PASM: A RECONFIGURABLE PARALLEL SYSTEM FOR IMAGE PROCESSING

Howard Jay Siegel, Thomas Schwederski, Nathaniel J. Davis IV, James T. Kuehn

PASM Parallel Processing Laboratory
School of Electrical Engineering
Purdue University
West Lafayette, IN 47907 USA
July 1984

Abstract

PASM is a multifunction partitionable SIMD/MIMD system being designed at Purdue for parallel image understanding. It is to be a large-scale, dynamically reconfigurable multimicroprocessor system, which will incorporate over 1,000 complex processing elements. Parallel algorithm studies and simulations have been used to analyze application tasks in order to guide design decisions. A prototype of PASM is under construction (funded by an equipment grant from IBM), including 30 Motorola MC68010 processors, a multistage interconnection network, five disk drives, and connections to the Purdue Engineering Computer Network (for access to peripherals, terminals, software development tools, etc.). PASM is to serve as a vehicle for studying the use of parallelism for performing the numeric and symbolic processing needed for tasks such as computer vision. The PASM design concepts and prototype are overviewed and brief examples of parallel algorithms are given.

I. Introduction

This is an overview of the design for the thousand-processor PASM system and the 25-processor prototype being built to validate the system design concepts. One way to do image processing faster is through the use of parallelism. Different modes of parallelism can be employed in a computer system. The *SIMD (single instruction stream - multiple data stream)* mode [Fly66] typically uses a set of N processors, N memories, an interconnection network, and a control unit (e.g., Illiac IV [Bou72], STARAN [Bat77], CLIP4 [Fou81], MPP [Bat82]). The control unit broadcasts instructions to the processors and all active (enabled) processors execute the same instruction at the same time. Each processor executes instructions using data taken from a memory with which only it is associated. The interconnection network allows interprocessor communication. An *MSIMD (multiple-SIMD) system* is a parallel processing system which can be structured as one or more independent SIMD machines of various sizes (e.g., MAP [Nut77]). The Illiac IV was originally designed as an MSIMD system [Bar68]. The *MIMD (multiple instruction stream - multiple data stream)* mode [Fly66] typically consists of N processors and N memories, where each processor can follow an independent instruction stream (e.g., C.mmp [WuB72], Cm* [SwF77], Ultracomputer [GoG83]). As with SIMD architectures, there is a multiple data stream and an interconnection network. A ***partitionable SIMD/MIMD system*** is a parallel processing system which can be structured as one or more independent SIMD and/or MIMD machines of various sizes (e.g., TRAC [SeU80]).

This paper was prepared for the proceedings of the Workshop on Algorithm-guided Parallel Architectures for Automatic Target Recognition, sponsored by the DARPA Tactical Technology Office (TTO), the Naval Research Laboratory (NRL), and the Army Night Vision and Electro-Optics Laboratory (NVL).

PASM is a partitionable SIMD/MIMD machine being designed at Purdue University to be a large-scale dynamically reconfigurable multimicroprocessor system [SiS81]. It is a special-purpose system aimed at exploiting the parallelism of image understanding tasks. PASM is being developed using a variety of problems in image processing and pattern recognition to guide the design choices. It can also be applied to related areas such as speech understanding and biomedical signal processing.

PASM is to serve as a research tool for experimenting with parallel processing. The design attempts to incorporate the needed flexibility for studying large-scale SIMD and MIMD parallelism, while keeping system costs "reasonable." Portions of PASM have been simulated and a prototype is under development.

In Section II, the PASM architecture is overviewed. Section III gives some examples of how PASM can be used for image processing. The PASM prototype is described in Section IV. The advantages of some of the features of PASM are discussed in Section V. Selected references for further reading about PASM appear at the end of this paper.

II. The PASM Architecture

A block diagram of the basic components of PASM is shown in Fig. 1. The *Parallel Computation Unit* (Fig. 2) contains $N=2^n$ processors, N memory modules, and an interconnection network. The *processors* are microprocessors that perform the actual SIMD and MIMD computations. The *memory modules* are used by the processors for data storage in SIMD mode and both data and instruction storage in MIMD mode. Each PE

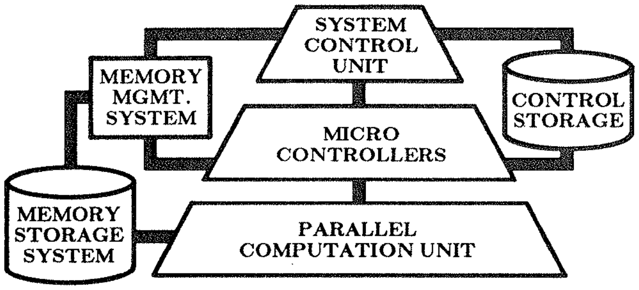

Fig. 1. Block diagram overview of PASM.

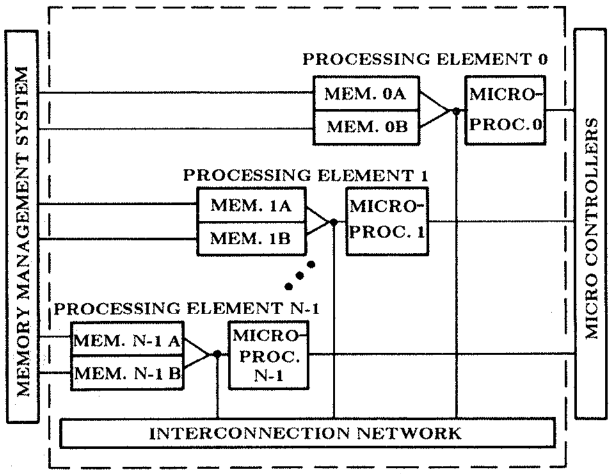

Fig. 2. Parallel Computation Unit.

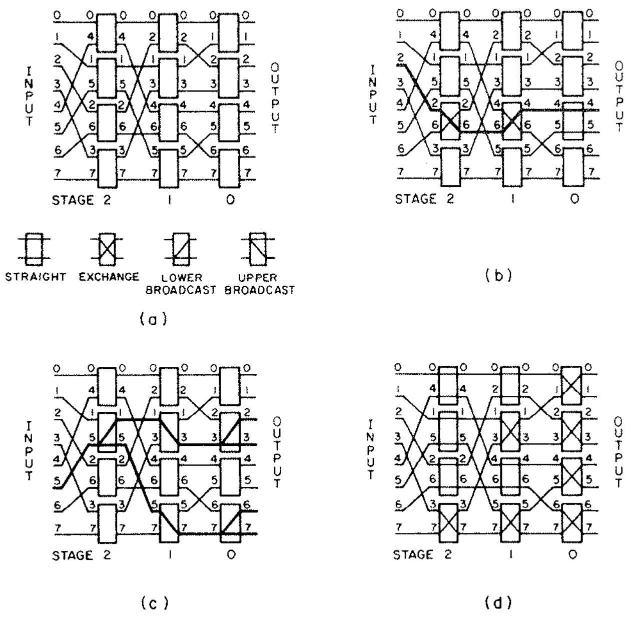

Fig. 3. (a) Generalized Cube topology, shown for N=8. (b) Example one-to-one connection (input 2 to output 4). (c) Example broadcast connection (input 5 to outputs 2, 3, 6, and 7). (d) Example permutation connection (input i to output i+1 mod N).

can operate in both the SIMD and MIMD modes of parallelism. A memory module is connected to each processor to form a processor - memory pair called a *Processing Element (PE)*. The N PEs are numbered from 0 to N−1 and each PE knows its number (address). A pair of memory units is used for each memory module to allow data to be moved between one memory unit and secondary storage (the Memory Storage System) while the processor operates on data in the other memory unit. A PASM N=16 prototype will use Motorola MC68010 processors; the final N=1024 system, which the architecture is designed for, will employ custom VLSI processors specially designed for parallel image processing. The *interconnection network* provides a means of communication among the PEs. Two types of multistage interconnection networks are being considered for PASM: the Generalized Cube [SiM81b] and the Augmented Data Manipulator (ADM) [SiM81a]. The ADM network is more flexible, but is more complex. The Extra Stage Cube network [AdS82] is a fault-tolerant variation of the Cube which is planned for inclusion in the PASM prototype. Features of the Generalized Cube network will be described to familiarize the readers with the properties of multistage networks.

The *Generalized Cube* network is representative of the multistage cube-type class of networks which include the baseline [WuF80], delta [Pat81], Extra Stage Cube [AdS82], indirect binary n-cube [Pea77], omega [Law75], STARAN flip [Bat76], and SW-banyan (S=F=2) [GoL73]. The Cube has N inputs and N outputs. It is shown in Fig. 3a for N=8. PE i, $0 \leq i < N$, would be connected to input port i and output port i of the unidirectional network. The Cube topology has $n = \log_2 N$ stages, where each stage consists of a set of N lines connected to N/2 interchange boxes. Each *interchange box* is a two-input, two-output device. The labels of the input/output *(I/O)* lines entering the upper and lower inputs of an interchange box are used as the labels for the upper and lower outputs, respectively. Each interchange box can be set individually to one of the four legitimate states shown in Fig. 3a. Figs. 3b, c, and d illustrate one-to-one, broadcast, and permutation connections, respectively. Note that many one-to-one and/or broadcasts can occur simultaneously.

The connections in this network are based on the cube interconnection functions [Sie77, Sie79]. Let $P = p_{n-1} \cdots p_1 p_0$ be the binary representation of an arbitrary I/O line label. Then the n cube interconnection functions can be defined as:

$$\text{cube}_i(p_{n-1} \cdots p_1 p_0) = p_{n-1} \cdots p_{i+1} \bar{p}_i p_{i-1} \cdots p_1 p_0$$

where $0 \leq i < n$, $0 \leq P < N$, and $\bar{p}_i$ denotes the complement of p_i. This means that the cube$_i$ interconnection function connects P to cube$_i$(P), where cube$_i$(P) is the I/O line whose label differs from P in just the i-th bit position. Stage i of the Cube topology contains the cube$_i$ interconnection function, i.e., it pairs I/O lines that differ only in the i-th bit position.

Routing tags are used as headers on messages; they (1) control each interchange box individually, and (2) allow network control to be distributed among the PEs. The n-bit routing tag for one-to-one connections is computed from the input port number and desired output port number. Let S be the source address (input port number) and D be the destination address (output port number). Then the routing tag $T = S \oplus D$ (where "$\oplus$" means bitwise "exclusive-or"). Let $t_{n-1} \ldots t_1 t_0$ be the binary representation of T. An interchange box in the network at stage i need only examine t_i. If $t_i=1$, an exchange is performed, and if $t_i=0$, the straight connection is used. For example, if $N=8$, $S=010$, and $D=100$, then $T=110$. The corresponding stage settings are exchange, exchange, straight (see Fig. 3b). Because the exclusive-or operation is commutative, the incoming routing tag is the same as the return tag. Since the destination PE has the routing tag to the source PE, it is easy to perform handshaking if desired. The address of the source PE can be computed by the destination PE using $S = D \oplus T$. Routing tags that can be used for broadcasting data are an extension of this scheme [SiM81b].

The *partitionability* of a network is its ability to divide the system into independent subsystems of different sizes. The Cube network can be partitioned into independent subnetworks of various sizes, where each subnetwork of size $N' \leq N$ will have all of the connection properties of a Cube network built to be of size N'. In PASM, the partitioning is accomplished by requiring that the addresses of all of the I/O ports in a partition of size 2^i agree (have the same values) in their low-order n−i bit positions. For example, in Fig. 4 subnetwork A consists of ports 0, 2, 4, and 6, and

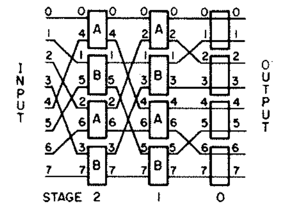

Fig. 4. Cube network of size eight partitioned into two subnetworks of size four based on the low-order bit position.

subnetwork B consists of ports 1, 3, 5, and 7. All ports in subnetwork A have a 0 in the low-order bit position; all ports in subnetwork B have a 1 in the low-order bit position. By setting all of the interchange boxes in stage 0 to straight, the two subnetworks are isolated. This is because stage 0 is the only stage which allows input ports which differ in their low-order bit to exchange data. Each subnetwork can be separately further subdivided, resulting in subnetworks of various sizes. The routing tag scheme can be used to enforce the partitioning by logically AND-ing the tags with masks to force to 0 tag positions which correspond to interchange boxes which should be forced to the straight state. The network partitioning property allows the set of PASM PEs to be partitioned into independent virtual machines of various sizes.

The *Micro Controllers (MCs)* (Fig. 5) are a set of microprocessors which act as the control units for the PEs in SIMD mode and orchestrate the activities of the PEs in MIMD mode. There are $Q=2^q$ MCs, physically addressed (numbered) from 0 to Q−1. Each MC controls N/Q PEs. PASM is being designed for Q=32 (Q=4 in the prototype). The MCs are the multiple control units needed in order to have a partitionable SIMD/MIMD system. Each MC memory module consists of a pair of memory units so that memory loading and computations can be overlapped. In SIMD mode, each MC fetches instructions and common data from its memory module, executing the control flow instructions (e.g. branches) and broadcasting the data processing

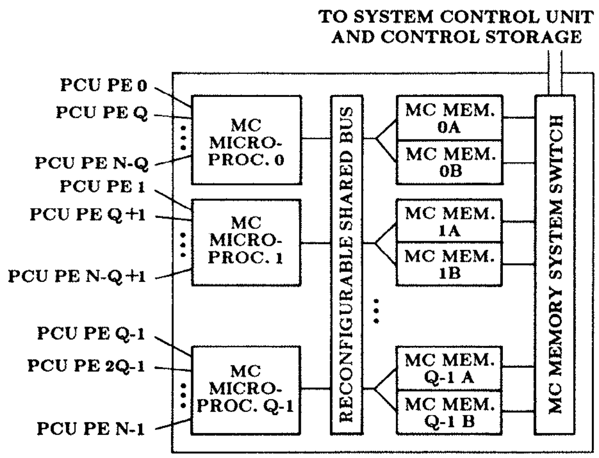

Fig. 5. PASM Micro Controllers (MCs). "PCU" is Parallel Computation Unit.

instructions to its PEs. In MIMD mode, each MC gets from its memory instructions and common data for coordinating its PEs.

The physical addresses of the N/Q processors which are connected to an MC must all have the same low-order q bits so that the network can be partitioned. The value of these low-order q bits is the physical address of the MC. A virtual SIMD machine of size RN/Q, where $R=2^r$ and $0 \leq r \leq q$, is obtained by having R MCs use the same instructions and synchronizing the MCs. The physical addresses of these MCs must have the same low-order q−r bits so that all of the PEs in the partition have the same low-order q−r physical address bits. Similarly, a virtual MIMD machine of size RN/Q is obtained by combining the efforts of the PEs associated with R MCs which have the same low-order q−r physical address bits. In MIMD mode, the MCs may be used to help coordinate the activities of their PEs. Q is the maximum number of partitions allowable, and N/Q is the size of the smallest partition.

The MC processors and MC memories are connected by a shared reconfigurable ("shortable") bus [ArP76, KaK79], as shown in Fig. 6. The MCs must be ordered on the bus in terms of the bit reverse of their addresses due to the partitioning rules. The MC connection scheme provides more program space for jobs using multiple MCs and provides a degree of fault tolerance, since known-faulty MC memory modules could be ignored.

The PEs within each partition are assigned *logical addresses*. Given a virtual machine of size RN/Q, the processors and memory modules for this partition have logical addresses (numbers) 0 to (RN/Q)−1, $R=2^r$, $0 \leq r \leq q$. The logical number of a PE is the high-order r+n−q bits of its physical number. Similarly, the MCs assigned to the partition are logically numbered (addressed) from 0 to R−1. For R>1, the logical number of an MC is the high-order r bits of its physical number. The PASM language compilers and operating system will be used to convert from logical to physical addresses, so a system user will deal only with logical addresses.

A *masking scheme* is used in SIMD mode for determining which PEs will be active, i.e., execute instructions broadcast to them by their MC. PASM will use PE address masks and data conditional masks.

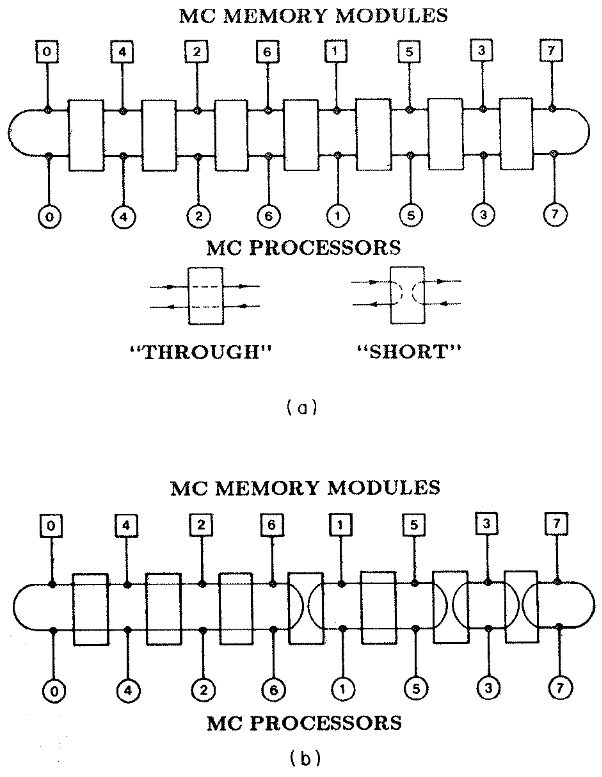

Fig. 6. (a) Reconfigurable shared bus scheme for interconnecting MC processors and MC memory modules, shown for Q=8, where each box can be set to "through" or "short." (b) Bus set for MCs 0, 2, 4, and 6 forming one virtual machine, MCs 1 and 5 forming a second virtual machine, MC 3 forming a third virtual machine, and MC 7 forming a fourth virtual machine.

The *PE address masking* scheme uses an n-position mask to specify which of the N PEs are to be activated. Each position of the mask will contain either a 0, 1, or X ("don't care") and the only PEs that will be active are those whose address matches the mask: 0 matches 0, 1 matches 1, and either 0 or 1 matches X. Square brackets denote a mask. Superscripts are used as repetition factors. For example, "MASK $[X^{n-1}0]$" activates all even-numbered PEs and "MASK $[0^{n-i}X^{i}]$" activates PEs 0 to $2^{i}-1$. A *negative PE address mask* is similar to a regular PE address mask, except that it activates all those PEs which do not match the mask. Negative PE address masks are prefixed with a minus sign to distinguish them from regular PE address masks. For example, for N=8, "MASK [-0X1]" activates all PEs except 1 and 3. PE address masks are specified in the SIMD program.

Data conditional masks are the implicit result of performing a conditional branch dependent on local data in an SIMD machine environment, where the result of different PEs' evaluations may differ. They are used when the decision to enable and disable PEs is made at execution time. As a result of a conditional *where statement* of the form

where <data–condition> *do* · · · *elsewhere* · · ·

each PE will set a flag to activate itself for either the "do" or the "elsewhere," but not both. The execution of the "elsewhere" statements must follow the "do" statements; i.e., the "do" and "elsewhere" statements cannot be executed simultaneously. For example, as a result of executing the statement:

where A < B *do* C ← A *elsewhere* C ← B

each PE will load its C register with the minimum of its A and B registers, i.e., some PEs will execute "C ← A," and then the rest will execute "C ← B." This type of masking is used in such machines as the Illiac IV [Bar68] and PEPE [Cra72]. "Where" statements can be nested using a run-time control stack.

There are instructions which examine the collective status of all of the PEs of a virtual SIMD machine, such as "if any," "if all," and "if none." These instructions change the flow of control of the program at execution time depending on whether any or all processors in the virtual SIMD machine satisfy some condition. For example, if each PE is processing data from a different section of an image, but all PEs are looking for enemy tanks, it is desirable to know "if any" of the PEs have discovered a tank. This requires communication among the MCs comprising the virtual SIMD machine. There is a set of buses shared by MCs for this purpose.

Control Storage contains the programs for the MCs. The loading of programs from Control Storage into the MC memory units is controlled by the System Control Unit.

The *Memory Storage System* provides secondary storage space for the Parallel Computation Unit for the data files in SIMD mode, and for the data and program files in MIMD mode. It consists of N/Q independent *Memory Storage Units*, numbered from 0 to (N/Q)−1. Each Memory Storage Unit is connected to Q PE memory modules. For $0 \leq i < N/Q$, Memory Storage Unit i is connected to those PE memory modules whose physical addresses have the value i in their n−q high-order bits. Recall that, for $0 \leq k < Q$, MC k is connected to those PEs whose physical addresses have the value k in their q low-order bits. This is shown for N=32 and Q=4 in Fig. 7.

A virtual machine of RN/Q PEs, $R = 2^{r}$, $0 \leq r \leq n$, logically numbered from 0 to (RN/Q)−1, requires only R parallel block loads if the data for the PE memory module whose high-order n−q logical address bits equal i is loaded into Memory Storage Unit i. This is true no matter which group of R MCs (which agree in their low-order q−r physical address bits) is chosen. As an example, consider Fig. 7, and assume a virtual machine of size 16 is desired. The data for the PE memory modules whose logical addresses are 0 and 1 is loaded into Memory Storage Unit 0, for memory modules 2 and 3 into Unit 1, etc. Assume the partition of size 16 is chosen to consist of the PEs connected to MCs 1 and 3. Given this assignment of MCs, the PE memory module whose physical address is 2i+1 has logical address i, $0 \leq i < 16$. The Memory Storage Units first simultaneously load PE memory modules physically addressed 1, 5, 9, 13, 17, 21, 25, and 29 (logically addressed 0, 2, 4, 6, 8, 10, 12, and 14), and then simultaneously load PE memory modules physically addressed 3, 7, 11, 15, 19, 23, 27, and 31 (logically addressed 1, 3, 5, 7, 9, 11, 13, and 15). No matter which pair of MCs is chosen (i.e., MCs 1 and 3, or MCs 0 and 2), only two parallel block loads are needed. This same approach can be taken if only $(N/Q)/2^{d}$ distinct Memory Storage Units are available, where $0 \leq d \leq n-q$. In this case, however, $R2^{d}$ parallel block

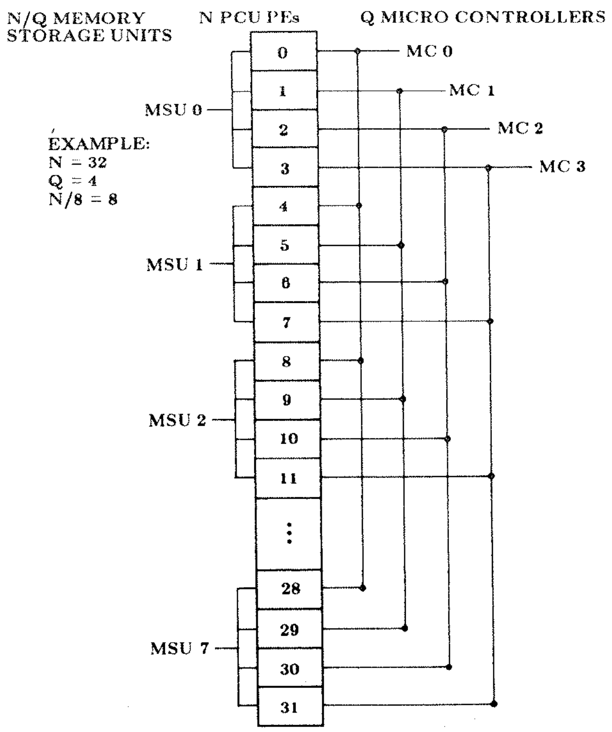

Fig. 7. Organization of the Memory Storage System, shown for N=32 and Q=4. "MSU" is Memory Storage Unit. "PCU" is Parallel Computation Unit.

loads will be required instead of just R to load a virtual machine of RN/Q PEs.

The *Memory Management System* controls the transferring of files between the Memory Storage System and the PEs. It is composed of a separate set of microprocessors dedicated to performing tasks in a distributed fashion. This distributed processing approach is chosen in order to provide the Memory Management System with a large amount of processing power at low cost. The division of tasks chosen is based on the main functions which the Memory Management System must perform, including: (1) generating tasks based on PE load/unload requests from the System Control Unit; (2) scheduling Memory Storage System data transfers; (3) controlling input/output operations involving peripheral devices and the Memory Storage System; (4) maintaining the Memory Management System file directory information; and (5) controlling the Memory Storage System bus.

The *System Control Unit* is responsible for the overall coordination of the activities of the other components of PASM. The types of tasks the System Control Unit will perform include program development, job scheduling, and coordination of the loading of the PE memory modules from the Memory Storage System with the loading of the MC memory modules from Control Storage. By carefully choosing which tasks should be assigned to the System Control Unit and which should be assigned to other system components (e.g., the MCs and Memory Management System), the System Control Unit can work effectively and not become a bottleneck. For the N=1024 PASM, the System Control Unit may consist of several processors in order to perform all of its functions efficiently. In the N=16 prototype, the System Control Unit is a microprocessor and the program development functions are performed by the host computer network.

III. Parallel Image Algorithms

A number of SIMD and MIMD algorithms to perform common image processing tasks have been developed by our group. Image processing algorithms which have been structured for parallel execution include: image smoothing, histogramming, 2-D FFT calculation, local area histogram equalization, local area brightness and gain control, feature extraction, maximum likelihood classification, contextual statistical classification, image correlation (convolution, filtering), block truncation coding, resampling, rectification, rotation, translation, scaling, elevation/location determination, median filtering, Sobel edge sharpening, clustering feature enhancement, scene segmentation, Karhunen-Loeve transformation, 3-D shape analysis using Fourier descriptors, and a computer vision task including edge labeling, perimeter, area, center of mass, and hole count computations. These have been analyzed in terms of machine-size/problem-size relationships, processor capability needed for efficient execution, memory requirements, and inter-PE communication requirements. Different algorithm strategies have been explored and compared. From these studies, we are assessing the ways in which parallelism can be used for vision. Current research includes the development of a simulator for a parallel implementation of a LISP interpreter for use in image understanding applications.

In this section three brief examples of parallel image processing algorithms are given. The first is a simple SIMD image smoothing algorithm, the second a more complex SIMD histogramming algorithm, and the third an SIMD/MIMD contour extraction algorithm. The contour extraction algorithm study was done as part of ongoing research at Purdue in the area of automatic target recognition [MiK82].

As an example of a simple parallel algorithm, consider the *smoothing of an image* [SiS81]. The algorithm described here smooths a gray level input image. The algorithm has "I" as an input image and "S" as an output image. Assume both I and S contain 512-by-512 *pixels* (picture elements), for a total of 512^2 pixels each. Each point of I is an eight-bit unsigned integer representing one of 256 possible gray levels. The gray level of each pixel indicates how "dark" that pixel is, where 0 means white and 255 means black. Each point in the smoothed image, S(i,j), is the average of the gray levels of I(i,j) and its eight nearest neighbors, I(i−1,j−1), I(i−1,j), I(i−1,j+1), I(i,j−1), I(i,j+1), I(i+1,j−1), I(i+1,j), and I(i+1,j+1). The top, bottom, left, and right edge pixels of S are not calculated since their corresponding pixels in I do not have eight adjacent neighbors.

Consider how this could be implemented on an SIMD machine with N = 1024 PEs, logically arranged as an array of 32-by-32 PEs as shown in Fig. 8. Each PE stores a 16-by-16 subimage block of the 512-by-512 image I. Specifically, PE 0 stores the pixels in columns 0 to 15 of rows 0 to 15, PE 1 stores the pixels in columns 16 to 31 of rows 0 to 15, and so on. Each PE smooths its own subimage, with all PEs doing this simultaneously. At the edges of each 16-by-16 subim-

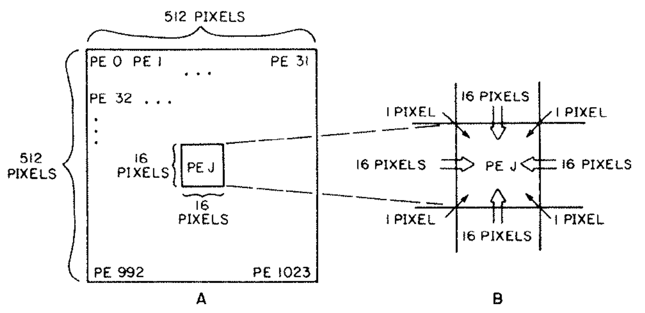

Fig. 8. Data allocation and inter-PE transfers for the image smoothing algorithm.

age, data must be transmitted between PEs in order to calculate the smoothed value. The necessary data transfers are shown for PE J in Fig. 8. Transfers between different PEs can occur simultaneously; e.g., when PE J−1 sends its upper right corner pixel to PE J, PE J can send its upper right corner pixel to PE J+1, PE J+1 can send its upper right corner pixel to PE J+2, etc.

In order to perform a smoothing operation on a 512-by-512 image by the parallel smoothing of 1024 subimage blocks of size 16-by-16, $16^2 = 256$ parallel smoothing operations are performed. As described above, the neighbors of the subimage edge pixels must be transferred in from adjacent PEs. Using either the Cube or ADM networks, the total number of parallel data element transfers needed is $(4 * 16) + 4 = 68$: 16 for each of the top, bottom, left side, and right side edges, and four for the corners (see Fig. 8). The corresponding serial algorithm needs no data transfers between PEs, but $512^2 = 262{,}144$ smoothing calculations must be performed. If no data transfers were needed, the parallel algorithm would be faster than the serial algorithm by a factor of $262{,}144/256 = 1024 = N$. If the inter-PE data transfer time is included and it is assumed that each parallel data transfer requires at most as much time as one smoothing operation, then the time factor improvement is $262{,}144/324 = 809$. The inter-PE transfer time approximation is a conservative one. Thus, the overhead of the 68 inter-PE transfers that must be performed in the SIMD machine is negligible compared to the reduction from 262,144 to 256 smoothing operations.

As an example of a parallel algorithm that is not a straightforward decomposition of the corresponding serial algorithm (as the smoothing example was), consider an SIMD algorithm for computing the ***global histogram of an image*** [SiS81]. Assume there are $B = 2^b = 256$ bins in the histogram, N=1024, and the image is 512-by-512 pixels. The B-bin histogram of the image contains a j in bin i if exactly j of the pixels have a gray level of i, $0 \leq i < B$. Assume the image is equally distributed among the 1024 PEs, i.e., each PE has $512^2/1024$ pixels, and $B \leq 512^2/1024$. Since the image is distributed over 1024 PEs, each PE will calculate a B-bin histogram based on its subimage. Then these "local" histograms will be combined using the method described below.

In the next b steps, each block of B PEs performs B simultaneous recursive doublings [Sto80] to compute the histogram for the portion of the image contained in the block (see Fig. 9). In the first step, each block of

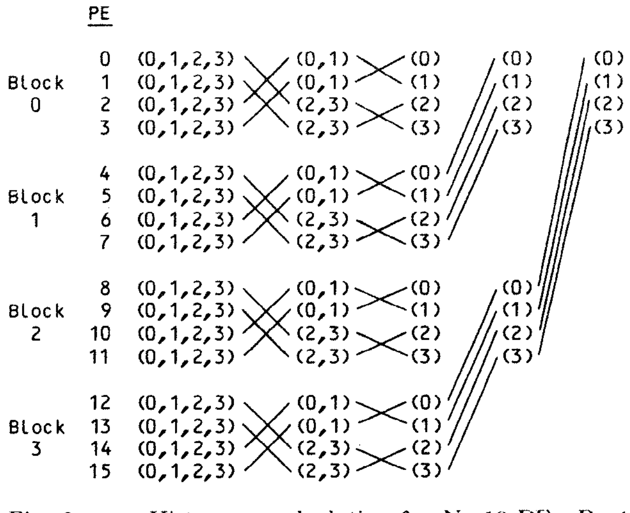

Fig. 9. Histogram calculation for N=16 PEs, B=4 bins. (w,...,z) denotes that bins w through z of the partial histogram are in the PE.

PEs is divided in half such that the PEs with the lower addresses form one group, and the PEs with the higher addresses form another. Each group accumulates the sums for half of the bins, and sends the bins it is not accumulating to the other group. The "lower-numbered group" accumulates the sums for the first half of the bins while the "higher-numbered group" accumulates the second half of the bins. For each successive merging step, the groups of PEs are re-subdivided with the accumulated subtotals for each bin being combined into half as many PEs at each step. The subdividing process continues until there is one PE in each group and each PE has the total value for one bin from the portion of the image contained in the B PEs in its block. The next n−b steps combine the results of these blocks to yield the histogram of the entire image distributed over B PEs. The sum for bin i will end up in PE i, for $0 \leq i < B$. This is done by performing n−b steps of a recursive doubling algorithm to sum the partial histograms from the N/B blocks, shown by the last two steps of Fig. 9. The recursive doubling steps are done for all B bins simultaneously.

A sequential algorithm to compute the histogram of an M-by-M image requires M^2 additions. The SIMD algorithm uses M^2/N additions for each PE to compute its local histogram. At step i in the merging of the partial histograms, $0 \leq i < b$, the number of parallel data transfer/adds required is $B/2^{i+1}$. A total of B−1 transfer/adds are therefore performed in the first b steps of the algorithm. Then n−b parallel transfers and additions are needed to combine the block histograms. This technique therefore requires B−1+n−b parallel transfer/add operations, plus the M^2/N additions needed to compute the local PE histograms. For example, for N=1024, M=512, and B=256, the sequential algorithm would require 262,144 additions; the parallel algorithm uses 256 addition steps plus 257 transfer/add steps. Again, both the Cube and ADM multistage networks can perform each of the required inter-PE data transfers in one step.

Now consider an SIMD/MIMD parallel implementation of ***contour extraction*** [TuA83]. Two algorithms from a contour extraction task are ***edge-guided thresholding (EGT)*** and ***contour tracing***. The EGT algorithm is used to determine a set of optimal thresholds for quantizing the image [MiR81]. The contour tracing algorithm uses the set of thresholds to segment the image and trace the contours. It is assumed that the image to be processed is distributed among the PEs as in the smoothing example.

The EGT algorithm consists of three major steps. First, the Sobel edge operator [DuH73] is used to generate an edge image in which gray levels indicate the magnitude of the gradient. A figure of merit which indicates how well a given threshold gray level matches edges in the edge image is then computed for every possible threshold. Finally, the maximum value of the figure of merit function is chosen to determine the threshold level. This is done for each PE's subimage independently; thus, the threshold levels may differ from one subimage to the next. The window-based Sobel operator calculations and inter-PE communications used in the SIMD EGT algorithm are very similar to those discussed earlier for the image smoothing algorithm.

The MIMD contour tracing algorithm has two phases. In Phase I, PEs segment their subimages based on the threshold value each calculated using the EGT algorithm. All local contours (both closed and partial) are traced and recorded. In Phase II, partial contours traced during Phase I are connected.

In Phase I there is no PE-to-PE communication. Each PE creates a segmented subimage for a particular threshold T by assigning a value of one to subimage pixels having a grey level greater than or equal to T, and a value of zero to the others. Contour tracing begins with each PE scanning the rows of its segmented subimage beginning with the first pixel of the top row. Scanning stops when a start point of a new contour is found. A start point is a pixel with value one which has a zero-valued neighbor to either or both sides. Contours are traced in either a clockwise or counter-clockwise direction and the Freeman direction codes [Fre61] of the "chain" of pixels are recorded. When a pixel from an adjacent subimage would be required to determine the next direction of the contour, a point of indecision is reached. Such a point is recorded as an end point, and the algorithm returns to the start point to trace the contour in the opposite direction until another point of indecision is reached. Closed contours that are contained within a subimage are traced completely during Phase I.

In Phase II, each PE attempts to connect its partial contours to those located in neighboring PEs. PEs consider each partial contour's end point in turn and try to find a possible extending contour in a neighboring PE. Once such an extending contour is found, the process is repeated, if necessary, by following the contour to the next PE until the contour is closed or cannot be extended. A protocol is necessary to prevent more than one PE from trying to use the same partial contour as an extending contour at the same time. Phase II is complete when all of the contours have been connected.

The contour extraction task demonstrates the advantages of several features of PASM. The local neighbor inter-PE transfers needed for the SIMD EGT algorithm can be performed by all PEs simultaneously, as was discussed for the SIMD smoothing algorithm. The global inter-PE communications for the MIMD contour tracing can be performed efficiently by either the multistage Cube or ADM networks. Lastly, the ability of the PEs to operate in either SIMD or MIMD mode allows the most appropriate type of parallelism to be employed by each algorithm in the task.

IV. The PASM Prototype Design

A prototype of the PASM system is currently being constructed (see Fig. 10). All processors in the system

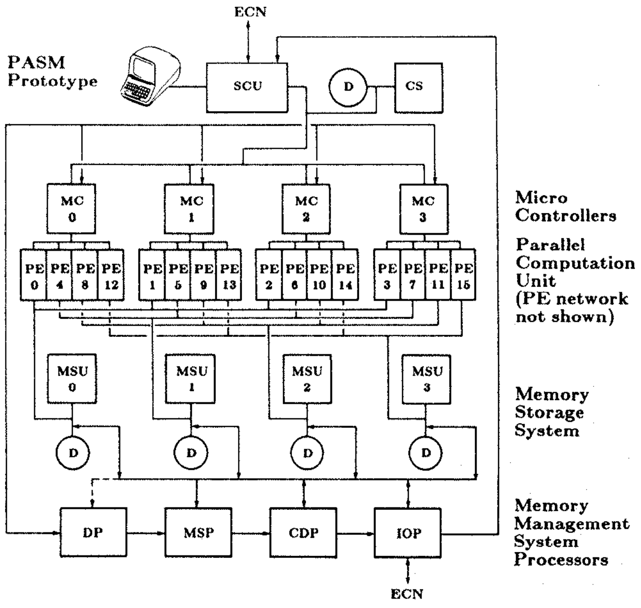

Fig. 10. The PASM parallel processing system prototype.

are based on the Motorola MC68010 16-bit microprocessor. Only off-the-shelf components are used; this eliminates the need for VLSI chips and reduces development time and construction costs. By using a modular design concept, only five different types of physical boards are needed: a CPU board containing the MC68010 microprocessor, a dynamic memory board with up to 1 Mbyte of memory, an I/O board, a network interchange box board, and a specialized board used in the MCs. The System Control Unit, MCs, PEs, Memory Management System processors, Control Storage and Memory Storage Unit processors, and the PE interconnection network can all be implemented by using these boards in different configurations.

The Parallel Computation Unit will consist of 16 PEs, connected by an Extra Stage Cube interconnection network. The Parallel Computation Unit is controlled by four MCs. The prototype could be readily extended to 8 MCs and 8 PEs per MC (pending the availability of funding). The System Control Unit (SCU) for the prototype is a dedicated microprocessor responsible for the overall orchestration of the activities of the system. It shares its mass storage device (D) with the MCs. The device is managed by the Control Storage (CS) microprocessor. In order to allow a larger group of users to access PASM, the System Control Unit will serve as a link between PASM and the Engineering Computer Network (ECN). ECN is a local network of about twenty DEC VAX and PDP-11 computers at Purdue University. The user's terminal will be physically connected to an ECN host computer. The host will provide the environment for the development, compilation, and debugging of SIMD and MIMD programs to prevent the System Control Unit microprocessor from being burdened. Commands (jobs) initiated by users are sent by the host to the System Control Unit, which schedules the jobs to be run on the parallel machine. The prototype system console is used for system startup and monitoring of PASM activities. Mass storage for the PEs is provided by four high capacity Winchester technology disk drives (D), each controlled by a Memory Storage Unit (MSU) microprocessor. The Memory Storage Units are managed by the Memory Management System, consisting of a Directory Lookup Processor (DP), a Memory Scheduling Processor (MSP), a Command Distribution Processor (CDP), and an Input/Output and Reformatting Processor (IOP). User programs and data can be received from or sent to ECN peripherals such as additional mass storage or image input/output devices.

The PE execution unit is a Motorola MC68010 microprocessor. Two 16-bit dynamic memories implement the PASM double-buffered memory scheme. Each of these memories can store 256 Kbytes of program or data. The overall memory capacity can be increased to 2 Mbytes per PE if needed. The I/O board contains parallel ports that interface the PE CPU to the interconnection network. Data can be sent to or read from the network by I/O port read and write instructions. This can be done either by the PE CPU directly or by using a DMA controller. The MC/PE communication controller is also found on the I/O board. This controller allows data exchange between an MC and its PEs via a shared bus. This data link is necessary to allow the MC to coordinate PE activities in MIMD mode.

When all active PEs request an instruction in SIMD mode, an instruction word is broadcast from the MCs and placed on all the PE data busses simultaneously. Each PE decodes the instruction and performs the operation or requests additional operand words. In MIMD mode, instructions and data are contained wholly within a PE's memory and no instructions are broadcast by the MC. Since the instructions for each PE are stored in the PE's local memory (for MIMD operations) or are broadcast to the PE by its MC (for SIMD operations), the interconnection network will be used solely for inter-PE data communication rather than both inter-PE communication and instruction fetch operations.

The PASM prototype's interconnection network is a circuit switched implementation of the Extra Stage Cube network [AdS82] which is a fault-tolerant version of the multistage Cube network discussed earlier. This network is single-fault tolerant and has been shown to be very robust under multiple faults [AdS84]. Circuit switching was chosen for its ease of implementation as well as its particular suitability (when compared to packet switching) for the anticipated large data transfers under DMA (direct memory access) control. Using a circuit switched network, prior to any message transmission between a network source-destination pair, a physical path through the network must be made connecting the pair. This path is established through the use of a request-grant protocol. This connects the source-destination pair for the duration of the message transmission. Individual words within the message are transferred from the source PE to the destination PE using a handshaking protocol between the parallel ports that interface the PEs to the network.

The PASM prototype network is being constructed from readily available SSI/MSI integrated circuits. It is anticipated that a full 1024 PE system would integrate custom designed VLSI components into the network design. In its current configuration of 16 PEs, the network consists of five stages of interchange boxes, with eight boxes per stage. It is estimated that a path through the network can be established in approximately 1000 ns (assuming no delays due to network conflicts). The data itself can be transmitted from a network input port to a network output port at a rate of one 16-bit word every 400 ns. With the PEs themselves operating on a 10 Mhz system clock, these transfer times are fast enough so that the network will not act as a bottleneck in the computation process under execution.

An MC is composed of essentially the same physical boards as a PE but has an extra board containing specialized logic used for time-critical MC functions. The MC execution unit coordinates its PEs in MIMD mode; in SIMD mode, it performs program flow operations (such as loop counting and branching) and the masking operations.

One of the specialized components of the MC is the fetch unit. It is a finite state machine that fetches SIMD instructions from fetch unit memory, determines if they are MC (control) instructions or PE (computational) instructions, and sends them to the appropriate unit. A four-bit tag associated with each instruction word in the fetch unit memory specifies the sequence of actions the fetch unit has to perform on the remaining 16 bits.

When the fetch unit encounters PE instructions, it enqueues them to a FIFO buffer. When all of the PEs associated with an MC request an instruction, one is dequeued and broadcast to them. The FIFO buffer allows overlap between the operations of an MC and its PEs in SIMD mode, thereby improving performance.

For programming the PASM prototype, a parallel assembly language for the MC68010 for SIMD operations has already been implemented. Also, an MC68010-based SIMD simulation system has been developed to allow the exploration of additional PASM design features. Work on the simulation system is continuing, and it will be expanded for MIMD use. Studies of parallel programming languages based on the "C" and "ADA" languages are underway.

V. Summary and Proposed Future Research

This paper provided an overview of the PASM design concepts and prototype and examples of parallel image processing algorithms. Table 1 summarizes the PASM design parameters. A reading list for further information about PASM is provided at the end of this paper. The rest of this section summarizes some of the PASM design features and discusses future research plans.

The possible advantages of a reconfigurable system such as PASM include:

(a) fault tolerance - If a single PE fails, only those virtual machines (partitions) which must include the failed PE need to be disabled. The rest of the system can continue to function.

Table 1. The PASM design parameters, based on current plans.

	general	full PASM	PASM prototype
Number of PEs	N	1024	16
Number of network stages (Extra Stage Cube)	$\log_2 N+1$	11	5
Number of MCs	Q	32	4
Number of PEs per MC	N/Q	32	4
Number of Memory Storage Units	N/Q	32	4
Number of Memory Management System processors	fixed	4	4
Smallest size partition	N/Q	32	4
Maximum number of partitions	Q	32	4

(b) multiple simultaneous users - Since there can be multiple independent virtual machines, there can be multiple simultaneous users of the system, each executing a different program.

(c) program development - Rather than trying to debug a program on, for example, 1024 PEs, it can be debugged on a smaller size virtual machine of 32 PEs.

(d) variable machine size for efficiency - If a task requires only N/2 of N available PEs, the other N/2 can be used for another task.

(e) subtask parallelism - Two independent subtasks that are part of the same job can be executed in parallel, sharing results if necessary.

(f) multiple processing modes - An algorithm can be executed using either the SIMD or MIMD mode of parallelism, whichever is more efficient. A task requiring algorithms that use both modes of parallelism can be executed using the same set of PEs.

The advantageous features of both the multistage Cube and the ADM networks include:

(a) up to N simultaneous transfers are possible;

(b) they are partitionable into independent subnetworks;

(c) they can be controlled in a distributed fashion using routing tags;

(d) one PE can broadcast to all or a subset of the others;

(e) there are a variety of implementation options for these networks;

(f) they can be used in SIMD and/or MIMD operations; and

(g) they can support efficient global as well as local (nearest neighbor) inter-PE communications.

An additional advantage of the Extra Stage Cube is that it is single-fault tolerant.

Permanently assigning a fixed number of PEs to each MC has several advantages over allowing a varying assignment such as used in MAP [Nut77]:

(a) scheduling - The operating system need only schedule (and monitor the "busy" status of) Q MCs, rather than N PEs (when Q=32 and N=1024, this is a substantial savings).

(b) hardware simplicity - No crossbar switch is needed for connecting PEs and control units (such as proposed for MAP [Nut77]).

(c) software simplicity - There is no need to do the bookkeeping of recording PE to MC assignments.

(d) network partitioning - The fixed assignment supports network partitioning.

(e) secondary storage - The fixed assignment allows the efficient use of multiple secondary storage devices.

The main disadvantage of this approach is that each virtual machine size must be a power of two, with a minimum value of N/Q. However, for PASM's intended experimental environment, flexibility at reasonable cost is the goal, not maximum PE utilization.

Advantages of the MC memory organization include:

(a) the use of the reconfigurable bus for sharing memory modules and tolerance of a faulty memory module; and

(b) the ability to overlap computation and MC memory module loading due to the double-buffering.

The Memory Storage System design allows the loading/unloading of a virtual machine of RN/Q PEs in R parallel block moves. The double-buffered PE memory modules permit this loading/unloading to be overlapped with PE execution. The Memory Management System, which coordinates these memory transfers, is implemented with multiple processors, providing parallelism at another level.

The PASM operating system functions will be distributed over various system components to prevent the System Control Unit from being a bottleneck. These components are the System Control Unit, the MCs, the Memory Management System, the Memory Storage Unit processors, and the Control Storage processor.

We have a group of faculty at Purdue who want to study a variety of topics relating to the application of parallel processing to vision and the implementation and use of the PASM prototype. These topics include:

Parallel Computer Vision: We will continue to study the structuring of image understanding tasks for efficient parallel execution. In particular, we will investigate how to use the reconfigurable PASM system for both the numeric and symbolic processing that is necessary for vision tasks. Our algorithm studies will involve complexity analyses, simulation, and execution on the prototype.

Reliability: Limited fault-tolerance is built into PASM due to its partitionability, use of a fault-tolerant interconnection network, etc. Further work includes defining the overall system reliability goals and methods for their evaluation, and the development of fault detection and recovery algorithms and their hardware and software requirements.

Software - Languages: We will investigate programming language notations for specifying both implicit and explicit specification of parallelism. Using our algorithm studies as a basis, we want to develop optimizing compilation techniques for the efficient mapping of applications onto parallel architectures. We plan to continue our research on parallel LISP, and to develop a LISP system for the PASM prototype.

Software - Operating Systems: The special distributed operating system problems which we want to study include: mechanisms for the efficient switching between the SIMD and MIMD modes of parallelism; automatically reconfiguring a task when a fault requires it to run on fewer processors than it was compiled for; methods for efficiently implementing the distribution and subsequent coordination of operating system functions over the various system components; and tools for algorithm debugging and hardware supported performance collection from many points in the system.

Hardware: The PASM prototype incorporates the MC68010 as the basic building block for the PEs and MCs, using external hardware to interface and adopt these processors for the prototype. In preparation for the construction of larger PASM systems based on our experience with the prototype, we wish to study the incorporation of these features into custom VLSI processor implementations specifically geared towards use in multifunction multiprocessors, emphasizing easily extensible and highly reliable systems.

Additional topics of interest include: the comparison of design and implementation choices for (1) a fault-tolerant interconnection network, (2) communications between the PEs and MCs, (3) communications among the MCs, and (4) the connection scheme linking the multiple secondary storage devices to the PE memories; the development of additional operating system and hardware features to facilitate the use of one reconfigurable system for both the numeric processing and symbolic manipulation typical of complete vision tasks; and the design of real-time input/output device interfaces to PASM.

In conclusion, the objective of the PASM design is to achieve a system which attains a compromise between flexibility and cost-effectiveness for a specific problem domain. A dynamically reconfigurable system such as PASM will be a valuable tool for both image understanding and parallel processing research.

Acknowledgements -- Portions of this overview of the PASM system and its use in parallel image processing are based on [SiM81b], [SiS81], and [TuA83]. The contributions of the co-authors on these papers are gratefully acknowledged. Some of the figures are from ***Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies***, by H. J. Siegel, Lexington Books, Lexington, MA, 1984. The work surveyed in this paper was supported by the following grants/contracts: United States Air Force Command, Rome Air Development Center, under contract F30602-83-K-0119; United States Army Research Office, Department of the Army, under grant DAAG29-82-K-0101; National Science Foundation, under grant ECS 80-16580; Ballistic Missile Defense Agency, under contract DAS60-80-C-0022; Defense Mapping Agency, monitored by the United States Air Force Command, Rome Air Development Center, under contract F30602-C-0193; Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant AFOSR-78-3581.

References

[AdS82] G. B. Adams III and H. J. Siegel, "The extra stage cube: A fault-tolerant interconnection network for supersystems," *IEEE Trans. Computers*, Vol. C-31, May 1982, pp. 443-454.

[AdS84] G. B. Adams III and H. J. Siegel, "Modifications to improve the fault tolerance of the extra stage cube interconnection network," *1984 Int'l. Conf. Parallel Processing*, Aug. 1984, to appear.

[ArP76] R. Arnold and E. Page, "A hierarchical, restructurable multimicroprocessor architecture," *3rd Symp. Computer Architecture*, Jan. 1976, pp. 40-45.

[Bar68] G. Barnes, et al., "The Illiac IV computer," *IEEE Trans. Computers*, Vol. C-17, Aug. 1968, pp. 746-757.

[Bat76] K. E. Batcher, "The flip network in STARAN," *1976 Int'l. Conf. Parallel Processing*, Aug. 1976, pp. 65-71.

[Bat77] K. E. Batcher, "STARAN series E," *1977 Int'l. Conf. Parallel Processing*, Aug. 1977, pp. 144-153.

[Bat82] K. E. Batcher, "Bit serial parallel processing systems," *IEEE Trans. Computers*, Vol. C-31, May 1982, pp. 337-384.

[Bou72] W. J. Bouknight, et al., "The Illiac IV system," *Proc. IEEE*, Vol. 60, Apr. 1972, pp. 369-388.

[Cra72] B. A. Crane, et al., "PEPE computer architecture," *COMPCON 1972*, Sept. 1972, pp. 57-60.

[DuH73] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley and Sons, New York, NY, 1973.

[Fly66] M. J. Flynn, "Very high-speed computing systems," *Proc. IEEE*, Vol. 54, Dec. 1966, pp. 1901-1909.

[Fou81] T. J. Fountain, "CLIP4: progress report," in *Languages and Architectures for Image Processing*, M. J. B. Duff and S. Levialdi, editors, Academic Press, London, 1981, pp. 281-291.

[Fre61] H. Freeman, "Techniques for the digital computer analysis of chain-encoded arbitrary plane curves," *Proc. NEC*, Vol. 17, Oct. 1961, pp. 421-432.

[GoL73] L. R. Goke and G. J. Lipovski, "Banyan networks for partitioning multimicroprocessor systems," *1st Symp. Computer Architecture*, Dec. 1973, pp. 21-28.

[GoG83] A. Gottlieb, R. Grishman, C. P. Kruskal, K. P. McAuliffe, L. Rudolph, and M. Snir, "The NYU Ultracomputer -- designing an MIMD shared memory parallel computer," *IEEE Trans. Computers*, Vol C-32, Feb. 1983, pp. 175-189.

[KaK79] S. I. Kartashev and S. P. Kartashev, "A multicomputer system with dynamic architecture," *IEEE Trans. Computers*, Vol. C-28, Oct. 1979, pp. 704-720.

[Law75] D. H. Lawrie, "Access and alignment of data in an array processor," *IEEE Trans. Computers*, Vol. C-24, Dec. 1975, pp. 1145-1155.

[MiK82] O. R. Mitchell, F. P. Grogan, and D. J. Charpentier, "A shape extraction and recognition system," *Southcon 82*, Mar. 1982, pp. 4/1:1-4/1:4.

[MiR81] O. R. Mitchell, A. P. Reeves, and K-S. Fu, "Shape and texture measurements for automated cartography," *1981 IEEE Computer Soc. Conf. Pattern Recognition and Image Processing*, Aug. 1981, pp. 367.

[Nut77] G. J. Nutt, "Microprocessor implementation of a parallel processor," *4th Symp. Computer Architecture,* Mar. 1977, pp. 147-152.

[Pat81] J. H. Patel, "Performance of processor-memory interconnections for multiprocessors," *IEEE Trans. Computers,* Vol. C-30, Oct. 1981, pp. 771-780.

[Pea77] M. C. Pease, III, "The indirect binary n-cube microprocessor array," *IEEE Trans. Computers,* Vol. C-26, May 1977, pp. 458-473.

[SeU80] M. C. Sejnowski, E. T. Upchurch, R. N. Kapur, D. P. S. Charlu, and G. J. Lipovski, "An overview of the Texas Reconfigurable Array Computer," *AFIPS 1980 Nat'l. Computer Conf.,* June 1980, pp. 631-641.

[SiM81a] H. J. Siegel and R. J. McMillen, "Using the augmented data manipulator network in PASM," *Computer,* Vol. 14, Feb. 1981, pp. 25-33.

[SiM81b] H. J. Siegel and R. J. McMillen, "The multistage cube: a versatile interconnection network," *Computer,* Vol. 14, Dec. 1981, pp. 65-76.

[SiS81] H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller, Jr., H. E. Smalley, and S. D. Smith, "PASM: a partitionable SIMD/MIMD system for image processing and pattern recognition," *IEEE Trans. Computers,* Vol. C-30, Dec. 1981, pp. 934-947.

[Sie77] H. J. Siegel, "Analysis techniques for SIMD machine interconnection networks and the effects of processor address masks," *IEEE Trans. Computers,* Vol. C-26, Feb. 1977, pp. 153-161.

[Sie79] H. J. Siegel, "A model of SIMD machines and a comparison of various interconnection networks," *IEEE Trans. Computers,* Vol. C-28, Dec. 1979, pp. 907-917.

[Sto80] H. S. Stone, "Parallel computers," in *Introduction to Computer Architecture,* 2nd edition, edited by H. S. Stone, Science Research Associates, Inc., Chicago, IL, 1980, pp. 363-425.

[SwF77] R. J. Swan, S. H. Fuller, and D. P. Siewiorek, "Cm*: a modular, multi-microprocessor," *Nat'l. Computer Conf.,* June 1977, pp. 637-644.

[TuA83] D. L. Tuomenoksa, G. B. Adams III, H. J. Siegel, and O. R. Mitchell, "A parallel algorithm for contour extraction: advantages and architectural implications," *1983 IEEE Comp. Soc. Symp. Computer Vision and Pattern Recognition,* June 1983, pp. 336-344.

[WuB72] W. A. Wulf and C. G. Bell, "C.mmp - a multi-miniprocessor," *Fall Joint Computer Conf.,* Dec. 1972, pp. 765-777.

[WuF80] C. L. Wu and T. Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Computers,* Vol. C-29, Aug. 1980, pp. 694-702.

Reading List of Selected PASM Related Publications

Reconfigurable Organization:

H. J. Siegel, P. T. Mueller, Jr., and H. E. Smalley, Jr., "Control of a partitionable multimicroprocessor system," *1978 Int'l. Conf. Parallel Processing,* Aug. 1978, pp. 9-17.

H. J. Siegel, L. J. Siegel, F. C. Kemmerer, P. T. Mueller, Jr., H. E. Smalley, and S. D. Smith, "PASM: a partitionable SIMD/MIMD system for image processing and pattern recognition," *IEEE Trans. Computers,* Vol. C-30, Dec. 1981, pp. 934-947.

Parallel Memory Management System:

H. J. Siegel, F. Kemmerer, and M. Washburn, "Parallel memory system for a partitionable SIMD/MIMD machine," *1979 Int'l. Conf. Parallel Processing,* Aug. 1979, pp. 212-221.

J. T. Kuehn, H. J. Siegel, and M. Grosz, "A distributed memory management system for PASM," *IEEE Comp. Soc. Workshop Computer Architecture for Pattern Analysis and Image Database Management,* Oct. 1983, pp. 101-108.

Interconnection Network - Multistage Cube:

R. J. McMillen and H. J. Siegel, "The hybrid cube network," *Distributed Data Acquisition, Computing, and Control Symp.,* Dec. 1980, pp. 11-22.

R. J. McMillen, G. B. Adams III, and H. J. Siegel, "Performance and implementation of 4x4 switching nodes in an interconnection network for PASM," *1981 Int'l. Conf. Parallel Processing,* Aug. 1981, pp. 229-233.

H. J. Siegel and R. J. McMillen, "The multistage cube: a versatile interconnection network," *Computer,* Vol. 14, Dec. 1981, pp. 65-76.

G. B. Adams III and H. J. Siegel, "The extra stage cube: a fault-tolerant interconnection network for supersystems," *IEEE Trans. Computers,* Vol. C-31, May 1982, pp. 443-454.

G. B. Adams III and H. J. Siegel, "The use of 4x4 switching elements in the multistage cube network," *1st Int'l. Conf. Computers and Applications,* June 1984, pp. 585-592.

G. B. Adams III and H. J. Siegel, "A modification to improve the fault tolerance of the extra stage cube interconnection network," *1984 Int'l. Conf. Parallel Processing,* Aug. 1984, to appear.

Interconnection Network - ADM:

S. D. Smith, H. J Siegel, R. J. McMillen, and G. B. Adams III, "Use of the augmented data manipulator multistage network for SIMD machines," *1980 Int'l. Conf. Parallel Processing,* Aug. 1980, pp. 75-78.

R. J. McMillen, G. B. Adams III, and H. J. Siegel, "Permuting with the augmented data manipulator network," *18th Allerton Conf. Communication, Control, and Computing,* Oct. 1980, pp. 544-553.

H. J. Siegel and R. J. McMillen, "Using the augmented data manipulator network in PASM," *Computer,* Vol. 14, Feb. 1981, pp. 25-33.

R. J. McMillen and H. J. Siegel, "Performance and fault tolerance improvements in the inverse augmented data manipulator network," *9th Int'l. Symp. Computer Architecture,* Apr. 1982, pp. 63-72.

G. B. Adams III and H. J. Siegel, "On the number of permutations performable by the augmented data manipulator network," *IEEE Trans. Computers,* Vol. C-31, Apr. 1982, pp. 270-277.

R. J. McMillen and H. J. Siegel, "Routing schemes for the augmented data manipulator network in an MIMD system," *IEEE Trans. Computers,* Dec. 1982, pp. 63-72.

Interconnection Network - Multistage Cube/ADM Comparisons:

H. J. Siegel and S. D. Smith, "Study of multistage SIMD interconnection networks," *5th Symp. Computer Architecture,* Apr. 1978, pp. 223-229.

H. J. Siegel, "Interconnection networks for SIMD machines," *Computer,* Vol. 12, June 1979, pp. 57-65.

H. J. Siegel, R. J. McMillen, and P. T. Mueller, Jr., "A survey of interconnection methods for reconfigurable parallel processing systems," *1979 Nat'l. Computer Conf.,* June 1979, pp. 529-542.

H. J. Siegel, "The theory underlying the partitioning of permutation networks," *IEEE Trans. Computers,* Vol. C-29, Sept. 1980, pp. 791-801.

G. B. Adams III and H. J. Siegel, "A survey of fault-tolerant multistage networks and comparison to the extra stage cube," *17th Hawaii Int'l. Conf. System Sciences,* Jan. 1984, pp. 268-277.

H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies,* Lexington Books, Lexington, MA, 1984.

R. J. McMillen and H. J. Siegel, "Evaluation of cube and data manipulator networks," *Journal of Parallel and Distributed Computing,* scheduled to appear in 1984.

Distributed Operating System:

H. J. Siegel, L. J. Siegel, R. J. McMillen, P. T. Mueller, Jr., and S. D. Smith, "An SIMD/MIMD multimicroprocessor system for image processing and pattern recognition," *1979 IEEE Comp. Soc. Conf. Pattern Recognition and Image Processing,* Aug. 1979, pp. 214-224.

D. L. Tuomenoksa and H. J. Siegel, "Application of two-dimensional bin packing algorithms for task scheduling in the PASM multimicrocomputer system," *19th Allerton Conf. Communication, Control, and Computing,* Oct. 1981, pg. 542.

D. L. Tuomenoksa and H. J. Siegel, "Analysis of the PASM control system memory hierarchy," *1982 Int'l. Conf. Parallel Processing,* Aug. 1982, pp. 363-370.

D. L. Tuomenoksa and H. J. Siegel, "Analysis of multiple-queue task scheduling algorithms for multiple-SIMD machines," *3rd Int'l. Conf. Distributed Computing Systems,* Oct. 1982, pp. 114-121.

D. L. Tuomenoksa and H. J. Siegel, "A distributed operating system for PASM," *17th Hawaii Int'l. Conf. System Scienes,* Jan. 1984, pp. 69-77.

D. L. Tuomenoksa and H. J. Siegel, "Task preloading schemes for the PASM dynamically reconfigurable parallel processing system," *IEEE Trans. on Computers,* scheduled to appear in Vol. C-32, 1984.

Prototype Design:

J. T. Kuehn and H. J. Siegel, "Simulation studies of PASM in SIMD mode," *1981 IEEE Comp. Soc. Workshop Computer Architecture for Pattern Analysis and Image Database Management,* Nov. 1981, pp. 43-50.

J. T. Kuehn, H. J. Siegel, and P. D. Hallenbeck, "Design and simulation of an MC68000-based multimicroprocessor system," *1982 Int'l. Conf. Parallel Processing,* Aug. 1982, pp. 353-362.

Parallel Programming Language:

P. T. Mueller, Jr., L. J. Siegel, and H. J. Siegel, "A parallel language for image and speech processing," *IEEE Comp. Soc. 4th Int'l. Computer Software and Applications Conference (COMPSAC 80),* Oct. 1980, pp. 476-483.

C. Cline and H. J. Siegel, "Extensions of Ada for SIMD parallel processing," *IEEE Comp. Soc. 7th Int'l. Computer Software and Applications Conf. (COMPSAC 83),* Nov. 1983, pp. 366-372.

C. Cline and H. J. Siegel, "A comparison of parallel language approaches to data representation and data transferral," *Computer Data Engineering (COMPDEC) Conf.,* Apr. 1984, pp. 60-66.

Parallel Image Processing:

L. J. Siegel, P. T. Mueller, Jr., and H. J. Siegel, "FFT algorithms for SIMD machines," *17th Allerton Conf. Communication, Control, and Computing,* Oct. 1979, pp. 1006-1015.

P. T. Mueller, Jr., L. J. Siegel, and H. J. Siegel, "Parallel algorithms for the two-dimensional FFT," *5th Int'l. Conf. Pattern Recognition,* Dec. 1980, pp. 497-502.

P. H. Swain, H. J. Siegel, and J. El-Achkar, "Multiprocessor implementation of image pattern recognition: a general approach," *5th Int'l. Conf. Pattern Recognition,* Dec. 1980, pp. 309-317.

H. J. Siegel and P. H. Swain, "Contextual classification on PASM," *IEEE Comp. Soc. Conf. Pattern Recognition and Image Processing,* Aug. 1981, pp. 320-325.

L. J. Siegel, E. J. Delp, T. N. Mudge, and H. J. Siegel, "Block truncation coding on PASM," *19th Annual Allerton Conf. on Communication, Control, and Computing,* Oct. 1981, pp. 891-900.

L. J. Siegel, "Image processing on a partitionable SIMD machine," in *Languages and Architectures for Image Processing,* M. J. B. Duff and S. Levialdi, editors, Academic Press, London, 1981, pp. 294-300.

T. N. Mudge, E. J. Delp, L. J. Siegel, and H. J. Siegel, "Image coding using the multimicroprocessor system PASM," *IEEE Comp. Soc. Conf. Pattern Recognition and Image Processing,* June 1982, pp. 200-205.

L. J. Siegel, H. J. Siegel, and A. E. Feather, "Parallel processing approaches to image correlation," *IEEE Trans. Computers,* Vol. C-31, Mar. 1982, pp. 208-218.

L. J. Siegel, H. J. Siegel, and P. H. Swain, "Performance measures for evaluating algorithms for SIMD machines," *IEEE Trans. Software Engineering,* Vol. SE-8, July 1982, pp. 319-331.

M. R. Warpenburg and L. J. Siegel, "Image resampling in an SIMD environment," *IEEE Trans. Computers,* Oct. 1982, pp. 934-942.

H. J. Siegel, P. H. Swain, and B. W. Smith, "Remote sensing on PASM and CDC Flexible Processors," in *Multicomputers and Image Processing: Algorithms and Programs,* K. Preston and L. Uhr, eds. Academic Press, New York, NY, 1982, pp. 331-342.

L. J. Siegel, H. J. Siegel, P. H. Swain, G. B. Adams III, W. E. Kuhn III, R. J. McMillen, T. A. Rice, K. D. Smith, and D. L. Tuomenoksa, "Distributed computing for signal processing: modeling of asynchronous parallel computation, 1983 progress report," Purdue University, School of Electrical Engineering, Technical Report No. TR-EE 83-11, Mar. 1983, 292 pages.

D. L. Tuomenoksa, G. B. Adams III, H. J. Siegel, and O. R. Mitchell, "A parallel algorithm for contour extraction: advantages and architectural implications," *1983 IEEE Comp. Soc. Symp. Computer Vision and Pattern Recognition,* June 1983, pp. 336-344.

L. J. Siegel, H. J. Siegel, P. H. Swain, G. B. Adams III, G. M. Lin, D. L. Tuomenoksa, and T. A. Rice, "Parallel processing approaches to production scenarios for mapping applications," Purdue University, School of Electrical Engineering, Technical Report No. TR-EE 83-27, Aug. 1983, 265 pages.

T. A. Rice and L. J. Siegel, "Parallel algorithms for computer vision," *Workshop on Computer Architecture for Pattern Analysis and Image Database Management,* Oct. 1983, pp. 93-100.

J. T. Kuehn and H. J. Siegel, "Simulation studies of a parallel histogramming Algorithm for PASM," *7th Int'l. Conf. on Pattern Recognition,* July 1984, pp. 646-649.

J. T. Kuehn, H. J. Siegel, D. L. Tuomenoksa, and G. B. Adams III, "The use and design of PASM," in *Image Processing: From Computation to Integration,* edited by S. Levialdi, Academic Press, London, 1984, to appear.

T. A. Rice and L. Siegel, "Parallel processing for computer vision," in *Image Processing: From Computation to Integration,* S. Levialdi, editor, Academic Press, London, 1984, to appear.

☆**P**URDUE
☆**A**LL-
☆**S**TAR
☆**M**ACHINE