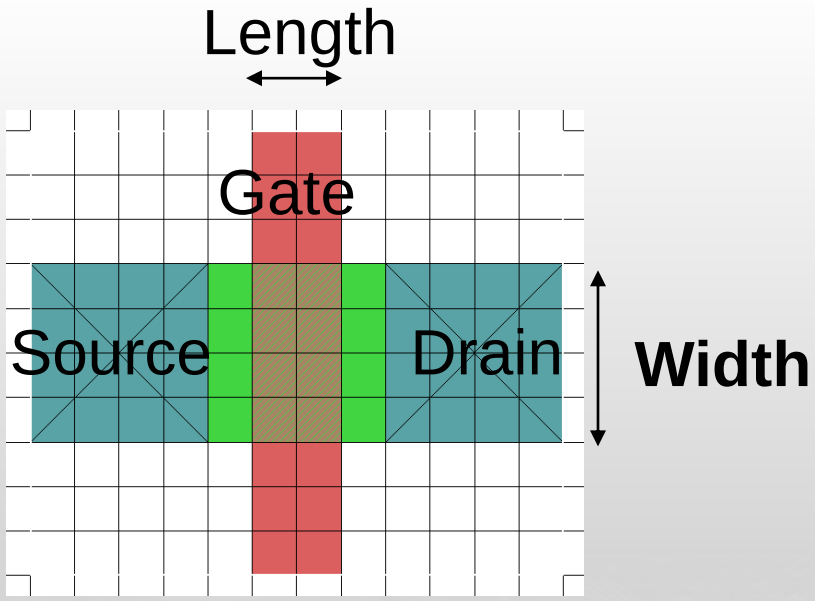
Trans. Resistance Analogy: Valve Friction

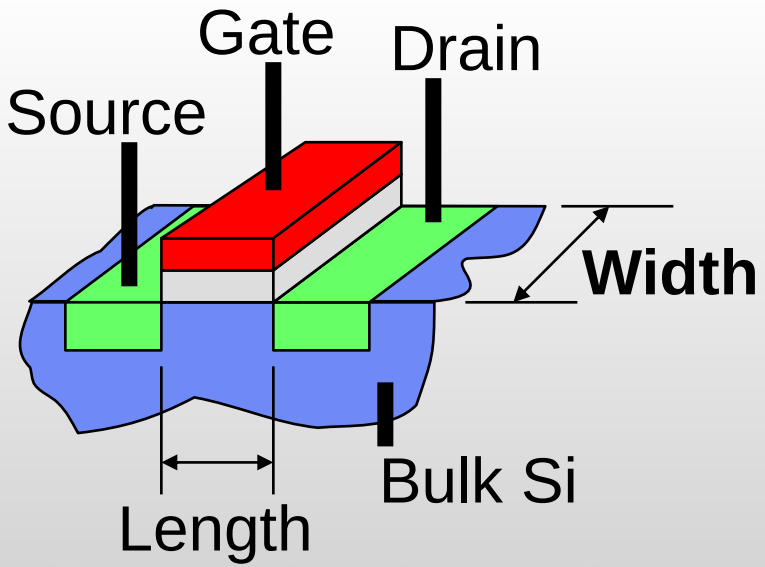


- Increase valve “width”, lower resistance
- Decrease valve “length”, lower resistance
- Main source of transistor resistance
- Result: faster switching



Transistor Geometry: Width

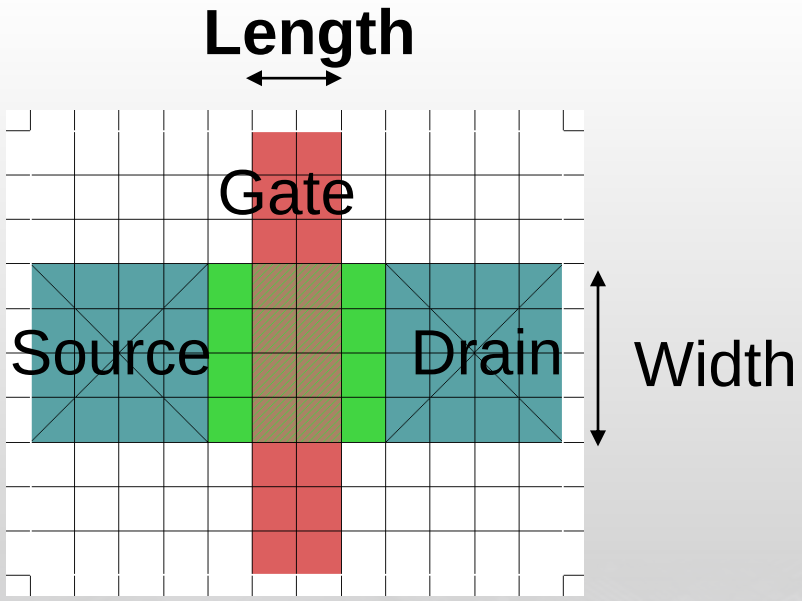


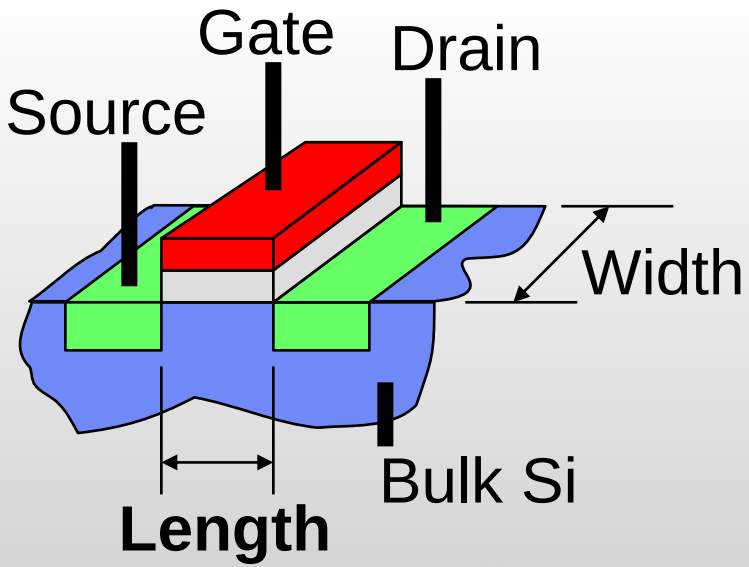


Diagrams © Krste Asanovic, MIT

- **Transistor width**, set by designer on a per-transistor basis
- Wider transistors:
  - **Lower resistance** of channel (increases drive strength)
  - But, **increases capacitance** of gate/source/drain
- Result: set width to balance these conflicting effects

Transistor Geometry: Length & Scaling

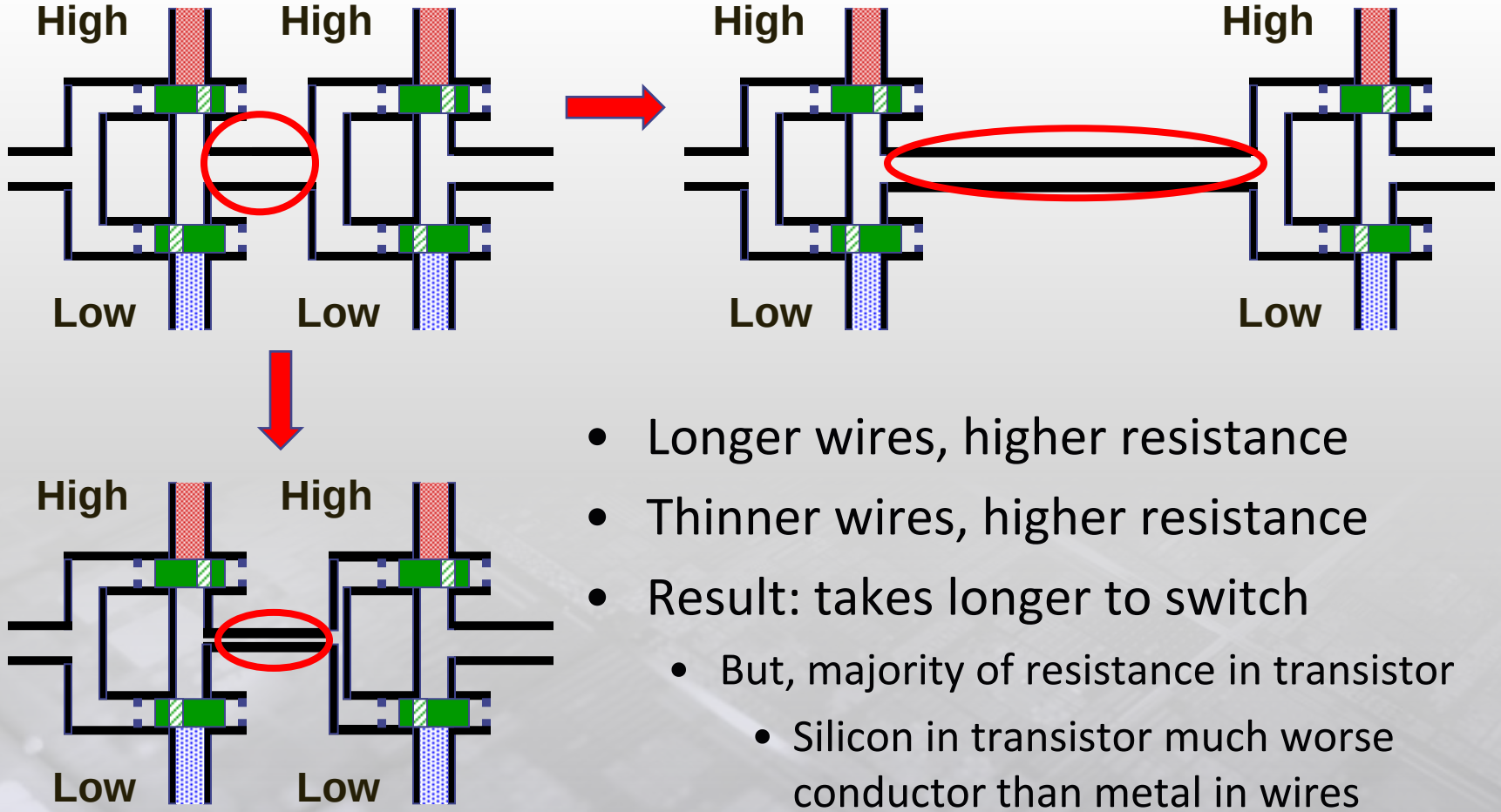




Diagrams © Krste Asanovic, MIT

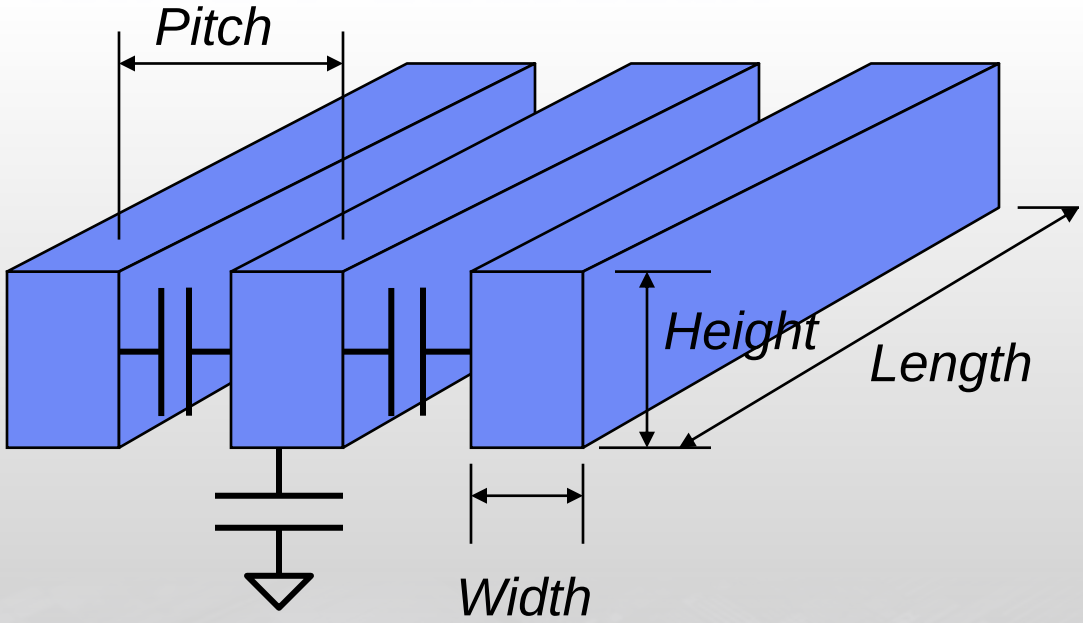
- **Transistor length**: characteristic of "process generation"
  - 90nm refers to the transistor gate length, same for all transistors
- Shrink transistor length:
  - Lower resistance of channel (shorter)
  - Lower gate/source/drain capacitance
- Result: transistor drive strength linear as gate length shrinks

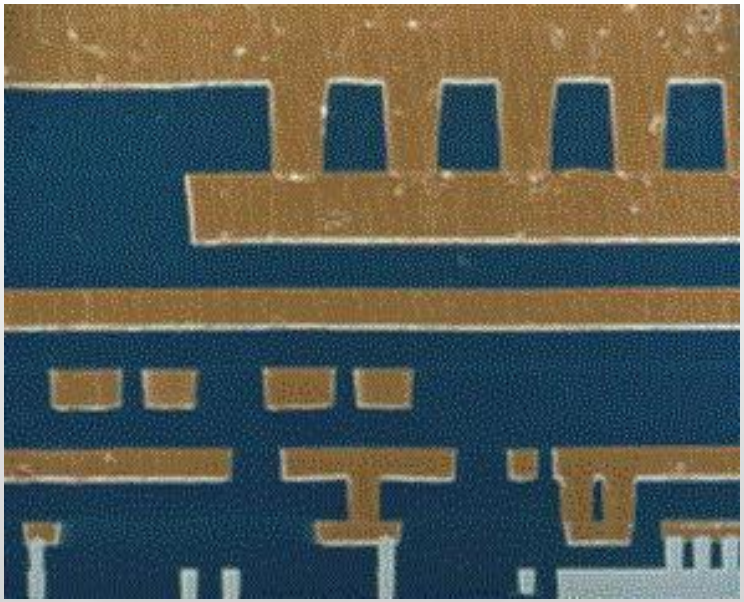
Wire Resistance Analogy: Tube Friction



- Longer wires, higher resistance
- Thinner wires, higher resistance
- Result: takes longer to switch
  - But, majority of resistance in transistor
    - Silicon in transistor much worse conductor than metal in wires
  - So, only significant for long wires

Wire Geometry





From slides © Krste Asanovic, MIT

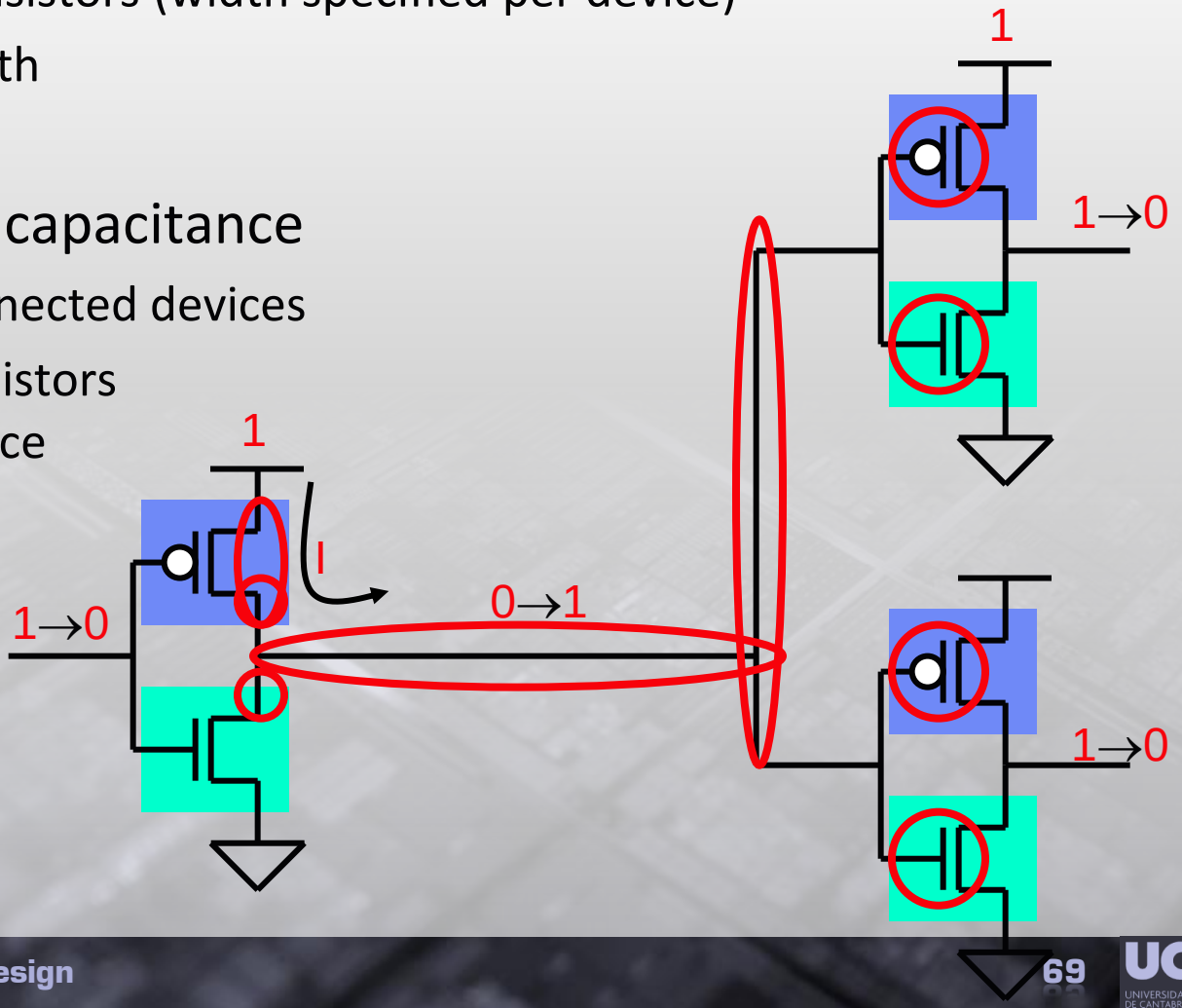
- Transistors 1-dimensional for design purposes: **width**
- Wires 4-dimensional: **length**, **width**, **height**, **"pitch"**
  - Longer wires have more resistance
  - "Fatter" wires have less resistance
  - Closer wire spacing ("pitch") increases capacitance

Wire Delay

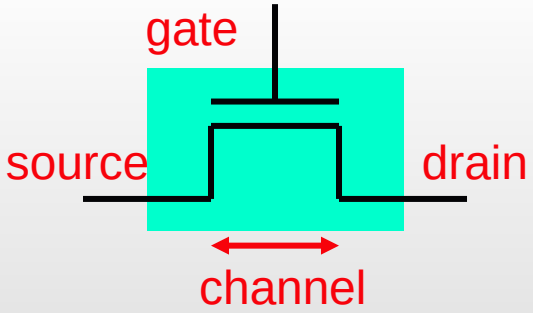
- RC Delay of wires
  - Resistance proportional to length / cross section
    - Wires with smaller cross section have higher resistance
    - Type of metal (copper vs aluminum)
  - Capacitance proportional to length
    - And wire spacing (closer wires have large capacitance)
    - Type of material between the wires
- Result: delay of a wire is quadratic in length
  - Insert “inverter” repeaters for long wires to
  - Bring it back to linear delay, but repeaters still add delay
- Trend: wires are getting relatively slow to transistors
  - And relatively longer time to cross relatively larger chips

RC Delay Model Ramifications

- Want to reduce resistance
  - Wide drive transistors (width specified per device)
  - Short gate length
  - Short wires
- Want to reduce capacitance
  - Number of connected devices
  - Less-wide transistors (gate capacitance of next stage)
  - Short wires



Moore's Law: Technology Scaling



- **Moore's Law**: aka "technology scaling"
  - Continued miniaturization (esp. reduction in channel length)
  - \+ Improves switching speed, power/transistor, area(cost)/transistor
  - – Reduces transistor reliability
  - Literally: DRAM density (transistors/area) doubles every 18 months
  - Public interpretation: performance doubles every 18 months
    - Not quite right, but helps performance in three ways

Moore's Effect #1: Transistor Count

- Linear shrink in each dimension
  - 180nm, 130nm, 90nm, 65nm, 45nm, 32nm, …
  - Each generation is a 1.414 linear shrink
    - Shrink each dimension (2D)
  - Results in 2x more transistors (1.414*1.414)
- More transistors reduces cost
- More transistors can increase performance
  - Job of a computer architect: use the ever-increasing number of transistors
  - Examples: caches, exploiting parallelism (ILP, TLP, DLP)

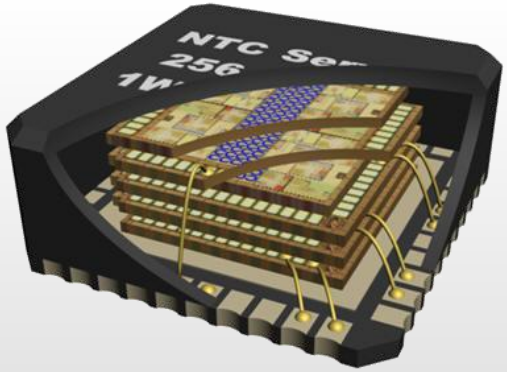
Moore's Effect #2: RC Delay

- **First-order: speed scales proportional to gate length**
  - Has provided much of the performance gains in the past
- Scaling helps wire and gate delays in some ways…
  + Transistors become shorter (Resistance↓), narrower (Capacitance↓)
  + Wires become shorter (Length↓ → Resistance↓)
  + Wire "surface areas" become smaller (Capacitance↓)
- Hurts in others…
  – Transistors become narrower (Resistance↑)
  – Gate insulator thickness becomes smaller (Capacitance↑)
  – Wires becomes thinner (Resistance↑)
- What to do?
  - Take the good, use wire/transistor sizing & repeaters to counter bad
  - Exploit new materials: Aluminum → Copper, metal gate, high-K

Moore's Effect #3: Psychological

- **Moore's Curve**: common interpretation of Moore's Law
  - "CPU performance doubles every 18 months"
  - Self fulfilling prophecy: 2X every 18 months is ~1% per week
    - Q: Would you add a feature that improved performance 20% if it would delay the chip 8 months?
  - Processors under Moore's Curve (arrive too late) fail spectacularly
    - E.g., Intel's Itanium, Sun's Millennium

Moore's Law in the Future



- Won't last forever, approaching physical limits
  - But betting against it has proved foolish in the past
  - Likely to "slow" rather than stop abruptly

- Transistor count will likely continue to scale
  - "Die stacking" is on the cusp of becoming main stream
  - Uses the third dimension to increase transistor count

- But transistor performance scaling?
  - Running into physical limits
  - Example: gate oxide is less than 10 silicon atoms thick!
    - Can't decrease it much further
  - Power is becoming a limiting factor (next)

Power/Energy: Increasingly Important

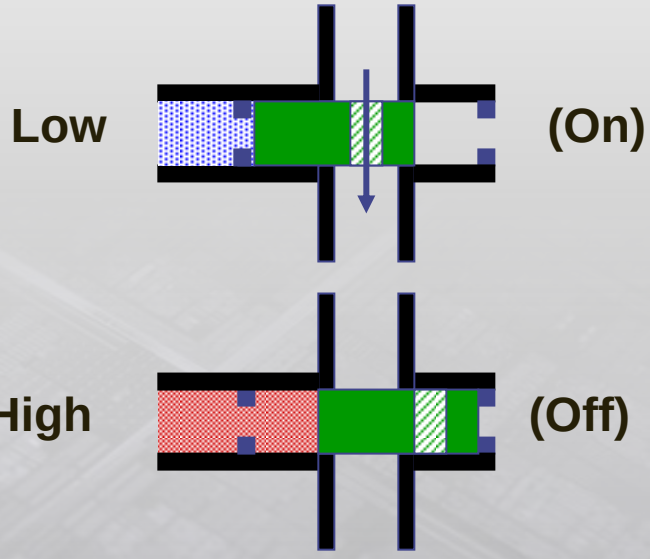
- **Battery life** for mobile devices
  - Laptops, phones, cameras
- **Tolerable temperature** for devices without active cooling
  - Power means temperature, active cooling means **cost**
  - No room for a fan in a cell phone, no market for a hot cell phone
- **Electric bill** for compute/data centers
  - Pay for power twice: once in, once out (to cool)
- **Environmental concerns**
  - "Computers" account for growing fraction of energy consumption

Energy & Power

- **Energy**: measured in Joules or Watt-seconds
  - Total amount of energy stored/used
  - Battery life, electric bill, environmental impact
  - Instructions per Joule (car analogy: miles per gallon)
- **Power**: energy per unit time (measured in Watts)
  - Related to "performance" (which is also a "per unit time" metric)
  - Power impacts power supply and cooling requirements (cost)
    - Power-density (Watt/mm^2): important related metric
  - Peak power vs average power
    - E.g., camera, power "spikes" when you actually take a picture
  - Joules per second (car analogy: gallons per hour)
- Two sources:
  - **Dynamic power**: active switching of transistors
  - **Static power**: leakage of transistors even while inactive

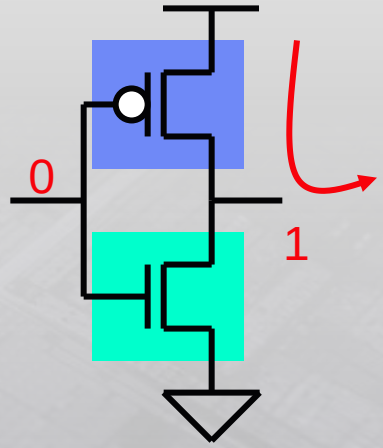
Recall: Tech. Basis of Transistor Speed

- Physics 101: delay through an electrical component ~ **RC**
  - **Resistance (R)**
    - Slows rate of charge flow
    - **Analogy: the friction of air flowing through a tube**
  - **Capacitance (C)**
    - Stores charge
    - **Analogy: volume of tubes**
  - **Voltage (V)**
    - Electrical pressure
    - **Analogy: compressed air pressure**
  - **Threshold Voltage (V_t)**
    - Voltage at which a transistor turns "on"
    - **Analogy: pressure at which valve switches**
  - Switching time ~ to (R * C) / (**V – V_t**)
    - **Analogy: the higher the pressure, the faster it switches**



Dynamic Power

- **Dynamic power ($P_{dynamic}$)**: aka switching or active power
  - Energy to switch a gate (0 to 1, 1 to 0)
  - Each gate has capacitance (C)
    - Charge stored is ~ C * V
    - Energy to charge/discharge a capacitor is ~ to QV which is ~$C * V^2$



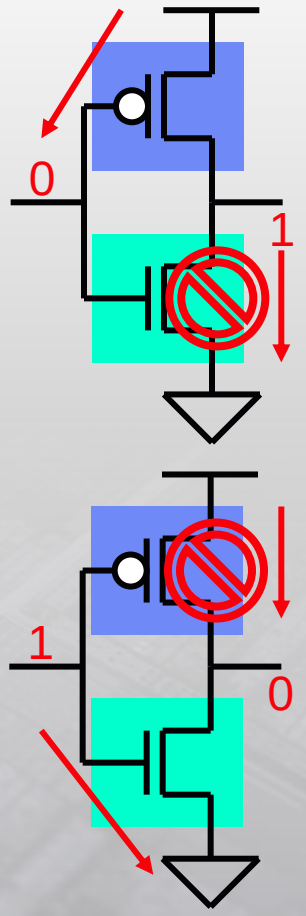
- **$P_{dynamic}$ ~ N * C * V^2 * f * A**
  - N: number of transistors
  - C: capacitance per transistor (size of transistors)
  - V: voltage (supply voltage for gate)
  - f: frequency (transistor switching freq. is ~ to clock freq.)
  - A: activity factor (not all transistors may switch this cycle)

Reducing Dynamic Power

- Target each component: $P_{dynamic} \sim N * C * V^2 * f * A$
- **Reduce number of transistors** (N)
  - Use fewer transistors/gates
- **Reduce capacitance** (C)
  - Smaller transistors (Moore's law)
- **Reduce voltage** (V)
  - Quadratic reduction in energy consumption!
  - But also slows transistors (transistor speed is ~ to V)
- **Reduce frequency** (f)
  - Slower clock frequency (reduces power but not energy) ***Why?***
- **Reduce activity** (A)
  - "Clock gating" disable clocks to unused parts of chip
  - Don't switch gates unnecessarily

Static Power

- **Static power (P_{static})**: aka idle or leakage power
  - Transistors don't turn off all the way
  - Transistors "leak"
    - Analogy: valves are not perfectly sealed
  - $P_{static} \sim N * V * e^{-Vt}$
  - N: number of transistors
  - V: voltage
  - **V_t (threshold voltage)**: voltage at which transistor conducts (begins to switch)
- Switching speed vs leakage trade-off
- The lower the **V_t**:
  - Faster transistors (linear)
    - Transistor speed ~ to $V - V_T$
  - Leakier transistors (exponential)



Reducing Static Power

- Target each component: $P_{static} \sim N * V * e^{-Vt}$
- **Reduce number of transistors** (N)
  - Use fewer transistors/gates
- **Reduce voltage** (V)
  - Linear reduction in static energy consumption
  - But also slows transistors (transistor speed is ~ to V)
- **Disable transistors** (also targets N)
  - "Power gating" disable power to unused parts (long latency to power up)
  - Power down units (or entire cores) not being used
- **Dual V_t** – use a mixture of high and low V_t transistors
  - Use slow, low-leak transistors in SRAM arrays
  - Requires extra fabrication steps (cost)
- **Low-leakage transistors**
  - High-K/Metal-Gates in Intel's 45nm process
- Note: reducing frequency can actually hurt static energy. ***Why?***

Dynamic Voltage/Frequency Scaling

- **Dynamically trade-off power for performance**
  - Change the voltage and frequency at runtime
  - Under control of operating system
- Recall: $P_{dynamic} \sim N * C * \mathbf{V^2} * \mathbf{f} * A$
  - Because frequency ~ to V…
  - $P_{dynamic}$ ~ to $\mathbf{V^3}$
- Reduce both V and f linearly
  - Cubic decrease in dynamic power
  - Linear decrease in performance
    - Thus, only about quadratic in energy
  - Linear decrease in static power
    - Thus, only modest static energy improvement
- Newer chips can do this on a per-core basis

Dynamic Voltage/Frequency Scaling

	Mobile PentiumIII "**SpeedStep**"	Transmeta 5400 "LongRun"	Intel X-Scale (StrongARM2)
f (MHz)	300–1000 (step=50)	200–700 (step=33)	50–800 (step=50)
V (V)	0.9–1.7 (step=0.1)	1.1–1.6V (cont)	0.7–1.65 (cont)
High-speed	3400MIPS @ 34W	1600MIPS @ 2W	800MIPS @ 0.9W
Low-power	1100MIPS @ 4.5W	300MIPS @ 0.25W	62MIPS @ 0.01W

- Dynamic voltage/frequency scaling
  - **Favors parallelism**
- Example: Intel Xscale
  - 1 GHz → 200 MHz reduces energy used by 30x
    - But around 5x slower
  - 5 x 200 MHz in parallel, use **1/6th the energy**
  - Power is driving the trend toward multi-core

Moore's Effect on Power

+ Moore's Law reduces power/transistor…
  - Reduced sizes and surface areas reduce capacitance (C)
– …but increases power density and total power
  - By increasing transistors/area and total transistors
  - Faster transistors → higher frequency → more power
  - Thermal cycle: hotter transistors leak more
- What to do? Reduce voltage (V)
  + Reduces dynamic power quadratically, static power linearly
    - Already happening: 486 (5V) → Core2 (1.1V)
  - Trade-off: reducing V means either…
    – Keeping V_t the same and reducing frequency (F)
    – Lowering V_t and increasing leakage exponentially
  - Pick your poison … or not: new techniques like high-K

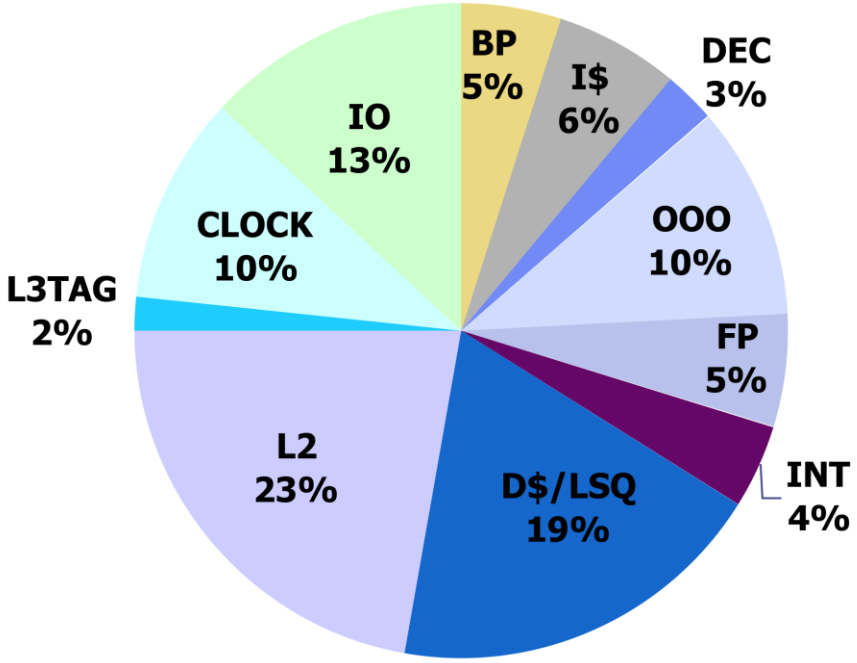
Trends in Power

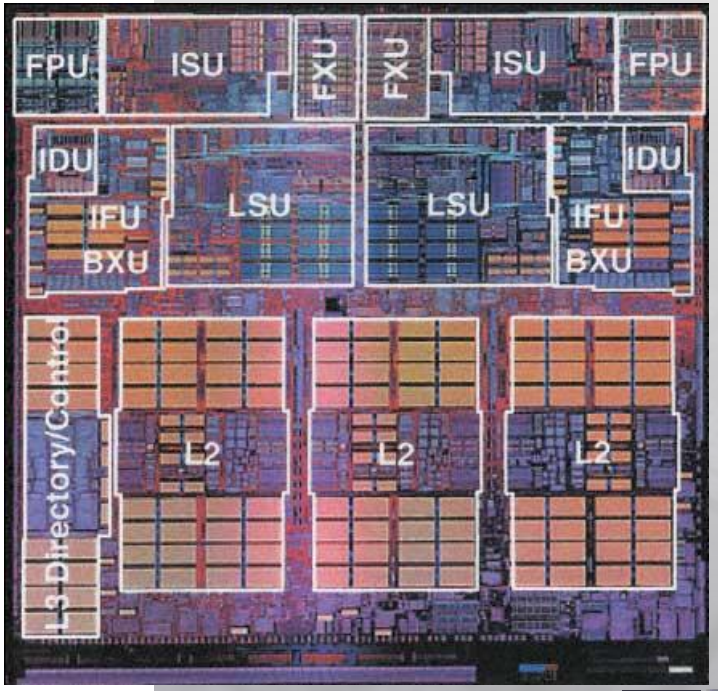
	386	486	Pentium	PentiumII	Pentium4	Core2
Year	1985	1989	1993	1998	2001	2006
Technode (nm)	1500	800	350	180	130	65
Transistors (M)	0.3	1.2	3.1	5.5	42	291
Voltage (V)	**5**	**5**	**3.3**	**2.9**	**1.7**	**1.1**
Clock (MHz)	16	25	66	200	1500	3000
Power (W)	1	5	16	35	**80**	**75**
Peak MIPS	6	25	132	600	4500	24000
MIPS/W	6	5	8	17	56	320

- Supply voltage decreasing over time
- Emphasis on power starting around 2000
  - Resulting in slower frequency increases

Processor Power Breakdown

- Power breakdown for IBM POWER4
  - Two 4-way superscalar, 2-way multi-threaded cores, 1.5MB L2
  - Big power components are L2, D$, out-of-order logic, clock, I/O
  - Implications on complicated versus simple cores





Implications on Software

- Software-controlled dynamic voltage/frequency scaling
  - OS? Application?
  - Example: video decoding
    - Too high a frequency – wasted energy (battery life)
    - Too low a frequency – quality of video suffers
- Managing low-power modes
  - Don't want to "wake up" the processor every millisecond
- Tuning software
  - Faster algorithms can be converted to lower-power algorithms
  - Via dynamic voltage/frequency scaling
- Exploiting parallelism

This Unit

- What is a computer and what is computer architecture
- Forces that shape computer architecture
  - Applications
  - Semiconductor technology
- Evaluation metrics: parameters and technology basis
  - Cost
  - Performance
  - Power
  - Reliability

Reliability

- **Mean Time Between Failures (MTBF)**
  - How long before you have to reboot or buy a new one

- CPU reliability small in grand scheme
  - Software most unreliable component in a system
    - Much more difficult to specify & test
    - Much more of it
  - Most unreliable hardware component … disk
    - Subject to mechanical wear

Moore's Bad Effect on Reliability

- CMOS devices: CPU and memory
  - Historically almost perfectly reliable
  - Moore has made them less reliable over time

- Two sources of electrical faults
  - Energetic particle strikes (from sun)
    - Randomly charge nodes, cause bits to flip, **transient**
  - Electro-migration: change in electrical interfaces/properties
    - Temperature-driven, happens gradually, **permanent**

- Large, high-energy transistors are immune to these effects
  - Scaling makes node energy closer to particle energy
  - Scaling increases power-density which increases temperature
  - Memory (DRAM) was hit first: denser, smaller devices than SRAM

Moore's Good Effect on Reliability

- The key to providing reliability is **redundancy**
  - The same scaling that makes devices less reliable…
  - Also increase device density to enable redundancy

- Classic example
  - Error correcting code (ECC) for DRAM
  - ECC also starting to appear for caches (Power7 in L3)

- Today's big open questions
  - Can we protect logic?
  - Can architectural techniques help hardware reliability?
  - Can architectural techniques help with software reliability?

Summary: A Global Look at Moore

- Device scaling (Moore's Law)
  - + Increases performance
    - Reduces transistor/wire delay
    - Gives us more transistors with which to reduce CPI
  - + Reduces local power consumption
    - – Which is quickly undone by increased integration
    - – Aggravates power-density and temperature problems
  - – Aggravates reliability problem
    - + But gives us the transistors to solve it via redundancy
  - + Reduces unit cost
    - – But increases startup cost
- Will we fall off Moore's Cliff? (for real, this time?)
  - What's next: nanotubes, quantum-dots, optical, spin-tronics, DNA?

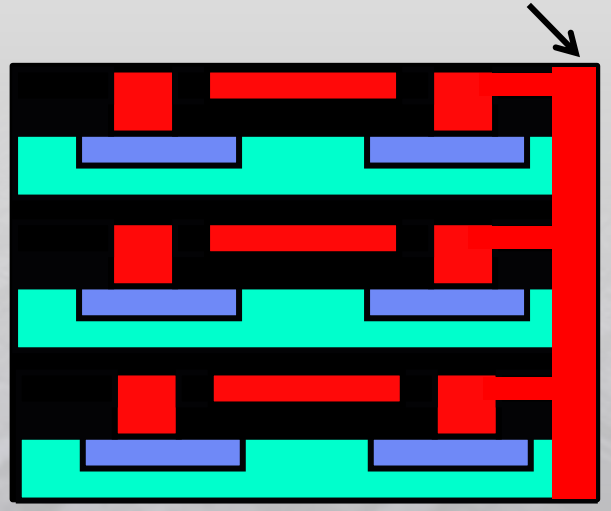
More than Moore

- Before physical limits: IC fab Cost-wall
  - Last 32nm: 2000 M$

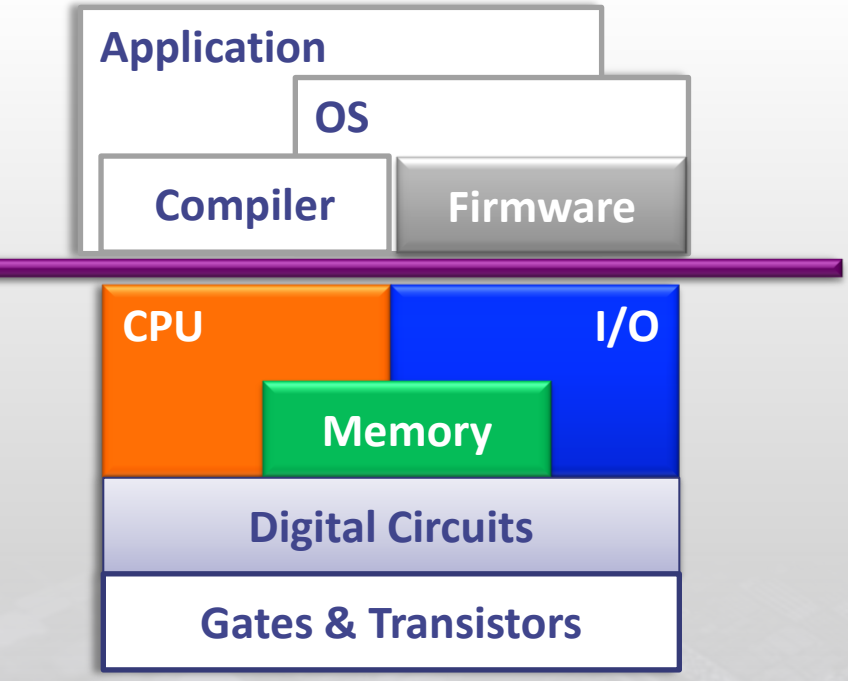
Next 22 nm : 8000 M$

- 3D-Stacking
  - Is happening now!
  - Samsung is selling FLASH 3D stacked
  - Next big thing

Through Silicon Via (TSV)



A Computer Archtecture Picture



- Mostly about micro-architecture
- Mostly about CPU/Memory
- Mostly about general-purpose
- Mostly about performance
- **We'll still only scratch the surface**

Acknowledgments

- Slides developed by Amir Roth of University of Pennsylvania with sources that included University of Wisconsin slides by Mark Hill, Guri Sohi, Jim Smith, and David Wood.
- Slides enhanced by Milo Martin and Mark Hill with sources that included Profs. Asanovic, Falsafi, Hoe, Lipasti, Shen, Smith, Sohi, Vijaykumar, and Wood
- Slides re-enhanced by V.Puente of Universidad de Cantabria