Energy Efficient System Architectures

Principles and Examples

Avinoam Kolodny

Electrical Engineering Department
Technion – Israel Institute of Technology

Green Photonics – TU Berlin 2015
The Disruption Principle

- A problem grows quietly.....
The Disruption Principle

- A problem grows quietly.....
The Disruption Principle

• A problem grows quietly.....
• .... Until it becomes critical.
The Disruption Principle

• A problem grows quietly.....
• .... Until it becomes critical.
• At this point something must be done!
Example:
Power Dissipation in Processor Chips
Power Dissipation in Processor Chips Until 2005

http://cpudb.stanford.edu/
Power Dissipation in Processor Chips
After 2005 – flat!

http://cpubdb.stanford.edu/
What Caused the Disruption?

- **Power limit (thermal)**
  Too much heat for air-cooled computers

- **Energy limit**
  Battery life issue in mobile devices
What were the reasons for growth in power dissipation?
What were the reasons for growth in power dissipation?

1. **Non-ideal scaling**
   - Classical Dennard scaling = Fixed power for a fixed chip area.
   - Many parameters were not scaled ideally:
     Supply voltage, Frequency, Leakage, Interconnect, ........

2. **Wasteful spending of power**
   while optimizing for performance
What has been done to stop the growth of power?

- Savings at all levels
  - Power is cumulative
- Reduction of waste
- Compromise on speed
Main Architectural Changes in Microprocessors Since 2005

http://cpudb.stanford.edu/
Evolution of Multi-Core Chips (CMPs)

Single Core (Uni-Core)

CPU
Cache

Dual Core

CPU1
Cache

CPU2
Cache

Quad Core

CPU1
Cache

CPU2
Cache

CPU3
Cache

CPU4
Cache

Performance = $\alpha_1 \sqrt{(\text{Area})}$

Power = $\alpha_2 (\text{Area})$

* F. Pollack, MICRO 1999.
* S. Borkar, DAC 2007.
Why Big Uni-Cores are Inefficient?

The Interconnect Bottleneck

Interconnect Delay is dominant

* H. Bakoglu and J. Meindl, TVLSI 1985
* M. Bohr, IEDM 1995
Why Big Uni-Cores are Inefficient?

The Interconnect Bottleneck

Interconnect Delay is dominant

Interconnect Power is dominant

Bit-Transportation energy is larger than computation energy!!

* H. Bakoglu and J. Meindl, TVLSI 1985
* M. Bohr, IEDM 1995

Pollack’s Empirical Rule

Performance = $\alpha_1 \sqrt{\text{Area}}$

Power = $\alpha_2 \text{Area}$

Uniprocessor Performance

AREA (or POWER)

* F. Pollack, MICRO 1999.
* S. Borkar, DAC 2007.
Processor Architectures:  
Uni-Core, Symmetric multicore

PERFORMANCE

POWER or AREA

Symmetric CMP

Uni-Core

Power Wall
Locality Principle

• Power-efficiency requires many parallel local computations

• To utilize multiple cores, software needs to run in parallel threads
How Much Parallelism Exists in Real Programs?

Serial Code

Parallel

? Different phases

Time Axis
Characterizing a Multi-Threaded Software Program

Performance

Number of assigned Worker Threads

?
Study of Parallelism in Real Software

• Use PARSEC benchmark suite
• Capture the parallelism inherent in the algorithm
  – Limited by inter-thread synchronization
• Assume an “ideal machine”
  – Architecture model with no constraints
  – No shared resources
  – Perfect memory system – 1 cycle latency

* O. Itzhak, I. Keidar, A. Kolodny and U. Weiser, “Performance scalability and dynamic behavior of Parsec benchmarks on many-core processors,” SFMA 2014
Perfect parallelism scalability: blackscholes ("Embarrassingly Parallel")
Good parallelism scalability: fluidanimate
Explaining the Difference:
Some threads work part-time only!
Poor parallelism scalability: raytrace

![Graph showing performance (IPC) vs. worker-threads (Theoretical max vs. Actual)]
Challenge of Parallel Programming

• Many multi-threaded applications are not scalable: They cannot exploit a large number of cores because some of the code must run serially.
Amdahl’s Principle

If a fraction $f$ of the code must run serially, parallel speedup is limited to $1/f$ when all cores are identical.

“Hurry UP” Principle: Waiting Wastes Power

- Idle cores waste power (clock, leakage,...)
- Other system components consume power
- It makes sense to hurry up, finish all work and shut down
Asymmetric (=Heterogeneous) Multi-Core

- Small cores of area: $a$
- Large core area: $\beta a$ – used for serial code
- Parallel phases execute on all cores

Processor Architectures:
Uni-Core, symmetric multicore, Asymmetric

Performance vs. Power Wall

Uni-Core
Symmetric CMP
Asymmetric CMP
Future VLSI systems

• Multiple computing cores + cache memory banks
• Special-purpose Accelerators
• Power Management:
  – Dynamic voltage and frequency scaling (DFVS)
  – “Dark silicon”
• Network on Chip (NoC)
Heterogeneous Computing:
Application Specific Accelerators

Energy Efficiency
[Performance/power]

Applications range

General Purpose Cores

Accelerators
Packet-Switched Network-on-Chip (NoC)

- **Network** instead of **dedicated wires** and **buses**
  - Inherently parallel
  - Efficient sharing of wires

- **Advantages:** scalable, cost-effective throughput

* E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny - Journal of systems architecture, 2004
Rent’s Principle (locality revisited)

The original rule (for wires in systems)

Rent’s exponent < 1

Terminals per module

1000
100
10
1

10^0 10^1 10^2 10^3 10^4

Transistors per module
Rent’s Principle (locality revisited)

- Rent’s Rule for NOCs – Bandwidth version

\[ B = kG^r \]

- Low \( r \) – local traffic
- High \( r \) – global traffic

Basic Disadvantage of NoCs: Latency

Few packets, which go long distances across the chip:
- Consume most of the network’s bandwidth
- Increase the average latency at light load

* R. Manevich, I. Cidon, and A. Kolodny, "Handling Global Traffic in Future CMP NoCs", SLIP 2012
Latency is even worse when NoC is **Loaded**!

A loaded network quickly reaches a **saturation point**!

---

Reducing Hop-Count by Express Links

Example: PyraMesh Topology

- Hierarchical 2D mesh.
- Global packets are routed through higher hierarchy levels.

✔ Overall hops-count is reduced.

✔ Average latency is reduced.

✔ Average BW per router is reduced.

* R. Manevich, I. Cidon, and A. Kolodny, "Handling Global Traffic in Future CMP NoCs", SLIP 2012
NoC-Aware Chip Layout
Example: “Nahalal Architecture”

- Partitioning of cache lines by “shared” vs. “private”
- Keep shared data in the center cache-banks
- Use outer cache-banks for private data

* Z. Guz et al., [SPAA-2008], [CA-Letters’07]
NoC-Aware Chip Layout
Example: “Nahalal Architecture”

- Partitioning of cache lines by “shared” vs. “private”
- Keep shared data in the center cache-banks
- Use outer cache-banks for private data

* Z. Guz et al., [SPAA-2008], [CA-Letters’07]
Bypassing The Challenge of Parallel Programming: Run UNRELATED Programs on multi-core computers
Beyond a Single Chip
Beyond a Single Chip: The External Memory Wall
von Neumann (Architecture) Bottleneck

A bottleneck of both throughput and power!
# Math is Cheap, Data Movement is Expensive!

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy/Op (45 nm)</th>
<th>Cost (vs. ALU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALU operation</td>
<td>0.05 pJ</td>
<td>1X</td>
</tr>
<tr>
<td>Move 10 nm on-chip</td>
<td>2.4 pJ</td>
<td>50X</td>
</tr>
<tr>
<td>Load from on-chip SRAM</td>
<td>2.5 pJ</td>
<td>50X</td>
</tr>
<tr>
<td>Send to off-chip DRAM</td>
<td>320 pJ</td>
<td>6,400X</td>
</tr>
</tbody>
</table>

*Data from J. Brunhaver, W. Dally, M. Horowitz, S. Kvatinsky, Stanford University*
Why accelerators are efficient

Architected for Specific Applications

High data locality and parallelism

Application

Technology

Architecture
Example: Camera Signal Processing Pipeline

* Slide from S. Kvatinsky, Stanford University
Hold Intermediate in Line Buffers

* Slide from S. Kvatinsky, Stanford University
Hold Intermediate in Line Buffers

* Slide from S. Kvatinsky, Stanford University
Hold Intermediate in **Line Buffers**

* Slide from S. Kvatsinsky, Stanford University
Hold Intermediate in Line Buffers

Line buffers are SRAM

* Slide from S. Kwatinsky, Stanford University
Hold Intermediate in Line Buffers

Line buffers are sliding windows

* Slide from S. Kvatinisky, Stanford University
Hold Intermediate in Line Buffers

Line buffers are small and constant in size

* Slide from S. Kvatinsky, Stanford University
• In Image processing, it is easy to exploit locality and parallelism.
• Can we do something similar for general-purpose programs?
Let’s look at a curve of Constant Throughput.

- Large bandwidth utilization ➔ Large power
- Increase on-die-memory (e.g. innovative cache? new ideas: Memory Intensive Architectures)
• In the past, memory was added primarily for enhancing performance
• Now, we need memory-intensive systems for energy-efficiency
Cache Machines

• Many cores behind a shared cache

Performance

To External Memory

# Threads

Cache Non Effective point (CNE)
Multi-Thread Machines (e.g. GPGPU)

- Memory latency shielded by multiple thread execution

![Threads Architectural States Diagram]

- Performance vs. Max performance
- Bandwidth Limitations
- To External Memory
- Execution vs. Memory access
A Unified Machine Model

- Use both cache and many threads to shield memory access
  - The uniform framework renders the comparison meaningful
  - We derive simple, parameterized equations for performance, power, BW,..
Unified Machine Performance

- 3 regions: Cache efficiency region, The Valley, MT efficiency region

Wasteful Effects of Shared Resources

(↑ Energy, ↓ Throughput)

• Threads might disturb each other as they need a shared cache, memory bus, network bandwidth, disk, ....

• **Destructive Interference:** Causing unnecessary work
  Example: threads pollute each others’ cache

• **Collisions:** Requesting the same resource
  The waiting cores waste energy.
New Research Direction: New Memory-Intensive Architectures

- Embed memory in execution units?
- Add computational capabilities within memory?
New memory devices: Memristors

Current

Voltage

Decrease resistance

Current

“0” = High Resistance
“1” = Low Resistance
Memristors Enable New Structures

- Memory on-top of logic – high density
- Store data locally above circuit which needs it
Example of Embedding Memory in a Processor Pipeline: Multistate Register Concept

• CMOS register – a single state
• Multistate register – multiple states, a single state is active
Operation of Multistate Register

Mode 2b
State switch
Read new state
state to RRAM

R. Patel, S. Kvatsinsky, E. G. Friedman, and A. Kolodny, "Multistate Register Based on Resistive RAM," IEEE Transactions on Very Large Scale Integration (VLSI)
Continuous Flow Multithreading

- Using multistate registers
- Enhanced performance to energy ratio
  (44% improvement over SoE, 310% over SMT)

From Small Systems to Data Centers
Data Centers

- Size: 1 million m²
- Power: 100 MegaWatts (only ~30% of it for computing!)
- Total power cost in U.S. 2014 - $10B\(^{(1)}\)

FIGURE 1.4: Approximate distribution of peak power usage by hardware subsystem in one of Google’s datacenters (circa 2007).

Lious Borrosso talk http://www.cs.berkeley.edu/~rxin/db-papers/WarehouseScaleComputing.pdf
Let’s Use the Same Principles to Save Power in Data Centers!
Example: Avoid Unnecessary DRAM usage?

Data source (SSD/NIC)

Front End

DRAM

Operations CPU

Data Copy

Energy of Write/Read data: $E$

Cache/Memory are not effective if:

Memory related:
Reuse distance: >1G access

Cache related:
Reuse distance: >1M access

* T. Morad et al., unpublished
Example: O/S Scheduling of Threads for Energy Efficiency

* T. Morad et al., unpublished
Example: Avoid Interference on Links in Data-Center Network?

Although each tenant gets his own servers, tenant 1 performance depends on Tenant 2 traffic.

Possible consequences:
- Performance degradation >50%
- Unpredictable performance or response time

* E. Zahavi et al., unpublished
Solution Example:
Links as a Service, to Isolate Tenants in the Cloud

- Dedicating Links to Tenants – No Shared Links!
- Guarantee the resulting tenant network:
  - Support any admissible traffic pattern
  - Rearrangeable non-blocking routing

* E. Zahavi et al., unpublished
Summary

• Many opportunities to improve energy efficiency:
  – Energy-efficient architectures
  – Power-aware system management
Thanks
This talk is based on joint work with many students and collaborators:

Eitan Zahavi
Evgeny Bolotin
Nir Magen
Oved Itzhak
Shahar Kvatsinsky
Tomer Morad
Yaniv Ben-Itzhak
Yoni Aizik
Yuval Nacson
Zigi Walter
Zvika Guz

Avi Mendelson
Eby Friedman
Idit Keidar
Isaac Keslassy
Israel Cidon
Uri Weiser
Yoav Etzion