Proceedings of the Second International Conference on

Vector and Parallel Processors in Computational Science

Oxford, England
28 – 31 August 1984

Editors

I. S. DUFF
J. K. REID

NORTH-HOLLAND – AMSTERDAM

Vector and Parallel Processors in Computational Science

Proceedings of the Second International Conference on
Vector and Parallel Processors in Computational Science

Oxford, 28–31 August 1984

Editors

I.S. DUFF
J.K. REID

AERE, Harwell

1985

NORTH-HOLLAND – AMSTERDAM

ISBN 0 444 86974 3

PUBLISHED BY:
NORTH-HOLLAND PHYSICS PUBLISHING, a division of
Elsevier Science Publishers B.V.
P.O. Box 103
1000 AC Amsterdam, The Netherlands

SOLE DISTRIBUTORS FOR THE USA AND CANADA:
Elsevier Science Publishing Company Inc.
52, Vanderbilt Avenue
New York, N.Y. 10017, USA

REPRINTED FROM COMPUT. PHYS. COMMUN. VOL. 37 (1985) Nos. 1–3

Library of Congress Cataloging-in-Publication Data

International Conference on Vector and Parallel
Processors in Computational Science (2nd : 1984 :
Oxford, Oxfordshire)
Vector and parallel processors in computational
science.

"Reprinted from Computer physics communications,
vol. 37"--T.p. verso.
Includes index.
1. Parallel processing (Electronic computers)--
Congresses. I. Duff, Iain S. II. Reid, John Ker.
III. Computer physics communications. IV. Title.
TK7885.A1I564 1985 004.35 85-15975
ISBN 0-444-86974-3

Printed in The Netherlands

VECTOR AND PARALLEL PROCESSORS
IN COMPUTATIONAL SCIENCE

PREFACE

In August 1981, a highly successful conference on Vector and Parallel Processors in Computational Science (VAPP) was held in Chester and Proceedings for this conference were published in Computer Physics Communications 26 (1982) 217–489 as a special issue. The growth in the design and use of vector and parallel machines has risen sharply since then and it was felt appropriate to stage VAPP II in August 1984, this time in Oxford. This volume contains papers from most of the invited talks and from several of the contributed talks and poster sessions. Indeed, out of the 64 presentations at the Conference, 45 are included in this volume.

As at the previous conference, the programme of talks covered the three areas of hardware/software/languages, numerical methods and algorithms, and applications. We have adhered to the same subdivision in these proceedings, except that some papers which span more than one area have been grouped as "general papers".

It is interesting to reflect on the development of the field over the past few years. Although there are a few more machines on the market (for example, the CRAY X-MP, the FPS 164, the Denelcor HEP, and vector machines from Fujitsu and Hitachi in Japan), the most noticeable change is in the sophistication of the software support and the maturity of the user community. In particular, much more is understood about the design and use of MIMD machines (the X-MP and HEP being examples of these) where quite independent processes can be performed simultaneously.

The papers in this volume include descriptions of new machines (both research and commercial machines), languages and software aids, and general discussions of whole classes of machines and their uses. Numerical methods papers include Monte Carlo algorithms, iterative and direct methods for solving large systems, finite elements, optimization, random number generation and mathematical software. The specific applications covered include neutron diffusion calculations, molecular dynamics, weather forecasting, lattice gauge calculations, fluid dynamics, flight simulation, cartography, image processing and cryptography. Most machines and architecture types are being used by these applications.

The conference itself was truly international with 45% of the delegates from outside the United Kingdom. The papers reflect this and also the fact that one and a half times as many attendees came from laboratories and industry as from academe.

The sell-out crowd of nearly 300 illustrated the continuing interest in the area of vector and parallel processors and the fact that all talks, including those on the last day, were well-attended testified to the energy and commitment of delegates. Handling a conference of this size requires a considerable administrative effort, and we are indebted to Les Evans, Enid Eakins and Rita Holdbrook of the Education and Training Centre (AERE Harwell) and to our secretary, Rosemary Rosier, for enabling the smooth functioning of such a large meeting. We were also greatly aided by the staff of Keble College who readily adapted to our special requirements and helped make a most pleasant conference atmosphere.

The meeting was sponsored by AERE Harwell and we are grateful to them for underwriting the meeting and risking financial embarrassment had our confident prediction of 250 attendees been an overestimate. Financial support for some of the social events was provided by Cray Research (UK) Ltd and by Floating Point Systems (UK) Ltd, and we are grateful for their generosity.

Finally, we would like to thank the programme committee both for the organization of such an attractive programme and for doing the lion's share of the refereeing for the proceedings, and we would like to thank Keith Roberts, who was Principal Editor while we were preparing this volume, for being so helpful with the processing of the papers.

March 1985

I.S. DUFF and J.K. REID

CONTENTS

Computer Physics Communications 37 (1985) 1–5
North-Holland, Amsterdam

APPLICATIONS OF MIMD MACHINES

B.L. BUZBEE

Computing and Communications Division, MS B260, Los Alamos National Laboratory, Los Alamos, NM 87545, USA

Parallel processing via the application of MIMD machines offers the promise of high performance, and experience with parallel processing is accumulating rapidly. This paper briefly surveys recent results from three classes of MIMD machines – shared memory systems, non-shared memory systems, and a dataflow system. This data confirms that rapid progress is being made in the application of MIMD machines and that parallel processing can yield high performance. It also confirms that major research issues remain to be addressed.

1. Introduction

We are entering an era in which fundamental changes to computer architecture seem assured. In particular, we are about to cross the threshold from serial computation to parallel computation via application of MIMD machines. We are motivated to make this transition by basically two factors.

1. Performance improvements in monoprocessor systems are deemed unlikely to exceed a few hundred percent over the next few years.
2. We need to solve a broad spectrum of problems whose computational complexity far exceeds the capability of the most powerful computing systems available.

Parallel processing is by no means guaranteed to produce the desired performance levels. On the other hand, it is guaranteed to engender significant changes in our approach to computation. This raises the question of what evidence is there that parallel processing will, in fact, yield high performance on a broad spectrum of applications? In other words, will it be worth the cost of the changes required to realize it? This paper attempts to shed some light on these issues by surveying a portion of the substantial amount of research experience with parallel processing that has accumulated in the recent past. We will begin with a short discussion of metrics for evaluating the performance of parallel systems, then we will look at experiences with two shared memory systems, two non-shared memory systems, and one dataflow system. The results confirm that broad and rapid progress is being made, at the same time major issues remain to be addressed.

2. Metrics

Our objective for parallel systems is high performance as a function of the parallelism invoked on a spectrum of applications. Given full-scale systems we can quickly determine the performance via direct usage. However, few full-scale systems exist, and those that do have only modest amounts of parallelism. On the other hand, relatively slow research systems exist that contain substantial parallelism. Thus, we need a metric that measures the performance as a function of parallelism and that offers some degree of comparability between systems. We will use the metric of efficiency. That is, let

T_1 = the time required to execute an application using an optimal serial formulation on a single processor;

$T(p)$ = the time required to execute the same application with a parallel formulation and using p processors.

Then,

$$\text{efficiency} = T_1/pT(p).$$

0010-4655/85/$03.30

Since efficiency usually lies in the unit interval, it offers comparability independent of absolute processor speed. However, the metric has one liability in that it does not clearly illuminate cases where $T_1 > T(p)$.

In general, this writer cannot guarantee that the efficiencies reported herein use an optimal T_1. Many experimentors measure T_1 using parallel formulation of the application executed on one processor. Then they attempt to correct for the additional work introduced by parallel formulation. Some experimentors fail to make even this correction. On the other hand, we ultimately need to solve very large problems, and many of the experiments discussed herein involve relatively small problems. That difference may more than compensate for any inaccuracies in the measurement of T_1.

3. Shared memory systems

Two full-scale shared memory systems are currently operational in the field, and experience with them is accumulating rapidly. These systems are the Cray X-MP-2 and the Denelcor heterogenous element processor (HEP).

3.1. Cray X-MP-2

The Cray X-MP-2 is one of the most powerful state-of-the-art systems. It has two vector processors sharing 32 banks of memory. One of the critical issues in the design of high-performance computers is I/O bandwidth both between the memories and the processors, as well as between memories and secondary storage devices. Careful attention was paid to these issues in the design of the X-MP-2; thus its overall system performance will provide a valuable source of information for future designs.

Some preliminary experiences with parallel processing on the X-MP-2 are summarized in table 1.

Table 1
Cray X-MP-2 performance

Application	Efficiency
Short-term weather forecast [1]	0.95
Plasma simulation [1]	0.93
3D seismic migration [1]	0.95
Linear algebra [2]	0.43–0.9
Aerodynamic simulation [3]	0.8–1.1

In view of the efficiencies obtained, these results are encouraging because some of the problems involved are not especially large. For example, the weather forecast application was two-dimensional with 100 by 192 grid points. The plasma simulation contains only 37000 particles. The linear algebra experiments involved matrices of dimensions ranging from 50 to 600 (note that an efficiency of 0.4 implies T_1 was less than T_2 for some of the systems).

The operating system for the X-MP-2 supports invocation of multiple tasks at the job level. Associated user-visible software includes library routines for task creation and scheduling, intertask communication and synchronization, and protection of resources shared among tasks. Fortran will be extending via a "TASK COMMON" capability. Associated data will be local to a task and shared by subroutines executed within the task.

3.2. The Denelcor HEP

The Denelcor HEP attempts to overcome many of the latencies that have traditionally impaired the performance of monoprocessor systems. It can have up to 16 processing element modules (PEMs) sharing up to 128 memories. Each PEM offers a second level of concurrency via a particularly innovative architecture. A PEM can multitask up to 50 user processes in an 8-stage instruction pipe that switches tasks every machine cycle. Further, the PEM architecture incorporates an aspect of dataflow. Each PEM is a register-to-register machine. If the requisite operands for a selected task are not yet available in local registers, then the instruction is "waved off" and another task is selected. The architecture also includes facilities for locking data in units of one memory word.

Computation of the efficiency metric on the HEP is complicated by the fact that optimal resutlts are usually obtained by multitasking 10 to 15 processes through the 8-stage instruction pipe.

Table 2
Denelcor HEP performance

Application	Efficiency
Plasma simulation [4]	≈1
2D Lagrangian hydro [4]	≈1
Reactor analysis [4]	≈1
Linear algebra [5]	0.7–0.9
Quadrature [6]	≈1
Ordinary differential eqs. [7]	0.5–0.8
Linear transport eqs. [8]	≈1

Table 3
PAX-32 applications

Application	Efficiency
"Eulerian" Monte Carlo	not recommended
"Lagrangian" Monte Carlo [9]	0.95–0.98
Gauss–Jordan elimination [9]	0.55–0.88
FFT [9]	0.70–0.85

More than eight processes are required to achieve optimal performance because some instructions require more than eight cycles to execute, for example, division and memory references. The efficiencies reported in table 2 reflect an arbitrary decision of the author to divide optimal speedups by the number of stages in the pipe (i.e., eight).

Details of the implementation of these applications can be obtained from the indicated references. From the perspective of this paper the key point is that high efficiences have been obtained from the HEP over a fairly broad spectrum of applications.

The HEP system includes a Fortran compiler that contains only two extensions to support parallel processing. With the possible exception of debugging it has proved to be remarkably easy to use. Thus, another significant result obtained from the HEP is evidence that Fortran can support parallel processing via a relatively modest set of language extensions and control structures.

4. Non-shared memory systems

As the number of processors increases in the system, the problem of sharing memory among them becomes increasingly complex. Thus, a key research question is the applicability of non-shared memory systems. Non-shared memory systems for which substantial experience is available include the PAX-32 and the hypercube.

4.1. The PAX-32 system

The PAX system is a two-dimensional rectangular array of processors. Communication is nearest neighbor with end around boundaries. Also, a global bus provides broadcast capabilities to all processors. An eight by four system (PAX-32) is operational and larger systems are under development. Experience with PAX-32 is summarized in table 3.

"Eulerian" Monte Carlo is a problem formulation in which physical space is subdivided into subspaces and each processor processes everything happening in one or more subspaces. Thus, in a time-dependent problem, processor loading may become unbalanced if particles become concentrated in a few subspaces. "Lagrangian" Monte Carlo is a problem formulation in which particles are divided equally among processors independent of spatial location, and thus processor loading is relatively stable over problem time.

One of the most significant results obtained from the PAX project is a predictive model for this class of architecture. The model was validated by accurately predicting the measured efficiences given in table 3. Using the model, performance of these applications was extrapolated to a Pax system incorporating thousands of high-performance processors and high-performance communication technology. These extrapolations predict that high efficiencies can be sustained for these applications on systems with up to 10000 processors.

4.2. The hypercube

The hypercube is a non-shared memory architecture that uses an elegantly simple interprocessor communications geometry. In an M-processor system, each processor is assigned a unique label from the set of integers zero through $M-1$. The Ith processor is connected to all processors labeled J such that the binary representation of J differs

Table 4
CIT hypercube applications

Application	Efficiency		
	$p = 8$	$p = 16$	$p = 64$
Laplacian [10]			
32×32 grid		0.85	0.70
64×64 grid		0.93	0.83
Matrix multiply [10]			
32×32	0.82		
4D Ising model [10]			
$12 \times 12 \times 12 \times 12$			0.97

from the binary representation of I in just one bit position. Thus, a key issue for this architecture is its applicability to a broad spectrum of applications.

Researchers at California Institute of Technology (CIT) have a 64-processor (6-cube) system operational, and results from it are summarized in table 4. Once again the efficiencies are impressive, and there is a high degree of confidence among many researchers that these efficiencies can be sustained over much larger systems.

5. Manchester dataflow machine

Dataflow systems have the potential to be the Rolls Royce of parallel processing systems because

1. The number of processors can be increased without changing applications code.
2. Computation is deterministic.
3. Parallelization is automatic.

The second of these points is particularly appealing because on many parallel systems computation is not necessarily repeatable. That is, successive executions of the same code using the same data may not yield identical results. This indeterminacy of computation reflects the asynchronous character of MIMD architecture. Indeterminacy is most painful in debugging, but it also manifests itself in other ways [11]. Although dataflow architecture enjoys popular support among computer scientists, relatively few systems are operational, and thus relatively little experience is available. A notable exception is the dataflow machine at the University of Manchester. This machine uses twelve processors and a tagged token architecture, and some results from it are summarized in table 5. When these results are combined with the potential of dataflow systems to maximize software productivity, then one quickly understands the worldwide interest in these systems.

Table 5
Manchester dataflow applications

Application	Efficiency
Quadrature [12]	$\geqslant 0.70$
FFT [12]	≈ 1
Laplace eq. [12]	≈ 1
Logic simulation [12]	≈ 1
Lee Router [12]	$\sim\ \sim 1$

6. Summary

We began this discussion by noting our need for higher performance computing systems and by asking for evidence that parallel processing will provide it. We then looked at an ensemble of experiments on three classes of MIMD architecture. Collectively, these experiments suggest the following:

1. A broad spectrum of scientific computation is amenable to parallel processing.
2. Significant gains in speed (efficiency) can be achieved through parallel processing, at least in systems with a few processors.
3. A small number of control structures may be adequate to support parallel computation. Further, only a small number of extensions to Fortran may be required (undoubtably, many extensions are desirable, but only a few appear to be required).

These are encouraging results. They reflect breadth and rapid progress in parallel processing research. They also show the impact and importance of having equipment available for experimentation. However, these positive aspects should not be overestimated because these experiments also expose some of the difficult issues that remain to be solved.

1. Software tools. Problem analysis and decom-

position for parallel formulation in nontrival. Debugging can be decidedly difficult because of nonrepeatability emanating from asynchronous computation, and new tools for global dependency analysis, graphical representation of flow, state information, etc., are required.

2. System issues. The research thus far has properly focused on the fundamental questions of can scientific computation be parallel processed and, if so, how well? Now it is time to also address fundamental system issues such as I/O, secondary storage, interactive graphics support, and network support.
3. Massively parallel systems. The architectures developed thus far typically have less than 100 processors. Now we need to measure the performance of computational kernels on systems with hundreds and thousands of processors.

Acknowledgement

This work was performed under the auspices of the US Department of Energy.

References

[1] S. Chen, in: High Speed Computation, ed. J.S. Kowalik, NATO ASI Series (Springer, Berlin, 1984) p. 59–67.
[2] S. Chen, J. Dongarra and C. Hsiung, Mathematics and Computer Science Division Technical Memorandum no. 24, Argonne National Laboratory (February 1984).
[3] K.G. Stevens, presented to the 1984 Oregon Conf. on Experiences in Applying Parallel Processing to Scientific Computation.
[4] M. Moore, R. Hirmoto and O. Lubeck, Parallel Computing, in press.
[5] J. Dongarra and R. Hiromoto, Parallel Computing, in press.
[6] D.H. Grit and J.R. McGraw, Lawrence Livermore National Laboratory preprint UCRL-88710 (May 1983).
[7] J.S. Kowalik, R.E. Lord and S.P. Kumar, High-Speed Computation, ed. J.S. Kowalik, NATO ASI Series (Springer, Berlin, 1984).
[8] B.R. Wienke and R.E. Hiromoto, Comput. Phys. Commun. 37 (1985) 363.
[9] T. Hoshino, T. Shirakawa, T. Kamimura, S. Sekiguchi, Y. Oyanagi and T. Kawai, Proc. 1983 Intern. Conf. on Parallel Processing, p. 95.
[10] J. Salmon, presented at the 1984 Oregon Conf. on Experiences in Applying Parallel Processors to Scientific Computation.
[11] P. Frederickson, R. Hiromoto, T. Jordan, B. Smith and T. Warnock, Parallel Computing, in press.
[12] J. Gurd and I. Watson, Proc. IFIP World Computer Congress (North-Holland, Amsterdam, 1982) p. 545.

Computer Physics Communications 37 (1985) 7–13
North-Holland, Amsterdam

HIGH SPEED VECTOR PROCESSORS IN JAPAN

Keiichiro UCHIDA and Mikio ITOH

Mainframe Division, Fujitsu Ltd., 1015, Kamiodanaka, Nakahara-Ku, Kawasaki 211, Japan

In 1982–1983, three Japanese companies announced new vector processors. These are Fujitsu's FACOM VP-100/VP-200, Hitachi's HITAC S-810 MODEL 10/MODEL 20, and NEC's SX-1/SX-2. Their maximum performance ranges from 250 to 1300 MFLOPS.

This paper describes the state-of-the-art technologies of vector processors in Japan, particularly used in the FACOM VP system as an example.

1. History and current status of high speed vector processors in Japan

The first vector processor, in Japan, the Fujitsu FACOM 230-75 Array Processing Unit was installed at the National Aerospace laboratory in 1977. In that era, there was little need for vector processors in Japan, and only two FACOM 230-75 Array Processing Units were shipped. However, a tremendous amout of experience was gained by the development of this vector processor, which has been fully utilized in the development of the FACOM VP system.

Hitachi and NEC have also developed Integrated Array Processors (IAPs), and attached them to their large scale general purpose computers.

In 1982 and 1983, Fujitsu, Hitachi and NEC announced vector processors as supercomputers. These vector processors are the FACOM VP-100 and VP-200 by Fujitsu, the HITAC S-810/10 and S-810/20 by Hitachi, and the NEC SX-1 and SX-2 by NEC.

These Japanese supercomputer manufacturers share some interesting characteristics.

Firstly, Fujitsu, Hitachi and NEC are also large semiconductor manufacturers in the world and can utilize the latest semiconductor technology.

Secondly, these Japanese supercomputer manufacturers add the vector architecture to their general purpose computer architecture. Especially, Fujitsu and Hitachi employ and IBM compatible instruction set for their general purpose computers.

Vector processors manufactured by Fujitsu and Hitachi have been shipped to major universities in Japan and are now fully operational. The FACOM VP-100's are in use at the Institute of Plasma Physics at the Nagoya University and the Kyoto University, and a HITAC S-810/20 is in use at the University of Tokyo. In addition, two CRAY 1-S systems are being used in Japan.

In Japan large scale numerical computations are primarily required in such areas as nuclear engineering, aerodynamics, meteorology and molecular science.

2. Supercomputers in Japan

Table 1 shows the hardware specifications of supercomputers manufactured by Hitachi, Fujitsu, NEC and Cray Research. It is to be noted that Japanese supercomputers have large scale main storage.

Table 2 shows the performances of Livermore Kernel on the FACOM VP-200 system, HITAC S-810/20 and CRAY X-MP (single CPU) as on January 1984. The performance of the FACOM VP-200 was measured by Fujitsu at its Numazu works. Table 3 summarizes the technologies used in the FACOM VP system. The performances of

0010-4655/85/$03.30

Table 1
Comparison of hardware specifications

	FACOM VP		HITAC S-810		NEC		CRAY X-MP
	100	200	/10	/20	SX-1	SX-2	
Announcement	1982	3Q	1982	3Q	1983	2Q	1983 2Q
First Customer service	1983	4Q	1983	4Q	not yet		1983 4Q
Machine cycle (ns)	7.5		15		7	6	9.5
Peak performance (MFLOPS)	250	500	315	630	570	1300	400
Vector registers (Kbyte)	32	64	32	64	40	80	8
Mask register	16 word ×256	32 word ×256	256 word×8		128 word ×8	256 word ×8	64 word ×8
MSU maximum capacity (Mbyte)	128	256	128	256	256		32
MSU maxiumum interleave	128 WAY	256 WAY	NA		512 WAY		32 WAY
MSU RAM chip	64 Kbyte MOS SRAM		16 Kbyte MOS SRAM		64 Kbyte MOS SRAM		4 Kbyte BIP
I/O maximum transfer rate (Mbyte/s)	96		96		50		224
Number of channels	16/32		8/16 /24/32		32		16

the HITAC S-810/20 and CRAY X-MP are taken from ref. [6]. The NEC SX system is still in the development stage and its performance is not available.

Table 2
Comparison of performance of Livermore kernels (unit = MFLOPS)

Kernel	FACOM **) VP-200	HITAC *) S-810/20	CRAY X *) CFT 1984
1	331.4	228.0	153.0
2	180.4	239.4	76.6
3	338.2	211.9	95.8
4	88.1	59.2	41.1
5	10.0	5.4	8.7
6	9.5	4.6	8.0
7	331.0	232.7	167.9
8	90.4	48.8	95.7
9	260.8	207.6	163.0
10	85.9	49.0	59.9
11	4.8	9.8	3.1
12	115.3	83.0	76.5
13	6.2	4.2	4.8
14	13.8	8.5	6.9
Average	133.3	100.2	68.6

*) Ref. [6]; **) measured at Numazu works.

3. Design philosophy of the FACOM VP system

Based on the extensive analysis of more than 1000 FORTRAN programs in scientific and engineering applications, the following primary objectives have been identified during the design phase of the FACOM VP system:

- To realize a system with high performance for users' application programs rather than just the peak performance.
- To realize a system which is easy to use for

Table 3
Technology used in the FACOM VP system

Logic LSI	ECL LSI	400 gate/chip	350 ps delay
	ECL LSI	1300 gate/chip	350 ps delay
RAM	bipolar	4 Kbit/module	5.5 ns access
	MOS static	64 Kbit/chip	55 ns access
PCB	MCC	121 LSI/MCC	31×31 cm^2 14 layers
	mem.card	1 Mbyte/PCB	24×38 cm^2 6 layers
Unit	MCC stack	13 MCC/stack	41 MCC/VPU

ordinary programmers in developing application programs.

In the FACOM VP system, these two objectives are achieved through the implementation of the following items in both hardware and software.

For increasing effective performance

- Vectorization of the DO loops including IF statements.
- Efficient vector edit instructions.
- Flexible definition of vectors.
- Dynamically reconfigurable vector registers.
- High speed scalar unit.
- High data transfer rate in vector load/store operations.
- Concurrent operations.

For ease of use

- Optimization control lines (OCL).
- Interactive vectorizer.

4. The FACOM VP system hardware

4.1. System configuration

Two models are available in the FACOM VP system: the VP-200 and the VP-100. Fig. 1 is a block diagram of the FACOM VP system.

The FACOM VP system consists of a vector processing unit (VPU), a main storage unit (MSU), and a channel processing unit (CHP). The VPU consists of a vector unit (VU) which processes vector data and a scalar unit (SU) which processes scalar data.

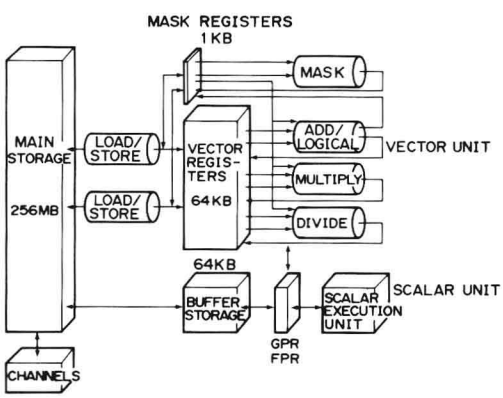

Fig. 1. The FACOM VP system block diagram.

4.2. Scalar unit (SU)

The SU has 64 Kbyte of high speed buffer storage and scalar registers such as 16 general-purpose registers, 8 floating-point registers. The scalar unit fetches and decodes all the instructions, and vector instructions are issued to the vector units. There is a direct path between the scalar unit and the vector unit, so that general purpose registers and floating point registers can be referenced or updated from the vector unit.

4.3. Vector unit (VU)

The VU mainly consists of vector registers (VR's), mask registers (MR's), two load/store pipelines, a mask operation pipeline, an add/logical pipeline, a multiply pipeline and a divide pipeline.

In the VP-200 system, all the arithmetic and mask operation pipelines operate with a 7.5 ns machine cycle, and each pipeline can process 2 elements per cycle. The divide pipeline has 1/7 of the throughput of the add or the multiply pipeline. Two load/store pipelines are provided to transfer data between VR's and the MSU. In the case of the FACOM VP-100 system, the throughput of each pipeline is half of that for the FACOM VP-200 system.

4.4. Vector registers (VR's) and mask registers (MR's)

The VP-200 system has 64 Kbyte VR's with the basic configuration of 8-byte word $\times$ 32 elements $\times$ 256 registers. (In the VP-100 system the basic configuration is 8-byte word $\times$ 16 elements $\times$ 256 registers). The vector registers are built with bipolar RAM's with 5.5 ns access time.

The VP-200 system has 1 Kbyte MR's to store the logical values ("TRUE"/"FALSE"), with the basic configuration of 1-bit-word $\times$ 32 elements $\times$ 256 registers. The MR's are built with same bipolar RAM's as vector registers. The dynamic reconfiguration feature of these registers is described in section 5.4.