GFLOPS COMPUTER: AN IMAGE PROCESSING PARALLEL ARCHITECTURE

Dominique Houzet, Abdelkrim Fatni
IRIT-ENSEEIHT-INP,
2 rue Camichel, 31071 Toulouse, France
Tel: +33 61 58 83 18; fax: +33 61 58 82 09
houzet@enseeiht.fr

Abstract
The real-time Image and signal processing applications, such as vision, image synthesis, HDTV, signal processing, neural networks, require both computing and input/output power. The GFLOPS project is dedicated to the study of all the aspects concerning the design of such computers. Its aim is to develop a parallel architecture as well as its software environment to implement those applications efficiently. The proposed architecture supports up to 512 processor nodes, connected over a scalable and cost-effective network at a constant cost per node. GFLOPS-2 is a single-user machine which is designed to be used as a low-cost parallel co-processor board in a desk-top work station.

1 INTRODUCTION

Most recent MPP systems employ fast sequential microprocessors surrounded by a shell of communication and synchronization logic. The GFLOPS-2 computer [5] provides an elaborate shell to support global-memory access, prefetch, atomic operations, barriers, and block transfers. This shell is integrated within the processors to build a homogeneous tightly-coupled architecture. Based on this design, a simple but powerful parallel extension to the C language has been designed with the goal of extracting the full performance capability out of this machine. The basic approach is to provide a full C on each node, with a rich set of assignment operations on the collective global address space. With GFLOPS-2, the semantics of the hardware primitives for global operations are essentially at the same level as the language primitives. This work originates from the SYMPATI project. This project led to the joint development of the Line Processor SYMPATI-2 [1] developed by our laboratory and the CEA-LETI-Saclay Nuclear Agency. This Line Processor has been commercialized since 1993 by the CENTRALP company. SYMPATI-2 provides excellent performances as shown through our response to Preston's Abingdon Cross benchmark [2].

To address a wide array of applications, low latency communication is necessary in addition to high throughput. Latency of fine grain communication (such as a transfer of a few words) is especially important for good performances. Therefore, GFLOPS-2 provides 3 classes of architectural mechanisms that implement an automatic locality management strategy: globally shared memory, integrated message-passing and latency tolerance. This paper describes the experience gained by designing, fabricating, and running a complete parallel system. Section 2 describes the GFLOPS-2's implementation and shows how the mechanisms combine to produce a coherent system. Section 3 discusses related work on parallel architectures and finally, section 4 summarizes the insight gained from implementing GFLOPS-2.

2 THE GFLOPS-2 MACHINE

The GFLOPS-2 architecture is organized as shown in Figure 1. Memory is physically distributed over the processing nodes. Each processing node is identical, composed of two high-performance custom processors, a common floating point unit, a network module and an I/O interface. The network module forms the heart of the node, integrating the memory controller, the processor interface and the network interface. This integration allows for low hardware overhead while supporting high performance communication mechanisms. The hardwired data movement logic achieves low latency and high bandwidth by supporting highly-pipelined data transfers. 16 nodes form a first-level ring. A communication cell, on the right of the figure, is used to built a larger configuration with up to 512 processors organized as a multi-ring structure.

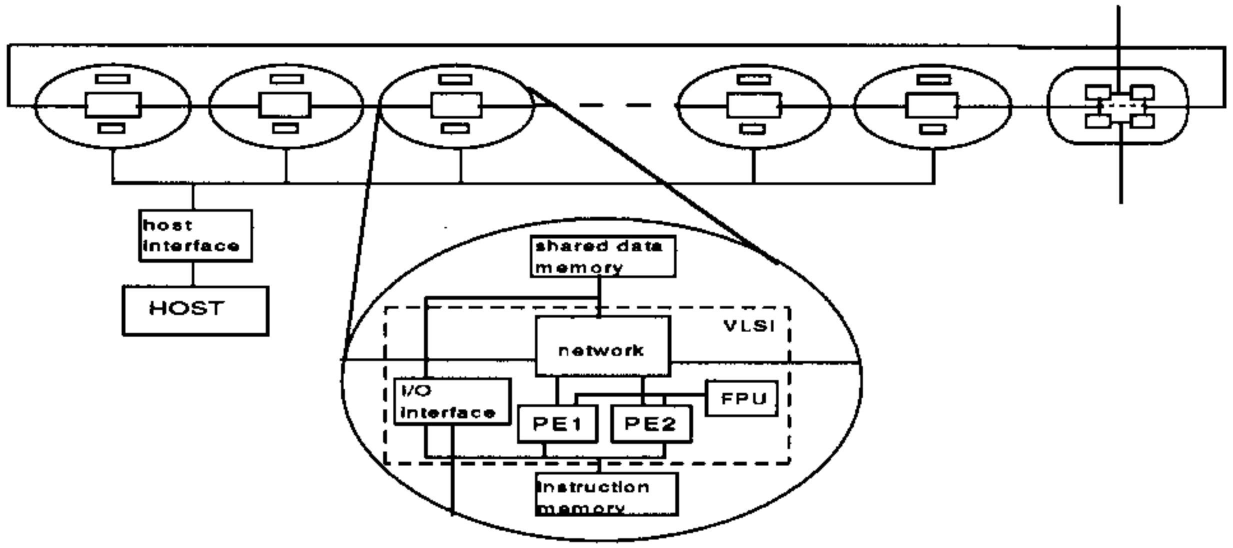

Figure 1. The GFLOPS-2 architecture.

2.1 GFLOPS-2 Processor

The aim of this architecture is to obtain supercomputing power, such as the CM-2 for integer operations, at low-cost. It is possible through the use of different levels of parallelism: coarse grain parallelism at the architecture level and fine grain parallelism at the processor level. We have designed a new custom-built processor structure to adapt the processor power and the network bandwidth. The latency and bandwidth of the network has been analysed to design the processor [14]. Also we have built a tightly coupled cluster in a single VLSI in order to obtain high performance. Each PE is a RISC/VLIW processor using 3 general integer units (3 32-bit ALUs) operating in parallel at each instruction performed in one clock cycle, and linked to a multi-ported register file. The instructions are 64-bit wide. Two processors share a common floating point unit (single precision +, -, *). The defined structure is such a structure that at each instruction, each PE can perform 3 arithmetic or logical operations, the access to the next instruction, an external memory and a local memory accesses. The local internal memory is used to store local static variables and the program stack. Thus, with a 33 Mhz clock rate, one PE leads to the peak computational rate of 165 Mega Instructions Per Second (MIPS). The power of a configuration with 32 PEs is about 5 GIPS.

Several new ALU operations and addressing modes have been introduced in the processor. These operations are specific to shared-memory multiprocessors. We have introduced them to improve the execution of codes generated by the C// [8] compiler. The first example is the following addressing mode (with "ri" the register number i):

```
ri=expr;
rj=load(symbol+(ri*4)/NPE);
```

The address obtained here is a scale of an index register "ri" divided by the number of processors (NPE), and added to a symbolic constante (the base address of an array); The C corresponding statement is:

```
var=symbole[expr/NPE];
```

This kind of addressing mode improves slightly the performances of the machine because most of the parallel computation is performed on parallel arrays. The second example is the 4-byte vector-operations introduced in the ALUs. Each 32-bit data word can thus be processed as a vector of 4 bytes simultaneously. Many typical basic Image processing algorithms can use these operations, with 4 grey-level pixels in 32-bit word or with one RGB pixel (3 components). Most of these regular pixel-level algorithms are improved by a factor of 3 to 4.

2.2 The GFLOPS-2 Network Module

Following SYMPATI-2, AIS-5000 [3], or KSR1 [12], a processor organization around a one-dimensional linearly expandable network has been defined. For many vector based algorithms used today, the 1D linear array architecture can be shown as one of the most effective parallel architecture. With this organization it is easy to increase the number of processors in order to adapt the performance of the structure according to the application requirements. The network we have designed is a multi-ring interconnection network. Each processor can have direct access to a subset of four memory banks without routing conflicts with other processors. Each processor can also communicate directly with its two neighboring processors through register transfer, implementing a systolic communication style as iWarp [6] does. This style of communication can be performed only with synchronous accesses for regular algorithms. The GFLOPS-2 processor provides an efficient and tight coupling between the processor pipeline and the communication network. All the network modules are connected to built a bidirectionnal ring with two FIFOs per module. Two processors are connected to each module to form a cluster. The distant communications are performed with atomic messages: one address word and one data word presented in figure 2. The messages use the FIFO in each network module to reach the target memory. The routing protocol used is a simple wormhole like protocol [13].

- remote write message: 2 words of 32 bits:

| type | N° target processor | target memory address |
| data to write |

- remote read message: 2 words of 32 bits:

| type | N° target processor | target memory address |
| | N° source processor |source memory address |

Figure 2. Messages format.

Although GFLOPS-2 provides the abstraction of globally shared memory to programmers, the system's physical memory is statically distributed over the nodes in the machine. On each node a network module fields memory requests from two processors and determines wether requests access local or remote memory. Thus for performance reasons much of the underlying software is implemented using message passing. The performances of all of the layers of software that help manage locality (including the compiler, libraries and run-time system) depend on an efficient communication mechanism. Features in the processor and the network module combine to provide a stream-lined interface for transmitting and receiving messages: both system and user code can quickly describe and atomically launch a message directly into the interconnection network; a direct memory access mechanism allows data to flow between the network and memory.

All synchronization capabilities are available with instructions such as "barrier", "test and increment" and a "wait" on the completion of all the distant accesses. Two hops in the ring are performed at each processor clock cycle. The bandwidth per network module is twice the processor bandwidth. Figure 3 presents the internal structure of the network module. The memory managment unit for both local and remote accesses and for the two external bus uses a fixed priority scheme to resolve memory conflicts. Remote read and write are non-blocking. A sync statement is used to perform a blocking remote memory access. A remote read, which is in fact an exchange of data between the target memory and the source memory, implements a prefetch controled by the program. The compiler automatically generates a temporary local variable to store the data read in the remote memory, and generates a "wait" for the completion of the read. When a remote memory read reaches the target network module, the read is performed and a remote write message is sent to the source memory bank. Overlapping of remote accesses (software prefetching) with treatment of data is performed at compile time for basic blocks and for loops. This mechanism attempts to tolerate the latency of inter-processor communication when it cannot be avoided. Prefetching allows code to anticipate communication by requesting data or locks before they are needed.

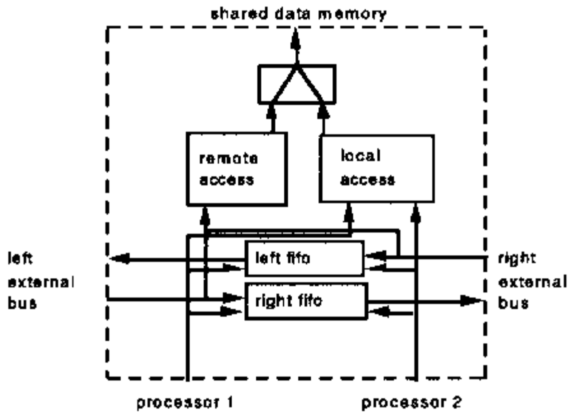

Figure 2. Block diagram of the network module.

2.3 Prototype

The first VLSI of the processor was realized to build a first prototype. This first prototype implementation, running since the beginning of 1995, demonstrates that a parallel system can be both scalable and programmable through the use of the virtually shared memory paradigm, physically implemented with atomic message passing. The second VLSI, with two processors, a floating point unit and a network module, has been designed in VHDL and will be available at the beginning of 1996. It is a CMOS 0.7 µm chip with 256 pins. A processing node is composed of one VLSI and three SRAM 256k*32 chips with 20 ns access time. The two inter-VLSI bus and data-memory bus are 32-bit wide. The instruction bus is 64-bit wide and both the data address and instruction address bus are 24-bit wide. The interface bus is 4-bit wide. Both the gate count of the network module and the floating point unit are nearly 15k gates each. Each processor gate count is nearly 35k gates. The clock frequency obtained is 33 Mhz. A ring board contains 16 VLSI along with their external memories and the host interface. This interface is designed with a Xilinx [15] FPGA 4005PG156-4. GFLOPS-2 implements the PCI standard bus for its I/O subsystem. The board mesures 12" by 16". The second level ring is not implemented yet. User access to a GFLOPS-2 machine is through a host IBM-PC computer. Client interface software is connected to the GFLOPS-2 machine via LINUX sockets to a server process running on the host.

3 RELATED WORK

Two families of programmable computers have been used to implement Image or signal processing applications: the general purpose computers and the application specific computers. A number of application specific computers has been proposed to treat efficiently applications such as image processing. Most of them are SIMD processors like SYMPATI-2 [1], SCAM [9], MGAP [10] or MasPar MP-1 [11] which obtain very good results on synchronous regular algorithms. Few MIMD dedicated processors have been developped such as IUA, PASM [4] or SM-IMP [7]. The general purpose GFLOPS computer is aimed at obtainning equivalent computing and input/output power as the dedicated SIMD architectures.

5 Conclusion

The diversity of algorithms in the Image processing field needs the use of general purpose architectures. The proposed architecture is both scalable and programmable in a high level language without sacrifying the processing and input/output capabilities. The C// parallel language will give us the way to program such architectures and develop complete image processing applications. The latency of fine grain communication is important to reduce the overhead of remote memory access. The communication in GFLOPS-2 uses small and simple two-word packets. Packets are created with a single instruction, transferred by self-routing on the network, and treated in the remote memory module without interupting the processor.

The working machine demonstrates that both the shared-memory and message-passing models permit efficient and scalable implementation; moreover, the two models should be unified in a same framework. At this time, effort is underway to built a single board 16-node machine. Although GFLOPS addresses many of the issues of large-scale multiprocessing, it is essentially a single-user machine which is designed to be used as a low-cost parallel co-processor board in a desk-top work station.

REFERENCES

[1] D. Juvin, J.L. Basille, H. Essafi, and J.Y. Latil. SYMPATI-2, a 1.5 D Processor Array for Image Application. In Signal Processing IV: Theories and Apllications. Elsevier Science Publishers B.V. (North Holland), 1988.

[2] K. Preston. The Abingdon Cross Benchmark Survey. IEEE Computer, pages 9-18, July 1989.

[3] L.A. Schmitt and S.S. Wilson. The AIS-5000 Parallel Processor. In Pattern Analysis and Machine Intelligence, 1987.

[4] J.T. Kuehn, H.J. Siegel, and D.L. Tuomenoksa. The use and design of PASM. In Integrated Technology for Parallel Image Processing, ed. S. Levialdi, Academic Press, pages 133-152, London, 1985.

[5] D. Houzet and A. Fatni. A 1-D linearly expandable interconnection network performance analysis. IEEE Int. Conf. on Application Specific Array Processors, pages 572-582, Venise Italy, October 1993.

[6] S. Borkar et al. Supporting Systolic and Memory Communications in iWarp. ACM Int. Symp. on Computer Architecture, pages 70-81, Seattle, may 1990.

[7] J.G.E. Olk and P.P. Jonker. A Programming and Simulation Model of a SIMD-MIMD Architecture for Image processing. In IEEE CAMP'95 Workshop, pages 98-105, Italy, Sept. 1995.

[8] A. Fatni and D. Houzet. C//, une extension parallèle du C pour machines multiprocesseurs. RenPar'7. Mons Belgique, Juin 1995.

[9] R.P. Rogers, I.G. MacDuff, and S.L. Tanimoto. Systolic Cellular Logic: Architecture and Performance Evaluation. In IEEE CAMP'95 Workshop, pages 51-58, Italy, Sept. 1995.

[10] H.N. Kim, M.J. Irwin, and R.M. Owens. MGAP Applications in Machine Perception. In IEEE CAMP'95 Workshop, pages 67-73, Italy, Sept. 1995.

[11] T. Blank. The MasPar MP-1 Architecture. In 35th IEEE COMPCON Spring'90 Proc., pages 20-24, Feb. 1990.

[12] Kendall Square Research. KSR1 Principles of Operation. Waltham, MA, 1991.

[13] L.M. Ni and P.K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. IEEE Computer, pages 62-76, 1993.

[14] M.C. Herbordt and C.C. Weems. Towards the Empirical Design of Massively Parallel Arrays for spatially Mapped Applications. In IEEE CAMP'95 Workshop, pages 59-66, Sept. 1995.

[15] Xilinx, "The Programmable Gate Array Data Book," 1994.