SYNCHRONIZATION AND PIPELINE OVERHEAD MEASUREMENTS ON THE FPS-5000 MIMD COMPUTER

by

I. J. Curington and R. W. Hockney
Floating Point Systems UK, and Reading University UK

16 August 1985

Submitted to PARALLEL COMPUTING 85
North-Holland, 1985.

I. J. Curington
Floating Point Systems
Apex House
London Road
Bracknell, Berkshire RG12 2TE
England

R. W. Hockney
Department of Computer Science
Reading University
Whiteknights Park
Reading, Berkshire RG6 2AX
England

SYNCHRONIZATION AND PIPELINE OVERHEAD MEASUREMENTS ON THE FPS-5000 MIMD COMPUTER

I. J. Curington and R. W. Hockney
Floating Point Systems UK, and Reading University UK

ABSTRACT

A method of characterizing performance of the MIMD architecture of the FPS-5000 is presented, using the three parameter r_∞, $n_{1/2}$, and $s_{1/2}$ method, with further extensions to the $f_{1/2}$ parameter. This performance model examines the relationships of theoretical vector performance, overheads for filling pipelines, overheads for the start and synchronization of multiple co-processors, and the effects of memory bandwidth on performance. The penalty for MIMD computation can be measured by the cost of synchronization of multiple instruction streams in terms of lost work. The parameter $s_{1/2}$ is used to measure synchronization, while the figure $n_{1/2}$ is a measure of the initial delay required to start a vector operation. The parameter r_∞ is the measure of asymptotic performance, and the parameter $f_{1/2}$ is the measure of floating point operations per memory reference to achieve half the asymptotic performance. Results are presented with favorable comparisons to similar measurements of other supercomputer architectures.

INTRODUCTION

Although the scientific computer industry has used MFLOPS (millions of floating point operations per second) as a single performance parameter for years, it has long been shown inadequate to measure performance on a vector or parallel computer architecture [1,2,4], due to ignoring vector startup overhead. Although manufacturers still do not publish two or three parameter descriptions, these figures have been measured on a number of different supercomputers [4,5,6]. The movement in the computer industry to exploit parallel architectures for higher performance has left a gap in our ablility to characterize to performance by MFLOPS alone.

A more accurate model for characterizing performance is presented here, using two parameters to measure asymptotic performance and vector startup costs, and an additional two parameters to describe multiple processor synchronization costs and memory bandwidth efficiency. The list below summarizes the terms used in the FPS-5000 performance analysis.

SUMMARY OF TERMS:

r_∞ is the asymptotic performance in MFLOPS, as vector length goes to infinity.

$n_{1/2}$ is the startup delay for a vector operation, measured in units of floating point operation lost during the startup time.

$s_{1/2}$ is used to measure synchronization. A measure of the number of floating-operations required to achieve 50%

of r_∞ on an MIMD architecture.

f is the number of floating-point operations performed per main memory reference.

$r_\infty(f)$ is the asymptotic performance in MFLOPS for a given f.

$\hat{r}_\infty$ is the asymptotic performance in MFLOPS as f goes to infinity.

ARCHITECTURE

The FPS-5000 MIMD architecture is composed of a central Control Processor (CP), several XP32 co-processors, a GPIOP I/O processor, and a large System Common Memory (SCM) (See Figure 1). The model under test was an FPS-5230A and a VAX 11/780 computer system was used for initiation for the test programs, and accumulation of timing results.

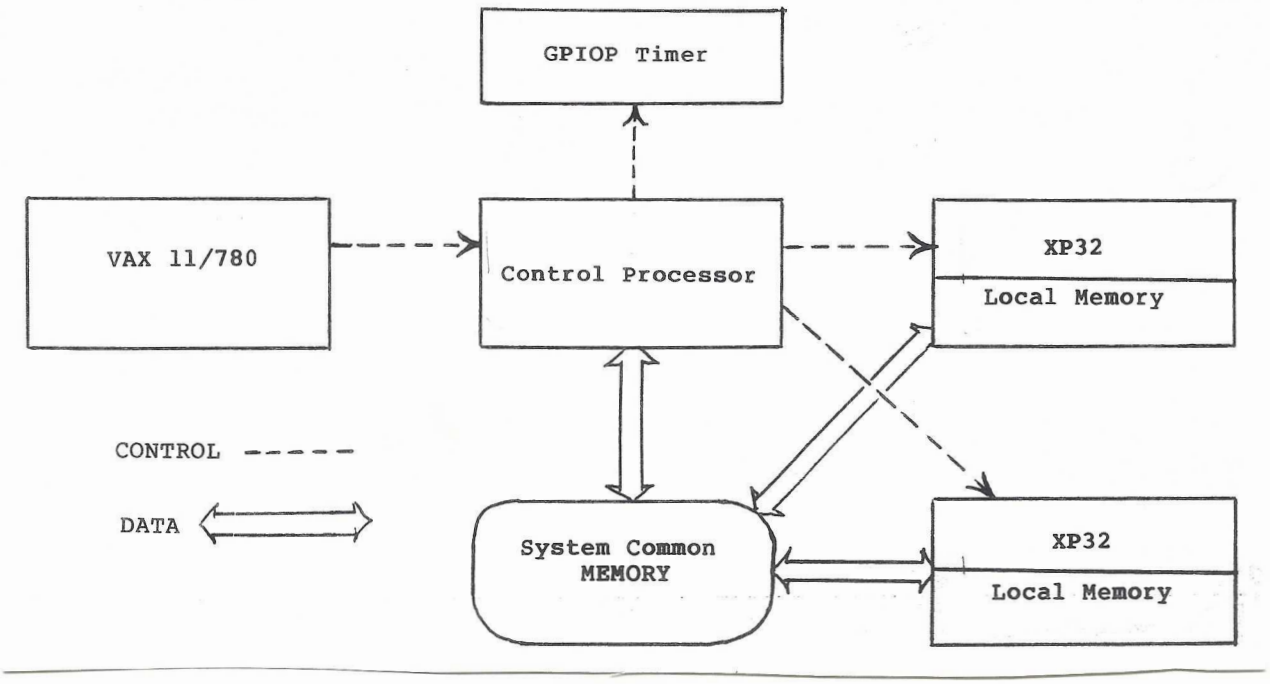

Figure 1. FPS-5000 MIMD Architecture Diagram.

The Control Processor is an SIMD architecture with a separate program storage memory, a floating point adder and a floating point multiplier. Both arithmetic elements produce results at 6 MHz. The Control Processor directly executes floating point operations, as well as providing control to the other parts of the FPS-5000.

The XP32 co-processor is also an SIMD architecture with a separate program storage memory, two floating point adders and one floating point multiplier. All three of these produce results at the instruction clock rate of 6 MHz. Unlike the CP, the XP32 contains a high speed local memory, and a separate controller for movement of data between the local and the System Common Memory (SCM). One FPS-5000 chassis may contain up to three XP32s, however the tests presented in this paper used an FPS-5320A configuration which

contains two XP32s.

The FPS-5000 system architecture contains an SISD class programmable I/O processor, the GPIOP, with a separate instruction stream from the other system elements. For purposes of performance analysis, the GPIOP is used to count machine cycles, recording elapsed time to an accuracy of one microsecond. The timing results are collected by the Control Processor, then passed to the VAX for final reporting.

The primary data storage area of the FPS-5000, directly accessible by the VAX, Control Processor, and XP32s is the System Common Memory. Vector arithmetic operations in the Control Processor, as those used in this analysis, operate directly upon this memory. As the XP32 operates on local data storage, data is moved from the SCM to the XP32 and back for processing.

PARAMETERIZATION

An SIMD computer uses a single instruction stream to control multiple arithmetic units, all operating concurently on distinct data elements. Both the Control Processor (C.P.) and the XP32 co-processor in the FPS-5000 have SIMD architectures, in the form of pipelined floating point arithmetic elements. Such an architecture can be characterized by the two parameter model of $(r_\infty, n_{1/2})$ [4]. In this case, the time t for a vector operation of length n is approximated by

$$t = r_\infty^{-1}(n + n_{1/2}) .$$

In an MIMD computer, such as the FPS-5000, each individual processor may be characterized by such a two parameter model, but an additional parameter is needed to describe synchronization and control overhead in using multiple processors for a particular operation. The parameter $s_{1/2}$ characterizes the synchronization overhead much like the vector startup overhead, but for the case where the vector operation work is shared by multiple processors. The critical path through an MIMD program can be considered as a sequence of work segments, between which program synchronization must occur [6]. Within a work segment, each processor works independently on a particular task. The synchronization includes the Control Processor making requests to each of the XP32 co-processors and to itself, then recognizing that all processors have completed their tasks, or becoming synchronized again. The time taken to perform this synchronization with the Control Processor may be measured independently of the asymptotic performance of the the individual processors, or the combined asymptotic performance of the complete system. The estimate of the time t for a work segment consisting of s floating point operations is

$$t = r_\infty^{-1}(s + s_{1/2}) .$$

These estimates usually depend on the type of vector operation being performed, and what architectural elements hold the data. Over the broad range of algorithms for which a characterization is desired, an additional parameter is necessary, one which will characterize the proportion of arithmetic work to data movement in and out of the main memory. The XP32 co-processor arithmetic

elements operate on data stored in local high speed memory, while a separate controller moves data (usually in blocks) to or from the main System Common Memory (SCM). The number of floating point operations on each data element transfered forms the ratio f, which will directly effect any measurement of r_{oo} and $s_{1/2}$. The value of f for which half of the asymptotic performance is reached is termed $f_{1/2}$.

SOFTWARE

The tests were conducted using the simplest of software interfaces available on the array processor architecture, with no exploitation of low-level languages or non-standard communication mechanisms [3]. The tests were written in FORTRAN-77 using the CPFORTRAN cross compiler for the Control Processor. Vector library calls and XP32 operation calls are made directly from FORTRAN, so the $n_{1/2}$ vector startup measurements include parameter passing and control structures of the FORTRAN programming environment.

MEASUREMENTS

Timing measurements were made on the FPS-5320A for various configurations and two types of vector operators. All timing results were obtained using the GPIOP I/O Processor, running a program to measure elapsed clock cycles, with timing accurate to one microsecond. Two types of arithmetic operations were tested: vector multiply, and vector scalar multiply and scalar add. The vector multiply (VMUL and ZVMUL) operators are dyadic, forming the element-by-element product of two input vectors, as $C_i=A_i*B_i$, for i=1 to n. The performance of this operator is limited by memory references, since it requires two reads and one write for every multiply. In addition, the floating-point adders are idle during this process, so the asymptotic rate r_{oo} is well below maximum.

The vector scalar multiply and scalar add (VSMSA and ZVSASM) operator has the same number of floating operations as memory references, and is able to use the multiplier and adder elements in parallel, performing $D_i=A_i*B + C$, for i=1 to n.

VMUL/ZVMUL from CPFORTRAN:

Configuration	r_{oo} (MFLOPS)	$n_{1/2}$ or $s_{1/2}$
CP Only	1.45	13.7
1 - XP32	4.0	470.
2 - XP32s	8.0	1320.
2 - XP32s + CP	9.2	1545.
1 - CRAY-1 Only	22.	18.
2 - CPU CRAY X-MP22	130.	5700.

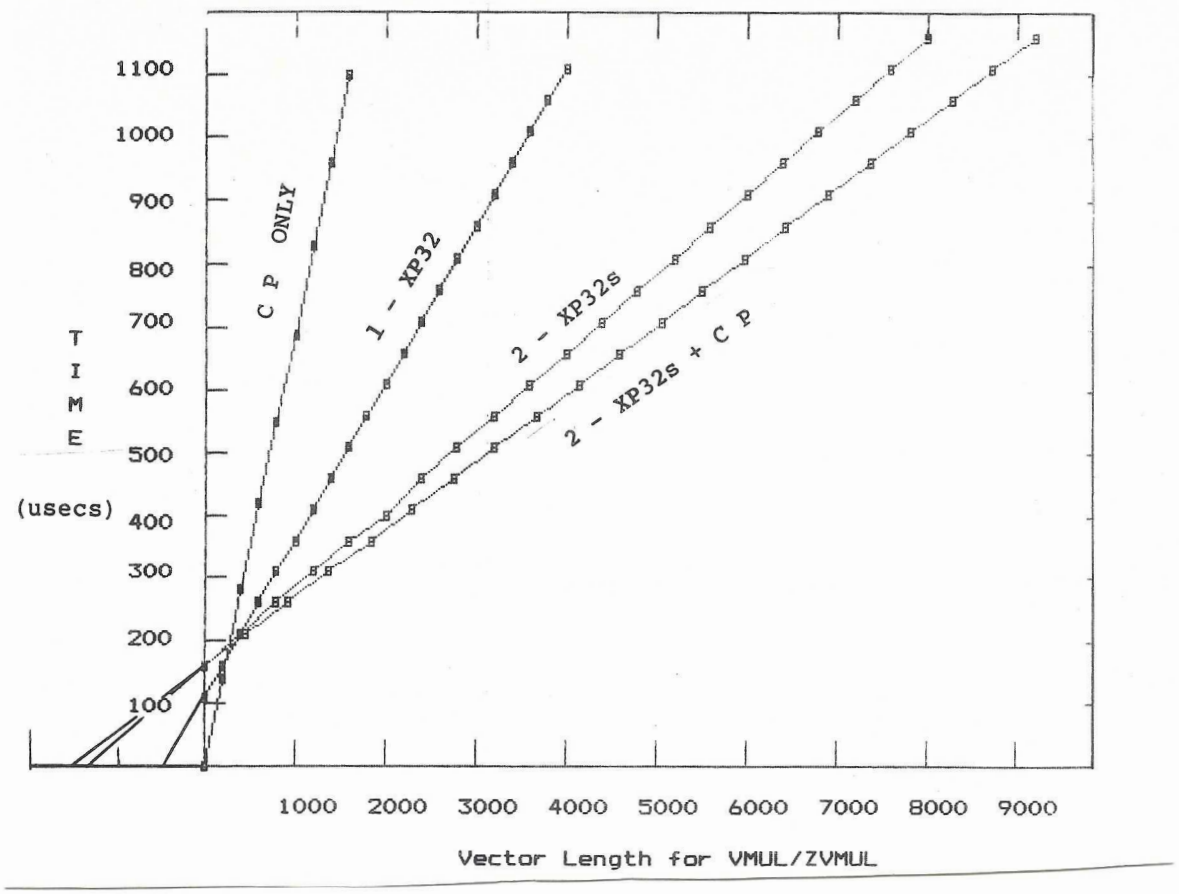

Figure 2. Vector Multiply Performance Graph.

The linear relations shown in figure 2 indicate the vector multiply performance and synchronization overheads for various configurations. The CRAY timing has been performed using the same two parameter model, with the dual CPU synchronization using the TASK method [7]. The CP compares favorably with the CRAY-1 with a lower vector startup overhead, while the 2 XP32 $s_{1/2}$ is much lower than the CRAY X-MP22, showing the effects of much tighter hardware and software coupling between the individual CPUs in the FPS-5000.

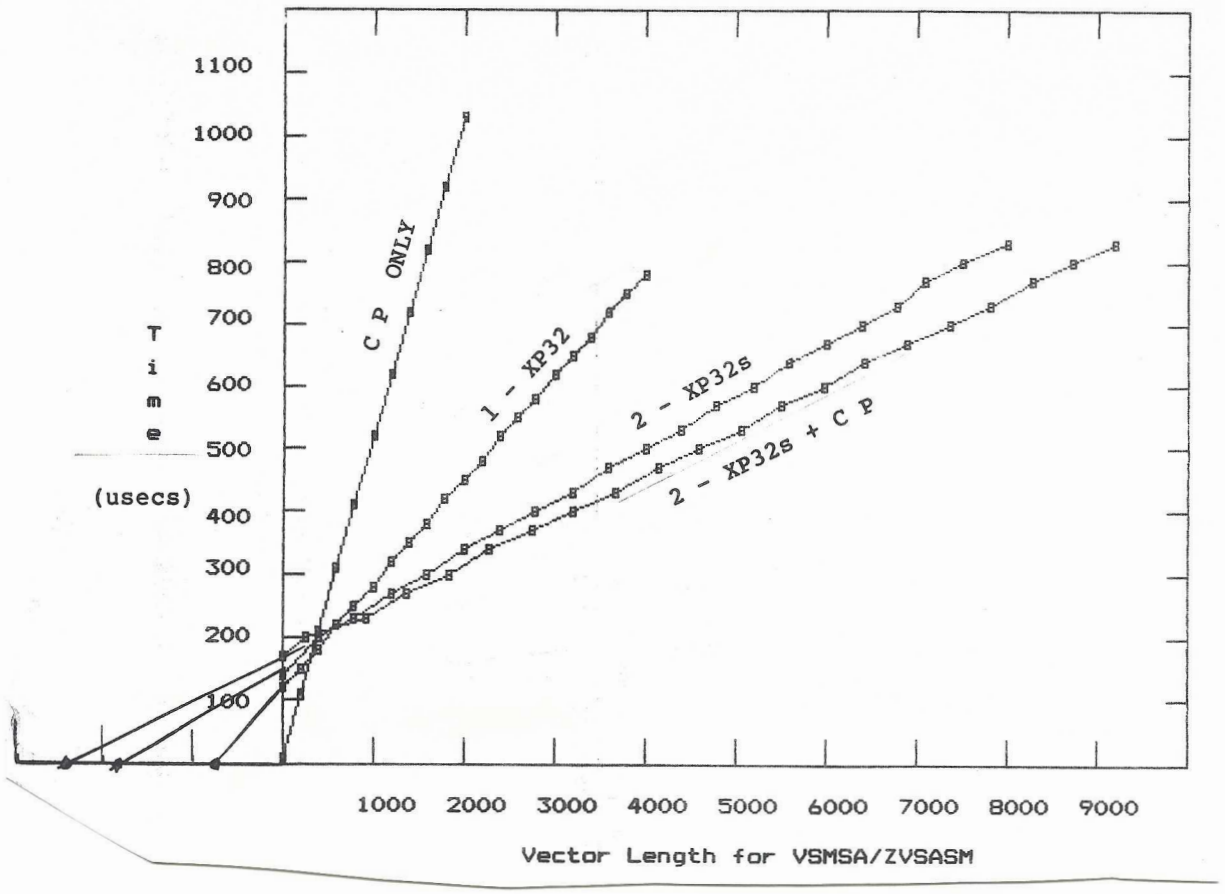

Figure 3. Vector Scalar Multiply Scalar Add Performance Graph.

VSMSA/ZVSASM from CPFORTRAN:

Configuration	r_∞ (MFLOPS)	$n_{1/2}$ or $s_{1/2}$
CP Only	3.9	39.6
1 - XP32	12.0	1490.
2 - XP32s	24.0	4200.
2 - XP32s + CP	27.7	4820.

The linear relations shown in figure 3 indicate the vector scalar multiply and scalar add performance and synchronization overheads for various configurations. The $s_{1/2}$ measurements for this operator are higher than for the vector multiply measurements due to equivalent startup times, but higher asymptotic performance.

ZVSASM, XPDMAR on 1 - XP32 from CPFORTRAN:

Configuration	$\hat{r}_\infty$ (MFLOPS)	$f_{1/2}$
Sequential I/O & ZVSASM	12.5	4.2
Overlapped I/O & ZVSASM	12.6	2.2

NOTE: Results are deliberately rounded to only three significant figures.

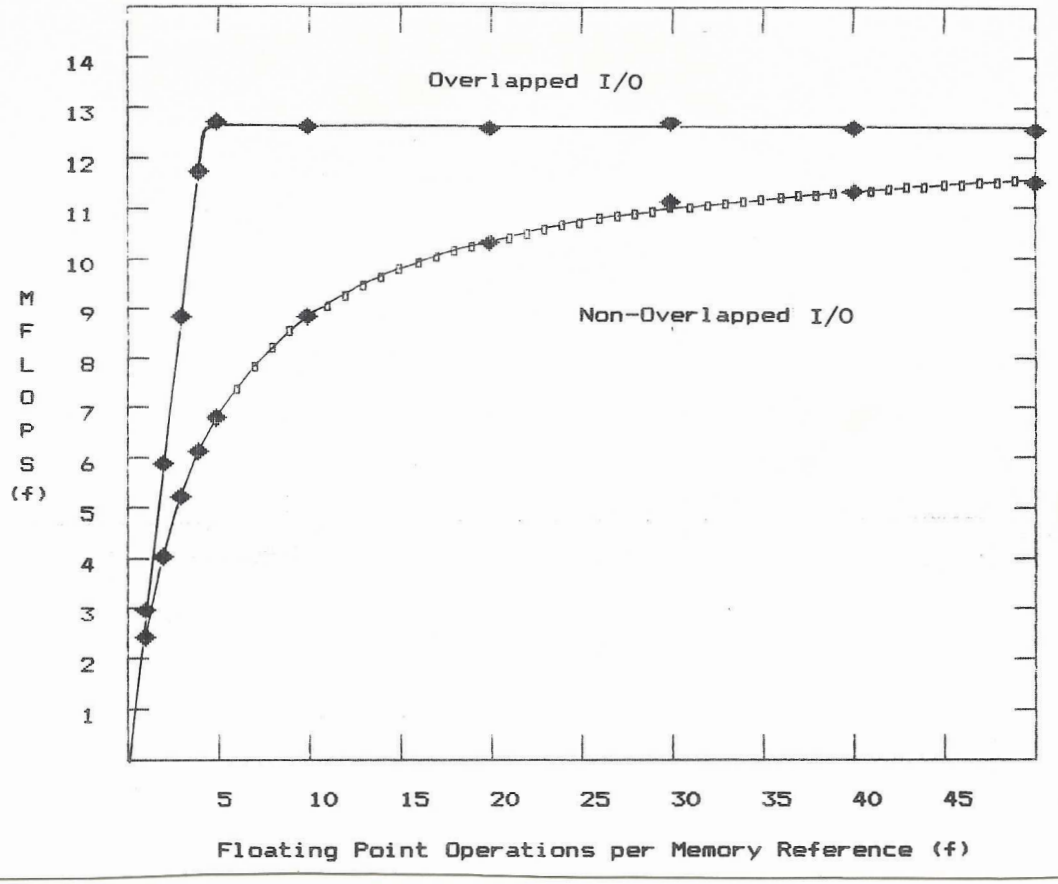

Figure 4. XP32 to System Common Memory Bandwidth Graph.

Figure 4 shows the relationship between memory bandwith and asymptotic performance for the ZVSASM operator for one XP32 co-processor. The overlapped I/O case uses the controller on the XP32 to move data in and out during the ZVSASM operator execution, using double buffering, while the sequential measurements are for the case where this is not possible. For the sequential test, data is moved into the into the XP32, the operation is performed, and the data is returned to S.C.M. with no overlap. The number of I/O calls compared with the number of ZVSASM calls determines the value f. As the lower curve of Figure 4 shows, the measured data fits the predicted curve very closely, where

$$r_{\infty}(f) = r_{\infty}^{-1}(1+(f_{1/2}/f)) \ .$$

The overlapped I/O and computation test exploited the architectural features of the XP32 in which the local memory has sufficient bandwidth to allow both I/O and computation concurrently. The $f_{1/2}$ parameter is half the non-overlapped case, since the I/O is concurrent with, not added to computation time. Also, the form of the graph is different from the non-overlapped model with two distict linear curves, joined at the point where computation equals I/O time. The first part of the graph is linearly increasing performance as more computations are performed per data transfer. The $f_{1/2}$ has its value at the point where this linear ramp equals 50% of the peak value. When the computation time exceeds I/O time the asymptotic performance is purely a function of the CPU, as transfer time is hidden.

CONCLUSIONS

A performance model and measurments have been reported for the asymptotic performance, vector startup overhead, MIMD synchronization, and memory bandwidth in the FPS-5000 MIMD array processor. The XP32 co-processor is approximately three times faster than the Control Processor on the vector operations measured, and two XP32s double the performance. The startup and synchronization costs of using the XP32 are many times higher using CPFORTRAN calls than in-line calls using the Control Processor. The synchronization mechanisms are of better efficiency to the dual CPU CRAY X-MP, while maintaining a simple FORTRAN interface.

REFERENCES

[1] Calahan, D.A., "Algorithmic and Architectural Issues Related to Vector Processors" Proc. Intl. Symp. Large Eng. Syst., New York, Pergamon, 1977.

[2] Calahan, D.A., and Ames, W.G., "Vector Processors: Models and Applications" IEEE Transactions on Circuits and Systems, CAS-26, pp. 715-726, 1979.

[3] Curington, I.J., "Performance Estimation Methods for XP32 MAXL" Proceedings of FUSE-84, Windermere, England, Floating Point Systems Pubs., March 1984.

[4] Hockney, R.W., and Jesshope, C.R., "Parallel Computers - Architecture, Programming and Algorithms" Bristol,and Boston Adam Hilger Publisher, 1981.

[5] Hockney, R.W. and Snelling, D.F. "Characterizing MIMD Computers: e.q. the Denelcor HEP" Parallel Computing 83, M. Feilmeir, G.R. Joubert and U. Schendel (eds.), Amsterdam, Elsevier Science Publ., North-Holland, pp. 521-526, 1984.

[6] Hockney, R.W., "Novel Computer Architectures" Proceedings of Conference on Vector and Parallel Processors for Scientific Computation, IBM Rome, May 1985, to be published by Accademia Nazionale dei Lincei, Rome, 1986.

[7] Hockney, R.W., " $(r_\infty, n_{1/2}, s_{1/2})$ Measurements on the 2-CPU CRAY X-MP" Parallel Computing 2, 1-14, 1985