Davis, E. W., T. Nordström and B. Svensson, “Issues and applications driving research in non-conforming massively parallel processors,” in *Proceedings of the New Frontiers, a Workshop of Future Direction of Massively Parallel Processing*, I. D. Scherson Ed., McLean, Virginia, 1992, pp. 68-78.

Issues and Applications Driving Research in Non-Conforming Massively Parallel Processors

Edward W. Davis[+], Tomas Nordström[‡], and Bertil Svensson*

[+] North Carolina State University
Raleigh, North Carolina

[‡] Luleå University of Technology
Luleå, Sweden

* Chalmers University of Technology
Gothenburg, Sweden

Abstract

Concepts such as modularity and heterogeneity are becoming important for a growing number of applications that use massively parallel computer architectures. Application areas which seem to require these concepts appear in real world computing and action oriented systems. In many instances the current offerings of high performance, parallel, general purpose computers are not well suited to these applications since they do not address issues like real-time, time determinism, heterogeneous communication, physical size, power consumption, etc. These issues are important in special systems that can be viewed as non-conforming to general purpose markets. The differences in needs will be explored by looking into two examples of modular and heterogeneous systems: high performance instrumentation systems and action oriented systems. We raise some research issues that need to be resolved in order for modular and heterogeneous systems to be used effectively and efficiently.

Key words:

Massively Parallel, Non-Conforming Computers, Modular, Heterogeneous, Embedded, Real-World Computing, Real-Time, Action Oriented Systems.

1. Introduction

The goal of the Frontiers '92 Workshop on Processor Architectures was to identify problems that could be addressed through research, and whose solutions would promote the availability and use of massively parallel processing. In recent years the meaning of "massively parallel processing" has broadened. In 1986, when the first Frontiers symposium was held, the phrase meant systems with more than 1000 processing elements, and the architectural model was definitely SIMD. Now it rightly includes MIMD architectures, and variations on these two models. It is less right, however, to call systems with 32 processors massively parallel, as has been done in some instances. In this paper our view of architectural models is quite liberal, encompassing heterogeneous computing environments, and we definitely are concerned with systems having thousands of processors.

The workshop produced discussion on a broad range of research issues. We expand on the discussion in selected areas and identify problems and research issues from those areas. The nature of this paper, consistent with the workshop, is not to report research results. It is to identify research that needs to be done. Our interest is not systems targeted for general purpose, high performance computing. It is systems that are in some ways special purpose or application specific. The use of "non-conforming" in the title is meant to indicate systems that differ in substantial ways from the commercial offerings, or that have unique requirements that are not well met by current offerings. We begin by defining terms and concepts.

Two important concepts in this paper are modularity and heterogeneity. By "modular" we mean that a suitable architecture can be achieved by combining a number of building blocks (modules). Each module can be a computer in its own right, and in our context each module could be a homogeneous parallel computer module as well. "Heterogeneous" indicates that these modules can be of different kinds, that is, they can differ in parallelism, control, I/O support, and other aspects of their architecture. By having different kinds of modules it is possible to use a module that fits a certain part of the application very well, and by combining modules we get a very good fit between an application and the architecture for many applications. An abstract view of such a system is given in Figure 1, where there are three modules of different types which communicate with each other and with the external world of peripheral devices, instruments, sensors, actuators, etc.

There are several additional concepts of importance. The notion that a system is "resource adequate" means that computational resources, including I/O, have the necessary power to accomplish the task in an allotted amount of time. "Embedded" systems are just those in which resources are closely coupled to other parts of a system, such as sensors or actuators. Having the capability to configure modular, heterogeneous systems means that we can achieve embedded systems with resource adequacy, which relieves us from many of the real-time resource sharing problems.

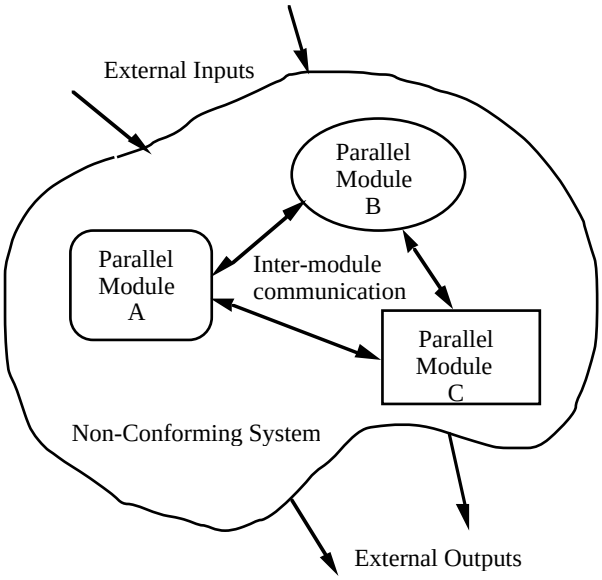

Figure 1. A modular, heterogeneous, parallel system.

By studying two areas where modular and heterogeneous architectures are suggested, specifically instrumentation systems for large experiments and action oriented systems, many important architectural aspects will be found. In addition to the two areas we mention, similar problems and opportunities can be found in applications originating in manufacturing, military, medical, environmental, spaceborne, and other endeavors.

In major experiments, such as the new particle colliders proposed in the U.S. and Europe, data must be acquired from thousands of physically distributed sensors. Data rates are high and real-time processing is needed to select important data and reject the rest. Parallel processing can be used to advantage on the images of particle tracks and energies. However, large packaged parallel machines are not very suitable since processing requirements and the physical environment makes it necessary to modularize and embed processing resources in proximity to the sensors. Other characteristics of this environment, and the research issues it raises, are further discussed in Section 2.

In an action oriented system, sensory, motor, and processing parts, all possibly utilizing neural network principals, are seen as an integrated system capable of interacting with the environment in real-time. Integrated should not mean that there is only one block of computation, instead it should be seen as a number of cooperating smaller blocks. Each block is carrying out different styles of signal processing, e.g. pattern recognition, vector quantization, or error correction. Many blocks are potentially implemented as artificial neural networks (ANN). This modularization corresponds very well to how the brain is organized, where real neurons often can be found to be grouped into larger structures (hundreds of thousands of neurons). In Section 3 we further explore the idea of action oriented systems.

1.1 Demands placed on systems

The intended application areas, and the key concepts, lead to a number of demands on these massively parallel systems. The first demand is *real-time performance*. The implication of this demand is that for each task there must always be enough computational and I/O resources guaranteed. Since hardware is often cheap it is natural to use resource adequacy as a hardware design philosophy. Different tasks require different computing paradigms and system architectures. As a consequence, the final system may be heterogeneous. Therefore, we need to find overall system architectures in which we can still deal with, and guarantee, the real-time demands.

The second demand is *embedded implementations*. Miniaturization and low power consumption are necessary to achieve an embedding of resources. By taking advantage of the advances made in VLSI technology and packaging, e.g. multichip module techniques, the goals of both miniaturization and low power consumption can be met. In many cases embedding also leads to distributed systems, which in turn implies that modularization is required. Communication needs can be critical. Processing is frequently required to be close to sensors and massively parallel I/O becomes an important issue.

The third demand is support for *safety critical functions* and preparedness for *harsh environments*. Fault tolerant processor arrays and communication, and/or fault tolerant computational models are increasingly important for applications where human safety is involved. Alternatively, fault tolerance is needed when applications require computing equipment to operate in places where conditions are harsh or repair is difficult.

The fourth demand is the ability to function in dynamic, *real-world environments*. Adaptability to changing environments, and self-organization in relation to input patterns that have never before been encountered, are necessary functions. An emerging technology in order to achieve such advanced behavior is the application of neural network principles. A way to cope with the complexities involved with advanced systems functioning in natural

environments is to use a multitude of cooperating ANNs, organized in layers and hierarchies. Support for this must then be given in the architecture.

The fifth, and final, demand that we consider here is the need for *new development methods and tools.* Action-oriented systems, as well as other systems where the environment can not be fully modeled, must be developed through interaction. This interaction exists both between the system designer and the system, and between the system and the environment. Developing/training the system on-line, using the real sensors and actuators, must be supported.

1.2 Contrasting system design goals

Unfortunately the problem areas mentioned above, coupled with the system demands, do not fit very well into the current offerings of high performance, parallel, general purpose computers. General purpose computers do not have the need or luxury to address the range of issues of the embedded systems we are investigating. As examples of different design goals and constraints consider:

General Purpose	*versus*	*Embedded Systems*
• Maximum performance		• Resource adequate
• Throughput oriented		• Real-time, time determinism
• Size is of minor concern		• Size is important
• Standard languages like HPF		• Custom programming
• Normal I/O capabilities		• High I/O bandwidth
• Standard data formats		• Data transformations

When the first massively parallel machines were developed each processing element (PE) was very simple, often bit-serial [4, 16, 18]. Many of the organizations that developed such machines now have moved towards highly parallel computers where each PE is much more powerful [13, 19]. This is partly a result of technology advances, and partly a reaction to the market pressure towards high maximum performance and general purpose usage. As a vendor put it at the workshop: "The market is only interested in using 32-bit data".

Certainly the grand challenge problems present strong motivation and incentive to architects and corporations [9]. While all of the problems require very high computational rates, most do not have real-time requirements. The one exception is weather forecasting where response time, although real, is not short. All can be handled using hardware configured as large systems in controlled environments. This has left the field of real-time, embedded, and action-oriented systems without major support. Many of the design goals of a general purpose highly parallel computer do not apply for this class of machines.

This paper suggests that modular, heterogeneous systems will play an important role in the future use of massively parallel processing. Sections 2 and 3 give examples of these systems and describe some unique aspects and environments of their use. Section 4 emphasizes the research issues that must be addressed for success with these special systems that do not conform to current commercial offerings.

2. High performance instrumentation systems

An informative example of massive parallelism in an embedded real-time system occurs in the field of high energy physics (HEP). Particle colliders like the Superconducting Super Collider (SSC) in the U.S., or the Large Hadron Collider (LHC) in Europe produce very large quantities of experimental data in very short time spans. At projected beam collision intervals of 15 ns, one of the sensing instruments for the new colliders will output data at rates in the neighborhood of 10^{13} bytes/s [3]. The problem is to acquire interesting data from the vast quantity produced, and then to find "events" of interest in the data such as particle tracks and peaks of energy. These results are further processed to determine energy, momentum, time duration, and other aspects that represent the physics of the events.

Hierarchical and parallel configurations of computers are used extensively for data acquisition and processing in the instrumentation systems of colliders [8]. Detectors surround the site of collisions in a physically large volume extending approximately 10 meters along the beam axis and 2 meters in diameter. In the terminology of the physicists, the computer structure is in levels of "triggers". For the LHC the first level trigger acts as a filter by identifying regions of interest within the total set of detectors. The second level trigger typically locates particle tracks or peaks of energy. It thus reduces the data but increases the information passed to a third level where the physics of the events is processed. Our own work is at the second level. A massively parallel processing array based on the Blitzen SIMD device [5] is being evaluated for use by CERN in the LHC [7].

2.1 System properties

This instrumentation system displays many of the properties that were described in Section 1. It also provides examples of research issues that must be addressed to realize trigger structures or comparable systems. Note that SSC and LHC are in the early phases of development, with initial experiments expected in the late 1990's, and thus these issues are current and ongoing research issues.

If we consider the requirements of HEP instrumentation, the need for modular and heterogeneous properties becomes apparent. The system is real-time in that collisions occur at definite time intervals and the interesting

data for a collision must be gathered and processed, at least through the second level where it is reduced such that it can be saved by the third level for off-line physics calculations. Collisions occur at 15 ns intervals. It is the responsibility of the first level to determine which collisions are producing potentially interesting data and to pass that reduced amount of data to the second level. A tentative goal for the second level in LHC is to gather and process at a 100 kHz decision frequency. Thus, the real-time processing interval is just 10 μs. To put this in a computer time context, the overhead for sending one message between processors in the Intel Paragon is about 25 μs.

The system is heterogeneous in that different processing technologies and architectures are used at different levels of the hierarchical arrangement. The general structure is shown in Figure 2 [6]. Data rates differ dramatically, as do processing requirements. SIMD parallel processing arrays are being evaluated for the second level, but MIMD is likely for the third level. Analog devices may be used in the first level. Within a level, the processors are homogeneous, but they must communicate with different types of processors in the other levels.

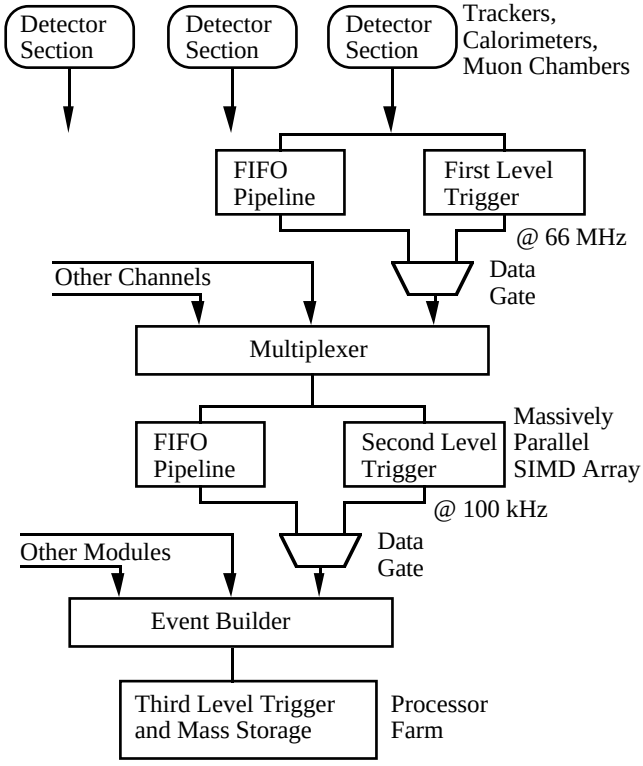

Figure 2. Hierarchical trigger structure with heterogeneous processors.

The system is modular in that processors must be physically near the source of data to accommodate high bandwidth transfers from the sensors and the collision frequency. Since sensors are densely distributed over the 10 m length of the instrument and around the circumference, the three levels in the processing hierarchy are also distributed. There is a tree structure to the hierarchy with the first level being the leaves. Thus modules of SIMD arrays can be used for the second level, and a reduced number of MIMD modules, or possibly just one, for the third level. Essentially, the system is modular since it is necessary to use piecewise coverage of detectors in the collision volume. It is also massively parallel due to the number of processing elements needed to provide the interfaces and processing for those detectors.

A further characteristic of the systems we are studying is that processors are embedded with other electronics and mechanics. This is clearly the case with instrumentation for HEP. It is impractical as well as unworkable to think of running 560,000 wires transferring a cumulative 10^7 megabytes per second, as expected for the transition radiation detector (TRD) [3], from the instrument site to a room with a computer system.

The final characteristic for this example has been introduced in the paragraphs above. There is a high bandwidth I/O requirement. For the TRD example, the major burden is on input with an expected bandwidth of 10^7 MB/s into the first level and $7.7*10^5$ MB/s into the second. In general, some formatting or transformation of data may be necessary. For this example application it is necessary since the instruments produce small outputs that can be represented in a few bits. The bits are gathered into 32-bit words for transmission over HiPPI channels, then must be transformed by a corner turning process for alignment with processing elements.

Figure 3 shows one SIMD module with 1024 PEs as it may be used for the second level trigger. Several such modules are needed for the total system. A region of interest is selected by the first level and delivered via a HiPPI channel to this level. The array of PEs was sized to satisfy I/O and processing rates, and the region is mapped to the array. Results of feature extraction algorithms are passed to the third level. The flow of regions of interest is continuous during an experiment, with a goal of 10 μs for processing each 16 by 240 pixel region.

All of these characteristics are described to emphasize the need for application specific solutions rather than commercially available systems. This became apparent in practice through the process used by CERN to selectively refine the choices for the final system to be used in the LHC. In their process, a progressive set of evaluations is made. They first identified candidate technologies, then specified benchmark tests. Results were presented at a conference held at CERN, in Geneva Switzerland. A candidate commercial massively parallel system had effective decision frequencies that met the desired rate. However, it achieved the good rate only by accumulating a large number of events and processing events in parallel. This pro-

duced long latencies for the earlier events accumulated, followed by a burst of results for all events. The irregular, bursty nature of the processing was not acceptable in the overall design of the instrumentation. The lack of modularity in a fixed commercial system was a detriment in this case. Other researchers using systems with more modularity proposed adequate resources, still using a high degree of parallelism, that more closely matched the problem size. A single event could be processed using parallelism, but events were not accumulated into parallel data sets. Results were produced at the uniform time interval of input data arrival for the events.

2.2 Design problems

Application specific systems like the second level trigger described above frequently require the solution of many interesting problems. There are research issues and research approaches to the problem solutions. We briefly identify some issues that relate to this example, then defer to Section 4 for the main discussion of research direction for these non-conforming massively parallel computers.

An interesting high level design problem is mapping the application, expressed as algorithms and data properties, into an architecture. The problem exists from two points of view: selecting the processing resources and configuring those resources. The richness of devices and architectural arrangements of those devices into solutions provides many parameters that can be exploited during a design phase. Tools and techniques to assist in the mapping could be very useful.

Algorithms and data rates for high energy physics strongly imply a heterogeneous, hierarchical structure. Research is needed in methods for partitioning tasks and communicating through the hierarchy. Fault tolerance and reliability issues must be addressed since experiments are expensive and the system is complex.

Real-time computing in HEP has the constraint of very short response or computation times. Providing real-time for a problem with response times in the microseconds is different from that for one with milliseconds of time. Fortunately, the problem does not require response to a wide set of irregular inputs driven by interrupts. It is real-time from the point of view of needing to complete a fixed routine and perform related I/O within a very short time span. This is well-suited to the resource adequacy notion, assuming sufficiently adequate devices are available.

In the next section we give a second example of applications where standard parallel processing systems are not suitable. It contrasts in several ways to the instrumentation example above, but it also has similarities and presents several of the same research needs.

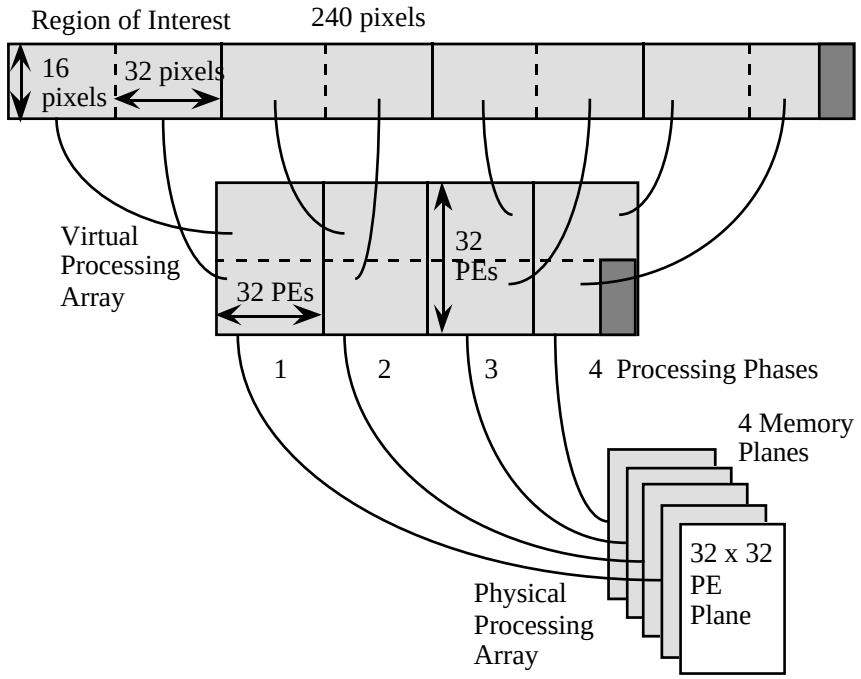

Figure 3. Mapping detector data to a processing array in the second level trigger.

3. Action oriented systems

An action oriented system (AOS) is used as the second example of an application area where modular and heterogenous computers will be needed. Here these aspects are combined with embeddedness, real-time responses, high I/O and internal communication bandwidth, etc., which generates a need for non-conforming massively parallel processing systems.

The concept of action oriented systems (or computing) has been developed by Michael Arbib for many years and is often called "sixth generation computers" by Arbib himself [1]. That particular term has become more appropriate since the introduction of the MITI real world computing (RWC) program in Japan [10], which replaces the much debated fifth generation project. This is because many of the goals of the RWC program are focused on expanding the knowledge in the field of action oriented systems!

Many of the ideas of action oriented systems are drawn from the organization and function of the human brain. The key-concepts of action oriented computing are:

- *Cooperative computing*

Using the brain as a model, we find a number of cooperating areas instead of a large homogeneous information processing facility. Each area is, of course, highly parallel and can for now be approximated as homogeneous in structure. This makes it very natural to suggest a heterogeneous modular computer for simulating action oriented systems. As each module communicates with many other modules in a massively parallel way, this structure puts strong demands on intermodule communication.

- *Perceptual robotics*

An AOS generates its knowledge about the surrounding world by exploration. The system interacts with the environment through a process of Perception -> Decision -> Action. The perception can be sensors for images, speech, tactile information, etc. As these sensors are massive, inexact, and many times incomplete, the system must be highly parallel, robust and fault tolerant. And as real-world information is volatile, the system must work in real-time.

- *Learning*

In contrast to the computers of today which need exact instructions (rules or programs) to function, an action oriented system will base much of its actions on learning. The system should be able to self-organize the information it gains from exploration of the world, and integrate information from many different sources to create an internal model of the world. From past experiences incorporated in the internal model, it should be able to make decisions on what actions are appropriate. Much of this learning will be implemented as artificial neural networks, but certainly any universal or domain knowledge can and should be incorporated in advance (like the laws of Newton).

In Section 1.1 most of the demands of an AOS were summarized. But besides issues mentioned there, the close interaction with sensors and actuators will need attention. This closeness makes it desirable to have the computations take place in the sensors/actuators in a massively parallel fashion. Another issue is development methods for applications using AOSs. As learning instead of programming is emphasized, the possibility to develop/train the system on-line or in-the-loop (using the real sensors and actuators) seems desirable. Still the system must support ways to handle timing constraints in a natural way. Future development environments for AOS should probably be graphically based, using domain specific symbols (hierarchically), and time attenuations [14, 17]

After looking at one early example of an AOS in Section 3.1, we will discuss suitable architectures for AOSs in Section 3.2. Following that, we find that not only will AOS influence research in non-conforming computers, but also the research in the field of artificial neural networks.

3.1 Application areas for action oriented systems

The areas where action oriented systems first will be introduced are in manufacturing, robotics, autonomous vehicles, and the control field in general. As these areas already are action oriented and modular (but not necessarily ANN based) it is not hard to realize that ANN based AOSs are interesting. Often the problems are complex enough to need the complexity of multiple ANNs.

One recent example of an action oriented system is COLUMBUS [20], an autonomous mobile robot developed at CMU. This robot's single goal is to maximize its information of the initially unknown environment. The project has so far concentrated on the algorithms to be used and not so much on the computer architecture to run the algorithms. COLUMBUS uses a mixture of different algorithms (as could be expected of an AOS). There are two separate ANNs for sensor interpretation and confidence estimations. This information is then used to enhance an exploration map at a higher level. By means of a modified dynamic programming algorithm this map is used to decide on an action. The decision is based on where there are unexplored areas and where there are obstacles.

Although it appears that the implementors of COLUMBUS have not concentrated on the architecture, they have produced a modular and distributed implementation, using several SUN SPARC workstations in parallel. As the processing power is not embedded (on the robot) a transmission of sensor and control information by a radio link to and from the robot is needed. By dropping some sensor information, and modifying the dynamic programming algorithm used for planning, it has been possible to reach close to real-time performance (each action taking from 3 to 12

seconds). Even if the authors indicate that they are satisfied with this performance, we feel that a different kind of architecture could help to speed up parts of the system by addressing the problem of embeddedness, I/O performance, and parallelism used.

3.2 Architectures for action oriented systems

The best architecture for an AOS is still an open research question. An architecture we suggested in [17] views the system as a number of *nodes* that communicate through logical *channels*. The real-time concept is supported by demanding time-determinism for all parts in the system. By time-determinism we mean that it should always be possible to determine the execution or cycle time for each computation and communication. To accomplish time determinism we suggest that *local real-time databases* between the nodes and the channels be introduced. The three main concepts of this architecture are described below:

Nodes can be either an I/O interface or a computational entity, or function as a combination. Each node can differ in functionality, and communicate with other nodes via the logical channels. The computation is cyclic [11, 12] and time-deterministic (no interrupts). We expect many of the nodes to be implemented as SIMD computers.

Channels also need to adhere to time-determinism and at the same time give as high communication bandwidth as possible to the communicating nodes. We expect fiber-optics to be used, together with time division multiplexing. If this type of multiplexing is not enough, frequency multiplexing may be considered in addition.

Local real-time databases are needed to store the shared data from other nodes and are updated cyclically from the channels. The data stored in the database reflects the best available information for its node at a certain time.

Figure 4 shows an implementation of this architectural concept. Four Operating Nodes, some incorporating massively parallel I/O and each with a processor array (PE array) connected to a local real-time database (LRTDB), are shown. The Operating Nodes are cyclically controlled by control units (CU). Channels between the nodes are established using time and/or frequency multiplexing on the shared fiber-optic medium.

The figure also shows a Development Node which is connected both to the network of operating Nodes and to a Local Area Network (LAN) of workstations (WS) running the development system. The Development Node may be a PE array (as shown) but may also be another type of computer, but with the same interface to the shared medium. The LAN can be removed without affecting the running system.

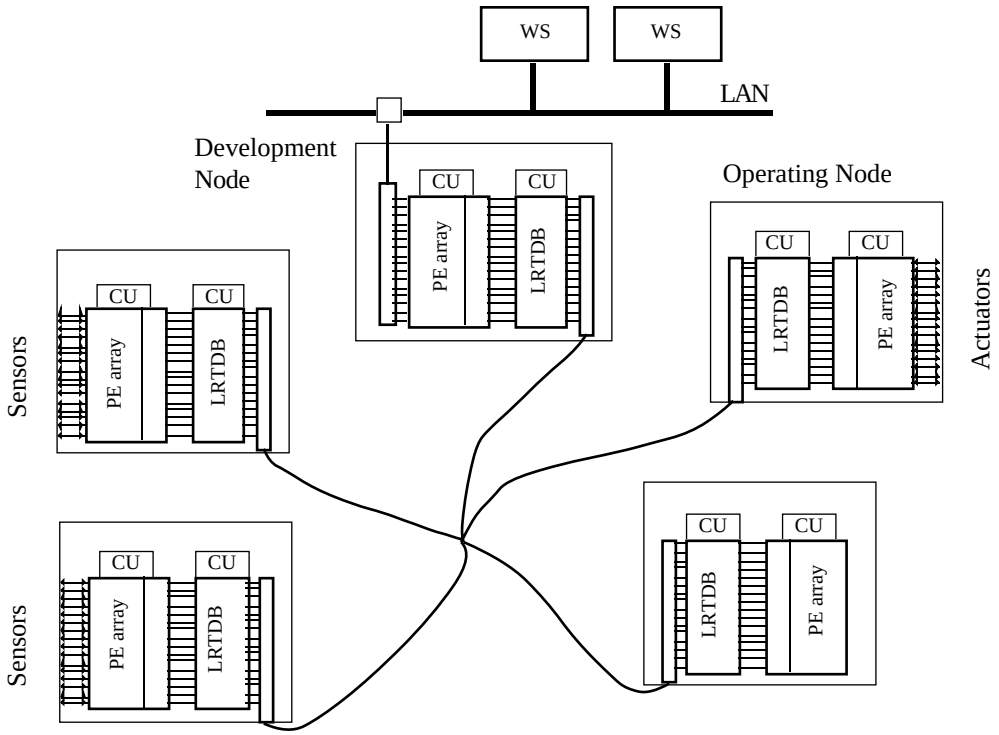

Figure 4. An architecture implementing a modular, heterogenous, parallel system for action oriented computing.

A more detailed description of this architecture concept can be found in [14, 17]. To test these ideas and explore possibilities in the design of the nodes, an experimental system has been built using field-programmable logic devices. Experience from this project, called REMAP [2], will lead to a VLSI design of modules that can be the base for building non-conforming computers, especially in the areas of AOSs. The architecture is intended to be open for the emerging technologies like analog ANN VLSI chips that can do much of the computations close the sensors at extremely high speed.

3.3 AOS Influence on ANN research

During the last ten years a formal explosion of artificial neural network research has lead to a number of different models. Most of the models are naturally parallel and can easily be implemented on highly parallel computers. In a study it has been found that for most ANNs a highly parallel SIMD computer with simple communication (broadcast) is enough [15]. But when it comes to a number of cooperating ANN modules relatively few experiments have been done, and there is no hardware around with the capacity to do real-time simulation of multi-ANN systems big enough to be interesting. Many aspects of multi-ANN are unclear at this time, leading to a need for flexibility in the systems that implement them, to cope with changes in models etc.

Some aspects of ANNs in AOS that need to be addressed are influenced by the real-time aspects. New ANN models need to be developed since real-time creates a need for models that can learn continuously, which is not the case for one of the most popular ANN models (back-propagation learning). And as learning methods using relaxation or structural adaptation are not time-deterministic, it will be hard to use such models for real-time AOS as well. Real-time also means that some types of parallelism in the ANN models [15] can not be used. That is, the training example and training session parallelism are batch oriented and simply are not viable for continuous learning. (We should use the parallelism in weights and nodes instead). Other aspects of ANNs in AOS that need more research are those of planning and creating useful internal world models.

4. Research issues

Very early in our workshop session it became apparent that we could not discuss processor architecture in isolation from the other three workshop topics. They are interrelated. Thus some of the topics below stray from narrow processor issues. Even so, the broader topics all have an impact on processors and must be considered in arriving at processor architecture decisions. Our topics do not include VLSI design issues, even though those also have an impact on processor architecture.

In previous sections we have given examples of modular and heterogenous computers. In each example, general purpose highly parallel computers were not capable of solving the problems within certain performance constraints. Instead we found that a special purpose heterogenous architecture using homogeneous modules was necessary to generate more effective solutions. But before such special systems can become readily available as solutions for a broad class of problems, there are a number of research issues that have to be addressed. Our discussion is representative rather than exhaustive. We highlight some specific issues at the system and processor levels, and then indicate other areas with open research issues.

4.1 Configurable systems

Systems issues are the global concerns that cannot be resolved by considering one PE in isolation, yet they affect each PE's architecture. The systems of interest in this paper are by definition different from those which can be produced in quantity in a commercial manufacturing setting. They are special in some ways and have unique properties. The challenge is to provide the tools, techniques, and methodologies that can lower the cost and improve the quality of systems configured for special applications. From the starting point of a problem specification or an algorithm, coupled with acceptance criteria, how does one arrive at a good system that satisfies the criteria.

An important aspect is that, in the kind of systems we discuss, the modules need not be as much general purpose as processor arrays for "traditional" massively parallel computers. We accept that some modules very strictly follow the SIMD paradigm to be very efficient on some types of computations (and worse on others). For more irregular computations, other paradigms (SPMD, MIMD,...) are used in the modules. That is, we accept heterogeneity in the system. Assuming that our problem is sufficiently large that parallelism becomes advantageous, several more specific issues can be identified:

- How is the overall problem partitioned into modular parts?
- Which style of parallelism (SIMD, MIMD, combination) is appropriate? Can we know from problem parameters?
- Which devices, preferably commercially available, provide the best fit with the problem?
- How do we size a system to provide resource adequacy?
- Does an architecture scale from prototype to full size?
- Can we logically rearrange resources due to lasting changes in the environment or the task to achieve more generality?
- How can we specify benchmarks such that they provide the desired insight to design alternatives?
- Can custom configurations be developed quickly for evaluation?
- How is the evaluation process controlled, given the richness of possibilities with parallel architectures?

4.2 Processor architectures

Individual processor issues are more directly related to the processing power and flexibility of each processor. An architectural decision affecting a processor will co-influence system decisions such as array size. Thus, to do a good processor design, one can not look only at maximum MIPS or FLOPS, but must look at the whole system's performance and the environment in which the system is supposed to operate. Several specific issues are:

- What functional capability is given at each processor?
- What application oriented features for artificial neural networks, associative processing, signal processing, etc. are needed?
- What local control features should be implemented for SIMD processors?
- How is massively parallel I/O incorporated?
- How is memory capacity and access balanced with processing resources?
- What processing granularity is best, from bit-serial processors to full 64-bit widths?
- What interprocessor communication granularity is best?
- In order to achieve maximum performance per watt, what trade-off should be made between clock speed and number of PEs per chip?

4.3 Communications and I/O

As seen in Figure 1, there are communication paths between the system and the external environment, and between modules of the system. If individual modules are parallel processors there is also intra-module communication. Since interconnection networks is the subject area of one of the other workshops, we mention here only the aspects that seem especially important. In general, the problems are all concerned with high bandwidth movement of data between different types of modules with different I/O mechanisms and structures. In many cases, processing is required to be close to sensors or actuators, which may deal with analog signals. Efficient methods for data format conversion and corner turning are needed. Interfaces for HiPPI and other high speed channels must be developed for continuous modes of operation.

4.4 Fault Tolerance

Systems to be used in safety critical and/or harsh environments need to be fault tolerant. Many new possibilities of fault tolerance have emerged with ANN models. This, together with a multi-modular structure of ANN, raises many research issues on how to combine the fault tolerance in ANN structures with fault tolerance in the hardware structure.

- How do we retain the inherent fault-tolerance characteristics of ANNs when we map them onto a processor array?
- What are the methods to distribute fault detection and correction over a large number of modules?
- How can one reason about correctness in time as well as values, in an action oriented framework?

Redundancy and reconfigurability can be used to increase chip yield as well as provide reliability. This becomes increasingly important as VLSI technology allows more processors per chip and die size increases. Research is needed to investigate reconfiguration methods for various interconnection schemes.

4.5 Software development paradigms

As mentioned in Section 3 new development paradigms are needed for non-conforming massively parallel computers. This is especially apparent for AOSs where an on-line or in-the-loop application development method is needed. Some key-concepts for new paradigms will be: graphical interface, data visualization, data parallelism, incremental development on a running system, and software component reuse. All of these concepts need more research, especially concerning their use in systems such as we are describing.

4.6 Artificial neural networks

There are still many challenges for ANN researchers before large modular ANN (AOS) can be built and its function understood. Some of the research issues are stated in Section 3, and many more can be found in the article by Kahaner describing the MITI's real world computing program [10], many of the research issues regarding multi ANNs are listed. For example, the aspects of how the AOS should handle information, its representations, storing and recalling information, integration of multiple information sources, etc. Other research aspects are concerned with the ways learning and self-organization are best carried out. The results of this research have great influence on both system and module architecture of the computing platform

5. Final comments

Identifying research topics has the problem of knowing where to start, and then, where to stop. Given the general area of processor architectures for massively parallel systems, we have chosen to emphasize systems which are unique or special in some way. We refer to these systems as non-conforming since they differ from commercial offerings. The advantage is that they provide a rich environment, one with many degrees of freedom, and perhaps some difficult constraints, for architectural and related research. Two quite different system examples were presented. They were intended to show that massive parallelism can be used in various application areas and to show the need for further research to achieve the desired end results.

Acknowledgments

We would like to acknowledge the support of several organizations for enabling our participation in the Frontiers '92 Workshop and the preparation of this paper. Davis acknowledges support of the U.S. Environmental Protection Agency through Cooperative Agreement CR 820636-01-0, and the North Carolina Supercomputing Center for support during a leave from North Carolina State University (NCSU) and for the provision of computing resources. Nordström acknowledges support of The Board of Postgraduate Studies and Research at Luleå University of Technology through a visiting scholarship to NCSU (920616 – FNTNr 215/91). Svensson acknowledges the support for the REMAP project from The Swedish National Board for Industrial and Technical Development (NUTEK) under contracts No. 900-1583 and 900-1585. And finally we all acknowledge the bright ideas, stimulating discussions, and hard work of the Blitzen groups at NCSU and the University of Padova in Italy, and the whole REMAP group at Halmstad University, Luleå University of Technology, and Chalmers University of Technology.

References

[1] Arbib, M. A. "Schemas and neural network for sixth generation computing." *Journal of Parallel and Distributed Computing*. Vol. 6(2): pp. 185-216, 1989.

[2] Bengtsson, L., A. Linde, T. Nordström, B. Svensson, M. Taveniku and A. Åhlander. "Design and implementation of the REMAP^3 software reconfigurable SIMD parallel computer." *Fourth Swedish Workshop on Computer Systems Architecture*, Linköping, Sweden, 1992.

[3] Bialas, P., J. Chwastowski, P. Malecki and A. Sobala. "Benchmarking with data from the transition radiation detector." (CERN/EAST note 91-11), CERN, Geneva, Switzerland, 1991.

[4] Blank, T. "The MasPar MP-1 Architecture." *Proceedings of COMPCON Spring 90*, pp. 20-24, San Francisco, CA, 1990.

[5] Blevins, D. W., E. W. Davis, R. A. Heaton and J. H. Reif. "Blitzen: A highly integrated massively parallel machine." *Journal of Parallel and Distributed Computing*. Vol. 8: pp. 150-160, 1990.

[6] Bock, R. K. et al. "Embedded Architectures for Second level Triggering in LHC experiments (EAST)." (CERN/DRDC/90-56), CERN, Geneva, Switzerland, 1990.

[7] Centro, C. S., E. W. Davis, P. Ni, D. Pascoli and E. Siliotto. "Results of second level trigger algorithms using the Blitzen parallel machine." *Proceedings of the 1992 Conference on Computing in High Energy Physics*, C. Verkerk and W. Wojcik ed., Annecy, France, 1992.

[8] CHEP 92. *Proceedings of the 1992 Conference on Computing in High Energy Physics*, C. Verkerk and W. Wojcik ed., Annecy, France, 1992.

[9] GC. "Grand challenges: high performance computing and comunnications". *A report by the committee on physical, mathematical, and engineering sciences*. National Science Foundation. Washington, DC. 1992.

[10] Kahaner, D. "Special report: MITI's real world computing program." *IEEE Micro*. (August): pp. 70-79, 1992.

[11] Lawson, H. W. "Cy-Clone: an approach to the engineering of resource adequate cyclic real-time systems." *The Journal of Real-Time Systems*. Vol. 4(1): pp. 55-83, 1992.

[12] Lawson, H. W. and B. Svensson. "An architecture for time-critical distributed/parallel processing." *Euromicro Workshop on Parallel and Distributed Computing*, Gran Canaria, Spain, 1992.

[13] MP2. "Second-generation MPP system boosts speed fivefold: MasPar array 32-bit CPUs." *Electronic Engineering Times*. (October 5): pp. 14, 1992.

[14] Nilsson, K., B. Svensson and P.-A. Wiberg. "A modular, massively parallel computer architecture for trainable real-time control systems." *AARTC'92: 2nd IFAC Workshop on Algorithms and Architectures for Real-Time Control*, Seoul, Korea, 1992.

[15] Nordström, T. and B. Svensson. "Using and designing massively parallel computers for artificial neural networks." *Journal of Parallel and Distributed Computing*. Vol. 14(3): pp. 260-285, 1992.

[16] Potter, J. L. *The Massively Parallel Processor*. MIT Press. Cambridge, Mass. 1985.

[17] Svensson, B., T. Nordström, K. Nilsson and P.-A. Wiberg. "Towards modular, massively parallel neural computer." *Swedish National Conference on Connectionism, SNCC'92*, Skövde, Sweden, 1992.

[18] Thinking Machines Corporation. "Connection Machine, Model CM-2 technical summary." (Version 5.1), T M C Cambridge, Massachusetts, 1989.

[19] Thinking Machines Corporation. "The Connection Machine CM-5 technical summary.", T M C Cambridge, Massachusetts, 1991.

[20] Thrun, S. B. "Exploration and model building in mobile robot domains." *Proceedings of the IEEE International Conference on Neural Networks*, San Francisco, CA, 1993.