The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture

G. F. Pfister, W. C. Brantley, D. A. George,
S. L. Harvey, W. J. Kleinfelder, K. P. McAuliffe,
E. A. Melton, V. A. Norton, J. Weiss

IBM T.J. Watson Research Center
Yorktown Heights, NY

Abstract: As a research effort to investigate both hardware and software aspects of highly parallel computation, the Research Parallel Processor Project (RP3) has been initiated in the IBM Research Division, in cooperation with the Ultracomputer Project of the Courant Institute of NYU. The RP3 machine being designed is a highly parallel MIMD design with a uniquely flexible organization encompassing both shared memory paradigms and local memory message-passing paradigms, as well as mixtures of the two chosen at run time. It is being designed to accommodate 512 state-of-the-art microprocessors. A full configuration will provide up to 1.3 GIPS, 800 MFLOPS, 1-2 Gbytes of main storage, 192 Mbytes/second I/O rate, and 13 Gbytes/second inter-processor communication. Performance evaluations indicate that approximately 1 GIPS performance should be sustainable. This is the first of a set of papers describing the RP3 [9] and the performance analysis [10,12] on which its design is based.

1.0 Introduction

To investigate both hardware and software aspects of highly parallel computation, the Research Parallel Processor Project (RP3) has been initiated in the IBM Research Division, in cooperation with the Ultracomputer Project of NYU. This paper describes the architecture of the RP3 and the initial software support planned; and discusses characteristics of the project as a whole. Companion papers submitted to this conference will discuss the machine design in more detail [9] and present the performance analysis techniques and results that lead to the RP3 design [10,12].

2.0 Overview

RP3 is a highly parallel, high performance MIMD processing system with a uniquely flexible organization encompassing both shared memory paradigms (e.g., the NYU Ultracomputer [4]) and local memory message-passing paradigms (e.g., the Cal Tech "Cosmic Cube" [14]), as well as mixtures of the two chosen at run time. It is being designed to accommodate 512 state-of-the-art microprocessors. A full configuration will provide up to 1.3 GIPS, 800 MFLOPS, 1-2 Gbytes of main storage, 192 Mbytes/second I/O rate, and 13 Gbytes/second inter-processor communication. Hardware support for performance monitoring is included throughout the system; it allows measurement of many performance-related parameters with minimal impact on system performance.

The performance sustainable by a full system as a function of instruction and data cache miss ratios is illustrated in Figure 1. This figure does not take into account algorithmic and software overheads due to locking, serialization, etc., but does account for data access delays due to contention in the interconnection network, at the memory modules, etc.; its derivation is described in a companion paper [10]. Here we note that, at least for scientific application code, we expect that instruction miss ratios will be in the 1%-2% range and data miss ratios will be in the 20% range; this has been verified by simulation of some codes. Under those circumstances, RP3 should deliver sustained performance of approximately 1 GIPS. This estimate is conservative, since it assumes that all memory traffic is global, and hence passes through the switch; it does not take into account fairly obvious optimizations of placing code and/or local data in local memory, where its access bypasses the switch.

Given the processing speed possible with RP3, we consider its peak I/O bandwidth to be much less than optimum. Lacking other firm guidelines, it would be preferable to follow the traditional rule of one megabyte/second/MIP, implying a peak I/O bandwidth of 1G byte/second. Unfortunately, at 3M bytes/second peak rate for the I/O devices used, this would imply in excess of 300 DASD units, which is physically impractical.

RP3 is a mixed technology system: The processing elements — processor, main memory, and support hardware — are constructed from FET logic and are air-cooled. A major portion of the interconnect network — a variant of an Omega network [8] — is, on the other hand, constructed from high-speed water cooled bipolar logic packaged in thermal conduction modules (TCMs) [1]. Our performance evaluation, described in a companion paper [10], has shown that this provides the low-latency access to global memory needed to meet our performance goal of 1 GIP. The relative access speeds for cache, local, and global

0190-3918/85/0000/0764$01.00 © 1985 IEEE

memory, using the bipolar network, are 1:10:15. This use of mixed technologies in support of shared memory is a potentially controversial aspect of RP3. It is discussed in more detail in a later section.

A sub-network constructed from FET logic provides the function of combining memory references as proposed for the NYU Ultracomputer [4]; a companion paper indicates why we consider this function necessary [12]. (The bandwidth, but not the latency, of the combining network is actually identical to that of the bipolar network. However, only the bipolar network's bandwidth was included in the 13G bytes/second figure quoted above.) The network is described in a later section.

Figure 2 shows the floor plan of a full RP3 configuration. The outer diameter is approximately 32 feet. Each "H-shaped" unit contains a full 64-way subsystem, complete with memory, I/O connections, and a portion of the network. The parallel elements of the "H" contain processing elements, memory, etc; the central crossbar of the "H" contains the bipolar network switch logic. Notice that the network consumes substantially less than one third of the volume of the system. Our estimates indicate that the network accounts for far less than one half of the development and manufacturing cost of the system.

3.0 Purpose

It must be emphasized that the purpose of the RP3 project is research into both hardware and software aspects of parallel processing. There are several implications of this:

1. RP3 is not a product, nor is it associated with any product currently under development. We of course hope it will influence future products.
2. Very few copies of the machine will be built. As this is written, there are no firm plans for replication or for the field support that replication would imply.
3. Given the prior item, whenever a trade-off was possible between a large quantity of hardware and additional design effort, we chose the large quantity of hardware.
4. Finally, it is intended that RP3 be an "open" project: We hope to collaborate with a number of different organizations, primarily in software development. Most aspects of hardware collaboration are impractical due to the proprietary nature of the technologies used in the design.

4.0 Architecture

The architecture of RP3 is illustrated in two figures: Figure 3 shows the processor/memory element (PME) structure and network connection, while Figure 4 shows support for I/O, performance monitoring, and console functions.

4.1 Processor/Memory Element (PME)

Each processor/memory element (PME) contains a high-performance 32-bit state-of-the-art microprocessor; 2M-4M bytes of memory; a 32K byte "intelligent" cache memory; floating-point support; and an interface to an I/O and support processor (ISP) (shown in Figure 4). As part of address translation, the PMEs provide a memory mapping function unique to RP3, allowing memory to be dynamically partitioned between efficiently-accessible global and local memory. The latter is described in the next section.

The RP3 processor is a proprietary design based on the "801" [13] philosophy that all instructions should nominally complete in a single cycle; unlike other examples of what has been called the RISC organization (e.g., [11,6,7]), it has an extensive instruction set and performs necessary interlocks internally in hardware. The floating-point support unit provides floating-point operations on 32- and 64-bit quantities; both scalar and storage-to-storage vector operations are provided (see companion paper [9]).

Referring to Figure 3, memory references issued by a processor, floating point unit, or I/O interface are first translated by a memory mapping unit. In addition to providing conventional segment/page mapping, address translation provides additional information unique to RP3. Some of this information controls the local/global memory support of RP3, described later. Other information is used to help solve the "cache coherence" problem: avoiding having one datum with several values because it appears in several caches. (For a discussion of this problem, see [16] and the references it contains).

RP3 solves the cache coherence problem in software, with hardware assist: A high level language programmer can declare appropriate data "shared." The compiler, in cooperation with the linker and run time storage allocation system, then puts that data in pages marked "uncacheable" in the memory map. We do not believe that this will prove onerous, on two grounds: First, the programmer must already know which data is shared for program operation to be correct. Second, an optimizing compiler must know which data are shared, since certain optimizations cannot be performed on shared read/write data (e.g., moving invariant computations out of loops and allocating data only in registers). In addition to the above technique, cache control operations are included which allow cached data to be invalidated on a line, page, or segment basis (see companion paper [9]). In conjunction with a store-through cache policy, this allows higher performance to be obtained when the program structure gives processes exclusive access temporarily to portions of shared data: the shared data can be cached during the time the process has exclusive access.

Once a memory reference has either bypassed the cache or resulted in a cache miss (see Figure 3), it reaches the network interface with a global address, part of which specifies the PME in which the referenced data lies. If the

reference is to data in the PME where the request originated, the network interface simply sends the request directly to the PME's memory, without involving the network. If the reference is to data in another PME, the reference is sent over the network, where another PME's network interface responds to it, altering or retrieving the appropriate information. In either event, a response is returned through the network to the initiating PME. There the response is treated as if it had been generated by the local memory.

4.2 Addressing Structure

RP3 allows all of primary memory to be partitioned between global, i.e., equally accessible memory; and local memory, i.e., memory accessible with minimal contention by a single processor. This is done as part of memory mapping by specifying whether, after normal translation, interleaving and hashing[1] is to be performed on the address.

One way of using this feature is shown in Figure 5, which schematically represents how the global address space is distributed across the PMEs. Part of each PME's memory (shown at left) is allocated to interleaved, global memory; there, memory and network contention is reduced by wide interleaving and hashing to a uniform quantity included in performance analysis. The rest of each PME's memory (shown at right) can be used as true local memory, entirely located within the PME, accessible rapidly bypassing the network.

Moving the local/global boundary to the far right makes RP3 a pure shared-memory machine like the NYU Ultracomputer. Moving it to the far left makes RP3 a pure local-memory machine whose processes can communicate by message-passing (implemented using inter-memory block transfer and other features). Intermediate boundary positions provide a readily-used form of "mixed mode" computation: shared-memory oriented applications can allocate private data locally, gaining efficiency; message-oriented applications can use global memory to aid load balancing. Code may reside locally or globally.

The interleave amount is variable, set by a field in the mapping tables. Together with bounds registers loadable from the console and diagnostic system, this allows RP3 to be partitioned into completely independent sub-machines. Thus one RP3 complex can simultaneously host a "floor" OS, an experimental OS, application measurement experiments, etc.

4.3 Network

The interconnection network of a shared memory parallel processor is a dominant component with respect to system performance. Low latency, adequate bandwidth, and the ability to "combine" synchronization requests are all required to robustly achieve the level of system performance desired. As previously mentioned, to achieve these goals the RP3 interconnection network is composed of two networks. One network provides low latency. The other network has the ability to combine messages, in particular interprocessor coordination function, directed to the same memory location.

The combining network has the geometry of Lawrie's Omega network. The low latency network is a rectanguar SW banyan [3], similar to an Omega network but providing dual source-sink paths.

Both networks use a mixture of circuit- and packet-switching: A message, e.g. a read or write request, is pipelined across switch stages as if circuit-switched; but when blocked some or all of the message is queued within a switch stage much like packet-switching. Since messages are pipelined but complete paths need not be allocated or used, routing control can be localized at the switches, and throughput is robustly maintained with high traffic levels.

The PME to network routing paths form a bipartite graph, i.e., all network messages are transmitted between processors and memories. Direct processor-to-processor paths do not exist. The PMEs are physically connected to the networks via 4x2 concentrators and 2x4 deconcentrators; these effectively multiplex each network port among four PMEs. One output of the concentrators routes to the low latency network, while the other output port routes to the combining network. The port chosen is set by the type of request: In the usual mode of operation, synchronization operations (e.g., fetch-and-OP) are sent to the combining network and all other requests (e.g., cache line loads, loads, stores) are sent to the low latency network. Configuration controls allow all messages to be sent through either network for experiments.

The low latency network has 128 ports and is constructed with four levels of 4x4 switches. The network provides dual paths between each source and sink. Two such dual path networks are actually used, one for requests (processors to memories), and one for replies (memories to processors). This both provides increased bandwidth and avoids the possibility of deadlock. It is constructed of high-speed bipolar logic; time of flight delay between switches equals logic delay in a switch. 50M bytes/second is attained on each connection, so the total bandwidth is 12.8G bytes/second (128 connections each for request and response). The data path is one byte wide, with parity, and approximately 500 nsec. is required for a message with 8 bytes of data to traverse it (assuming no contention). It uses a single gate-array chip, replicated voluminously. The concentrators and deconcentrators, constructed from that same chip, have similar characteristics. The switch design is conservative: without altering the underlying technology, it may be possible to increase the bandwidth by a factor of 2 and decrease the latency by a factor of 4.

The combining network is a 64 port network constructed from six levels of 2x2 switches. To route into the combining network a second level of 4x2 and 2x4

[1] By "hashing" we mean page-dependent one-to-one reordering of addresses within a page.

(de)concentrators are used (the 4x2 concentration provides a level of redundancy in the network). The switches are constructed from high-density NMOS logic. Each stage can combine queued references to identical memory locations, and includes an ALU to implement "fetch-and-OP" operations. The response network is integrated with the request network, as required by combining. A wider data path makes its peak bandwidth the same as the low latency core switch; but its total latency is much higher, due to slower logic and more switch stages. This does not produce proportionally increased memory access time, since the switch is only one of several elements involved in latency, and the other elements do not change (e.g., concentrators, memory access time, etc.).

A single low-latency switch would be preferable to our current design. But logic fast enough to provide the required latency does not have high enough integration levels to make this practical at the present time.

4.4 I/O and Other Support

Support for I/O and performance monitoring, along with system initialization and configuration, is mediated by the I/O and Support Processors (ISPs). These are independently programmable machines, each containing 2-4 Mbytes of memory and the same processor used in the PMEs.

As shown in Figure 4, each ISP supports eight processor/memory elements through a bus attachment that is independent of the network. Each ISP contains the interface necessary to drive a standard S/370 channel (OEM interface) connected to device controllers. It is currently planned to support only disk storage units (DASD) and channel-to-channel adapters directly from RP3. The DASD will be shared with a companion host S/370 to provide a path for bulk data transfer to and from RP3. The channel-to-channel adapter(s) will also be used for S/370 communication, allowing either side to interrupt the other and providing fast transmission of smaller amounts of data (e.g., terminal I/O). Through multiple interfaces provided in each device controllers, multiple ISPs will have access to each string of DASD.

Through the ISP/PME bus, the ISPs also have access to the LSSD chains [2] of each PME. These are shift registers that chain through every bit of each PME's state, allowing for complete initialization, configuration control, and in-place diagnosis of the PMEs.

In addition to the connections shown in Figure 4, groups of eight ISPs are connected to IBM PC/ATs which bring up and diagnose the ISPs themselves and perform the functions of a system console. The eight PC/ATs of a full RP3 configuration will themselves be connected by a local area network.

The ISP/PME bus also provides for collection by the ISP of performance data captured in each PME and in the combining sub-network. From the ISP collection points, this data can be transferred to the host S/370, or to dedicated I/O devices, or back into the RP3 PMEs themselves. Alternatively, it can be partly processed and sent to the PC/AT consoles to provide real-time performance monitoring displays.

5.0 Software Support

RP3's initial software support can be directly used for applications by experienced programmers; but its primary purpose is to serve as infrastructure supporting creation and evaluation of a variety of computational models.

BSD 4.2 UNIX[2] will be extensively modified internally by members of the NYU Ultracomputer project to provide a familiar, popular operating system for RP3. Most modifications will be user-invisible; they consist of replacing many internal OS algorithms with serialization-free equivalents developed at NYU, e.g., enqueue/dequeue operations for scheduling [4,5]. While the system will do dynamic load balancing, a user will optionally be able to "lock" processes into processors to do application-dependent static (or dynamic) load balancing.

User-visible modifications will include inter-process shared memory; load- and/or run-time use of distributed and local memory; and a "spawn" primitive, analogous to UNIX' "fork," allowing simultaneous creation of many processes without having that operation cause a serial bottleneck.

Programming languages available on RP3 will initially include C, FORTRAN, and possibly PASCAL. These will be extended to allow data declared as distributed vs. local and shared vs. private. Some simple parallel constructs (e.g., "parallel DO loops") will be initially provided by preprocessing to simplify programs creation; the repertoire of such constructs is expected to grow as our experience with the needs of parallel applications increases.

Prior to RP3 hardware availability, however, applications can be written and debugged using an experimental multi-processor environment called VM/EPEX. This uses standard facilities of the VM operating system to mimic RP3's local/global memory facilities. Since it runs code in native mode, it provides real speedup for multiprocessor S/370s. Used with many more processes than there are processors, it allows "virtual" speedup measurements for much higher degrees of parallelism. Source code compatibility between VM/EPEX and RP3 will be maintained.

6.0 Shared Memory and Mixed Technology

As previously noted, RP3's use of mixed technology in support of shared memory is potentially a controversial issue. Two points can be raised in this area: First, is shared memory an appropriate paradigm for parallel processing? Second, can the FET/bipolar speed differential utilized be maintained with expected advances in the underlying technology?

[2] UNIX is a trademark of Bell Laboratories and AT&T.

The difficulties of programming a global shared-memory machine are well known. However, the use of underlying shared-memory hardware must be distinguished from the use of higher-level paradigms for programming. It is abundantly clear that very stringent programming discipline must be used to avoid interminable difficulties in parallel programming. The difficulty is deciding which of a number of proposed disciplines is effective not only in avoiding problems but also in expressing and utilizing the concurrency we would like to exploit. In this regard, shared memory — and, in RP3 the ability to avoid using it — provides a blank slate on which a variety of disciplines can be implemented and experimented with in a non-trivial context.

Regarding underlying technology, it may be the case that physical extrapolations to ever-smaller feature sizes indicate that the many of the characteristics of FET and bipolar transistors will converge [15]. However, as far as we can extrapolate with a good degree of confidence, there are several mitigating factors:

1. The majority of the delay in accessing RP3's global memory is not due to the bipolar network, but rather to the FET components, including the memory itself. We have in fact verified that the *next* generation of FET components can be used with the *current* bipolar network design with an efficiency virtually identical to than of the current design.
2. The majority of the size and power used by RP3 is due not to the bipolar elements, but to the FET components. Higher levels of integration and additional I/O pins in the FET components could result in major reductions in the total machine size, allowing physically closer placement of the network components. The packaging that could then be used — all of which exists now — would allow nearly a factor of two improvement in the current bipolar network latency, with the *current* bipolar technology. This was *not* taken into account in the analysis mentioned in point 1 above.
3. The above indicates that there is little difficulty extending the RP3 architecture to better technology in the near term. In the longer term, packaging must be taken into account. The network does not require extremely high levels of logic density, but does require high power levels and heat dissipation; the processors and memories, on the other hand, primarily require high logic density. The implied differences in packaging result in very different effective characteristics, even if the underlying silicon fabrication process is uniform.

It is worth noting that we would not have predicted the first two points above before going through the design process as far as we have. Indeed, our initial mechanical system layout was based on unrealistically optimistic assumptions about the size of the PMEs. It is our experience that the interaction between silicon and packaging technology is quite complex and can lead to radical alterations in the organization of a system; interpolation made on the basis of only one of these technologies is inherently questionable; and the current universal lack of experience in designing highly parallel systems makes anything less than *full-scale,* detailed design based on known, existing elements a highly uncertain enterprise.

7.0 Conclusion

RP3 provides a highly flexible vehicle for research into highly parallel processing. Its local/global memory organization provides for the effective implementation of a wide variety of program organizations and computational models. Its performance has been deliberately targeted high enough to warrant the effort needed to convert real applications to parallel form, thereby providing better evaluation of the system.

The performance evaluation and detailed physical and logical design already performed for the RP3 hardware have already born fruit in the form of unexpected results. E.g., the network required for a highly parallel shared memory machine need not be a very large fraction of the total machine cost or volume. We hope that many lessons can similarly be learned as we begin to put the RP3 to use.

8.0 Acknowledgements

We would like to thank the the following people who have contributed or are now contributing to the RP3 project: T. Agerwala, G. Almasi, J. Anthony, J. Brody, R. Bryant, M. Cassera, F. Darema-Rogers, M. Elliott, A. Gottlieb, W. Groh, J. Hall, Y. Hsu, R. Jackson, B. James, D. Jefferson, M. Kumar, B. Lau, D. Lee, H. Liberman, M. Malek, P. Meehan, A. Moretti, R. Mraz, E. Nowicki, D. Ostapko, M. Pullen, D. Rathi, T. Richardson, K. So, J. Stone, C. Tan, P. Teller, M. Thoennes, M. Tsao, F. Tsui, S. Wakefield, and M. Wong. We would also like to acknowledge the contributions of others in the Data Systems, General Technology, System Technology, and Federal Systems Divisions of IBM; and the Ultracomputer Project of NYU.

And special thanks to Jennifer Hall, who has made it much easier for all of us to get this job done.

References

[1] Blodgett, A. J. and Barbour, D. R., "Thermal Conduction Module: A High-Performance Multilayer Cermic Package," *IBM J. of Research and Development,* vol 26 No. 1, January, 1982, pp.30-36.

[2] Eichelberger, E. B. and Williams, T. W., "A Logic Design Structure for LSI Testability," *Proc. of the Fourteenth Design Automation Conference,* New Orleans, LA, 1977, pp. 462-468.

[3] Goke, L.R., and Lipovsky, G.J., "Banyan networks for partitioning multiprocessor systems," *Proceedings*

First Annual Symposium on Computer Architecture, 1973, pp. 21-28

[4] Gottleib, Allan, Grishman, R., Kruskal, C. P., McAuliffe, K. P., Rudolph, L., and Snir, Marc, "The NYU Ultracomputer — Designing an MIMD Shared Memory Parallel Computer", *IEEE Trans. on Computers,* February 1983, pp.175-189.

[5] Gottlieb, A., B. D. Lubachevsky, L. Rudolph, "Coordinating Large Numbers of Processors", *ACM TOPLAS,* January 1982.

[6] Hennessy, J. L., Jouppi, N., Baskett, F., and Gill, J., "MIPS: A VLSI Processor Architecture," *Proc. CMU Conference on VLSI Systems and Computations,* October, 1981.

[7] Hennessy, J. L., Jouppi, N., Baskett, F., and Gill, J., "Hardware/Software Tradeoffs for Increased Performance," *Proc. Architectural Support for Programming Languages and Operating Systems,* March, 1982.

[8] Lawrie, D., "Access and Alignment of Data in an Array Processor", *IEEE Trans. on Computers,* No. C-24, 1975, pp. 1145-1155.

[9] McAuliffe, K.P., Brantley, W.C., and Weiss, J., "The RP3 Processor/Memory Element," *Proc. of the 1985 International Conference on Parallel Processing,,* to appear.

[10] Norton, V.A., and Pfister, G.F., "A Methodology for Predicting Multiprocessor Performance," *Proc. of the 1985 International Conference on Parallel Processing,,* to appear.

[11] Patterson, D. A. and Sequin, C. H., "RISC-I: A Reduced Instruction Set VLSI Computer Architecture," *Proc of the Eighth Annual Symposium on Computer Architecture,* Minneapolis, Minn., May, 1981.

[12] Pfister, G.F., and Norton, V.A., "'Hot Spot' Contention and Combining in Multistage Interconnection Networks,", *Proc. of the 1985 International Conference on Parallel Processing,,* to appear.

[13] Radin, G., "The 801 Minicomputer," *IBM Journal of Research and Development,* Vol. 27, No. 3, May 1983, pp. 237-246.

[14] Seitz, C.L., "The Cosmic Cube," *Communications of the ACM,,* Vol. 28 No. 1, January 1985, pp. 22-33.

[15] Seitz, C.L., California Institute of Technology, private communication

[16] Yen, W.C., Yen, D.W.L., and Fu, K.-S., "Data Conerence Problem in a Multicache System," *IEEE Trans. on Computers,,* Vol. C-34, No.1, January 1985, pp. 56-65.

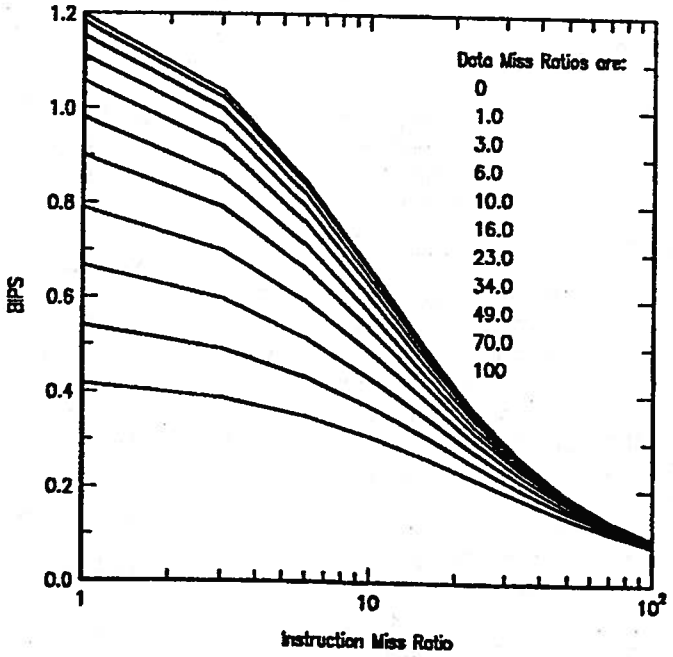

Figure 1. Performance of RP3 as a Function of Cache Hit Ratios: Instruction hit ratios are shown on the lower axis. Each curve represents a different data hit ratio. Starting from the top curve, data miss ratios are 0%, 1%, 3%, 6%, 10%, 23%, 34%, 49%, 70%, and 100%.

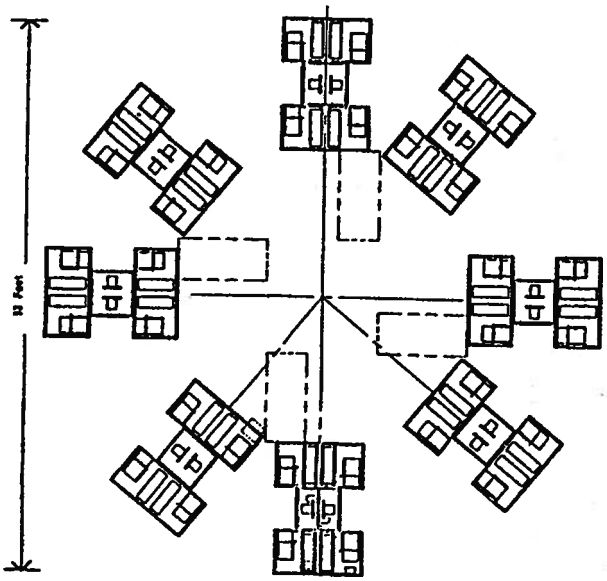

Figure 2. Floor Plan of a Full-Scale (512-way) RP3.

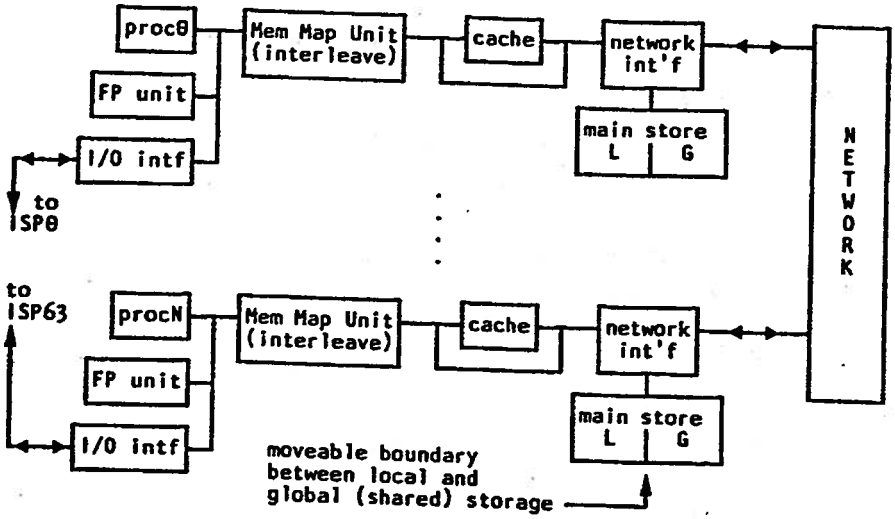

Figure 3. RP3 Processing Architecture

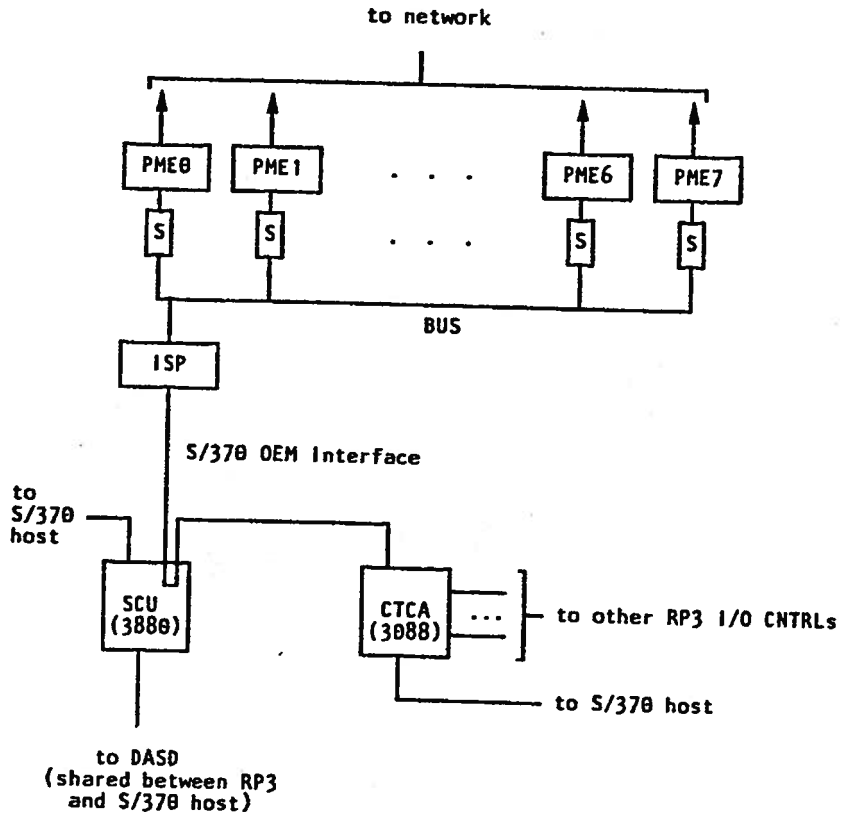

Figure 4. RP3 I/O Architecture

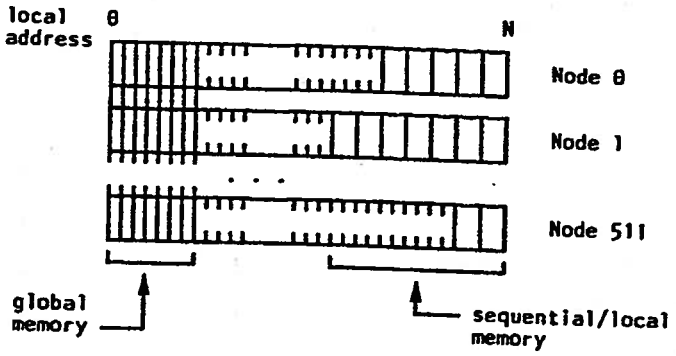

Figure 5. Typical "mixed mode" addressing in RP3