SIMD Parallel Processing

Michael Sung
6.911: Architectures Anonymous
February 22, 2000

1 Introduction

SIMD represents one of the earliest styles of parallel processing. The term SIMD stands for "Single-Instruction Multiple-Data," which aptly encapsulates the parallel processing model. Closely related to vector processing, the basic idea is to operate the same instruction sequence simultaneously on a large number of discrete data sets. SIMD machines are geared toward applications that exhibit massive amounts of data parallelism without complicated control flow or excessive amounts of inter-processor communication. Typical applications for SIMD machines include low-level vision and image processing, discrete particle simulation, database searches, and genetic sequence matching.

2 Background

The history of SIMD machines began with the ILLIAC IV project, started in 1962. The machine was the first large-scale multiprocessor, composed of 64 64-bit processors. The project itself was pretty infamous for its failure; estimated costs of $8 million ballooned to $31 million by 1972. The actual performance of 15 MFLOPS was far below the original estimates of 1000 MFLOPS, partially because only a quarter of the planned machine was ever constructed. In addition, the machine took another 3 years of engineering to actually work following its delivery to NASA in 1972. Needless to say, the project slowed interest and investigation of SIMD architectures for quite a while. Eventually, Danny Hillis resurrected the SIMD architecture in 1985 with his Connection Machine. However, following a short stint in the 80's by several commercial companies such as Thinking Machines and MasPar, SIMD has once again fallen by the wayside in the arena of commercial general-purpose computing.

3 General Description

SIMD machines can be classified as processor-array machines; a SIMD machine basically consists of an array of fine-grained computational units connected together in some sort of simple network topology. This processor array is connected to a control processor, which is responsible for fetching and interpreting instructions. The control processor issues arithmetic and data processing instructions to the processor array, and handles any control flow or serial computation that cannot be parallelized. For flexibility in implementing algorithms, processing elements can usually be individually disabled for conditional execution. The instructions issued by the control processor are executed by the processor array in lockstep operation. Thus, control for a SIMD machine is vastly simplified, and synchronization issues can be avoided. Since the individual processing elements are usually very simple in nature, SIMD machines can typically run at very high clock rates and process data very quickly.

When designing a SIMD machine, the primary factors to consider are:

- Processing element selection
- Communications/network topology
- Instruction issue

The primary trade-off for SIMD machines is between processor simplicity and cost. Traditionally, SIMD processor elements are very rudimentary, shunning complex control and generalized function in favor of simple interfaces and implementation. Oftentimes, SIMD machines employ bit processors which operate on only one bit at a time (bit serial); thus a 32-bit operation would require 32 cycles for a bit processor to work through. One of the chief benefits of using very simple processors is that they are readily optimized and uncomplicated to control. An added benefit is the fact that bit processors can work on arbitrary length data. In addition, since SIMD machines are targeted at specific applications, no silicon is wasted in implementing functionality that is not used.

SIMD processing elements are usually connected to their nearest neighbors, forming 2D or 3D meshes. The benefit here is the physical connections often mimic the physical phenomena the program is trying to simulate (e.g., cellular automata for microphysical simluations). SIMD-style processing works best when there is little data exchange between processors, since communication is costly. The ideal case has each processing element munging on data stored locally on each processing node without any data dependencies.

The method by which instructions are issued to the processor array are of primary concern. The control processor may broadcast instructions to the processor arrays, or each processing node can hold a local cache of instructions to operate. Distribution of instructions to the processing nodes is a serious issue when designing SIMD machines.

4 SIMD Machines

The three SIMD machines covered in this paper are the Connection Machine by Danny Hillis, the Abacus Project at the MIT AI Lab, and the CAM-8 machine by Norman Margolus. These three machines give a pretty accurate sampling of the type of SIMD machines that were constructed as well as an idea of the motivations for creating the machines in the first place.

The Connection Machine was composed of 65,536 bit processors. Each die consisted of 16 processors with each processor capable of communicating with each other via a switch. These 4,096 dies formed the nodes of a 12th dimension hypercube network. Thus, a processor was guaranteed to be within 12 hops of any other processor in the machine. The hypercube network also facilitated communication by providing alternative routes from source processor to destination. Each node was given a 12-bit node ID, and different paths between two nodes in the network could be traversed based on how the node ID was read. The network allowed for both packet and circuit-based communication for flexibility.

The second machine discussed is the Abacus machine created at the MIT AI Lab. This machine was constructed primarily for vision processing. The machine consisted of 1024 bit processing elements set in a 2D mesh. The primary concept of interest from the design was that the processing elements were configurable, and used reconfigurable bit parallel "RBP" algorithms instead of traditional bit serial computation. This means that each PE emulated logic for part of an arithmetic circuit (be it an adder, shifter, multiplier, etc) based on a RBP algorithm. The motivation for having these configurable processing elements was to save on the silicon area needed to implement arithmetic. However, because there was a necessary overhead for reconfiguration and the implementation did not easily allow for pipelining due to data dependencies, it was not clear that having configurable processing elements was a definite win.

The final machine of the talk was for the CAM-8 cellular automata machine, a machine that mimics the basic spatial locality of physics. Thus, each processing element would get it's own "piece" of the physical entity that is being modeled, such as a location in space or a particle, etc. The processing elements are connected in a 3D mesh, a natural topology for describing microphysical simulations. Each processing element consisted of a programmable lookup-table with an associated local memory. Since there are usually not enough processing elements to assign one to each cell of the system that was to be simulated, each PE was assigned a specific region and performed updates to each cell virtually. Cellular automata applications include microscopic physics/lattice gas simulations (for which the machine was originally intended), statistical mechanics, and data visualization/ image processing.

4 Issues with SIMD

Although SIMD machines are very effective for certain classes of problems (namely embarrassingly parallel problems with very high parallelism), they do not perform well universally. The architecture is specifically tailored for data computation-intensive work; as such, SIMD machines are rather inflexible and perform poorly on some classes of important problems. In addition, SIMD architectures typically do not scale down in competitive fashion when compared to other style multiprocessor architectures. That is to say, the desirable price/performance ratio when constructed in massive arrays becomes less attractive when constructed on a smaller scale. Finally, since SIMD machines almost exclusively rely on very simple processing elements, the architecture cannot leverage low-cost advances in microprocessor technology. A combination of these factors and others have lead SIMD architectures to remain in only specialized areas of use.

5 Conclusion

Even though SIMD machines have not found their way into ubiquitous use, they have not completely died out. Why? For one, SIMD architectures still make a lot of sense for special applications that require a great deal of independent data computation. In addition, multimedia and graphics applications have become very commonplace with the advent of the internet and computer-gaming. Since multimedia such as video or sound processing are inherently parallelizable tasks that involve computation on data streams, small-scale commercial SIMD variants have come into the marketplace. Thus, SIMD has once again found a way to survive in the form of ISA multimedia extensions, including Intel's MMX and SSE, AMD's 3D-Now, and Motorola's AltiVec.

6 Bibliography

1) J. Hennessy and D. Patterson, "Computer Architecture: A Quantitative Approach," Morgan Kaufmann Publishers, SF, CA., 1996
2) W.D. Hillis, "The Connection Machine," MIT Press, Cambridge, MA., 1985
3) M. Bolotski, et. al., "Abacus: A 1024 Processor 8 ns SIMD Array," Proceedings of Advanced Research in VLSI '95
4) N. Margolus, "CAM-8: a computer architecture based on cellular automata," MIT Laboratory of Computer Science, Cambridge, MA., 1993