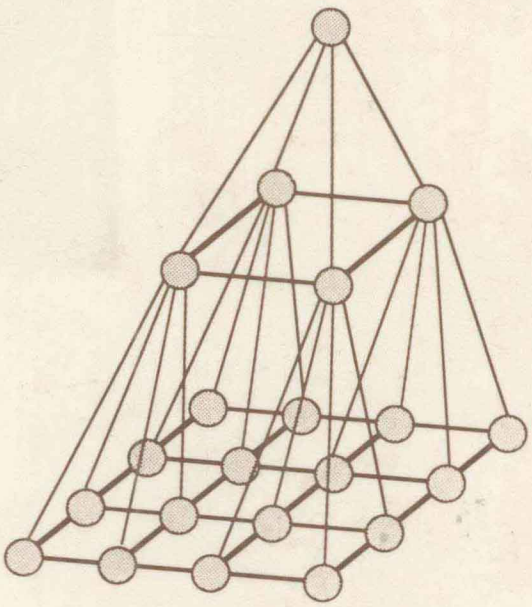
MAPCON IV

Multiprocessor and Array Processor Conference

SPECIAL PROCESSING

Edited by

Howard L. Johnson



SCS INTERNATIONAL

THE SOCIETY FOR COMPUTER SIMULATION INTERNATIONAL

MAPCON IV

SPECIAL PROCESSING

Proceedings of the Fourth SCS Multiconference on
Multiprocessors and Array Processors
3-5 February 1988
San Diego, California

Edited by
Howard L. Johnson
Information Intelligence Sciences, Inc.

A Society for Computer Simulation International (Simulation Councils, Inc.) publication
San Diego, California

Rosemary A. Whiteside, Managing Editor

SIMULATION COUNCILS, INC.
(The Society for Computer Simulation International)
P.O. Box 17900
San Diego, California 92117

ISBN 0-911801-31-6

PRINTED IN THE UNITED STATES OF AMERICA

MAPCON IV

MULTIPROCESSOR AND ARRAY PROCESSOR CONFERENCE, 1988

Preface

This Proceedings contains papers from most of the presentations given at the Multiprocessor and Array Processor Conference (MAPCON) held in San Diego, California, in February 1988, under the sponsorship of The Society for Computer Simulation International. The previous MAPCON was held in 1987 and was a successor to the earlier Peripheral Array Processors Conferences in 1982 and 1984. The 1987 conference addressed the emerging mini-supercomputer products in many scientific applications.

This 1988 conference departed from the traditional topic of vendor products to instead examine the state of the emerging science and technologies associated with "Special Processing". Chip technologies and automated design/engineering software have opened the way for research laboratories and companies to build special purpose devices that achieve high performance/cost benefits not attained through general purpose approaches.

This volume consists of papers selected from abstracts submitted in response to an SCS Call for Papers. They have been grouped into appropriate subtopics of special processing. These subtopics were used to organize the conference and the contents of this Proceedings.

Part 1: Introduction and Overview
Part 2: Systolic Arrays, Special Function Units and Accelerators
Part 3: Special Processors for Simulation
Part 4: Committing General Solvers to Hardware
Part 5: Image and Graphics Processors
Part 6: Applications Topics

As currently planned, the 1989 MAPCON will again follow the previous format of presenting vendor products in a way beneficial to potential users. A special topic will be chosen for the 1990 conference.

Howard L. Johnson, Editor
Information Intelligence Sciences, Inc.
Aurora, Colorado

Part 1
INTRODUCTION AND OVERVIEW

CONTENTS

Part 1
INTRODUCTION AND OVERVIEW

ISBN 0-911801-31-6

Special hardware to achieve high performance: The science

Howard L. Johnson
Information Intelligence Sciences, Inc.

BACKGROUND

We are accustomed to using general purpose hardware to solve computational problems in individual applications. Until recently, building one's own computer for a specific problem was out of the question. The development cycle for a machine generally took 10 years of work by a variety of highly skilled, extremely scarce people. This required a significant initial investment, where investors normally seek a low risk. Further, the economic advantages of high volume demanded a significant market to justify initial development cost.

There have been a few cases where it was felt that gain in performance/cost justified building special machines:

- Signal processors for defense, medical, and oil applications of fast Fourier transforms, convolutions, and similar algorithms
- Database management machines to handle large database applications
- Special functional accelerators for graphics and image processing
- Fault-tolerant machines for critical applications requiring greater redundancy than available in general purpose architectures.

WHAT HAS CHANGED?

The single most important contribution to change in interest in special processors is emphasis by top universities in all aspects of computer sciences. A large cadre of graduates capable of addressing the problems associated with special computing is being interjected annually into all areas of the scientific work force.

A second factor is the new environment created by advances in the semiconductor technologies. Off-the-shelf VLSI chips (and soon ULSI, where chip complexity exceeds 2 million devices) provide inexpensive building blocks. As is often the case, DoD is leading the way (e.g., in the VHSIC program). Adaptation can be accomplished at a very low level in gate arrays, standard cell systems, programmable logic arrays, and use of silicon compilers.

Automated techniques and tools for developing algorithms, designing corresponding architectures, and implementation of the hardware are now supporting two-to-three year developments of complex systems.

Another important driver is that entrepreneurs are seriously considering applications that require throughput not achievable by general purpose approaches for cost and/or performance reasons.

CLASSES OF SPECIAL HARDWARE

The powerful and highly complicated general purpose computer has not been the product of amateurs. Designs offered in the marketplace used the latest in technology along the course of the design cycle. Few people have qualified for involvement in this enormous undertaking. Costs and risks have been high and a payoff has resulted only from a high volume of sales applied to a wide set of applications.

This did not stop the theoreticians from imagining the possibilities for implementing specific algorithms in hardware using massive parallelism. [1] recalls important papers by J. von Neumann "The General Logical Theory of Automata," 1951 [2], F.C. Hennie "Iterative Arrays of Logical Circuits," 1961 [3], and D.L. Slotnick, W.C. Borck, and R.C. McReynolds, "The Solomon Computer," 1962 [4].

In current efforts, there seem to be four directions for application specific computing:

- Seeking out concurrency in algorithms and implementing it at the cell level in cellular automata such as cellular arrays, systolic arrays, or wavefront machines
- Adapting the architecture to special characteristics and requirements of an application, including vector or array dimension, word size, computational mix, computational functionality; as well as matched memory, I/O, and processing
- Speeding up often used functions with special function units and add-on accelerators that use concurrency and/or special hardware (optical or analog techniques)
- Committing to hardware the general problem solvers previously available and used extensively in software versions

ARCHITECTURAL CONSIDERATIONS

Examining characteristics of parallel and vector architectures provides insight to approaches to designing special processors.

Control Mechanism

Most general purpose architectures are driven by a clock, where a fixed relation exists between clock steps and functions performed. In architectures with massively parallel cooperating elements, compensating for clock skew can be a problem. A few applications

adapt themselves to even a tighter lockstep of functions to gain interface simplicity. At the other extreme, data flow approaches eliminate difficult synchronization issues since computed results are the synchronization for the next step. For a single application or a single algorithm, the data flow graph identifies where parallelism can be exploited at any of the subinstruction, instruction, function, and program levels. However, in data flow, buffers must be well planned to handle processing delays. Reduction techniques assist in the data flow analysis, however, for known, well specified problems, reduction as a control mechanism is probably not useful in special architectures.

Data Mechanism

We are accustomed to referencing data in memory by address. Another approach is to reference by content. The associative search of data moving through microprocessor comparators is often simpler and faster than determining specific addresses for storage and retrieval.

Concurrency Type

Special architectures achieve their greatest performance gains by exploiting concurrency. Concurrency can exist in the multiple execution of a single instruction stream or of multiple instruction streams executed simultaneously. It can exist in the chaining of functions to create process pipelining and in the movement of data. Implementation is in array operations on vectors, in pipelining of multistaged functions or operations, in machines with independent and sometimes heterogeneous processing elements, and in the interleaving of memory. When one considers what concurrency can exist at the subinstruction, instruction, and process level, it is easy to see that sequential processing has been done more for simplicity of control and stucture, not because it is right.

*As in splitting the distance of a transcontinental trip between a bicycle and an airplane, it makes very little difference timewise to charter a faster plane. Emphasis must be placed on the slowest mode of computation and on those activities that occupy the critical path. Nature provides much less synchronous (or vector) parallelism than what man imposes for simplicity (e.g., equal grids or equal time steps in an algorithm). From a computation standpoint on a single problem, these unnecessary inefficiencies resulting from forced regularity may not be desirable.

Connectivity

In a special processor, problem dictated connectivity exists; nothing-more nothing-less. Also, commensurate with the demands of the problem, the communications system has the appropriate bandwidth, direction of data flow, multiplexing of problem data types, and the necessary communications flexibility. General connection approaches (crossbar, hypercube, bus) are designed to best accommodate unknown traffic patterns. In a special processor the data flow is usually known, often precisely. Connectivity issues such as topology isomorphism, scalability, balance, and logical reconfigurability are usually not issues; only a set of mutually compatible rates, minimizing design costs where high performance is not required.

Communications

Direct connectivity and limited switching accommodates predictable traffic patterns and volumes. A set of mutually compatible parallel data flows should, in addition, overlap processing to the maximum degree possible, and achieve a mutually compatible time budget. The best technologies available should be used to accommodate high performance demands, with economical technologies used elsewhere.

Memory Distribution

Special architectures often give rise to the functionalization of memory and dedication to a single purpose. Both local and global memories are found. Knowledge of the data flow can allow staging and buffering to minimize use of high speed, expensive technologies.

Partitioning

In special processors, the partitioning is natural and the architecture general corresponds to the natural partitioning of the problem, avoiding forced partitioning (e.g., divide and conquer).

Scheduling and Synchronization

Activities are usually known and deterministic. Synchronization can be accomplished by scheduling separate activities to a clock step. Even more natural is synchronization and scheduling of activities following a data flow or reduction approach.

Instructions

Often, the entire process can be incorporated in hardware in a special purpose machine. One of the biggest problems, however, is that a slight change to the formulation of the problem results in an expensive hardware design change. The alternative is programmability. This could be at the microprocessor level or a single instruction stream being fed to many processing elements. In other cases, the program can be data driven using simple control codes to replace complex instructions.

A sophisticated special processor can be accompanied by an applicative language that identifies the characteristics of the problem (e.g., pixels, dimensions, step size, and presence or absence of computational terms). This greatly simplifies operating the processor for non-programmers.

Data

Scaling requirements of $\pm 10^{**}38$ in 32 bit floating point arithmetic and $\pm 10^{**}308$ in 64 bit floating point arithmetic are usually overkill in individual problems. The same is true with the significance representable by 24 bits (7 decimal digits of significance) or 48 bits (14 decimal digits of significance). Hardware logic can be saved by employing fixed point and required manual scaling through software, thereby reducing cost and potentially increasing performance.

Data Structure

In a special processor, data are addressable and transmitted in natural quantities. A word is often

inefficiently small and efficiency of a page depends on statistics and probabilistic usage. Instead, for example, a pencil of data along a 3D solution space, a row of data in a video scene, or a sentence of textual information are more appropriate.

PERFORMANCE POTENTIAL

The increase in performance between a standard off-the-shelf architecture and a special purpose architecture depends on the problem, how adaptable it is to the standard architecture, and possible exploitation by a special architecture. If a problem has the right vector length and the correct I/O mix for the Cray (say the XMP-4), one would be hard pressed to build a special processor faster than the Cray. However, most typical numerical applications only run at 25% - 35% of peak speed on the Cray. A non vectorizable problem may only be able to use the scalar part of the Cray, reducing the performance even more. It is important to remember that non vectorizability does not necessarily mean non parallelizability. Most problems are parallelizable at some level. Further, a 2D or 3D parallel array architecture does not suffer pipeline startup and slowdown costs present in a vector architecture.

Most general purpose processors are either sequential and therefore depend on technology for their speed, or they are built to excel in a small class of numerical problems (e.g., vector and matrix, linear equations, or FFT). Consider the processing required for artificial intelligence applications (e.g., symbolic processing and optimal search) or highly repeated functions (e.g., spatial transformations or database search). There is a very high potential performance gain available when these are committed to hardware.

The tradeoff is then between standard chips available or buildable and the technology that the large manufacturers can bring to bear (e.g., IBM and Cray). These latter include clock speeds ten to twenty times greater, higher density chips, faster internal communications, and massive storage. Typical performance tradeoff situations between types of general purpose processors and special purpose processors might be:

GENERAL PURPOSE	SPECIAL PURPOSE
Supercomputer	
Technology Advantage x10-20	Adaptation Advantage x5-1000
General Purpose Sequential	
Technology Advantage x5	Adaptation Advantage x100-10K
General Purpose Parallel	
Technology Advantage x2	Adaptation Advantage x50-5000
Other Factors	
Reliability	Fault Tolerance
Accommodate Change	Real Time
Standard Language	Applicative Language

In conclusion, we can expect in a special purpose processor from zero to over three orders-of-magnitude performance advantage over current general purpose processors, depending on the problem. This suggests analysis of problems in detail for parallel, systolic, or other characteristics on which a special approach could capitalize.

COST POTENTIAL

Performance advantages between the cases above probably will hold with time, though specific performance will increase. Cost advantage is more difficult to determine and realize. Factors to be considered include:

- Cost, schedule, and technological risk in design/development
- Technological shelf-life
- Volume parts/labor cost reduction
- Commercial vendor margin (amortization of development costs)
- Engineering and design costs (in-house versus purchased)
- Development costs for first few (prototype)

Technology is changing by about an order-of-magnitude every five to seven years with product top positioning of three years or less. A special processor under normal circumstances should be conceived and built in less than three years to avoid the next generation. Development costs will probably be at least twice that of the vendor engineering laboratories.

The introduction of cost, schedule, and performance risks in the development of a special processor dictates that projected cost/performance should be significantly less (e.g., half) that of the off-the-shelf product.

Doing the project without experienced qualified people greatly increases risk and risk impact.

WHAT IS NEEDED TO ADVANCE THE SCIENCE

The goals are to bring about the creation of standard inexpensive chip sets and connections that allow a three year design/prototyping cycle. The manufacturers must accomplish this with a less sophisticated designer in mind, providing more and better instructions and handholding services.

As in any science, knowledge is gained both from theory and experiment. Simulation is be used to verify the theory. There are two important pursuits: developing architectures from the application algorithms and implementing the algorithms. A methodology for architecture tradeoffs can include the current and projected cost and risk experience factors that are decreasing at a very rapid rate due to development tools.

The science needs tools: computer aided experimentation with approaches, memory partitioning, data flow algorithms, critical path models and queueing models. However, like any frontier, it must be conquered by pioneers like von Neumann, Hennie, Slotnick, H.T. Kung, C.E. Leiserson [5], and a multitude of others.

References:

[1] Fortes, Jose A. B., and B.W. Wah "Systolic Arrays from Concept to Implementation," IEEE Computer, Vol.20 No. 7, July 1987, pp. 12-17.

[2] von Neumann, J., "The General Logical Theory of Automata," in Cerebral Mechanisms in Behavior - The Hixon Symposium, L.A Jeffries, ed. 1951, John Wiley & Sons, New York.

[3] Hennie, F.C., Iterative Arrays of Logical Circuits, 1961, MIT Press, Cambridge, Massachusetts.

[4] Slotnick, D.L., W.C. Borck, and R.C. McReynolds, "The Solomon Computer," Proceedings AFIPS Fall Joint Computer Conference, 1962, Spartan Books, Washington D.C., pp. 97-107.

[5] Kung, H.T. and C.E. Leiserson,"Systolic Arrays (for VLSI)," Sparse Matrix Proceedings 1978,1979, Academic Press, Orlando, Florida, pp. 256-282.

Part 2
SYSTOLIC ARRAYS, SPECIAL FUNCTION UNITS & ACCELERATORS

ISBN 0-911801-31-6

Flexible communications simulation on a high-speed PC workstation

Kurt Matis
MODCOM, Inc.
14 Wood Dale Drive
Ballston Lake, N.Y. 12017

and

James W. Modestino
ECSE Department
Rensselaer Polytechnic Institute
Troy, N.Y. 12180-3590

ABSTRACT

We describe the application of a new board-level arrary processor to the continuous-time simulation of signal processingsystems. This product, called the VORTEX, is manufactured by SKY Computer Systems, Inc. A 20 MLOP peak throughput capability allows a PC based host to rival the overall computational capability of typical mainframes. A unique software development environment provides easy conversion of FORTRAN source code to execute within the VORTEX. This environment includes a vectorizing preprocessor, called VEX, that allows various levels of code conversion, depending on the time requirements of each module. A memory-mapped interface to the VORTEX avoids the latency of DMA transfers and provides the user with flexibility in partitioning the simulation problem.

We demonstrate the utility of the PC-based approach to simulation within the context of the Workstation Communications Simulator (WCS) System. This system provides the capability for interactive simulation of point-to-point digital communication links. This system, originally developed on a VAX 11/780 acting as host to a Floating Point Systems array processor, has been ported to the PC/AT-VORTEX configuration. We provide typical graphical simulation outputs to illustrate the useage of the WCS system.

I. INTRODUCTION

It's generally well-appreciated that digital simulation can provide a useful and effective adjunct to either analytical performance evaluation or direct hardware evaluation of modern engineering systems. A number of simulation techniques and methodologies are described elsewhere in these proceedings. In this paper we describe our experience in the design and development of a high-speed workstation-based approach to interactive simulation of digital point-to-point communication systems. Our approach to the design of this system depends crucially on the unique features of a new type of application procesor, called the VORTEX, manufactured by SKY Computer Corporation. This single-board processor provides a hierarchical software development environment and includes a multitasking executive supporting parallel operation of multiple processors.

Digital simulation of communication systems (cf. [1]) is generally a computationally demanding task if executed on present-day general purpose machines. This is due mainly to the large number of repetitive signal processing operations that must be performed in order to obtain a statistically valid measure of system performance. The high cost and/or long running times associated with execution of centralized large main-frame computing facilities has motivated the search for more effective cost performance alternatives.

One approach, as described in [2], is to make use of a dedicated minicomputer hosting a floating-point array processor. These array processors are, in essence, peripheral floating-point accelerators accessed through a high-speed bus under control of the host machine which provides overall system control. For example, the Interactive Communications Simulator (ICS), as described in [2], made use of a DEC machine (either a VAX or PDP-11) hosting a Floating Point Systems, Inc. AP-120B floating-point array processor. This processor utilizes technology which is approximately a decade old and offers a potential throughput rate 12M floating-point operations per second (12 MFLOPS). In this approach, all signal processing software is executed in the AP-120B and is written in assembly language. Executive control software and graphics are written in FORTRAN and are executed in the host processor.

While the ICS has proven useful as both a research and educational tool, several factors have motivated the search for still more effective cost/performance solutions to communication systems simulation. In the first place, the hardware configuration required to support the ICS is, although fairly commonplace, still relatively expensive and difficult to justify for a dedicated simulation facility. Secondly, the software is not readily portable to other host configurations. Finally, there does not exist a rich programming environment for efficient development of AP-120B assembly language code for new signal processing modules which, through evolution, have been found to be useful additions to the original ICS repertoire.

Fortunately, recent developments in high-speed workstation technology have provided solutions in many of those problems and at this time provide a very attractive cost/performance alternative hardware configuration. These workstation are relatively inexpensive and with standardized operating systems, together with device independent drivers, result in highly portable software. Furthermore, these workstations provide a rich program development environment allowing relatively easy addition of new software modules. Finally, and perhaps most importantly, single-board memory-mapped array processors are available for floating-point acceleration offering potential throughputs as high as 20 MFLOPS - a considerable improvement over the large minicomputer-hosted array processors such as the AP-120B.

In this paper, we will describe a PC workstation-based communications simulator called WCS. This system is essentially a modern low-cost

version of ICS. WCS was developed to meet the needs of a wide variety of users including: system developers, communication researchers and educators desiring a cost-effective, high-performance simulation tool.

The remainder of this paper is organized as follows: In Section II we provide an overview of the VORTEX application processor describing some of the features which make it particularly well-suited to the simulation application. This is followed, in Section III, with a description of the WCS system and how it employs the VORTEX applications processor.

In Section IV, we describe the operation of the WCS system through illustration of typical graphical output and probide triming benchmarks for typical simulation scenaries. Finally, in Section V, we provide a summary and conclusions.

II. The VORTEX Application Processor

The block diagram in Fig. 1 illustrates the basic architecture of the VORTEX Application Processor and the interface to the iLBX-II bus. The VORTEX is a high-speed arithmetic processor capable of performing integer and floating-point computations at a rate of 20 MFLPOS per second. The VORTEX employs a powerful instruction set and an object-oriented software architecture to perform scalar, vector and matrix operations. It performs high-speed floating-point arithmetic in IEEE-P754 single and double-precision formats as well as two's complement integer and logical functions. Used as a coprocessor in the ENCHANCED WCS system, a single VORTEX provides an increase in computational throughput of about three orders of magnitude.

The VORTEX consists of a user-programmable control processor, an arithmetic unit that executes floating-point, integer and logical operations; a host bus programmed I/O (PIO) interface and a large internal memory. A unique feature of the VORTEX architecture is the fact that both the PIO register set and the large internal memory are memory-mapped to the host 80286 address space. Consequently, the VORTEX memory is directly addressable by the host application program without the latency of DMA transfers. This has very significant ramifications to overall simulation efficiency in systems employing such an attached processor.

In the conventional approach, data is transferred by means of a DMA initiated by the host application program. Depending on the host and operating system, the overhead incurred in initiating a host-AP transfer can be on the order of several milliseconds. In order to obtain resonable efficiency, the simulation problem must be partitioned in such a fashion to allow the AP processing time to be greater than the DMA overhead plus transfer time. For some communications subsystems, such as digital filters, this is quite easily accomplished, since generic routines are available for the implementation of such functions. For more specialized modules that cannot easily be decomposed into simpler, generic, operations this is usually not the case. Because of complicated internal logic and special operations, subsystems such as decoders and adaptive equalizers do not admit such a representation. In this case, the simulation developer is usually forced to program applications modules directly in AP microcode. Needless to say, this makes the incorporation of some new types of modules time-consuming even for the experienced developer.

With the memory-mapped approach to interfacing, an attached processor can process much smaller segments of a simulation problem during each call to the VORTEX. Even scalar operations are increased in speed due to the speed of the command decoder and a low latency arithmetic unit. In most cases, an executing applications program can be working in parallel with the VORTEX, performing outer loop control and supervision while issuing function calls to the VORTEX at the appropriate times. A schematic diagram of this interface is shown in Fig. 2. Our approach is to perform all computations within a common data area which actually resides in VORTEX memory in the ENHANCED WCS system. This FORTRAN common block is typically mapped above the 1 Mbyte boundary in the 80286 address space. The actual address is strappable, and depends on the amount of installed physical memory. In the WCS, the area between 1-2 Mbytes is used for dynamic buffer pool storage so the applications processors are mapped above this.

VORTEX *Programming Environment*

Due to the nature of the memory-mapped hardware interface and the ensuing increased flexibility for problem partitioning, a new approach to AP software development is required. Application processor programming aids have traditionally been limited to FORTRAN-callable function libraries and microcode development support. Compliers have been developed for some machines but, by and large, these have tended to be relatively inefficient.

The VORTEX programming environment, as indicated in Fig. 3, affords a four-level hierarchical approach to applications processor code development. The first three levels invoke a vectorizing preprocessor, called VEX. The VEX preprocessor provides the applications developer with access to VORTEX facilities directly from a high-level language. VEX reads a FORTRAN-77 file and translates this into an equivalent file containing instructions for both the host and the AP.

At the first level, the programmer can use VEX transparently to translate an existing FORTRAN application to an equivalent one adapted for use with the VORTEX. The resulting file contains some residual FORTRAN control statements, interspersed with calls to elementary VORTEX functions.

At the second level of access, the user familiar with VORTEX operations can rearrange the original source code to include explicit calls to VORTEX operations. This is accomplished through the insertion of explicit calls to invoke specific VORTEX operations.

At the third level of access, the programmer can use the VEX facilities to create new VORTEX command subroutines that implement time-critical or frequently used functions. The subprogram is translated into a special VORTEX command subroutine that gets loaded into VORTEX memory at the same time as the VORTEX microcode.

The final level of access to the VORTEX involves custom microcode development using the VORTEX microassembler. We have found it necessary to develop microcode for the WCS system only for the lowest-level time-critical functions such as random