A 2-LAYERED SIMD/MIMD PARALLEL PYRAMIDAL "ARRAY/NET"

by

Leonard Uhr

Computer Sciences Technical Report #409

December 1980

A 2-Layered SIMD/MIMD Parallel Pyramidal "Array/Net"

Leonard Uhr
Computer Sciences Department

Murray Thompson, Joseph Lackey
Physical Sciences Laboratory

University of Wisconsin, Madison

Abstract

This paper describes and examines a 2-layered multiprocessor system designed to serve as an SIMD (Single Instruction, Multiple Data stream) synchronous array and also as an MIMD (Multiple Instruction, Multiple Data stream) asynchronous network.

The Parallel Pyramidal "Array/Net" (PPAN) has 16 processor-Groups. Each processor-Group has a 4 by 4 array of 16 (8-bit) "Slave" processors and one (64-bit) "Master" sequencer. The 16 processor-Groups are arranged in a 4 by 4 array. This gives a 16 by 16 array of 256 SIMD processors, and also a 4 by 4 array of 16 MIMD processors.

This system was designed with scene analysis and description, pattern recognition, image processing and numerical problems primarily in mind for the First Layer of 256 SIMD processors. The Second Layer of 16 more powerful MIMD processors was designed to handle the "higher" levels of perceptual programs and also data base and artificial intelligence problems for which a network can be useful.

NOTE: The Array/Net was designed at the general level chiefly by Uhr and Lackey, with key ideas from Thompson. Lackey did most of the work on the detailed design. We are grateful to a number of other people for the important help they gave, in frequent discussions of design, algorithm development and programming issues. These include Colin Cryer, Heinz Kuettner, Larry Schmitt, Alwyn Scott, and Larry Travis.

This work was partially supported by NSF research and equipment grants.

Introduction

This paper describes a Pyramidal "Array/Net"work system of processors that was designed for perceptual problems (scene description and analysis, pattern recognition, image processing), number crunching on large matrices, and a variety of artificial intelligence and data base problems. Our original intent was to build (or buy) an SIMD array - probably a CLIP4 (Duff, 1976) - and also a small network of a dozen or so microprocessors (like those being designed by Wittie, 1976, Despain and Patterson, 1978, and others).

We wanted to combine the array and the network, along with a host computer (a VAX) into a converging pyramid-like system. This would serve as a laboratory for perception programs that used parallel-serial cone/pyramid structures of processors; e.g., the systems being developed by Levine (1978), Tanimoto (1976, 1978) and Uhr (1972, 1976). It would also serve (chiefly the array) for large number-crunching problems of the sort being explored for parallel systems by Cryer (1979), Scott (1977) and others. And (at least in the network) it would serve to explore and develop algorithms for a variety of problem-solving, semantic memory, robotics, and data-base management problems.

We originally planned to build only the network. But for a variety of reasons it was not clear whether we would be able to acquire an array built elsewhere, and we began to design our own system. Here we were working under severe (but reasonable) constraints: We had less than $150,000 for both the array and the network. And we were not able to take advantage of the special-purpose LSI chips being designed for arrays. These chips had not yet been checked out; nor was it clear that we would have been able to acquire any from the people developing them. (These included the CLIP-4 (Duff, 1976) 8-processor chip; the ICL DAP (Reddaway, 1978) 4-processor chip; the Goodyear-Aerospace ASPRO (anon, 1979) 32-processor chip; and the Goodyear-Aerospace MPP (Batcher, 1980) 8-processor chip.

We wanted as much flexibility as possible in designing a system that would have good machine language operations for perceptual, logical and numerical tasks. We therefore chose to design and build using bit-slice chips, with their great potentiality for custom design, their good microcode capability, and their powerful yet mature technology.

For processing images (e.g., the roughly 250 by 250 television picture) and large arrays of numbers we needed as large an array of processors as possible. For the much smaller network we wanted as powerful and flexible as possible a processor. But far fewer were needed. We also wanted to combine these two systems in as intimate and as usable a fashion as possible.

These considerations soon led us to explore integrating the array and the network into a single design. The resulting combination appears to benefit in a number of ways, and also to have some disadvantages.

The sections that follow describe this system, compare it briefly with other arrays and networks, examine how it might be used, discuss its strong and weak points and make some suggestions for future designs.

The Array/Net Described

The Array/Net was designed to be built from 2900 bit-slice chips. Each arithmetic logic unit (ALU) chip is 4-bits wide. This allows one to build a processor of any (reasonable) size that is a multiple of 4. These ALU chips can be used in conjunction with other chips in this complete family of chips to custom design and build microcodable computers.

The Array/Net was designed in a highly modular fashion, with 16 separate "Groups" of processors. Each Group was designed to inhabit a single multiwire board approximately 37 cm by 50 cm in size. Each Group consists of approximately 400 16-pin equivalent chips. (Fitting a whole Group onto a single board was a very important consideration from the point of view of economics, since it greatly simplified the design.)

A Group of 16 SIMD Processors

Each Group consists of a 4 by 4 array of 16 "Slave" processors and a "Master" controller that sequences through and executes code.

Each Slave is an 8-bit processor (built from two 4-bit ALU chips). A Group's 16 Slaves are arranged, in traditional array fashion, in a 4 by 4 square. Each Slave can fetch data from either its own memory or the memory of any one of its 4 square neighbors, and store data into its own memory only.

Each Slave has 16K bytes of memory (i.e., 8 16K dynamic rams). Therefore the whole Group has 256K bytes of memory.

All Slaves execute the same instruction, but each executes it on the different set of data stored in its own memory. Thus one such Group operates, and can only operate, in true SIMD fashion.

Each such Group has a single Master controller. It sequences through microcode words that are 64 bits wide. Microcode words are fetched from the Group memory by having the 16 Slaves each fetch an 8-bit word, giving two 64-bit instructions.

The Total SIMD System of 16 Groups

Each Group is in its turn a member of the total 16-Group array. The 16 Groups are themselves arranged in a 4 by 4 array. Since each Group is a 4 by 4 array of 16 Slaves, the total array is a 16 by 16 array of 256 Slave processors.

Each Slave is connected to its 4 nearest square neighbors in the entire array, linking between as well as within groups.

At the outside borders the Slaves (under program control) connect either 1) to input (at the leftmost-column), or 2) leftmost-column to rightmost-column and top-most row to bottom-most row (if there is enough space on the boards). Or, alternately, 3) the outer borders can be set to contain a value (e.g., as with CLIP systems, a Ø to signify empty-background-white and a 1 to signify filled-foreground-black).

To execute the same sequence of instructions over the entire array of 256 processors, every Group must be given the same program. That is, the same program must be stored 16 times, once in each of the 16 Groups' memories. (This was decided on because of the cheap 16 K dynamic rams that were available, coupled with the increased hardware expense if a single sequencer were connected to all Slaves, so that each board would no longer be a nicely self-contained module.)

The Network of 16 Master Processors

Each Group can also serve as a single independent processor, with all 16 Groups working together in typical MIMD network mode.

The Master can be used as either an 8-bit, 16-bit, 24-bit or 32-bit processor, and not merely as an 8-bit processor in the manner of the Slaves. This is achieved by configuring either one, two, three or four slaves as a single processor.

The Master controller can fetch data from any of its 16 Slaves' memories (plus the memories of any of those 16 Slaves' 4 nearest-neighbors) and store data into any of its 16 Slaves' memories. Therefore when working as a single Master computer each processor has its own 256K bytes of memory at its disposal (and this is memory that is shared with the 16 Slave processors).

Masters can communicate in several ways:
A) They can pass information from one Group to its neighbor by using the standard Slave method of fetching from a neighbor. (When the Slaves work in SIMD fashion there can be no contention problem. But now the Masters might contend, and the programmer must handle that problem.)
B) The Masters are all connected to a common bus, which is itself connected to the VAX.

Each Master can execute a different program, since each is a complete computer and each works on a program stored in its own Group's memory. But, if desired, one or more sets of 2,3,...16 Masters can execute the same program, if that program is given to each.

A Set of Up To 16 Different SIMD Groups

There is still another mode in which this system can operate. Each Group can execute in SIMD mode, with each of its 16 Slaves executing the same instruction. Since, as described above for the Masters, Groups can be partitioned into any combinations of sub-sets of 1,2,...16 Groups, each sub-set executing the same instruction over all its Groups' Slaves, a variety of SIMD sub-arrays of 1-bit processors can be established.

This would appear to have great potential power in perceptual systems. For example, after operating for a while in SIMD mode to look for simple features and characteristics that imply what types of objects might be in the input picture, the system could assign different Groups to look for these different types of objects. Thus the lower Groups might be assigned to look for cars and trees, the upper Groups to look for clouds and airplanes. The Group where window-like features were found could be assigned to look for a house, doors, pillars, etc.

Input and Output

The images to be processed will be shifted into the array using a line into the leftmost Slave of each of the rows.

The program, initial code to start up the left-right broadside loading, and any other information from the VAX, will be input over the bus.

Timings, Hardware Considerations, Compromises and Problems

The total memory of the 16-Group system is 4 million bytes. This is unusually large, and was chosen in terms of a number of tradeoffs. (Note that CLIP4 has 32 bits for each of 10,000 processors, giving 40,000 bytes; DAP has 4,000 bits for each of 4,000 processors, giving 2 million bytes; MPP has 1,000 bits for each of 16,000 processors, giving 2 million bytes.)

Tradeoffs in Memory Size, Processor Speed, and Economy

The 16K dynamic ram is cheap, and big. It allows us to handle the need for a copy of the program in each of the 16 Groups' memories with little strain. But it is also slow, on the order of 400 nanoseconds. Indeed, this system is limited in speed by the memory, since the bit-slice chips are much faster (around 120 nanoseconds).

The large memory is also needed because this system must be used with images and other arrays much larger than 16 by 16. For example, a 128 by 128 array must be stored with an 8 by 8 sub-array of 64 cells in the memory of each Slave. Each Slave then processes this sub-array in typical serial form, and any intermediate results must be stored for each of the 64 cells.

Packing an entire Group onto a single board was of great importance, both for simplicity of the modular design and for economy. This also necessitated larger memory to handle separate copies of the program.

But these decisions opened up the several unique new possibilities for MIMD and mixed SIMD-MIMD operation of the system.

Speed of Input and Output, and of Shifting

Several compromises were made because of space and/or money constraints. Left-right shifting takes 600 nanoseconds for 8 bits; but up-down shifting takes 400 nanoseconds for 1 bit (because board-packing problems will probably force us to use only the shift inputs designed into the 2900 for up-down shifting). This asymmetry will be hidden from the programmer.

Since shifting of the image into the array goes left-right, the entire array can be filled with a 16 by 16 1-byte image in 16 600 nanosecond shifts, or 9.6 microseconds. A 128 by 128 8-bit image would take 64 times as long. But this system is designed to speed up the very slow vision programs (which typically take many minutes), and for number crunching programs, and the burden of input is far less than for those systems designed for image enhancement and other short sequences of image processing operations.

The Network Interconnection Pattern

The Masters are not interconnected as closely as one would wish, since they merely share a common bus, plus their Slaves' nearest-neighbor connections. A Master can jump information from one edge or corner to the opposite; then the adjacent Master can fetch it over the edge; then jump it again; etc. This means (considering that the outer edges wrap around) that no pair of processors are more than 6 steps distance. And the bus itself may be acceptable for most programs.

But looking to the future, as the number of Masters increases there will be a need for a better interconnection pattern. We were interested in giving the system a limited reconfiguring capability (see, e.g., Batcher (1976), Siegel (1979), Lipovski (1977)). But it appeared that even a limited capability would add at least $20,000 to $50,000 in costs. These are not large sums, but they are appreciable with respect to our $150,000 total budget. And the expected increases in performance seemed very small (although such a capability would be very attractive from the research point of view, in helping us determine better future designs).

Parallel Fetches for Logic Vs. 8-Bit Arithmetic Capabilities

The 8-bit wide processor could be designed to fetch, e.g., 4 bits from its own memory and 1 bit from each of the 4 square neighbors, or 8 bits from the 8 square and diagonal neighbors. This is very attractive for the frequently used logical operations that combine features in thresholded images. But the ability to handle 8-bit arithmetic, for numerical operations on grey-scale and colored images, and for small weights, appeared to be much more useful. And using the processors for 8-bit processors increases their power substantially over the typical array's 1-bit processors.

Our estimates are that this 16 by 16 array should be almost as powerful as a 32 by 32 DAP or a 96 by 96 CLIP4 for many types of operations (although DAP handles some matrix operations extremely well). The 8-bit processor will often give an 8-fold increase in speed. CLIP can do a true parallel operation over the nearest neighbors. But we estimate that, although this is very useful for the "lower levels" of image processing it is rarely needed at higher levels, or for number-crunching, and therefore in our mix of problems would give little increase in performance. DAP is 4 times as big, and 2 or 3 times as fast. CLIP is 36 times as big, but 10 to 20 times slower. DAP, and especially CLIP, because of their smaller memories, are constrained to handle smaller problems, or to slow down because of additional input-output of partial arrays.

These are very rough estimates, and one can only tell by extensively comparing the different systems after a good range of problem-mixes has been developed. But it is encouraging that such a one-of-a-kind system, quite different and with a number of additional network abilities, should be competitive at all.

Discussion: More General Converging Layered Systems, and Plans

The Array/Net integrates array and network very simply and intimately, since both share a common memory. With the present design, this means that only the array Slaves or the network Master can be executing.

This suggests the possibility of a multi-layered system where a processor at each "higher" layer looks at the memory shared with a number of "lower" level processors. This forms a system of converging layers in the shape of a pyramid.

For example, the lowest-level largest array (at the base of the pyramid) might be a typical SIMD system of 1-bit processors. The next layer might have 1/4th the number of 4-bit processors; the third layer 1/4th the number of 16-bit processors; the 4th layer 1/4th the number of 32-bit processors.

A deeper pyramid could be built, since there is no need for exact compatability in the address size of a Master and all of its Slaves. The important point is that such a system could com-

municate through its memory in a very simple, and appropriate, way. For such a system would, essentially, be used to pipeline (at the same time reducing) an image from base to apex of the pyramid. The output from each layer of processors would naturally be the input for the next layer of processors, and it resides, just where it should, in that next layer's memory, immediately available for further processing.

Each processor would share some memory with its offspring processors toward the base of the pyramid; and it also would share some memory with its parent toward the apex of the pyramid.

We have not decided to go ahead with the construction of the Array/Net, for several reasons:

Our (relatively firm) final cost estimates were about $170,000 (about half for chips and hardware), significantly exceeding the money in hand. (It was not certain that all the features could have been incorporated at our original estimate.)

The effort needed to develop software for this raw hardware seemed too great to expend, especially considering that our major thrust is to develop algorithms and programs for parallel arrays and networks.

It appeared that it would be possible to purchase an array that had already been designed and built, and for which software would be provided, elsewhere. But, although our decision not to build this system was made in August, 1979, it is still not clear whether we will be able to purchase, rather than build, a complete system.

Summary

The Parallel Pyramidal Array/Net system was designed to operate in several different modes:

1) as a 16 by 16 SIMD array of 256 1-bit processors;

2) as a 16-processor MIMD network of 8-bit, 16-bit, 24-bit or 32-bit Master processors (connected over a bus, and also as a 4 by 4 array);

3) as a set of sets of 2 to 16 SIMD 1-bit processor sub-arrays;

4) as a set of sets of 2 to 16 SIMD Master processors in a network;

5) as a set of sets of 4 by 4 1-bit processor SIMD sub-arrays and MIMD Masters.

The Array/Net is made of 16 Groups, as follows:

A Group consists of a 4 by 4 array of 16 (8-bit processor) Slaves, each connected to its 4 square neighbors. All Slaves execute the same instruction, under the control of a 64-bit Master controller. Each Slave has 16K bytes of memory. The entire Group (which is approximately 400 16-pin-equivalent chips) resides on a single multiwire board.

The Array/Net is made from 16 Groups, arranged in a 4 by 4 pattern, giving a 16 by 16 array of 256 Slaves.

Each Slave can fetch from its own memory and from the memory of its 4 nearest square neighbors, and it can store into its own memory. (Leftmost and rightmost columns wrap around, as do topmost and bottommost rows.) The Masters can intercommunicate through their Slaves, in this same manner, and also over their common bus.

Pictures are shifted from television camera, tape or disk into every row of the leftmost column. Code and other information is also input from the VAX over the bus.

The Array/Net was designed within several constraints: Our budget was limited to around $150,000 (which seems a reasonable price for this kind of research tool). We needed to design and build using existing chips; but none of the new array processor LSI chips were available. We wanted a system that we could use to develop and run algorithms for perception, number crunching problems, and artificial intelligence and data base management problems. For this we needed great increases in throughput from a high degree of parallelism (in SIMD mode), and also the flexibility and power of a network of independent MIMD processors. Our goal was a combination of array and network that would make use of and enhance the features of each.

References

anon, Unpublished paper on ASPRO, Goodyear-Aerospace, Akron, Ohio, 1979.

Batcher, K.E., The flip network in STARAN, Proc. 1976 Int. Conf. on Parallel Processing, Aug., 1976, pp. 65-71.

Batcher, K.E., Architecture of a massively parallel processor, 7th Annual Symp. on Computer Arch., ACM, 1980, 168-174.

Cryer, C.W., Successive over-relaxation methods for solving linear complementarity problems arising from free boundary problems, Proc. Seminar on Free Boundary Problems, E. Magenes (ed.), Pavia, 1979.

Despain, A.M. and Patterson, D.A., X-tree: a tree structured multi-processor computer architecture, Proc. Fifth Annual Symposium on Computer Architecture, April, 1978, 144-151.

Duff, M. J. B., CLIP4: a large scale integrated circuit array parallel processor, Proc. Int. Joint Conf. on Pattern Recognition, 1976, 4, 728-733.

Levine, M. D., A knowledge-based computer vision system, In: Computer Vision Systems, A. Hanson and E. Riseman (eds.), New York: Academic Press, 1978, pp. 335-352.

Lipovski, J., On a varistructured array of microprocessors, IEEE Trans. Comp., 1977, 26, 125-138.

Reddaway, S.F., DAP - a flexible number cruncher, Proc. 1978 LASL Workshop on Vector and Parallel Processors, Los Alamos, 1978, pp. 233-234.

Scott, A., Neurophysics, New York: Wiley, 1977.

Siegel, H.J., et al., PASM: A Partitionable Multimicrocomputer SIMD/MIMD System for Image Processing and Pattern Recognition, School of Electrical Engineering TR-EE 79-40, Purdue University, West Lafayette, 1979.

Tanimoto, S. L. Pictorial feature distortion in a pyramid, Comp. Graphics Image Proc., 1976, 5, 333-352.

Tanimoto, S. L. Regular Hierarchical Image and Processing Structures in Machine Vision, In: Computer Vision Systems, A. R. Hansen and E. M. Riseman (eds.), New York: Academic Press, 1978, pp. 165-174.

Uhr, L. Layered "recognition cone" networks that preprocess, classify and describe. IEEE Trans. Computers, 1972, 21, 758-768.

Uhr, L., "Recognition cones" that perceive and describe scenes that move and change over time, Proc. Int. Joint Conf. on Pattern Recognition, 1976, 4, 287-293.

Wittie, L.D., Efficient message routing in mega-micro-computer networks, In: Proc. 3d Annual Symposium on Computer Architecture, New York: IEEE, 1976.