

Power Efficient Supercomputing

William Dally

Bell Professor of Engineering, Stanford University
Chief Scientist and Sr. VP of Research, NVIDIA



High performance computing is
power limited.

High-Performance Computing is Power Limited

- What can be put on a chip is limited by power, not area.
- Cost of energy to run a cluster equals purchase cost in less than 18 months.
- Cost of provisioning a machine room with adequate power is typically many times cost of cluster.
- What matters now is j/FLOP

Energy Cost of Operations

Operation	Energy (pJ)
64b Floating FMA (2 ops)	100
64b Integer Add	1
Write 64b DFF	0.5
Read 64b Register (64 x 32 bank)	3.5
Read 64b RAM (64 x 2K)	25
Read tags (24 x 2K)	8
Move 64b 1mm	6
Move 64b 20mm	120
Move 64b off chip	256
Read 64b from DRAM	2000

Some Observations (1) Instruction Supply

- Reading an instruction from a cache (33pJ) is $\frac{2}{3}$ the cost of a DP FLOP (50pJ).
- DP FLOPs are only $\frac{1}{3}$ of operations in typical code.
- Reading instructions from cache is 2x the energy of a typical code mix (1 FP OP + 2 INT Add).

Observations (2) Local Data Supply

- Accessing 3 operands from registers (10.5pJ) is 20% of a FLOP but 10x the cost of other instructions.
- Reading 3 operands from local cache (100pJ) is 2x the cost of a FLOP.
- Reading 3 operands globally on chip (20mm away) 360pJ+100pJ is >9x cost of a DP FLOP
- Reading 3 operands off chip 762pJ is >15x cost of a DP FLOP
- Reading 3 operands from DRAM 6nJ is 120x the cost of a DP FLOP

Summary of Observations

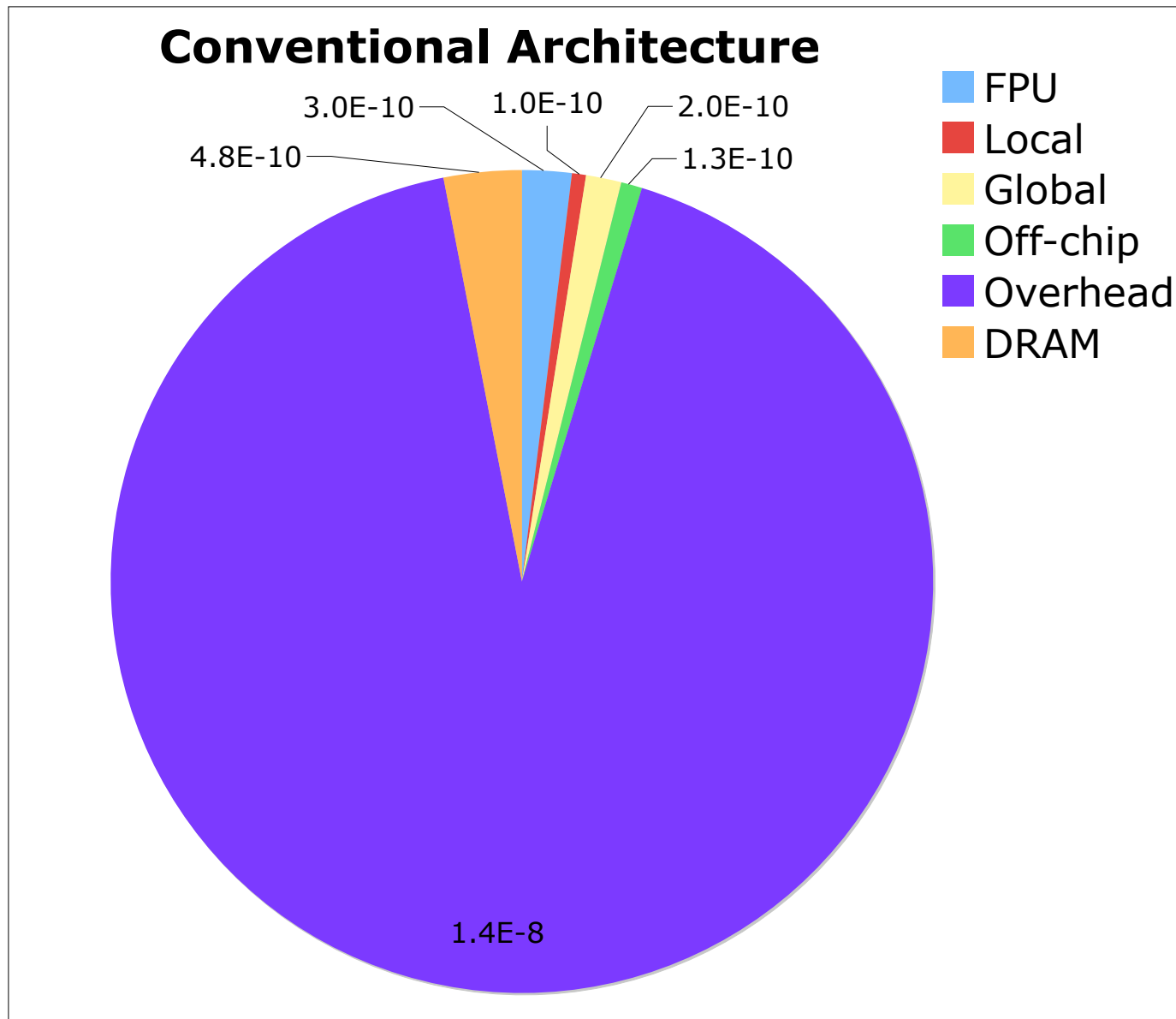
Operation	Energy (pJ)	DP FLOPs	Insts*
I\$ Fetch	33	0.67	2.0
Register Access (3W)	10.5	0.2	0.6
Access 3 D\$	100	2	6
Access 3 L2 D\$	460	9	27
Access 3 off chip	762	15	45
Access 3 from DRAM	6000	120	360

Energy dominated by data and instruction movement.

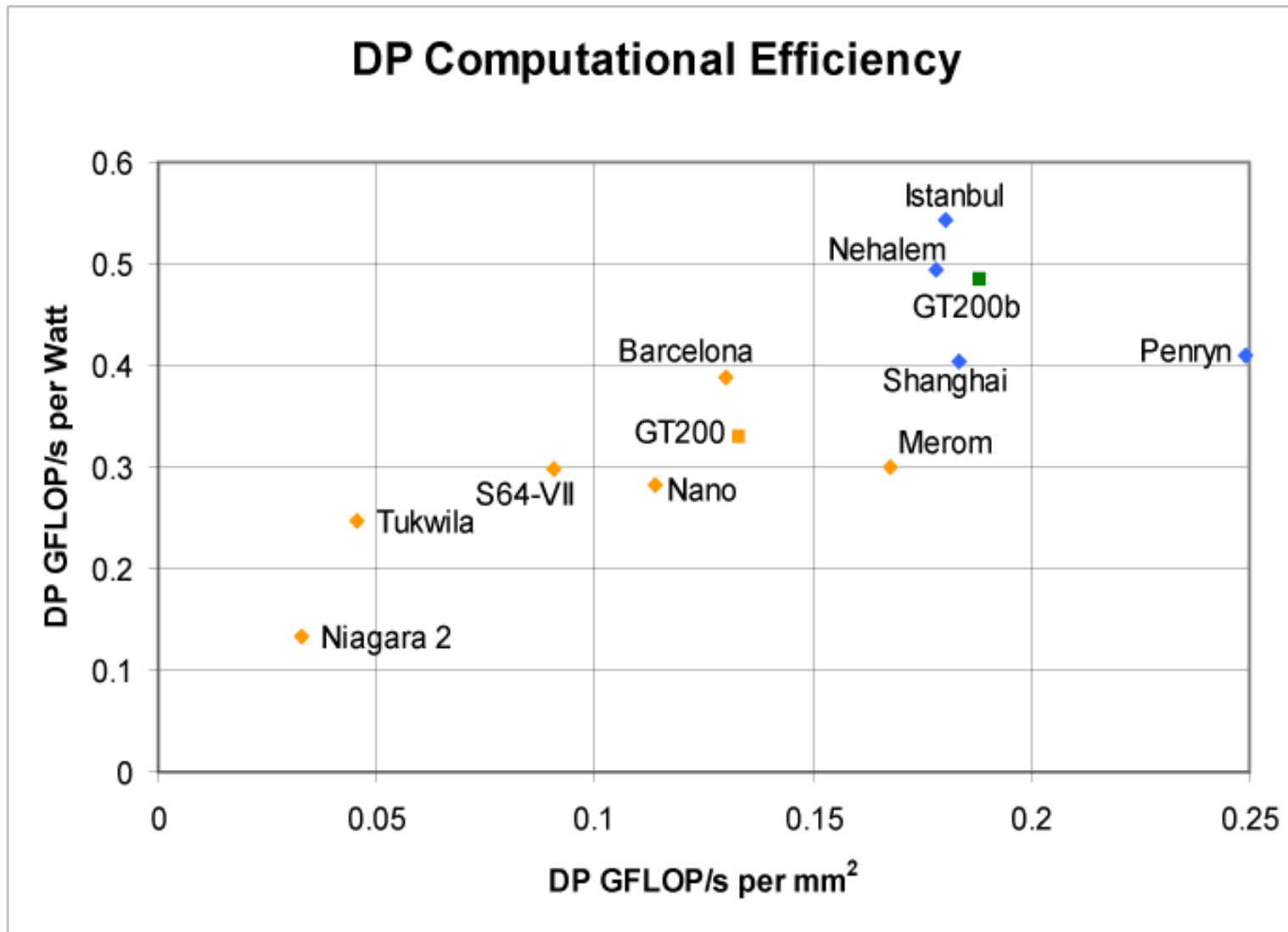
* Insts column gives number of average instructions that can be performed for this energy.

Conventional Architecture (90nm)

Energy is dominated by overhead



Efficiency of Conventional Processors



From <http://www.realworldtech.com>

Many of these would be much lower without SSE

Why so much overhead?

- Complex control to optimize single-thread performance.
 - Register renaming, branch prediction, speculative execution, ILP, complex decoding (x86), ...
- Much of complexity is
 - Complex ways of dealing with high memory latency.
 - Hiding parallelism from the programmer.
 - Denial architecture – hiding the facts of memory hierarchy and parallelism.

Power-Efficient Supercomputing: Goal

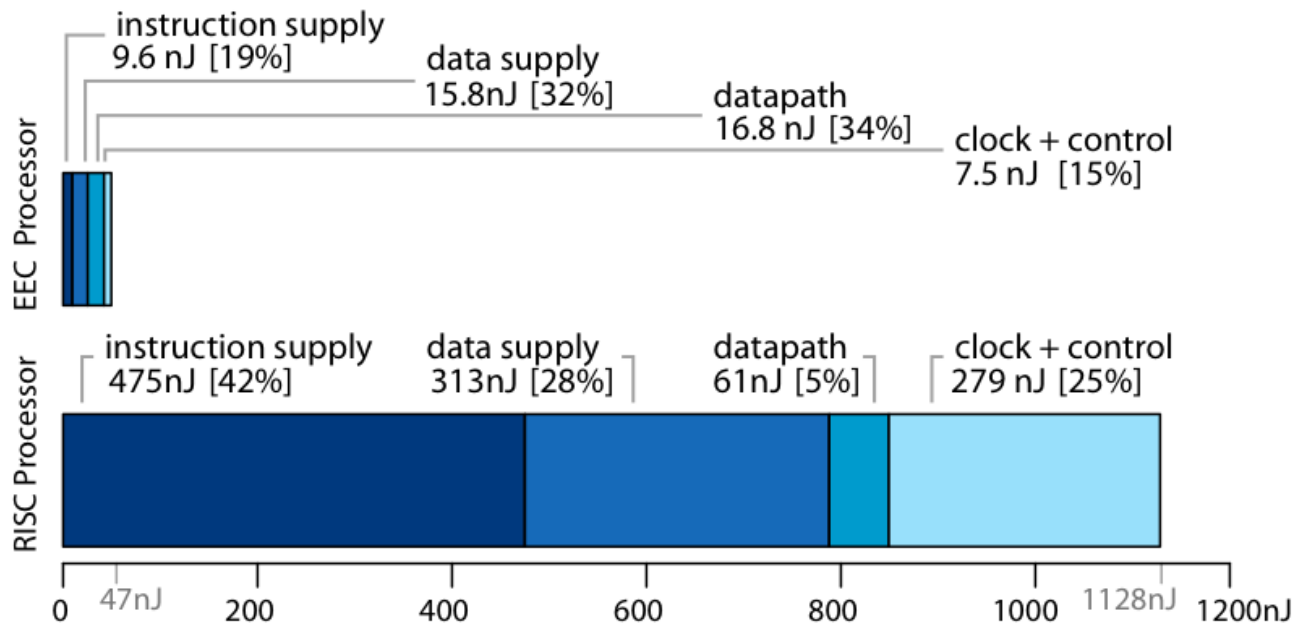
- 200pJ/FLOP (5GFLOPS/W) sustained
- 25% of energy in FPU

Power-Efficient Supercomputing: Approach

- Eliminate overhead
 - Hide latency explicitly
 - Simple control
- Energy-optimized architecture
 - Efficient data supply
 - Efficient instruction supply
 - Reduce the number of instructions
 - Agile memory hierarchy
 - Over-provisioned architecture
- Optimized components
 - Low-energy interconnect
 - Low-energy memories

Borrows from our ELM Project

- ELM Processor Architecture
 - Efficient data and instruction supply
 - Demonstrated 30x energy efficiency vs SPARC Leon core



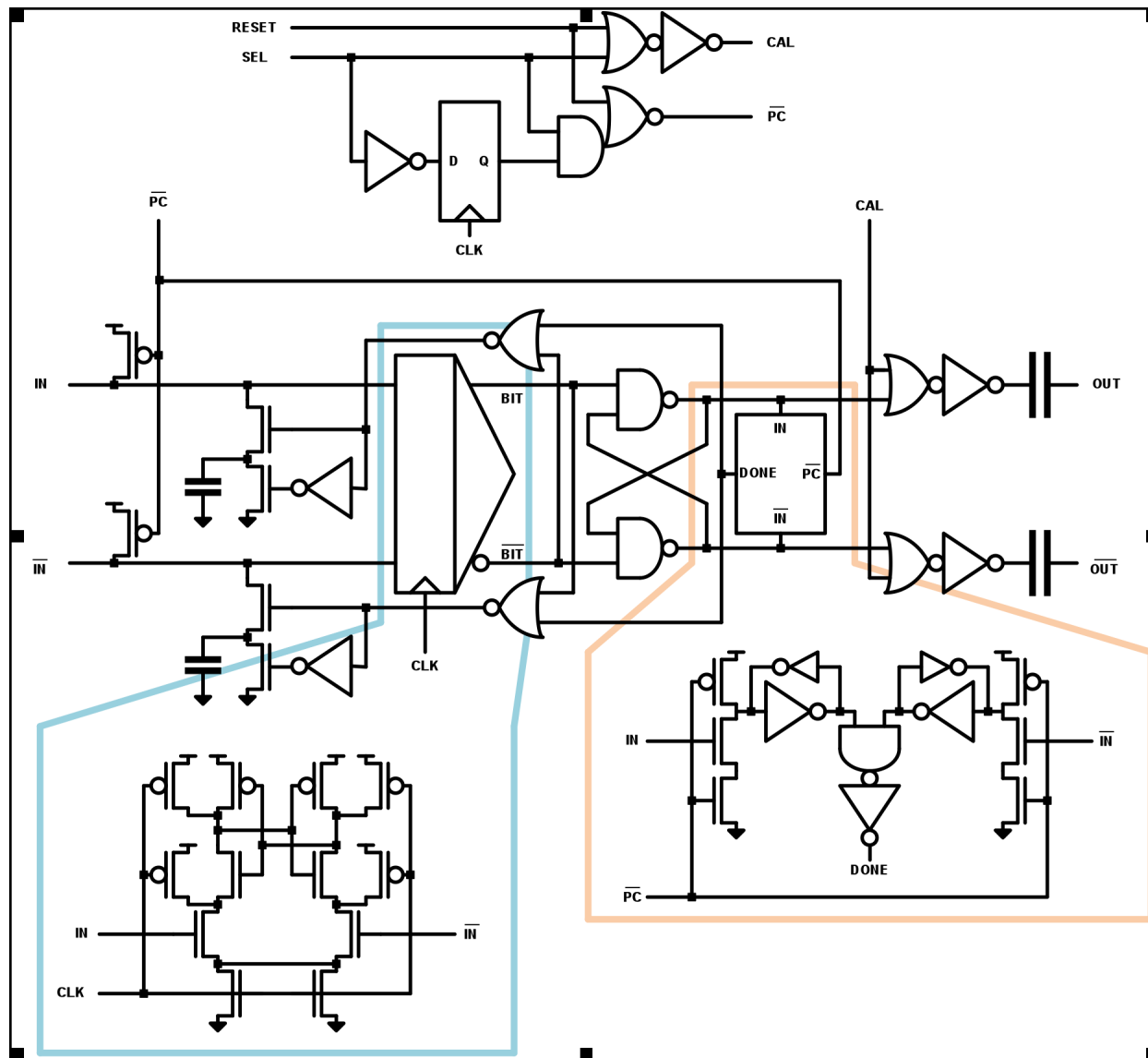
ESC is a similar problem

- Easier because DP FLOPs use more energy than integer operations.
- Harder because workload has less locality.

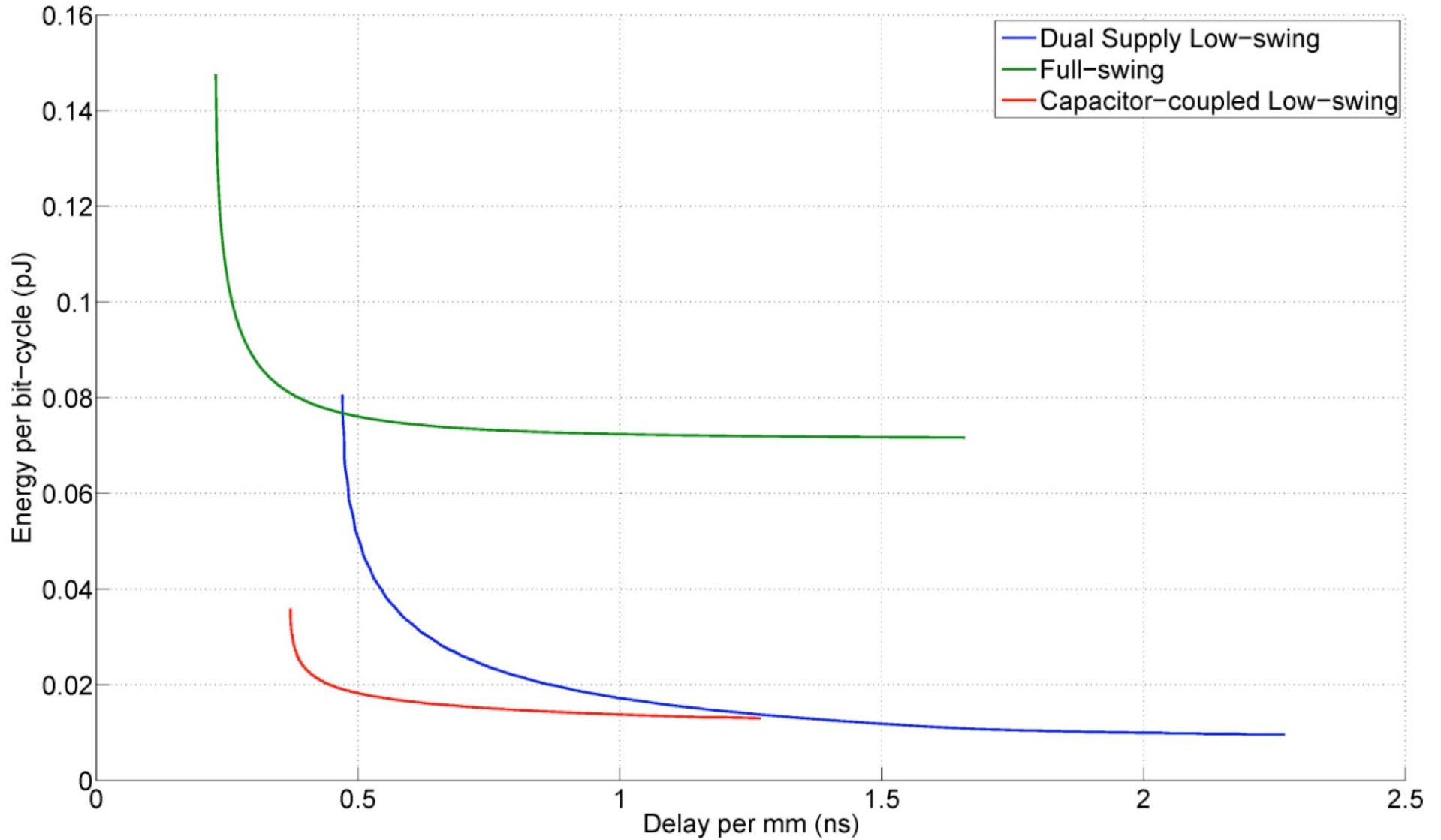
Efficient Circuits

- Static CMOS is good at making efficient function units.
- We can substantially improve on data transport and storage circuits.
- Data transport
 - On-chip low-swing capacitively-coupled circuit
 - Off-chip, efficient I/Os
- Storage
 - Energy-optimized RAMs – minimize capacitance switched
 - RAMs with low-swing write

Low-Swing Capacitively-Coupled Channel Circuit with Offset Cancellation

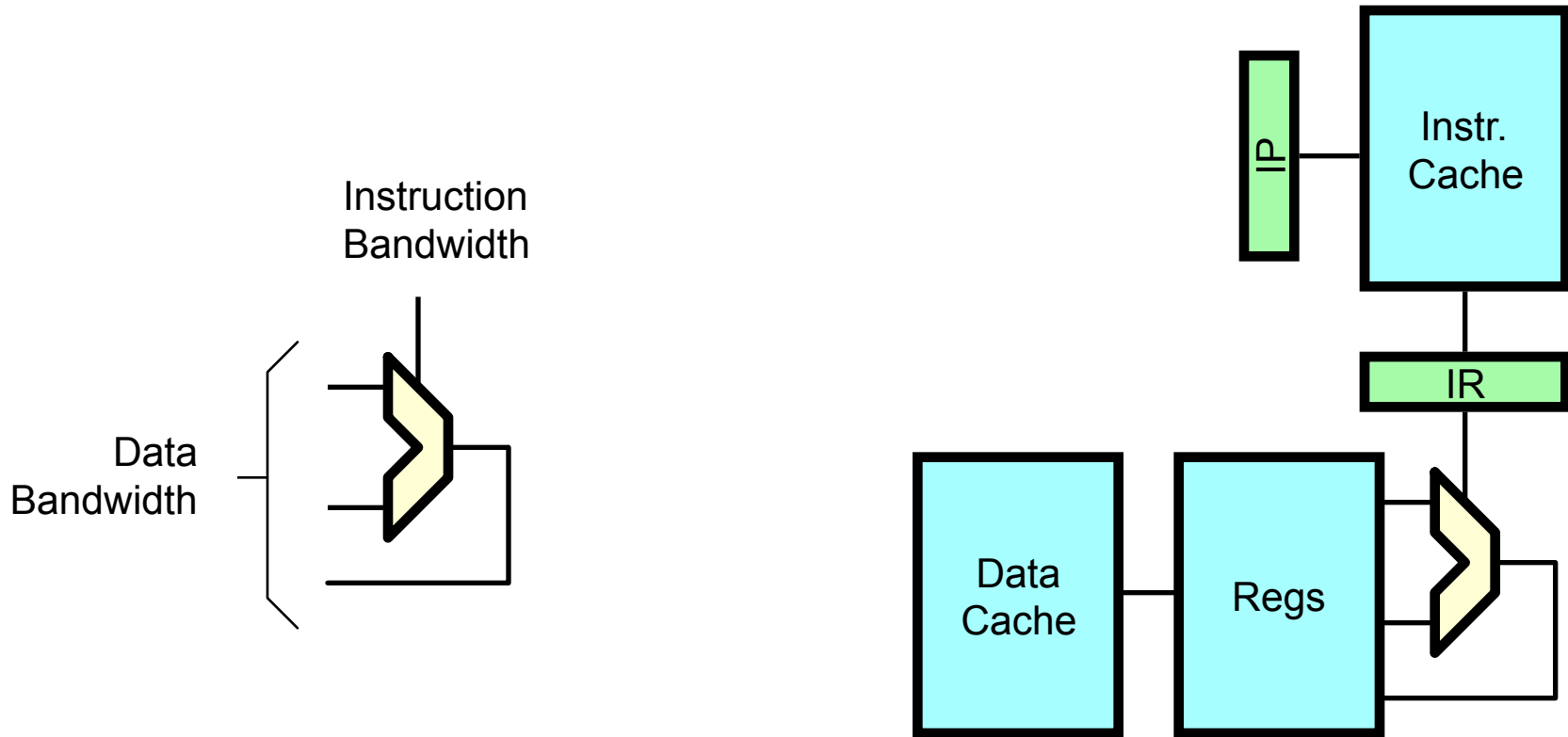


Comparison of Signaling Methods



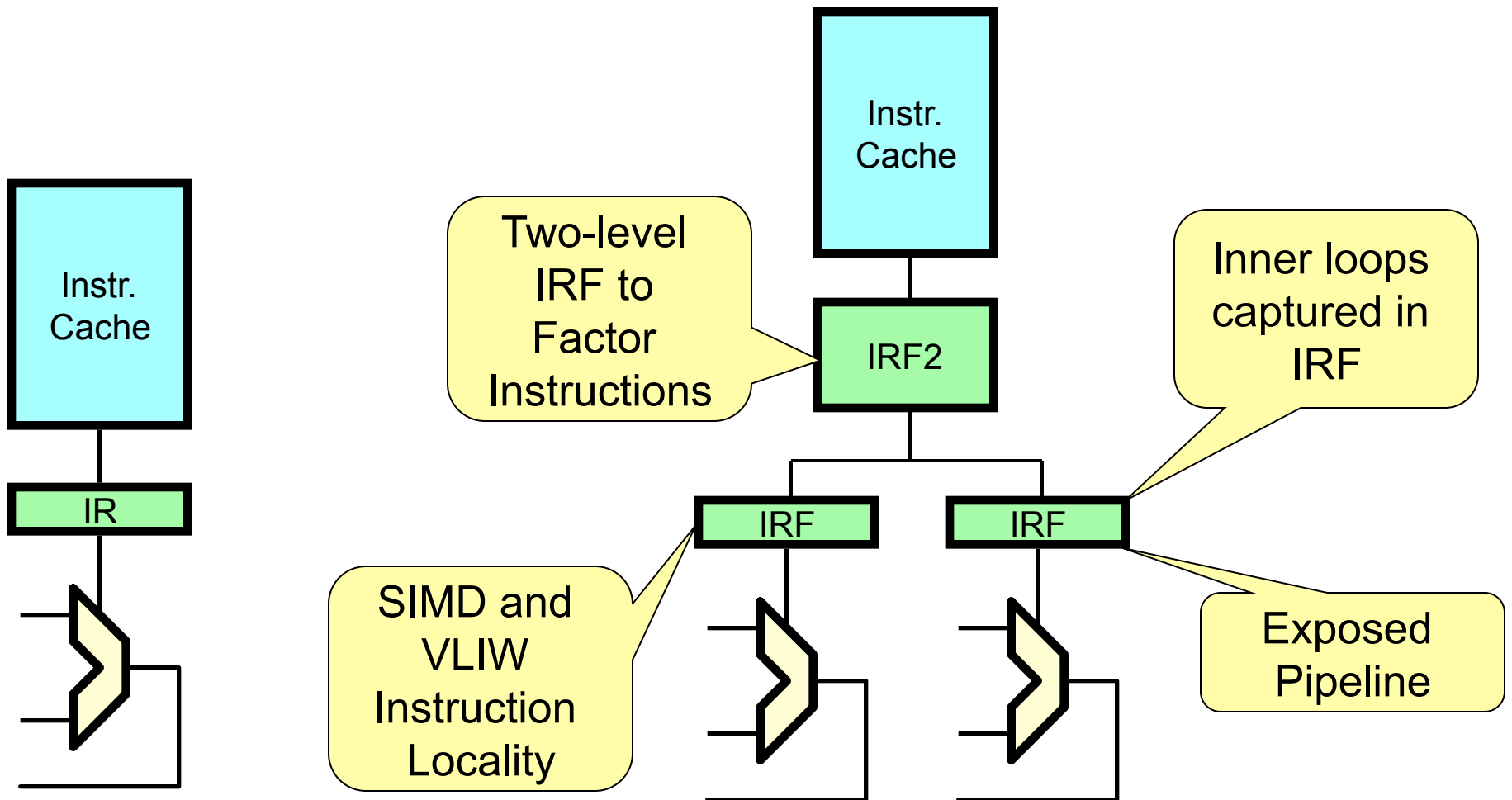
Instruction and Data Supply

Care and Feeding of ALUs



'Feeding' Structure Dwarfs ALU

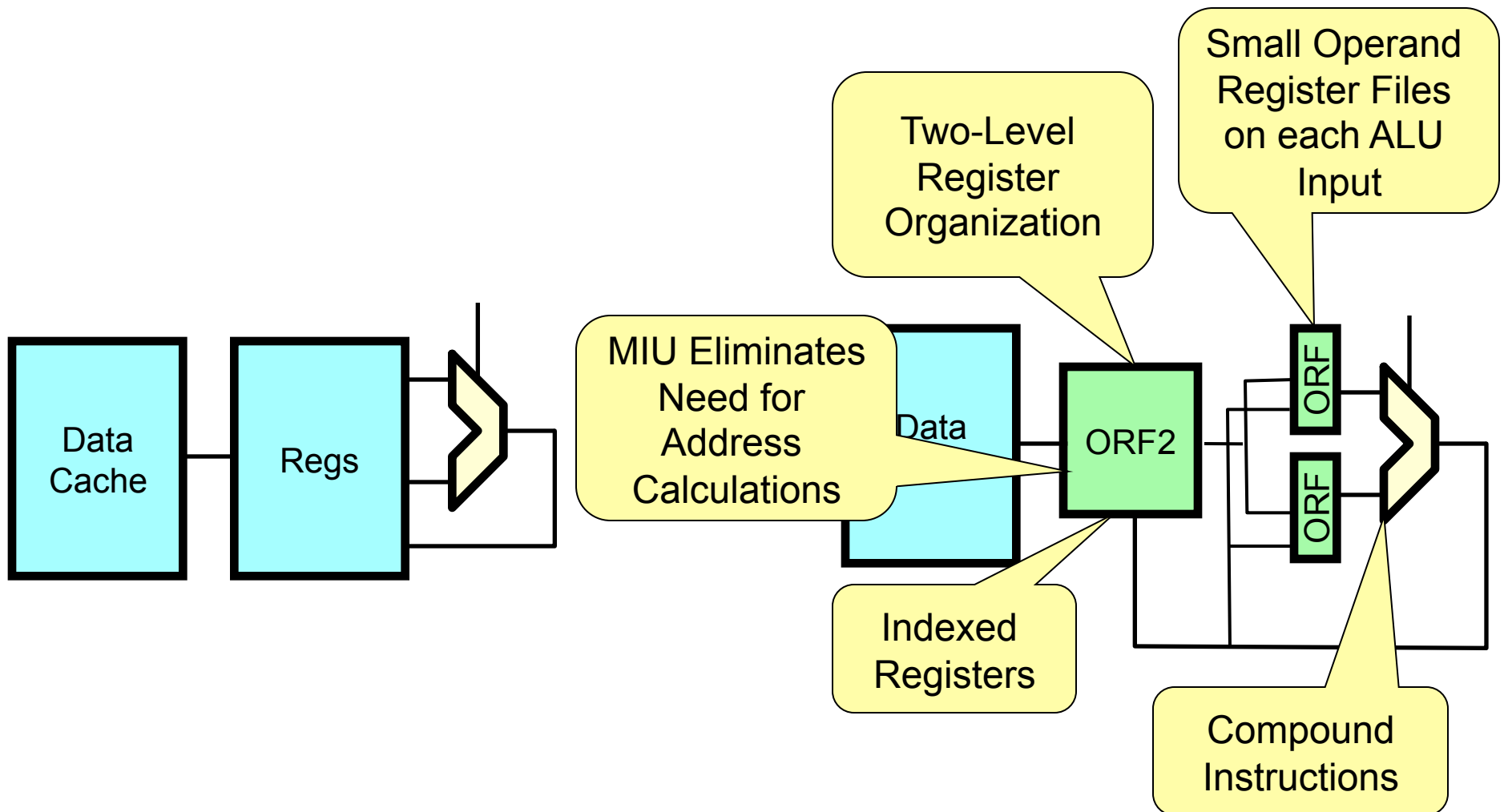
Efficient Instruction Supply



Efficient Instruction Supply

- Most instructions delivered from IRF (or small L0 I\$)
3.5pJ/op vs 33pJ/op.
- MIMD execution for irregular code segments.
- JSIMD to amortize instruction fetch in data parallel code segments.
- VLIW amortizes control and exploits ILP.
- Simple in-order pipeline
 - about 2.5pJ/op for control vs nJ.
- Average instruction supply energy determined by reuse at each level.

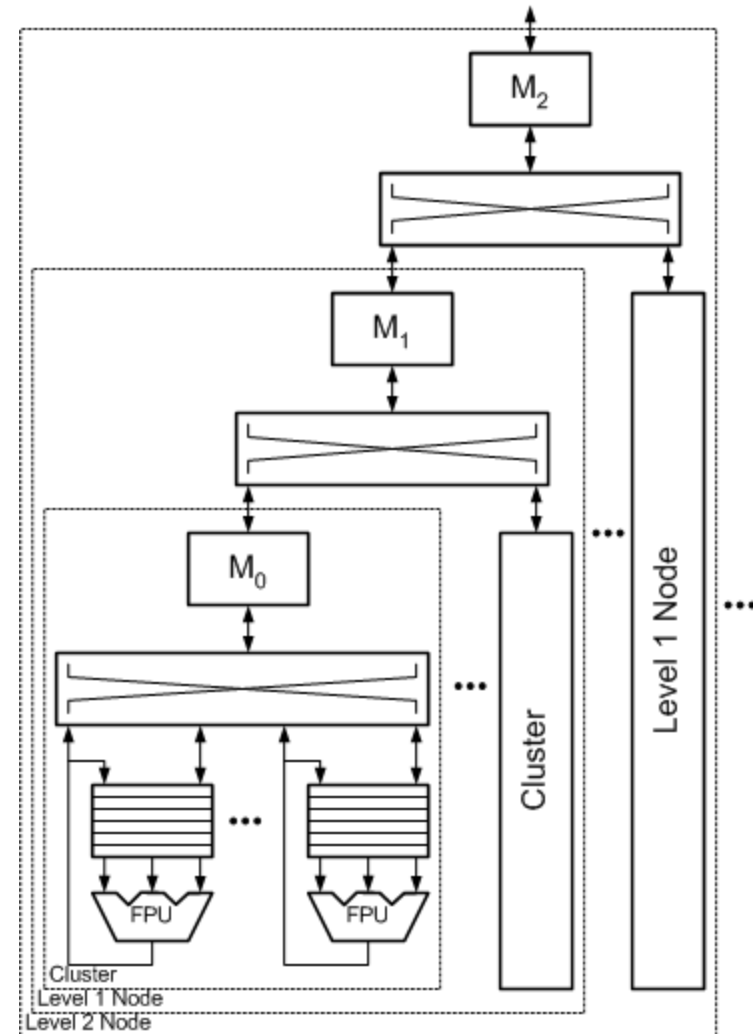
Data Storage and Movement



Agile Storage Hierarchy

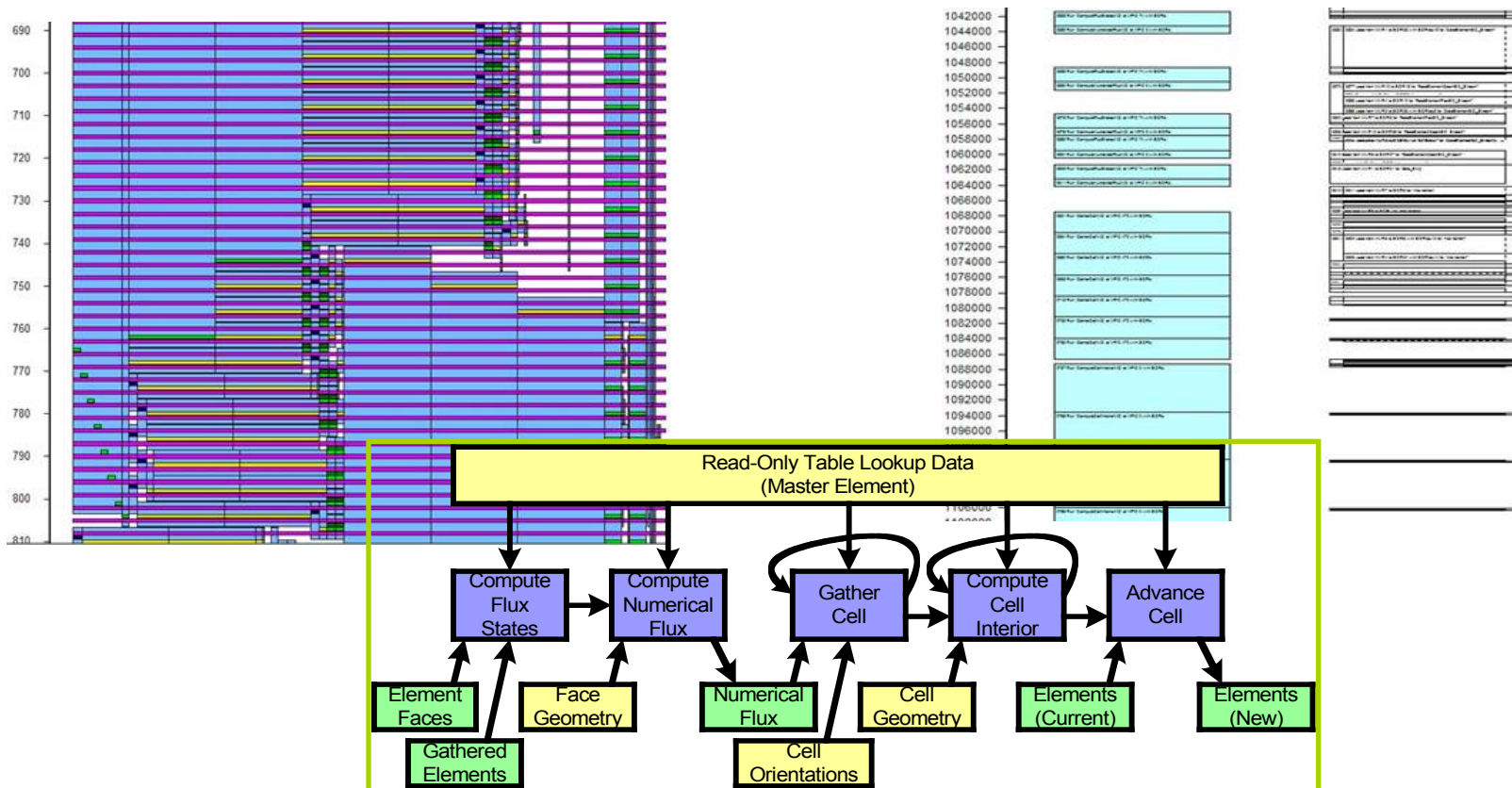
Agile Storage Hierarchy

- RFs, L0, and L1 local to each processor.
- Higher levels constructed from array of memories embedded in NoC.
 - L2, L3 different way of viewing NUCA memory
- Each level can be managed explicitly, as a cache, or both.
- Bulk transfers with gather/scatter between levels.
- Programmer expresses abstract locality
- Compiler/Run-Time/Autotuner optimize data movement.



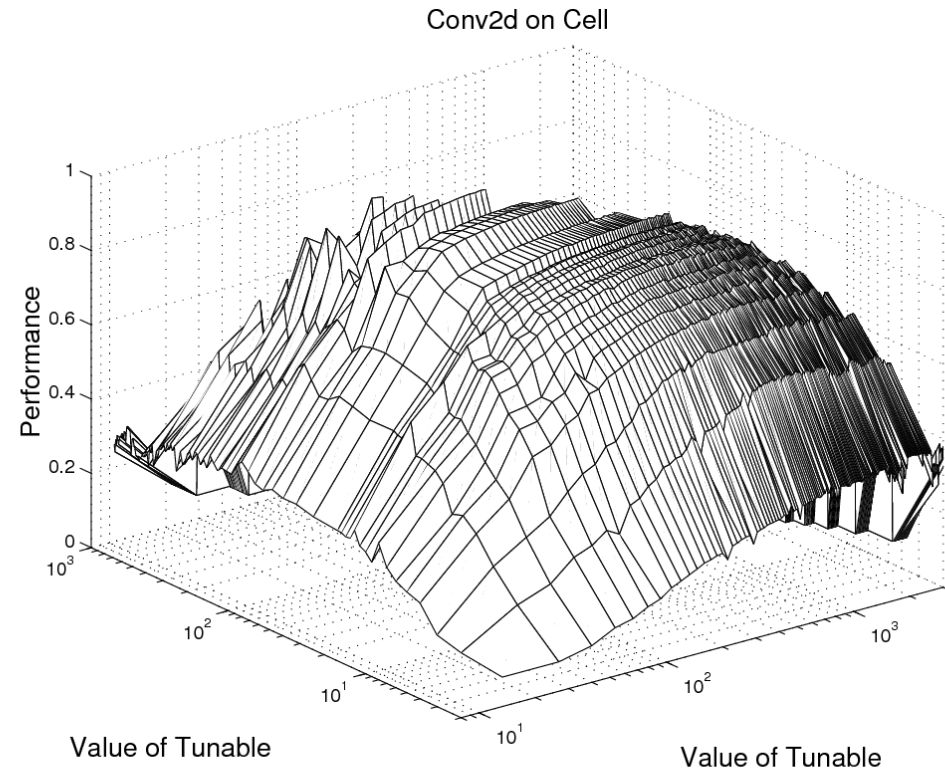
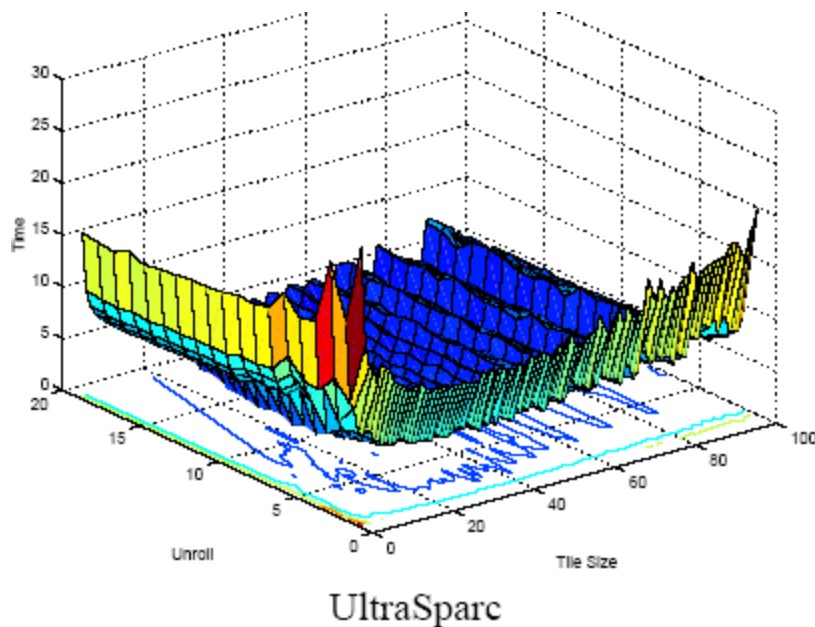
Agile Memory Enables Optimization

- Merrimac example
- Explicitly managed data movement
 - Reduces demand, increases utilization



Explicitly Managed Memory Gives Smooth Optimization Surfaces

Execution Time of Matrix Multiplication
for Unrolling and Tiling



T. Kisuki and P. M. W. Knijnenburg and Michael F. P. O'Boyle
Combined Selection of Tile Sizes and Unroll Factors Using Iterative Compilation.
In IEEE PACT, pages 237-248, 2000.

Performance of Auto-tuner

		Conv2D	SGEMM	FFT3D	SUmb
Cell	Auto	96.4	129	57	10.5
	Hand	85	119	54	
Cluster	Auto	26.7	91.3	5.5	1.65
	Hand	24	90	5.5	
Cluster of PS3s	Auto	19.5	32.4	0.55	0.49
	Hand	19	30	0.23	

Measured Raw Performance of Benchmarks: auto-tuner vs. hand-tuned version in GFLOPS.

For FFT3D, performances is with fusion of leaf tasks.

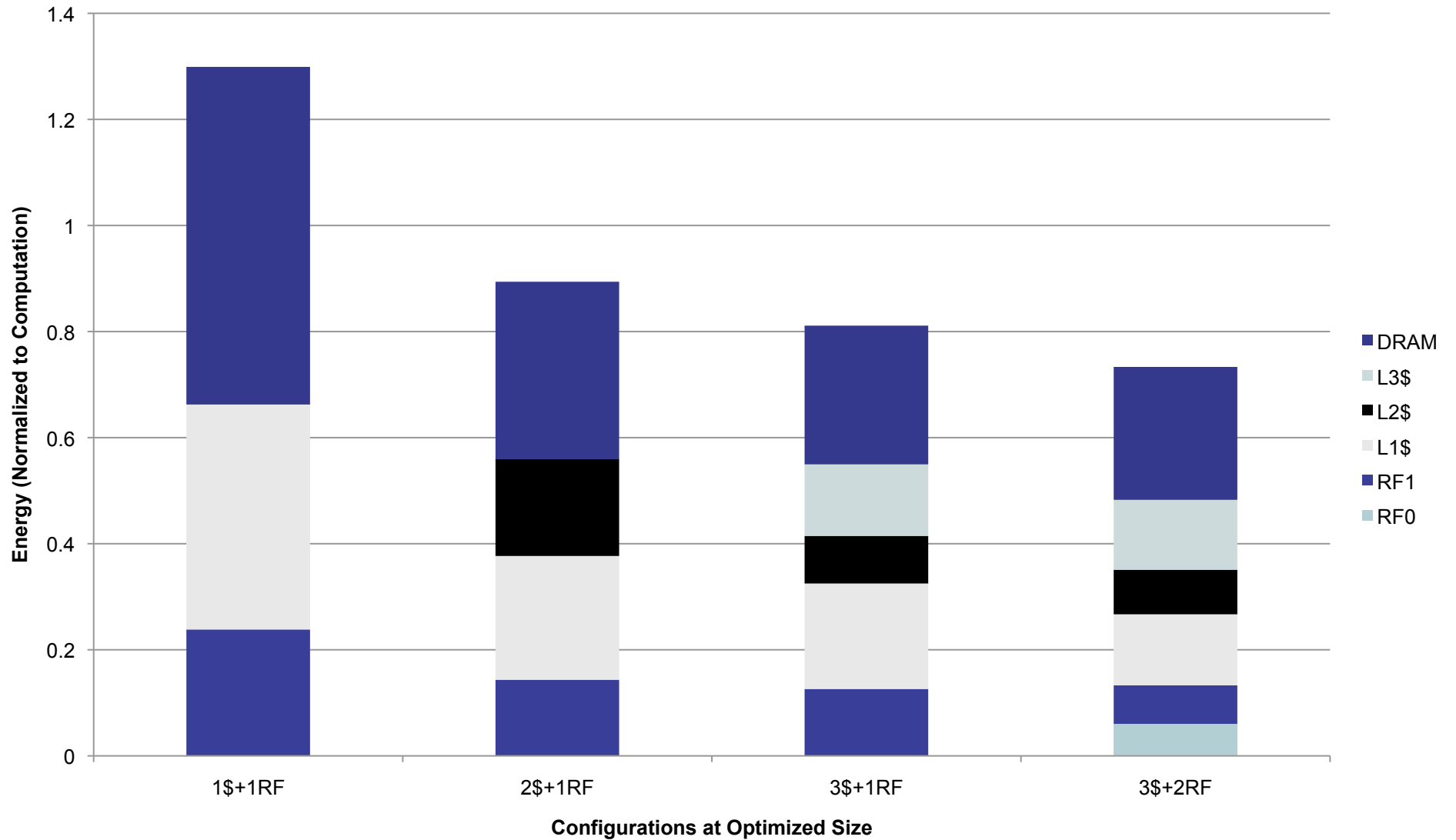
SUmb is too complicated to be hand-tuned.

Optimal Blocking of DGEMM

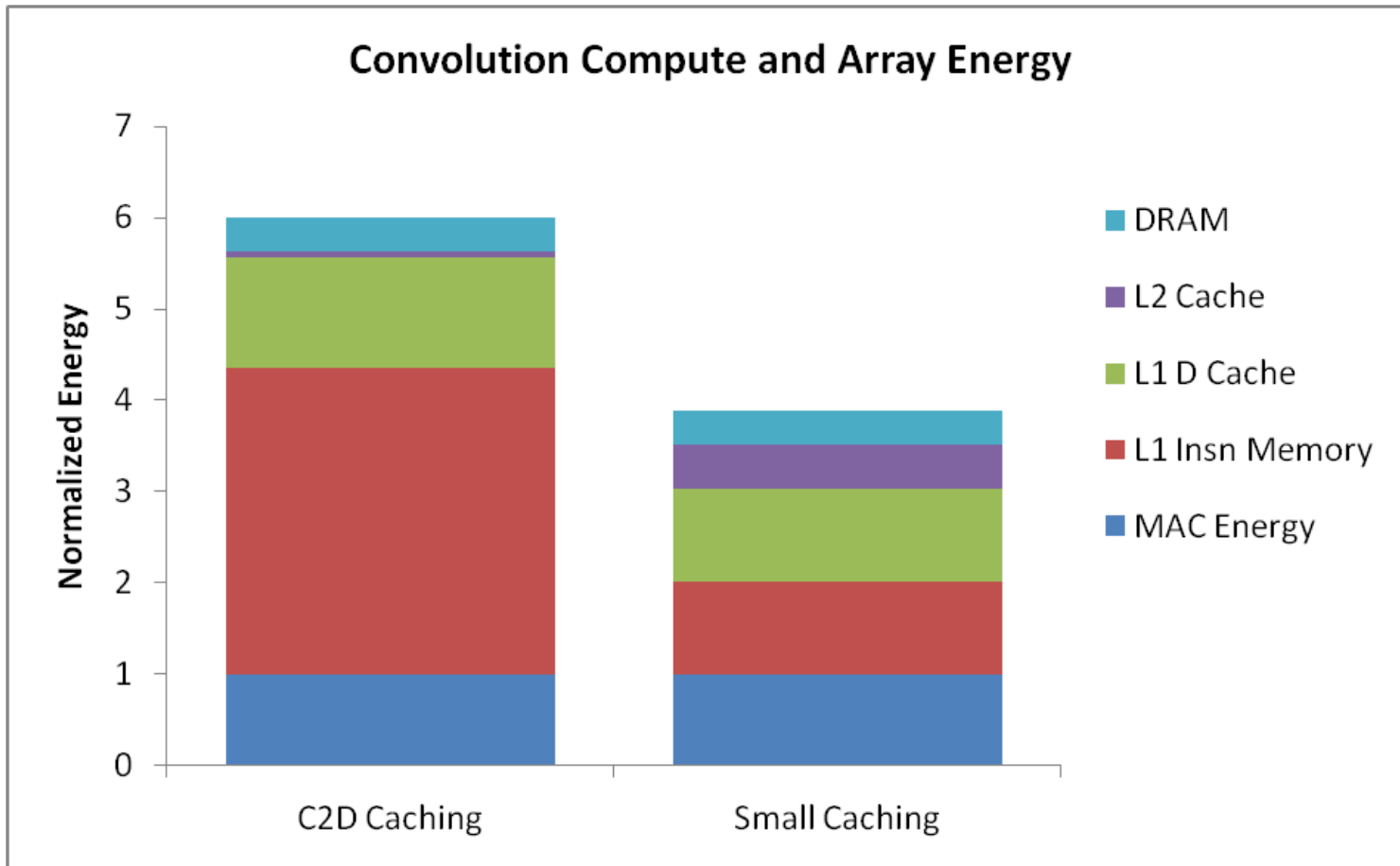
- Block for each level of the memory hierarchy: size is approximately 1/3 of level size ($A+B+C=1$)
- The lowest level register file has $\sim 3N^3$ access for N^3 flops
 - Each additional level has $3N^3/\text{sqrt}(C_{i-1})$ accesses, including the DRAM
 - Using this formula, energy can then be optimized

Size in 64b Words	RF0	RF1	L1	L2	L3
1RF + 1\$	-	110	24503	-	-
1RF + 2\$	-	61	5229	88685	-
1RF + 3\$	-	52	2377	23967	145100
2RF + 3\$	18	146	3736	29551	158111

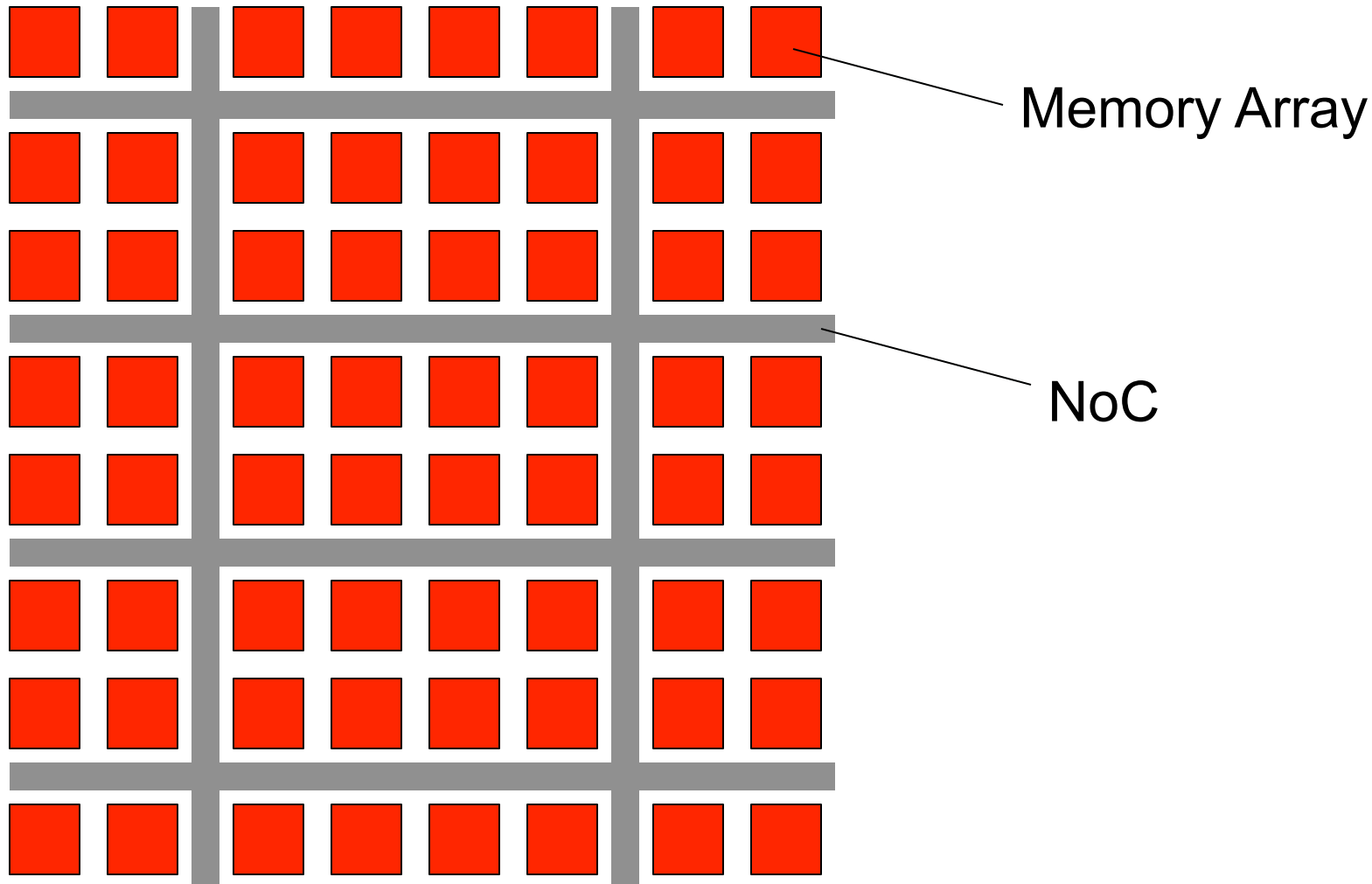
Cache Levels and DGEMM



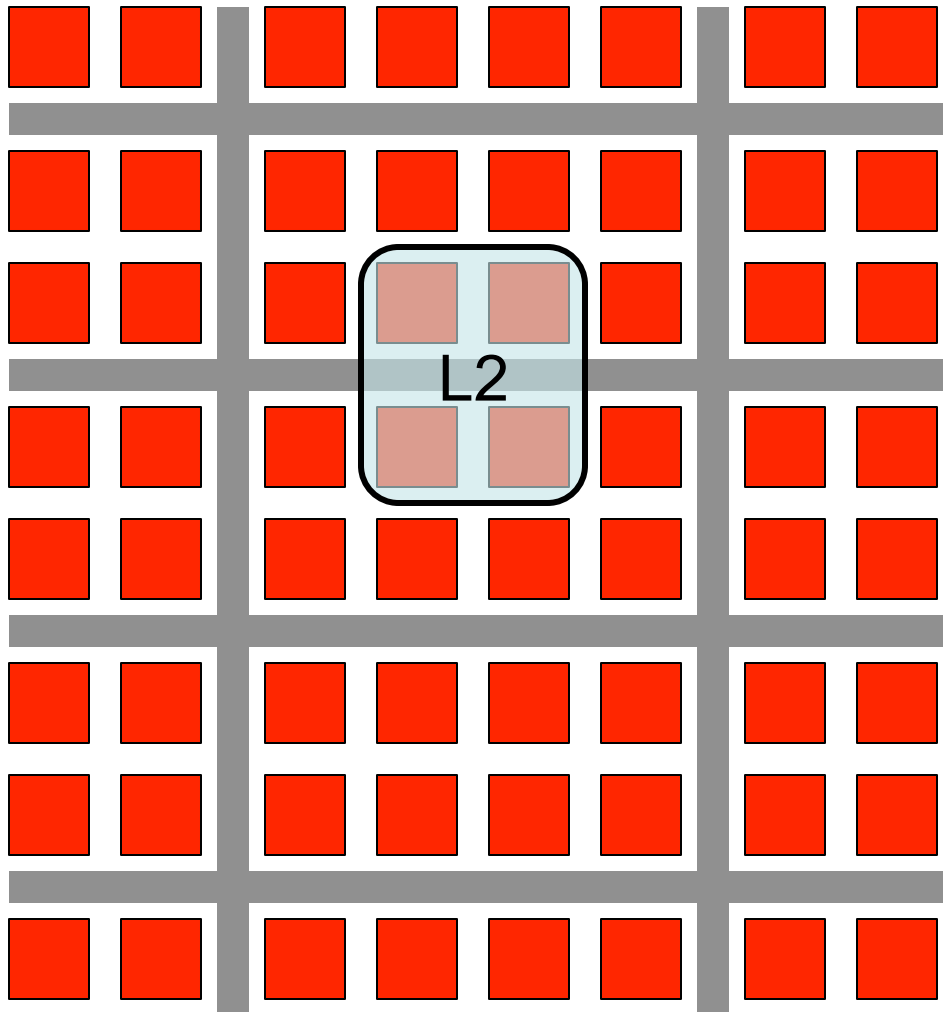
Cache Optimization for Convolve



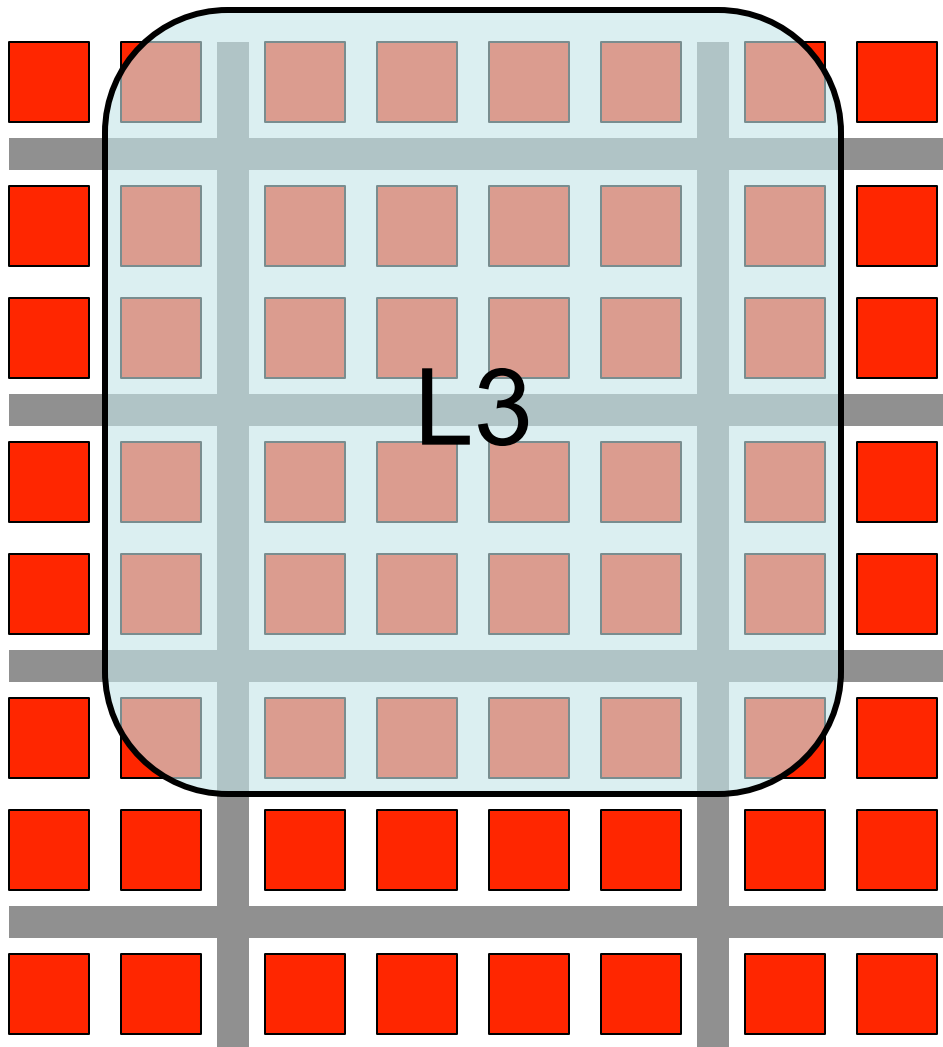
Number of Levels and Sizes Can be Optimized at Runtime



Number of Levels and Sizes Can be Optimized at Runtime



Number of Levels and Sizes Can be Optimized at Runtime



HW Provides:

- Array of memory arrays
- NoC for communication
- Data transfer engines
- Small set of primitives

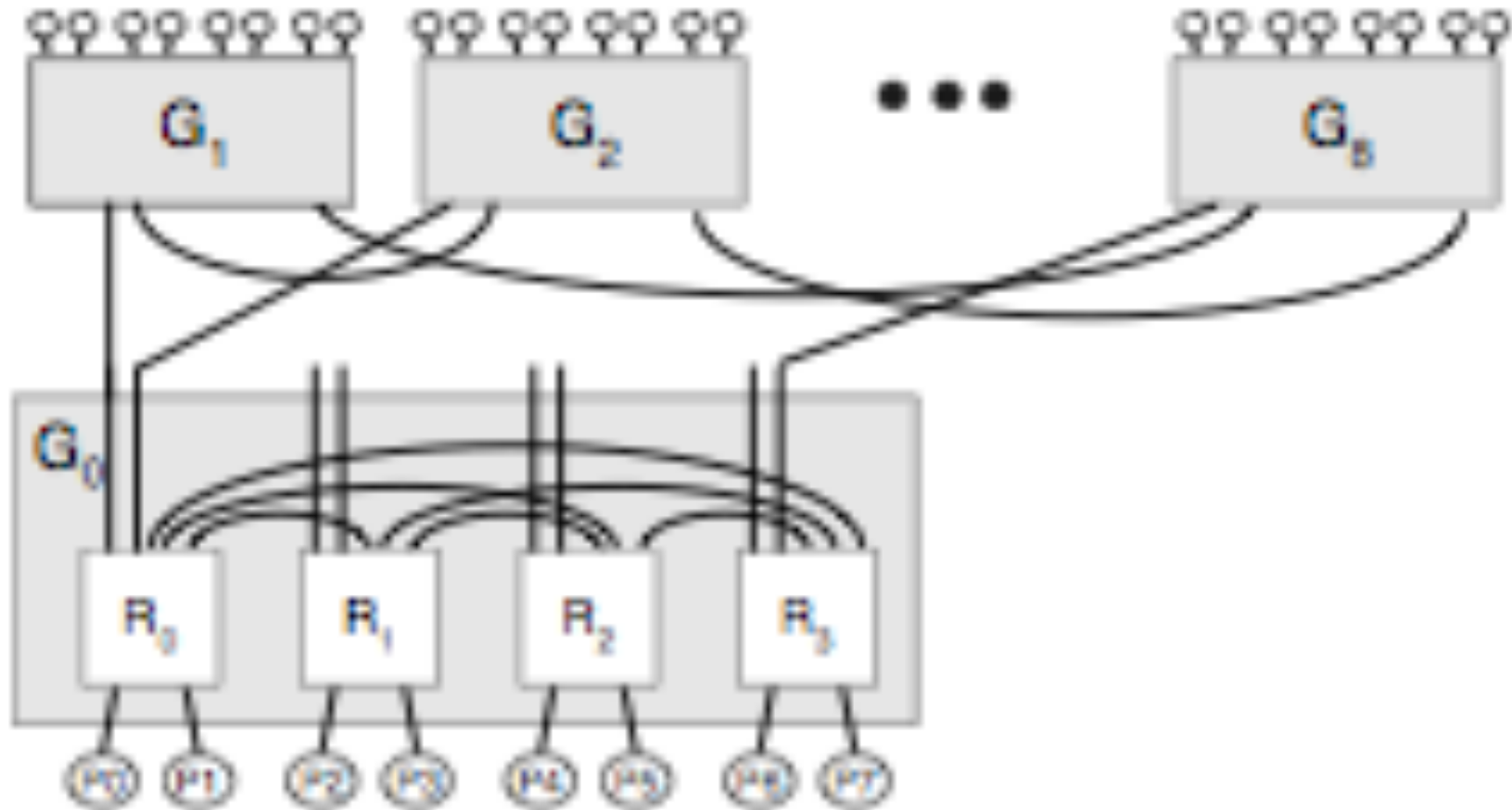
SW defines

- Mapping
- Replacement
- Coherence

From Chips to Systems

Efficient Network (e.g., Dfly)

Global address space
Message-driven execution



Closing Remarks

GPU Computing

- GPUs are already much of the way to being efficient supercomputer nodes
 - Simple control
 - Explicit control of storage hierarchy
 - Large fraction of energy goes to FLOPs
- Future GPUs will close the gap
 - More efficient instruction and data supply
 - Agile memory
 - Seamless integration into larger machines
- 5GFLOPS/W and beyond – soon

Conclusion

- Supercomputers are energy limited.
 - On the chip and in the data center.
- Conventional processors are very inefficient
 - 15nJ/op in 90nm 5nJ/op in 45nm
 - Largely due to overhead
- Fundamentally, energy is dominated by data and instruction movement.
- Efficient supercomputing
 - 200pJ/op in 45nm
 - Efficient communication and memory circuits
 - Efficient data and instruction supply
 - Agile memory system