Extreme Scale Computer Architecture: Energy Efficiency from the Ground Up

Josep Torrellas
Department of Computer Science
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

WNTC Workshop
December 2012
Wanted: Energy-Efficient Computing

- **Extreme Scale computing**: 100-1000x more capable for the same power consumption and physical footprint
  - Exascale \((10^{18} \text{ ops/cycle})\) datacenter: 20MW
  - Petascale \((10^{15} \text{ ops/cycle})\) departmental server: 20KW
  - Terascale \((10^{12} \text{ ops/cycle})\) portable device: 20W
Energy-Efficiency Gap

- Goal:
  - 20W Tera-Op (sustained)
  - 20 pJoules/operation

- In comparison:
  - IBM Power7 released 2010: MCM 800W for 1TFlop Peak
    - Problem is harder than it looks:
      - Machines spend much of the energy transferring data
      - Minimizing E in data transfer, not ALU op is the challenge
Recap: How Did We Get Here?

• **Ideal Scaling** (or Dennard Scaling): Every semiconductor generation:
  - Dimension: 0.7
  - Area of transistor: $0.7 \times 0.7 = 0.49$
  - Supply Voltage ($V_{dd}$), $C$: 0.7
  - Frequency: $1/0.7 = 1.4$

$$P_{dyn} \propto CV_{dd}^2 f$$

Area: $A$

- $x$ transistors
  - Power density: $CV_{dd}^2 f/A$

Area: $0.7^2 A$

- $x$ transistors
  - Power density: $0.7C \times 0.7^2V_{dd}^2 \times 1.4f/0.7^2A$
  - $= CV_{dd}^2 f/A$

**Constant power density**
Recap: How Did We Get Here? (II)

- **Real Scaling**: $V_{dd}$ does not decrease much.
  - If too close to threshold voltage ($V_{th}$) $\rightarrow$ slow transistor
  - Delay of transistor is inversely proportional to $(V_{dd} - V_{th})$

\[
T_g \propto \frac{V_{dd} L_{eff}}{\mu (V_{dd} - V_t)^\alpha}
\]

- Dynamic power density increases with smaller tech

- Additionally: There is the static power

**Power density increases rapidly**
Design for E Efficiency from the Ground Up

• New designs for chips with 1K cores:
  – Efficient support for high concurrency
  – Data transfer minimization

• New technologies:
  – Low supply voltage ($V_{dd}$) operation
  – Efficient on-chip voltage regulation
  – 3D die stacking
  – Resistive memory
  – Photonic interconnects
Thrifty Multiprocessor

- Funded by DOE, DARPA, NSF
- Runnemedede project lead by Intel and funded by DARPA UHPC [HPCA2013]

Josep Torrellas
Extreme Scale Computing
Low Voltage Operation

• $V_{dd}$ reduction is the best lever for energy efficiency:
  • Big reduction in dynamic power; also reduction in static power
  • Reduce $V_{dd}$ to bit higher than $V_{th}$ (Near Threshold Voltage--NTV)
    • Corresponds to $V_{dd}$ of about 0.55V rather than current 0.9V

• Advantages:
  • Potentially reduces power consumption by more than 40x

• Drawbacks:
  • Lower speed (1/10)
  • Increase in gate delay variation
Basics of Parameter Variation

- Deviation of device parameters from nominal values: eg Vth, Leff

Additionally: Same $\Delta V_{th}$ causes higher $\Delta f$ and $\Delta P$ at NTV
Variation in Thrifty Manycore

- Larger f variation at NTV
- Memories more vulnerable
- Power varies more

Using VARIUS-NTV by Karpuzcu et al.
Multiple $V_{dd}$ Domains at NTV: Hardly Effective

- On chip regulators have a high power loss (10+%)  
- To reduce costs, only coarse-grain (multiple-core) domains  
  - Already has variation inside the domain  
- Small $V_{dd}$ domain more susceptible to load variations  
  - Larger $V_{dd}$ droops → need increase $V_{dd}$ guardband

Work with:
Ulya Karpuzcu (U Minn) and Nam Sung Kim (U Wisc)
Propose: Energy Efficiency with a Single $V_{dd}$ Domain

One $V_{dd}$ domain, many $f$ domains

- Simple hardware, simple & effective core allocation

- Each cluster in the chip is a $f$ domain
- Allocation in units of multiples of clusters called Ensembles
  - Whole ensemble clocked at a single $f$
- Simpler variation-aware core allocation
Effectiveness of Single $V_{dd}$ Domain per Chip

Single $V_{dd}$ is more efficient

Normalized MIPS/Watt

288-core chip with 8-core clusters

Josep Torrellas
Extreme Scale Computing
Needed: Efficient On-Chip $V_{dd}$ Regulation

- Voltage regulators (VRs) have to be designed for high efficiency
  - Hierarchical design:
    - First level placed on a different die; second level regulate a small range only

- Energy-efficient design requires short Vdd guardbands
  - Need to tackle voltage droops due to load variation

From Nam Sung Kim, UWisc
Streamlined 1K-core Architecture

- Very simple cores (no structures for speculative execution)
- Cores organized in clusters with memory to exploit locality
- Each cluster is heterogeneous (has one large core)
- Special instructions for certain ops: fine-grain synch
- Single address space without hardware cache coherence
Managing the Power of On-Chip Memories

• On-chip memory leakage: major contributor of the NTV chip power
• Coarse-grained proposals are insufficient
  • Turn off some memory modules / disable cache ways / …
• Needed: power-on only the lines that contain useful data
• Proposal
  • Use on-chip memory technology that does not leak (eDRAM) ---
    but needs to be refreshed
  • Use fine-grain, intelligent refresh of the on-chip memory
• Great opportunity of major power savings
  • Much of the on-chip memory contains useless data!
When Useless Refresh Happens

- Cold lines: Lines not used or used far apart in time

- Hot lines: Lines actively used
Polyphase: Intelligent Refresh

- When to refresh:
  - Divide the retention period into equal intervals called Phases
  - Maintain for each line: phase in which it was last accessed (or refreshed)
  - A line is refreshed only when the same phase arrives in the next retention period.
Polyphase: Intelligent Refresh

• What to refresh:
  • Use state of the line:
    • Valid data but timeout: WB (n,m)
      • Dirty lines refreshed n times before writeback
      • Clean lines refreshed m times before inval
Simple Hardware

When to refresh:
- Cache controller keeps, for each line, the phase it was last refreshed/accessed
- At the beginning of phase: controller checks for lines with matching phase
- For each line: 2 bits for phase, 1 for valid

What to refresh:
- Keep a per-line countdown of refreshes
  - Reset at access
  - Decrement at refresh.
- When counter reaches zero, wb/inval

- 40-60% reduction in on-chip memory energy with no slowdown
Minimizing Data Movement

- Thrifty has several techniques to minimize data movement:
  - Many-core chip organization based on clusters
  - Mechanisms to manage the cache hierarchy in software
  - Simple compute engines in the mem controllers → Processing in Memory (PIM)
  - Efficient synchronization mechanisms
Software Managed Caches (SMC)

- When core references data, HW brings a copy of line to cache from first level of cache it finds it in
  - May not be latest version
- Writes do not invalidate/update other copies of the line
- Need instructions to perform explicit write-back and invalidate
SMC Programming

- Programmer/compiler inserts data-movement instructions at synchronization points
- Hopefully minimizes data transferred over hardware coherence

```
Thread 1
ST A[i]  ST A[i]
WB A[i]  WB A[i]

Thread 2
barrier

Current epoch

WB = W_{this} \cap R_{others}

Next epoch

INV = W_{others} \cap R_{this}

Past epoch

Josep Torrellas
Extreme Scale Computing
Micron’s Hybrid Memory Cube (HMC) [Micron10]:

- Memory chip with 4 or 8 DRAM dies over 1 logic die
- Can be placed in an MCM with processor dies
- DRAM dies only store data while logic die handles DRAM control

Future use of logic die:

- Support for Intelligent Memory Operations?
  - Preprocessing data as it is read from memory
  - Performing processor commands “in place”
Supporting Fine-Grain Parallelism

- Synchronization and communication primitives
  - Efficient point-to-point synch between two cores (F/E bits)
  - Dynamic hierarchical hardware barriers
What We Learned

• Naively translating programs written for coherent caches into SMC results in inefficient codes
  – Need: good development tools
• Using fine-grain synchronization with F/E bits is hard
  – It typically requires complete re-write of the code
Programmability

- Programming highly-concurrent machines has required heroic efforts
- Extreme-scale architectures, with emphasis on power-efficiency, may make it worse
  - Low $V_{dd}$ requires more concurrency to attain same performance
  - Need carefully manage locality and minimize communication
How to Program for High Parallelism?

• Expert programmers
  • Hooks to manage power and $V_{dd}$/frequency
  • Ability to map and control tasks
• Novice programmers:
  • High level programming models that express locality
    • *Hierarchical Tiled Arrays (HTA)*: computes in recursive blocks
    • *Concurrent Collections (CnC)*: computes in a dataflow manner
• Autotuning?

• … open problem
Conclusion

• Presented the challenges of Extreme Scale Computing:
  • Designing computers for energy efficiency from the ground up
  • Described some of the architecture and design ideas
  • Programmability may suffer: need focus on the software
  • There is a tradeoff between energy efficiency and resilience
Extreme Scale Computer Architecture:
Energy Efficiency from the Ground Up

Josep Torrellas
Department of Computer Science
University of Illinois at Urbana-Champaign
http://iacoma.cs.uiuc.edu

WNTC Workshop
December 2012