Lecture 24: Power-efficient Designs

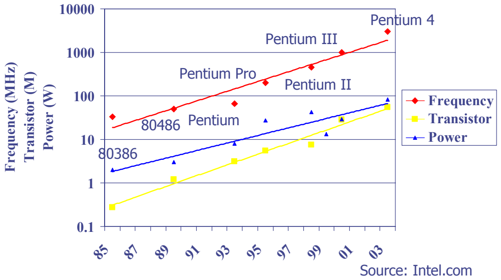
Dynamic and static power, processor power distribution, low power techniques in processor design, examples

Credits: Zhichun Zhu Thesis defense, HPCA'01 Low Power Tutorial, WRL Cacti Model

Importance of Low-power Designs

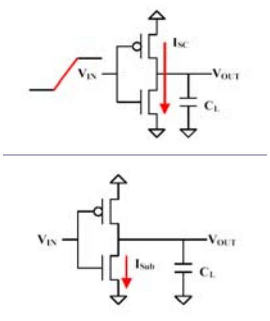
- Cost factor for high-end systems
  - High-end systems
    - Cooling and package cost
      - > 40 W: 1 W → $1
      - Air-cooled techniques: reaching limits
    - Electricity bill
    - Reliability
  - Desktop PCs consume around 10% power in US
- Usability of Portable systems:
  - Battery lifetime
- Restriction factor for high-performance server design
  - Power determines processor density

Processor Performance vs. Power Trends



Source: Intel.com

Dynamic vs. Static Power



- Dynamic:
  - Charge/discharge capacitors when switching between 0 and 1
  - Short-circuit currents on transitions
- Static (Leakage)
  - From sub-threshold currents

Sources of Power Consumption

- Dynamic (dominant) [Tutorial:HPCA-7]

$$P_{dync} = \frac{1}{2} C \cdot V^2 \cdot A \cdot f$$

- Static (2~5%) [Butts:MICRO-33]

$$P_{static} = N \cdot V \cdot k_{design} \cdot \widehat{I}_{leak}$$

C: capacitance, V: supply voltage, A: activity factor, f: clock rate
N: # transistors, k_{design}: design parameter, I_{leak}: leakage current

Importance of Low-power Architecture Designs

- Low power CMOS and logic designs alone can no longer solve all power problems.

$$P_{dync} = \frac{1}{2} C \cdot V^2 \cdot A \cdot f$$

$$\left.\begin{array}{r} V' = 0.7V \\ C' = 0.7 \times 2C \\ f' = 2f \end{array}\right\} \Rightarrow P'_{dync} = 1.4 P_{dync}$$

Low-power Techniques

- Physical (CMOS) level
- Circuit level
- Logic level
- **Architectural level**
- OS level
- Compiler level
- Algorithm/application level

Power-aware Architecture Designs

- Utilize low-power circuit techniques
- Exploit application characteristics
- Play an important role in low-power designs
  - Pentium III 800 MHz processor [CoolChip'00]
    - Scaled from Pentium Pro: 90 watts.
    - After architectural design and optimization: 22 watts.

Tradeoff between Performance and Power

- Objects for general-purpose system
  - Reduce power consumption without degrading performance
- Common solution
  - Access/activate resources only when necessary
- Question
  - When is necessary?

Metrics for Power-Performance Efficiency

- Performance (CPU time or <u>D</u>elay)

$$D = I \cdot CPI \cdot \frac{1}{f}$$

- Power consumption (P)
- Energy consumption (E)

$$E = P \cdot D$$

Metrics for Power-Performance Efficiency

- In most cases

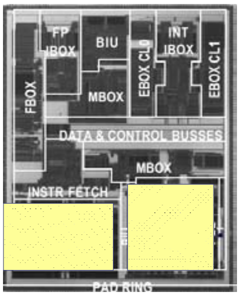
low power consumption ⇔ low performance

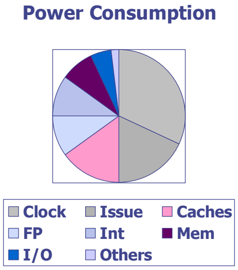
- $\downarrow f \Rightarrow \downarrow P \quad (P \propto f)$
- $\downarrow f \Rightarrow \uparrow D \quad (D \propto {}^{1}\!/_{f})$

- Energy-efficiency metric

$$EDP = E \cdot D = PD^{2}$$

Processor Power Distribution Example (Alpha 21264)





Source: CoolChip Tutorial

Low Power Processor Design

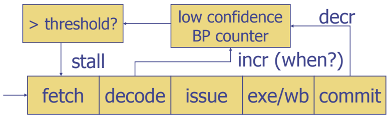
- Reduce power consumption of processor core
  - Voltage/frequency scaling: reduce supply voltage and/or frequency when processor is idle
  - Clock gating: disable clocks to inactive components
  - Pipeline gating: reduce mis-speculated instruction execution
  - Pipeline balancing: adjust effective pipeline ways for available IPC
  - Efficient issue logic: cluster structure, adjust effective issue queue size, no matching for ready entries, reducing tag matching entries

Low Power Memory Design

- Reduce power consumption of memory components
  - Banked or hierarchical register file
  - Sub-banked cache
  - Sequential access or way prediction caches
  - Dynamically adjusting cache size
  - Decay cache for reducing static power
  - Low power DRAM with deep sleeping modes: four modes in Rambus

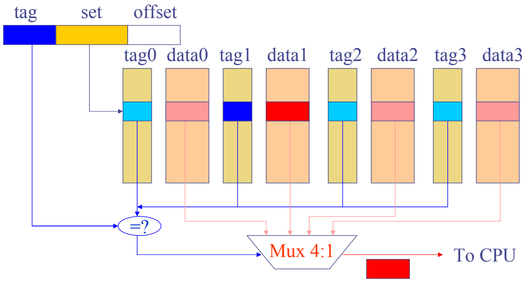
Pipeline Gating

- Mis-speculated instruction increase energy consumption, typically 16%-105% overhead
- Pipeline gating: stall fetching when confidence is low
- Prevent "bad" instructions from entering the pipeline: may reduce 38% of wrong inst



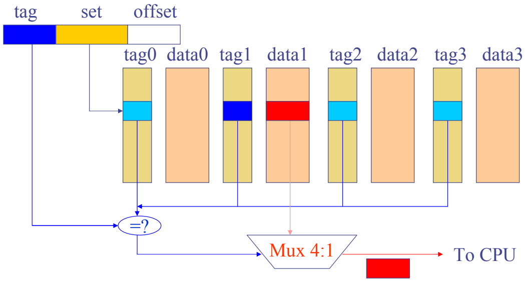
Pipeline gating: speculative control for energy reduction, isca 1998

Set Associative Cache



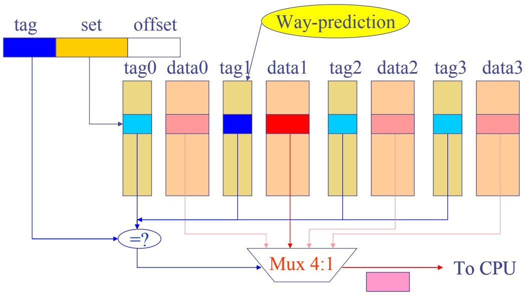
Power per access: 4T + 4D

Phased N-way Cache



Power per access: 4T + 1D
But access time increases

Way-prediction N-way Cache



Correct prediction: 1T + 1D

Low Power Server Design

- Low power considerations in supercomputing
  - Is high-performance processor the best choice?
  - IBM Blue Gene: 64K nodes with PowerPC 440 processors designed for low power
- Power management for high-performance servers
  - Meet performance with minimal active nodes

Power Evaluation Tools

- Processor
  - Wattch
    - Analytical
  - SimplePower
    - Analytical (e.g. cache)
    - Transition-sensitive (e.g. FU)
- Cache
  - CACTI
    - Analytical

Low Power Technique Summary

- Power is critical in processor design: cost and dependability
- Power distributions: clock, issue logic, cache, etc.
- Architectural approaches
  - scale voltage, frequency, and/or pipeline width with required performance
  - reduce mis-speculated execution, eliminate unnecessary cache accesses and data
  - Many others
- System approaches: high-performance by low power processors

Now low power is as important as performance