Power and Energy

Charles Li and Deepak Pallerla

Power: A First-Class Architectural Design Constraint

Motivations

- IT was 8% of US electricity usage in 2000
  - Increasing over time
- Chip die power density increasing linearly
  - Eventually can’t cool them
- Very general motivations
  - Appropriate for a general overview

CMOS Power Basics

- **$P = ACV^2f + \tau AVI_{short} + VI_{leak} = P_{switching} + P_{short} + P_{leakage}$**
  - ACV^2f = Activity × Capacitance × Voltage^2 × Frequency
  - τAVI_{short} = Short circuit time × Activity × Voltage × Short circuit current
  - VI_{leak} = Voltage × Leakage current

- Reduce voltage?
  - Reduces max frequency unless you reduce MOSFET V_{th}
  - Reducing V_{th} increases I_{leak}

- **Reducing V will decrease $P_{switching}$ and increase $P_{leakage}$ until $P_{leakage}$ dominates**

What Does Efficiency Mean?

- Portable devices carry fixed amount of energy in battery
  - Minimizing **energy per operation** better than minimizing power
  - MIPS/W a common metric (simplifies to instructions per Joule)
  - MIPS/W can be misleading for quadratic devices (CMOS)

- Non-portable devices should minimize power
  - Different from minimizing energy per operation

Power Reduction - Logic

Clock tree is a significant power consumer. What can you do about it?

- **Clock gating -** Turn off clocks to unused logic
  - Increases clock skew but solved by better tools
- **Half frequency -** Use rising *and* falling edges, run at half frequency
  - Increases logic complexity and area
- **Half swing -** Clock swing only half of supply voltage
  - “Increases the latch design’s requirements”
  - Hard to use when supply voltage is already low

Power Reduction - Logic (cont.)

- **Asynchronous logic -** Clocks use power, so don’t use clocks. Many problems.
  - Extra logic and wiring required for completion signals
  - Absence of design tools, difficult to test
    - Still true 20 years later?
  - Amulet - asynchronous ARM implementation

- **Globally asynchronous, locally synchronous logic**
  - Reduce clock power and skew on large chips
  - Ability to reduce frequency and voltage to specific parts of chip
  - Best of both worlds

Power Reduction - Architecture

Dynamic power loss upon memory access, leakage loss from being turned on.

- **Memory - Filter cache**
  - Extremely small cache ahead of L1 cache
  - Sacrifice performance but keep L1 cache at low power most of the time
- **Memory - Banking**
  - Split memory into banks, turn on bank being used
  - Requires spatial locality and disk backup for off banks

Power Reduction - Architecture (cont.)

Memory buses are a significant source of power usage.

- Gray code addresses reduces switching for sequential addresses.
- Compression reduces data transfer amounts
  - Presumably saves more power than compression and decompression

Decimal	Gray			
	4	3	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1

Power Reduction - Architecture (cont.)

- **Pipelining** is done to increase clock frequency (reduce critical path length)
  - Limits voltage reduction
- **Parallel processing** improves efficiency
  - General purpose computation (SPEC benchmarks) not very parallel
  - DSPs are highly parallel and power efficient
    - This points towards accelerators for further improvements

Power Reduction - Operating System

Operating system can support voltage scaling. How do we use it best?

- **Application controlled -** Apps use OS interface to scale voltage for itself
  - Requires app modification
- **OS controlled -** OS detects when to scale voltage
  - No app modification needed
  - Difficult to make detection optimal

Applications for Efficient Processors

- **High MIPS/W** (low energy per operation)
  - “The obvious applications [...] lie in mobile computing.”
  - “mobile phones will surpass the desktop as the defining application environment for computing”
    - Pretty accurate in 2020
- **Low power**
  - Servers and data centers
  - More compute for same power

Future Challenges

- Smaller FETs need lower V_{th}
- Lower V_{th} increases leakage current
  - Use low V_{th} FETs for high frequency paths
  - Use high V_{th} FETs for low frequency paths
- In general power must be considered early in design process
  - Currently happening
- Tools must support power analysis
  - Currently happening

Strengths

- Broad overview of power saving techniques at different levels
- Distinguishes between power and energy
- Predicts rise of mobile computing

Weaknesses

- Individual techniques vaguely described
- Heterogeneous designs not mentioned (ex. big.LITTLE)
- OS section only sort of discusses energy aware scheduling
- Nearly 20 years old, what's new?

Power Struggles: Revisiting the RISC vs CISC Debate on Contemporary ARM and x86 Architectures

Motivation

RISC v. CISC pt.1

- First debates in 1980s
  - Focused on desktops and servers
  - Primary design constraints
    - Area
    - Chip design complexity

RISC v. CISC pt.1

- "RISC as exemplified by MIPS provides a significant processor **performance** advantage."
- " ... the Pentium Pro processor achieves 80% to 90% of the **performance** of the Alpha 21164 ... It uses an aggressive out-of-order design to overcome the instruction set level limitations of a CISC architecture. On floating-point intensive benchmarks, the Alpha 21164 does achieve over twice the **performance** of the Pentium Pro processor."
- "with aggressive microarchitectural techniques for ILP, CISC and RISC ISAs can be implemented to yield very similar **performance**."

RISC v. CISC pt.2

- 2013
  - Smartphones and tablets in addition to desktops and servers
  - Primary design constraints
    - Energy
    - Power
  - New markets
    - ARM servers for energy efficiency
    - x86 for mobile and low power devices for performance

Does ISA affect performance, power, energy efficiency?

Framing the Impacts

Table 1. Summary of RISC and CISC Trends.

	Format	Operations	Operands
RISC / ARM	○ Fixed length instructions ○ Relatively simple encoding ○ ARM: 4B, THUMB(2B, optional)	○ Simple, single function operations ○ Single cycle	○ Operands: registers, immediates ○ Few addressing modes ○ ARM: 16 general purpose registers
CISC / x86	○ Variable length instructions ○ Common insts shorter/simpler ○ Special insts longer/complex ○ x86: from 1B to 16B long	○ Complex, multi-cycle instructions ○ Transcendentals ○ Encryption ○ String manipulation	○ Operands: memory, registers, immediates ○ Many addressing modes ○ x86: 8 32b & 6 16b registers
Historical Contrasts	○ CISC decode latency prevents pipelining ○ CISC decoders slower/more area ○ Code density: RISC < CISC	○ Even w/ μcode, pipelining hard ○ CISC latency may be longer than compiler's RISC equivalent	○ CISC decoder complexity higher ○ CISC has more per inst work, longer cycles ○ Static code size: RISC > CISC
Convergence Trends	○ **μ-op cache minimizes decoding overheads** ○ **x86 decode optimized for common insts** ○ **I-cache minimizes code density impact**	○ **CISC insts split into RISC-like micro-ops; optimizations eliminated inefficiencies** ○ **Modern compilers pick mostly RISC insts; μ-op counts similar for ARM and x86**	○ **x86 decode optimized for common insts** ○ **CISC insts split into RISC-like micro-ops; x86 and ARM μ-op latencies similar** ○ **Number of data cache accesses similar**
Empirical Questions	○ How much variance in x86 inst length? Low variance ⇒ common insts optimized ○ Are ARM and x86 code densities similar? Similar density ⇒ No ISA effect ○ What are instruction cache miss rates? Low ⇒ caches hide low code densities	○ Are macro-op counts similar? Similar ⇒ RISC-like on both ○ Are complex instructions used by x86 ISA? Few complex ⇒ Compiler picks RISC-like ○ Are μ-op counts similar? Similar ⇒ CISC split into RISC-like μ-ops	○ Number of data accesses similar? Similar ⇒ no data access inefficiencies

Choosing Platforms

- Want as many similarities as possible
  - Technology node
  - Frequency
  - High performance/low power transistors
  - L2-Cache
  - Memory Controller
  - Memory Size
  - Operating System
  - Compiler
- Intent: Keep non-processor features as similar as possible.

Choosing Platforms: Best Effort

- ARM/RISC
  - Cortex-A9
  - Cortex-A8
- x86/CISC
  - Sandy Bridge (Core i7)
  - Atom
- Differences in tech node and frequency handled by estimate scaling to 45nm and 1GHz

Table 2. Platform Summary.

	32/64b x86 ISA		ARMv7 ISA	
Architecture	Sandybridge	Atom	Cortex-A9	Cortex-A8
Processor	Core 2700	N450	OMAP4430	OMAP3530
Cores	4	1	2	1
Frequency	3.4 GHz	1.66 GHz	1 GHz	0.6 GHz
Width	4-way	2-way	2-way	2-way
Issue	OoO	In Order	OoO	In Order
L1 Data	32 KB	24 KB	32 KB	16 KB
L1 Inst	32 KB	32 KB	32 KB	16 KB
L2	256 KB/core	512 KB	1 MB/chip	256 KB
L3	8 MB/chip	—	—	—
Memory	16 GB	1 GB	1 GB	256 MB
SIMD	AVX	SSE	NEON	NEON
Area	216 mm^2	66 mm^2	70 mm^2	60 mm^2
Tech Node	32 nm	45 nm	45 nm	65 nm
Platform	Desktop	Dev Board	Pandaboard	Beagleboard
Products	Desktop	Netbook Lava Xolo	Galaxy S-III Galaxy S-II	iPhone 4, 3GS Motorola Droid

Choosing Workloads

- RISC and CISC both claim to be good for mobile, desktop, and server
- Single-threaded core-focused

Table 3. Benchmark Summary.

Domain	Benchmarks	Notes
Mobile client	CoreMark	Set to 4000 iterations
	WebKit	Similar to BBench
Desktop	SPECCPU2006	10 INT, 10 FP, test inputs
Server	lighttpd	Represents web-serving
	CLucene	Represents web-indexing
	Database kernels	Represents data-streaming and data-analytics

Metrics

- Performance
  - Wall-Clock Time
  - Built-In Cycle Counters
- Power
  - Wattsup
  - Multiple runs for average system power; control run for board power
  - Chip power = system power - board power

Key Findings (Perf)

- Execution time varies greatly
- Upon normalization to CPI and instruction count/mix, performance differences are explicable by microarchitectural differences (branch pred/cache size)

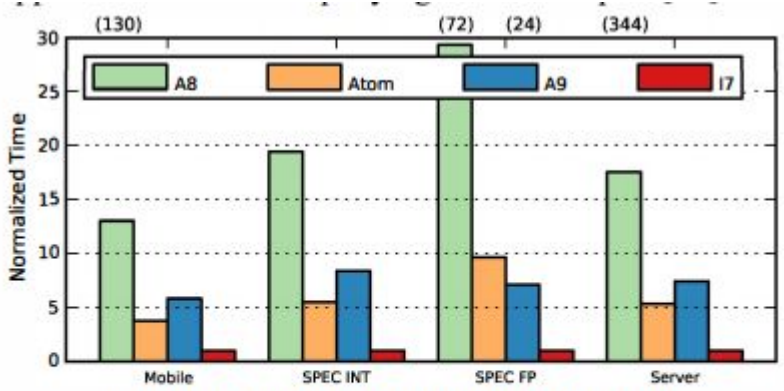

Figure 2. Execution Time Normalized to i7.

Ratio	Mobile	SPEC INT	SPEC FP	Server
A8 to Atom	3.4 (34)	3.5	4.2 (7.4)	3.7 (103)
A9 to i7	5.8	8.4	7.2 (23)	7.4

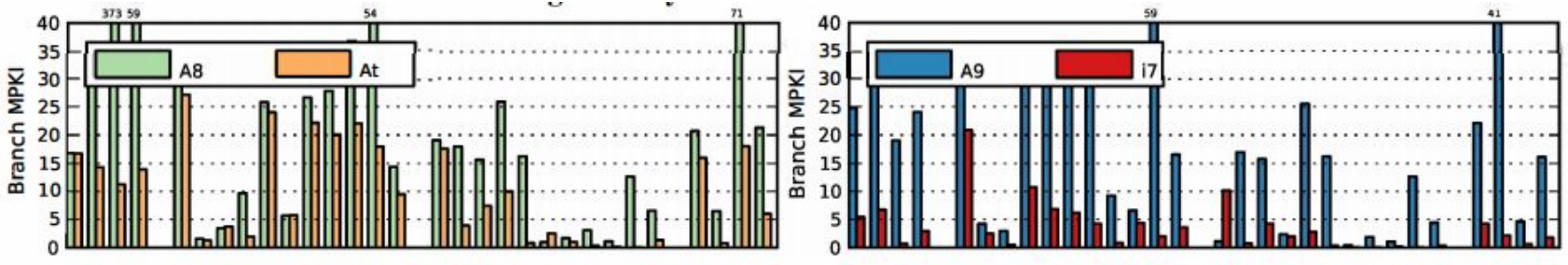

Figure 8. Branch Misses per 1000 ARM Instructions.

Key Findings (Power)

- i7 core is not power optimized so it has exceptionally high power
- Generally, core power is based on its optimization level
- Most differences in energy can be explained by differences in performance (e.g. BP) and power (Optimized for or not)

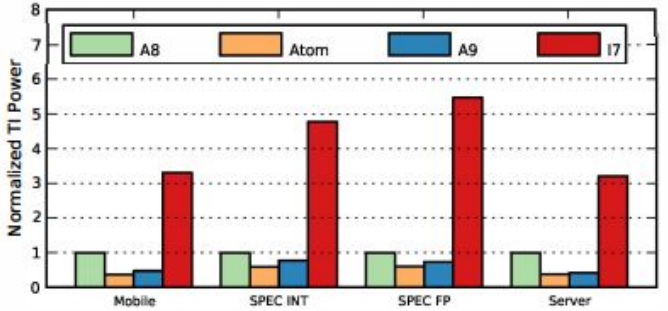

Figure 12. Tech. Independent Avg. Power Normalized to A8.

Ratio	Mobile	SPEC INT	SPEC FP	Server
Atom to A8	0.6	0.6	0.6	0.6
i7 to A9	7.0	6.1	7.4	7.6

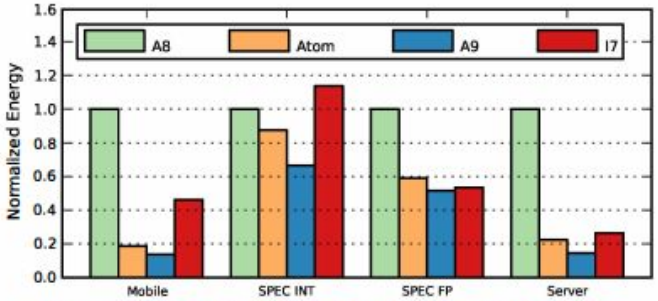

Figure 13. Raw Average Energy Normalized to A8.

Ratio	Mobile	SPEC INT	SPEC FP	Server
A8 to Atom	0.8(0.1)	0.9	0.8 (0.6)	0.8(0.2)
i7 to A9	3.3	1.7	1.7 (1.0)	1.8

Trade-Off Analysis

- Cubic trade-off in power and performance
- Quadratic trade-off in energy and performance
- Pareto optimality not dependent on ISA

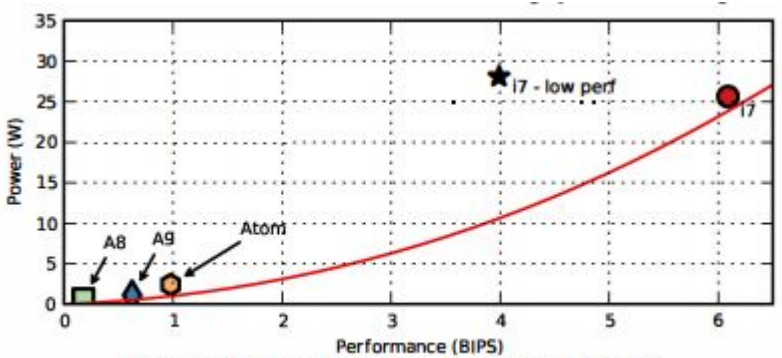

Figure 14. Power Performance Trade-offs.

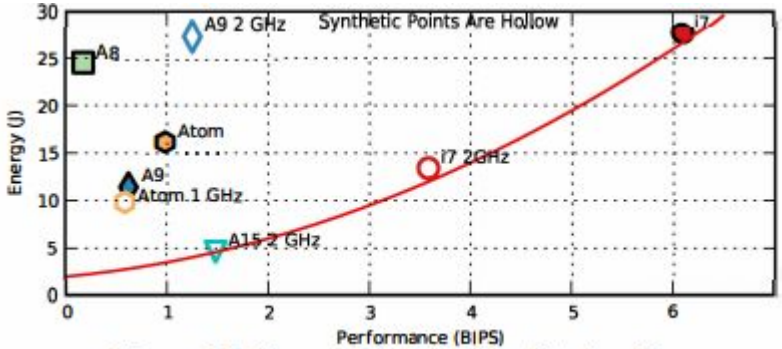

Figure 15. Energy Performance Trade-offs.

ISA does NOT affect
performance,
power,
energy efficiency

Strengths

- Presents intuition first, then affirms with results
- Does a good job of drawing relevant data and conclusions with a severely limited scope
- Admit to several limitations in the paper itself

Weaknesses

- Comparison to performance optimized i7 Sandy Bridge core seems shaky -- could have used more similarly optimized technology for better results
  - Option 1: More test points so we can maybe group into power optimized, perf optimized, and somewhere in the middle
  - Option 2: Same number of test points but homogenous in use case
- Normalizing the cores to a specific frequency and technology node obfuscates the original purpose of the cores, which might differ from core to core (EDP?)
- Evaluation is now 7 years old, what differences might we expect to see in 2020 v 2013?