Proposal for a High-Performance, Scalable Multiprocessor [1]

José Eduardo Moreira
João Antônio Zuffo
Sérgio Takeo Kofuji

moreira@csrd.uiuc.edu
Center for Supercomputing Research and Development
University of Illinois at Urbana-Champaign
465 CSRL, 1308W Main St
Urbana, IL 61801
phone: 217-244-0049
fax: 217-244-1351

Laboratório de Sistemas Integráveis
Departamento de Engenharia Eletrônica
Escola Politécnica da Universidade de São Paulo
São Paulo, SP - Brazil

ABSTRACT

This report describes a proposed architecture for a massively parallel, shared memory computer, using commercially available microprocessors (DEC Alphas). The architecture is scalable from 4 processors to 65536 processors. Each processor has a peak performance of 200Mflops, and therefore this proposed architecture allows us to build systems with peak performance greater than 10 Tflops. Four processors are organized as a tightly coupled multiprocessing cluster. Multiple clusters (up to 8) can share a host, and multiple hosts can be used to build larger systems. Clusters communicate by means of a scalable interconnection network, that allows processors from one cluster to access the memory in a different cluster, thus providing a single memory space. Coherent caches are used to both reduce and hide the latency of inter-cluster communication.

[1]This work was supported in part by the National Science Foundation under grant NSF CCR 89-57310, the Department of Energy under grant DOE DE-FG02-85ER25001, FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo), and the Brazilian agencies Finep and CNPq. José Moreira is with Center for Supercomputing Research and Development and Laboratório de Sistemas Integráveis. João Zuffo and Sergio Kofuji are with Laboratório de Sistemas Integráveis.

1 Introduction

Despite all the recent progress in high-performance computing, with the introduction of several machines with performance in the 10 Gflops range (Cray YMP C90, Thinking Machines' CM-5, NEC SX-3, Hitachi S-3800, Intel Paragon), there is still need for even higher computational speeds (in the Teraflops range), in order to investigate the class of scientific and engineering problems known as "Grand Challenges" [9]. The "Grand Challenges" will play a strategic role in the ability of a nation to compete in the future global market, and include, among many others, weather prediction, semiconductor design, oil and gas recovery and drug design.

Current massively parallel machines have already promised to break the Teraflops barrier, but have been unable to fulfill that promise, in part due to the difficulty of programming distributed memory machines. Cray computer has already announced (to be introduced late 93, early 94) a shared memory massively parallel machine (Cray T3D), that might bring us one step closer to the Teraflops performance (peak performance for such a machine will be 200-400 Gflops). Fujitsu has announced the VPP500, with a peak performance of 350 Gflops.

A multiprocessor system aimed at achieving a computation speed of 1 Teraflops of better, should have the following characteristics:

1. Use thousands of commercially available, powerful multiprocessors. Traditional supercomputer companies have already announced massively parallel systems based on commercial multiprocessors (Convex with the HP-PA Risc, Cray with the DEC Alpha, Thinking Machines with the Sparc). The development cost of a new processor, and the speed of these new microprocessors, make it infeasible to custom design a processor.

2. Present the programmer with a single shared memory space. This offers an already established programming model, and facilitates the implementation of the many paradigms for parallel processing. The access time for this memory will most likely be non-uniform (NUMA machines), since the memory will be physically distributed with the processors.

3. Use a high-performance memory network between the processors and memory. Since this is a shared memory machine, the traffic between processors and non-local memory is very intense, and the network must be able to handle all the traffic, with acceptable latency.

4. It should be able to both reduce and hide the latency of accesses to non-local memory. For a very large machine, the latency in accessing memory located far away in the machine can be very high, compared to accessing local memory. It is important that most accesses be

performed with an effective latency as small as possible.

5. Offer scalability of processors, memory and interconnection network. An ideal massively parallel machine can start with just a few processors, and then be upgraded to more and more processors as the applications demand more computing power.

This paper describes an architecture that has all of the above characteristics, and therefore is a good candidate for a massively parallel machine. The paper is organized as follows: we first describe our choice of microprocessor; then we proceed by describing the cluster, the building block of the system; we then describe the interconnection network and the cache coherence mechanism; we discuss some system issues and finally present some conclusions.

2 The Microprocessor

The commercial microprocessor chosen for this architecture is the DEC Alpha [3]. There are other 64 bit processors already available (HP-PA Risc, R4400) and others coming (Sparc V9), and any of them would be a good choice for a massively parallel machine. The HP-PA Risc is the closest competitor to the Alpha, in performance. A 64-bit address space is fundamental to allow a single address space for the whole machine. The reasons for choosing the Alpha in particular are the following:

1. **Very high performance available now:** DEC is already selling a 200MHz (200Mflops) version of the Alpha. The Sparc V9 is not yet available.
2. **Second source availability:** A large chip manufacturer will second source the Alpha. The HP-PA Risc is, so far, only available through Hewlett-Packard.
3. **On chip first-level cache:** The HP-PA Risc has an off-chip first level cache, and therefore is more difficult to use in a design.
4. **Easy interface:** For a 200MHz Alpha, the external bus can run at 100, 50 or 25MHz, facilitating the interface with other subsystems.
5. **True 64-bit architecture:** The Alpha was designed from the start as a 64-bit architecture. All the other 64-bit microprocessors are evolutions of 32-bit designs.
6. **Support for multiple instruction issue and multiprocessing:** The execution model used by Alpha facilitates multiple instruction issue and shared memory multiprocessing.

7. **Upgradability:** the Alpha architecture was designed to last 25 years, with a predicted performance improvement of 1000 times during this time period. This makes the Alpha architecture a safe investment.

8. **Software availability:** DEC is an active member of OSF (Open Software Foundation), which guarantees a great availability of software (operating systems, compilers, applications) for the Alpha.

Table 1 is an excerpt from [7], and it compares the (single processor) performance of several high-performance machines, in the HEP benchmarks. These are scientific Fortran codes, mostly scalar. The results show how today's 64-bit microprocessors offer scalar performance greater than that of the most expensive supercomputers.

3 The Cluster

According to [2], 90% of the supercomputer cycles in an institution such as Lawrence Livermore National Laboratory are expended in *capacity* mode, and 10% are expended in *capability* mode. Capacity problems can typically consume a few hours of CPU, and require six million words of memory or less, while capability mode requires hundreds of hours of CPU time, and many tens of millions of words of memory.

From the above remarks, we conclude that a useful massively parallel system must be able to offer high single processor performance to hundreds or thousands of users (to tackle the capacity problems), as well as offer high multiple processor performance to a few users (to handle the capability problems). The ability to serve a multitude of users in capacity mode is one of the great advantages of MPPs as compared to traditional multiprocessors. Since there are hundreds or thousands of processors on an MPP, a new user can get exclusive access to one or a few processors, without degrading the performance as seen from other users.

In this architecture proposal, 4 microprocessors are grouped into a single-board cluster, that also contains memory, and the circuitry to connect to other clusters. A cluster is a tightly coupled multiprocessor in itself, and can be effectively used by serial or modestly parallel programs. The cluster is the building block of the MPP, as will be seen later, and the memory in the cluster can be accessed by any processor inside the cluster, as well as by processors from another cluster. The collection of memories from all cluster form the global shared memory of the machine.

Figure 1 is a block diagram of the cluster. Each of the 4 processors has its own second level cache of

512kB (the first level cache, of 16kB, is built into the microprocessor). The cluster is built around a 128-bit, 20ns, synchronous cluster bus, that delivers an 800MB/s bandwidth. This is the same bus bandwidth DEC uses in its high-end multiprocessor product (DEC 10000 AXP), to connect up to 6 Alphas.

The cluster memory is always accessed through the cluster bus, and whenever the caches want to access the cluster local memory, or some other cluster's memory, they must go through the cluster bus. In order to reach another cluster, a memory access passes to the *link* shown in the block diagram. This connects the cluster to the interconnection network, and therefore to other clusters. Also connected to the bus is a *directory*. This is responsible for the system's cache coherence, both inter-cluster and intra-cluster.

Also part of the cluster is a *4x4 crossbar switch*, used to build the interconnection network. This will be discussed in more detail in the next section.

The cluster is connected to a host machine, which performs the input/output operations, and where the major part of the operating system is run, thus allowing the processors in the cluster to concentrate on computation. The host suggested here is a SPARCcenter 2000 [10], and the cluster connects directly to the system bus of the host.

There are several good reasons for using the SPARCcenter 2000 as a host:

1. **Availability:** Sun Microsystems' products are the most popular workstations and servers available.
2. **Design for multiprocessing:** The SPARCcenter 2000 was designed to be a multiprocessor system, with up to 20 SPARC processors. Used as a host, it can be configured with 4 SPARC processors and 8 Alpha clusters (32 processors).
3. **Industry standard interfaces:** SBus, SCSI-2, Ethernet and serial interfaces are all standard in the SPARCcenter 2000. FDDI, token ring, HSI, faster SCSI, are all offered as an option.
4. **Fast system bus:** The dual 64-bit buses deliver a total bandwidth of 640MB/s, providing ample communication bandwidth between the host and the clusters.
5. **Large mass storage:** 36GB of disk space can be installed in the main cabinet. Up to 600GB can be installed using expansion cabinets.
6. **Small dimensions:** The main cabinet measures only 143cm H, 77cm W, 99cm D.

7. **Large board size;** The 9U boards used in the SPARCcenter 2000 offer plenty of space for implementing a 4-processor cluster.

Other machines could have been selected as hosts. The Silicon Graphics servers, in particular, fulfill all the performance requirements; however, they are not as popular and widely available as Sun's products.

3.1 The Second Level Cache

As stated before, each processor has a second level cache of 512kB. This cache is organized as a direct-mapped, 16k lines × 32 bytes/line cache. The internal, first level cache, also has 32 bytes/line, it is direct-mapped, and it is enforced to be a subset of the contents of the second level cache.

The interface between the processor and the second level cache is 128 bits wide, therefore it takes two transactions to transfer a cache line between the processor and the cache. Bits 5 to 18 of the processor's physical address are used to select the cache line, and the least significant bits are used to select the appropriate word in the line. Bits 19 to 33 (the 15 most significant bits) are used as a tag, to verify the presence of the proper cache line.

The second level cache is connected to the 128-bit wide system bus. The unit of transfer between the second level cache and the local cluster memory or a remote cluster memory is always a cache line. Therefore it takes two bus transactions to accomplish one transfer.

A cache can have multiple pending writes and reads. The multiple pending reads are used to implement prefetch. A prefetch operation is initiated by the processor, through a prefetch instruction (implemented as a write to a specific memory location, with the address of the cache line to be prefetched as the data), and it does not block the processor, but it starts a read by the cache (if the data is not already in the cache). The access is performed, and the line is loaded into the cache. When the processor actually accesses the data, the prefetch has (hopefully) already concluded, and the cache supplies the data in a very short amount of time.

The second level cache is responsible for providing two key features for massively parallel machines: latency reduction and latency hiding. The cache provides latency reduction by keeping the data that the processor uses most frequently very close to the processor, and it provides latency hiding by performing prefetch of data that the processor will need in the future, while the processor is doing something else.

3.2 The Cluster Memory

Each cluster has a local memory, that can be accessed by the 4 processors in the cluster directly through the cluster bus, and by processors in different clusters through the link (and through the cluster bus). The collection of all cluster memories is the system global shared memory.

The cluster memory is 128 bits wide (same width as the bus). Since all transfers are 256 bits (32 bytes) wide, nibble mode or page mode is used to speed up the second transaction of a pair.

The memory can use 4 or 16 Mbit memory chips, for a total size of 64 to 256 Mbytes. The memory will also accept 64 Mbit chips (the next generation of memory chips), for a maximum size of 1 Gbyte.

All the memory of the system will be organized as a single physical address space of up to 16 Tbytes, with 44 bits of address necessary to identify any single byte. These 44 bit are divided as follows: 14 bits select a cluster in the system (maximum of 16384 clusters), while 30 bits are used to select a memory position within a cluster (maximum of 1 Gbyte/cluster).

Since the current generation of Alphas only output 34 bits of physical address, we have to provide a method for mapping 34 bits into 44. We do that by using a table of 64 entries of 16 bit each. The 6 most significant bits of a physical address (bits 28 to 33) are used to access an entry of this table, and the 16 bits in the entry are concatenated to the 28 least significant bits of the address to form one 44 bit address. This table can be implemented on a per cluster, or per processor basis, and can be read and written by the processors, giving to any processor simultaneous access to up to 64 clusters, without having to modify the table.

3.3 The Link

The link is used to interconnect multiple clusters, both inside the same host and across multiple host. It implements 8 unidirectional (4 in, 4 out) high-speed optical serial channels, with a bandwidth of 1.5 Gbit/s each, for a total, unidirectional, bandwidth of 6 Gbit/s. That is 750 Mbyte/s, or almost the bandwidth of the cluster bus. All this bandwidth is necessary though, because we will be using these links to interconnect hundreds and even thousands of clusters.

The unit of transfer through the link is also one cache line (32 bytes, 256 bits), and the link can read and write the cluster memory, as well as access the directory (this is necessary in order to enforce cache coherence, as it will be seen later).

Due to the structure of the interconnection network (described in the next section), the link may

receive, in one of the input channels, a message that is not destined to this cluster, and it must be able to forward this message, through one of the output channels, to the appropriate cluster.

The 4×4 crossbar switch will be discussed in section 4 (Interconnection Network), while the directory will be discussed in section 5 (Cache Coherence).

4 The Interconnection Network

The interconnection network for a massively parallel machine must have some key properties: it must be scalable in size, cost and performance, and it must be practical to build and use. The function of such a network is to provide a link between any processor and any memory module in the system. Since in our system, the processors are organized as tightly coupled clusters, the function of the network is to provide a link between processors and memories in different clusters.

Many different organizations have been proposed and used for processor-memory interconnection networks. At the low and high ends of the cost-performance spectrum are the time-shared bus and the full crossbar switch, respectively. Neither of these two extremes are practical to use in a massively parallel computer. Multistage interconnection networks offer an interesting compromise in cost-performance, but they are not, in general, scalable in size, since the ratio of processors/switches varies with the number of processors. Multistage networks with a more scalable behavior have been proposed [8], but they are difficult to build.

For this massively parallel machine, we are proposing the use of a single stage (recirculating) shuffle-exchange network [5], built with 4×4 crossbar switches. A single stage shuffle-exchange network can be used to interconnect N clusters using only $N/4$ 4×4 switches. Figure 2 is an example for 16 clusters.

A message (memory access) from one cluster to another may have to go through the network several times (up to $\log_4 N$ times), before reaching the destination. For the network in Figure 2, suppose cluster 0 wants to send a message to cluster 8. The message has to go through the network once to reach cluster 2, and once again to reach cluster 8.

The maximum number of switches that a message has to traverse is no larger than in the case of a multistage Omega network (it is always $\log_4 N$ for the Omega network), but the bandwidth of the single stage is less than that of the Omega by a factor of $\log_4 N$ (since that is the number of stages in the Omega network). We can compensate that problem (up to a certain point) by using multiple single stage networks in parallel.

Notice that a single stage shuffle-exchange uses $N/4$ crossbar switches to interconnect N clusters. Since each cluster has one 4×4 switch, we can build 4 single stage interconnection networks in parallel. This arrangement works for any number of cluster that is a multiple of 4 (routing is trivial for powers of 4 only, but it can be implemented efficiently for any multiple of 4), and it actually offers the same or more bandwidth than a multistage Omega network for systems with up to 256 clusters (1024 processors). In an Omega network, the ratio of processors to switches varies with the size of the network, which is a scalability disadvantage.

4.1 Routing

A single stage shuffle-exchange network is composed by a *shuffle* substage and an *exchange* substage. The shuffle substage is simply a reordering of the inputs, obtained by applying the shuffle function of radix 4 ($S_4(i)$), defined as follows: let $i = a_{n-1}a_{n-2}\ldots a_1a_0$ be an input number in 4-radix representation (two bits for each digit), then

$$S_4(a_{n-1}a_{n-2}\ldots a_1a_0) = a_{n-2}\ldots a_1a_0a_{n-1}$$

This corresponds to a cyclic shifting of the number. The exchange substage consists of $N/4$ 4×4 crossbar switches. The first crossbar is connected to the first 4 (shuffled) inputs and outputs, the second crossbar to the next 4 inputs and outputs and so on.

The routing information can be represented as a sequence of 4-ary digits: $r_{n-1}\ldots r_0$, where n is the number of times that a message has to go through the network (we can force n to be always $\log_4 N$). The digit r_i is used to select the output of a switch during the $i-th$ passage of the message through the shuffle-exchange. Each time a message has to go through a shuffle exchange, it can select any of the 4 networks, since they all implement the same interconnection.

If N is a power of 4, then $r_{n-1}\ldots r_0$ is simply the 4-ary number of the destination cluster, with the digits in reversed order. If N is not a power of 4, then a table lookup is necessary to generate the routing information from the destination address.

4.2 The Switch

The 4×4 crossbar switch must be able to recognize the routing information in a message, and send it to the appropriate output. It must also be able to handle the appropriate bandwidth. The inputs and outputs of a switch are connected to the 1.5 Gbit/s serial links of different clusters. Therefore the total bandwidth through a switch is 6 Gbit/s.

Such a switch can be implemented either as a narrow, very high speed switch (say 8 bits wide, at 200 MHz), or an wide, medium speed switch (say 32 bits wide, at 50 MHz). The choice depends on the technology available.

5 Cache Coherence

There are two levels of cache coherence in the proposed architecture: intra-cluster and inter-cluster. Both levels of cache coherence are guaranteed through information stored in the cluster directory. The intra-cluster cache coherence is based on a centralized directory scheme. The inter-cluster cache coherence is based on the Scalable Coherent Interface (SCI, IEEE Standard P1596 [6]), a distributed directory scheme.

The cluster directory contains information regarding each cache line stored in the four processor caches. For each cache line, the directory keeps information about which of the four processors in the cluster have that line in their second level cache, and it has pointers to the next and previous clusters that also have that line cached. For each cached line, all the clusters that have that line in cache form a double linked list.

When one of the processors in the cluster wants to write a cache line for the first time, it must notify the directory. It will then invalidate that cache line in the other processors inside the cluster, as well as send invalidation messages to the (up to) two clusters that it knows also share the line. When a cluster receives an invalidation message from another cluster, it must pass the invalidation down the list, and invalidate its caches.

When a processor caches a line for the first time (among the processors in the same cluster), the cluster directory must insert itself into the linked list of clusters sharing that line. Therefore each line in memory must have a tag identifying the first cluster in the list of clusters sharing that line. Correspondingly, when a cache line is replaced, and there are no more copies of that line in the cluster, the cluster directory must disconnect itself from the linked list.

The above inter-cluster cache coherence scheme may seem inefficient, since an invalidation has to serially propagate through all the clusters that are sharing a given cache line. Cache studies, however, indicate that most sharing is among a few processors [1, 4], and therefore the SCI scheme can be efficient.

6 System Size

The architecture proposed in this report supports up to 16384 clusters, or 65536 processor. The limitation is imposed not by the interconnection network, but by the memory addressing and routing. The 200MHz Alpha has a peak performance of 200Mflops, and therefore the maximum peak performance of this architecture is 13 Tflops.

With 1024 clusters (4096 processors) we can come very close to the 1 Tflops barrier (820 Gflops). Such system would require 128 hosts (8 clusters/host), that would fill a 20-meter square room (allowing $3m^2$ for each host), and draw approximately 700kW of electrical power. While these numbers are impressive, they are similar to the current requirements of supercomputer centers.

7 Conclusion

We have briefly described a scalable architecture that can be used to implement a shared memory massively parallel computer with thousands of processors, and peak performance in excess of 10 Tflops. This architecture is based on powerful, commercially available microprocessors, and represents a feasible alternative to Teraflop computing.

Some more detailing of the description, and some performance studies to validate the approach, are necessary before we can proceed with the actual design.

Machine	GM	Range
SGI R4000 50MHz Irix f77 -O4 -mips2 dyn 10/92	13.60	12.60 - 14.50
NEC SX-3 Super-UX f77sx scalar 10/91	15.20	13.20 - 18.50
IBM RISC6000/970 50MHz -O3 hssngl dyn 10/92	15.30	12.80 - 23.60
CRAY C90 (4.16ns) UNIX CFT77 vector 10/92	17.10	13.90 - 23.00
DEC10000AXP/610 (5.0ns) VMS FTN T3.3 11/92	26.00	22.10 - 35.90
HP9000/735 99MHz f77 +)3 -archive 11/92	27.20	20.40 - 34.00

Table 1: This table compares the uniprocessor performance of microprocessor based machines (SGI R4000, IBM RISC6000, DEC10000/AXP and HP9000/735) with that of conventional supercomputers (NEC SX-3, CRAY C90), for mostly scalar scientific codes. The performances are relative to the VAX8600. The GM column is the geometric mean for all benchmarks, while the Range column gives the minimum and maximum performance observed for the different benchmarks. If the results for the SGI R4000 are scaled for the 75MHz R4400, we get a GM of 20.4, putting it ahead of the NEC and CRAY.

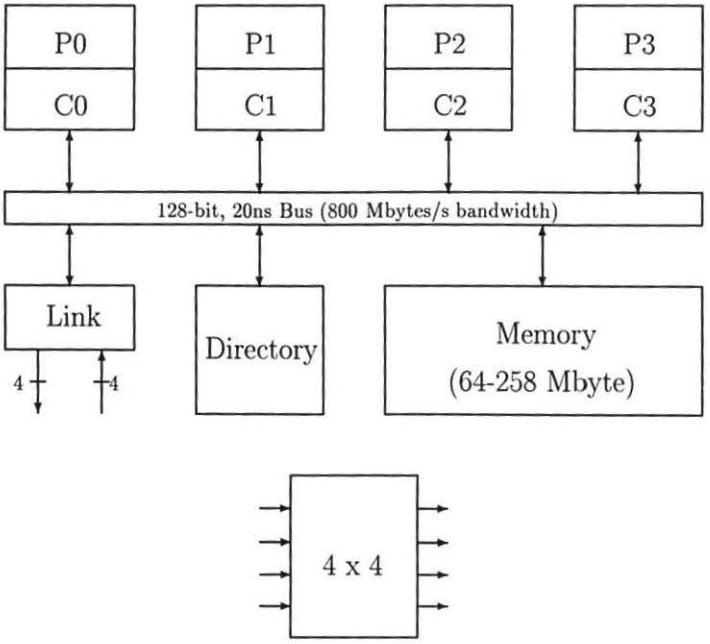

Figure 1: Block diagram of the 4-processors cluster.

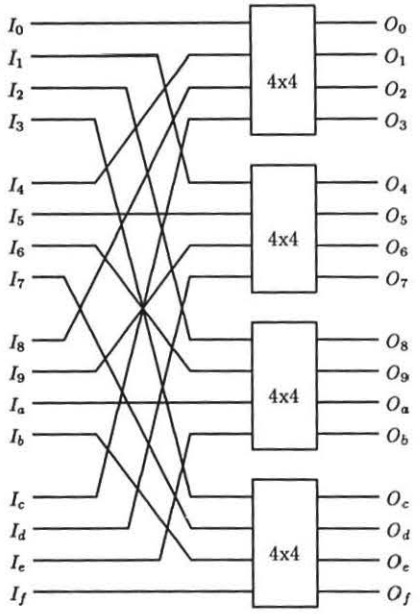

Figure 2: 16 clusters interconnected through a single stage shuffle-exchange network (network input I_x and output O_x are connected to cluster x). Up to two passes ($\log_4 16$) through the network may be necessary to deliver a message from one cluster to another.

References

[1] Anant Agarwal, Richard Simont, John Hennessy, and Mark Horowitz. An Evaluation of Directory Schemes for Cache Coherence. In *Proceedings of the 15th Annual International Symposium on Computer Architecture*, Honolulu, Hawaii, 1988.

[2] Eugene D. Brooks. Massively Parallel Computing. In *The 1992 MPCI Yearly Report: Harnessing the Killer Micros. Lawrence Livermore National Laboratory*, August, 1992.

[3] Digital Equipment Corporation, Maynard, Massachusetts. *Alpha Architecture Handbook - Preliminary Edition*, 1992.

[4] Susan J. Eggers and Randy H. Katz. A Characterization of Sharing in Parallel Programs and Applications to Coherence Protocol Evaluation. In *Proceedings of the 15th Annual International Symposium on Computer Architecture*, Honolulu, Hawaii, 1988.

[5] Kai Hwang and Fayé A. Briggs. *Computer Architecture and Parallel Processing.* McGraw-Hill, 1984.

[6] David V. James. SCI (Scalable Coherent Interface) Cache Coherence. Cache and Interconnect Architectures in Multiprocessors, Dubois and Thakkar editors, 1989.

[7] Eric McIntosh. Benchmarking Computers for HEP. Technical report, CERN School of Computing. L'Aquila, Italy, 1992.

[8] José E. Moreira. Multiple Omega Networks for Parallel Processing. In *Proceedings of the IV Brazilian Symposium on Computer Architecture - High Performance Processing. São Paulo, Brazil*, October 26-29, 1992.

[9] The FY 1992 U.S. Research and Developmente Program. Grand Challenges: High Performance Computing and Communications, 1992. A Report by the Committee on Physical, Mathematical, and Engineering Sciences, Federal Coordinating Council for Science, Engineering, and Technology, Office of Science and Technology Policy.

[10] Sun Microsystems Computer Corporation, Mountain View, California. *SPARCcenter 2000 (sales brochure)*, 1992.