The Multiplus/Mulplix Project

Júlio S. Aude

NCE and IM/UFRJ
PO Box 2324 - Rio de Janeiro - RJ - 20001-970 - Brazil
e-mail: salek@nce.ufrj.br

ABSTRACT

The MULTIPLUS/MULPLIX project aims at the development of a modular distributed shared-memory parallel architecture able to support up to 1024 processing elements based on SPARC microprocessors and at the implementation of MULPLIX, a Unix-like operating system which provides a suitable parallel programming environment for the MULTIPLUS architecture. The project currently includes research effort in four areas: parallel architectures, operating systems, CMOS IC design and parallel programming environments. This paper firstly presents an overview of the MULTIPLUS architecture and briefly describes the implementation of its basic hardware modules. Secondly, developments in the area of CMOS IC designs for use within the MULTIPLUS architecture are presented. Following, the MULPLIX operating system is described and the parallel programming primitives available within MULPLIX are presented. The implementations of Alchemist, a visual parallel programming environment, and of M-PVM and Pthreads parallel programming libraries within the MULPLIX system are also discussed. Finally, the perspectives for future development of the project are presented.

1. INTRODUCTION

The MULTIPLUS project [Aude96] has been under development at NCE/UFRJ for some years now and has provided a nice and challenging framework for research work in several areas related to the world of High-Performance Computing: Parallel Architectures, Operating Systems, IC Design, Parallel Programming Environments and Parallel Algorithms. This short paper presents the current status of the project under development within the FINEP Academic PAD Program.

Section 2 reviews the main features of the MULTIPLUS distributed shared memory parallel architecture and the current implementation of its four basic hardware modules: the Processing Elements, the I/O Processor, the Multistage Interconnection Network and its Interface to each MULTIPLUS cluster of processors. Section 3 discusses the design of VLSI circuits for use within the MULTIPLUS architecture, including the development of the NCESPARC microprocessor. Section 4 describes the MULPLIX operating system and its parallel programming primitives. Section 5 briefly describes the implementation of two parallel programming libraries within MULPLIX: M-PVM and Pthreads and the implementation of Alchemist as a visual parallel programming environment. Finally, in Section 6, planned future developments are presented.

2. THE MULTIPLUS ARCHITECTURE

MULTIPLUS is a distributed shared-memory multiprocessor designed to have a modular architecture which is able to support up to 1024 processing elements and 32 Gbytes of global memory address space. Within MULTIPLUS, up to four processing
elements can be interconnected through a 64-bit double-bus system making up a cluster. The MULTIPLUS NUMA (Non-Uniform Memory Access) architecture supports up to 256 clusters interconnected through an inverted n-cube multistage network and uses a distributed I/O system architecture.

Design decisions have been taken to simplify the problem of maintaining consistency among the private caches of the processing elements within the MULTIPLUS architecture [Mesl92]. The first one is to have one cluster bus dedicated to instruction/data access and the other dedicated to block transfer operations. Only the instruction/data bus needs to be "snooped" by the cache controller and, as a result, the cache consistency problem can be solved within a cluster with the methods usually adopted in bus-based systems. In addition, a software approach is adopted to keep cache consistency between clusters.

The MULTIPLUS Processing Element is based on the use of SPARC processors. Its first implementation used the Cypress SPARC chipset and could support a 64-Kbyte cache and up to 32 Mbytes of memory belonging to the global address space. The new implementation of the Processing Element will be able to have up to 4 SuperSPARC II or HyperSPARC modules and to support up to 256 Mbytes of memory.

The I/O Processor [Oliv92] uses two CPUs. The first one manages the I/O requests sent by the Processing Elements to a dual-port Command Memory, performs the Disk Cache control and sends commands to be executed by the I/O devices through the Communication Memory. The second CPU controls the execution of the internal tasks issued by the first CPU. It controls a SCSI and a Parallel Interface, the Disk Cache, a DMA Controller which transfers data from the SCSI and Parallel Interface to the Disk Cache, and a BIFIFO temporary buffer for data to be transmitted between the Disk Cache and the Processing Elements.

The MULTIPLUS multistage interconnection network consists of 2x2 cross-bar switching elements with FIFO buffers assigned to each input [Bron96]. Separate networks are used to interconnect the instruction/data and the block transfer busses in different clusters. The communication paths between switching elements in the network are unidirectional and nine bits wide. The transmitted messages can be as long as 128 bytes. Wormhole routing is used and a single destination address bit is used by each network stage to route the message.

The Network Interface interconnects the cluster bus systems to the Multistage Interconnection Network and also performs the functions of bus arbiter and bus reset generation. It consists of two identical sections to deal with each cluster bus. In addition, it has a DMA Controller which is programmed through the instruction/data bus and performs data block transfers between clusters through the block transfer bus.

3. IC DESIGNS FOR THE MULTIPLUS ARCHITECTURE

The mainstream of the research efforts in the area of IC design within the MULTIPLUS/MULPLIX project is the design of NCESPARC [Barb90], a 32-bit RISC microprocessor, using CMOS 1.0u technology. The NCESPARC architecture follows the SPARC version 7.0 definition. The 32-bit Data Path [Aude96a] consists of a three-port Register File, an ALU, a Barrel Shifter and auxiliary registers.

The architecture is implemented as a four-stage pipeline: fetch; decoding and operand fetching; execution; and writing of the result in the register file. For each pipeline stage, there is an instruction register associated with it which stores the code of the instruction under processing at that stage. For each pipeline stage, a set of logic equations
describe the control logic which has been implemented using the standard-cell approach [Aude 95].

In addition, to the design of the NCESPAC microprocessor, a CMOS implementation of the MULTIPLUS bus arbiters [Barb96] has been performed using CMOS 1.0μm technology. This chip has been tested after fabrication and has performed according to the specifications.

4. THE MULPLIX OPERATING SYSTEM

MULPLIX [Azev93] is a UNIX-like operating system designed to support medium-grain parallelism and to provide an efficient environment for running parallel applications within MULTIPLUS. MULPLIX results from extensions to Plurix, an earlier Unix-like operating system developed to support multiprocessing within the Pegasus SMP architecture [Fall89].

Within MULTIPLUS, the operating system needs to support applications consisting of a large number of processes running in parallel, demanding synchronization and a lot of context switching operations. To solve this problem, the concept of thread has been introduced in the MULPLIX definition. Within MULPLIX, a parallel application consists of a process and its set of threads. When switching between threads of a same process, only the current processor context needs to be saved. Information on memory management and resource allocation is unique for the process and, therefore, remains unchanged in such operations. MULPLIX libraries have already been written to work safely within a multi-threaded environment [Barr96].

MULPLIX provides a set of primitives to deal with threads. The system call, "thr_spawn", creates a group of threads. The number of threads to be created, the name of the procedure to be executed by these threads and a common argument are its basic parameters. An optional parameter defines preferential processing elements for the execution of each thread. A second version of this system call, "thr spawns", allows the creation of threads in synchronous mode. Three additional primitives for thread control are also provided: "thr_id", which returns the identification number of a thread; "thr_kill", which allows any thread to kill another thread within the same process; and "thr_term", which allows a forced termination of the thread.

The MULPLIX memory management system worries about data locality and allows memory sharing between threads of the same process. The MULPLIX memory management system is also concerned with maintaining cache consistency between MULTIPLUS clusters. The memory allocation primitives perform shared ("meAlloc") and private data allocation ("mePalloc"). Process scheduling is another area that requires a special attention concerning data locality. Separate queues of threads which are ready to run are implemented in each cluster. Every queue can be accessed by any processor. However, a processor only looks for a thread to run in another cluster if it finds its own cluster queue empty.

In relation to synchronization, primitives for the manipulation of mutual exclusion and partial order semaphores are made available. Mutual exclusion primitives are provided for creating ("mx_create"), allocating ("mx_lock"), extinguishing ("mx_delete") and releasing ("mx_free") a semaphore. The primitive "mx_test" allocates a semaphore if it is free but does not keep a thread waiting if the semaphore is still occupied. Partial ordering semaphores, which implement barrier-type synchronization, are also supported through primitives for creating ("ev_create"), asynchronous signalling ("ev_signal"),
waiting on the event occurrence ("ev_wait"), synchronous signalling ("ev_swait") and extinguishing ("ev_delete") an event.

5. PARALLEL PROGRAMMING ENVIRONMENTS

In addition to the native MULPLIX parallel programming environment, three other environments have been implemented: M-PVM, Pthreads and Alchemist. M-PVM [Sant97] is an implementation of PVM which is not totally compatible with the standard PVM, but can provide higher performance within the MULTIPLUS/MULPLIX platform. Each PVM task is mapped onto a MULPLIX thread and the message passing functions are implemented using the MULPLIX shared memory among threads of the same process. M-PVM is in fact a hybrid environment which provides applications with efficient implementations of PVM message passing functions and with the possibility of using shared memory. M-PVM is currently available on Solaris, through the use of a library of MULPLIX primitives implemented on top of Solaris LWPs.

The implementation of Pthreads within the MULPLIX system aims at offering to the user a powerful multi-threaded parallel which simplifies the portability of parallel applications to the MULTIPLUS/MULPLIX platform. The current Pthreads implementation within the MULPLIX system [Barr97a] is running on Solaris SPARC stations and does not support the association of multiple user threads with a single MULPLIX thread.

Alchemist [Barr97b] is a visual programming environment for parallel software development. It is based on multithreaded programming and uses shared memory for communication. Alchemist is written in Java and works on meta-schemes of parallel programming models. Currently it can generate parallel C code for the MULPLIX native model, Pthreads and Solaris threads. Alchemist is currently available on Solaris platforms.

6. CURRENT STATUS AND PERSPECTIVES

An initial MULTIPLUS prototype with four Processing Elements organized into a single cluster is currently operational and has been demonstrated during the V Expociência in the FINEP boot at the 49th Annual Meeting of the SBPC that took place in Belo Horizonte from July, 8th to 13th. The prototype was running parallel SOR and was also playing Gomoku against the exhibition visitors. It played around 900 times and won 92% of the games. The prototype stayed up and running for about 44 hours without showing any problem. In this prototype, the MULPLIX operating systems was partially ported and operational.

The future developments planned within the MULTIPLUS/MULPLIX project are the following ones:

- Re-design of the Processing Element to support up to 4 SuperSPARC or HyperSPARC modules and 256 Mbytes of memory;
- Test of a MULTIPLUS prototype with two or more clusters and with network connection;
- Complete porting of the MULPLIX operating system to the MULTIPLUS platform;
- Design of NCESPARC+ [Aude97] with hardware support to multiple contexts of threads;
• Implementation of an optimized memory management system to support inter-cluster cache consistency;
• Optimization of the implementation of the synchronization primitives within MULPLIX;
• Implementation of an optimized process and thread scheduler for the MULPLIX system;
• Development of an optimized MPI library for the MULTIPLUS/MULPLIX platform;
• Optimization of the M-PVM implementation within the MULTIPLUS/MULPLIX platform;
• Research work in the areas of: combined hardware-and-software mechanisms for maintaining cache consistency; alternatives for hiding the latency of memory access operations through the Interconnection Network; Interconnection Networks with support to multicasting;
• Parallel algorithms in the areas of: robotics; CAD for VLSI; signal processing for geophysics; optimization;

7. ACKNOWLEDGEMENTS

The author would like to thank FINEP, CNPq, RHAE and FAPERJ for the support given to the development of this research work. The author would also like to thank the research team directly involved with the development of the MULTIPLUS/MULPLIX project: Alexandre M. Meslin, Alexandre M. Gomes, Cláudio M. P. Santos, Gerson Bronstein, Gladstone Moisés, Mário O. Barros, Márcio T. Young, Márcio A. S. Barbosa, Mario João Jr., Sidney de C. Oliveira

8. REFERENCES

[Barr97a] “Implementação do Padrão Pthreads para o Sistema Operacional Mulplix”, M.O.


