Energy-Performance Trade-offs in Processor Architecture and Circuit Design: A Marginal Cost Analysis

Omid Azizi
Aqeel Mahesri, Ben Lee, Sanjay Patel, Mark Horowitz
Stanford University, UIUC

ISCA 2010
June 21, 2010
The Power Problem

- Processor designs today are power-constrained
  - $V_{DD}$ has stopped scaling, so the problem will only get worse

[Graph showing power consumption over years for different processor models, with a power ceiling marked.]
We have to be careful with power consumption in designs
  Many design features offer performance, but come at a power cost

Question: How should you spend your power budget?
  What design features are worth including?
  How can we optimize designs for energy efficiency?

The New Design Objective: Design for Energy Efficiency
The Energy-Performance Design Space

- Every design can be plotted in the performance-energy space
  - We want designs on the energy-efficient frontier

![Graph showing the energy-performance design space with various processor benchmarks plotted. The graph is a scatter plot with energy per operation on the y-axis and performance on the x-axis. The energy-efficient frontier is highlighted.]
Optimizing for Energy Efficiency

- **Goal:** Find the processors on the efficient frontier

- **Study:** Consider a large part of the processor design space
  - High-level architectures
    - In-order vs out-of-order, single-issue vs dual-issue vs quad-issue, etc.
  - Micro-architectural design knobs
    - Cache sizes, pipeline depth, instruction window sizes, etc.
  - Circuit design
    - Gate sizing, circuit topology, circuit style, etc.
Outline

- Quick review of optimization and marginal costs

- Experimental Methodology
  - Modeling approach for performance and power
  - Integrated architecture-circuit optimization framework

- Results
  - Compare designs from a simple single-issue in-order core...
  - ...to an aggressive quad-issue out-of-order processor
Finding efficient designs is a trade-off analysis problem
- A design feature usually affects both performance and energy

To gauge efficiency of design choices, we use marginal costs
- Want those choices with the lowest cost per unit performance

Marginal Cost of \( x = \Delta E / \Delta P = \frac{\partial E}{\partial x} / \frac{\partial P}{\partial x} \)
- Energy cost of \( x \)
- Performance benefit of \( x \)

If we know marginal costs, then we can optimize a design
- “Buy” parameters with a low marginal cost, “sell” parameters with high cost
Current power modeling tools use fixed energy costs for circuits

But circuits can be designed in different ways

- Trade-off: faster circuits require more energy, slower circuits save energy

For true optimization, we need circuit-aware architectural models
Example: Simple In-order Processor

How big should I make my I-cache?
How fast should I run it?

How fast should I run my multiplier?

How big should I make my D-cache?
How fast should I run it?
Optimization Framework Overview

Benchmark App(s) → Simulate Random Designs → Fit Architecture Model → Architecture Circuit Link → Optimizer (GP Solver) → Energy Budget → Optimized Micro-Architecture

Circuit Tradeoffs Library:
- ADDER
- MULTIPLIER
- REG FILE
- I-CACHE
- …
Optimization Framework Overview

- **Step 1: Create Architectural Models**
  - Use statistical inference to capture a large design space
**Statistical Performance Modeling**

**TRADITIONAL**

PERFORMANCE MODELING & DESIGN OPTIMIZATION

- Design Optimization Loop
  - Architecture Configuration
  - Simulator
  - Performance Data Point
  - Evaluate Design

**STATISTICAL INFERENCE**

PERFORMANCE MODELING & DESIGN OPTIMIZATION

- Design Optimization Loop
  - Random Architecture Configurations
  - Simulator
  - Statistical Inference (Data Fit)
  - Analytical Performance Model
  - Evaluate Design
Optimization Framework Overview

- Step 2: Characterize Circuit Trade-offs

Circuit Tradeoffs Library

- ADDER
- MULTIPLIER
- REG FILE
- I-CACHE
- ...
Optimization Framework Overview

- Step 3: Integrate circuit trade-offs into architectural models
  - To create *circuit-aware* models
Optimization Framework Overview

- **Step 4: Optimize**
  - Use special mathematical models to enable *convex optimization*

[Diagram showing the optimization framework with steps and components]

1. Simulate Random Designs
2. Fit Architecture Model
3. Architecture Circuit Link
4. Optimizer (GP Solver)
5. Energy Budget
6. Optimized Micro-Architecture

- Benchmark App(s)
- Macro Architecture
- Circuit Tradeoffs Library
- ADDER
- MULTIPLIER
- REG FILE
- I-CACHE
- ...
Experimental Setup

- 90nm CMOS technology
  - Static logic, except for SRAMs

- Energy-delay trade-offs
  - Logic units: use synthesis tools
  - Large memories: use CACTI

- Architectural Simulator
  - Joshua simulator from UIUC

- Applications
  - SPECint

- Let’s look at the design space without voltage first…
Energy-Performance Tradeoff Space

- Optimization of a dual-issue out-of-order processor
  - Significant performance-energy trade-off range as we tune underlying parameters

Graph:
- TSMC 90nm
- 1.2 V
- ~3x energy
- ~6x performance
Energy-Performance Tradeoff Space

- Optimization of a dual-issue out-of-order processor
- Significant performance-energy trade-off range as we tune underlying parameters

Clock Cycle: 18.6 FO4
Integer Unit: 1 cycle
I-cache: 32Kb @ 2 cycles
D-cache: 42Kb @ 1 cycle
Instr. Window Size: 8 entries

Clock Cycle: 19.0 FO4
Integer Unit: 1 cycle
I-cache: 32Kb @ 2.2 cycles
D-cache: 18Kb @ 1 cycle
Instr. Window Size: 9 entries

Clock Cycle: 28.4 FO4
Integer Unit: 1 cycle
I-cache: 32Kb @ 1.6 cycles
D-cache: 10Kb @ 1 cycle
Instr. Window Size: 9 entries

TSMC 90nm
1.2 V

~3x energy
~6x performance
Exploring High-Level Architectures

2-issue out-of-order architecture
Exploring High-Level Architectures

Graph showing energy (pJ per instruction) versus performance (MIPS). The graph includes a curve labeled "1-issue In-order architecture."
Exploring High-Level Architectures
Exploring High-Level Architectures

4-issue in-order architecture
Exploring High-Level Architectures

![Graph showing energy consumption versus performance (MIPS).](chart.png)

- 1-issue out-of-order architecture
Exploring High-Level Architectures

Energy (pJ per instruction) vs. Performance (MIPS) for different architectures. The graph shows that the 4-issue out-of-order architecture has the highest energy consumption at high performance levels.
Exploring High-Level Architectures

Optimal Architecture:

- in-order, 1-issue
- in-order, 2-issue
- in-order, 4-issue
- out-of-order, 1-issue
- out-of-order, 2-issue
- out-of-order, 4-issue

1-issue out-of-order, never efficient
Voltage Scaling

- Voltage is a powerful parameter
  - Just turn up the voltage a bit, and everything runs faster

- So let’s add voltage scaling to the study now...
Voltage Scaling

- Voltage is a powerful parameter
  - Just turn up the voltage a bit, and everything runs faster

Voltage Range: 0.7V – 1.4V, normalized to 0.9V

- ~4x energy
- ~3x performance
Optimization: It’s All About Marginal Costs

- To optimize, you want the cheapest source of performance

- Broadly, we consider two sources…
  - You can buy from or sell to either source (with no transaction/exchange fees)

![Diagram showing two sources of performance: Architecture & Circuit Design and Voltage Scaling, with current prices labeled.]
What the Vendors are Offering:

Energy-Performance Cost Profiles

- **Energy**
  - Current Price: 1%

- **Performance**
  - Current Price: 5%

- **Cost Profiles**
  - Architecture & Circuit Design
  - Voltage Scaling

![Graphs showing energy vs. performance and voltage scaling](image-url)
Scenario #1: Unoptimized Design

- **Architecture & Circuit Design**: Current Price = 5%
- **Voltage Scaling**: Current Price = 1%
Scenario #1: Unoptimized Design

Question: What should you do?
Scenario #1: Unoptimized Design

Architecture & Circuit Design
- Current Price: 2%

Voltage Scaling
- Current Price: 1.1%

150 MIPS lost
50 pJ/op saved

150 MIPS regained
16 pJ/op spent
Scenario #1: Unoptimized Design

- Architecture & Circuit Design
  - Current Price: 2%

- Voltage Scaling
  - Current Price: 2%

Graphs showing performance (MIPS) vs. energy (pJ per instruction) and speedup multiplier vs. energy multiplier.
Scenario #2: Changing Costs

- Let’s say you start with your now optimized design
  - But you want more performance…so you start buying from both categories
- But let’s say Voltage Scaling costs never change
  - While Architecture & Circuit Design quickly become more expensive
    - You use up all the good architecture & circuit design techniques

![Diagram showing cost comparison between Architecture & Circuit Design and Voltage Scaling](image-url)
Scenario #2: Changing Costs

- **Architecture & Circuit Design**
  - Current Price: 2%
- **Voltage Scaling**
  - Current Price: 2%
Scenario #2: Changing Costs

Optimal architecture/circuit design never changes
Voltage Scaling Marginal Costs

- Marginal cost profile for voltage scaling is relatively steady
  - Costs don’t change too rapidly

\[ MC\% = 2.3 \]

Voltage Range: 0.7V – 1.4V, Normalized to 0.9V

\[ MC\% = 0.8 \]

\[ MC\% = \% \text{ Energy Cost for 1\% Performance} \]
- Compare voltage scaling vs architectural marginal costs

![Diagram showing energy per instruction vs performance (MIPS) with various MC% values: MC% = 14.3, MC% = 3.2, MC% = 0.92, MC% = 0.66, MC% = 0.25, MC% = 0.49.](image)
Matching Marginal Costs

- Recall: For optimality marginal costs must match
Matching Marginal Costs

- Recall: For optimality marginal costs must match

Architecture + Circuit Design
Trade-off Curve

Voltage Scaling Curve
Matching Marginal Costs

- Recall: For optimality marginal costs must match

Architecture + Circuit Design
Trade-off Curve

Voltage Scaling Curve
Matching Marginal Costs

Recall: For optimality marginal costs must match
Architecture Sweet Spot

- Interesting space is where marginal costs match with voltage MC’s
**Architecture Sweet Spot**

- Interesting space is where marginal costs match with voltage MC’s

---

**Clock Cycle:**
- MC% = 14.3
  - Clock Cycle: 19.6 FO4
  - Integer Unit: 1 cycle
  - I-cache: 32Kb @ 2.2 cycles
  - D-cache: 14Kb @ 1.1 cycle
  - Instr. Window Size: 10 entries

**Clock Cycle:**
- MC% = 3.2
  - Clock Cycle: 20.6 FO4
  - Integer Unit: 1 cycle
  - I-cache: 32Kb @ 2.3 cycles
  - D-cache: 12Kb @ 1.1 cycle
  - Instr. Window Size: 11 entries

**Clock Cycle:**
- MC% = 0.92
  - Clock Cycle: ...
  - Integer Unit: ...
  - I-cache: ...
  - D-cache: ...
  - Instr. Window Size: ...

**Clock Cycle:**
- MC% = 0.66
  - Clock Cycle: ...
  - Integer Unit: ...
  - I-cache: ...
  - D-cache: ...
  - Instr. Window Size: ...

**Clock Cycle:**
- MC% = 0.25
  - Clock Cycle: ...
  - Integer Unit: ...
  - I-cache: ...
  - D-cache: ...
  - Instr. Window Size: ...

**Clock Cycle:**
- MC% = 0.49
  - Clock Cycle: ...
  - Integer Unit: ...
  - I-cache: ...
  - D-cache: ...
  - Instr. Window Size: ...
Full Optimization With Voltage Scaling

![Graph showing energy (pJ per instruction) versus performance (MIPS) for different configurations: in-order, 1-issue, in-order, 2-issue, in-order, 4-issue, out-of-order, 1-issue, out-of-order, 2-issue, out-of-order, 4-issue.]
Recall: Without Voltage Scaling

Optimal Architecture:
Full Optimization With Voltage Scaling

With voltage scaling:
Two architectures dominate energy-efficient frontier

Optimal Architecture:
2-issue in-order  2-issue ooo
A Few Designs Can Go A Long Way

- Voltage scaling with two fixed designs (architecture and circuits)
  - Can still achieve within 3% of optimal for a large part of the design space!

![Graph showing energy overhead vs. performance with 3% overhead line]
Conclusion

- Joint optimization of architecture and circuits is possible
  - All you need is a performance simulator and circuit libraries

- When optimizing, always consider marginal costs
  - Our framework helps do this in a systematic fashion

- Efficient processor design
  - Architecture/circuits have rapidly changing marginal costs; voltage less so
    - Law of diminishing returns sets in rapidly for the architecture/circuit design
    - Small set of architecture/circuit features are efficient

- Important to pick a good architecture (in the sweet spot)
  - Want well-tuned design (cache sizes, cycle time, etc.)
  - Then voltage scaling can go a long way to achieve the desired performance target
Thank You!