Computational RAM: The case for SIMD computing in memory

Duncan Elliott
University of Alberta

Michael Stumm
University of Toronto

Martin Snelgrove
Carleton University

Building SIMD processors into memory at the sense-amps.
URL http://www.ee.ualberta.ca/~elliott/cram/

June 1997

elliott@ee.ualberta.ca (403)492-5357 fax (403)492-1811

Abstract

DRAMs contain a very wide datapath internal to the chip which can deliver orders of magnitude greater memory bandwidth inside the chip than through the pins. The dilemma lies in deciding what to connect to the other end of this wide pipe to memory. We compare two alternatives: microprocessors (including MIMD multiprocessors) and SIMD processors. Microprocessors are better accepted and are suited to more applications. SIMD processors can better utilize the memory bandwidth available internal to the DRAM chips. These approaches can coexist.

Move the Processors into the Memory

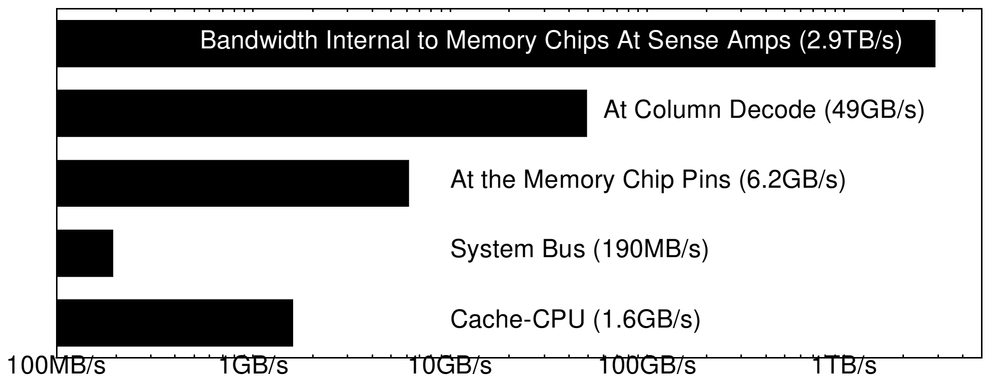
- "Processors in Memory" are gaining momentum
- Higher memory bandwidth, compact, energy efficient

Options:

- Microprocessor + DRAM
  wide path to memory for cache line fills
  potential for high degree of interleaving to support multiple outstanding reads
  familiar programming model
- Vector processor
  fast vector loads, stores; particularly for "page-mode" accesses
- MIMD
- SIMD (Single Instruction stream, Multiple Data stream)
  highest bandwidth utilization
  low overhead (our approach: bandwidth rich, PE complexity poor)
  unfamiliar programming model
- Hybrid: Microprocessor + SIMD + DRAM

Bandwidth in a Current Workstation

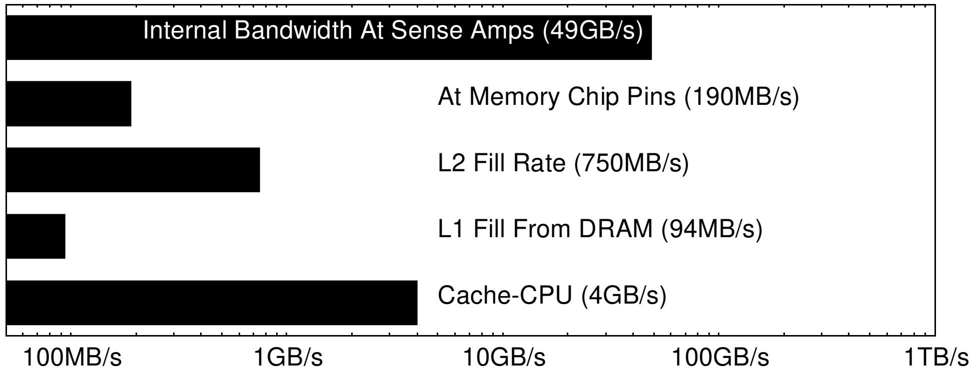
- 3 orders of magnitude more memory bandwidth inside the chip
  (256MB of 16Mb 50ns page mode DRAM, 64bit bus, 200MHz CPU)



- Video RAMs, Synchronous Mode DRAMs, and RAMBUS all clock the data at higher rates but are still pin limited
- C•RAM exploits a 2Mb datapath without going off-chip

Bandwidth in a Future Microprocessor + 1Gb DRAM

- 128MB, 85ns cycle time, 4.5ns interface, 500MHz processor (1 data port)
  1 word line for L1 cache, 8 word line for L2 cache, random row accesses



- Single level cache for a high performance microprocessor is ambitious
- Internal memory bandwidth grows ~ with the square root of memory density
- Still an order of magnitude more bandwidth at the sense amps than the microprocessor could consume

Low Power Processing

- More bits used per memory row-decode operation
  64 to 2048 times as many
- PEs use 10-25% of total chip power
- Computation with out data driven off chip

example applications:

- Image processing for a Digital Camera
- Multi-Media for a Laptop computer

What is the Question to be Solved?

How can the memory bandwidth to the processor(s) be increased?

- Tap into 2Mb wide data path at the sense amplifiers.

How can 'smarts' be added to memory while maintaining density?

- Put PEs along one edge of 2D memory array without affecting the fundamental memory structure.

How can the most cost-effective computing power be obtained?

- Memory has the greatest fraction of component cost in almost all computers. The processor must meet the memory on the memory's terms. Also amortizes instruction decode over more processors and reduces the system pin count.

All have the same answer.

These assume the application has the parallelism to be exploited.

Why SIMD?

- Doesn't SIMD limit what kinds of applications can be run?
  Yes
- Don't some PEs sit idle in conditional code?
  Yes
- High performance per transistor (per silicon area, per volume, per $)
  Build in more peak performance to offset utilization.
- Memory structure provides a long word (e.g. 1Kb - 4Kb)
  Don't throw away bandwidth by using part of word
  Common row address, can't use MIMD
  Don't change memory architecture and add overhead
  $\Rightarrow$ Use SIMD
- Simple synchronization and communication
- Shared instruction decode allows a 75 transistor PE

What is C•RAM?

- Computational RAM (C•RAM) = RAM + SIMD
  Single Instruction stream Multiple Data stream processor
- Processing Elements (PEs) at sense amps
  Small area (3-20% increase)
  Big performance
  (e.g. 32MB yields 13,000 MIPS)
- “Shared memory” interface to host processor
- SRAM Chips: 64 PE in 8Kb, 512 PE in 240Kb
- DRAM Designs at the 4Mb and 16Mb generations

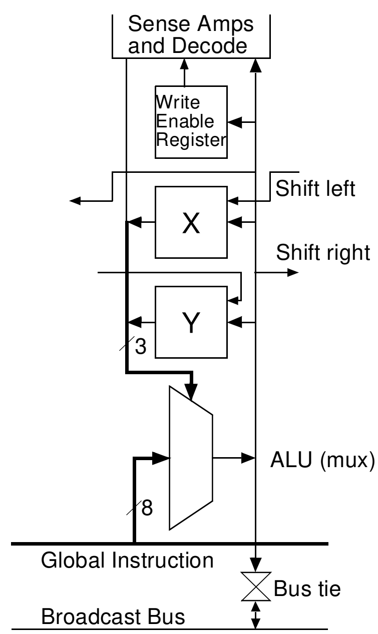
PE Architecture

Processing Element

- ALU is a multiplexer
  Memory and 2 registers as inputs
- Result to registers or memory
- Write enable permits conditional operations
- Bit-serial arithmetic
- Registers double as left-right communication
- Bi-directional driver to Broadcast Bus
- Instructions fed from off chip

Memory

- Conventional memory interface is maintained for host
- Row address is local memory address
- Each PE sees its column(s) as local memory



Processor Resource Utilization

If applications have a massive degree of parallelism and are suited to SIMD computation there are cost/performance advantages to using more copies of a simpler processor whereby those processors are more thoroughly utilized.

The C•RAM PE has:

- One ALU always in use
- Single stage of logic
- Minimalist 75 transistor implementation
- Small register set and no cache since it operates at memory speeds
- More efficient arithmetic algorithms

Asymptotic Advantages of Bitserial Arithmetic

hardware algorithm	datapath width	trans-istors	area	time	area-time product
	Addition				
bit serial	1	$O(1)$	$O(1)$	$O(n)$	$O(n)$
ripple carry	n	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$
carry save	n	$O(n)$	$O(n)$	$O(1)$	$O(n)$
carry save, binary out	n	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$
carry look-ahead	n	$O(n\log n)$	$O(n^2)$	$O(\log n)$	$O(n^2\log n)$
	Multiplication				
bit serial	1	$O(1)$	$O(1)$	$O(n^2)$	$O(n^2)$
ripple carry	n	$O(n)$	$O(n)$	$O(n^2)$	$O(n^3)$
carry save	n	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$
Wallace tree	n	$O(n^2)$	$O(n^2)$	$O(\log n)$	$O(n^2\log n)$
	Bit permutation				
bit serial	1	$O(1)$	$O(1)$	$O(n)$	$O(n)$
bit-wise	n	$O(n)$	$O(n)$	$O(n)$	$O(n^2)$
memory look-up	n	$O(n2^n)$	$O(n2^n)$	$O(1)$	$O(n2^n)$
Batcher banyan	n	$O(n\log n)$	$O(n^2)$	$O(\log n)$	$O(n^2\log n)$

Performance

program	C•RAM run time	SUN Sparc run time	C•RAM speedup
Vector Quantization	25.7ms	33.8s	1312
Masked Blt	18.2µs	443ms	24310
3x3 Convolution 16M	17.6ms	113s	6404
FIR 128K tap, 40b	99µs	312ms	3144
FIR 4M tap,16b	1.04ms	5.14s	4929
LMS Matching	0.20ms	251ms	1253
Data Mining	70.6ms	192s	2724
Fault Simulation	89µs	3.9s	43631
Satisfiablility	23µs	959ms	41391
Memory Clear	1.6µs	8.8ms	5493

32MB of 150ns C•RAMs simulated vs. 70MHz Sparc executed
Thousands of times faster to compute in the memory than the CPU

Summary

- Memory bandwidth will continue to dominate computer performance
- Must tap into memory at highest bandwidth, at the bitlines or sense amps
- Must meet the most economical memory (DRAM) on its terms,
  No modifications to design rules or core structure
- Memory constrains the circuits which constrain the architecture
- Potential thousand-fold increase in performance/price of desk-top systems for applications with the necessary massive parallelism

- Microprocessor + DRAM is the next evolutionary step
  low cost applications first, high performance last
- SIMD + DRAM is the revolutionary path

- Last 15 years of RISC has produced some surprising changes in direction
- The next 15 year of computing in memory will hold some surprises, but computing is migrating there nonetheless