NASA Technical Memorandum 87819

Report From the MPP Working Group to the NASA Associate Administrator for Space Science and Applications

James R. Fischer
Goddard Space Flight Center
Greenbelt, Maryland

Chester Grosch
Institute for Computer Applications in Science and Engineering
Langley Research Center
Hampton, Virginia

Michael McAnulty
University of Alabama
Birmingham, Alabama

John O'Donnell
Indiana University
Bloomington, Indiana

Owen Storey
Stanford University
Stanford, California

NASA
National Aeronautics
and Space Administration

Scientific and Technical Information Division

1987

Acknowledgments

The significant contributions of many individuals and organizations made this report possible. In particular--

The **MPP Working Group members** and their technical staffs, who committed their own valuable time during the past two years to explore the potential of the MPP in unique areas of research.

The four group leaders: **Dr. Chester Grosch, Dr. Michael McAnulty, Dr. John O'Donnell,** and **Dr. Owen Storey,** who integrated the reports from the four discipline groups.

Science Applications Research (SAR) and **Goddard personnel** who provided systems, applications, and user support for the MPP.

--Jim Fischer

TABLE OF CONTENTS

EXECUTIVE SUMMARY

History of the MPP Working Group

Practical scientific applications using massively parallel computer hardware first appeared on the research scene during the 1980's. Their development has been motivated by the need for computing power orders of magnitude beyond that otherwise available today for tasks such as numerical simulations of complex physical and biological processes, generation of interactive visual displays, satellite image analysis, and knowledge-based systems. Representative of the first generation of this new class of computer hardware is the Massively Parallel Processor (MPP)[1], located at NASA's Goddard Space Flight Center in Greenbelt, Maryland. Even though the MPP was constructed by NASA specifically to process high pixel resolution satellite image data, such as Landsat or Space Telescope images, its design also provided for reasonable high-speed scientific computation. During this same period, computer scientists were making substantial theoretical progress in the development of efficient algorithms for grid-connected array processors. Believing in the untested potential of parallel architectures to meet a wide variety of computational problems, NASA took the risky step of making the MPP readily available to a diverse national community of scientific investigators who were charged with discovering what its limits were.

Through a national solicitation[2] initiated in 1984 by NASA's Office of Space Science and Applications (OSSA), a pioneering team of 40 scientists[3] was provided the opportunity to test and implement their computational algorithms on the MPP beginning in the fall of 1985. By the end of a year, enough interesting results were acquired from these projects to warrant convening the **First Symposium on the Frontiers of Massively Parallel Scientific Computation**[4] held September 24-25, 1986, at the Goddard Space Flight Center. The research endeavors of the MPP investigators span a broad variety of applications including Earth science, physical science, signal and image processing, computer science, and graphics. The performance of many of these applications on the MPP was in the supercomputer range to well beyond any existing capabilities.

A majority of the applications implemented on the MPP by the investigators have been numerical simulations, and their diversity has drawn national attention to the general-purpose capabilities of the MPP's single-instruction-stream multiple-data-stream (SIMD) architecture.

Role of Parallel Scientific Computation

Since the MPP investigators began their work, a wide variety of **coarse grain parallel** processors have become available in the commercial marketplace. These processors have established themselves as having scientific value and are now beginning to permeate all fields of scientific computation.

In this same timeframe, commercial **massively parallel** computer systems also became available and production models were delivered to more than twelve scientific and military sites in the United States. We are now witnessing the development of massively parallel computer systems, from specialized processors for DoD to those having broad applications in the university and commercial arenas. Numerous applications are being developed on these commercial machines in areas ranging from fluid simulation and image processing to artificial intelligence. They are producing a wealth of the "new intuition" necessary for their effective use, and their emergence as inexpensive alternatives to other supercomputers, combined with their ability to be upgraded fairly easily, present compelling scientific and marketplace arguments for their accelerated development. Today, the need is to build up first hand experience using these systems and to compare and document the computational techniques that emerge.

Findings

The MPP Working Group made the following observations after 12 months of MPP use:

1. The Working Group provided a broad range of applications tests for the MPP.

2. The MPP architecture is suitable for a wide range of applications–much broader than initially anticipated.

3. For certain classes of problems, the MPP's SIMD architecture is easier to program than a VAX or CYBER 205. These classes include:
   - lattice simulation
   - neural network simulation
   - image problems
   - grid-based fluid dynamic simulation
   - systolic VLSI simulation–those where distances are short.

4. "Data Structure" parallel algorithms[5], a style of parallel implementation distinct from "Control Structure" parallel algorithms, are appropriate for the MPP.

5. The bit serial processing element (PE) architecture of the MPP is good, and can be very important for certain applications. One application achieved a speed-up of 16,000 times over a VAX. A bit serial algorithm was used. This kind of algorithm is most efficient for the MPP and inefficient for machines based on conventional architecture.

6. Surprisingly, finite element analysis, which intuitively was not expected to map well to the MPP, runs as fast on the MPP as on other supercomputers such as the CYBER 205 and CRAY-1.

7. The MPP is **still** a national resource–it should be put onto the National Science Foundation (NSF) networks. The MPP is the only massively parallel machine available over a network.

8. The MPP is being used as a tool to design other architectures.

9. NASA provided people with computer smarts to be the support group to users. This situation will persist. The support provided worked well. The difference to-date between the MPP and machines such as the CYBER 205, is that the MPP support people had to get involved in each investigator's problem.

10. An MPP user must be dedicated to get results. The machine still requires a high level of user sophistication and computing experience.

11. The Working Group stimulated interdisciplinary interactions. Many of its members would not otherwise have had access to a parallel machine.

12. Because of the magnitude of the problems being run on the MPP, visualization via image display is important for tracing the state of programs running in the machine and the flow of data. By looking at images, the user can conclude whether or not the processing is running correctly. It is an implicit diagnostic/debugger.

13. The MPP's Parallel Pascal compiler lacks optimization. Currently, it produces code that is far from optimum. By contrast, users of the British Distributed Array Processor (DAP) can gain only 10 percent performance increase by going to assembly language. Parallel Pascal overhead is estimated to be:

- a factor of 2 for floating point math. (C.Grosch–ocean dynamics).
- a factor of 4-5 for short integers. (H.Hastings–neural net model).
- a factor of 40 for logic. (F.Sullivan–Ising Spin simulation migrated to PEARL).

14. Some Working Group members found Parallel Pascal superior to FORTRAN because of certain features in the Pascal language. For an experimental research environment Parallel Pascal is a good compromise.

15. There is currently no library of professional, user-friendly science subroutines available on the MPP.

16. Users accessing the MPP from remote locations found this means of access viable. Some remote users used the portable MPP simulation environments[6,7] to develop programs on their own VAXes, then moved them to the MPP via network to run them.

17. The MPP's 1K bits of memory per processor were found to be limiting for numerous applications. Currently this memory limitation is the system's major bottleneck on performance; enlarging it is the most obvious way to improve the performance of many applications. Some MPP users are doing large 2-dimensional dynamics simulations, but they cannot go to 3-dimensional because of the small size of this processor memory.

18. The host VAX's slow I/O is another major throughput bottleneck, especially for applications using large data sets. The MPP is not the I/O bottleneck. The host VAX's response time deteriorates during prime shift, for example, a 90-second job at midnight could be an hour job at worst-case prime time. MPP data I/O is still difficult to program.

19. The MPP provides an environment for educating scientists and their students.

- **Scientist education**–the Working Group activity is unique because of its breadth, taking in the non-space scientist community.

- **Academic education**–access to the MPP has been a useful educational tool. Numerous papers and Masters and Ph.D theses in the area of parallel processing are being generated. See *Appendix D* for lists of students involved, papers, theses, and grants.

Technical Recommendations (in priority order)

1. Provide a functionally complete handbook for users, describing in one place, all aspects of MPP use.

2. Envelop Parallel Pascal within an accepted programming environment including a symbolic debugger.

3. Increase the size of the array memory and main control unit memory, as much as possible.

4. Implement all standard features of Parallel Pascal or some similar functionally complete Pascal derivative.

5. Improve I/O between the staging memory and both the host and image displays.

6. Provide code optimization for the Parallel Pascal compiler.

7. Extend Parallel Pascal to support asynchronous transfer of data from host memory to both the main control unit memory and the staging memory. This will allow computation to take place simultaneously with data transfer.

8. Make the MPP simulation environment identical to the host environment, and portable to both VMS and UNIX operating systems.

9. Create a library of professional, user-friendly science subroutines and corresponding documentation. There is a common need among Working Group members for subroutines that perform generic algorithms, such as Fast Fourier Transforms or random number generation. The DAP scientific subroutine library[8] could be used as a model for such a library for the MPP.

10. Implement a time-sharing user environment on the host computer. This will allow the MPP to be used interactively by multiple users.

11. Provide transparent software for manipulating data arrays with sizes that are multiples of 128 x 128.

12. Implement virtual array unit and staging memory software.

Strategic Recommendations

1. Issue a new announcement to expand the Working Group.

2. Solicit collaboration with other agencies to broaden technology applications.

3. Institutionalize support for operations of the MPP at current levels, thereby making the MPP available to an expanded Working Group to include Research and Technology Objectives Proposals (RTOP's) or NASA grant proposers who are prepared to explore the potential of this new architecture for future applications to their science.

4. Initiate a study of the role of future parallel processors for Space Station, Earth Observing System (EOS) and the Great Observatories era.

Contributors

The following MPP Working Group principal investigators and technical staff members contributed to the preparation of this report:

John Barnden	Indiana University
Michael Bielefeld	Computer Sciences Corporation
Paul Chiang	Washington State University
Seog Yeon Cho	University of Iowa
Edward Davis	North Carolina State University
Howard Demuth	University of Idaho
James Earl	University of Maryland
Eugene Greenstadt	TRW
Chester Grosch	NASA–Langley Research Center; Old Dominion University
Robert Gurney	NASA–Goddard Space Flight Center
Martin Hagan	Oklahoma State University
Susanne Hambrusch	Purdue University
Paul Harten	University of Cincinnati
Harold Hastings	Hofstra University
Sara Heap	NASA–Goddard Space Flight Center
Nathan Ida	University of Akron
James Koga	Southwest Research Institute
Donald Lindler	Advanced Computer Concepts
Michael McAnulty	University of Alabama–Birmingham
A.P. Mullhaupt	University of New Mexico
John O'Donnell	Indiana University
Martin Ozga	U.S. Department of Agriculture
John Reif	Duke University
Narayan Sahoo	State University of New York–Albany
Owen Storey	Stanford University
James Strong	NASA–Goddard Space Flight Center
Peter Suranyi	University of Cincinnati
James Tilton	NASA–Goddard Space Flight Center
Lloyd Treinish	NASA–Goddard Space Flight Center
Richard White	Space Telescope Science Institute
Lo Yin	NASA–Goddard Space Flight Center

Chapter 1 - The MPP Working Group

1.0 Introduction

NASA is interested in evaluating the performance and utility of the MPP for appropriate scientific applications to specific problems arising in space and Earth sciences. In addition, for purposes of obtaining a broader understanding of the performance of the MPP, NASA also considers investigations in the general physical and mathematical sciences. To achieve these goals, a Space Science and Applications Notice (AN) entitled **Computational Investigations Utilizing the Massively Parallel Processor (MPP)**[2], was signed on December 20, 1984, by Dr. Burton I. Edelson, NASA Associate Administrator for Space Science and Applications. It announced an ongoing opportunity to carry out computational investigations exploiting the unique characteristics of the MPP. This letter was distributed nationally to an address list of more than 2,000 scientists and institutions. Investigators who are accepted into this program are required to participate in a Working Group in order to assist NASA in assessing the usefulness and potential of the MPP for a wide variety of computational applications that stress different uses of the MPP's hardware and software.

NASA provides each investigator with free time on the MPP and its host VAX-11/780 computer, documentation of the hardware and software, and limited user assistance. Each member of the Working Group is expected to provide all expenses required for his/her use of the machine, including expenses for personnel required for programming and execution of software, salaries, and related expenses. Prospective investigators are also expected to obtain funding for their research activities through other program offices at NASA or from other agencies or institutions. Acceptance into this program does not carry with it any guarantee of research funding. Participation is limited to investigators at U.S. institutions, and proposals may be submitted at any time.

Each proposed investigation should present a well-defined scientific or engineering study that will make a distinct contribution to one or more of the following NASA objectives:

- assess the MPP's unique capabilities for carrying out scientific research,
- measure and document advantages of parallel processing techniques over conventional high-speed computers,
- enhance the MPP system of applications software for general user availability,
- evaluate the future needs for advanced MPP-type machines.

Proposals should demonstrate a computational need for the MPP's capabilities to carry out the scientific research. Available time on the machine is limited, and NASA accepts only those investigations that can be reasonably accommodated and accomplished using the current configuration.

1.1 Selection

Each proposal is reviewed and evaluated for scientific quality and technological merit by a panel of disciplinary peers with final selection of the investigations made by NASA Headquarters. The final decision on a proposal's acceptance takes into consideration the potential ability of the investigation to assess the MPP, the balance between different scientific disciplines, and the availability of MPP computing time.

On March 20, 1985, the Technical Advisory Committee (TAC) for this AN met to review the proposals that had been received as of close of business on March 19, and to prepare a recommendation to the Headquarters Selection Committee. The TAC consisted of:

Dr. Daniel Slotnick	University of Illinois	Chairman
Prof. Ashok Agrawalla	University of Maryland	
Dr. David Randall	Goddard, Code 610	
Dr. John Barker	Goddard, Code 620	
Mr. James Fischer	Goddard, Code 630	
Dr. Kenneth Iobst	Goddard, Code 630	
Dr. James Strong	Goddard, Code 630	
Dr. James Tilton	Goddard, Code 630	
Dr. Sheldon Green	Goddard, Code 640	
Dr. Richard Shine	Goddard, Code 680	
Dr. Robert Silverberg	Goddard, Code 690	Vice Chairman
Mrs. Ai Fang	NASA HQ, Code EI	Executive Secretary
Mr. Ken Wallgren	NASA HQ, Code RTC	

Dr. Allan Gottlieb (New York University) and Mr. Rudy Faiss (Goodyear Aerospace) were appointed to the TAC but were unable to attend.

Forty proposals had been received and all were reviewed. Most proposals were assigned in advance to at least three TAC members for reading. Assignments were made with the intention of matching proposal subject matter to TAC member background. In the review meeting, each proposal in turn was given five minutes for presentation by its readers and was then discussed and graded into one of four categories:

OUTSTANDING GOOD POOR JUDGMENT DEFERRED

When all proposals had been considered, the floor was opened to allow the grade given to any proposal to be changed. The final ratings of the 40 proposals received were:

15 - outstanding
17 - good
5 - poor
3 - judgment deferred (only abstract received)

On April 12, 1985, the Selection Committee met at NASA Headquarters to review the work of the TAC and select the initial membership of the Working Group. Committee members were:

Dr. Caldwell McCoy	NASA HQ, Code EI, Chairman
Dr. Milton Halem	GSFC, Code 630, Co-Chairman
Dr. John Theon	NASA HQ, Code EE
Dr. Dan Spizer	NASA HQ, Code EZ
Mr. Lee Holcomb	NASA HQ, Code R
Dr. John Lehman	NSF
Dr. Paul Schneck	Supercomputing Research Center

The Selection Committee:

- selected the 32 proposals rated outstanding or good by the TAC,
- changed the ranking of one proposal from poor to good and accepted it,
- reviewed and accepted two proposals that had been rated "judgment deferred" by the TAC (the proposals had subsequently been received).

A total of 35 Working Group projects were selected by the April 12, 1985, Selection Committee. These proposals roughly divide into four disciplines: Earth sciences, theoretical physics and astrophysics, signal and image processing, and computer science and graphics. Six times since then, as new proposals have been received, selection committees have been convened and have accepted 7 additional Working Group members. Two members have resigned. Currently, there are 40 Working Group projects[3] (see *Appendix B - Working Group Membership*).

1.2 Charge

Each member of the Working Group is expected to conduct a well-focused research effort that will provide insight into some aspect of the MPP system, and a final report covering the results of those investigations. Two-day workshops are conducted semiannually by NASA-Goddard. Workshops were held in August 1985, February 1986, September 1986, and February 1987. These workshops are designed to enhance the exchange of ideas and to discuss individual results. All Working Group members are expected to participate in workshop discussions and contribute to workshop reports. NASA provides a limited travel allowance for workshop participation.

In exchange for free use of the MPP and its associated computing resources, Working Group members are required to provide to NASA documented software subsequent to using it in their MPP investigations. If required, NASA will provide proprietary protection for data or algorithms for 1 year following the start of each investigation.

Chapter 2 - The MPP Program

The MPP is the product of an Office of Aeronautics and Space Technology (OAST) research and technology program designed to evaluate the application of a computer architecture containing thousands of processing elements, all operating concurrently, to the computational requirements of the sensor systems of the 1980's and 1990's. The MPP was delivered to NASA-Goddard in May 1983, by Goodyear Aerospace Corporation following 4 years of development. After delivery, Goddard installed a VAX-11/780 as the MPP's host and developed an MPP user environment connecting the MPP and the VAX. The unique software and hardware systems making up the current MPP user environment[9,10], including the MPP high-level languages and the operating system, are described below. For more detail, see *Appendix E - Technical Summary of the MPP* .

2.1 MPP User Environment

User Languages

The initial high level language implemented in 1983 was **Parallel Pascal**[11], which is an extended version of standard Pascal. Experience gained in the development and use of Parallel Pascal recommended[12] a modified language, **MPP Pascal**[13], which is currently in user field test. As part of one Working Group project, a highly interactive language, **MPP Parallel FORTH**[14], became operational in December 1986. The MPP is also programmable in two assembly languages[15,16].

Operating System

The MPP operating system provides support for running applications code and for interactive debugging[17]. It links the program running on the MPP with the program running on the VAX through a message passing system, and supports an extensive set of I/O service routines[18].

Policy: An MPP system software policy, established in February 1986, directed that all subsequent MPP system software would be upward compatible, and that the existing system software would be supported by NASA for a period of at least 2 years.

MPP User Consultants

When the Working Group began in October 1986, Goddard established a goal of making every effort to help the group members succeed in what they had proposed to do on the MPP. Instead of just waiting for questions, Goddard wanted to make available the MPP experience that had been developed, and to be involved in the Working Group's algorithm design process from the beginning. To offer support in this way, Goddard chose eight staff members skilled in MPP applications development, and assigned each of them to provide consultative help to roughly four Working Group members. In many cases, the choice reflected the special interests of the staff member.

The scope of consultation encompassed all aspects of algorithm and program design, including specific coding. The consultants were also charged with identifying low-level routines that would be of general use to other MPP users. Since the consultants developed indepth understanding of the applications with which they were working, requests for debugging advice were often directed to them. The Working Group members were encouraged to visit Goddard to work directly with their consultant and the MPP staff. A consultant could also visit the Working Group member.

Subroutine Libraries

A library is being established to collect computational subroutines as they are developed. NASA is not able to validate those subroutines that come from users, because of the effort required.

MPP Simulation Environments

Two MPP simulation environments have been developed and distributed to user sites remote from Goddard. The **MPP Simulator**[6] supports the development, testing, and refinement of Parallel Pascal, MPP Pascal, or assembly language applications programs on any VAX operating under VMS. The **Parallel Pascal Translator**[7] allows the development of Parallel Pascal programs on most computers that have a Pascal compiler.

Documentation

	Date first available
Overviews	
General Description of the MPP	4/83
Computing on the MPP	2/86
User Manuals	
MPP User's Guide	1/86
MPP Primer	7/85
CAD User's Manual	4/83
MPP Simulator User's Manual	12/85
Library Manuals	
MPP MCL Macros and Subroutines	6/86
MPP Pascal Callable Procedure Library	2/86
Language Manuals	
Parallel Pascal Language Reference Manual	1/85
MPP Pascal Language Reference Manual	9/86
MPP Main Control Language Manual-MCL	4/83
MPP PE Array Language Manual-PEARL	4/83
System Software Internals Documents	N/A
Hardware, Architecture and Maintenance	4/83

2.2 MPP Physical Environment

Access to the MPP

The MPP is in the Image Analysis Facility of the Space Data and Computing Division at Goddard. It is physically located in Building 28 and is accessed either on-site or from remote sites by means of:

SPAN	9600 baud	since 10/85
ARPANET	9600 baud	since 2/86
TELENET	1200 baud	since 2/86
Dial-In	2400 baud	upgraded from 1200 baud 10/86

The MPP is normally available for use from 0900 Monday through 2400 Saturday and from 1000 through 1800 Sunday.

On-site Facilities for MPP Users

Work Areas: Prior to August 1986, work areas for visiting Working Group members were provided on an as-needed basis in existing terminal room areas. Since that time, a large MPP User Room was made available with several VT100 terminals, four modern office cubicles, and one International Imaging Systems (IIS) image analysis terminal connected to the MPP's host VAX.

On-Line Data Storage: Limited on-line disk storage space is provided to Working Group Members for keeping their MPP programs and data. In addition, "scratch" disk space is available to all Facility users; scratch space is initialized weekly.

2.3 Scope of the FY87 MPP Program

FY87 Funding Profile

506-45-11	Computer Science	$200K
656-20-26	MPP Software	$250K
656-13-25	MPP Maintenance & Operations	$250K
	TOTAL	$700K

Allocation of FY87 Funds

MPP User Support Office	$ 25K
MPP System Software Development	$ 50K
MPP Algorithm Development	$191K
MPP Working Group Conferences	$ 45K
MPP Hardware Maintenance	$ 73K
Computer Facility Operations	$246K
Miscellaneous	$ 3K
IMS tax	$ 67K
TOTAL	$700K

Chapter 3 - Discipline Findings

3.1 Earth Sciences Group

3.1.1 Applications/Algorithms

The Earth Science studies encompass a number of diverse applications. In summary, they are:

1. Solution of the barotropic quasi-geostrophic equations for a model of synoptic scale dynamics in ocean circulation.

This involved solving partial differential equations on a nonuniform grid using a relaxation method and an alternating direction implicit method.

2. Classification of Landsat data using a maximum likelihood algorithm.

This involves applying a discriminant function to each pixel and determining which class yields the maximum value.

3. Solution of the equations of a numerical model of heat and moisture flow within a hillslope.

The equations are solved by calculating the spatial derivatives of the moisture flux using finite difference techniques and spatial derivatives of the heat flux by a force restore method, and then computing the time integral using an Adams-Bashforth predictor-corrector method.

4. Solution of the hydrodynamic equations for two-dimensional, nondivergent flow on a sphere as used in numerical weather prediction.

A Poisson equation for the stream function is solved by Fourier transforming and solving sets of tridiagonal equations. The vorticity equation is then updated in time.

5. Solution of the equations of a numerical model that describes the transport and removal of photochemical oxidants and acidic species and precursors in the troposphere.

A Crank-Nicolson Galerkin Finite Element Method for transport equations and Semi-Implicit Method for chemistry equations were used. A time-splitting technique was employed to achieve a high degree of parallelism.

6. Contextual classification of multispectral Earth observation imagery using a combination of spatial and spectral information in a statistical approach.

The approach is an expansion of the standard maximum likelihood classification approach, where information from a local neighborhood of image points is considered in calculating the discriminant function rather than just from the pixel being classified.

3.1.2 Summary of Findings

In general, all of the investigators found that the MPP was an effective tool for solving their problems, although they found some bottlenecks. In those applications that used finite difference methods on grids, the nearest-neighbor connectivity of the MPP suited the problem well and led to an efficient solution. In the one case (application 4) where a partial spectral method required large amounts of long range communication across the array, the performance was seriously degraded; here is an example of an algorithm that is not well suited to the MPP architecture. However, in another case (application 5), a major component of the photochemical model is the solution of the equations for the chemistry at every grid point at every time step. In doing this, there is no interprocessor data transfer needed; this is an application that is very well fitted to the MPP. Two classification applications have been implemented on the MPP with good results–one of the two requires only nearest neighbor and short distance communication and the other required no communication at all. Thus, both are well suited to the MPP architecture.

With one important exception, discussed below, all of the investigators programmed in the MPP's high level language, Parallel Pascal. All the applications have fairly large codes, largely using floating point arithmetic. It would not have been feasible to code these in assembly language with the time and manpower available. Many of the investigators had difficulties at the beginning of the study because of errors in the floating point primitives. These errors have now been corrected, and all of the investigators find Parallel Pascal a good programming language for the MPP.

The Parallel Pascal compiler is believed to generate code that is less efficient than that which could be produced by hand coding in assembly language. The degree of inefficiency is not known. One estimate, based on floating point operation counts and the number of cycles for each, suggests that the speed of the code could be doubled by coding in assembly language.

In one application, the hillslope model, a very significant speedup of a factor of 3 or 4 was obtained by multitasking. That is, the main control unit (MCU) and the PE control unit (PECU) were used concurrently with the array unit (ARU). This was done by assembly language coding because there is no provision in Parallel Pascal to indicate that a coarse grained parallelism exists in the algorithm and can be exploited by multitasking. In addition, this application required higher precision arithmetic than that provided by the standard 32 bit floating point primitives. This was provided by custom programming of arithmetic modules.

The members of the Earth sciences group found that the ARU memory (1K bits) was barely adequate for their applications. They expressed the desire to have a larger ARU and MCU memory, as well as a larger staging memory. This enlarged memory requirement ranged from an additional 1K bits for the classification calculation to 64K bits for the flow calculations. It must be emphasized that this requirement is for additional ARU memory, and in some cases MCU memory, and could not be met by increasing the size of the staging memory.

In one of the calculations, classification of the Landsat data, the front end VAX proved to be a severe I/O bottleneck. It would seem that this would be true of any application where large amounts of data must be moved in and out.

The performance of these codes on the MPP were timed and compared to codes, embodying the same algorithms, run on CYBER 205's, and CRAY-1's and 2's. The detailed results are given in the *Proceedings of the First Symposium on the Frontiers of*

Massively Parallel Scientific Computation[4]. Overall the performance on the MPP was at least comparable to that on CYBER's and CRAY's, and in some cases much better. We believe that the MPP-type architecture has more potential for development than do the more conventional architectures.

3.1.3 Recommendations

1. Retain special status of the MPP because it requires some experience and expertise to use effectively.

2. Continue the Working Group. Another announcement of MPP availability should be made.

3. Make it a major priority to bring up all standard features of Parallel Pascal, and make the software user friendly.

4. Greatly improve I/O between the host and the staging memory and also between the staging memory and image display devices.

5. Increase the size of the local ARU memory as much as possible. We think at least 64K bits per PE are needed.

6. If a new MPP is to be built larger than 128 x 128, which would be desirable, one would need long range communications. Studies should be done now to decide what form this should be; some possibilities are, at most a hypercube, a pyramid, or a shuffle. I/O bandwidth and internal memory size and software must be examined.

7. It is necessary to know the limitations on a small array machine and one should be obtained, perhaps a Mini-DAP. This has more memory per PE and some long-range communication.

8. For future MPP-like machines, the approach of using a far fewer number of more powerful processors is not recommended.

3.2 Physics Group

3.2.1 Applications/Algorithms

The following scientific problems were studied by the members of the Physics Group:

1. Electronic structures and associated hyperfine properties of atoms and condensed matter systems.
2. Charged particle transport.
3. Real-time animation of space plasma phenomena.
4. Simulation of beam plasma interactions.
5. Particle simulation of plasmas.
6. Phase separation by Ising Spin simulations.
7. Phase transitions in lattice field theories.
8. Wave scattering by arbitrarily shaped targets–direct and inverse.
9. Free-electron laser simulations.
10. Dynamics of collisionless stellar systems.

For the full titles and other details of these research programs, see *Appendix B–Working Group Membership.*

When these scientific problems are formulated theoretically, they are transformed into mathematical problems. A partial list of these problems is given below, together with the methods adopted for solving them on the MPP. The bracketed numbers refer to the original applications, listed above.

- Linear partial differential equations in one or two space-like dimensions, with time derivatives also involved in some cases. They are solved by direct numerical integration [10], by transform methods [4,5], or by a Monte-Carlo method.

- Equations of discrete particle dynamics in 2 dimensions. Solved by numerical integration, with the particle data kept either at fixed locations, in the processor array [4], or at locations corresponding to the instantaneous positions of the particles in the simulation domain [5].

- Analysis of statistical systems, solved by a Monte-Carlo method [7].

- Matrix operations, including inversions and eigenvalue problems [1].

- Numerical integrations of functions of up to three variables [1].

Such problems are not, of course, confined to theoretical physics, but also arise in many other scientific disciplines.

3.2.2 Summary of Findings

The experience gained in the realization of their research programs, and notably the factors affecting the performance of the algorithms on the MPP, as summarized in *Appendix C–Quantifications*, led the investigators to the following conclusions:

1. The scientific problems that map best onto the MPP are those that are inherently two-dimensional, and for which the algorithms require less than 1K of memory per processing

element, with no need for communication between remote pairs of PE's. For such problems, the MPP programs generally execute faster than the corresponding programs on conventional supercomputers such as the CRAY-1.

2. Other programs tend to be limited to speeds lower than that of a CRAY-1, either by the insufficiency of the PE or stager memories, or by the slowness of input/output operations.

3. The availability of bit-serial arithmetic proved to be important in some cases, by creating possibilities for the use of algorithms that would have been infeasible without it. The lattice gas approach to computational fluid dynamics is a case in point.

4. In general, the need was felt to develop new algorithms for the MPP, rather than recode for the MPP the algorithms that are preferred for conventional serial processors.

5. For the majority of investigators, who were located at sites other than GSFC, network access was essential. The reliability of the Space Physics Analysis Network (SPAN) was not always satisfactory.

3.2.3 Recommendations

The Physics Subgroup recommends the following steps to make the MPP system more responsive and useful for the problems currently pursued by the subgroup. The project should:

1. Investigate the possibility of increasing the capacities of the MCU and ARU memories.

Some physics programs exceed in length the capacity of the MCU memory, which obliges the programmer to use the host memory as a back-up, with a consequent speed penalty. Likewise, a larger ARU memory would reduce I/O and simplify programming; it would also allow the MPP to compute or simulate larger scale or more complicated physical problems.

2. Add virtual memory to the Parallel Pascal compiler.

It would be helpful if the performance of a program suffered only graceful degradation when it ran out of memory at each level. This result could be achieved by providing virtual ARU memory in the stager, and virtual MCU memory in the host.

3. Create a library of professional, user-friendly science subroutines and corresponding documentation.

There is a common need among group members for subroutines that perform simple algorithms, such as Fast Fourier Transforms, random number generation, sorting routines, and histogram sums. The DAP scientific subroutine library[8] could be used as a model for such a library for the MPP.

4. Investigate improvements to network reliability.

Group members have found difficulty in accessing the MPP via SPAN and ARPANET more often than they consider desirable; while this problem is not primarily an MPP responsibility, the group believes that the MPP should make its needs for increased reliability known to the network managers.

5. Optimize the Parallel Pascal compiler.

Currently, the Parallel Pascal compiler produces code that is far from optimum for simple and commonly used operations such as data shifts and rotates. Improvements are also needed in the compiler's use of the PE memory for temporary results, because PE memory is a very precious commodity in the MPP.

6. Develop a capability for handling arrays larger than 128 x 128, with software transparent to the user.

Some Group members make logical use of arrays larger than 128 x 128, e.g., 256 x 256. The MPP compiler should be capable of implementing such large arrays, and also three dimensional arrays, e.g., 16 x 32 x 32, without major reprogamming by the user.

7. Make other improvements to Parallel Pascal.

Currently, neither disk I/O nor the operations of the staging memory are integrated into Parallel Pascal, nor does Parallel Pascal implement the capability, which exists in the hardware, of transferring data back and forth between the stager and the host simultaneously with array operations. Moreover, Parallel Pascal has no double precision arithmetic for real numbers, no complex arithmetic, and no exponent function. Some existing intrinsic functions, such as the reduction sum, should be made more rapid.

8. Develop a shell or other means of supplying transparency and portability to user codes.

Users invest appreciable resources in developing and running code on the MPP system, so far without any assurance that their codes will be usable later or in other environments. Continued enthusiasm for the MPP system will depend on perception by the users that their future computing is secure from obsolescence. Item 6 above would help in this objective.

9. Incorporate comprehensive graphic output in the I/O hardware and software.

Many projects require visual interpretation of their results in conceptual or graphic form. This is especially true for the kind of complicated problems that recommend use of the MPP. Computer graphic representation is routinely taken for granted in such cases , and should be a standard available output medium for the MPP system. Both hardware and software support are needed.

10. Install a symbolic debugger.

The current MPP debugger (CAD) is rather primitive by modern standards. The Parallel Pascal compiler already produces the information required by a fully symbolic debugger; it should be a high priority to write such a debugger for the MPP and provide good documentation for it.

3.3 Signal and Image Processing Group

3.3.1 Applications/Algorithms

Scientific problems addressed by the Signal and Image Processing group fall into two major categories. The first of these fits closely with one of the original performance areas of the MPP, which is the highly parallel analysis, using loosely coupled local procedures on images and other two-dimensional data. Projects include:

- Processing of Seasat and SIR-B Synthetic Aperture Radar (SAR) phase history data into images.
- Development of an automated technique for matching corresponding pixels in stereo image pairs to determine elevations.
- Reconstruction of Coded-Aperture X-ray images.
- Biomedical Image Analysis.
- Three-dimensional reconstruction.

The other set of problems involve an abstract sequential model to generate artificial images (computer graphics) or to be "discovered" in an existing image (computer vision). Projects in this area include:

- Animated sequences from a polygon patch model.
- Recursive subdivision to form randomly textured surfaces.
- Array pattern recognition.

This might be termed the nonintuitive class of problems, because many of them have been met with solutions that were not, at the outset, obvious. The use of an abstract model requires a procedure to map the model to the array or to abstract the model from the array. In the straightforward image-processing tasks this is a simple wholesale raster transfer, and the challenges involve the design of the local computations. Graphics and vision models are inherently three-dimensional, so that the major initial challenge is developing efficient transformation of the model to fit the array.

3.3.2 Summary of Findings

As might be expected, image-processing projects experienced significant speedups, near the theoretical limit possible considering the number of processors and the speed of calculations. More surprising is that the model-based applications have also achieved significant speeds, well beyond that of a sequential model traversal. The fractal subdivision algorithm has, in fact, been timed to be twice as fast as a special-purpose chip designed for that purpose.

Any existing or projected function in this group can be done by special purpose hardware. We feel it is important to view the MPP not as an "image engine" or "graphics engine," but rather as a robust prototyping medium for a large class of conceivable engines. Many procedures need considerable simulation and fine tuning before committing their details to fabrication, and the MPP provides an ideal bed for this sort of work. For many of our applications, any of which might be simulated on a sequential machine, the MPP makes possible turnaround times that cannot be approached sequentially. Trial-and-error fine tuning, a highly iterative process, is possible in a matter of days rather than weeks.

Particularly in the non-intuitive problems, it is clear that the space of possible approaches is much larger than originally expected and that we are only at the beginning of discovery in this field, a finding which was not necessarily predictable. We expect this to have profound implications for both the theory and practice of computing.

In a general mode, the subgroup agreed upon several key points. The first of these is that each member is engaged in activities that would not have been contemplated were it not for the existence and availability of the MPP. Second, related to the fact that we are in the first years of a new system, most members experienced a great deal of hesitation in committing ideas to code, and thus "dragged their feet" until the operating environment became less volatile. Related to this, several noted the unavailability of a library of low-level routines of the type that one is reluctant to write until sure that the routine does not exist elsewhere in the system. This last point is particularly true of remote users who are not able to walk down a hall and find out who knows what.

The one member who did not drag his feet experienced considerable initial difficulty that would appear to justify the self-imposed delays of the others. This involved more than one generation of Parallel Pascal code generator, as well as considerable input-output difficulty. This person was able to make progress since he is on-site at Goddard, with daily access to the User Support staff.

While specific products of our involvement will be listed below, it is worth mentioning that the two group members on university faculties have been able to expose students to the MPP by use of the simulators running at our home institutions.

The subgroup agreed that no one wished to code in MPP assembler language (MCL) or microcode language (PEARL), and that only code in a high-level language is likely to survive in any event, which makes maintenance and support of Parallel Pascal of supreme importance, since this is at present the most likely gateway to other architectures.

While most applications are sized to fit the MPP's constraints as a matter of convenience, and few feel restricted, all agree that more processors and more memory per processor would be useful and advantageous. Only one member felt limited immediately by the size of local memory.

Most of the group use the MPP via remote access over SPAN, and the group has found this to be generally satisfactory. The applications are all data intensive, and are thus sensitive to the strengths and weaknesses of the front end computer's I/O capabilities.

As a practical detail, the VAX front-end computer has not been entirely satisfactory both because of its apparent overloading, and because of the difficulty of transferring data to and from the MPP.

The general impression of the subgroup is that the programming environment was probably undercommitted. The major symptom of this is the lack of a strong utility library, and the user support personnel necessary to maintain it. Those members who have made extensive use of User Support have indeed made considerable progress, but we feel that the resource is better spent on more widely useful products, which would include a well-documented library.

3.3.3 Recommendations

The support environment has changed markedly in the past 2 years. A complete documentation package is half of a file drawer, but the most recent MPP User's Guide[10] and the new MPP Pascal manual[13] provide a working basis, with some extensions, for a truly portable and useful starter set of information for new investigators. Some further determination of essential information, and its packaging, is still required. This recommendation immediately affects the next two.

At its current level the MPP is not suitable for inclusion in a general supercomputer facility, largely because the user support requirement would exhaust Goddard's resources very quickly. A concise "how to" document, and a truly portable simulator, might change this assessment. However, there still appear to be wrinkles and idiosyncracies that require ironing out before going public.

The Working Group can be expanded, subject to several considerations. First, it should be demonstrated that the first cohort made progress, otherwise we simply invite others to a shared futility. This might be determined by some threshold number of papers or working systems. Second, documentation should be to the level that new users do not require extensive assistance from the User Support staff. Specifically, new users should be able to develop their own programs, reserving the User Support staff for debugging and similar technical assistance.

A broad spectrum of applications and programming styles has now been applied to the MPP for several years, and, in the process, many investigators have accumulated and, reinvented in many cases, a considerable body of private knowledge. At some point in the Working Group's life, it would be desirable to collect some of that experience, if only to centralize papers and technical reports prepared elsewhere.

At present, the Reeves Parallel Pascal appears to be the most aggressively ported parallel language with any direct relevance to the MPP environment. It should be noted that many "parallel" languages are in fact targeted at the multiple instruction multiple data stream (MIMD) environment, achieving their parallelism by explicit fork and join instructions (in UNIX terms). In order to provide a gateway for existing and future MPP software it seems desirable to make MPP Pascal compatible with Parallel Pascal, and to bring it up to the capability of sequential Pascal (enumerations, sets, etc.). In a related vein, there appears to be considerable room for improvement in the optimality of compiled code, and once the new MPP Pascal code generator has been tested that issue should be examined.

The other software requirement we see is a more comprehensive and accessible library of utility functions to extend capabilities to other MPP features that are currently not available through Pascal. The group feels strongly that support and improvement of MPP Pascal and establishment of a library are the major and exclusive priorities for support resources.

It is in the interest of both NASA and the scientific community at large that massively parallel computation be pursued both at the research and production levels. First, a great body of tasks exist that adapt well to parallel computation and that cannot be done any other way. Second, the scope for future expansion and enhancement of parallel architectures is orders of magnitude greater than for vector processors. We feel that the particular hardware that may augment or supplant the MPP is not nearly so critical to continued success as is the preservation of a stable programming environment.

3.4 Computer Sciences Group

3.4.1 Algorithms/Application

The algorithms considered within the Computer Science group include several general areas of interest, including:

1. Finite element analysis
2. Parallel nested dissection
3. Local relaxation with feedback
4. Parallel graph theory algorithms
5. Image processing algorithms
6. Routing algorithms
7. Simulation of systolic VLSI systems.

The applications of these algorithms are diverse:

1. Two-D and three-D electromagnetic field computation
2. Neural networks and connectionist models of artificial intelligence
3. Three-D interactive graphics in real time
4. Image processing
5. Long range communication
6. Functional programming language implementation
7. Solving sparse linear systems.

These projects constitute research in parallel algorithms, instead of using algorithms to carry out research in some other branch of science. Consequently, the members of the Computer Science Group are experimenting with their programs, and do not anticipate reaching a point where the programs are finished and used to process large amounts of data. This makes the MPP working environment–remote access, the compilers, the host operating system, etc.–especially important.

Data Structure Parallelism

Several of these research projects involve "data structure parallelism" instead of the more common "control structure parallelism." In the past, most work on parallel algorithms has focused on control structures rather than data structures, so it is especially important that the MPP Working Group is contributing to this promising area.

A MIMD computer uses control structure parallelism: the programmer or compiler finds a way to partition an algorithm into a set of processes that can execute concurrently, and schedules each process to run on a separate processor. The programming languages developed for this approach use control statements to specify where there is parallelism. Examples include fork/join, cobegin/coend, etc.

In data structure parallelism, the algorithm has a sequential organization with no explicit concurrent processes. However, the algorithm may execute instructions that perform global computations in parallel on a large data structure. For example, consider the task of marking every element of a list or set (this problem is important in symbolic programming languages). The conventional method would be to mark the elements one by one, following a pointer from each element to the next. A program using parallel control structures would try to make several processors execute that algorithm. In contrast, the parallel data structure approach would provide a single instruction that marks all the elements. The processor issues the instruction (as part of a sequential program), but *the memory that contains the data structure performs the actual work.*

The MPP supports data structure parallelism by combining processing power with each record in the memory. It is necessary to have a very large number of processors, but it is not necessary for the processors to be powerful or to have a very large amount of memory. The Connection Machine is another good architecture for data structure parallelism, and it would be useful to compare its performance with the MPP's for several parallel data structure algorithms. The Connection Machine has a richer interconnection network than the MPP, but the cycle time of the MPP is faster. Consequently some parallel data structure algorithms will run faster on the Connection Machine and others will run faster on the MPP. It would be very fruitful to analyze these issues further.

3.4.2 Findings and Recommendations on Programming Languages

The Computer Sciences group has, in general, found that the present MPP Pascal high-level language facility is a reasonable one for research and development using the MPP, although it does need to be embedded in a more sophisticated working environment.

The availability of a good high level-language such as MPP Pascal has been found especially important for MPP projects in which the program changes from experiment to experiment (as opposed to projects in which only the data sets change between experiments). Nevertheless, some members of the group have found it useful or desirable to write some or all of their systems in one or both of the low-level languages (MCL assembler language or PEARL microcode language) to get adequate efficiency. The need for the low-level languages arises particularly in applications that involve unusual forms of data transfer among ARU processing elements.

Several programs developed by the computer science group do not involve extensive numerical computation, in contrast to the programs developed by the other groups. For nonnumerical programs, and even perhaps for many numerical ones, a Pascal-based high-level language is more attractive than FORTRAN-based languages, such as those usually used on CRAY or CYBER 205 systems. To this extent the MPP programming environment is more congenial than CRAY or CYBER 205 environments.

Various significant changes that have been made to MPP Pascal during the life of the Working Group have been very beneficial in themselves but have required major program changes by some group members from time to time. Also, some of the more advanced but standard data-structuring facilities of Pascal are only now being incorporated, so that the programming environment is not yet as stable as one would wish.

The recent introduction of streamlined input/output facilities into MPP Pascal is welcomed as it enhances the ease, speed, and reliability of programming. One point still causing concern is the need to have some user-programmed ancillary FORTRAN modules to accompany an MPP Pascal program.

Recommendations on Programming Languages

- MPP Pascal should be strongly supported and maintained.

- Enhancements to MPP Pascal should continue.

- Effort should **not** be diverted into developing high-level languages for the MPP other than MPP Pascal. It is much more important to concentrate on making the MPP Pascal environment as good as possible.

- The **need** for user-written FORTRAN modules to supplement MPP Pascal programs should be eliminated.

- However, the facilities for supplementing MPP Pascal programs with FORTRAN modules, as well as MCL and PEARL modules, should be maintained and enhanced.

3.4.3 Findings and Recommendations on the MPP Working Environment

Most new users of the MPP need considerable help in organizing their algorithms and using the MPP software. The Goddard User Support Office has done an excellent job of providing the Working Group Members with support.

Interactive network access has been very successful. Most users find it most productive to develop their programs locally on their own VAXes using the MPP Simulator or the MPP Pascal Translator. They then transfer the source program files to the MPP and log on interactively to run the programs. This procedure is relatively easy and uses both researcher and machine. It allows the researchers to use their own computing environment while testing and debugging, and it reduces the load on the MPP.

Recommendations on the MPP Working Environment

We recommend several enhancements to the software environment which would improve productivity:

- Install a good screen editor that works on all standard ASCII terminals. Emacs would be the best choice.

- Provide software for data compression in order to support higher effective bandwidth over low cost communications paths.

- The MPP Pascal compiler should be made compatible with the Translator. This would make it easier to develop programs, and would reduce the demand on the MPP hardware for debugging. Since the MPP Pascal compiler doesn't implement the full language, several users found that programs developed on the Translator required major changes before they would work on the MPP.

3.4.4 Findings on Architecture

There are several very different types of parallel computer currently available, and techniques that work well on one type of machine may not transfer well to other types. The MPP is representative of fine grain machines with bit serial processors and a mesh interconnection. The results of the Working Group members' projects indicate how successful this architecture is for a wide range of problems.

The MPP's processing element architecture is extremely well designed and implemented. Frequently each processing element can do several operations simultaneously, and the shift register supports efficient arithmetic algorithms.

The choice of a bit serial architecture is a good design tradeoff. Many algorithms use integer operands (sometimes with short lengths), and some algorithms are inherently bit serial. The MPP's architecture can execute all of these classes of algorithm efficiently. If the processing elements had been designed with floating point hardware but no bit serial capability, the machine would have cost much more but would have been suitable for a smaller set of applications.

The current configuration of the MPP has several limitations, although in most cases it is possible to work around them. The most serious problem is that each processing element has only 1024 bits of local memory. Most of the users had to work carefully to avoid exceeding this limit, and some had to resort to using the staging memory as a bit plane virtual memory. A large number of applications would work better if each processing element had a somewhat larger memory, around 64K bits. (However, some three dimensional simulation algorithms would require a far larger memory in each processor.) The Main Control Unit memory is also sometimes a limitation.

The mesh interconnection network limits the speed of long-distance communication in the array unit. However, the fast cycle time allows short and medium distance messages to travel very rapidly. The MPP is best suited for algorithms that rely mostly on communications over short paths.

The current VAX-11/780 front end interface is a bottleneck. Upgrading the front end system will improve the system response and increase MPP I/O throughput.

3.4.5 Findings and Recommendations on Performance

The MPP has delivered outstanding performance in a wide variety of algorithms and applications, some of which are far removed from its original intended uses.

Since the MPP was designed for image processing, its superb performance in that area is not surprising. Image processing and similar problems typically involve many logical and small integer operations, and local or short-range communications. The MPP is well matched to these problems because its processing elements are directly connected to their four nearest neighbors in a grid topology.

In addition, the MPP has also proven very effective at traditional number crunching problems such as solving systems of linear equations. Much of the success in the CRAY-type problems is due to the use of shift registers to speed multiplication in the processing elements. Efficient coding of algorithms also appears to be very important.

Several members of the Working Group are developing algorithms that involve symbolic computing, graph theory, routing, neural network computation, etc. Some of these areas do not have an obvious two-dimensional grid structure. This implies that the MPP's area of applicability is much broader than originally thought.

Several promising lines of research in computer science are based on large numbers of small computing elements that communicate with each other. For example, neural networks, cellular automata, systolic VLSI systems, and data-parallel algorithms all fall in this category. The MPP is a good architecture for studying such systems, and it seems to be more natural for these research areas than vector supercomputers or coarse-grain MIMD architectures.

Recommendations on Performance

1. Develop more cellular-automaton algorithms. For example, how fast does Wolfram's fluid flow algorithm run on the MPP? Also, give further study to the Ising Spin exchange model.

2. Develop more parallel data structure algorithms. How well does the MPP compare with related architectures, such as the Connection Machine, for executing such algorithms?

3. Profile existing algorithms to understand performance bottlenecks. Find out what performance gains would accrue from:

   - more memory
   - more efficient scalar communications (broadcasting)
   - more connectivity.

4. Determine more precisely the benefits of massive parallelism in number crunching. Compare MPP with intermediate designs (e.g., 128 8086/8087's). How much CRAY time is communication?

5. Continue to develop theoretical techniques that shed light on the inherent complexity of algorithms, and try to use these to develop better parallel algorithms.

3.4.6 Findings and Recommendations on the Working Group

Research using the MPP has been an extremely rewarding experience for the members of the Computer Science group. Several of us are working on problems that would be out of reach using conventional computers. Access to the MPP is allowing the working group to experiment with state-of-the-art computing facilities, and it is bringing parallel computing to the universities long before they would be able to afford it using their own resources. Several graduate students are learning about parallel programming.

The Working Group has shown that the MPP is useful for a much wider range of applications than originally intended. Providing access to new users will probably extend this range further.

In addition to showing the usefulness of the MPP, this work has a long-term significance. The future development of parallel architectures will rely on experience with many different algorithms and many architectures. By supporting the MPP Working Group, NASA has provided 35 new data points that will improve the state-of-the-art for all parallel computation. This experience will be useful to the designers of future parallel computers, and it will add to our knowledge of parallel algorithms.

Recommendations on the Working Group

Since the MPP Working Group has been so successful, it should be continued. The basic structure works well: researchers who wish to use the MPP write a proposal that is reviewed by a panel of scientists who are expert in the capabilities of the machine and who can assess the quality of the proposed research. The authors of successful proposals will enter the Working Group.

Since the capacity of the MPP is not fully used at present, the size of the Working Group could be allowed to grow substantially. It might be best to add about 10 new members per year.

A new user must make a substantial effort to learn how to use the MPP effectively. The present members of the Working Group should use their experience to try to help new members get started as quickly and easily as possible. It is important to allow new members several years of access to the MPP, because short-term access would not give them time to learn to use the system, develop their algorithms, and obtain useful results. The Working Group workshops at Goddard have been helpful for the current Working Group members, and more should be held for the new members. Experienced users could give short, intensive, hands-on guidance to the people learning to use the MPP.

The User Support Group has been very helpful, and much of the success of our research is due to them. It would also be helpful to develop some simple examples of program development, as well as enhanced documentation and subroutine libraries.

Appendix A - Significant Accomplishments

My application, simulation of a neural net model, was far easier to program on the MPP than on conventional computers. The simulation model is based largely on two-dimensional arrays processed in a way that can easily be represented in the MPP Pascal high-level language, and that allows efficient, natural use of the MPP Array Unit. A conventional computer would require more complex programming in order to avoid intolerable inefficiency. My application involves extensive experimentation with different versions of the neural net model, resulting in frequent, reasonably major program modifications. It is, therefore, important that changes be easy to make.

–John A. Barnden

A new gridless model for particle simulation has been developed. The gridless model is fully parallelized with respect to the conventional particle-in-cell model due to the architecture of the MPP. The parallel nature of the MPP has made the discrete Fourier Transform (DFT) a viable alternative to the Fast Fourier Transform (FFT). The program is cost efficient relative to CRAY CPU time.

–Chin Lin

In my application (Synthetic Aperture Radar signal processing), I have been able to take advantage of the full parallelism of the MPP Array Unit for computations, the staging memory for in-place transposition of intermediate data, and the high bandwidth between the ARU and the staging memory for minimizing the intermediate data I/O time. Even with Parallel Pascal programming the "kernel" process in my application, a 4096-complex element Fourier transform, needs only 1.25 milliseconds. The total computation time on the MPP for single-look processing of a 4096 x 4096 array of SIR-B signal data is only 19 seconds. (The wall clock time is much larger due to conventional I/O hardware on the host VAX computer.)

–H.K. Ramapriyan

Since 1979, I have been doing research on parallel computer hardware to support functional (or applicative) programming languages. My hardware model is well suited for VLSI implementation, but is impossible to simulate adequately on a conventional machine, on a vector supercomputer, or on a MIMD machine with a small number of processors. Since this is an experimental architecture, a VLSI implementation would be very costly and inflexible; fast software emulation on a parallel fine-grain SIMD machine is needed. The MPP and the Connection Machine are the only two currently available machines that are suitable. Access to the MPP has made it possible for me to make rapid progress in this research in the last year. In particular, I have developed:

- A general solution to the functional aggregate update problem, which has been a major problem for data flow architectures and applicative architectures. It now appears that a machine like the MPP would make an excellent structure memory for data flow systems.
- A parallel garbage collector.
- A parallel combinator reduction system on the MPP is in progress. This looks promising, but much work remains.

–John T. O'Donnell

The goal of my project is to develop parallel algorithms for interactive, real-time manipulation of stereo pair graphic images. Terrain models defined over grids are the images of interest. Standard graphics operations of rotation, translation, scaling, perspective projection, and shading are computed for a 128 x 128 grid of elevations. The second image of a stereo pair is computed from the first by a rotation about the Y axis. Preliminary results based on one test case indicate that computational aspects of rendering stereo images can be done faster than the refresh rates of display devices. That is, in less than 1/30 of a second as needed for real-time manipulation. Similar computation on a VAX-11/780 took approximately 100 seconds, yielding a speedup of 3000 times for the MPP compared to a VAX. It would not be feasible to carry out this research on graphics algorithms without access to the MPP.

–Edward Davis

In my implementation of the contextual classifier on the MPP, I made no concerted effort to come up with the most efficient implementation possible. Still, this relatively inefficient implementation provides better than a 100-fold speed-up over a fairly efficient VAX-11/780 implementation of the algorithm. This amount of speedup is sufficient to make it possible for the first time to study the effectiveness of this classifier on several different data sets of reasonable size (e.g., 512 x 512 pixels).

–James Tilton

A note on comparing the MPP implementation of the contextual classifier with a CRAY implementation: It makes little sense to compare the speed of the contextual classifier on a vector supercomputer such as a CRAY to the speed of the implementation on the MPP. Devising an implementation on the MPP that effectively uses the parallelism of the MPP is very easy and natural, whereas it would be much more difficult to develop an implementation on a vector supercomputer that effectively exploits that type of parallelism. Being such an easy and natural implementation, the MPP implementation lends itself much more effectively to experimentation with variations of the algorithm.

–James Tilton

A matching algorithm has been implemented on the MPP. Cross-correlation between local neighborhoods is performed to find corresponding pixels in both images. Increased resolution at each iteration is obtained by decreasing the size of the neighborhood used in the matching step. An interactive shell program has been developed on the MPP to allow modifying input parameters such as neighborhood size at each step in the matching process. The matching algorithm has been applied to SIR-B synthetic aperture radar images and to stereo cloud-cover images taken from synchronous orbit. In both cases, matches (for areas where the signal-to-noise ratio was high) were obtained automatically with as much accuracy as those produced by human observers.

–James Strong

I have demonstrated the ability to do operations on the MPP that would normally require generalized routing. One example is table look-up. This is based on a sort routine with a modified comparison step.

–John Dorband

The mapping of inherently nongrid problems to the MPP array has generated a number of unexpected possible strategies. The space of solutions appears much broader and richer than previously imagined. The coherence sought in graphics rendering algorithms is well suited to the array communication structure.

–Michael McAnulty

The MPP made it possible for the first time to collect sufficient statistics to investigate vortex-vortex correlations in the planar spin model in a quantitative manner.

–Peter Suranyi

A calculation was performed on the MPP of intermittency for a Boolean delay equation of theoretical importance in modeling Quarternary Glaciations, and also related to the predictability of a deterministic white noise. The algorithm was programmed in MPP Parallel FORTH using 32-bit fixed precision arithmetic. The 1K bit PE memory limitation forced the choice of an inferior algorithm to that run on serial and vector machines. The algorithm achieved a speedup of 4,000 compared to a VAX-8650, and is estimated to run 50 times faster than on a CRAY-1 compensating for the difference in algorithms.

–A.P. Mullhaupt

Our initial analysis of two-center integrals in molecular wave-function calculations has convinced us that the MPP will be a very effective computational tool and substantially reduce the computational time that one now needs with conventional computers. However, one needs very different algorithms and reprogramming of current procedures, which we are currently working on.

–N. Sahoo, S.N. Ray, and T.P. Das

As researchers at a university, my colleagues and I were pleased to discover that the opportunity of working with the MPP is highly attractive to graduate students, and not only within the Department of Electrical Engineering, to which we belong. Our two salaried research assistants, both from the Department of Computer Science, cooperated with us enthusiastically and productively, while two other graduate students came voluntarily to work on our project for academic credit alone.

–Owen Storey

Appendix B - Working Group Membership

Although this appendix lists only the 39 principal investigators and technical staff for the 40 current Working Group projects, a separate document, *Abstracts of the MPP Working Group*[3] is available with more detailed descriptions of each project.

Earth Science Group

Tropospheric Trace Gas Modeling on the MPP
Dr. Gregory R. Carmichael, Chemical and Materials Engineering
University of Iowa, Iowa City, Iowa
Seog Y. Cho, Department of Chemical & Materials Engineering
University of Iowa, Iowa City, Iowa
David M. Cohen, Department of Computer Science
University of Iowa, Iowa City, Iowa
Mehmet H. Oguztuzun, Department of Computer Science
University of Iowa, Iowa City, Iowa

Kalman Filtering and Boolean Delay Equations on an MPP
Dr. Michael Ghil, Department of Atmospheric Sciences
University of California, Los Angeles, California
Dr. A.P. Mullhaupt, Department of Mathematics & Statistics
University of New Mexico, Albuquerque, New Mexico

Adapting a Navier-Stokes Code to the MPP
Dr. Chester E. Grosch, ICASE
NASA–Langley Research Center, Hampton, Virginia and
Old Dominion University, Norfolk, Virginia
R. Fatoohi, ICASE
NASA–Langley Research Center, Hampton, Virginia

A Physically-Based Numerical Hillslope Hydrological Model with Remote Sensing Calibration
Dr. Robert J. Gurney, Hydrological Sciences Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. P.J. Camillo, Hydrological Sciences Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
E.T. Engman, Hydrology Laboratory
U.S. Dept. of Agriculture, Beltsville, Maryland
A.A. van de Griend, Hydrological Sciences Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
A.R. Irving
Science Applications Research, Lanham, Maryland
Judith E. Devaney
Science Applications Research, Lanham, Maryland

A Comparison of the MPP with Other Supercomputers for Landsat Data Processing
Mr. Martin Ozga, Statistical Reporting Service
U.S. Department of Agriculture, Washington, D.C.

Use of Spatial Information for Accurate Information Extraction
Dr. James C. Tilton, Information Analysis Facility
NASA–Goddard Space Flight Center, Greenbelt, Maryland

A Magnetospheric Interactive Model Incorporating Current Sheets (MIMICS)

Dr. Elden C. Whipple, Jr., Center for Astrophysics and Space Sciences
University of California at San Diego, La Jolla, California

Physics Group

Investigations on Electronic Structure and Associated Hyperfine Properties of Atoms and Condensed Matter Systems Using the MPP
Dr. Tara Prasad Das, Department of Physics
State University of New York, Albany, New York
Dr. Surendra N. Ray
ST Systems Corp., Lanham, Maryland
Narayan Sahoo, Department of Physics
State University of New York, Albany, New York

Numerical Calculations of Charged Particle Transport
Dr. James A. Earl, Department of Physics and Astronomy
University of Maryland, College Park, Maryland

Simulation of Beam Plasma Interactions Utilizing the MPP
Dr. Chin S. Lin, Department of Space Sciences
Southwest Research Institute, San Antonio, Texas
A.L. Thring, Department of Space Sciences
Southwest Research Institute, San Antonio, Texas
J. Koga, Department of Space Sciences
Southwest Research Institute, San Antonio, Texas
R.W. Janetzke, Department of Space Sciences
Southwest Research Institute, San Antonio, Texas

Particle Simulation of Plasmas on the MPP
Dr. L.R. Owen Storey, STAR Laboratory,
Stanford University, Stanford, California
Dr. I.M.A. Gledhill, STAR Laboratory
Stanford University, Stanford, California
Dr. Oscar Buneman, STAR Laboratory
Stanford University, Stanford, California

Phase Separation by Ising Spin Simulations
Dr. Francis Sullivan, Scientific Computing Division
National Bureau of Standards, Gaithersburg, Maryland
Dr. Raymond D. Mountain, Thermophysics Division
National Bureau of Standards, Gaithersburg, Maryland

A Study of Phase Transitions in Lattice Field Theories on the MPP
Dr. Peter Suranyi, Department of Physics
University of Cincinnati, Cincinnati, Ohio
Paul Harten, Department of Physics
University of Cincinnati, Cincinnati, Ohio

Wave Scattering by Arbitrarily Shaped Targets–Direct and Inverse
Dr. William Tobocman, Department of Physics
Case Western Reserve University, Cleveland, Ohio
William H. Vonder Haar, Department of Physics
Case Western Reserve University, Cleveland, Ohio

Free-Electron Laser Simulations on the MPP
Dr. Scott Von Laven,
KMS Fusion, Inc., Ann Arbor, Michigan and
Mission Research Corporation, Albuquerque, New Mexico
Lorie M. Liebrock
Michigan Technological University, Houghton, Michigan

The Dynamics of Collisionless Stellar Systems
Dr. Richard L. White,
Space Telescope Science Institute, Baltimore, Maryland

Signal & Image Processing Group

Fixed Point Optimal Nonlinear Phase Demodulation
Dr. Richard S. Bucy, Department of Aerospace Engineering
University of Southern California, Los Angeles, California

Pattern Recognition on an Array Computer
Dr. Y. Paul Chiang, Department of Electrical & Computer Engineering
Washington State University, Pullman, Washington

Graphics Applications of the MPP
Dr. Edward W. Davis, Department of Computer Science
North Carolina State University, Raleigh, North Carolina
Dr. David F. McAllister, Department of Computer Science
North Carolina State University, Raleigh, North Carolina
Sanjay Pol, Department of Computer Science
North Carolina State University, Raleigh, North Carolina

Automatic Detection and Classification of Faint Galaxies on Deep-Sky Astronomical Pictures
Dr. Sara R. Heap, Astronomy Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Donald J. Lindler
Advanced Computer Concepts, Inc., Potomac, Maryland

Application of Parallel Computers to Biomedical Image Analysis
Dr. Robert V. Kenyon, Dept. of Electrical Engineering and Computer Science
University of Illinois, Chicago, Illinois

Comet Halley Large-Scale Image Analysis
Dr. Daniel A. Klinglesmith, III, Science Operations Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. John E. Dorband, Image Analysis Facility
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Archibald Warnock III,
ST Systems Corporation, Lanham, Maryland

Synthetic Aperture Radar Processor System Improvements
Dr. Stephen A. Mango
Naval Research Laboratory, Washington, D.C.
S. Walter McCandless
User Systems, Inc., Annandale, Virginia

Development of Automatic Techniques for Detection of Geological Fracture Patterns
Dr. H.K. Ramapriyan, Information Analysis Facility
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. Paul D. Lowman, Geophysics Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. Herbert W. Blodget, Geophysics Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Mr. Edward J. Seiler,
Science Applications Research, Lanham, Maryland

Development of an Improved Stereo Algorithm for Generating Topographic Maps Using Interactive Techniques on the MPP
Dr. James P. Strong, Information Analysis Facility
NASA–Goddard Space Flight Center, Greenbelt, Maryland

Reconstruction of Coded-Aperture X-Ray Images
Dr. Lo I. Yin, Solar Physics Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. Michael J. Bielefeld
Computer Sciences Corporation, Baltimore, Maryland

Computer Sciences Group

Diagrammatic Information Processing in Neural Nets
Dr. John A. Barnden, Computer Science Department
Indiana University, Bloomington, Indiana

Sorting and Signal Processing Algorithms: A Comparison of Parallel Algorithms
Dr. Howard B. Demuth, Department of Electrical Engineering
University of Idaho, Moscow, Idaho

Space Plasma Graphics Animation
Mr. Eugene W. Greenstadt, Space Sciences Department
TRW, Redondo Beach, California
Karen F. Jordan, Space Sciences Department
TRW, Redondo Beach, California

Parallel Algorithms for Graph-Theoretic Problems
Dr. Susanne E. Hambrusch, Computer Sciences Department
Purdue University, West Lafayette, Indiana

Applications of Stochastic and Reaction-Diffusion Cellular Automata
Dr. Harold M. Hastings, Department of Mathematics
Hofstra University, Hempstead, New York
Dr. Michael Conrad, Department of Biological Sciences
Wayne State University, Detroit, Michigan
Dr. Stefan Waner, Department of Mathematics
Hofstra University, Hempstead, New York

Sorting and Signal Processing Algorithms: A Comparison of Parallel Architectures
Dr. Martin T. Hagan, Electrical Engineering Department
University of Tulsa, Tulsa, Oklahoma

Solution of Complex, Linear Systems of Equations
Dr. Nathan Ida, Electrical Engineering Department
University of Akron, Akron, Ohio
Kapila Udawatta, Electrical Engineering Department
University of Akron, Akron, Ohio

FORTH, An Interactive Language for Controlling the MPP
Dr. Daniel A. Klinglesmith III, Science Operations Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. Arne A. Henden,
ST Systems Corporation, Lanham, Maryland
Dr. John E. Dorband, Image Analysis Facility
NASA–Goddard Space Flight Center, Greenbelt, Maryland

Algorithmic Commonalities in the Parallel Environment
Dr. Michael A. McAnulty, Department of Computer and Information Sciences
University of Alabama, Birmingham, Alabama
Dr. William M. Siler, Clinical Computing
Carraway Methodist Medical Center, Birmingham, Alabama
Michael S. Wainer, Department of Computer and Information Sciences
University of Alabama, Birmingham, Alabama

Simulating an Applicative Programming Storage Architecture Using the NASA Massively Parallel Processor
Dr. John T. O'Donnell, Computer Science Department
Indiana University, Bloomington, Indiana

Parallel Solutions of Very Large Sparse Linear Systems
Dr. John H. Reif, Computer Science Department
Duke University, Durham, North Carolina
Dr. Torstein Opsahl, Perkin-Elmer Advanced Development Center
MRJ, Inc., Oakton, Virginia

Massively Parallel Network Architectures for Automatic Recognition of Visual Speech Signals
Dr. Terence J. Sejnowski, Biophysics Department
Johns Hopkins University, Baltimore, Maryland
Dr. Moise H. Goldstein Jr., Dept. of Electrical Engineering & Computer Science
Johns Hopkins University, Baltimore, Maryland
Ben P. Yuhas, Dept. of Electrical Engineering & Computer Science
Johns Hopkins University, Baltimore, Maryland

Animated Computer Graphics Models of Space and Earth Sciences Data Generated Via the Massively Parallel Processor
Mr. Lloyd A. Treinish, National Space Science Data Center
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. Chester J. Koblinsky, Geodynamics Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Dr. Richard A. Goldberg, Electrodynamics Branch
NASA–Goddard Space Flight Center, Greenbelt, Maryland
Michael L. Gough
Science Applications Research, Lanham, Maryland
Mr. David Wildenhain
Sciences Applications Research, Lanham, Maryland

Appendix C - Quantifications

This Appendix presents quantitative data concerning the ways in which the Working Group has solved problems on the MPP. Various parameters are quantified for each MPP project in the tables that appear on the following pages. These parameters are:

(a) **Physical lattice size** - This item applies only to physical phenomena taking place in a two-dimensional spatial domain, that is divided up into a grid and mapped more or less directly onto the array of processing elements. Obviously, problems having a physical lattice size of 128 x 128 are very well suited for solution on the MPP.

(b) **Physical vector size** - This item is the maximum number of data that need to have the same mathematical operation performed upon them at a given step in the calculation. Problems where this number is smaller than 16,384 do not make full use of the parallel computing power of the MPP. Problems where the number is greater than 16,384 could, in principle, be solved faster on an even larger machine of the same type.

(c) **Word sizes (in bits) for the physical variables** - This item indicates how well the problem is suited to the 1-bit arithmetic used in the MPP array unit.

(d) **Memory required per processing element (PE)** - Requirements of fewer than 1K bits can be satisfied by the array memory. Requirements for a further 16K can be met by the staging memory. Total requirements exceeding 17K necessitate use of the host memory as well.

(e) **Communication between processing elements** - Since only nearest-neighbor connections exist between its PE's, the MPP is best suited to problems not calling for long-range communication. The following codes are used to indicate the type of communication required in each project:

(0) No communication needed.
(1) Communication with nearest neighbors only.
(2) Remote communication needed.

(f) **Percentage of time spent on input to and output from the array unit** - This item indicates to what extent the speed of solution of the problem is limited by I/O.

(g) **Overall performance relative to a CRAY-1** - The number tabulated here is the time required to complete the solution on a CRAY-1, divided by the corresponding time on the MPP; in some cases these times have been measured, in others only estimated. The algorithms need not be the same for the two types of computer. If performance statistics relative to a CRAY are unavailable, comparison is sometimes made between the MPP and a VAX.

Table 1. Quantifications

PHYSICS GROUP

Project	(a) Lattice	(b) Vector	(c) Word	(d) Mem.	(e) Comm.	(f) %I/O	(g) CRAY-1/MPP
DAS	n/a	not fixed	32	<1K	(1) & (2)	25%	no comparison made
EARL	128x128	16Kx31	3,16,19	1K	(1)	very small fraction	>>1 (estimated)
GREENSTADT	128x128 32x32	16K in both cases	32	<1K	(1)	>90%	350 x VAX (measured)
LIN	n/a	16Kx120	32	1K	(2)	probably large	0.1 (estimated)
STOREY	128x128	16Kx(1-120)	3,7,12, 20, & 32	17K (all)	(1) for particles (2) for fields	94% (est. particles)	0.1 (est. particles)
SULLIVAN	512x512	256K	1	1K (needs more)	(0) particles (2) for fields	--- 5%	2 (measured)
SURANYI	128x128 64x64 32x32	16K 4K 1K	1,32	1K	(1) & (2) for small lattices	0%	7-14 times (measured)
WHITE	128x128	16K	32	640	(1)	10-95%	~1(measured)

Table 1. Quantifications (continued)

EARTH SCIENCE GROUP

Project	(a) Lattice	(b) Vector	(c) Word	(d) Mem.	(e) Comm.	(f) %I/O	(g) CRAY-1/MPP
CARMICHAEL	128x128	16,384	32	17K	(0) 90% chemistry (1) 10% transport		30 (chemistry) <=1 (transport)
GROSCH	128x128 128x256		32	17K	(0) 30% (1) 50% (2) 20%	0% for 128x128 50% for 128x256	175 Mflops for 128x128 94 Mflops for 128x256 (CRAY-2 single processor 100 Mflops)

SIGNAL & IMAGE PROCESSING GROUP

Project	(a) Lattice	(b) Vector	(c) Word	(d) Mem.	(e) Comm.	(f) %I/O	(g) CRAY-1/MPP
WAINER (McAnulty)	128 x 128	---	32	1000	(1)	50%	1000 x VAX (est)
MCANULTY	128 x 128	---	16	1000	(1)	10%	100 x VAX (est)
HEAP	512 x 512 (images) 121 x 121 (matrixes)	4900 vectors of length 121	1 (logical) 8 (integer) 32 (floating point)	800	0 (60%) 1 (10%) 2 (30%)	20%	

Table 1. Quantifications (continued)

COMPUTER SCIENCE GROUP

Project	(a) Lattice	(b) Vector	(c) Word	(d) Mem.	(e) Comm.	(f) %I/O	(g) CRAY-1/MPP
BARNDEN	multiple 32x32	n/a	1-8 bits	most	(1) mostly (2) via sum-or	80% est.	not known
DAVIS	128x128	128x128	32	most	(1) or (0)	not measured	3000xVAX (small sample)
HASTINGS	128x128 x5x5	128x128 x5x5	1-12	800	(1) mostly (2) 5x5 region	<20%	2 (est.)
	128x128	128x128 x2	1-8	100	(1) mostly (2) 5x5 region	<10%	5 (est.)
	128x128	128x128 x2	1-8	100	(1)	<10%	5 (est.)
IDA			32				10 (est.)
O'DONNELL	n/a	n/a	1-8	700	(1) mostly (2) some	1-5%	16000 x VAX
OZGA	128x128	10^7	8,32	17K	(0)	70%	.6

Appendix D - Lists

Papers

Earth Science Group

"A SIMD Implementation of a Distributed Watershed Model," J.E. Devaney, P.J. Camillo, and R.J. Gurney, in *Proceedings of the Second International Conference on Supercomputing*, Santa Clara, California, May 1987.

"Massively Parallel Correlation Techniques to Determine Local Differences in Pairs of Images," J.P. Strong and H.K. Ramapriyan, *Proceedings of the Second International Conference on Supercomputing*, Santa Clara, California, May 1987.

"Implementation of a Four-Color Cell Relaxation Scheme on the MPP, FLEX/32, and CRAY/2," R.A. Fatoohi and C.E. Grosch, to appear in *The 1987 International Conference on Parallel Processing.*

"Chemical Network Problems Solved on NASA/Goddard Massively Parallel Processor Computer," S.Y. Cho, G.R. Carmichael, D.M. Cohen, and M.H. Oguztuzun, *AICHE Annual Conference*, 1986.

"Automated Matching of Pairs of SIR-B Images for Elevation Mapping," H.K. Ramapriyan, J.P. Strong, Y. Hung, and C.W. Murray, *IEEE Transactions on Geoscience and Remote Sensing*, GE-24, No. 4, July 1986, pp. 462-472.

Physics Group

"Charged Particle Transport Calculations on the Massively Parallel Processor," Dr. James Earl, paper submitted to the 20th International Cosmic Ray Conference, Moscow, U.S.S.R., August 1987.

Signal and Image Processing Group

"Limitations of Inherently Parallel Graphics Algorithms," Michael McAnulty, *Proceedings of the Second International Conference on Supercomputing* , Santa Clara, California, May 1987.

"A Fractal Subdivision Algorithm for Mesh-Connected Computer," Michael S. Wainer, *IEEE Transactions on Computers,* 1987 (accepted for publication).

"Reconstruction of Coded Aperture X-Ray Images," Michael Bielefeld, *Proceedings of the SPIE Conference,* August 1987 (in press).

"Block Iterative Restoration of Astronomical Images on the MPP," Sara Heap and Don Lindler, *Proceedings of the 169th Meeting of the American Astronomical Society*, Pasadena, California, January 1987.

"Deconvolution of Quasar 2130 + 099," Sara Heap and Don Lindler, *Proceedings of the 169th Meeting of the American Astronomical Society*, Pasadena, California, January 1987.

"Classification of Multispectral Data Using Contextual Information," Dr. James Tilton, invited paper at An Interdisciplinary Workshop on the Theory and Methods of Pattern Recognition With Applications to Multispectral Data Analysis, Utah State University, Logan, Utah, May 8-9, 1986.

"Computational Investigations Using the Massively Parallel Processor," Dr. James Tilton, invited paper at An Interdisciplinary Workshop on the Theory and Methods of Pattern Recognition With Applications to Multispectral Data Analysis, Utah State University, Logan, Utah, May 8-9, 1986.

"Estimation of A Priori Probabilities for Bayesian Classification of Multispectral Image Data," Dr. James Tilton, presentation at Workshop on Analytical Methods in Remote Sensing for Geographic Information Systems, sponsored by the International Association on Pattern Recognition Technical Committee 7, Paris, France, October 23-24, 1986.

Computer Sciences Group

"Principles of Evolutionary Learning–Design for a Stochastic Neural Network," H. Hastings and S. Waner, *BioSystems,* **18** (1985), 105-109.

"Biologically Motivated Machine Intelligence," H. Hastings and S. Waner, *SIGART Newsletter,* 19-31, January 1986 (not referreed).

"Evolutionary Learning of Complex Modes of Information Processing," H. Hastings and S. Waner (submitted).

"Functional Aggregate Update on a Parallel Processor Architecture," John T. O'Donnell.

"Data Parallel Garbage Collection," John T. O'Donnell (in progress).

"Finely Grained Parallelism in an Applicative Architecture," Dr. John T. O'Donnell, *Proceedings of the Workshop on Future Directions in Computer Architecture and Software,* Army Research Office, May 1986.

"Simulating VLSI Systems Using the Massively Parallel Processor," Dr. John O'Donnell, *Proceedings of the 1986 Summer Computer-Simulation Conference,* Society for Computer Simulation, July 1986.

"Parallel Implementation of Field Solution Algorithms," Dr. Nathan Ida, to be presented at COMPUMAG, Graz, Austria, August 1987. Accepted for publication in *IEEE Transactions on Magnetics.*

Technical Reports

Computer Sciences Group

"A Study of MPP Implementations for the Connected Component Labeling Problem," S.E. Hambrusch, L. TeWinkel, Technical Report, Purdue University.

"Complex Cognitive Information Processing: A Computational Architecture with a Connectionist Implementation," J.A. Barnden, Technical Report 211, Computer Science Department, Indiana University.

Theses

"Sort Computation and Conservative Image Registration," J.E. Dorband, Ph.D. Thesis, Department of Computer Science, Pennsylvania State University, December 1985.

"Solution of Large Linear Systems of Linear Equations on a Massively Parallel Processor," Kapila Udawatta, Ph.D. Thesis, University of Akron, August 1986.

"A Parallel Finite Element Analyzer," Wang Jian-She, Ph.D. Thesis in progress, University of Akron.

Students

Physics Group

[Storey] Shaw-Ben Shi, Computer Science major from Stanford University, worked on the MPP from 9/1/85 to 3/31/86 and again from 7/1/86 to 12/31/86.

[Storey] Igor Dayen, Computer Science major from Stanford University, worked on the MPP from 9/1/85 to 3/31/86.

[Storey] Elgin H. Lee, Electrical Engineering major from Stanford University, worked on the MPP from 9/1/86 to 12/31/86.

[Storey] Stanley H. Yau, Engineering Economic Systems major from Stanford University, worked on the MPP from 9/1/86 to 12/31/86.

[Grosch] Raad A. Fatoohi, Ph.D student, Department of Electrical Engineering, Old Dominion University.

Computer Sciences Group

[Demuth] Derek Brown, a graduate student at the University of Idaho, is using a variation of the Mandelbrot set calculation as a benchmarking tool. This was a research project in EE504: High Performance Computing.

[Demuth] Joe Hicklin, a graduate student at the University of Idaho, has implemented a neural network experiment tool to allow users to perform a broad range of experiments on large neural nets.

[Hambrusch] Jeffery Hostetler and Lynn TeWinkel, computer science graduate students at Purdue University, started working on the MPP in October 1986.

[McAnulty] Michael Wainer is a Ph.D. student, Department of Computer & Image Science, University of Alabama at Birmingham.

[Carmichael] Mehmet H. Oguztuzun, Ph.D student, Department of Computer Science, University of Iowa.

[Hastings] Steven Rosenthal, Mathematics Department, Hofstra University.

[Davis] Sanjay Pol, M.S. student in Electrical and Computer Engineering, North Carolina State University, Raleigh.

[Davis] K.L. Duh, Electrical and Computer Engineering, North Carolina State University.

[Ida] Kapila Udawatta, a Computer Engineering graduate student at the University of Akron, worked on the MPP from 9/85 to 9/86.

[Ida] Wang Jian-She, an Electrical Engineering graduate student at the University of Akron, began work on the MPP 10/86.

[Ida] All students in my Computer Algorithms course (20 students) use the MPP Simulator for homework assignments.

Grants

A proposal with the title "MPP Implementations for Graph and Computer Vision Problems" has been submitted to NSF, but no decision has been made. The principal investigators are S.E. Hambrusch and C. Guerra.

A grant titled "Investigation of the Usefulness of the Massively Parallel Processor to Study Electronic Properties of Atomic and Condensed Matter Systems" is cofunded by NASA and the State University of New York at Albany. The principal investigator is T.P. Das.

A proposal titled "Complex Cognitive Information Processing" has been submitted to NSF, AFOSR and ONR. The principal investigator is J.A. Barnden.

A grant titled "Utilization of MPP for Investigation of Electronic Structures and Hyperfine Properties of Atomic and Condensed Matter Systems," has been submitted to the Department of Energy, Principal Investigator is T.P. Das.

A grant titled "Particle Simulation of Plasmas on the Massively Parallel Processor," has been funded by the National Science Foundation. Principle Investigator is L.R.O. Storey.

A grant proposal titled "Graphics Algorithms on Parallel Architectures," was submitted to the National Science Foundation. Principle Investigators are D.F. McAllister and Edward W. Davis.

A grant titled " Computer Simulation of Rarefied Media," is funded by the NASA–Ames University Consortium. Principal Investigator is L.R.O. Storey.

A smaller grant from the Stanford–Ames Institute for Space Research was awarded to L.R.O. Storey.

A grant titled "Complex Cognitive Information Processing: A Computational Architecture with Connectionist Implementation," was received by John Barnden.

A grant application is being submitted to the National Science Foundation in the Fall of 1987. Principal Investigator will be Harold Hastings.

"A Pyramid Approach to Image Segmentation," is a proposal funded by the NASA–GSFC Director's Discretionary Fund for FY87. Principal Investigator is James Tilton. Part of this project is funded for Curtis E. Woodcock of the Department of Geography, Boston University for a proposal titled, "The Use of Spatial Features in Image Segmentation."

A grant proposal titled "Graphics Algorithms on Parallel Architectures," was submitted to the National Science Foundation. Principal Investigators are Dr. D.F. McAllister, and Dr. E.W. Davis.

"Generation of Fractally Textured Surfaces by Recursive Subdivision," is a project performed by Michael S. Wainer under a NASA Graduate Student Fellowship, 1985-present.

"A grant proposal titled "Investigating Net Behavior by Simulation," was submitted by Michael McAnulty to the U.S. Army.

"Parallel Computing for Finite Element Analysis," is a proposed grant from NASA–Lewis Research Center, (pending), Principal Investigator is Nathan Ida.

Appendix E - Technical Summary of the MPP

INTRODUCTION

The Massively Parallel Processor[1] (MPP) is an advanced computer architecture termed single-instruction stream multiple-data stream (SIMD) that shows promise of delivering enormous computational power at lower cost than other existing architectures. Its computational element, the array unit, consists of a 128 by 128 array of small 1-bit processors, each containing 1,024 bits of local memory, and having nearest neighbor connectivity. A secondary storage unit, the staging memory, holds 32 megabytes of data and connects to the array memory via an 80 megabyte per second data path. An array control unit broadcasts control signals to all processors in the array unit. The MPP is a back-end processor for a VAX-11/780 host, which supports its program development and data needs. (See Figure 1.)

The MPP was built for the Goddard Space Flight Center by Goodyear Aerospace Corporation. Delivery of the system took place in May 1983. At that time, the construction of a digital processor using the very high degree of parallelism embodied in the MPP had not been previously attempted.

MPP SOFTWARE

Since its delivery to Goddard, an extensive language system and a unique operating system have been implemented for the MPP. This system software repertoire is relied on daily by dozens of teams of scientific investigators who are developing, testing, and running parallel algorithms.

High Level Languages

The initial high level language implemented in 1983 was **Parallel Pascal**[11]. This language was designed to be independent of the computer's architecture, thus allowing portability of applications programs between diverse parallel computers having Parallel Pascal compilers. Experience gained in the development and use of this approach showed that the 128 by 128 square grid architecture of the MPP could not easily be hidden from the programmer by using current compiler writing technology.

A modified language, **MPP Pascal**[12,13], was then implemented that is architecture dependent and that possesses important semantic features allowing the programmer to make very efficient use of the hardware's capabilities. The MPP Pascal compiler is capable of producing highly optimized code and is sufficiently flexible to allow easy modification. MPP Pascal is currently in user field test.

Under the auspices of one of the Working Group proposals, the **FORTH**[14] language was ported to the MPP and extended to permit the MPP to be operated in an interactive manner. MPP Parallel FORTH became operational in December 1986. The MPP is also programmable in assembly language[15,16].

Operating System

The MPP operating system provides interactive debugging aids[17] in addition to support for running applications code. The software that performs these tasks is shared by all MPP users, greatly reducing the demand on the host's main memory. The debugging aids include performance monitoring, error reporting of MPP hardware-detected faults, breakpointing, single-step, and status display. A first-come-first-served queue is the central arbiter controlling user access to the MPP.

All MPP applications programs must be prepared as two parts. One part runs in the MPP control unit and the other part runs on the host. They are linked together through a message passing system. A master/slave control relationship exists between the MPP and the host. The host resident program is the highest level of control. This program interacts with the user and starts MPP programs. MPP programs, in turn, use the host as an I/O server, directly accessing the host's disk and image analysis terminals through an extensive set of I/O service routines.

A device driver that communicates directly with the MPP hardware runs at the lowest level in the host operating system. This driver is the hub of the entire system, controlling the execution of programs in the MPP as well as the flow of data throughout the system. The bulk of the operating system interacts directly with this device driver to accomplish tasks in support of a running application such as initializing the hardware, loading programs, starting and stopping programs, reading and writing data, and delivering messages between running programs in the host and the MPP.

Libraries

Computational: A number of libraries of computational subroutines are supported. One type holds more than 270 microcoded subroutines that define the actual instruction set of the MPP. These include the basic arithmetic and transcendental functions as well as multi-precise arithmetic and special user-written instructions. A second type of library holds MPP Pascal-callable subroutines, including fast fourier transforms, a random number generator, a sort computation package, linear algebra routines, and utility programs.

Data I/O[18]: For many applications, having only 1,024 bits of memory available to each of the 16,384 MPP processors has been a serious constraint. This limitation was imposed by the memory chip technology available in 1980 when the system was designed. As an alternative to an expensive hardware upgrade, bit-plane I/O software was developed that treats the staging memory as individual bit planes. A system was implemented that provides each processor with 16K bits of virtual array memory. The penalty is an increase in memory access time from 0.1 microseconds to 25 microseconds per bit plane; however, many applications benefit from this virtual memory as they effectively overlap computation with data transfer. In addition to bit-plane I/O, another system, SMM I/O, gives the user access to the powerful data reformatting capabilities of the staging memory. A set of libraries holds I/O subroutines which control the movement of data within the system.

Simulation Environments

Two MPP simulation environments have been developed and distributed to user sites remote from Goddard.

MPP Simulator[6]: The MPP Simulator supports the development, testing, and refinement of MPP Pascal or assembly language applications programs on any VAX operating under VMS. It allows a user the convenience of a local dedicated MPP that does not have to be shared with other users. In addition, its use at remote sites off-loads program preparation work from the MPP/VAX system at Goddard. Code that runs on the Simulator will run on the MPP after adjusting any references to the size of the array unit (usually simulated as a 16 x 16 array to speed execution).

Parallel Pascal Translator[7]: The Parallel Pascal Translator takes Parallel Pascal source code as input and produces equivalent serial Pascal source code as output. The serial Pascal can be compiled and executed using a standard Pascal compiler system. The Translator allows the development, testing and refinement of applications programs on most computers that have a Pascal compiler.

MPP HARDWARE

Array Unit

The Array Unit is the 128 x 128 array of processing elements (PE's) that supplies the MPP's computational power. Each PE has a local 1,024 bit random access memory and is connected to its four nearest neighbors–north, south, east and west. Opposite array edges can be connected together to form either a plane, a horizontal cylinder, a vertical cylinder, or a torus. Arithmetic and logic in each PE are performed in a bit-serial manner. All operands are fetched in bit-serial fashion from the 1,024-bit local memory and results are constructed in the memory in like fashion. The cycle time is 100 nanoseconds. Table 1 shows the raw computing speeds for selected arithmetic operations. The data-bus states of all 16,384 PEs are combined in a tree of inclusive-or logic elements whose single-wire output is used in the Array Control Unit for operations such as finding in parallel the maximum or minimum value in an array.

MPP Processing Element: A single PE is shown functionally in Figure 2. The P-register, together with its input logic, performs all Boolean logic functions on two variables and can also receive data from the P-register in any one of its four nearest neighbors. The A-, B-, and C-registers, the shift register, and associated logic form an arithmetic unit. The G-register controls masking of arithmetic, logic and routing operations. (Unmasked operations are performed in all PE's. Masked operations are performed only in those PE's where the G-register is set.) The S-register is used to shift data to and from the Staging Memory without disturbing PE operations. A custom integrated circuit (IC) holds eight PE's, exclusive of the 1,024 bits of random access memory, which is on a separate IC chip. A one-bit wide bidirectional data bus connects the memory and the internal components of the PE.

Microcircuit Technology: All components of eight PE's, exclusive of random access memory, are packaged on a custom integrated circuit. This chip uses high speed complementary MOSFET (HCMOS) technology. This chip contains a two row by four column array of PE's.

Array Control Unit

The Array Control Unit broadcasts control signals and memory addresses to all PE's in the Array Unit and receives Array Unit status bits. It is designed to perform bookkeeping operations (address calculation, loop control, branching, subroutine calling, etc.), and control the Array Unit simultaneously. It contains three parts: (1) the Main Control Unit, (2) the PE Control Unit, and (3) the I/O Control Unit.

The Main Control Unit executes the application program stored in its program memory. It performs the scalar arithmetic operations required, calls the PE Control Unit for all array logic and array arithmetic operations, and calls the I/O Control Unit for all I/O operations. Both sets of calls are queued to await execution while the Main Control Unit moves on to generate other calls.

The PE Control Unit generates all Array Unit instructions except those pertaining to the S-register (data I/O). It executes microcoded routines stored in its program memory to perform all array operations required by applications programs.

The I/O Control Unit controls the shifting of I/O data through the Array Unit S-registers as well as the transfer of I/O data between the S-registers and the Array Unit memory. It executes I/O channel programs stored in the Main Control Unit's program memory.

Staging Memory

The MPP system includes a Staging Memory for buffering Array Unit data. This memory provides both the "corner turning" function, which converts conventional byte or word oriented data into the bit plane form needed by the Array Unit, and the "multidimensional access" function, which allows large multidimensional arrays of data located in the Staging Memory to be read out or written in along arbitrary orderings of array dimensions. The current capacity of the Staging Memory is 32 megabytes, upgradable to 64 megabytes.

Data moves between the Array Unit and the Staging memory via 128 parallel lines. The upper limit on the transfer rate is 1.28 billion bits/second. Goddard's MPP currently supports .64 billion bits/second. Data movement in both directions can be overlapped with processing.

Host Processor

A DEC VAX-11/780 computer manages data flow between MPP units, loads progams into the Control Unit, executes system test and diagnostic routines, and provides program development facilities. The MPP is interfaced to the VAX through a 5 megabyte/second DR-780 channel. Remote access to the VAX is provided through ARPANET, SPAN, BITNET, TELENET, and dial-in.

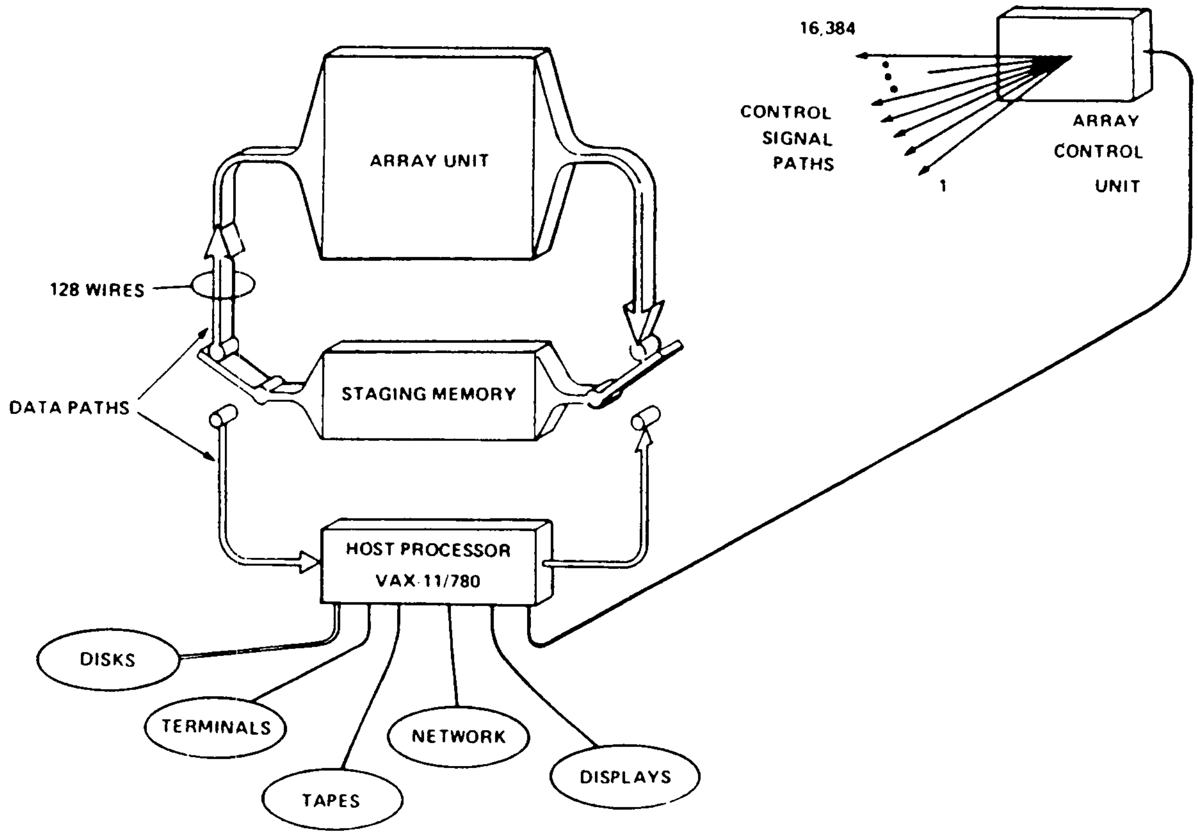

Figure 1. Configuration of the MPP

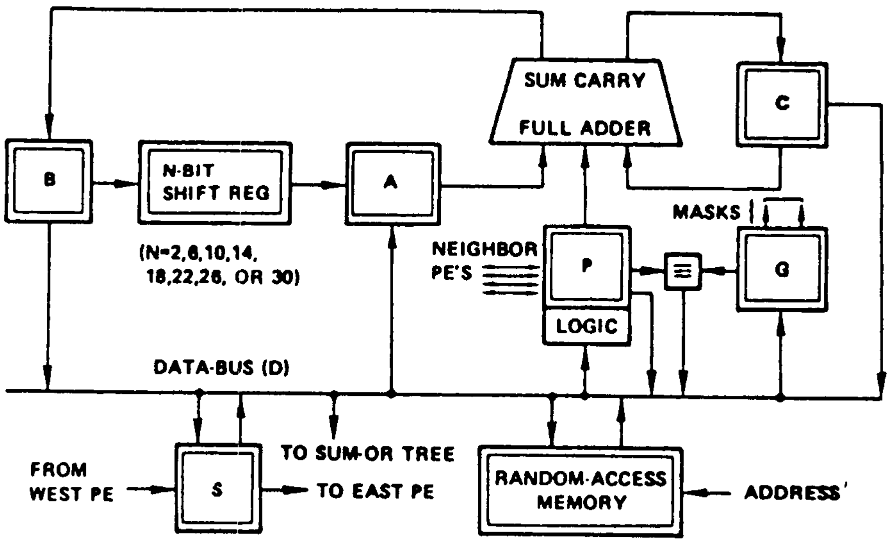

Figure 2. Functional Units of One PE

SPEED OF TYPICAL OPERATIONS

OPERATION	EXECUTION RATE (MOPS)
Addition of Arrays	
8-Bit Integers (9-Bit Sum)	6553
12-Bit Integers (13-Bit Sum)	4428
32-Bit Floating Point Numbers	430
Multiplication of Arrays	
8-Bit Integers (16-Bit Product)	1861
12-Bit Integers (24-Bit Product)	910
32-Bit Floating Point Numbers	216
Multiplication of Array By Scalar	
8-Bit Integers (16-Bit Product)	2340
12-Bit Integers (24-Bit Product)	1260
32-Bit Floating Point Numbers	373

*MOPS - Million Operations Per Second

Table 1. Speed of Typical Operations

References

All MPP documents, exclusive of [1], may be obtained from the MPP User Support Office, Code 635, NASA/Goddard Space Flight Center, Greenbelt, MD 20771.

[1] *The Massively Parallel Processor*, J.L. Potter, ed., ISBN: 0-262-16100, MIT Press, 1985.

[2] NASA Space Science & Applications Notice, *Computational Investigations Utilizing the Massively Parallel Processor*, December 1984.

[3] *Computational Research Using the Massively Parallel Processor–Abstracts of the MPP Working Group*, September 1986.

[4] *Proceedings of the First Symposium on the Frontiers of Massively Parallel Scientific Computation*, J. Fischer, editor, NASA-CP 2478, 1987.

[5] *Introduction to Data Level Parallelism,* Thinking Machines Technical Report 86.14, April 1986.

[6] *The MPP Simulator: Vol. I, User's Manual*, J.E. Devaney and Scott McEwan, December 1985.

[7] *Parallel Pascal Development System*, Version 2.0, October 1986, A.P. Reeves, Cornell University.

[8] Distributed Array Processor (DAP) Subroutine Library, Queen Mary College, London, England.

[9] *Computing on the MPP at the Goddard Image and Information Analysis Center*, February 1986.

[10] *MPP User's Guide*, February 1986.

[11] "Parallel Pascal: An Extended Pascal for Parallel Computers," A.P. Reeves, *Journal of Parallel and Distributed Computing*, Vol. 1, 1984, pp. 64-80.

[12] "MPP Pascal: Mapping a Language to the Machine," T. Busse, T. Opsahl, and J. Abeles.

[13] *MPP Pascal User's Guide*, September 1986.

[14] *MPP Parallel FORTH User's Guide*, J.E. Dorband, September 1986.

[15] *MPP Main Control Language - MCL*, September 1985.

[16] *MPP PE Array Language - PEARL*, January 1986.

[17] *Control and Debug (CAD) User's Manual*, GER 17142, May 1983.

[18] *MPP Pascal Callable I/O Procedure Library*, January 1986.

National Aeronautics and Space Administration

Report Documentation Page

1. Report No. NASA TM-87819	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Report From the MPP Working Group to the NASA Associate Administrator for Space Science and Applications		5. Report Date November 1987
		6. Performing Organization Code 635.0
7. Author(s) James R. Fischer, Chester Grosch, Michael McAnulty, John O'Donnell, and Owen Storey		8. Performing Organization Report No. 87B0265
		10. Work Unit No.
9. Performing Organization Name and Address Goddard Space Flight Center Greenbelt, Maryland 20771		11. Contract or Grant No.
		13. Type of Report and Period Covered Oct. 1,1985—Sept. 30,19 Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546-0001		14. Sponsoring Agency Code

15. Supplementary Notes

Chester Grosch is affiliated with ICASE at Langley Research Center; Michael McAnulty is affiliated with the University of Alabama; John O'Donnell is affiliated with Indiana University, and Owen Storey is affiliated with Stanford University.

16. Abstract

Through a national solicitation initiated in 1984 by NASA's Office of Space Science and Applications (OSSA), a pioneering team of 40 scientists was provided the opportunity to test and implement their computational algorithms on the Massively Parallel Processor (MPP) located at NASA's Goddard Space Flight Center in Greenbelt, Maryland beginning in the fall of 1985.

On February 24, 1987, the Working Group presented this report to Dr. Burton I. Edelson, Associate Administrator of OSSA, concisely stating their results from using the MPP for one year. The report addresses: algorithms, programming languages, architecture, programming environment, the way theory relates, and performance measured. The findings point to a number of demonstrated computational techniques for which the MPP architecture is ideally suited. For example, besides executing much faster on the MPP than on conventional computers, systolic VLSI simulation (where distances are short), lattice simulation, neural network simulation, and image problems were found to be easier to program on the MPP's architecture than on a CYBER 205 or even a VAX. The report also makes technical recommendations covering all aspects of MPP use, and strategic recommendations concerning the future of the MPP and machines based on similar architectures. Strategic recommendations called for issuance of a new applications notice to expand the Working Group, institutionalization of support for the MPP at current levels, collaboration with other agencies to broaden technology applications, and initiation of a study of the role of future parallel processors for Space Station, EOS, and the Great Observatories era.

17. Key Words (Suggested by Author(s)) Parallel Computer, Parallel Algorithm, Supercomputer, Simulation, Systolic, Image Processing, Bit-Serial		18. Distribution Statement Unclassified - Unlimited Subject Category 62	
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of pages 66	22. Price A04