Considerations Leading to an Asynchronous SIMD Architectural Approach for Exploiting Mixed Logic and Memory

Charles Weems
Computer Science Department
University of Massachusetts
Amherst, MA 01003-4610
weems@cs.umass.edu

Abstract

There is currently a great deal of interest in the architecture community over the possibility of developing processor architectures that have direct access to the internal bandwidth that is available within memory chips [1, 2]. In this paper we begin by examining some economic factors that affect the likely potential for such access, and their implications for the technology that may become available to architects as a result. An architectural concept is then discussed that has the potential to take advantage of that technology.

1. Economic Considerations

Computer architecture today is intimately intertwined with economic realities. Many an excellent architectural concept has been developed, only to be consigned to the dustbin of history because it failed to match the economics of its time. At present, computers are approaching an important transition in the economic model in which they operate. Their ubiquitous nature in embedded applications is turning them into not only commodity devices but some of their components are even showing possibilities of becoming consumables.

The market for commodity and consumable devices depends on extremely high volume and is highly price sensitive. It is a much different market than that of the traditional mainstream of computation (mainframes, servers, workstations, and PCs). These new markets generally do not demand the ultimate in computational performance, but rather require minimal cost, weight, size and power consumption. Because of the volumes involved, however, these are precisely the markets that are starting to drive the path that basic computing technology is taking.

In the past, computer technology was typically developed specifically for computers. For example, magnetic core memory, semiconductor memory, magnetic disk, and digital integrated circuits are all technologies that were created to the specifications of the computer industry. Today, however, mainstream computing is becoming a specialty market that is losing the economic clout to pioneer all-new technologies. CD-ROM is a hand-me-down from the entertainment industry, as is DAT tape. This trend appears to be widening as CRT manufacturers talk of unifying the computer monitor and HDTV product lines. Of particular relevance to the following discussion is that RAM manufacturers see their major market shifting from computers to products such as HDTV, home entertainment systems, digital cameras and camcorders, etc.

1.1 Technological Implications

As an example, consider that by the year 2000, Hitachi plans to deliver 4 Gb DRAM chips, and no doubt 8 Gb to 16 Gb chips will shortly follow. Even a single one of these chips is probably an excessive amount of RAM for a typical PC and multiple chips will likely be needed to provide sufficient bandwidth to sustain a high performance processor's demands. Thus, we might argue that the processor needs to move onto the chip where it can have access to more of the available internal bandwidth.

On the other hand, memory with this density (assuming the present price curves are followed) begins to look attractive as a replacement for magnetic disk, especially in applications where size, weight and power consumption are important. Given a battery backup system for long-term nonvolatility, and a Type II PC-card form factor, a removable device with a few chips providing up to 10 GB of storage has the potential to replace consumable media such as CD, removable magnetic disk, and videocassette. RAM thus makes an economic transition to a consumable with volumes that are orders of magnitude greater than at present.

In this economic environment, it is highly unlikely that the specialty market of mainstream computing will be able to convince memory manufacturers to alter their most advanced fabrication processes to enable on-chip computation, since such alterations increase costs. In fact, it is likely that no single market would have the necessary level of influence and it is even hard to imagine a confluence of markets that would settle on a single on-chip architecture so definitively as to drive the memory manufacturers to adopt a Logic-on-RAM process that goes beyond current SRAM cache on DRAM chip designs.

This rather bleak picture should not, however, call a halt to investigating integrated logic and RAM architectures. It merely means that the focus must be on different areas of RAM technology than the cutting-edge bulk DRAM processes. There are two technological possibilities that are worth exploring: SRAM and "back of the wave" DRAM.

SRAM offers several advantages over DRAM. Besides having lower latency and avoiding the need for refresh, the process technology is more compatible with on-chip logic. In economic terms, SRAM is a device that will continue to be guided by the demands of the computer industry because it is not suited to the applications that high-density DRAM will be courting. Of course, the disadvantages of SRAM are its lower density and higher power consumption.

In riding the DRAM technology wave, it isn't necessary to stay on the leading edge or even the peak of the wave. DRAM processes have life cycles that include a considerable period of production after newer processes have "replaced" them -- the back of the wave. As demand for these older chips starts to slip, it can be an opportunity for architects to approach memory manufacturers about customizing the processes and chips to extend the useful life of a fabrication facility. Today's 8 MB and 32 MB chips are the potential targets for these opportunities in a few years to come. If this seems insufficient, it should be noted that, depending on relative die sizes, in a few years it might be possible to fit multiple "obsolete" RAM bodies on a processor chip.

It should also be kept in mind that proposals to merge RAM and processor logic need to have an upward migration path. A point solution for using the available on-chip bandwidth that must be completely redesigned with the next generation of RAM chips is unattractive. In addition, a RAM-based architecture must have the flexibility to accommodate a variety of host environments in order to maximize its potential range of markets.

1.2 Some Historical Background

One premise on which a RAM-based architecture rests is that the available bandwidth of a RAM body can be employed effectively. However, this bandwidth is constrained by a high degree of locality. The column address extracts a single, contiguous column of bits from each block in the RAM body. If this set of bits corresponds well to some logical form of locality that is present in the computation, then the bandwidth is used effectively. If the correspondence is poor, then some portion of the bandwidth is wasted.

While much effort has gone into extracting parallelism at the instruction level of sequential programs, the level of parallelism and the regularity of access implied by the access restrictions inside a RAM body argue for explicit data parallelism. Certainly, SIMD arrays such as the Maspar [3] and CAAPP [4] have featured a heavy dependence on a large block of on-chip wide-access memory. At the other end of the data-parallel spectrum, we have the vector supercomputers which also relied on very wide data paths to memory. However, all of these machines have now exited the marketplace. We would do well to ask why they failed before embarking on a new design.

The simplistic answer is cost/performance. They were large, costly architectures that failed to scale up in performance as less costly microprocessors approached similar performance levels. So the first lesson to take home is that any new data parallel architecture must deliver significantly greater performance than a contemporary uniprocessor at a comparable price level (i.e., starting from a single chip configuration). A mere factor of 10 will not justify the added difficulty and risk of using a parallel architecture, but at 20 times the performance level, it becomes easier to justify (50 times would lead to serious interest).

Looking deeper into the source of the cost/performance deficit of these architectures we see that much of the cost stems from their synchronous nature. In the case of the SIMD array, there was a single, expensive, source of instructions and many slow (by today's standards) processor chips that all operated in lock step. Synchrony implied low clock skew, which resulted in reduced clock rates with the standard CMOS/PC board technology that was used. In the case of the vector supercomputers, synchrony at high clock rates required exotic technology and costly manufacturing processes to keep clock skew within tolerances. In both cases, synchrony limited upward scaling of performance, and indirectly made it difficult to scale the architectures down to lower cost levels.

Synchrony also means that interprocessor communication bandwidth is used on an all-or-nothing basis. Everybody communicates on the same instruction step, and then the communication channels sit idle for non-communication steps. Thus, SIMD architectures featured costly interconnection hardware that was often underutilized. To accomplish the same ends, vector machines depended on moving data rapidly between memory and vector registers. Unfortunately, this required a flat memory that is considerably larger (e.g., 2 GB) than what we can expect to have access to on a single chip in the near future.

2. Asynchronous SIMD: Architectural Concept

The potential cost/performance advantage of data parallel processors over uniprocessors stems from the replication of arithmetic units while retaining a single control unit (and some cost savings

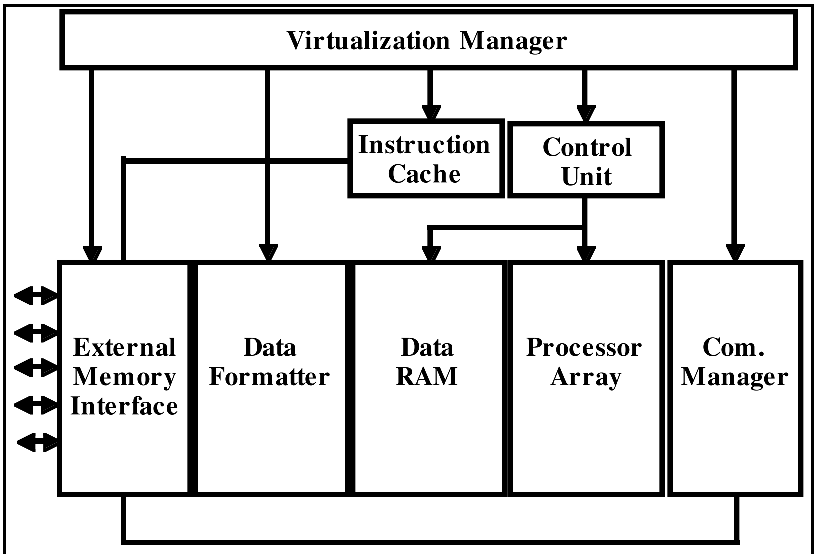

Figure 1. A Processor Block

resulting from the regularity of the resulting structures). However, we can see from the preceding discussion that when this approach is arbitrarily scaled upward, performance suffers or, alternatively, cost rises.

We thus propose a variation on the SIMD/Vector theme in which arithmetic units are replicated only to the point that the loss in performance due to the need to distribute a common clock falls within some acceptable bound. For example, we could arbitrarily set our bound to be that the clock must not fall below that of contemporary microprocessors. Limiting clock skew is certainly one factor that affects the ALU replication factor. Because synchronous data-parallel devices employ wider datapaths than microprocessors, an additional factor in their design is controlling supply-induced noise. The point is that there are multiple factors that combine to limit the amount of synchronous parallelism that can be obtained for a reasonable cost, and those factors present a new mix of engineering challenges to the VLSI designer.

For such a synchronous set of replicated arithmetic units, a control unit, instruction cache, data RAM body, and other management logic would be provided on the same chip. We refer to each of these units as a processor block (Figure 1). If there is room on a chip to accommodate multiple processor blocks, then they are constructed independently. Within a processor block, operation is in the traditional synchronous SIMD/Vector mode, but between blocks, execution and communication is asynchronous (although the asynchrony is bounded, much as in an SPMD processor). Hence the name, Asynchronous SIMD or ASIMD. Initially, we envision a single block per chip, with many systems containing just a single chip that is employed as a coprocessor.

2.1 Exploiting Virtualization

From the SIMD regime, we take the notion of virtual processors -- the emulation of a larger array of processors by a smaller physical array. It can be assumed that the number of processors in a block will be modest (e.g., less than 128) so that each operates at a high virtualization ratio on many applications. Thus, the physical block repeats the same set of cached instructions for each virtual processor block -- potentially many thousands of times, which represents a high degree of instruction locality.

Furthermore, this locality is entirely generated within and under the control of the processor, so that it can precisely predict when the current set of instructions will be replaced and issue a request for the next instruction block at the appropriate time. The instruction cache can be even more effective if it stores the instructions in a predecoded form, allowing rapid reissue for each tile. Instructions would be downloaded into external memory by a host processor, and the on-chip controller prefetches them as necessary with cycle stealing.

Local communication (such as shifting within a matrix) between processors mostly takes place within the virtual blocks assigned to a physical block (we call this set of virtual blocks a virtual tile). Local communication with adjacent physical blocks takes place only for the virtual blocks on the edges of a virtual tile. If the processing of these virtual blocks is appropriately scheduled, then the data at one edge of a tile can be sent while internal virtual blocks are being processed (Figure 2). Because of the high virtualization factor, this communication requires neither high bandwidth nor low latency to arrive before it is needed at the adjacent physical block.

Nonlocal communication, such as permutation routing, will have more of an effect on the communication bandwidth and latency requirements of the processor chips, as this involves moving data among all of the virtual processors. Nonetheless, we avoid the requirement for dedicating a large fraction of the available pins on a chip to local synchronous communication that is often underutilized. If the logical communication ports for a block are memory mapped, then that pin bandwidth can simply be folded into the external memory interface.

Between the Data RAM and the External Memory Interface, Figure 1 shows a block called the Data Formatter, which is responsible for ensuring that data is placed in the Data RAM with the necessary locality to enable direct access by the processors. The Formatter must be able to adjust the alignment of data either as it is being input from external memory or while it is in the Data RAM. It should support a wide range of data permutations to account for varying stride lengths, compaction of sparse vectors, and indexed access to memory. This may necessitate the equivalent of a Clos network, and dual-porting of the Data RAM.

From the hardware designer's perspective, the ASIMD organization has the advantage that it offers flexibility in applying the available RAM technology to achieve a

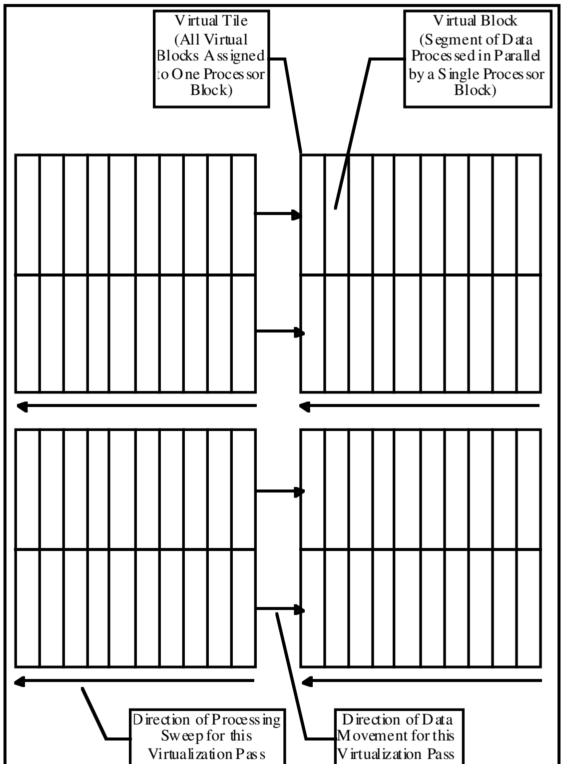

Figure 2. Four Virtual Tiles, each with 20 Virtual Blocks Showing Communication Pattern and Processing Order for One Virtualization Pass by the Corresponding Four Processor Blocks

better balance between computation and memory bandwidth. Memory blocks within RAM bodies and possibly multiple RAM bodies on a chip can be divided among multiple processor blocks or aggregated to serve a single block depending on the results of performance studies for the particular technologies being used [5].

Higher clock speeds imply greater memory bandwidth requirements to supply the processors with data. But as the previous discussion notes, almost all of the available pin bandwidth on the chip can be devoted to memory. It thus becomes a matter of balancing the processor throughput against the available bandwidth. Of course, some of the processor's data rate requirements are mediated by the on-chip data RAM. In addition, it should be noted that the pattern of data memory access is largely determined by the virtualization scheme. Thus, it is possible to predict data accesses well in advance, and to arrange for streaming that takes advantage of high-performance external RAM devices (e.g., SDRAM, RAMbus, SynchLink). In fact, much of the complexity involved in managing virtualization will likely be in the Data Formatter and the Virtualization Manager.

Because the number of processors on a chip will be fairly small (compared to the number in a traditional SIMD array), the probability is high that many entire tiles will have been deselected in a sparse phase of computation. This presents the opportunity for the virtualization management circuitry to identify each inactive tile and skip loading its data and reissuing the current group of instructions for it. A more aggressive strategy might even seek to gather the individual active elements from a sparse array and compress them, along the lines of the Gather/Scatter operation in a Cray vector unit.

2.3 A Back-of-the-Envelope Example

Much of the decision as to what level of parallelism should be provided in an ASIMD chip depends on the available sustained external memory bandwidth. Although a large amount of RAM is present on the chip, it is still tiny compared to actual problem sizes. For example, with back-of-the-wave DRAM technology providing 8 MB of on-chip RAM, a moderately complex image processing task would easily run out of space. With 32 MB, some of these tasks will be able to run entirely in on-chip RAM, but many will not. Careful management of the on-chip RAM will reduce the amount of off-chip bandwidth, but will not eliminate it. In degenerate situations, an ASIMD chip could well become memory bandwidth limited. Given Amdahl's law, these situations need not occur frequently in order to seriously reduce performance. Thus, it does not make sense to significantly overdesign the processors.

Consider, as an example, an ASIMD chip with a SynchLink interface (a pair of RAMbus interfaces would be similar) which provides up to 1.6 GB/s of sustained transfer capacity. If the processors operate at 200 MHz and achieve merely 16 operations per external access, then the memory could sustain a block with 32 32-bit processors, delivering 6.4 GOPS. If the processors are multi-gauge, then they could also offer 12.8 16-bit GOPS. This is shows the potential for ASIMD to provide the requisite factor-of-20 increase in performance over a microprocessor operating at the same speed, even under the assumption of a poor pattern of virtualization. It is probably reasonable to expect a factor of 50 on many problems. Assuming that there is sufficient VLSI real estate, other factors would probably limit the size of a processor block to about 128 elements. A block of this size would offer about 25 32-bit GOPS, and about 50 16-bit GOPS.

By combining multiple ASIMD chips into a system, of course, performance can be further increased as long as the level of virtualization remains high. Not only do we gain the additional processing power, but we also add to the available on-chip memory so that more tasks will be able to run entirely within RAM. Probably two to four ASIMD blocks could operate on a shared bus, but beyond that some distributed (possibly shared) memory mechanism would be required to provide the necessary communication bandwidth.

2.3 Tracking with Technology

In a traditional SIMD processor chip, as the number of processors increases, even on-chip clock-skew problems become significant. The use of multiple processor blocks on a chip, rather than simply increasing the number of processors in a block, is attractive because it avoids the need to distribute a full-rate global clock with low skew. As chip sizes scale up, the number of metal layers typically increases, but the upper levels are not suited to carrying high-speed signals. The multi-block approach allows the designer to assign slower global clock distribution, lower bandwidth direct interblock communication channels, and power distribution to the uppermost metal layers.

In addition, the need to sustain a higher level of on-chip memory bandwidth implies wider synchronized buses, which lead to increasing problems of crosstalk and supply-induced noise. In normal operation, of course, RAM chips avoid this problem by splitting the memory into banks and fetching from just one bank at a time, or pipelining interleaved accesses across multiple banks. When whole ASIMD chip bodies are replicated, these problems can be reduced by purposely skewing the clocks to the independent units on the chip, effectively mimicking the interleaved bank access pattern. The independent nature of the blocks also enables them to employ separate external memory interfaces to the high-performance RAM devices mentioned in the previous section, which can be difficult to operate in lock-step groups.

2.4 Software Impact

From the software developer's perspective, ASIMD retains the simple single control thread model with parallel types. Compiler technology to exploit this style of processing is well developed, and standard parallel languages for it also exist. In spite of its asynchronous nature, the fact that synchronization is maintained logically across all of the blocks means that debugging can employ traditional sequential techniques.

From the operating system writer's point of view, however, ASIMD is far more interesting than SIMD, which presented a single monolithic resource with a very high context switch cost. With ASIMD, there is no need for all of the blocks to be executing the same program. Blocks can be operated independently of each other simply by sending different instructions to their instruction caches. When a system has multiple blocks, they can be aggregated in any fashion, since communication between them is simply a matter of establishing a mapping between their communication ports. Thus, a collection of ASIMD blocks is automatically acquires the added flexibility of a Multi-SIMD array.

Multi-SIMD operation can improve processor utilization by enabling multiple smaller problems to be run simultaneously. For example, in an image processing task, all of the blocks might first work together in ASIMD mode to preprocess an entire image, but then split up into MSIMD mode for processing multiple regions of interest. Depending on the size of a region of interest, it could be assigned multiple blocks or just a single block.

Multi-SIMD operation also enables the array can to be shared among multiple users, avoiding the need to schedule the use of a single monolithic resource. However, it also implies that the architecture must fully support variable degrees of parallelism and virtualization in executing a single binary object program, since the operating system determines the number of ASIMD chips to allocate to the program at runtime.

Going beyond this level of sharing, it is possible that an ASIMD system could also support traditional time sharing. Whenever a tile is written out to memory, the majority of the chip's context has been saved. Thus, it might only be a modest additional cost to save the remaining context and load a new context in its place. Of course, if several ASIMD chips are operating in a cluster, it would be prudent to switch context for all of them together. Otherwise the ones that do not switch context will simply stall while waiting to communicate with the ones that did switch. In our image processing example, this approach can be further used to improve processor utilization if there are multiple small regions of interest, which do not individually make full use of a processor block. These regions can be assigned as a set of tasks to be executed on one block via time-sharing.

3. Summary

We should not count on convincing memory manufacturers to grant our wishes for adding logic to their most advanced DRAM processes. However, there are still opportunities with SRAM and "back of the wave" DRAM processes to greatly increase the available amount of memory and access bandwidth on a processor chip. Some architectures of the past that have attempted to exploit similar memory characteristics are SIMD arrays and vector supercomputers, but these have fallen on hard times in part due to excessive reliance on synchronous operation.

Asynchronous SIMD is an architectural concept that offers the potential for flexible use of combined logic and RAM technology, both by being able to utilize on-chip bandwidth and also by being able to internally manage the use of off-chip bandwidth. ASIMD is also particularly well suited for future VLSI technologies employing more levels of metal, due to its natural paritionability.

There is still considerable research to be done in order to determine whether Asynchronous SIMD is truly a viable path. This paper makes a case for pursuing that research based on the potential for high single-chip performance with scalability, and the considerably greater flexibility of the architecture with respect to traditional SIMD, while retaining a simple programming model and offering greater opportunity for resource management by an operating system.

4. References

1. David A.Patterson, "Microprocessors in 2020," Scientific American, Sept., 1995, pp. 48-51.

2. D.C. Burger, S. Kaxiras, and J.R. Goodman. "DataScalar Architectures," To appear in the 24th International Symposium on Computer Architecture (ISCA), June, 1997.

3. Tom Blank, "The MasPar MP-1 Architecture", Proceedings of IEEE CompCon, February, 1990.

4. Charles C. Weems, Steven Levitan, Allen R. Hanson, Edward M. Riseman, J. Gregory Nash, David Shu,"The Image Understanding Architecture", International Journal of Computer Vision, 2(3):252-282, 1989.

5. Martin Herbordt, "Performance Evaluation of Massively Parallel Array Architectures," Computer Science Ph.D. Dissertation, University of Massachusetts, Amherst, MA, August, 1994.