ENERGY-EFFICIENT ARCHITECTURES FOR EXASCALE SYSTEMS

Dr. Stephen W. Keckler
Senior Director of Architecture Research, NVIDIA
The Goal:
Sustained ExaFLOPS on Problems of Interest

... at reasonable cost
The End of Historic Scaling

Source: C Moore, Data Processing in ExaScale-Class Computer Systems, Salishan, April 2011
HETEROGENEOUS NODE

System Interconnect

[Diagram showing various components such as DRAM Stacks, NIC, NoC, L2 caches, LLC, MC, NVLink, and NV RAM]
How do we get to 50GFlops/Watt?
Start with an energy-efficient architecture
**CPU**
130 pJ/flop (Vector SP)
- Optimized for Latency
- Deep Cache Hierarchy

Haswell
22 nm

**GPU**
30 pJ/flop (SP)
- Optimized for Throughput
- Explicit Management of On-chip Memory

Maxwell
28 nm
CPU
2 nJ/flop (Scalar SP)
Optimized for Latency
Deep Cache Hierarchy

Haswell
22 nm

GPU
30 pJ/flop (SP)
Optimized for Throughput
Explicit Management of On-chip Memory

Maxwell
28 nm
HOW IS POWER SPENT IN A CPU?

In-order Embedded

- Instruction Supply: 42%
- Clock + Control Logic: 24%
- Data Supply: 17%
- ALU: 6%
- Register File: 11%

OOO Hi-perf

- Clock + Pins: 45%
- Fetch: 11%
- Rename: 10%
- Issue: 11%
- RF: 14%
- ALU: 4%
- Supply: 5%

Dally [2008] (Embedded in-order CPU)

Natarajan [2003] (Alpha 21264)
Latency-Optimized Core (LOC)

Throughput-Optimized Core (TOC)

PC
Branch Predict
IS
Register Rename
Instruction Window
Register File
ALU 1
ALU 2
ALU 3
ALU 4
Reorder Buffer

PCs
Select
IS
Register File
ALU 1
ALU 2
ALU 3
ALU 4
How do we continue to scale energy efficiency

...in a world where technology scaling is diminished?
Do Less Work

Eliminate waste and redundancy

Move fewer bits

Move data more efficiently
DO LESS WORK
Mixed Precision Arithmetic

double-precision
  1  11  52

5x precision bits
60x range

single-precision
  1  8  23

4x throughput
4x bandwidth
4x capacity
< 1/4 energy/op

half-precision
  1  5  10

Only use as much precision as you need
Exploit mix of representations
Scaled arithmetic
ELIMINATE WASTE

Temporal SIMT

Spatial SIMT (current GPUs)

32-wide datapath

time

1 cyc

thread

0

thread

31

1 warp instruction = 32 threads

Pure Temporal SIMT

1-wide
ELIMINATE WASTE
Temporal SIMT

32-wide (41%)

4-wide (65%)

1-wide (100%)

Increase efficiency on divergent code
ELIMINATE WASTE
Variable Warp Sizing

Small warps
+ Improved perf for divergent code
+ Better SIMD utilization

Emulate wide warp HW
+ Wider converged execution
+ Memory locality/convergence
+ Reduced power (frontend)

Rogers [ISCA 2015]
ELIMINATE REDUNDANCY

Scalarization

SIMT Execution

| scalar op | LD R2 ←<A> | LD R2 ←<A> | LD R2 ←<A> | LD R2 ←<A> |
| vector load | LD R3 ←R2, 1 | LD R3 ←R2, 2 | LD R3 ←R2, 3 | LD R3 ←R2, 4 |
| vector op     | ADD R4 ←R3, 2 | ADD R4 ←R3, 2 | ADD R4 ←R3, 2 | ADD R4 ←R3, 2 |

Scalarized SIMT Execution

| LD SR2 ←<A> | VLD SR3 ←SR2, 1 | ADD R4 ←SR3, 2 | ADD R4 ←SR3, 2 | ADD R4 ←SR3, 2 | ADD R4 ←SR3, 2 |

Lee [CGO 2013]
MOVE FEWER BITS
Register File Cache (RFC)

Small multi-ported register file
Capture locality of commonly used operands
Can reduce RF energy by 50%

Gebhart [ISCA 2011]
Move data more efficiently

Toggle-aware Compression

128-byte Uncompressed Cache Line

<table>
<thead>
<tr>
<th>4 bytes</th>
<th>4 bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00003A00</td>
<td>0x8001D000</td>
</tr>
<tr>
<td>0x0003A01</td>
<td>0x8001D008</td>
</tr>
</tbody>
</table>

8-byte flit

<table>
<thead>
<tr>
<th>8-byte flit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00003A00</td>
</tr>
</tbody>
</table>

Flit 0

XOR

| 0x00003A01 | 0x8001D008 |

Flit 1

= 0000...0010...0010...

# Toggles = 2

Pekhimenko [HPCA 2016]

Compression can increase power consumption

Goal: reduce bus toggling

128-byte FPC-compressed Cache Line

<table>
<thead>
<tr>
<th>8-byte flit</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x5 0x3A00 0x7 0x8001D000</td>
</tr>
<tr>
<td>0x5 0x3A01 0x7 0x8001D008</td>
</tr>
</tbody>
</table>

XOR

| 0x5 |

Flit 0

Metadata

<table>
<thead>
<tr>
<th>8-byte flit</th>
</tr>
</thead>
<tbody>
<tr>
<td>5 3A00 7 80001D000 5 1D</td>
</tr>
<tr>
<td>5 3A02 1</td>
</tr>
</tbody>
</table>

Flit 1

= 001001111...110100011000

# Toggles = 31
MINIMIZE DATA MOVEMENT

Packaging

- Reduces distance
- Increases bandwidth
- Offers opportunity to optimize signaling circuits

High-bandwidth on-package memory
MINIMIZE DATA MOVEMENT
Heterogeneous DRAM Architectures

Challenges
- Exploiting all available bandwidth
- Maximizing locality for frequently accessed data
MINIMIZE DATA MOVEMENT
Software-managed Caching with On-Package Memory

Strategies
- Aggressively migrate pages upon First-Touch to GDDR memory
- Pre-fetch neighbors of touched pages to reduce TLB shootdowns
- Throttle page migrations when nearing peak BW

Throughput Relative to No Migration

- Legacy CUDA
- First-Touch + Range Exp + BW Balancing
- ORACLE

Competitive with manual memory copy
Close to “perfect” prefetch

Agarwal [HPCA 2015]
MINIMIZE DATA MOVEMENT
Hardware Managed DRAM Cache

Tag overhead: hundreds of MB

Alloy tag and data in same DRAM row (Micro12)

Cache organization: optimize for bandwidth

Direct mapped, consecutive sets in same row

Results

Fine-grained transfers good for lower locality apps

Can eliminate some page migration overheads
LOOMING MEMORY POWER CRISIS

Bar chart showing power consumption and bandwidth for different memory types (GDDR3, GDDR5, HBM, HBM2, 1.5x Efficient HBM2) with a skull and crossbones symbol indicating a critical power level of 160W.
SUMMARY

Do Less Work

Eliminate waste and redundancy

Move fewer bits

Move data more efficiently