


11. Acknowledgments

MOSAID has hosted and supported this project for several years, and their staff have been generous in explaining the real-world constraints of DRAM to us. At various times Dick Foss, Peter Gillingham, Graham Allan, Iain Scott, Randy Torrance and David Frank in particular have offered insights that have shaped the project. Nortel has contributed financially and in fabrication for prototypes, some through the Canadian Microelectronics Corporation, and Nortel staff (David Somppi and Randy Gibson in particular) have also contributed, especially on ASIC memory issues.

The Natural Sciences and Engineering Research Council has funded this work through an Industrial Research Chair, operating grants, CMC and Micronet.

12. References


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

Table 2: Benchmark Speedups
10. Conclusions

The “Computing RAM” (C•RAM) is an integrated circuit that can function either as a conventional memory or as a SIMD processor. Functioning as memory is more a philosophy than a “feature”. For example, nothing with 200 pins can function as a memory chip in terms of the most critical specification: low cost.

Several manufacturers are experimenting with logic-in-memory, and others with SIMD in the CPU. Adding logic to memory is not a simple question of bolting together two existing designs, because memory and logic technologies have different characteristics and because a memory looks very different inside the chip than it does at the pins. Computing in RAM offers such huge speed and power advantages for image-processing and other large applications that the difficulties of redesign that this poses look likely to be overcome, offering ultra-low-cost image processing for consumers and long battery life without loss of capability for the rapidly growing portable-computing market.
serial computation and has left-right and wired-OR based communication. It can be implemented in only 75 transistors using dynamic logic, which allows it to be small in comparison to 2K transistors of “local memory” in a 2K × 2K device. The more complex 147-transistor version of Fig. 9 [9] was originally developed to complement a static RAM on an ASIC process, which has a much larger memory cell and in which memory cost dominates a larger processor. Its principal enhancements are that it has more registers to reduce the number of RAM cycles and that it has direct support for grouping adjacent processors to work on multi-bit data. In particular, the ripple-carry and registers allow substantial speed-up on bit-parallel multiplication.

![Fig. 8: A simple PE that can be implemented in dynamic logic with 75 transistors](image)

9. Benchmarks

A C++ “compiler/simulator” has been used that implements efficient C•RAM arithmetic and that allows us to count cycles for larger applications. Table 2 shows some representative timings for C•RAM simulated with a conservative 150ns cycle vs. a 70MHz microSPARC (measured). For these applications, the internal bandwidth of DRAM gives several orders of magnitude of speed-up over a conventional workstation.

Note that we do not claim this speed for all or even most applications: the Dhrystone rating of the SIMD processors would be very poor. The C•RAM philosophy is that largely sequential application belong on the host, and the massively-parallel component belongs in the memory. For many consumers, the massively-parallel (e.g. video/image processing and games) component is the smaller part of the value of the computer system (as compared to spreadsheets and word processors), and so C•RAM is designed for low cost rather than high feature-count.

Table 2 does not estimate power, but the energy improvements are related to the speed improvements because the PE power is smaller than sense-amplifier power — so a C•RAM-equipped computer consumes not much more power than one without, but finishes the task much sooner. In addition, the reduced need to pump images out over a bus and back should save power. C•RAM computation, however, uses more sense amps at once and typically has less effective memory caching, lessening the power advantage. Also, the memory access patterns of the parallel algorithm are not necessarily the same as the sequential algorithm.
7. DRAM Technology and its Effects on C•RAM

Cycle times in 16Mb DRAMS are typically in the range of 90 – 150ns, limited by the lengths of wordlines and bitlines. They can be made faster, but at a cost in density as the proportion of the chip area devoted to drivers and amplifiers rises at the expense of memory cells. In a given technology, a processing element has shorter lines and hence can cycle faster (and dissipate less power) than the DRAM array. This means that it is practical to interpose two processor cycles in each 110ns memory cycle thereby extending it, as suggested in Fig. 7.

Because it is not practical to construct a processing element in the single metal-metal pitch of a DRAM sense amplifier, processing elements are connected to at least four sense amplifiers and hence can have “fast page mode” access to at least four bits; the sense amplifiers form a primitive cache. This is of limited value if there is only a single “cache line”, since data will have to be very carefully arranged to avoid thrashing. If all four bits belong to one operand, for example, bitwise addition of two operands is guaranteed to force a DRAM cycle for each bit. The timing diagram of Fig. 7, for example, showed sum and carry computations and a new DRAM cycle. Adding even a second “cache line” would speed up two-operand functions (e.g. \( a = a + b \)) substantially. This can be done with a second bank of sense amplifiers (which are essentially static registers).

DRAM technology is quite different in its characteristics from the technologies usually used for processors [15]. In particular, DRAMs typically only have two layers of metal, because that is enough for a memory array. In contrast, high-performance processors use four or five layers. The difficulty is not technical, but economic: if the processor needs five layers of metal, then three of them are being bought and wasted over the entire DRAM array. A competing architecture that segregates processing and memory can then get a better cost for the (dominant) silicon area devoted to memory.

DRAMs also rely heavily (particularly early in the product life cycle) on the use of redundancy to get back yield that would otherwise be sacrificed to high densities and large dice. Processors, on the other hand, are usually designed without redundancy. This is acceptable for PEs, since they occupy a small fraction of die area and therefore will have limited effects on yield, but there may even be performance advantages to using redundancy, as that permits the use of more aggressive design rules [15]. The PEs in C•RAM can certainly be replaced on a column-by-column basis together with groups of bitlines, so most of the logic in C•RAM can benefit from the use of redundancy, but there is a problem in maintaining interprocessor communication. If a bad column is swapped out, then its connections to its neighbours have to be rewired too; alternatively, a simple switching network can be used to rewire the connections between PEs and local memory past a bad column [6]. The difficulties are only with column redundancy, since row redundancy is transparent to the PEs.

The use of redundancy for PEs is a research topic of its own, but one important point is that it would tend to favour very simple interprocessor communication schemes, where wiring around a bad processor is relatively simple.

8. C•RAM Architectures

The design space available for integrating processing with memory is very large, and even the low-cost SIMD corner that we have been exploring is very big. Two candidate PEs are shown in Fig. 8 and Fig. 9. The simpler of the two, Fig. 8, supports bit-
currents, and enormous power dissipation, if the pins were run at full speed. Any network efficient enough that it can move all 4k bits in or out of a chip in a cycle has to have this problem, so we are forced to accept limited interprocessor communication, and should choose something that gives the best fit to typical applications in a limited number of pins.

Even 2-D interconnect is expensive: if 4k PEs are arranged in a $64 \times 64$ square, the periphery has 256 processors that have to be connected to neighbours. We favour 1-D structures (rings, buses, and shift registers), perhaps interconnected externally so as to facilitate “North-South” shifting as shown in Fig. 6. These links can be single-bit for minimum cost, or widened for performance.

Some “free” interconnect is available, since the data buses go to other chips. The host uses this path already when moving data. In principle one could use the address bus for yet another function, but it is needed for opcodes on every instruction cycle.

---

Fig. 5: C\textbullet\textsc{RAM} external timing

Fig. 6: Chip-to-chip communication as a square array of shift registers
Culture Shock

A new computing model will certainly take time to propagate, but by starting with high-value well-defined applications such as real-time image processing in multimedia systems the number of programmers needed can initially be fairly limited; in addition, data-parallel C or C++ objects can allow programmers to manipulate vectors and images without concerning themselves unduly with the details. Other architectures will help to train programmers, too, as (for example) experience with the Intel MMX architecture gets more people accustomed to a type of SIMD programming.

6. DRAM Pinout and its Effects on C•RAM

DRAMs have traditionally conserved pins, by multiplexing row and column addresses and by having relatively narrow I/O buses. This has helped to keep packaging and board-level system costs low. The large die sizes of modern DRAMs — one to two square centimetres — make larger numbers of pins practical, but they still constrain interprocessor communication, I/O, and control. On a 1M × 16b platform, for example, C•RAM can be implemented in the standard 44 pin TSOIC package by multiplexing opcodes with addresses (or data) — they aren’t required simultaneously — as shown in Fig. 5. The standard 1K cycle refresh DRAM uses 37 of the 44 package pins (16 data, 10 address, 6 power and 5 for control). C•RAM requires one additional control pin, the opcode strobe (OPS), and 4 communications pins, and hence still fits in the JEDEC package. Conserving pins is also a power-control discipline, since pins drive buses and burn power (at 30mW each for 100MHz operation, as discussed above).

The interprocessor-communication constraint comes when a large system is to be composed of many C•RAM chips. For example, a shuffle network may require that one line per processor cross the chip boundary, bringing us to 4k pins for interprocessor communications alone. This number of pins would imply the need for a few thousand power-supply pins to handle the drive

Table 1: A Brief History of SIMD

<table>
<thead>
<tr>
<th>Machine</th>
<th>year</th>
<th>on-chip main mem.</th>
<th>mem. redundancy</th>
<th>mem. b / PE</th>
<th>local mem.</th>
<th>PEs / chip</th>
<th>PE redun-dancy</th>
<th>max PEs</th>
<th>data-path</th>
<th>autonomous memory addressing</th>
<th>autonomous network</th>
</tr>
</thead>
<tbody>
<tr>
<td>STARAN</td>
<td>1972</td>
<td>256</td>
<td>8</td>
<td>1K</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICL-DAP</td>
<td>1976</td>
<td>4096</td>
<td>✓</td>
<td>16</td>
<td>1K</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VASTOR</td>
<td>1978</td>
<td></td>
<td>✓</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPP</td>
<td>1981</td>
<td></td>
<td>✓</td>
<td>8</td>
<td>16K</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GAPP</td>
<td>1984</td>
<td>128</td>
<td>✓</td>
<td>72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CM-1</td>
<td>1985</td>
<td>4096</td>
<td>16</td>
<td>64K</td>
<td>1</td>
<td>note 1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pixel Planes 4</td>
<td>1987</td>
<td>72</td>
<td>✓</td>
<td>64</td>
<td>256K</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AIS</td>
<td>1988</td>
<td>32K</td>
<td>✓</td>
<td>8</td>
<td>1K</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BLITZEN</td>
<td>1989</td>
<td>1024</td>
<td>✓</td>
<td>128</td>
<td>16K</td>
<td>1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MasPar MP-1</td>
<td>1989</td>
<td></td>
<td>✓</td>
<td>32</td>
<td>16K</td>
<td>4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C•RAM</td>
<td>1989</td>
<td></td>
<td>✓</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DAP 610C</td>
<td>1990</td>
<td>256K</td>
<td>✓</td>
<td>64</td>
<td>4K</td>
<td>1 &amp; 8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VIP</td>
<td>1990</td>
<td>256</td>
<td>✓</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TI-SVP</td>
<td>1990</td>
<td>320</td>
<td>✓</td>
<td>1024</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CM-2</td>
<td>1990</td>
<td>1M</td>
<td>16</td>
<td>64K</td>
<td>1 &amp; 64</td>
<td>note 1</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MasPar MP-2</td>
<td>1993</td>
<td></td>
<td>✓</td>
<td>32</td>
<td>16K</td>
<td>32</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRC-PIM</td>
<td>1994</td>
<td>✓</td>
<td>2048</td>
<td>✓</td>
<td>64</td>
<td>256K</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC-IMAP</td>
<td>1994</td>
<td>✓</td>
<td>32K</td>
<td>✓</td>
<td>64</td>
<td>8</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EXECUBE</td>
<td>1994</td>
<td>✓</td>
<td>512K</td>
<td>✓</td>
<td>8</td>
<td>16</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C•RAM</td>
<td>1995</td>
<td></td>
<td>480</td>
<td>✓</td>
<td>512</td>
<td>4096</td>
<td>1+</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC-PIPRAM</td>
<td>1996</td>
<td>✓</td>
<td>128K</td>
<td>✓</td>
<td>128</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SONY-linear array</td>
<td>1996</td>
<td>✓</td>
<td>256</td>
<td>✓</td>
<td>4320</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C•RAM-DRAM</td>
<td>1996</td>
<td>✓</td>
<td>2048</td>
<td>✓</td>
<td>2048</td>
<td>✓</td>
<td>1M</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note 1: emulated in software
Note 2: replacement of entire memory blocks

Table 1: A Brief History of SIMD
use reduced voltage swings with properly-terminated transmission lines reducing the ringing effects that call for good noise margin, but then they have to burn $V^2/R$ power in $50\Omega$ terminations.

The $CV^2$ term (more precisely $CV_{DD}V_{swing}^3$) measures switching energy, coming to about 300pJ for a 30pF bus at $V_{DD} = V_{swing} = 3.3V$ and 3pJ for a bitline. These energies can be more mnemonically expressed as $300\mu W/MHz$ and $3\mu W/MHz$ respectively: 4k bitlines cycling at 20MHz require 240mW, 16 buses at 100MHz take $300 \times 10^{-6} \times (1600) = 480mW$.

Power consumption is rapidly becoming a key measure of a computer architecture, because it has been increasing with processing speed and because of the interest in portable computing. High-power chips are also costly because they require expensive packaging at both the chip and system levels, to conduct and then convect away heat, and unreliable because of the rapid aging that accompanies high-temperature operation.

4. Difficulties With SIMD

There is a certain mathematical elegance to SIMD computing, or perhaps a curiosity value, that has tended to draw interest and then disappoint, with the result that SIMD has a long and rather checkered history (Table 1) [2-11,16-33]. Is C•RAM subject to this curse? Some of the problems of SIMD are:

- Some processing elements sit idle when conditional code is executed, because the shared instruction stream forces the controller to execute all paths with one or more active processors (both the “then” and the “else” of an “if”; the maximum number of iterations of a loop; and so on)
- SIMD computers get high performance only on specialized applications, (basically because they handle conditionals so badly)
- They have had a bottleneck when transferring the data to and from a more conventional host by means of a narrow conventional bus, making the fast number-crunching irrelevant
- They have tended to be large and expensive “one-off’s” rather than commodity computers
- Wide buses have consumed a lot of pins and hence, board area
- There aren’t many programmers available who can understand the model

Many of these problems are tightly linked: for example the size and cost of the machines is driven largely by the low level of integration generally adopted, which in turn is driven by the need to minimize non-recurring costs rather than unit costs when designing for a specialist high-cost market.

5. How C•RAM Addresses the Weaknesses of SIMD

Sequential Code

C•RAM simply doesn’t attempt to handle the serial fraction of a computation well, leaving that to the host CPU.

The Host-SIMD Bottleneck

Since C•RAM is the host’s memory, there is in principle no need for data to move. This argument needs a qualifier, though, because if the organizations of data best suited to host and C•RAM are different then data may have to be shuffled.

Pin-count, size and price

The need for a lot of pins derives from the attempt to cut development costs by using commodity memory — a fault of most SIMD designs up to the late 1980’s. This loses access to the wide, low-power, internal data bus that is the raison d’être of C•RAM. Low unit costs in this technology can only be obtained by integrating the processors tightly with memory; there is no “cheap route” to the consumer market.

We also insist that the chip area be dominated by the memory itself, since a consumer facing a choice between 16MB of DRAM and 8MB of C•RAM will probably choose the DRAM — otherwise word processors and spreadsheets won’t run, making the advantages in multimedia and games look merely gimmicky.
only a factor of four. Redoing the example with a smaller or larger computer gives similar ratios, because memory size tends
to scale with processing power. Even going to a multicomputer architecture doesn’t change the ratios, again because they still
have large amounts of memory; adding C•RAM gives a MIMD-SIMD hybrid [14].

The internal “bus” is faster than the external interface, too, because it doesn’t have to drive such long wires. More importantly,
it is much more energy-efficient if the data can be kept local. The power required to drive a wire is \( P = \frac{1}{2} CV_{DD}^2 V_{\text{swing}} f \), where \( C \) is the capacitance of the wire, (in the range of \( 0.2 \text{pF/mm} \), depending on geometries), \( V_{DD} \) is the power-supply voltage, \( V_{\text{swing}} \) is the voltage swing used to represent data, and \( f \) is the rate at which data are clocked.

The \( C \) term favours short buses directly, and the \( V_{\text{swing}} \) term favours them indirectly (and relatively slightly) in that less noise margin is needed for reliable logic operation when currents are kept small and coupling is minimized. In a typical 16Mb DRAM, \( C = 0.3 \text{pF} \) for a single bit-line, but would be about \( 100 \times \) larger for a single bit of an off-chip bus running to a CPU. In the same memory, \( V_{\text{swing}} \) would normally be about 3.3V on-chip and closer to 5V off-chip. High-speed memories often
Once C•RAMs take over main memory, the generic computer may evolve to the structure of Fig. 3, in which the graphics controller runs processors in the frame buffer and in which the DRAM controller is promoted from a resource manager and timing generator to some type of microcontroller. There are many alternatives by that time though, including (at one extreme) versions in which DRAM chips or blocks have their own controllers and (at the other) versions in which the host CPU itself serves as the microcontroller.

The cache may also change, since it can be used to reorganize as well as to cache data; SIMD computations want related data to reside on the same memory chip, to reduce the cost of communication, while conventional architectures even store different bits of the same word in different chips.

The need to redesign controllers and caches is one of the costs that will slow uptake of the technology, but is not such a problem for frame buffers and other specialized memory-like devices. The need to reorganize data also disappears in any application where a single chip of memory suffices; at the 16Mb or 64Mb level this includes quite a few low-end consumer applications (such as games), and it also applies to portable devices (sub-notebooks, palmtops, organizers, etc.) where size and energy budgets are tight.

3. Memory Bandwidth and Power Consumption

The two key advantages of C•RAM come from the fact that DRAM is organized with a very wide, short, internal “bus”. The row-column architecture is usually roughly “square” because the same number of address bits are allocated to row and column (historically, because the address pins are multiplexed between row and column and the square architecture minimizes their number). Thus a 4M × 1b chip fetches (at least) 2K bits when the row address is given, then selects one for output according to the column address; similarly a 1M × 16b chip selects 16k bits with the 10-bit row address, then one of 1024 16-bit words for output when the column address is available. In each case, internal data is available at 1K-2K times the width of external data. In systems with large amounts of memory, more bandwidth is generally thrown out by multiplexing banks of RAM onto a narrow bus.

By way of example, imagine a high-end scientific workstation with a 64-bit bus running at 100MHz, and equipped with 256Mb of 16Mb 50nsec page-mode DRAM. The datapath at the sense amplifiers is 2Mb wide, making for a bandwidth in memory more than four orders of magnitude higher than that available at the system bus (see Fig. 4) and even a good cache wins back
• A RISC processor on memory: this makes a one-chip computer
• Photodiodes with analog processing: this makes for focal-plane processing
• Content-addressing hardware in memory
• SIMD in the CPU

These ideas are different, but not all mutually exclusive. For example, it makes little difference to our arguments for computing in memory whether the RISC “host” is on- or off-chip, and focal-plane processing can advantageously be done in a SIMD style.

The idea of placing processors in memory is quite general, and leaves a good deal of room for invention. Parameters of particular interest are the ratio of processors to memory (in terms of area demands, power consumption, and addressable range), interprocessor communication networks, the degree of local control available, and the degree of “futurism” involved. Our group has generally favoured low-cost, conservatively-designed, approaches — not because we don’t expect the future to happen, but because we’re interested in how to get there.

2. The Similarity of Memory and SIMD

An easy starting point in discussion of smart memory is to compare the internal structure of a dynamic random-access memory (DRAM) with a SIMD machine, as in Fig. 1. DRAMs are composed of large numbers of sub-blocks, each with a row-column structure. They put a simple sense amplifier at the bottom of each column of (typically) 256 bits, and use shared circuitry to select a single row of data. SIMD machines, similarly, share instruction-fetch and (usually) local address hardware over a number of processors, each having its own “column” of data and all connected to some sort of interprocessor communication hardware.

The physical similarity suggests that we could design a SIMD machine by adding processing elements (PEs) to the sense amplifiers of a DRAM. Adding one PE per sense amplifier is not very practical, because sense amplifiers are placed on a very narrow pitch, but it is practical enough for one PE to share four sense amplifiers.

SIMD can also be made compatible with memory at the system level, in that the SIMD device of Fig. 1 can be designed to work as a conventional memory as well. We will argue that this is of key importance, because it shares the cost of the “computing RAM” (C•RAM) with conventional memory functions, and because it provides a “shared memory” connection with a conventional host processor.

A generic computer architecture, with the memory shown at a more detailed level than the rest, appears in Fig. 2. It describes, roughly, PCs, network computers, workstations, and nodes of multicomputers. Smart memories can either become “devices” or they can replace the existing DRAM chips, either in the frame buffer or main memory. It is reasonable to suppose that they will start as relatively expensive chips, limiting them to frame buffers or specialized devices (such as plug-in image-processing or simulation engines) until accumulated sales volume drives their price-per-bit down to compete with DRAM [13].

![Fig. 1: Comparing RAM and SIMD structures](image-url)
Computing RAMs for Media Processing
Duncan Elliott, Martin Snelgrove, Christian Cojocaru and Michael Stumm
University of Alberta, Carleton University, Philsar Electronics and University of Toronto respectively

Abstract
Integrating processing elements in DRAM makes very large bus widths available: at least 2K processing elements fit in a 4Mb chip or 4K in a 16Mb DRAM. The processors can add an area overhead as low as 10% and power overhead of about 10-25%. To get these efficiencies, the processors have to be pitch-matched to the DRAM. Interprocessor communication is also severely limited, especially when going “off-chip” while retaining low-cost packaging. These “computing RAMs” (C•RAM) can form the main memory for SISD or MIMD hosts, making their contributions to the computing load scalable. The SIMD nature of C•RAM matches large image-processing tasks with high uniformity and locality of reference, making real-time DCT, anti-aliasing and a variety of transformations available at the low cost required for consumer applications.

Even given a PE “budget” of 70-200 transistors, and with the limited interconnect characteristic of low-cost DRAM, there are quite a few architectural choices available to the computer architect. These can be made to favour the data widths and operations needed for image processing while retaining good generality.

Keywords
SIMD, smart memory, C•RAM, DRAM, logic in memory, architecture.

1. Introduction
Image processing has historically been expensive because of the large quantity of data involved, which in turn calls for a lot of processing, and adding real-time constraints brings computation rates into the giga-op/s range. For multimedia applications to appeal to a mass market, the hardware required has to be cheap and flexible; for portable applications it will also have to be very energy-efficient. Consumers will become increasingly aware of artifacts resulting from poor anti-alias filtering and other algorithmic short-cuts, and will regard them as symptoms of poor quality.

There is more to multimedia than video, and hardware that can handle high-quality audio, effects, voice input, and image recognition and editing has to be very general; a solution that involves new custom hardware for each medium will be expensive and inflexible.

The emerging “smart memory” [1-12] architecture gets speed by exploiting the internal datapath width in memory chips, and energy efficiency by keeping computations localized on a millimetre scale. These chips are best seen as memories that supplement the generality of more conventional computer architectures, offering excellent performance on the “massively-parallel component” of computations without attempting to handle the serial fraction. For multimedia applications, the massively-parallel fraction of computation is large, so this area is likely to derive special advantage from the technology. The architectures are very simple, with their minimality a sort of guarantor of flexibility.

Our view is that single-instruction-stream, multiple-data-stream (SIMD) processing belongs on-chip with the frame buffer: there are other architectures that add

• Nothing to memory: because commodity memory is very cheap due to the huge market it enjoys

1. Further information: http://www.ee.ualberta.ca/~elliott/cram/
D. Elliott: Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada T6G 2G7; email elliott@ee.ualberta.ca; WWW: http://www.ee.ualberta.ca/~elliott/; phone: (403) 492-5357; fax: (403) 492-1811
M. Snelgrove: Dept. Electronics, Carleton University, email: snelgar@doe.carleton.ca; WWW: http://www.doe.carleton.ca/~snelgar/
C. Cojocaru: email: cdc@philsar.com
M. Stumm: email: stumm@eecg.toronto.edu; WWW: http://www.eecg.toronto.edu/~stumm/