

CIS 501

Computer Architecture

Unit 1: Technology

Slides originally developed by Amir Roth with contributions by Milo Martin at University of Pennsylvania with sources that included University of Wisconsin slides by Mark Hill, Guri Sohi, Jim Smith, and David Wood.

Readings

- H+P
 - Chapters 1
- Paper
 - G. Moore, "Cramming More Components onto Integrated Circuits"

This Unit

- Technology basis
 - Transistors
 - Transistor scaling (Moore's Law)
- The metrics
 - Transistor speed
 - Cost
 - Power
 - Reliability
- How do these change over time (driven by Moore's Law)?
- All roads lead to multi-core

Discussion of Moore's Paper

- Notes:

Recap: What is Computer Architecture?

- Design of interfaces and implementations...
- Under constantly changing set of external forces...
 - **Applications**: change from above
 - **Technology**: changes from below
 - **Inertia**: resists changing all levels of system at once
- To satisfy different constraints
 - CIS 501 mostly about **performance**
 - Cost
 - Power
 - Reliability
- Iterative process driven by **empirical evaluation**
- The art/science of tradeoffs

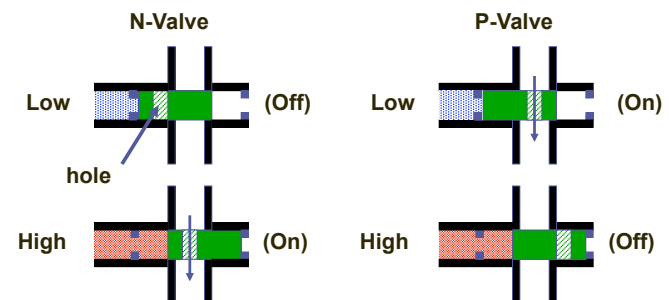
Review: Processor Performance

- Programs consist of simple operations (instructions)
 - Add two numbers, fetch data value from memory, etc.
- Program runtime = "seconds per program" =
 $(\text{instructions/program}) * (\text{cycles/instruction}) * (\text{seconds/cycle})$
- **Instructions per program**: "dynamic instruction count"
 - Runtime count of instructions executed by the program
 - Determined by program, compiler, instruction set architecture (ISA)
- **Cycles per instruction**: "CPI" (typical range: 2 to 0.5)
 - On average, how many *cycles* does an instruction take to execute?
 - Determined by program, compiler, ISA, micro-architecture
- **Seconds per cycle**: clock period, length of each cycle
 - Inverse metric: cycles per second (Hertz) or cycles per ns (Ghz)
 - Determined by micro-architecture, **technology parameters**
- This unit: transistors & semiconductor technology

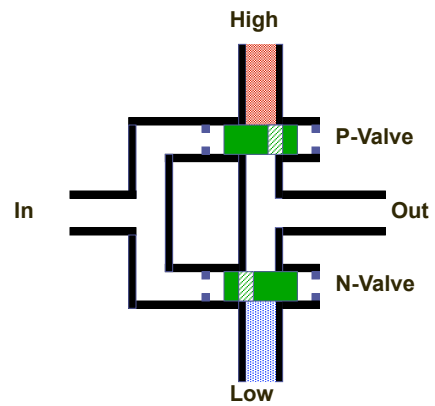
The Transistor

A Transistor Analogy: Computing with Air

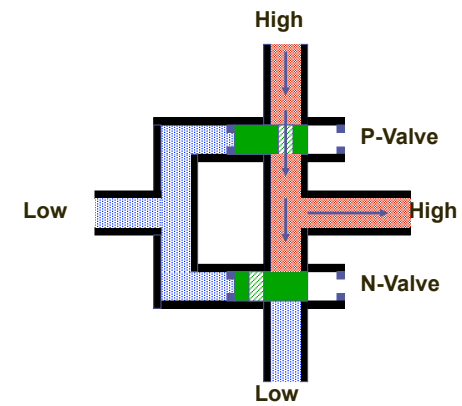
- Use air pressure to encode values
 - High pressure represents a "1" (blow)
 - Low pressure represents a "0" (suck)
- Valve can allow or disallow the flow of air
 - Two types of valves



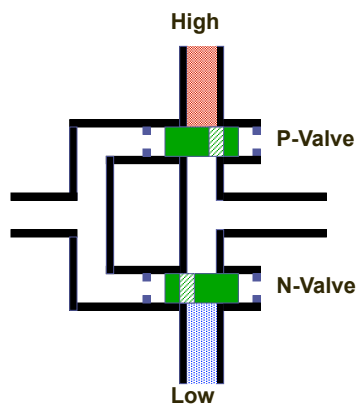
Pressure Inverter



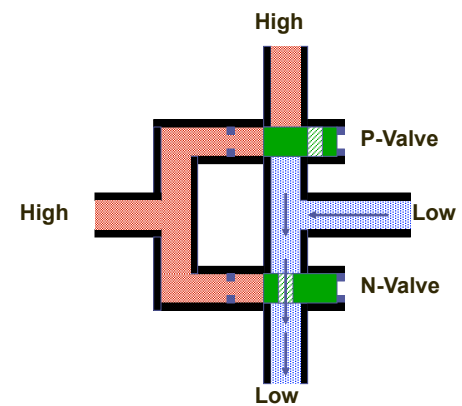
Pressure Inverter (Low to High)



Pressure Inverter



Pressure Inverter (High to Low)

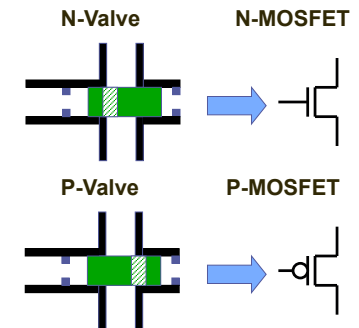


Analogy Explained

- Pressure differential → electrical potential (voltage)
 - Air molecules → electrons
 - Pressure (molecules per volume) → voltage
 - 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 → transistor
 - The transistor: one of the century's most important inventions

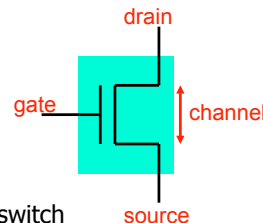
Transistors as Switches

- Two types
 - N-type
 - P-type
- Properties
 - Solid state (no moving parts)
 - Reliable (low failure rate)
 - Small (45nm channel length)
 - Fast (<0.1ns switch latency)



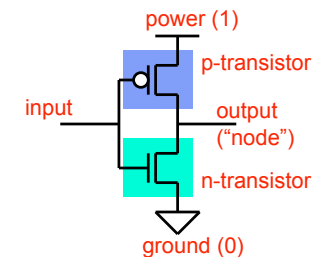
Semiconductor Technology

- Basic technology element: **MOSFET**
 - The invention of 20th century?
 - **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
 - An electrical "switch"
- **Channel length**: characteristic parameter (short → fast)
 - Aka "feature size" or "technology"
 - Currently: 0.045 micron (μm), 45 nanometers (nm)
 - Continued miniaturization (scaling) known as "**Moore's Law**"
 - Won't last forever, physical limits approaching (or are they?)



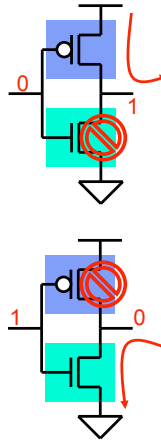
Complementary MOS (CMOS)

- 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 (i.e., gates)
 - And some non-gate elements too (important example: RAMs)



Basic CMOS Logic Gate

- **Inverter:** NOT gate
 - One p-transistor, one n-transistor
 - Basic operation
 - Input = 0
 - P-transistor closed, n-transistor open
 - Power charges output (1)
 - Input = 1
 - P-transistor open, n-transistor closed
 - Output discharges to ground (0)

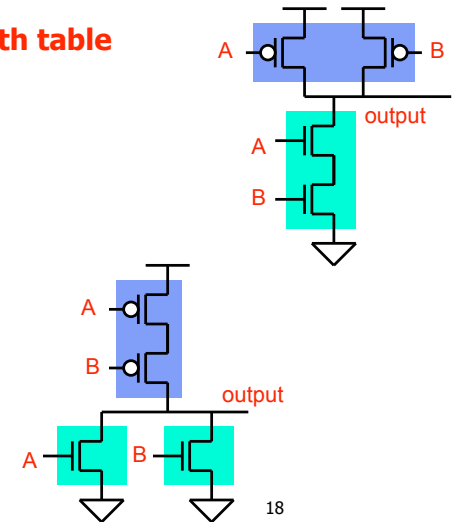


Another CMOS Logic Example

- What is this? Look at **truth table**

- 0, 0 → 1
- 0, 1 → 1
- 1, 0 → 1
- 1, 1 → 0
- Result: **NAND** (NOT AND)
- NAND is "universal"

- What function is this?



Transistor Speed, Power, and Reliability

- Transistor characteristics and scaling impact:
 - Switching speed (clock frequency)
 - Power/energy
 - Reliability
- "Undergrad" gate delay model for **architecture**
 - Each Not, NAND, NOR, AND, OR gate has delay of "1"
 - Reality is not so simple
- But first, how are these transistors manufactured?
 - First-order impact: **cost**

Cost

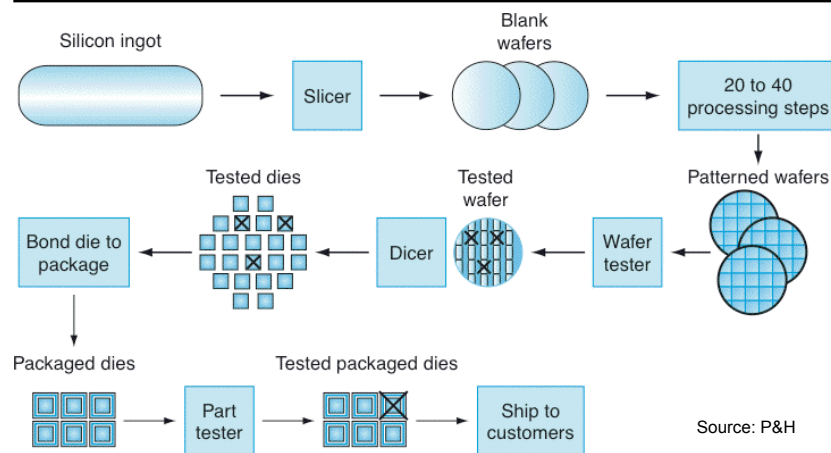
Cost

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

	Desktop	Laptop	Netbook	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, storage, software			

- We are concerned about chip cost
 - **Unit cost:** costs to manufacture individual chips
 - **Startup cost:** cost to design chip, build the manufacturing facility

Manufacturing Steps

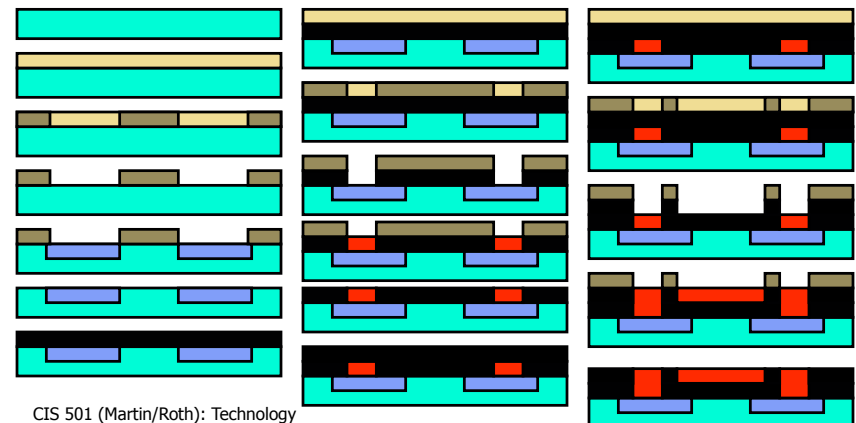


Cost versus Price

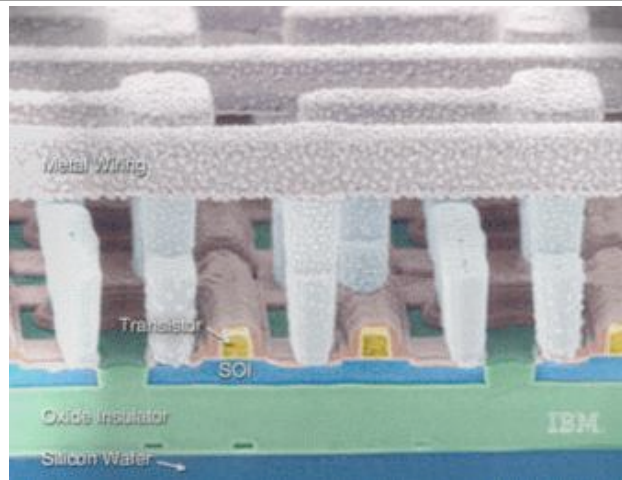
- **Cost:** cost to manufacturer, cost to produce
- What is the relationship of cost to price?
 - Complicated, has to do with volume and competition
- **Commodity:** high-volume, un-differentiated, un-branded
 - "Un-differentiated": copper is copper, wheat is wheat
 - "Un-branded": consumers aren't allied to manufacturer brand
 - Commodity prices tracks costs closely
 - Example: DRAM (used for main memory) is a commodity
 - Do you even know who manufactures DRAM?
- Microprocessors are not commodities
 - Specialization, compatibility, different cost/performance/power
 - Complex relationship between price and cost

Manufacturing Steps

- Multi-step photo-/electro-chemical process
 - More steps, higher unit cost
- + Fixed cost mass production (\$1 million or more)

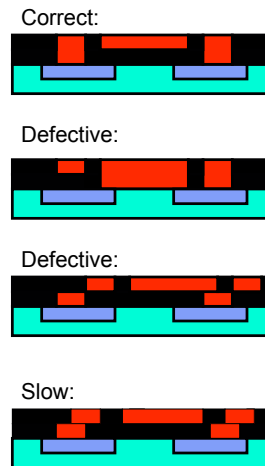


Transistors and Wires



From slides © Krste Asanović, MIT

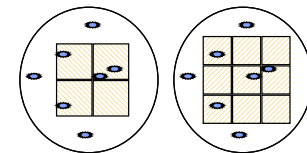
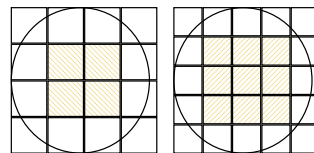
Manufacturing Defects



- 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
 - Can substantially improve yield

Unit Cost: Integrated Circuit (IC)

- Chips built in multi-step chemical processes on **wafers**
 - Cost / wafer is constant, $f(\text{wafer size, number of steps})$
- Chip (die) cost is related to **area**
 - Larger chips means fewer of them
- Cost is more than linear in area
 - Why? random defects
 - Larger chips means fewer working ones
 - Chip cost \sim chip area $^\alpha$
 - $\alpha = 2$ to 3



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

Additional Unit Cost

- After manufacturing, there are additional unit costs
 - Testing: how do you know chip is working?
 - Packaging: high-performance packages are expensive
 - Determined by maximum operating temperature
 - And number of external pins (off-chip bandwidth)
 - Burn-in: stress test chip (detects unreliability chips early)
 - Re-testing: how do you know packaging/burn-in didn't damage chip?

Fixed Costs

- For new chip design
 - Design & verification: ~\$100M (500 person-years @ \$200K per)
 - Amortized over “proliferations”, e.g., Xeon/Celeron cache variants
- For new (smaller) technology generation
 - ~\$3B for a new fab
 - Amortized over multiple designs
 - Amortized by “rent” from companies that don’t fab themselves
- Moore’s Law generally increases startup cost
 - More expensive fabrication equipment
 - More complex chips take longer to design and verify

Moore’s Effect on Cost



- Mixed impact on unit integrated circuit cost
 - + Either lower cost for same functionality...
 - + Or same cost for more functionality
 - Difficult to achieve high yields
- Increases startup cost
 - More expensive fabrication equipment
 - Takes longer to design, verify, and test chips
- Process variation across chip increasing
 - Some transistors slow, some fast
 - Increasingly active research area: dealing with this problem

All Roads Lead To Multi-Core



- + Multi-cores reduce unit costs
 - Higher yield than same-area single-cores
 - Why? Defect on one of the cores? Sell remaining cores for less
 - IBM manufactures CBE (“cell processor”) with eight cores
 - But PlayStation3 software is written for seven cores
 - Yield for eight working cores is too low
 - Sun manufactures Niagaras (UltraSparc T1) with eight cores
 - Also sells six- and four- core versions (for less)
- + Multi-cores can reduce design costs too
 - Replicate existing designs rather than re-design larger single-cores

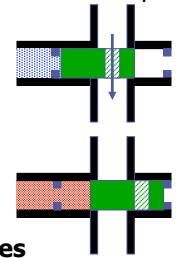
Transistor Speed

Technology Basis of Transistor Speed

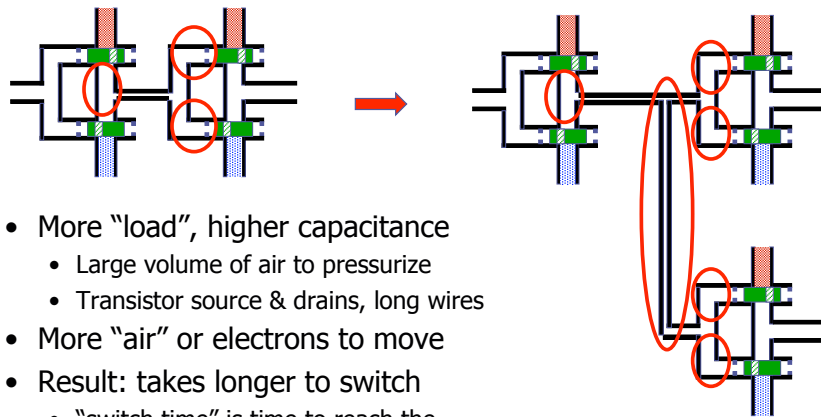
- Physics 101: delay through an electrical component $\sim RC$
 - **Resistance (R)**  \sim length / cross-section area
 - Slows rate of charge flow,
 - **Capacitance (C)**  \sim length * area / distance-to-other-plate
 - Stores charge
 - **Voltage (V)**
 - Electrical pressure
 - **Threshold Voltage (V_t)**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

Technology Basis of Transistor Speed

- Physics 101: delay through an electrical component $\sim RC$
 - **Resistance (R)**  \sim length / cross-section area
 - Slows rate of charge flow,
 - **Analogy: the friction of air flowing through a tube**
 - **Capacitance (C)**  \sim length * area / distance-to-other-plate
 - 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 \sim to $(R * C) / (V - V_t)$
 - **Analogy: the higher the pressure, the faster it switches**



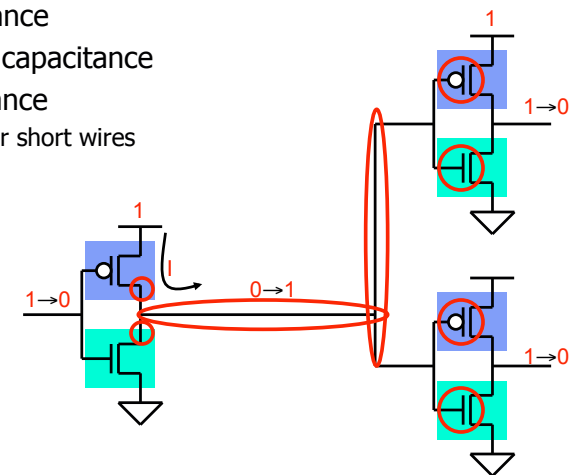
Capacitance Analogy: Air Capacity



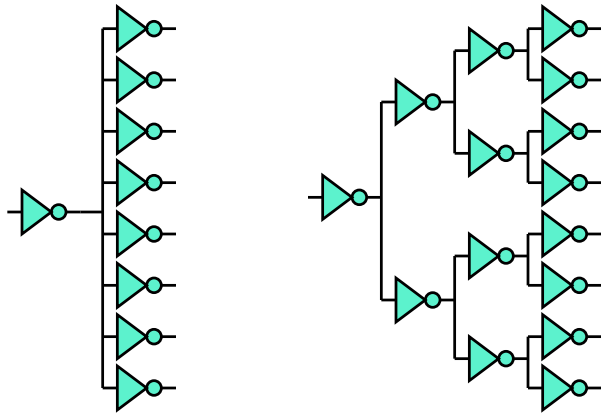
- More "load", higher capacitance
 - Large volume of air to pressurize
 - Transistor source & drains, long wires
- More "air" or electrons to move
- Result: takes longer to switch
 - "switch time" is time to reach the threshold pressure/voltage
- The "fan-out" of the device impacts its switching speed

Capacitance

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



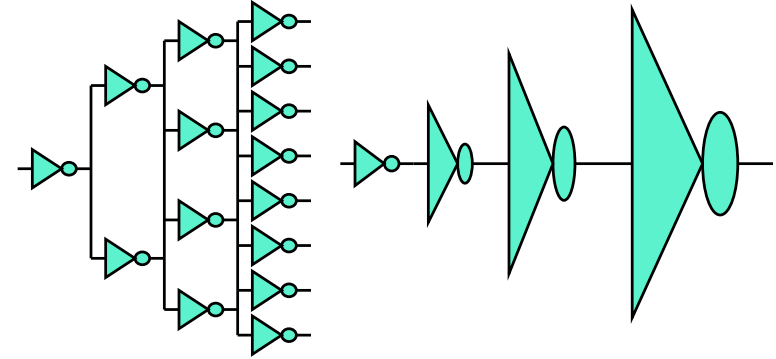
Which is faster? Why?



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

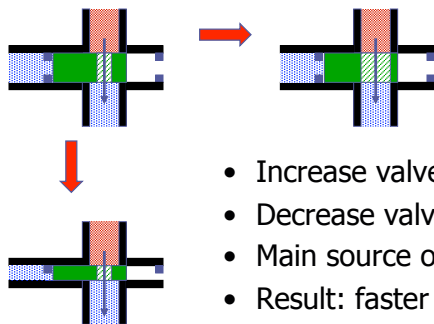
Transistor Width

- “Wider” transistors have lower resistance, more drive
 - Specified per-device

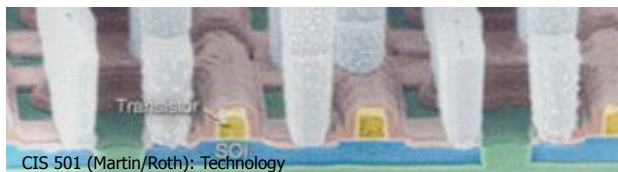


- Useful for driving large “loads” like long or off-chip wires

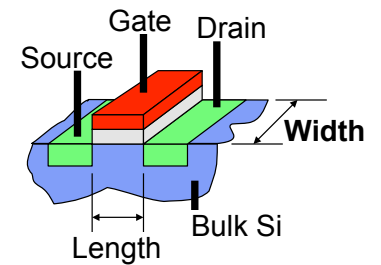
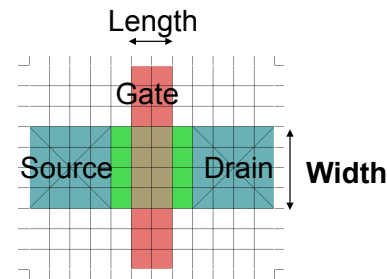
Trans. Resistance Analogy: Valve Friction



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



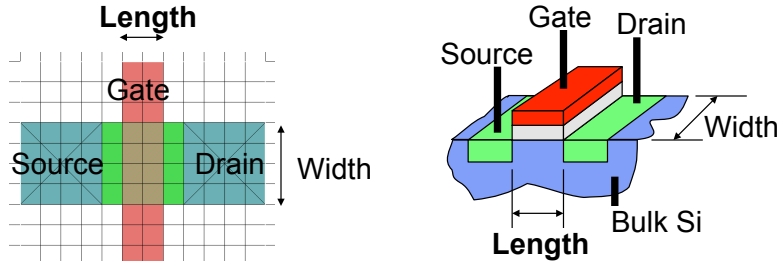
Transistor Geometry: Width



Diagrams © Krste Asanovic, MIT

- **Transistor width**, set by designer for each transistor
- Wider transistors:
 - **Lower resistance** of channel (increases drive strength) – good!
 - But, **increases capacitance** of gate/source/drain – bad!
- Result: set width to balance these conflicting effects

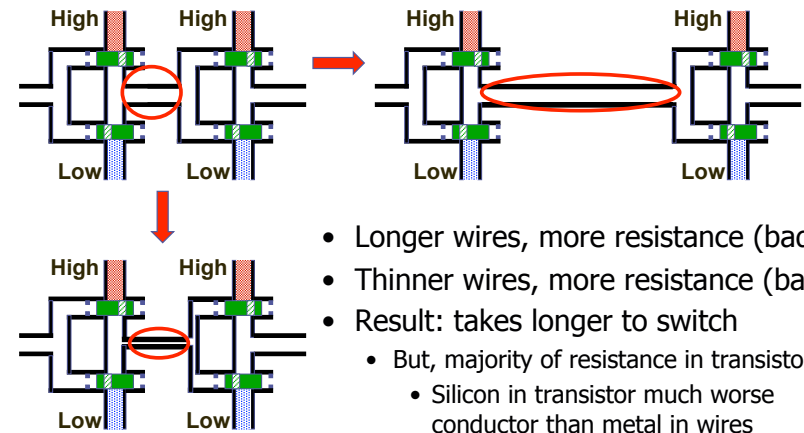
Transistor Geometry: Length & Scaling



Diagrams © Krste Asanovic, MIT

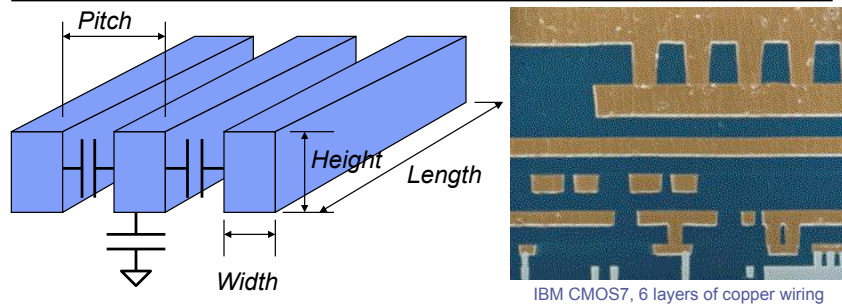
- **Transistor length:** characteristic of “process generation”
 - 45nm refers to the transistor gate length, same for all transistors
- Shrink transistor length:
 - Lower resistance of channel (shorter) – good!
 - Lower gate/source/drain capacitance – good!
- Result: switching speed improves linearly as gate length shrinks

Wire Resistance Analogy: Tube Friction



- Longer wires, more resistance (bad)
- Thinner wires, more resistance (bad)
- 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



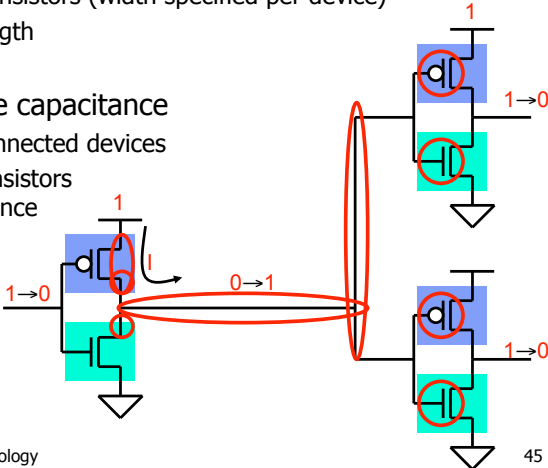
IBM CMOS7, 6 layers of copper wiring

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

Increasing Problem: Wire Delay

- RC Delay of wires
 - Resistance proportional to: $\text{resistivity} \times \text{length} / (\text{cross section})$
 - Wires with smaller cross section have higher resistance
 - Resistivity (type of metal, copper vs aluminum)
 - Capacitance proportional to length
 - And wire spacing (closer wires have large capacitance)
 - Permittivity or “dielectric constant” (of material between wires)
- Result: delay of a wire is quadratic in length
 - Insert “inverter” repeaters for long wires
 - Why? 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

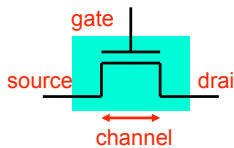
RC Delay Model Ramifications

- 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
- 

CIS 501 (Martin/Roth): Technology

45

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

CIS 501 (Martin/Roth): Technology

46

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)
- Reduces cost per transistor
- More transistors can increase performance
 - Job of a computer architect: use the ever-increasing number of transistors
 - Examples: caches, exploiting parallelism at all levels

CIS 501 (Martin/Roth): Technology

47

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

CIS 501 (Martin/Roth): Technology

48

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
 - "If something must eventually stop, it can't go on forever"
 - 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

Power/Energy Are Increasingly Important

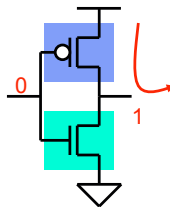
- **Battery life** for mobile devices
 - Laptops, phones, cameras
- **Tolerable temperature** for devices without active cooling
 - Power means temperature, active cooling means **cost**
 - No room for a fan in a cell phone, no market for a hot cell phone
- **Electric bill** for compute/data centers
 - 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
 - Joules per Instruction (car analogy: gallons per mile)
- **Power**: energy per unit time (measured in Watts)
 - Joules per second (car analogy: gallons per hour)
 - Related to "performance" (which is also a "per unit time" metric)
 - Power impacts power supply and cooling requirements (cost)
 - Power-density (Watt/mm²): important related metric
 - Peak power vs average power
 - E.g., camera, power "spikes" when you actually take a picture
- Two sources:
 - **Dynamic power**: active switching of transistors
 - **Static power**: leakage of transistors even while inactive

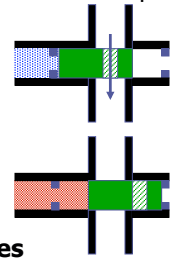
Dynamic Power

- **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 $\sim C * V$
 - Energy to charge/discharge a capacitor is $\sim C * V^2$
 - Time to charge/discharge a capacitor is \sim to V
 - Result: frequency \sim to V
 - **$P_{dynamic} \sim N * C * V^2 * f * A$**
 - N: number of transistors
 - C: capacitance per transistor (size of transistors)
 - V: voltage (supply voltage for gate)
 - f: frequency (transistor switching freq. is \sim to clock freq.)
 - A: activity factor (not all transistors may switch this cycle)



Recall: Tech. Basis of Transistor Speed

- Physics 101: delay through an electrical component $\sim RC$
 - **Resistance (R)** \sim length / cross-section area
 - Slows rate of charge flow,
 - **Analogy: the friction of air flowing through a tube**
 - **Capacitance (C)** \sim length * area / distance-to-other-plate
 - 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 \sim to $(R * C) / (V - V_t)$
 - **Analogy: the higher the pressure, the faster it switches**

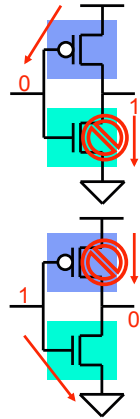


Reducing Dynamic Power

- 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 \sim 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: leaky valve
 - $P_{static} \sim N * V * e^{-Vt}$
 - 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 \sim to $V - V_T$
 - Leakier transistors (exponential)



57

CIS 501 (Martin/Roth): Technology

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 \sim 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?

CIS 501 (Martin/Roth): Technology

58

Continuation of Moore's Law

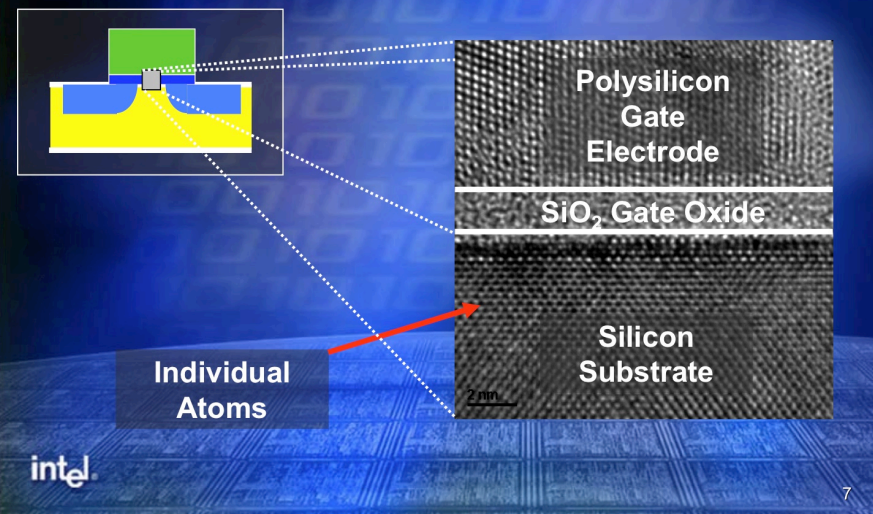
Process Name	P856	P858	Px60	P1262	P1264	P1266	P1268	P1270
1st Production	1997	1999	2001	2003	2005	2007	2009	2011
Process Generation	0.25 μ m	0.18 μ m	0.13 μ m	90 nm	65 nm	45 nm	32 nm	22 nm
Wafer Size (mm)	200	200	200/300	300	300	300	300	300
Inter-connect	Al	Al	Cu	Cu	Cu	Cu	Cu	?
Channel	Si	Si	Si	Strained Si	Strained Si	Strained Si	Strained Si	Strained Si
Gate dielectric	SiO ₂	SiO ₂	SiO ₂	SiO ₂	SiO ₂	High-k	High-k	High-k
Gate electrode	Poly-silicon	Poly-silicon	Poly-silicon	Poly-silicon	Poly-silicon	Metal	Metal	Metal

Introduction targeted at this time

Subject to change

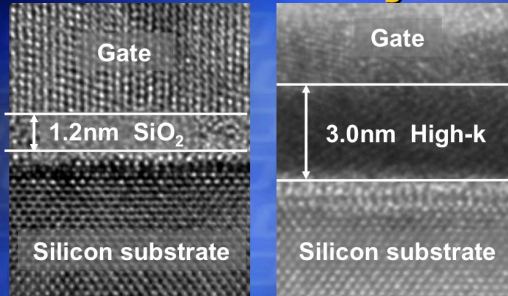
Intel found a solution for High-k and metal gate

Gate dielectric today is only a few molecular layers thick



7

High-k Dielectric reduces leakage substantially



Benefits compared to current process technologies

	High-k vs. SiO ₂	Benefit
Capacitance	60% greater	<i>Much faster transistors</i>
Gate dielectric leakage	> 100x reduction	<i>Far cooler</i>

intel.

10

Dynamic Voltage/Frequency Scaling

- **Dynamically trade-off power for performance**
 - Change the voltage and frequency at runtime
 - Under control of operating system
- Recall: $P_{\text{dynamic}} \sim N * C * V^2 * f * A$
 - Because frequency \sim to V ...
 - $P_{\text{dynamic}} \sim$ to V^3
- Reduce both V and f linearly
 - Cubic decrease in dynamic power
 - Linear decrease in performance (actually sub-linear)
 - 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

CIS 501 (Martin/Roth): Technology

62

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 \rightarrow 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

CIS 501 (Martin/Roth): Technology

63

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 \rightarrow higher frequency \rightarrow more power
 - Hotter transistors leak more (thermal runaway)
- What to do? Reduce voltage (V)
 - + Reduces dynamic power quadratically, static power linearly
 - Already happening: 486 (5V) \rightarrow 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 and dual- V_T

CIS 501 (Martin/Roth): Technology

64

Trends in Power

	386	486	Pentium	Pentium II	Pentium4	Core2	Core i7
Year	1985	1989	1993	1998	2001	2006	2009
Technode (nm)	1500	800	350	180	130	65	45
Transistors (M)	0.3	1.2	3.1	5.5	42	291	731
Voltage (V)	5	5	3.3	2.9	1.7	1.3	1.2
Clock (MHz)	16	25	66	200	1500	3000	3300
Power (W)	1	5	16	35	80	75	130
Peak MIPS	6	25	132	600	4500	24000	52800
MIPS/W	6	5	8	17	56	320	406

- Supply voltage decreasing over time
 - But “voltage scaling” is perhaps reaching its limits
- Emphasis on power starting around 2000
 - Resulting in slower frequency increases
 - Also note number of cores increasing (2 in Core 2, 4 in Core i7)

CIS 501 (Martin/Roth): Technology

65

Implications on Software

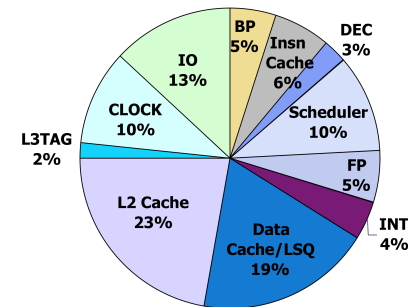
- Software-controlled dynamic voltage/frequency scaling
 - OS? Application?
 - Example: video decoding
 - Too high a frequency – wasted energy (battery life)
 - Too low a frequency – quality of video suffers
- Managing low-power modes
 - Don't want to “wake up” the processor every millisecond
- Tuning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

CIS 501 (Martin/Roth): Technology

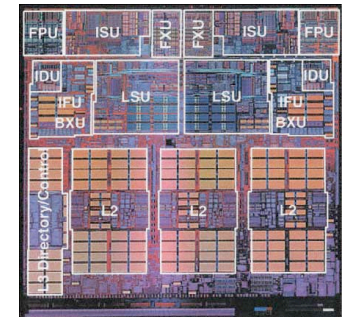
67

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, data cache, scheduler, clock, I/O
 - Implications on complicated versus simple cores



CIS 501 (Martin/Roth): Technology



66

Reliability

CIS 501 (Martin/Roth): Technology

68

Technology Basis for Reliability

- As transistors get smaller, they are less reliable
 - Wasn't a problem a few years ago, becoming a big problem
 - Small capacitance means fewer electrons represent 1 or 0
- **Transient faults**
 - A bit "flips" randomly, **temporarily**
 - Cosmic rays and such (more common at higher altitudes!)
 - Memory cells (especially memory) vulnerable today, logic soon
- **Permanent (hard) faults**
 - A gate or memory cell wears out, **breaks and stays broken**
 - Temperature & electromigration gradually deform components
- Solution for both: use **redundancy** to detect and tolerate

Memory Error Detection

- Idea: add extra state to memory to detect a bit flip
- **Parity**: simplest scheme
 - One extra bit, detects any single bit flip
 - Parity bit = $\text{XOR}(\text{data}_{N-1}, \dots, \text{data}_1, \text{data}_0)$
- Example:
 - $010101 \ 0 \wedge 1 \wedge 0 \wedge 1 \wedge 0 \wedge 1 = "1"$ so parity is "odd" (versus "even")
 - So, store "010101 **1**" in memory
 - When you read the data, and re-calculate the parity, say
 - 01**1**101 **1**, if the parity bit doesn't match, error detected
- Multiple bit errors? more redundancy can detect more

Aside: Memory Technology Families

- **SRAM**: "static" RAM
 - Used on processor chips (same transistors as used for "logic")
 - Storage implemented as 6 transistors per bit
 - An inverter pair (2 transistors each) + two control transistors
 - Optimized for speed first, then secondarily density and power
- **DRAM (volatile memory)**: "dynamic" RAM
 - Different manufacturing steps, not typically used on processor chips
 - Storage implemented as one capacitor + 1 transistor per bit
 - Optimized for density and cost
- **Flash (non-volatile memory)**:
 - Used for solid state storage
 - Slower than DRAM, but non-volatile
- Disk is also a "technology", but isn't transistor-based

Memory Error Detection

- What to do on a parity error?
- **Crash**
 - **Dead programs tell no lies**
 - Fail-stop is better than silent data corruption
 - Avoiding writing that "\$1m check"
- For user-level data, OS can kill just the program
 - Not the whole system, unless it was OS data
- Alternative: correct the error

SEC Error Correction Code (ECC)

- **SEC**: single-error correct (a hamming code)
- Example: Four data bits, three "code" bits
 - $d_1 d_2 d_3 d_4 c_1 c_2 c_3 \rightarrow c_1 c_2 d_1 c_3 d_2 d_3 d_4$
 - $c_1 = d_1 \wedge d_2 \wedge d_4$, $c_2 = d_1 \wedge d_3 \wedge d_4$, $c_3 = d_2 \wedge d_3 \wedge d_4$
 - Syndrome: $c_i \wedge c'_i = 0$? no error
 - Otherwise, then $c'_3 c'_2 c'_1$ points to flipped-bit
- Working example
 - Original data = 0110 $\rightarrow c_1 = 1, c_2 = 1, c_3 = 0$
 - Flip $d_2 = 0010 \rightarrow c'_1 = 0, c'_2 = 1, c'_3 = 1$
 - Syndrome = 101 (binary 5) \rightarrow 5th bit? D_2
 - Flip $c_2 \rightarrow c'_1 = 1, c'_2 = 0, c'_3 = 0$
 - Syndrome = 010 (binary 2) \rightarrow 2nd bit? c_2

SECDED Error Correction Code (ECC)

- **SECDED**: single error correct, double error detect
- Example: $D = 4 \rightarrow C = 4$
 - $d_1 d_2 d_3 d_4 c_1 c_2 c_3 \rightarrow c_1 c_2 d_1 c_3 d_2 d_3 d_4 c_4$
 - $c_4 = c_1 \wedge c_2 \wedge d_1 \wedge c_3 \wedge d_2 \wedge d_3 \wedge d_4$
 - Syndrome == 0 and $c'_4 == c_4 \rightarrow$ no error
 - Syndrome != 0 and $c'_4 != c_4 \rightarrow$ 1-bit error
 - Syndrome != 0 and $c'_4 == c_4 \rightarrow$ 2-bit error
 - Syndrome == 0 and $c'_4 != c_4 \rightarrow c_4$ error
 - **In general: $C = \log_2 D + 2$**
- Many machines today use 64-bit SECDED code
 - $C = 8$ (64bits + 8bits = 72bits, 12% overhead)
 - ChipKill - correct any aligned 4-bit error
 - If an entire memory chips dies, the system still works!

Moore's Bad Effect on Reliability

- Wasn't a problem until 5-10 years ago...
 - Except for transient-errors on chips in orbit (satellites)
 - ...a problem already and getting worse all the time
 - Transient faults:
 - Small (low charge) transistors are more easily flipped
 - Even low-energy particles can flip a bit now
 - Permanent faults:
 - Small transistors and wires deform and break more quickly
 - Higher temperatures accelerate the process
- Progression of transient faults
 - Memory (DRAM) was hit first: denser, smaller devices than SRAM
 - Then on-chip memory (SRAM)
 - Logic is starting to have problems...

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
- Examples
 - Error correcting code for memory (DRAM) and caches (SRAM)
 - Core-level redundancy: paired-execution, hot-spare, etc.
 - Intel's Core i7 (Nehalem) uses 8 transistor SRAM cells
 - Versus the standard 6 transistor cells
- Big open questions
 - Can we protect logic efficiently? (without 2x or 3x overhead)
 - Can architectural techniques help hardware reliability?
 - Can software techniques help?

Another Issue: Process Variability

- As transistors get smaller...
 - Small geometric variations have relatively larger impact
- Example: Gate oxide thickness
 - In Intel's 65nm process: only 1.2 nm, just a few molecules thick!
 - Small variation in gate oxide thickness impacts speed and energy
 - Too thick: slow transistor
 - Too thin: exponential increase in leakage (static power)
- Some parts of the chip slow than others (impacts yield)
- Complicates high-speed memory designs
- Limits circuit techniques ("dynamic" versus "static" circuits)
 - Intel's Nehalem (Core i7) moved to all static circuits

A Global Look at Moore

- Device scaling (Moore's Law)
 - + Reduces unit cost
 - But increases startup cost
 - + Increases performance
 - Reduces transistor/wire delay
 - Gives us more transistors with which to increase performance
 - + Reduces local power consumption
 - Which is quickly undone by increased integration, frequency
 - Aggravates power-density and temperature problems
 - Aggravates reliability problem
 - + But gives us the transistors to solve it via redundancy
- Will we fall off Moore's Cliff? (for real, this time?)
 - Difficult challenges, but \$\$\$ and smart people working on it
 - Example: 3D die stacking

Summary

Technology Summary

- Has a first-order impact on computer architecture
 - Cost (die area)
 - Performance (transistor delay, wire delay)
 - Power (static vs dynamic)
 - Reliability and variability
 - **All changing rapidly**
- Most significant trends for architects (and thus CIS501)
 - More and more transistors
 - What to do with them? → integration → **parallelism**
 - Logic is improving faster than memory & cross-chip wires
 - "Memory wall" → caches, more integration
 - Power, reliability, variability (more recent)
- This unit: a quick overview, just scratching the surface

} Rest of semester