CIS 371
Computer Organization and Design

Unit 12: (Low) Power and Energy

Power/Energy Are Increasingly Important

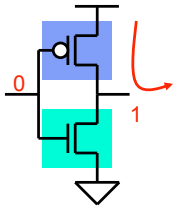
- **Battery life** for mobile devices
  - Laptops, phones, cameras
- **Tolerable temperature** for devices without active cooling
  - Power means temperature, active cooling means **cost**
  - No room for a fan in a cell phone, no market for a hot cell phone
- **Electric bill** for compute/data centers
  - 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
  - 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

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 C * V^2
    - Time to charge/discharge a capacitor is ~ to V
      - Result: frequency ~ to V
  - **$P_{dynamic}$ ~ 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)

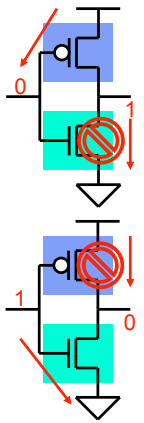


Reducing Dynamic Power

- 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"
  - $P_{static} \sim N * V * e^{-Vt}$
  - 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 higher the **V_t**:
  - Faster transistors (linear)
  - 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
- **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
- **Reduce voltage** (V)
  - Linear reduction in static energy consumption
  - But also slows transistors (transistor speed is ~ to V)
- **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 **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

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

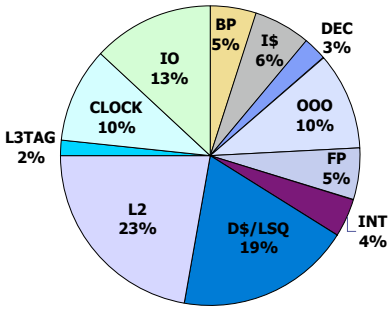
Trends in Power

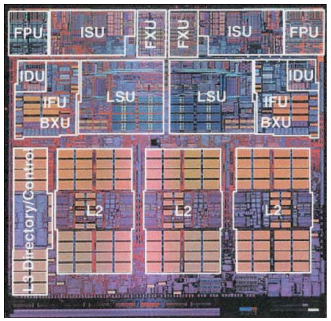
	386	486	Pentium	Pentium II	Pentium4	Core2	Core i7
Year	1985	1989	1993	1998	2001	2006	2009
Technode (nm)	1500	800	350	180	130	65	45
Transistors (M)	0.3	1.2	3.1	5.5	42	291	731
Voltage (V)	**5**	**5**	**3.3**	**2.9**	**1.7**	**1.3**	**1.2**
Clock (MHz)	16	25	66	200	1500	3000	3300
Power (W)	1	5	16	35	**80**	**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

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 out-of-order vs in-order





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