N88-16393

Application of Parallel Distributed Processing to Space Based Systems

J.R. MacDonald, H.L. Heffelfinger
TRW, Huntsville Operations
213 Wynn Drive
Huntsville, Alabama 35807

Abstract

This paper explores the concept of using Parallel Distributed Processing (PDP) to enhance automated experiment monitoring and control. Recent VLSI advances have made such applications an achievable goal. The PDP machine has demonstrated the ability to automatically organize stored information, handle unfamiliar and contradictory input data and perform the actions necessary. The PDP machine has demonstrated that it can perform inference and "knowledge operations" with greater speed and flexibility and at lower cost than traditional architectures. Current automated process and control algorithms use knowledge and inference mechanisms to solve problems which would ordinarily require the expertise of the best human practitioners. In applications in which the rule set governing an expert system's decisions is difficult to formulate, PDP can be used to extract rules by associating the information an expert receives with the actions taken. There are many potential applications for very large scale hardware parallelism in the execution of space based process monitor and control systems.

Introduction

The practical possibilities of large scale parallel machines have been significantly enhanced by recent technological advances in VLSI research and production. Relatively low cost and easy access to VLSI hardware has enabled researchers to more closely examine parallel processing problems in general, and the application of neural nets to real world problems. The application of knowledge based systems to on-line, real-time environments such as automated experiment monitoring and control typically demands large complex systems reasoning. Control of these experiment systems stresses the current space based

PRECEDING PAGE BLANK NOT FILMED

* C-3

computer environments well beyond current technology. Size, reliability, and response time constraints of the space environment quickly overload conventional von-Neumann machine knowledge applications. However, the fine grained parallelism made possible by PDP technology has provided an alternative route to space based process monitor and control.

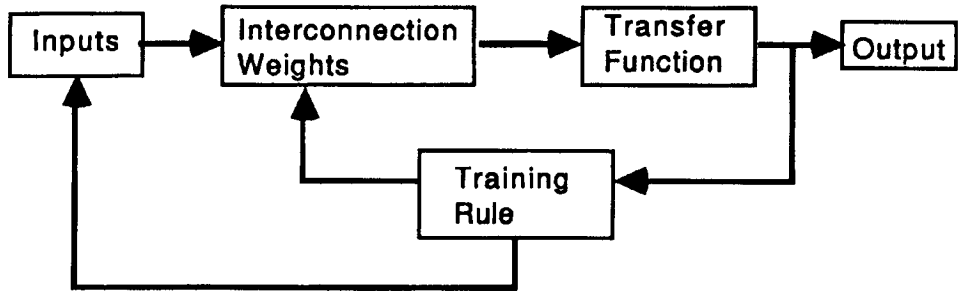
PDP networks are a concept of knowledge representation that consists of a number of processing elements interconnected in a weighted, user-specified fashion. The interconnection weights between the processor nodes are analogous to the memory of a conventional system. Each processing element calculates an output value based on the weighted sum of its inputs. A training rule is used to correlate the input data with the output or desired output (specified by an instructive agent) and adjust the interconnection weights. In this way the network is able to learn patterns or imitate rules of behavior. It is this ability to define and control decision making that would make this sort of system well suited to support space based processing problems. The division of the processing demands among processors makes it possible to achieve linear performance improvements for space based processing applications. This paper will explore PDP network techniques in the application of massively parallel primitive processors to achieve very high speed, fault tolerant, low cost support for the types of problems associated with controlling experiments in a space based environment.

Space Based PDP Architecture

PDP network technology is a concept of computational processing based upon highly interconnected low level processing units. PDP technology seeks to develop and enhance processing capabilities in areas such as real-time high-performance pattern recognition, knowledge processing for inexact knowledge domains, and fast, precise control of VLSI simulation. The aims of this technology are, therefore, clearly related to those of Artificial Intelligence (AI). Over 30 years of research has provided broad processing requirements for pattern recognition, knowledge processing, and simulation of processes. PDP machine research has been directed toward finding solutions to these difficult computing problems.

The PDP architecture is the formal specification of a particular network or bulk system configuration (i.e., the equations of the dynamical system that defines the PDP). This architecture is analogous to a mathematical algorithm or logical procedure being coded and run on a wide variety of computer hardware. The word implementation is reserved for use in describing the engineering process of developing an appropriate PDP architecture and selecting suitable implementation hardware for the specific application.

The PDP conceptual processing abilities are based upon an array of highly interconnected Processing Units (PU)[1,3]. Figure 1 shows a schematic diagram of a PDP network and an individual PU. The network is make up of input units, hidden units, and output units. The hidden units are used to extract progressively more complex features from the input units. This allows more complex tasks to be learned. Note that each PU receives several input signals and has a single output that fans out as input signals to other PUs. Each PU assigns a weight to each input they receive. The sum of the weighted signals determine whether a receiving PU will itself output a signal. Thus this process continues through the PDP machine. The processing cascades through many iterations and the output could be a binary answer to a question or a complicated signal to be transmitted across an external interface. The weights are determined by tuning the system until it consistently produces the right output based upon the system inputs. A typical processing element operation might be:



The PDP adjusts itself by way of feedback control systems that combine stimuli and feedback from the responses to adjust the weights so as to get increasingly correct responses. It is these adaptable, programmable connections that makes the PDP machine special. The PDP machine was initiated in the belief that coupling parallel processing and artificial intelligence could accelerate the rate of progress toward machines that could "learn" and make predictable, weighted decisions within the confines of their knowledge base. The PDP machine encodes knowledge in the connection of its processing units. Each set of connections represents a pattern of values for a few features. Together the features describe environmental states, that is, values that occur in the system's environment. The strengths of the connections encode the frequencies with which each of the different patterns occur in the environment. Thus, the interconnections are used to represent events in the environment. The design goal of a PDP machine for a space based environment should be guided by the following goals:

- Effective use of massively parallel processing.
- Flexibility of interaction with different sensors, actuators, and interfaces.
- Distributed parallel decision-making capabilities.
- Flexibility and ease of expansion of the system capabilities.
- Graceful integration of specific experiment control plans.

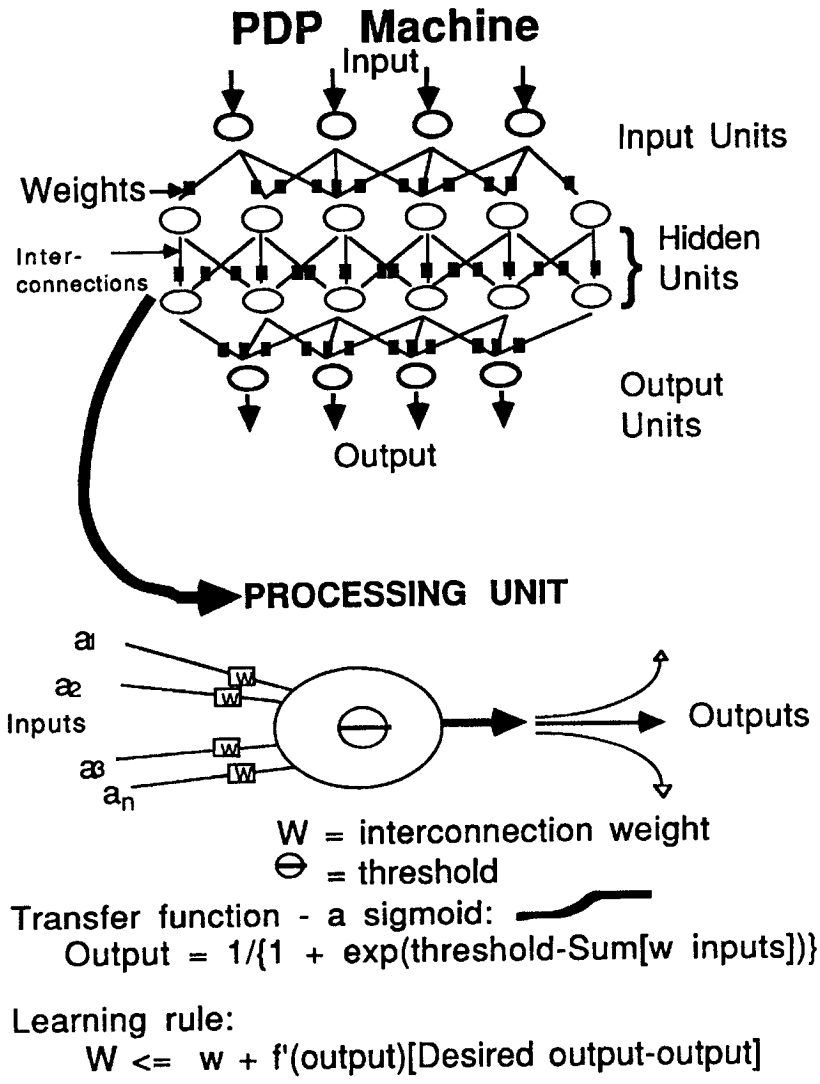


FIGURE 1. Schmatic diagram of a PDP Network

A key approach in the PDP architecture is the use of an adaptive filter. The adaptive filter can accept a real-world analog input and compare it to a stored pattern. The stored pattern represents the desired input level which can then be modified as necessary so that echo-free signals can be produced the correct results. A more abstract argument driving the design of the PDP network is machine efficiency, that is, the optimal utilization of the total computer circuitry. In the more familiar structure of a modern computer (the von Neumann architecture) most of the computer's circuitry in the memory is not in active use most of the time. This is very wasteful from a pure resource management standpoint. The PDP machine addresses this problem by using many processors and memories, but does so

on a much grander scale than most. The parallelism in the PDP machine is extremely fine-grained; it is essentially "data-level parallelism." The fine-grain division of labor and the high speed allows the accomplishment of tasks traditionally outside the operation capacity of conventional systems, among which are constraint search, pattern recognition, and so forth. Even though it fits quite neatly into the definition of an parallel processor system design, the PDP machine goes well beyond most machines in this category in the flexibility of its structure and its amenability to various problem types.

Command and control of typical space based experiments involves both telemetry and status monitoring and the generation of command messages. These tasks are typically carried out in conventional (sequential) computer systems. As systems and environments become more complex, distributed systems become increasingly attractive as well as necessary to achieve higher throughput for a given level of computational power and higher overall system availability. As knowledge-based systems grow in size and scope, they push conventional computing systems to their limits of operation. For space-based experiment control systems, response from a conventional implementation of an expert system may not be practical. Significant performance increases in process monitor and control systems applications could be realized through distributed processing or the use of specialized massively parallel hardware.

Typical space systems computer operations involve monitoring and control processes such as system initialization and shutdown and power system control. Although the performance of tasks such as sensor monitoring, particularly exception monitoring, is often automated, the corrective action - the reaction to anomalies - is typically done by on-board personnel. The exponential increase in system complexity and processing speeds in distributed processing systems pose a serious problem for meaningful, effective human interface as well as timely, effective corrective action. When these factors are coupled with non-linear increases in costs, safety considerations, and longer mission durations, they provide a significant incentive for improved knowledge based system processing concepts and applications. The demands of the space based environment, that is, the real time processing of experiment feed-back and other sensor based data, suggest the necessity for efficient handling of data and telemetry in the event of environment degradation, or sensor failure. The process and control system will need to respond to anomalous events that will be both instantaneous, non-specific, and dependent upon current machine state. The space based processing system must be effective and reliable in its response to both nominal and anomalous events.

A PDP network implementation of an expert system as shown in Figure 2 is well suited for the space based environment. An expert system is designed to explore and symbolically manipulate problems. Expert systems can be distinguished from other artificial intelligence systems in that by design they bring large amounts of knowledge to bear in problem solving. There are two general ways to define an expert systems. One way is by problem domain or

competency. An expert system is a computer system which uses domain-specific knowledge as well as inference procedures to solve problems. The specific problems involved are sufficiently difficult to require significant human expertise in the weighing and evaluation of data. Restated, an expert system operates in a complex domain and is competent at the level of a human expert.

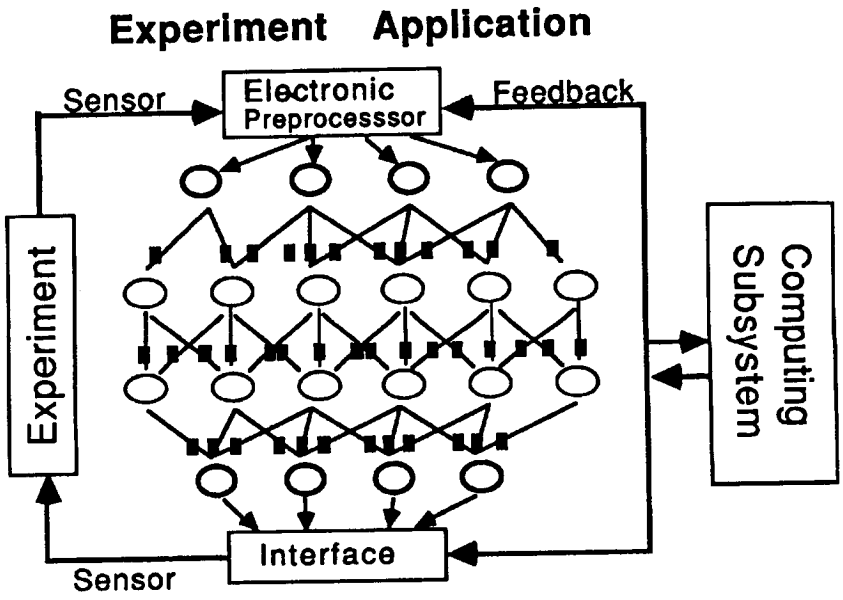


Figure 2. Space Based PDP Expert System

Another way of defining an expert system is by its structure. An expert system consists basically of a knowledge base and a control structure. The knowledge base contains facts about the domain. It also employs heuristics and rules of behavior. The control structure determines the direction of problem solving. In general, control structures provide for goal directed problem solving (solving a problem to reach a certain goal) or data directed problem solving (problem solving that makes use and sense of data available from the domain environment). A related structural aspect of expert systems is a working memory in which interim problem solving steps and other information may be temporarily stored while they are being used in working toward a solution. This structural definition of an expert system is clearly related to PDP networks functional capabilities.

Typical applications of expert modules are in the control and operation of sensors and actuators, interpretation of sensory and feedback data, devising strategies to accomplish proposed tasks and the execution of these strategies. A goal might be to operate complex

space based experiments independent of human intervention for significant periods of time. If this be the case, the long term space based systems goal is to define control mechanisms that will enable integrated experiments to detect error conditions in its working environment and either perform or provide corrective actions. Such systems must be able to interpret and integrate qualitatively different, sometimes incomplete, and sometimes conflicting sensory information. In other words, it must construct an internal model of the real world environment in which the experiment will operate. A general problem-solving technique must be employed. However, specific real-time information must also be integrated in the execution of plans to solve a given task. It must be a control system which is responsible for and capable of independent control execution through parallel and coordinated control of multiple sensors and actuators.

The system must evaluate the outputs from sensors and update the state of the experiment actuators according to the rules established by the principle investigator. Various rules are said to "fire" based on input. An example of a rule for an Earth-bound experiment environment might be " If temperature greater than 150 degrees then power up fan". The integrated computer system is in a constant state of sensing and updating the state of an experiment. A simple experiment environment might involve 1000 such rules. Because of memory requirements and execution speed, even this relatively small knowledge base would involve significant overhead in a conventional von-Neumann machine and could quickly overload the system. The PDP machine offers another method of implementation of constraint based rules. This alternative provides the speed necessary to respond to the space based system needs.

PDP Knowledge Acquisition Paradigm

In the preceding sections we have discussed a parallel processing architecture and its ability to build a knowledge base - to "learn." This learning process is, of course, a complex and controversial problem. Learning algorithms have been proposed as a way to program massively parallel processors [1,2,3,4,5]. Experiment control system programs using appropriate learning algorithms can be automatically decomposed into small sub-tasks. It is these sub-tasks that provide an opportunity to distribute processing across parallel processing units, which translates into the connections of a PDP machine.

The PDP machine assigns a weight to each input it receives; the sum of the weighted signals determines whether a receiving PU will itself fire a pulse, which in turn triggers other PU's. This process cascades through many iterations, and the result, is the output. The PDP machine consist of a number of PU's interconnected in a weighted, user-specified fashion. Each PU calculates an output value based on the weighted sum of its inputs. To program the PDP machine, the input data is correlated with the output or desired output

using a learning rule that will adjust the interconnection weights. In this way the machine learns patterns or imitates rules of behavior and decision making the allows the PDP machine to support space based processing. PDP model as presented in Figure 1 typically consists of many simple processing units that interact using weighted connections. Each unit has a "state" or "activity level", that is determined by the input received from other units in the network. The threshold term can be eliminated by giving every unit an extra input connection whose activity level is fixed. The weight on this special connection is the negative of the threshold, and it can be learned in just the same way as other weights.

The particular PDP machine architecture discussed here is a variation of the Rummelhart et. al. [3] multi-layer perceptron employing the generalized delta rule (GDR). The GDR provides a method of modifying any weight in a network, based on locally available information, so as to implement a gradient descent process that searches for those weights that minimize the error at the output units. The PDP system can learn to associate arbitrary input/output pairs by use of the generalized delta rule. Using this the rule, neural networks can learn to compute arbitrary input/output functions. The mathematics of this learning approach can answer many questions about the weight, adjustment, and summation of the interconnection weights. It can also provide some insight into noise sensitivity, feedback, and system layering. The GDR learning procedure is a generalization of the delta rule procedure that works for networks which have layers of hidden units between the input and output units. Multilayer networks can compute more complicated functions than networks that lack hidden units. However the price that must be paid is a slower learning process as the system explores the possible ways of using the hidden units.

The application of the generalized delta rule as presented in Figure 3 involves two phases. First the input is presented and propagated forward through the network to compute the output value for each unit. This output is then compared with the targets, resulting in an error signal for each output unit. The second phase involves a backward pass through the network during which the error signal is passed to each unit in the network and the appropriate weight changes are made. This second, backward pass allows the recursive computation of the error signal as indicated above. The first step is to compute the error signal for the output units and all of the hidden units. Notice that computation performed during the backward pass is very similar in form to the computation performed during the forward pass (though it propagates error derivatives instead of activity levels, and it is entirely linear in the error derivatives). The GDR generates a gradient descent method for finding weights in any feed-forward network with semilinear units. The learning procedure involves the presentation of a set of pairs of input and output patterns. The system first uses the input vector to produce its own output vector. Then this is compared to the desired output, or target vector. If there is no difference, no learning takes place. If any difference exists, the weights are changed to reduce the difference.

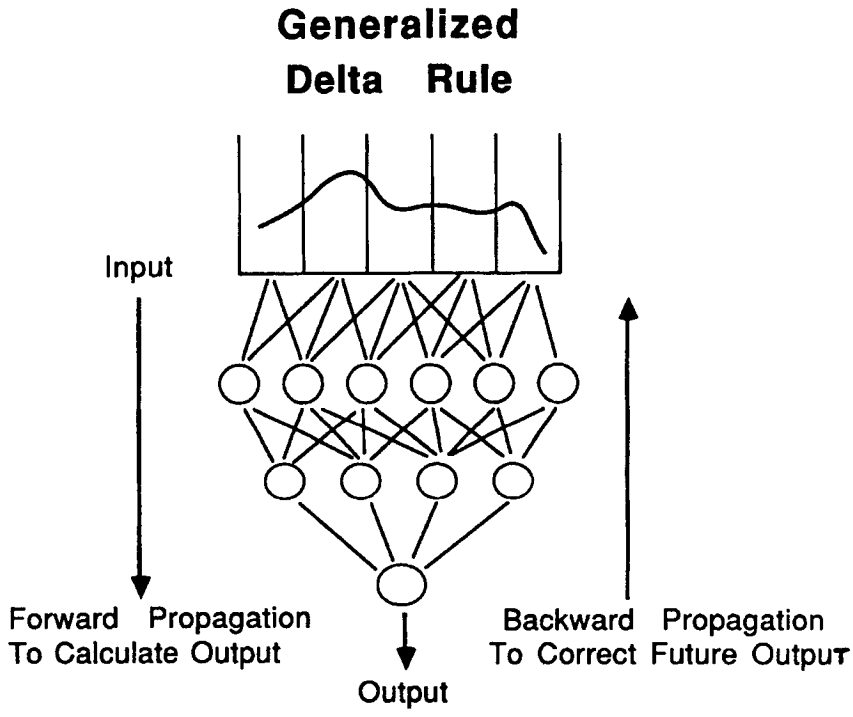


Figure 3. PDP Learning Procedure

When weight change increments are sufficiently small, this learning procedure is guaranteed to find the set of weights that gives the least mean squared error. The delta rule essentially implements gradient descent in sum-squared error for linear activation functions. The central idea of the generalized delta rule is that these derivatives can be computed efficiently by starting with the output layer and working backwards through the layers. The weight on each line should be changed by an amount proportional to the product of an error signal available to the unit receiving input along that line and the output unit sending activation along that line.

From the above learning methodology we see that the PUs in a PDP machine are trained by the cyclic input and output of data vectors. In this way a computer operating in the batch mode can be very effective in training the system. However there is a clear need to provide a real-time interface with a human in order to effect more particular training [5]. The iterative process through which a PDP machine learns is not suited for human interaction. Therefore there is a clear need to both enhance the man-machine interface and develop more efficient data/response patterns in a PDP architecture.

Conclusion

System designs based on the traditional von Neumann approach will be considerably slower than a systems based on the PDP machine simply because of the limited bandwidth of the memory-processor connection. These traditional systems cannot match the flexibility in function that the software-programmable connections, which are the hallmark of the PDP machine, allow. Networks of larger processors, the MIMD concept, can possibly outperform a PDP machine on computation-intensive problems; however, in constraint-intensive applications, they suffer from the same problem as the von Neumann designs. Traditional SIMD machines, such as systolic and pipeline machines, suffer from the problem of requiring a particular structure to solve a problem. Again, this problem is overcome by the flexibility of the PDP machine's communication network.

PDP machines employing fine grained parallelism consist of a number of processing elements interconnected in a weighted, user specified fashion. The interconnection weights act as memory for the system. Each processing element calculates an output value based on the weighted sums of its inputs. In addition, the input data is correlated with the output (or desired output) through use of a training rule that adjusts the interconnection weights. In this way, the network learns patterns or imitates rules of behavior and decision making. Process information is not obtained by passing through a normal process and control algorithm, but is provided by the interconnection structure of the network itself. It is our belief that PDP machines can in fact support high-speed execution of a very large class of space based process monitor and control systems. The number of processing elements in the interconnection network of a PDP machine makes overall network reliability and fault tolerance a key consideration in space based systems. Computer systems employing fine grained parallelism can provide an approach to a number of long standing problems involving space based experiment applications.

References

1) Hitton, G.E. (1987). *Connectionist Learning Procedures.* (Tech. Rep. No. CMU-CS-87-155). Pittsburgh, PA: Carnegie-Mellon University, Department of Computer Science.

2) Hopfield, J.J., Tank, D.W. (1985) *"Neural" computation of decisions in optimization problems.* Biological Cybernetics, 52, 141-152.

3) Rumelhart, D.E., McClelland, J.L., et. al. (1986). *Parallel Distributed Processing.* Cambridge, MA: MIT Press/Bradford.

4) Sejnowski, T.J., Rosenberg, C.R. (1986). *NETtalk: A Parallel Network that learns to Read Aloud.* (Tech. Rep. No. JHU/EECS-86/01). Baltimore, MD: John Hopkins University.

5) Shepanski, J.F., Macy, S.A. (1986). *Manual Training Techniques of Autonomous Systems Based on Artificial Neural Networks.* Redondo Beach, CA: TRW, Unpublished manuscript.