Architecture I

PTAH
Introduction to a new parallel architecture for highly numeric processing

Franck Cappello, Jean-Luc Béchennec, Jean-Louis Giavitto

LRI - UA 410 CNRS
Bâtiment 490 – Université de Paris-Sud
91405 Orsay Cedex
email : fci@lri.lri.fr

Abstract. This paper proposes a new architectural design for high performance parallel computers: the one-cycle machine. In such a computer the memory access, network access, instruction sequencing, data computation take the same duration: one clock cycle. We first consider the communication network efficiency as the main critical resource. We show that the adaptation of the network performance to the processing element power is more important than the CPU power in itself with respect to the global processing effectiveness. Two guidelines are derived from our analysis and conduct to the design of *PTAH*. Two simple examples are used to illustrate the interest of PTAH for the execution of numeric applications. Finally, some hardware features are proposed for a PTAH implementation being able to reach the TeraFLOPS.

I. Background

I.1. The "grand Challenge"

Substantial applications require a tremendous numeric computing power, as for example, climate modeling [1], electrical simulation, astrophysic, Quantum Chromo Dynamic (QCD) [2], fluids mechanic, thermodynamic, human genome decoding, medical imagery, etc. Two families of architecture have been devoted to the satisfaction of such intensive numeric computing requirements [3]: vector supercomputers like CRAY YMP C90 [4], NEC SX3 [5] and parallel supercomputers such as CM5 [6], GF11 [7], Paragon XP/S [8], TC2000 [9] and Monarch [10]

To face the increasing needs of numeric applications, supercomputer manufacturers have fixed the objective to reach the TeraFLOPS before the end of the decade. The solution to this challenge probably relies on a correct architectural design rather to a technological jump because a factor of 100 is needed to speed the available hardware up to the TeraFLOPS.

As a matter of fact, vector architectures are limited by the available technology and thus, parallel architectures will be more rapidly suitable to outperform the last generation supercomputers by a factor 100. Moreover, the low cost and the increasing performance of the current microprocessors (i860 XP from INTEL, RS6000 from IBM, T9000 from INMOS) make them attractive to be the processing elements (PEs) of the future parallel architectures. Such options are corroborated by the choices of the current supercomputers project like IBM Vulcan [11], CRAY Research MPP [12] and Thinking Machines CM5.

The communication network will make the principal difference between these architectures. To quote J. Smith et al. from [13]: "Data movement is always the most

difficult problem for the effective use of supercomputers. The performance of future supercomputers will ultimately be measured by how fast they can move data both within the system and across the network". This section is devoted to a critic, based on the analysis of the communication network usage, of three possible architectural designs.

I.2. The proposed solutions

Table 1 presents three architectures based on three CPUs. The CPU performance ranges from 1 MFLOP to 100 MFLOP corresponding to the next RISC generation. According to the CPU performance, column 2 gives the corresponding number of PEs required to reach the TeraFLOPS.

Such a number of PEs already implies the use of a mesh topology. At this network scale, 2D or 3D grids have been shown to be the more efficient topologies. Roughly, this is due to the fact that they take into account technologically-limited resources such as wiring density [14] or available pins number [15]. The choices retained for the actual projects of such architectures [12] [8] [16] [17] corroborate this analysis.

	MegaFLOP per PE	Number of PE	Typical CPU for the PE
M1	1	10^6 (practical limit)	16 bits (Mega PE, Mosaic PE)
M2	10	10^5	32 bits (680X0, 80X86 with co-processors)
M3	100	10^4	32-64 bits (next generation i860, RS6000,etc)

Table 1: A scale of number of PEs required to reach the TeraFLOPS according to the performance of one PE.

The PE performance directly involves the network link bandwidth. Table 2 shows the needed network bandwidth for two typical communication patterns. The program executed on each PE for these communication patterns needs one communication for each operation. For instance, a typical program instruction might be: compute a result from a value of an other PE and an internal value. So the program needs an operation/communication ratio of one (this assumption will be refined in figure 1). The comparisons are made on the basis of a static routing scheme (see paragraph II.1 for further discussions).

	Typical network topology (Mesh)	RCB Without Contention	RCB according to the bisection throughput	RCB (optimistic)
M1	100*100*100	532 Megabits/s	800 Megabits/s	*80 Megabits/s*
M2	47*47*47	2,5 Gigabits/s	3,76 Gigabits/s	*240 Megabits/s*
M3	22*22*22	12 Gigabits/s	17,6 Gigabits/s	*800 Megabits/s*

Table 2: The network characteristics and the link bandwidths corresponding to three typical architectures for the TeraFLOPS

The first communication pattern corresponds to a scheme where all PEs send a word to a receiver faraway off the communication network average distance (given in column 1). Moreover the communication is supposed without any contention. Column 2 gives the Related Channel Bandwidth (RCB). Let's d be the average distance and 6 the degree of a

PE (in the 3D grid). If we assume an idealized channel load rate of d/6 then RCB = (CPUthrougthput . wordlength . d/6). In fact, if all PEs emits a message to a neighbor, the (average of) channel load rate is 1/6 and so, if the messages are sent to a distance of *d*, the channel load rate is d/6.

The second communication pattern is an attempt to take into account the unavoidable contention. The model we propose consists in estimating that 1/4 of the messages cross over the network bisection. The computed value (column 3) is known as the bisection throughput. These values are unreachable with the current technology. Column 4 gives more realistic hypothesis about what can be obtained.

In order to obtain a TeraFLOPS with these more realistic network bandwidths, we have to examine the operation/communication ratio that reduces the PE throughput. The results are summarized in figure 1.

The resulting curves are plotted from the following relationship: $Ct = \frac{32\, n\, \lambda}{4\, R}$ where *Ct* represents the channel throughput, 32 the word length, *n* the side of the mesh, λ the PE throughput and 1/4 comes from the fact that only one quarter of the messages cross over the bisection. The parameter *R* represents the ratio operation/communication for a PE. This ratio is the same for all PE. Actually the interpretation of *R* is a little bit more sophisticated: it also includes the ratio *Application size/Computer size* (known as the *vp-ratio* in the Connection-Machine programming languages [18]). When the application size is larger than the computer size, the application is folded and this may reduces of inter-PEs communications: parts of the application that are mapped on the same PE communicate without leaving the PE.

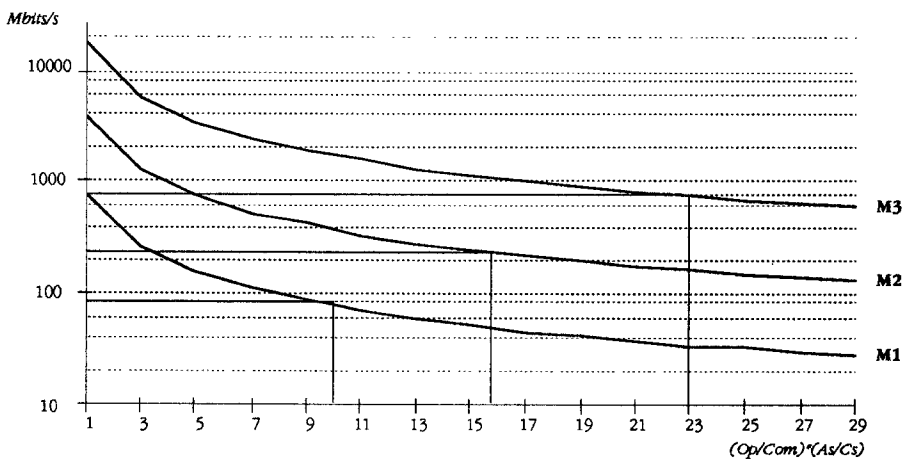


Figure 1: Number of operations per communication required for M1, M2 and M3 to reach the TeraFLOPS according to the realizable link bandwidth and the bisection throughput.

At least a ratio of 10 operations per communication is required for the architectures with one million of PEs to reach the TeraFLOPS. This performance is obtained with *M3* for a ratio of 23 operations per communication. So, these architectures become efficient only from a high *R*. The important conclusion we draw from the figure 1 is summarized in figure 2.

Fine-grained architectures (*M1*) are more effective to obtain as soon as possible the TeraFLOPS. Here fine-grained should be understood as a *low operations/communication*

ratio. To emphases our argument, let takes an application that fit the size of the computer. If the communication ratio is 10, the TeraFLOPS can be achieved with *M1*. With *M3* we have to group the application on less PEs with the hope to increase the operations per communication ratio. However, grouping the application on less PEs decreases the achievable computing performance.

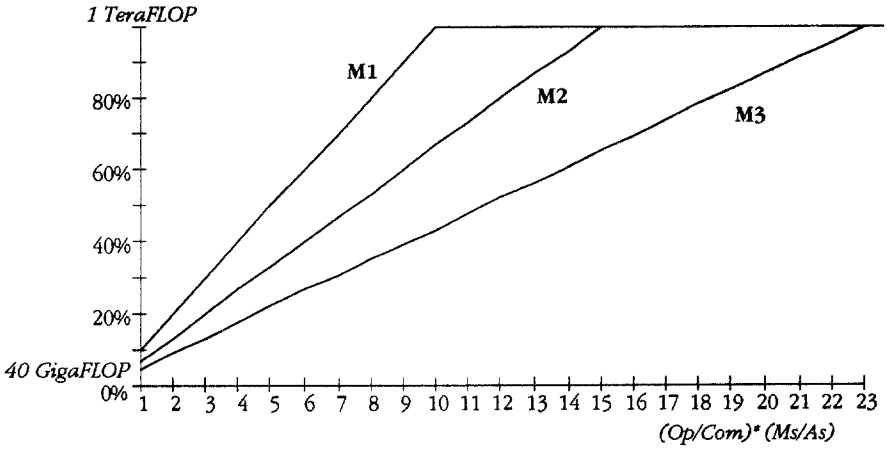


Figure 2: Achievable performance following R

I.3. CPU-Driven versus *Network-Driven* Architectures

As a consequence, we oppose *CPU-driven* to *network-driven* architectures. In CPU-driven architectures, the CPU performance exceeds the network capabilities. More the CPU is powerful more the ratio instruction/communication must be high (to obtain the TeraFLOPS). The PE activity is characterized by bursts of computation and a big number of PEs must be used to remedy the weak efficiency. In contrast, network-driven architectures are designed to adapt network and CPU performance. Because the network is able to feed the CPU at its nominal rate, *sustained performances are close to peak performance*. Thus, the TeraFLOPS is obtained more rapidly.

In this paper we develop *PTAH, a proposal for network-driven architecture*. We first present two consequences of the network driven architecture concept. Then, we develop the underlying execution model and we illustrate them on two toy examples. Finally, the implementation of a network of processors able to reach the TeraFLOPS is discussed.

II. One operation and two communications per PE and per cycle

PTAH is an architecture that follows a principle of effectiveness. This can be rephrased as to *optimize the use of the PEs*. The structure of the PE must be designed in order to furnish an optimally usable computational power.

From the application point of view, the efficiency of a PE is measured in the number of "useful" operations performed. Useful operations are the logic and arithmetic ones. Therefore the conclusion is that PTAH must be able to perform:

one operation and two communications per PE and per cycle.

We define a cycle as the duration between two executions of the basic operations: the floating point multiplication is a basic operation. At each cycle, each PE executes its programmed operation. So at each cycle, a PE consumes at most three operands: two for a dyadic operation and a Boolean value in case of a conditional execution. In addition, at each cycle, a PE may output two values (e.g. when swapping two values).

Operands of an operation are accessed both in the PE local memory or received from other PEs through the network. This also handle for the results: they may be delivered to the communication network.

II.1. Static versus Dynamic routing

Dynamic routing of the communications is used into many general purpose parallel architectures [6, 19, 20, 21].

The dynamic routing uses an interpretation level to route messages which therefore must embed a network information section (addresses) and an application data section. In addition, the dynamic routing strategy must be distributed over the PEs, leading to a far from optimal conflicts resolution. The values appearing in the previous tables are then far from realistic if considered with dynamic routing. For example, in the iPSC/2, the hardware does not limit the communication performance because it represents only 5% of the transmission delay [19]. This limitation is not linked to a special architectural design, e.g. it also holds for the CM2 where the typical duration of a *send* takes the time of 75 additions [6].

Parallel architectures with dedicated networks are built to execute a restricted family of applications. Their networks are directly defined to fit the communication requirements of the applications. Systolic architectures [22], which realize hardware operators, are examples of a such approach. Others examples include complete dedicated computer, such as the Yorktown Simulation Engine [23] built by IBM to speed-up logic simulations, the Wire Routing Machine [24] to route VLSI or the GF11 to compute QCD [2]. Several other dedicated machines have been build, especially in the field of image processing (MPP [25], PASM [26], Sphynx [27]).

Such architectures, from the network point of view, answer to the question: which data movements must be performed? They are very efficient because of the optimal matching between the structure of the processing architecture and the computing structure of the application. However, when these machines are used as general purpose architecture, they lose their efficiency.

The static routing is a way to combine the flexibility of dynamic routing and the efficiency of dedicated networks. However the static routing implies the compilation of the communications (see section III) and a careful analysis of the network properties.

II.2. The specification of the network properties

To define the PTAH network we must analyze the characteristics of the communications of an application which will be executed on a parallel computer. If we consider globally the whole execution of an application, its communication geometry looks like a "jungle". But three connection patterns can be exhibited at a given time: the *permutation*, the *broadcast* and the *reduction*. The permutation and broadcast connection patterns implies nodes that receive at most one communication at a time. In a reduction pattern, a node is able to receive two communications at the same time because the operations that appear in a reduction imply the dyadic operators performed by a PE. So a reduction is realized with a binary tree (for the associative operations). These simple

patterns can be combined to realize any complex pattern.

In consequence, to avoid inefficient hotspots, the degree of the communication network is fixed as two inputs and two outputs per PE. Moreover, the topology of the network must satisfy two additional properties:

- a *Benes* property [28] to implement any permutation without contention,
- a *partitionable* broadcast capability to allow independent partitions to execute local broadcasts (broadcast can be limited to a subset of PEs).

The next section investigates the adaptation of the processing structure to the applications. The architectural impacts of "one operation and two communications per PE and per cycle" are explored in section IV.

III. Adaptation of the architecture geometry to the application structure

The network must be able to match the CPU processing rate and so must perform at most two communications per cycle and per PE. Moreover, the optimal execution of an application distributed over a network of processors requires a suitable interaction between PEs. Therefore, the structure of the computer must be flexible enough to be well-fitted to the structure of any given computation. As a consequence, PTAH provides the

adaptation of the architecture geometry to the application structure.

The execution of an application is characterized by a *communication geometry* and by a *control geometry*. The communication geometry fixes the interconnection topology of the PEs which correspond to an optimal execution of the application. The communication geometry is often derived from the dataflow graph of the application. For example, a Perfect Shuffle topology can be used to support optimally the data movements of a FFT [29]; meshes are usually used to support finite differences schemes, etc.

The control geometry defines the optimal execution model of the application. As for the data organization, it exists a control organization which sequences the operations to perform. For example, the SIMD control model organizes the processing of the application as a single sequence of operations executed simultaneously by all PEs. The optimal control geometry depends on the application.

The adaptation of the architecture geometries to the application structure is required to provide for each application the best executing structure. The communication network and the control structure are the basic resources used for the geometry adaptation. The section V presents some performant communication and control geometries for some standard numeric applications.

The no-sens program proposed by Gasjki in [30] compares the efficiency of an ideal dynamic Dataflow architecture with an ideal SIMD or Vector architecture. This simple program shows the requirement of a *flexible* control structure that must be adapted to the algorithm specificities. Because the parallel computers use the microprocessors technology for the PE, they can use the microprocessor instruction cache or local memory to store the program and the microprocessor sequencer to control the program execution and take advantage of the large amount of instructions sequencing locality of the applications (most executed instruction sequences are within loops). However, this large amount of unorganized threads must be control to avoid the lost of efficiency. The *synchronous MIMD* control structure which combines a synchronization hardware with a multiple threaded architecture fits all these requirements.

IV. General design of a PTAH architecture

At this point, we have determined that the optimal use of each PE involves:

– the execution of one operation and two communications per PE per cycle;

– the adaptation of the geometry of the architecture to the application structure.

These two constraints lead to the concept of *one-cycle machine:* all operation, memory access, network access, instruction sequencing, data computations..., take one cycle. These constraints are very hard to satisfy and require hardware specificities (detailed in IV.2). In particular, the feasibility of the architecture implies a static approach of the execution to avoid expensive interpretation level.

IV.1. A compiled model of execution

The use of interpretation levels within communications (*dynamic routing*) and control (*micro-coding*) is well-known as involving performance limitations. Thus, efforts to reduce the use of interpretation levels has given birth to communication compilers [31] and MSIMD control model [32] (in the field of sequential machine, the RISC concept [33] participates of these efforts).

Therefore, the communication and the control structure must only depend on static parameters. It means that tasks creation/destruction, partitioning, placement and scheduling must be anticipated at compile time. Each communication collision must be resolved at compile time. The communication patterns and occurrences have to be known before execution. As a matter of fact, these severe constraints limit the target applications. However, such a class of applications is an important and pertinent one. Interpretation levels may be introduced, at the price of a lesser efficiency, to execute inherently dynamic applications (that is, application whose behavior cannot be anticipated). For example, it is possible to implement a dynamic routing scheme upon the static routing for highly unstructured applications or for some sparse matrix and graph algorithms. A hardware communication routing chip can be add to each switch node of a dynamic topology communication network. This results in a few increases of the hardware complexity since the switches in these communication networks are not limited by the transistor count but by their pin count. However, if this improvement is to expensive according to the irregular operation occurrences, the dynamic routing can be achieved by using the PE as the routing chips (sotfware emulated) of a static topology communication network.

The structure of each resource, memory, communication network, CPU,..., is very simple and so leads to a simpler compilation. Task scheduling may use techniques issued from the VLIW compilation and compilation of communication may profit of the technology developed for Router-Compilers.

IV.2. Specialized resources with balanced bandwidths

Each CPU has two inputs for the operands and two outputs for the results. So the private memory and the communication network allow four simultaneous accesses (two reads and two writes). Because all the architecture resources receive the same clock, the synchronization is implicit and all the actions made by each resource have a calibrated duration of one cycle.

The optimal use of the PEs requires to realize each architectural function (processing, control, communication, memorization, synchronization,...) with a dedicated hardware. Sharing hardware resources between functions induces bottlenecks and therefor delays the function performance. So each resource is specialized.

IV.2.a. An adaptable communication network

The communication network is the critical resource of a network of processors with compilable geometry. It fixes the duration of the cycle used to clock all resources. The switched circuits method [34] is used because it allows a short access time, a short latency and a high bandwidth. Communications are implicit in such a network and the connection patterns - sets of circuits - are defined at compile time and sequenced at execute time.

The topologies of the indirect networks such as the Crossbar, the Memphis Switch, the Benes, and the various over-dimensioned direct networks (as meshes) can realize the standard interconnection patterns : permutation, broadcast and reduction.

IV.2.b. A set of adaptable control units

As previously mentioned, a MIMD synchronisable control structure is flexible enough to fit the best executing control structure. A MIMD control structure can realize the control part of MIMD, MSIMD, SPMD and SIMD control modes. Each PE must have its private sequencer to realize all these modes. The synchronization of all PEs uses a hardware synchronization network. Each PE has one input and one output to access these networks. Each PE accesses to the synchronization networks are defined at compile time. Partitionable networks are used to allow the synchronization of independent sets of PEs. Two synchronization networks are required to realize hierarchical synchronization. For example, it is possible to synchronize multiple PEs within a partition and multiple partitions within the machine as needed for a MSIMD control structure.

The control structure provides exactly one instruction per cycle to the processing unit. The control structure uses private resources (memory, ALU) to realize control processing. An anticipation unit is used to execute the loop controls, the jumps and the other sequence breaks. So, the control processing is independent of the data processing.

IV.2.c. A set of dedicated processing units with fixed format

The goal of the processing unit is to provide one useful result per cycle. Useful results are results of processing under application data. Performing operation for program or communication controls is not considered as providing useful results. The processing unit is dedicated, i.e. it does not compute the controls or the communications. A Harvard memory structure is used to separate the storage of the application data and the storage of the control data. Because PTAH is designed for highly numeric processing, the format of the processing unit is fixed to 32 or 64 bits. But adaptable processing units can be used, if needed, with balanced control, storage and communication resources.

IV.2.d. The memory structure for the application data

The memory structure for the application data allows four accesses within each cycle. Therefore, it is important to provide exactly four accesses per cycle. Providing an average of four accesses per cycle is a weaker property which complicates the management of the other resources. Thus, the memory structure cannot be hierarchic (register, cache, static memory, dynamic memory,...). The natural memory structure is then composed of four memory banks. The optimal placement of the data in the memory banks is resolved at compile time.

Shared data are accessed through the communication network. The broadcast facilities of the communication network are used to realize simultaneous "reads" to the same data. A multiple "write" access uses the reduction method for an associative operation or a

sequential access for non-associative operations. The hotspots (contention places) are detected and resolved statically at compile time.

IV.2.e. An adaptable I/O access

PTAH provides an adaptable I/O access according to the application requirements: a direct access of all the PEs or a serial access of a dedicated PE. A direct I/O access of all the PEs is possible according to the I/O units capabilities. Each PE has two input and two output ports for the I/Os. Multiple dedicated PEs can be used to adapt the machine bandwidth with the I/O bandwidth. Then, the communication network is used as a concentrator-broadcaster. The access configuration and the management of the communication network for the I/Os are resolved at compile time.

V. Some Illustrations of the PTAH features

As a general principle of parallel computer architecture, PTAH can be used from small to very large parallel computers (from 2 to 64k PEs). So, we have deliberately chosen very simple and small examples to illustrate the principle of PTAH. Despite there limited usability, these examples show some of the architecture features. The implementation and performance results of real application (QCD, ...), library (Linpack, FFTpack, BLAS3) and benchmarks will be presented in future papers.

In the following examples, the geometry of the executing structure is adapted to the algorithm structure chosen in the example. We assume that the duration of the cycle is equal to the duration of a whole 32 or 64 bits floating point multiplication[1].

V.1. An implementation of a standard benchmark: the matrix-matrix product

The Basic Linear Algebra Subprograms (BLAS) [35] have become a standard library to support linear algebra. The BLAS operations are divided into three levels : level 1 concerns Vector-Vector operations, level 2 includes Matrix-Vector operations and level 3 is devoted to Matrix-Matrix operations. We will examine the computation of the Matrix-Matrix product, a level 3 BLAS operation. In particular, we will focus on the product of 3*3 matrix. The Figure 3 shows the geometrical evolution of PTAH while computing a 3*3 matrix product.

The subprogram consists in the product of the matrix $A=(a_{ij})$ by the matrix $B=(b_{ij})$. The result is stored in matrix $C=(c_{ij})$. We assume that: the executing structure has 9 PEs, all matrix are square and contains 9 elements and all elements are stored on a different PE. For sake of simplicity, we also assume that the PEs are numbered as the matrix elements and so a_{11}, b_{11} and c_{11} are stored on PE_{11}.

The Figure 3 show at cycle 1 the communications performed at cycle 0 and the operations executed at cycle 1. The communications are pipe-lined and realized one cycle before the operation which use them (see section VI). But for more simplicity we show the communications and their associated operations within the same cycle. So c_{11}, c_{21} and c_{31} are partly computed at cycle 1. The partial computations are the products part of the computation of the matrix C elements. Thus at cycle 1, PE_{31} receive a_{31} and b_{11} to compute the products part of c_{31}. At the same time PE_{31} transmits b_{31} to PE_{13}, PE_{23}

[1] Some comercial components and full custom CMOS VLSI realize such an operation in less than 50ns.

and PE_{33}. The results of the partial computation are stored and will be used at cycle 4 and 5. The cycle 2 and 3 partly computes c_{12}, c_{22}, c_{32}, c_{13}, c_{23} and c_{33}.

The cycle 4 and 5 perform the sum of the partial results previously computed and so compute the definitive value of each element of the matrix C. Thus at cycle 4, PE_{31} receives the result of $a_{32}*b_{21}$ from PE_{32} to compute $(a_{31}*b_{11})+(a_{32}*b_{21})$. At the same time, PE_{31} transmits the result of $a_{31}*b_{11}$ to PE_{31}. The cycle 5 execute the second half part of the sum. *Finally, the product of the matrix 3*3 takes 6 cycles to compute 27 multiplications and 18 additions i.e. 45 operations.* In the example, the seed-up of PTAH over a sequential computer is 7.5 and the efficiency is about 83%. With the assumption that any communication is required for the cycle 1, i.e. the elements of the matrix are placed on the PEs in such a manner that no communication is required to compute the product part of c_{11}, c_{21} and c_{31}, the speed-up becomes 9 and the efficiency becomes 100%! For this processing, the communications are permutations and broadcasts and a simple SIMD control structure is required. For matrix of size N*M on a machine with u*v PEs, if there is enough memory space to store the partial results the speed-up is bounded to u*v. Else, blocks of matrix are used and the speed-up is bounded by $\lceil N/u \rceil * \lceil M/v \rceil$.

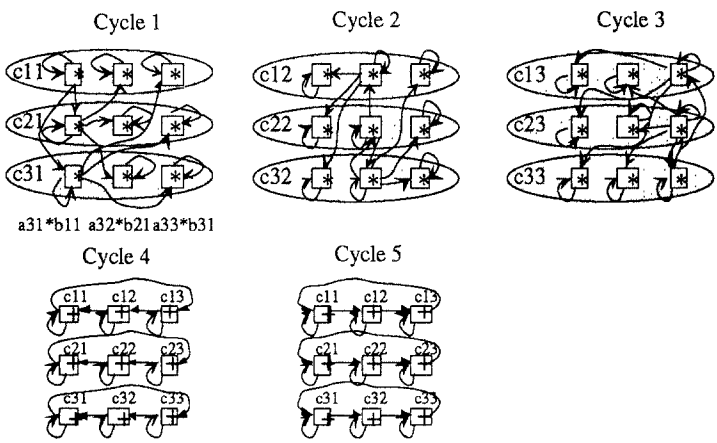


Figure 3: Adaptation of the PTAH geometry for the computation of a matrix 3*3 product.

V.2. A direct use of the systolic designs: implementation of the convolution

The convolution is a routine often used in signal processing, image processing, FFT... and which has given birth to the systolic concept. The convolution computes the result vector $\{y_1, y_2, \ldots, y_{n+1-k}\}$ with $y_i = w_1x_i+w_2x_i+\ldots+w_kx_{i+k-1}$ from a vector of weights $\{w_1, w_2, \ldots, w_k\}$ and a vector of samples $\{x_1, x_2, \ldots, x_n\}$. The computing structure presented in figure 4 has the same properties as the classical linear systolic structure [22]. Thus, according to the limited I/O bandwidth assumption, the inputs (x_i) and the outputs (y_i) are made in a serial manner and all inputs are used in the computing structure as long as possible.

All the data comes from the network and the weights are stocked in the local memory. At each cycle, each PE realizes one operation. The data of the input vector x are broadcasted to PE1, PE2, PE3 and PE4. Each intermediary result uses a specific communication channel. One sample is broadcasted at each cycle. The multiply and add

operations are pipelined. Within the first cycle, x1 is broadcasted to PE1, PE2, PE3 and PE4. During the next cycle x2 is broadcasted and x1 is multiplied with w1, w2, w3 and w4. At the third cycle, PE5 receives w1*x1, x3 is broadcasted and x2 is multiplied with w1 and w2. PE5 adds w1*x1 and 0 at cycle 4 and transmit the result to PE6 at cycle 5. The result of w2*x2, communicated in cycle 4, is buffered during cycle 5 and added with w1*x1 at cycle 6. At this time, five cycles are required to output the first result and the convolution is terminated at cycle 13. The memories of PE6, PE7 and PE8 are used as internal buffers. This geometry provides a speed-up of 2 and the efficiency is about 26%.

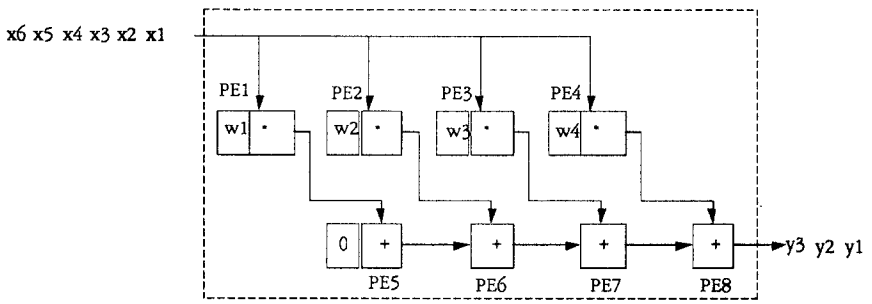


Figure 4: Two geometries of PTAH for the computation of a convolution.

VI. Guide lines for the implementation of a PTAH machine

A PTAH architecture is mostly function of the desired PE number. The number of PEs fixes the topology of the communication network according to the technological constraints [21]. The network also determines the granularity and the duration of one cycle and defines the bandwidth of the network links. According to the implementation choices, each link will have a high number of wires to transport the informations in a parallel manner or it will use serial links with high bandwidth wires. In this section, we will discuss a very large architecture designed to reach the TeraFLOPS.

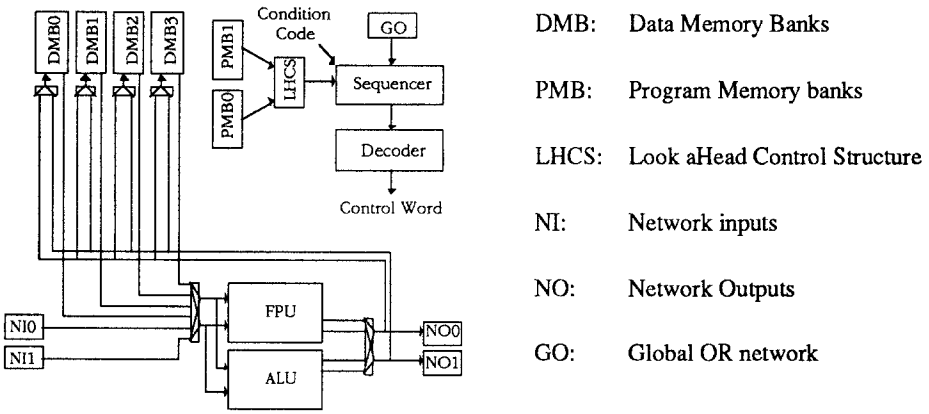
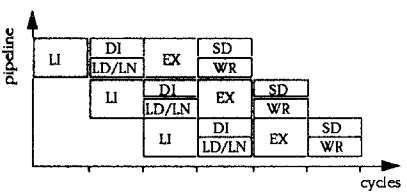


Figure 5: Simplified architecture of a PE for PTAH.

VI.1. The Processing Element

The PE is built from its processing unit which has a FPU for floating point operations and a ALU for fixed point, integer and logic operations. The PE granularity corresponds to the granularity of the processing unit, i.e. 32 or 64 bits. Figure 5 shows a simplified scheme of the common PE with four memory banks, four communication network accesses, the control structure, the anticipation unit, the FPU and the ALU.

Each PE must perform one useful operation per cycle to guarantee an optimal use of its processing unit. So a RISC-like architecture with a four stages pipeline is used (see figure 6). Because data can be accessed from the private memory or from the communication network, the pipeline stages 2 and 4 concerns both memory and network operations. A zero delay branching sequencer must be used to avoid the breaking of the PE pipeline. This can be realize with a sequencer that works faster than the FPU or with multiple program memory banks providing multiple instructions at each cycle.



LI: Load Instruction
DI: Decode Instruction
LD: Load Data
LR: Load from Network
EX: EXecute
SD: Store Data
WR: Write to Network

Figure 6: Execution pipeline of the PE

VI.2. The communication network

We now consider an implementation allowing to reach the TeraFLOPS with 64K PEs. The cycle is fixed at 60 ns. Each PE can provide 16 MFLOPs. The network topology is a 3D mesh with 40x40x40 nodes. The topology of a 3D mesh has regular and short connections and so allows the use of larger communication links and a higher network bandwidth.

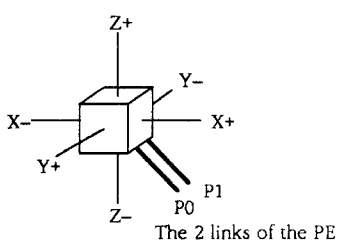


Figure 7: A connection box

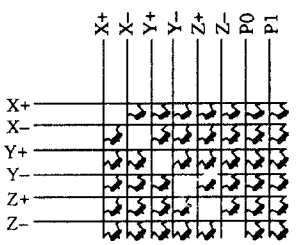


Figure 8: The CROSSBAR of a connection box

In a 3D mesh, a connection between two distant nodes crosses over intermediary nodes. To allow the connections without constraint between the nodes, further links available to cross over a node must be provided. These links cross over the node in the three directions without any node interaction. Links are bi-directional and the circulation

direction is fixed at compile time. The number of links in one direction depends on the length of the mesh in this direction. Thus for a 3D mesh with 40 nodes in each direction, the number of links for a node must be 40 in each direction to allow any interconnection pattern (even the permutation) in each direction.

This over-dimensioned 3D mesh can be realized with 40 independent 3D mesh. Each node has a connection box (see Figure 7). This box allows to connect multiple links of the three directions and the node links to the mesh links. Each node has one link to input data and one link to output data. While each node is able to receive two inputs and to provide two outputs within each cycle, two communication networks are required. So each node has 80 connection boxes. A connection box contains a small CROSSBAR (see Figure 8).

A connection box owns a pattern memory to store the different configurations to use. The pattern memory can store up to 1024 configurations. A pattern sequencer provides the address of the connection pattern according to the current instruction executed by the PEs. Figure 9 shows the node structure of a 64K PEs network with compilable geometry.

At hardware level there are 80 links between two neighbor nodes. The parallel communication on links is impossible and so serial communications are required. We consider that the performance of each PE is 16MFLOPs and that the data length is 32 bits. So a serial link at 512 Mbits/s is required to connect neighbor nodes. This is possible with a BiCMOS technology providing that each signal is associated with a ground. The number of wires between two neighbor PEs is then 160.

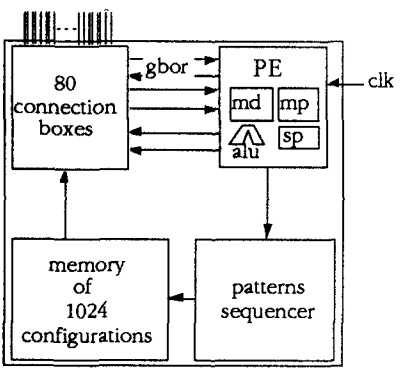


Figure 9: The node structure of PTAH 64k

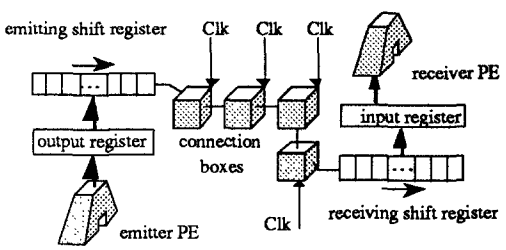


Figure 10: Communication pipelining in the network

In a 3D mesh, the task placement on the nodes fixes the distance (the number of nodes to cross over) between two communications nodes. For a 40x40x40 mesh, this distance can be up-to 120 nodes. But the duration of the data transport depends on this distance and the access time of the network fixes the duration of the cycle used for all resources. So, if we want to maintain a short cycle, the access time must be dissociated from the duration of the data transport. The solution retained is to pipeline the communications in the network. So each box works like one element of a shift register (see Figure 10).

The message cross over the network from connection boxes to connection boxes. Since two consecutive connection boxes are spatially neighbor, they can be easily synchronized. The communication pipelining provides the following advantage: while the communication frequency is 32 times higher than the computing frequency (for 32 bits data), the network is crossed over 32 time faster than if the same frequency was used for the computation and the communication.

This result requires some explanations: if T is the length of the message to

communicate and *D* the distance in number of connection boxes between the transmitter and the receiver - notice that there are always a minimum of two connection boxes between one transmitter and one receiver - the number of shifting cycle is: $C = \frac{(D+T)}{T}$.

Thus, in a neighborhood such as $D + T \leq 2T$, the transmission delay is constant and equal to 2 PE cycles. For T = 32, the neighborhood is 31 nodes in each direction. The drawback of the communication pipelining is that the distance between the two communicating node fixes the instant when the message must be send. So a clock must be built from the division of the pipeline *Clk* by *T*. This clock must be shifted of D.Clk cycles from the node clock of the receiver.

The 3D mesh of PTAH 64k is partitionable per blocks. The Blocks are sets of hardware connected nodes. A block can contain from one node to the whole machine.

The global 3D mesh can be viewed as containing multiple reduced 3D mesh. Each reduced 3D mesh can be independent and used as a reduced network of nodes with compilable geometry. This feature allows to execute multi-tasking or to share the machine between multiple users. The "Global-Or" synchronization networks are realized with two 3D meshes independent of the communication networks. Since only block partitions are possible on 3D meshes, each "Global-Or" network uses only a single wire. Each node receives 6 wires, one for each direction. Partition are defined at compile time. Each node has the required hardware to logically "cut" its wires to close the frontier of a partition. Once the partitions are realized, the two "Global-Or" networks can be seen as multiple independent "Local-Or" networks.

As we have previously mentioned, the I/O access can be limited to one dedicated node or extend up-to a direct access of all nodes. The over-dimensioned 3D mesh of the 64K nodes network can be uses as a concentrator-broadcaster for a limited I/O access or to realize all paths required for a direct I/O access. For a direct I/O access, the topology and the number of links of our over-dimensioned 3D mesh guarantee that all nodes can simultaneously input or output data to and from the I/O devices. All nodes can be viewed from one face of the 3D mesh within one cycle. For simultaneous inputs and outputs the two communication networks and two faces must be used.

A 3D packaging technology is required to build a 64K nodes network. The MegaPack [36] – a 3D VLSI packaging for direct construction of 3D computers – can be used to realize our network. The PE and the connection boxes can be made with ASIC components. An other way to build such a machine is to use commercial components mounted on classical boards. The boards must be studied to be connected directly without the use of back-plane or racks. The board assembling forms naturally the 3D networks. The machine is a cube of boards and its implementation will lead to a higher volume than with the MegaPack.

VII. Conclusion

Our examination of the actual massively parallel architectures shows that the communication systems (dedicated networks and general purpose networks) lead to a non-optimal use of the potential computing performance. So we have proposed to optimize the use of the PEs by the adaptation of the architecture geometry to the application structure and the execution of one operation and two communications per PE and per cycle.

The main consequence of our approach is: the bandwidth adaptation of the communication network, the memory and the control structure to the performance of the processing unit. The bandwidth adaptation leads to the use of hardware resources that are

dedicated. The performances required for the communication network require the use of the circuit switching technique. The sequencing of the connection patterns associates the efficiency of the dedicated networks to the flexibility of the general purpose networks.

Our execution model has been illustrated through two simple examples.

The implementation of a 64k PEs machine, designed to reach the TeraFLOPS, has been detailed. Some details of the hardware implementation have been described and particularly the node structure. The machine uses the communication pipelining to allow the reception and the emission of a total of 4 communications per cycle. A particular organization of the data memory and a zero delay branching program sequencer are required to allow the PE to make one useful computation per cycle.

The critical resource of this architecture is the communication network. We have proposed a network topology corresponding to the requirements of the architecture. The over-dimensioned 3D mesh can be partitioned with blocks. Its communication pipeline reduces the latency to 5 PE cycles despite a diameter of 120.

Acknowledgments

The PTAH Project is developed within the "Computer Architecture and VLSI Design" Research Group at LRI. The authors do thanks the other members of this group: C. Germain for his outstanding contribution, J.-P. Sansonnet and D. Etiemble for many helpful comments and support and F. Delaplace for fruitful discussions. This work is partially supported by the french national research program on New Computer Architectures (PRC-ANM) and by DRET under grant #89342320047050/.

Réferences

1. D. Korn, N. Rushfield, "Washcloth Simulation of Three-Dimensional Weather Forecasting Code", New York University, May 1983.
2. J.J. Dongarra, "Experimental Parallel Computing Architectures", North Holland, 1987.
3. R.W. Hockney, C.R. Jesshope, "Parallel Computer 2", Adam Hilger, 1998.
4. W.D. Robb, "The Cray YMP C90 Computer System", Supercomputing Europe 92, Paris, 1992.
5. T. Watanabe, "SX-3 Series Architecture & Technology Trend", Supercomputing Europe 92, Paris, 1992.
6. "The Connection Machine CM-5 Technical Summary", Thinking Machines Corporation, Oct 91.
7. J. Beetem, M. Denneau, D. Weingarten, "The GF11 Supercomputer", in Proceedings of the 12th Internationnal Symposium on Computer Architecture, IEEE Computer Society, Boston 1985.
8 "Paragon XP/S Product Overview", Intel Corporation,1991.
9. "The TC2000 Massively Parallel Supercomputer", in "Parallel Computing : Past, Present and Future", BBN Advanced Computers Inc, 1990.
10. R. D. Rettberg, W. R. Crowther, P. P. Corvey and R. S. Tomlinson, "The Monarch Parallel Processor Hardware Design", Computer, April 91.
11. J.R. Moulic, "Parallel Systems", Supercomputing Europe 92, Paris, 1992.
12. S. Nelson, "Toward TeraFLOP Computing", Supercomputing Europe 91 Conference, Fevrier 1991.
13. J.E. Smith, W.C. Hsu, C. Hsiung, Supercomputing 90.
14. W.J. Dally, "Wire-efficient VLSI Multiprocessor Communications", 1987

Stanford Conference on Advanced Research in VLSI, 1987, pp 391-415.
15. D.A. Reed, R.M. Fujimoto, "Multicomputers Networks – Message-based Parallel Processing", The MIT Press, 1987.
16. W.C. Athas, C.L. Seitz, "Multicomputers : Message-Passing Concurrent Computers", COMPUTER, Aug 1988.
17. C. Germain, J-L. Béchennec, D. Etiemble and J-P. Sansonnet, "A New Communication Design for Massively Parallel Message-Passing Architectures", IFIP Working Conf. on Decentralized Systems 1989, North-Holland ed.
18. TMC, *The Essential *Lisp Manual*, Cambridge MA, 1986
19. S. F. Nugent, "The iPSC/2 Direct-Connect Communications Technology", 3rd Conf. on Hypercube Concurrent Computers and Applications, 1988.
20. W. Crowther, J. Goodhue, E. Starr, R. Thomas, W. Milliken and T. Blackadar "Performance mesureament on a 128-nodes Butterfly Parallel Processor", proc. of the Int. Conf. on Parallel Processing, pp 450-457, 1985.
21. C. Germain, J-L Béchennec, D. Etiemble, J-P. Sansonnet, "An interconnection network and a routing scheme for a massively parallel message-passing multicomputer", 3rd Symposium on Frontiers of Massively Parallel Computation, Oct 8-10 1990, College Park, MD.
22. H.T. Kung, "Why Systolic Architecture ?", COMPUTER, Jan 1982.
23. M.M. Denneau, "The Yorktown Simulation Engine", ACM/IEEE 19th Desing Automation Conference Proceedings, 1982.
24. S.J. Hong, R. Nair, "Wire-Routing Machines, New Tools for VLSI Physical Design", Proceedings of the IEEE, Jan 1983.
25. K. Batcher, "Design of a Massively Parallel Processor", IEEE Transaction on Computer, Sep 1980.
26. H.J. Siegel, L.J. Siegel, F.C. Kemmer, P.T. Muller, H.E. Smalley, S.D. Smith, "PASM : A Partitionable SIMD/MSIMD System for Image Processing and Pattern Recognition", IEEE Transaction on Computer, Dec 1981.
27. A. Merigot, S. Bouaziz, P. Clermont, F. Devos, M. Echer, J. Mehat, Y. Ni, "SPHINX un processeur pyramidal massivement parallèle pour la vision artificielle", 7ième congrès RFIA, Nov 1989.
28. V. Benes, "Optimal Rearrangeable Multistage Connecting Networks", Bell System Technical Journal, Vol 43, no 4, Part 2, Jul 1964.
29. H.S. Stone, "Parallel Processing with the Perfect Shuffle", IEEE transaction on Computers, Feb 1971.
30. D.D. Gajski, D.H. Lawrie, D.J. Kuck, and A.H. Sameh, "CEDAR", IEEE COMPCON'84 Proceedings, March 1, 1984.
31. E. Denning Dahl, "Mapping and Compilated communication on the Connection Machine System", Proceedings of the fifth Distributed Memory Computing Conference, Apr 1990.
32. David Elliot Shaw, "SIMD and MSIMD Variant of NON-VON Supercomputer", IEEE COMPCON'84 Proceedings, March 1 1984.
33. D.A. Patterson, "Reduced Instruction Set Computers", Communication of the ACM, Jan 1985.
34. Tse-Yun Feng, "A Survey of Interconnection Network", Computer, December 81.
35. C. Lawson, R. Hanson, D. Kincaid, F. Krogh, "Basic Linear Algebra Subprograms for Fortran usage", ACM Transaction on Mathematic Software, 1979.
36. J.-L. Béchennec, F. Cappello, D. Etiemble, "A 3D Hardware Package for highly Parallel Architectures", Euromicro 91, 1991.