VLSI MULTIPROCESSOR FOR IMAGE PROCESSING*

K. S. Fu, K. Hwang, and B. W. Wah
School of Electrical Engineering
PURDUE UNIVERSITY
West Lafayette, Indiana 47907 USA

I. INTRODUCTION

Image analysis refers to the use of digital computers for Pattern Recognition and Image Processing (PRIP). On-line imagery data needs to be stored on disks and quickly retrieved for PRIP applications. This article presents a systematic approach to developing a special-purpose computer for processing pictorial information. This approach integrates both pattern-analysis and image-database-management capabilities into a unified design to meet the challenges. The integrated design is aimed at the development of a real-time and interactive computer system for both high-level pattern recognition and low-level image processing.

We shall examine special database machines suggested for handling imagery data. Recent efforts on VLSI hardware approaches to implementing PRIP algorithms and to language recognition will be discussed. The integrated architectural approach is initiated by the PUMPS architecture currently under development at Purdue University. We shall identify the desired architectural features, processing languages, image database systems, and underlying VLSI computing structures for developing intelligent computer imaging systems.

II. INTEGRATED IMAGE PROCESSING

A typical computer imaging system consists of four stages as depicted in Fig.1. The preprocessing stage includes image operations

* This research was supported by the U. S. National Science Foundation under grant ECS-80-16580.

NATO ASI Series, Vol. F18
Computer Architectures for Spatially Distributed Data
Edited by H. Freeman and G. G. Pieroni

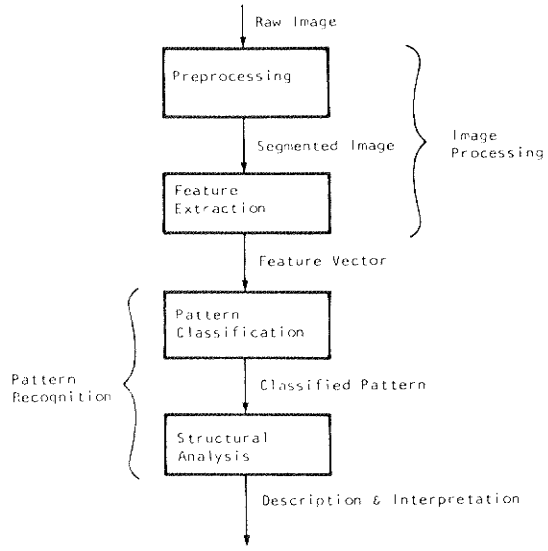

Fig. 1 Processing stages of a pattern-analysis computer system

like smoothing, enhancement, restoration, edge detection, and segmentation, etc. Raw images are reduced to segmented patterns by this initial stage. The next stage is for image segmentation and feature extraction, which further reduces the segmented image to a small set of feature vectors. Clustering techniques may be applied at this stage. The third stage is for pattern classification, which recognizes the membership of extracted features among known pattern classes. The fourth stage is for structural analysis and interpretation, to produce a concise description and interpretation of the pattern information.

Conventional computers are primarily designed to process one-dimensional strings of alphanumeric data. To process multi-dimensional information on SISD computers requires image coding and picture transformation (such as projection, registration, etc.). Sequential machines cannot efficiently exploit parallelism embedded in most PRIP operations. On the other hand, large parallel computers, such as (SIMD) array processors and (MIMD) multiprocessors, may not be necessarily cost-effective in implementing simple and repetitive image operations over very large and, sometimes, dynamically changing image databases.

An adequate pattern-analysis computer is expected to perform at least 100 megaflops with a memory bandwidth of at least 256 megabytes for applications in the 1980's, and many require a processing power of 1000 megaflops or higher for those applications in the 1990's. Two

earlier surveys on special computer architectures for PRIP have been given by Danielsson and Levialdi [6] and Hwang and Fu [22].

Computer Image Analysis

Identified below are desired architectural and functional features in an image processing computer. We focus on the interplay between computer architectures and PRIP applications. In general, a PRIP computer should be featured with as many of the following capabilities as possible:

(a) To explore spatial parallelism, a pattern-analysis computer may be equipped with replicated arithmetic/logic units operating synchronously in SIMD mode. Moreover, high degree of pipelining (temporal parallelism) is desired for overlapped instruction execution and pipelined vector arithmetic.

(b) Some PRIP computers choose a multiprocessor configuration to support asynchronous computations in MIMD mode. Data flow multiprocessor systems also been also suggested for PRIP or Artificial Intelligence computations.

(c) Hierarchical memory system is needed for image storage and manipulation. Large main memory with fast image cache must be employed to alleviate the problem of image data overflow. Fast and intelligent I/O and sensing devices are needed for interactive pattern analysis and image query processing.

(d) Special image database management systems or image database machines are demanded for fast image information retrieval. Toward this end, some high-level picture description/manipulation languages need to be developed, in addition to developing image query languages.

(e) PRIP computers should fully utilize state-of-the-art hardware componentsand available software packages. Dedicated VLSI devices are needed for PRIP at signal-processing level and at symbol-manipulation level. Special VLSI pattern recognizers and image filtering chips are needed for fast image construction, threshold-

ing, FFT, histogram analysis, feature selection, and syntax analysis.

Image Database Management

An image database system provides a large collection of structured imagery data (digitized pictures) for easy access by a large number of users. It provides both high level query support and low level image access. Most of these image database systems are implemented with specially developed software packages upon dedicated pattern analysis systems. It is highly desirable to develop a dedicated backend database machine for image database management. So far, several hardware attempts were suggested. But none of them has yet been implemented for image database management.

Image database management functions and peripheral supports are depicted in Fig. 2. First, we need faster and intelligent image input devices. The image features and structures (shape, texture, and spatial relationships) extracted by the host image processor should be converted into symbolic image sketches stored in a logical image database. For those unconverted raw images, the system must convert them into efficient codes stored in the physical image database.

Flexible image manipulation and retrieval functions must be established using high-level image manipulation languages and image description languages. The logical database is used for image reconstruction from relational sketches. The compressed raw images must be

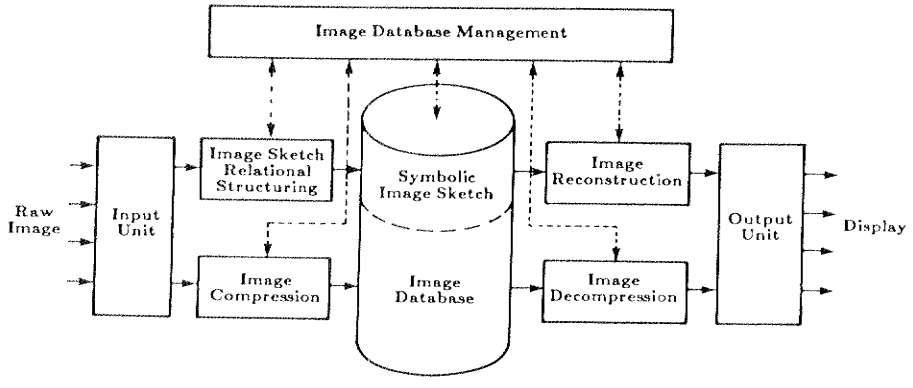

Fig. 2 An intelligent image database system and its management functions

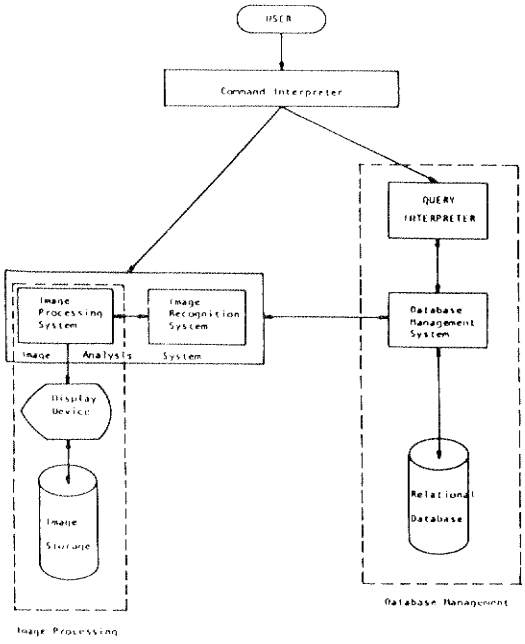

Fig. 3 Integrated image analysis and database management system

decompressed for high-resolution console display. The output unit is responsible for extracting results to be sent to the host computer. The above image database management functions should be supported with specially designed hardware units that constitute an image database machine.

The Integrated System

The three functions, image processing, pattern recognition, and image database management, must be integrated into an efficient pictorial information system. A data-flow block diagram of such an integrated system is shown Fig. 3. Three subsystems in the diagram correspond to the three addressed functions. These subsystems must interact and cooperate with each other to achieve the said objectives. The user communicates with the system through pictorial query language. The raw images are physically stored on disks (or tapes). A relational image database is established by mapping the physical images into the logically structured database. The image processing subsystem performs image preprocessing and feature extraction. The

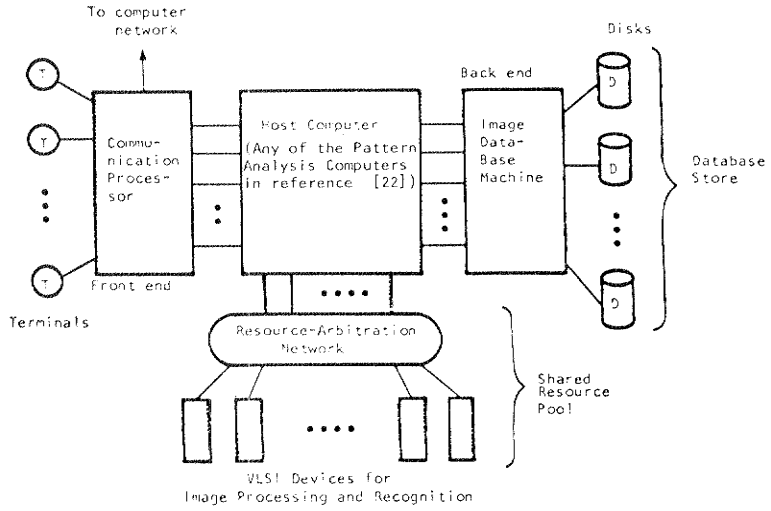

Fig. 4 Architecture of an integrated computer system for pattern analysis and image database management

pattern recognition subsystem performs pattern classification and picture interpretation operations. The image data management subsystem handles query processing and image database operations.

The system architecture of such an integrated image analysis computer is conceptually illustrated in Fig. 4. The system consists of four major subsystems, as shown by the four blocks in the drawing. The host computer can be any one of those existing pattern-analysis computers [22]. The backend database machine is specially developed for image database management. Either software or hardware approaches can be adopted in developing image database management systems. The front-end communication processor is used to handle terminal activities or to be connected to a computer network for remote users. The shared resource pool contains VLSI functional units or attached special processors for fast PRIP operations. A resource sharing network is needed between the host processors and the shared resource pool.

III. THE PUMPS ARCHITECTURE

PUMPS is a high-performance multiprocessor computer with a shared resource pool of VLSI devices. A block diagram showing the major components in PUMPs is given in Fig. 5. There are p processors in the system, each of which is multiprogrammed. The processors can operate

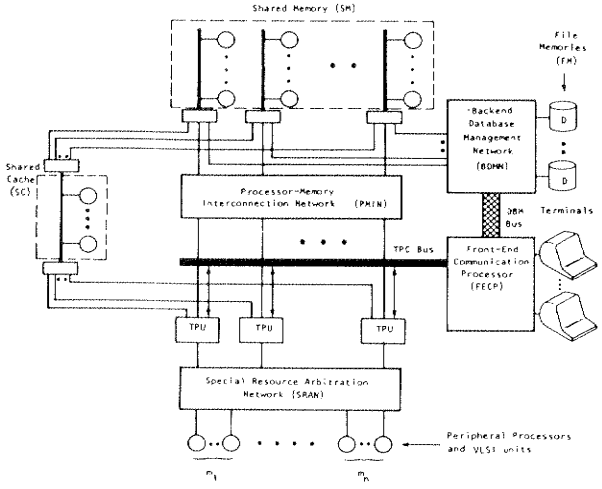

Fig. 5 System architecture of PUMPS

in an interactive fashion through the shared resource pool and shared memory system. They can communicate with each other via an interrupt bus. This intercommunication medium is very effective in passing interrupts, synchronization and other control signals.

All the processors are connected to the shared-resource pool via a Resource Sharing Network. This network provides connections between each processor and the desired peripheral processor or VLSI functional unit. As VLSI technology develops, modular switches will become more cost-effective because of their regular and local connections. In case several processors reference the same functional unit, some priority must be established to resolve the conflicts.

The allocation of the shared resources in the pool to the processors depends on the computational requirements of the active processes. The allocation is considered dynamic. Furthermore, the selection of the resource types in the pool is tailored to special application requirements. For example, one may wish to include an FFT processor, a histogram analyzer, and some VLSI array or pipeline processors in the pool for image analysis applications. In this sense, PUMPS has a dynamically reconfigurable structure. Different applicative environments may be equipped with different functional units. The remaining system resources such as processors and shared memories, are designed for MIMD computations needed in high-level pattern recognition applications.

The processors perform three basic functions: (i) dispatching and initiating tasks for the shared peripheral processors and VLSI units; (ii) executing purely sequential tasks; (iii) participating in MIMD

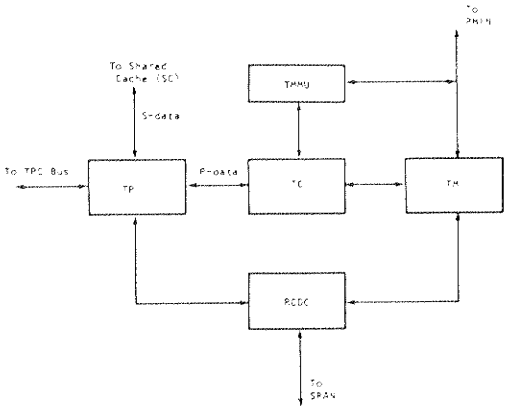

Fig. 6 Architecture of a task processing unit (TPU)

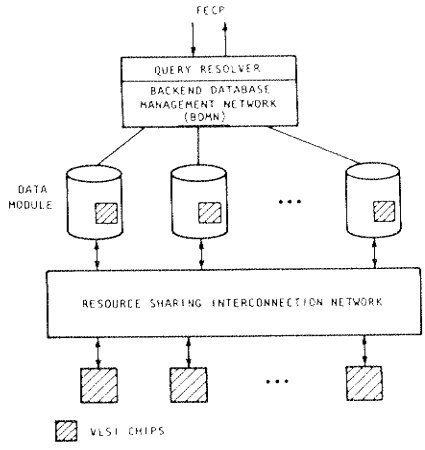

Fig. 7 A Conceptual view of an image database machine.

processes and running the operating system. In order to perform the first type of function, a local Memory is provided within each processor as depicted in Fig. 6. The local memory is partitioned into several segments. These consist of the unmapped memory, which is used for the operating system kernel and device drivers, the local image buffers and the local scratch-pad. The local image buffers are shared between the Task Processor and the resource pool. The processor, acting as a controller, must provide the peripheral devices with a continuous flow of data. A resource controller and data channels are used to format and channel the data between the task memory and peripheral units.

The PUMPS has a distributed memory organization using virtual memory addressing based on paged segments. Within the task cache, misses are serviced by initiating block transfer from the task memory. If a block does not reside in the task memory but is known to the process, a page fault occurs which is also serviced by the page-fault handler. The occurrence of a page fault causes the current active process in the task processor to be blocked, whereupon a task switch is made to a runnable process also residing in the processor. By distributing the memory management functions the task processors are relieved of performing memory management functions and thereby increase their effectiveness in performing useful computations.

The interleaved task memory serves as a high-spaced buffer between the processor and the Shared Memories, which are semiconductor memories organized to permit efficient block transfers of information. A block-transfer oriented Interconnection Network such as the delta

network, is used between the processors and the shared memories. The shared memories are also connected to the disk memories via an image database management network, which is designed to handle data transfers from multiple disks. The file memories, together with a backend computer and the backend network, comprise the database machine for image processing.

The architectural design of the database machine for image processing consists of three parts: a set of data modules, each of which includes a disk with the associated cellular logic for processing picture queries, a backend computer, and the backend database management network (Fig. 7). The necessity of providing a high-level language interface to the users complicates the design issues. However, the design of the database machine is based on various image data manipulation and retrieval operators. These operators are interpretable via a language interface which permits a logical representation of the images.

IV. VLSI IMAGE PROCESSORS

Recent advances in VLSI micro-electronic technology have triggered the thought of implementing some PRIP algorithms directly in hardware. VLSI image recognizers offer high speed and accuracy which are useful in real-time, on-line, pictorial information processing. This is the first step towards advanced automation and machine intelligence. Recently, many attempts have been made in developing special VLSI devices for signal/image processing and pattern recognition. Some of these approaches involve large-scale matrix computations and some syntactic parsing operations. We list in Table 1 some candidate PRIP algorithms that are suitable for VLSI implementation.

Statistical Pattern Recognition

Partitioned matrix algorithms can be used in L-U decomposition, matrix multiplication, matrix inversion, and solution of triangular systems of equations [9]. Pipelined VLSI networks have been developed to realize these partitioned matrix algorithms [10].

Table 1 Candidate image algorithms for VLSI

Image Processing	Enhancement, Filtering, Thinning, Edge Detection, Segmentation, Registration. Restoration, Clustering, Texture Analysis, Convolution, Fourier Analysis, etc.
Pattern Recognition	Feature Extraction, Template Matching, Statistical Classification, Graph Algorithms, Syntax Analysis, Change Detection, Language Recognition, Scene Analysis and Synthesis, etc.
Image Query Processing	Query Decomposition, Query Optimization, Attribute Manipulation, Picture Reconstruction, Search/Sorting Algorithms, Query-by-Picture-Example Implementation, etc.
Image Database Processng	Relational Operators (JOIN, UNION, INTERSECTION, PROJECTION, COMPLEMENT), Image-Sketch-Relation Conversion, Similarity Retrieval, Data Structures, Priority Queues, Dynamic Programming, Spatial Operators, etc.

Consider the following example. Given a triangular matrix U, we want to compute its inverse matrix $V = U^{-1}$

$$U^{-1} = \begin{bmatrix} U_{11} & U_{12} & U_{13} & U_{14} \\ 0 & U_{22} & U_{23} & U_{24} \\ 0 & 0 & U_{33} & U_{34} \\ 0 & 0 & 0 & U_{44} \end{bmatrix}^{-1} = \begin{bmatrix} V_{11} & V_{12} & V_{13} & V_{14} \\ 0 & V_{22} & V_{23} & V_{24} \\ 0 & 0 & V_{33} & V_{34} \\ 0 & 0 & 0 & V_{44} \end{bmatrix} = V$$

Each square box in Fig. 8 corresponds to a VLSI matrix arithmetic device for handling submatrix computations. The input to the pipeline is an n x n upper triangular matrix U partitioned into k^2 submatrices each of dimension m x m, where $n = k \cdot m$. Listed below are required submatrix computations in four sequential steps to generate the inverse matrix V in a partitioned fashion. Note that U_{ij} and V_{ij} are m x m submatrices and U_{ii} and V_{ii} are both upper triangular m x m submatrices.

The partitioned matrix inversion for $k = n/m = 4$ consists of the following steps:

Step 1. $V_{ii} = U_{ii}^{-1}$ for i=1,2,3,4

Step 2. $V_{i,i+1} = -V_{ii}(U_{i,i+1} \cdot V_{i+1,i+1})$ for i=1,2,3

Step 3. $V_{i,i+2} = -V_{ii}(U_{i,i+1} \cdot V_{i+1,i+3} + U_{i,i+2} \cdot V_{i+2,i+2})$; for i=1,2

Step 4. $V_{14} = -V_{11}(U_{12} \cdot V_{24} + U_{13} \cdot V_{34} + U_{14} \cdot V_{44})$

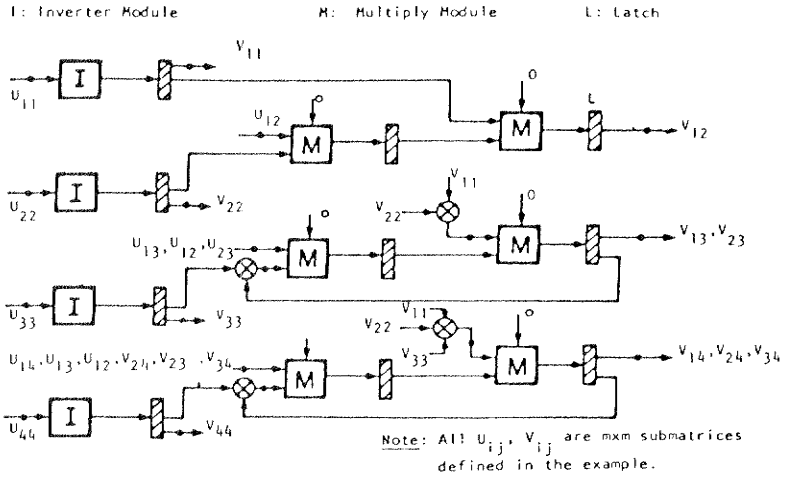

Fig. 8 A pipelined VLSI matrix inverter for partitioned matrix inversion with $k^2 = (n/m)^2 = 4^2 = 16$ submatrix computations.

The I-modules in Fig. 8 are used to perform the inversion of the m x m upper-triangular submatrices at step 1. The M modules are used to perform the cumulative matrix multiplications specified in Steps 2 through 4. The inputs and outputs at four successive computation steps are indicated at the I/O terminals. In general, to invert an n x n triangular matrix with this VLSI pipeline, k I-modules and $2(k - 1)$ M-modules must be used. Thus, the total VLSI module count equals $O(k) = O(n/m)$ for $n >> m$. The total time delay to generate $V = U^{-1}$ equals $O(n^2/m)$ for $n >> m$.

An application of these VLSI matrix manipulation networks is shown in Fig. 9 for the construction of a hardware feature extractor based on Foley and Sammon's algorithm. Arithmetic pipelines can be similarly constructed for matrix multiply, L-U decomposition, and training sample manipulation using these VLSI arithmetic modules. Details of these VLSI matrix solvers can be found in [9,10].

Context-Free Language Recognition

A VLSI systolic array for high-speech recognition of context-free languages is shown in Fig. 10. The recognition process is based on

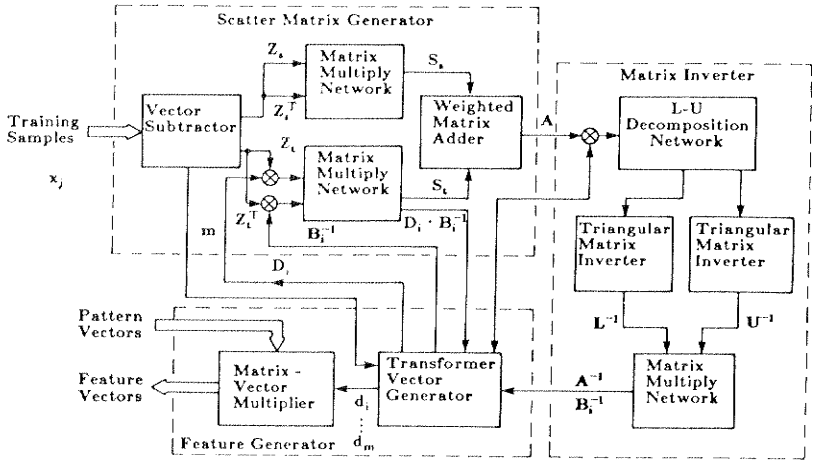

Fig. 9 Functional block diagram of a VLSI feature extractor

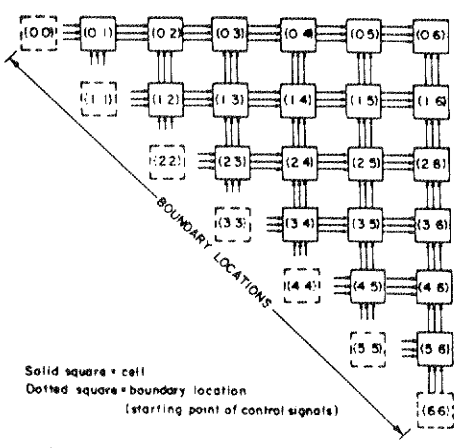

Fig. 10 Systolic array for context-free language recognition

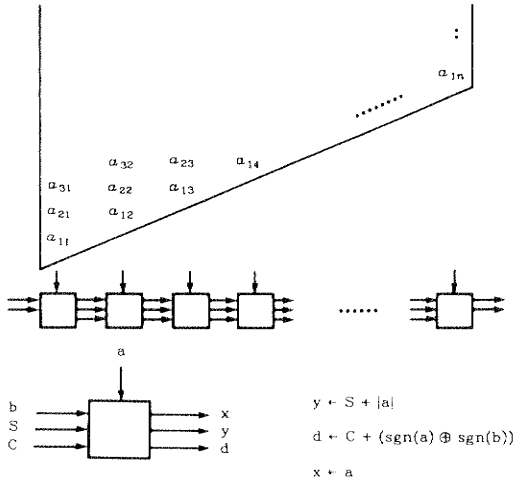

Fig. 11 Processor array, data movement and operations of each processor for feature extraction

the Cocke-Kasami-Younger algorithm. This pipelined triangular array, constructed of n(n+1)/2 processing cells, can be applied in syntactic pattern recognition. Each cell has two unidirectional data channels and one control line along each direction. Data appear as strings of symbols flowing through the recognition matrix from left to right and bottom to top. This two-dimensional array can recognize any input string of length n in 2n time units. This context-free language recognizer and its extension to recognize finite-state languages are described in more detail in [5]. A VLSI architecture for high-speed

recognition of context-free languages using Earley's algorithm has recently been proposed [4].

VLSI Seismic Classification

A special-purpose VLSI processor is presented below for fast classification of seismic waveforms [24]. This special-purpose processor which contains three systolic arrays can be attached to a host computer. Each systolic array has time complexity O(1) provided that input data can be properly supplied. The systolic array for feature extraction contains linearly connected processing elements as shown in Fig. 11. The input data, which are the digitized and quantized seismic waveform coded in binary form, are stored in separate memory modules in a skewed format. Two features, zero-crossing count and sum of absolute magnitudes, are computed. All the n PE's compute the two features simultaneously and pass the partial results to the next PE's.

It is well-known that the Levenshtein distance between two strings can be computed by a dynamic programming procedure. We have developed a processor array for this string matching computation in Fig. 12. The proposed string matcher can be used for any problem where the

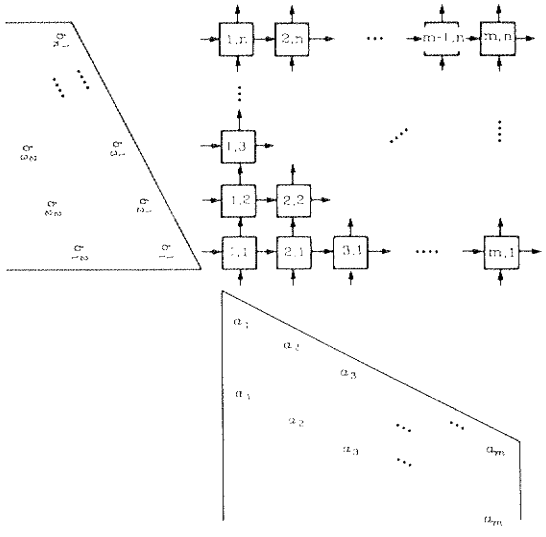

Fig. 12 Processor array and data movement for computing Levenshtein distance

Levenshtein distance computation is required. It can be used for string matching in our seismic recognition, for character string matching in information retrieval or for pattern matching in shape analysis if the object can be represented by a string. The primitive recognizer can also be applied to any minimum-distance recognition problem and vector pattern matching. Simulations have been performed for three systolic arrays: feature extraction array, primitive recognition array and string matching array.

V. DISTRIBUTED SCHEDULING OF VLSI RESOURCES

In general, an interconnection network routes requests from a set of source points to a set of destination points (they may coincide with each other). In a resource sharing mode, the destination points are identical (or sets of identical) resources such as special purpose VLSI chips for which requests or tasks can be delegated to. In this respect, jobs initiated at source processors can be sent to any one of the free resources of a given type at the destination. This is the important point that differentiates resource sharing from address mapping.

Since the system operates continuously, requests from source processors can be initiated at random times. At any time, a set of processors may be making requests and a set of resources are free. It is the function of a scheduler to route the requests in order to connect the maximum number of resources to the processors, that is, to have the maximum resource utilization.

The earliest study of networks for resource sharing has been realized with centralized control. A uni-bus is used in a time shared fashion for connecting peripheral I/O devices to the CPU. Multiple time-shared buses have been used in the PLURIBUS minicomputer multiprocessor. A cross-bar switch has been used in C.mmp although the network is mostly used in address mapping mode. The single or multiple bus approach is a source of bottleneck, and is the least expensive design.

The cross-bar switch is the most expensive network but has the least degree of blocking. A compromise is to use a less expensive network than the cross-bar switch and has less blocking probability than the single bus systems. This has been studied with respect to the Banyan network. A tree network is proposed to aid the scheduler

in choosing a resource to allocate. The tree network has a delay of $O(\log_2 n)$ in selecting a free resource (n is the total number of resources).

A solution which avoids the sequential scheduling of requests is to allow requests to be sent without any destination tags and it is the responsibility of the network to route the maximum number of requests to the free resources. In this way, the scheduling intelligence is distributed in the interconnection network. This approach permits multiple requests to be routed simultaneously. We termed this network a resource sharing network [25,26].

The distributed algorithm is implemented by distributing the routing intelligence into the interconnection network so that there is no centralized control. Each exchange box can resolve conflicts and route requests to the appropriate destinations. If a request is blocked, it will be sent back to the originating exchange box in the previous stage. Request routing is, thus, dynamic and all the exchange boxes operate independently.

The distributed algorithm is illustrated in Figure 13 on an 8 by 8 Omega network. Suppose resources R_0, R_1, R_4 and R_5 are available and status information are passed to the processors. The number on the output/input ports represent the status information received/sent. Assuming that P_0, P_3, P_4, and P_5 are requesting one resource each, the

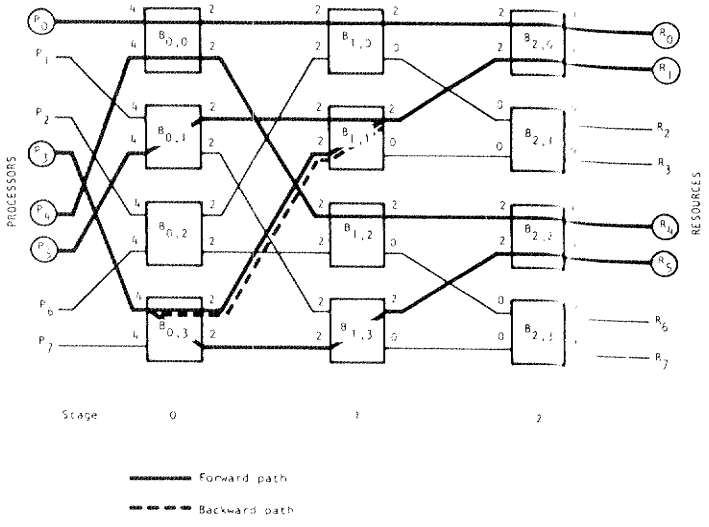

Fig. 13 Example of Omega network with four requesting processors and four free resources, (25% of requests are blocked and backtracked; 100% resource allocation; average delay = 3.50 units)

requests are sent simultaneously to the network after new status information arrives. In stage 0, no conflict is encountered. $B_{1,1}$ and stage 1 receives two requests. Since only one output terminal leads to free resources, the request originating from $B_{0,3}$ is rejected. This request, subsequently, finds another route via $B_{1,3}$ and $B_{2,2}$ to R_5. In this example, each request has to pass through 3.5 exchange boxes on the average before it finds a free resource.

VI. IMAGE DATABASE MACHINE

A database machine for image processing can be identified to have the following functional features. High level database functions such as selection, projection, and join are implemented. These operations are useful for manipulating the image database. On the other hand, low level image processing operations such as histogramming and edge detection are also implemented. An image database machine is, therefore, a conventional database machine enhanced with low level image processing hardware.

We have previously studied the design of a relational database system for images--IMAID [3]--and a relational database machine--DIALOG [16]. IMAID is designed as an integrated database system interfaced with an image understanding system for the efficient storage and retrieval of image and pictures. By using image processing and pattern recognition manipulation functions, structures and features of images are extracted and integrated into relational databases. A relational query language, query-by-pictorial example (QPE) is introduced for manipulating queries regarding spatial relations as well as conventional queries.

A general assumption about VLSI chips are that they are inexpensive. For complex operations, this is not really true due to the fact that external control, timing, memory, and software must be provided. Furthermore, as the types of VLSI chips increase and the degree of replication is large, the system becomes expensive. A solution to this problem is to use a resource sharing network so that a pool of common resources can be used. VLSI chips are distributed into each storage module. They can be used for real-time off-the-track processing. A pool of common resources is also shared among the storage modules. The resource-sharing interconnection network connects these resources to the storage medium.

VII. CONCLUSIONS

Towards the eventual realization of a VLSI multiprocessor system for the said purposes, we identify below a number of research topics. Some of the listed topics have been investigated for years at various research institutions. Before the technology can be applied for practical applications, still many problems need rigorous research efforts. Related previous researches are identified with the listed tasks. These tasks are not meant to be exhaustive, as the subject matter covers almost all disciplines in computer engineering.

- Develop image description languages, image manipulation language, and pictorial query languages [5].
- Study memory hierarchy for on-line image processing, in particular, fast cache memory for imagery data [12].
- Develop backend image database machines, including both image database structures and management policies [16].
- Compression/decompression techniques for image data communication, storage, and display [13,14].
- Develop specialized VLSI functional chips for image processing (see Table 1).
- Investigate dynamic image processing, change detection, and scene analysis towards computer vision and related applications [13].
- Develop resource arbitration networks for resource sharing in a multiprocessor system with shared VLSI resource pool [26].
- Study reconfigurable architectural controls, partitionable interconnection network design, and macropipelining requirements [12].
- Develop effective resource allocation schemes for multiple pipelining, multiple-SIMD array processing, and asynchronous multiprocessing [26].
- Investigate possible use of data flow concept in designing computers for PRIP and artificial intelligence computations [28].
- Integrate image analysis techniques with natural language and speech processing techniques in designing intelligent robots [12].

REFERENCES

[1] F. A. Briggs, K. S. Fu, K. Hwang, and B. W. Wah, "PUMPS Architecture for Pattern Analysis and Image Database Management," IEEE Trans. Computer, October 1982, pp. 969-982.

[2] F. A. Briggs, M. Dubois, and K. Hwang, "Throughput Analysis and Configuration Design of a Shared-Resource Multiprocessor System: PUMPS," Proc. of 8th Annual Symposium on Computer Architecture, May 1981, pp. 67-80.

[3] N. S. Chang and K. S. Fu, "Picture Query Languages for Pictorial Database Systems," IEEE Computer, Nov. 1981, pp. 23-33.

[4] Y. T. Chiang and K. S. Fu, "A VLSI Architecture for Fast Context-Free Language Recognition (Earley's ALgorithm)," Proc. 3rd International Conference on Distributed Computing Systems, Oct. 18-22, 1982.

[5] K. H. Chu and K. S. Fu, "VLSI Architectures for High-Speed Recognition of General Context-Free Languages and Finite-State Languages," Proc. 9th Int'l. Symp. on Computer Arch., Austin, Texas, April 1982, pp. 43-49.

[6] P. E. Danielsson and S. Levialdi, "Computer Architectures for Pictorial Information Systems," IEEE Computer, November 1981, pp. 53-67.

[7] M. J. Duff and S. Levialdi (Editors), Languages and Architectures for Image Processing, Academic Press, N.Y., 1981.

[8] K. S. Fu and T. Ichikawa (Editors), Special Computer Architecture for Pattern Processing, CRC Press, 1982.

[9] K. Hwang and Y. H. Cheng, "Partitioned Matrix Algorithms for VLSI Arithmetic Systems," IEEE Transactions on Computer, December 1982, pp. 1215-1224.

[10] K. Hwang and S. P. Su, "VLSI Architectures for Feature Extraction and Pattern Classification," Journal of Computer Vision, Graphics, and Image Processing, Vol. 24, No. 2, November 1983.

[11] K. Hwang, S. P. Su, and L. M. Ni, "Vector Computer Architecture and Processing Techniques," in Advances in Computers, Vol. 20 (Yovits, Ed.) Academic Press, 1981, pp. 115-197.

[12] K. Hwang and F. A. Briggs, Computer Architectures for Parallel Processing, McGraw-Hill Book Co., N.Y. March 1984.

[13] M. Onoe, K. Preston, and A. Rosenfeld (Editors), Real-Time/Parallel Computing: Image Analysis, Plenum Press, N.Y. 1981.

[14] K. Preston, Jr. and L. Uhr (Editors), Multicomputers and Image Processing, Academic Press, N.Y. 1982.

[15] J. Sklansky and G. N. Wassel, Pattern Classifiers and Trainable Machines, Springer-Verlag, N.Y. 1981.

[16] B. W. Wah and S. B. Yao, "DIALOG - A Distributed Processor Organization for Database Machine," Proc. of NCC, Vol. 49, AFIPS Press, 1980, pp. 243-253.

[17] F. A. Briggs, K. S. Fu, K. Hwang, and J. H. Patel, "PM^4 - A Reconfigurable Multiprocessor System for Pattern Recognition and Image Processing," Proc. of NCC, 1979, pp. 255-265.

[18] R. Hon and D. R. Reddy, "The Effect of Computer Architecture on Algorithm Decomposition and Performance," in High-Speed Computers and Algorithm Organization (Kuck, et al. editors), Academic Press, 1977, pp. 411-421.

[19] J. H. Patel, "Performance of Processor-Memory Interconnection for Multiprocessors," IEEE Trans. on Computers, Vol. C-30, No. 10, pp. 771-780, Oct. 1981.

[20] C. V. Ramamoorthy, J. L. Turner, and B. W. Wah, "A Design of a Cellular Associative Memory for Ordered Retrieval," IEEE Trans. on Computers, Vol. C-27, No. 9, Sept. 1978.

[21] B. D. Rathi, A. R. Tripathi and G. J. Lipovski, "Hardwired Resource Allocators for Reconfigurable Architectures," Proc. of 1980 International Conference on Parallel Processing, pp. 109-117, Aug. 1980.

[22] K. Hwang and K. S. Fu, "Integrated Computer Architectures for Image Processing and Database Management," IEEE Computer, Jan. 1983, pp. 51-61.

[24] H. H. Liu and K. S. Fu, "VLSI Systolic Processor for Fast Seismic Classification," Proc. of 1983 Int'l. Symp. on VLSI Tech., Systems, and Appl., Taipei, Taiwan, March 31, 1983.

[25] B. W. Wah and A. Hicks, "Distributed Scheduling of Resources on Interconnection Networks," Proc. of NCC, AFIPS Press, June 1982.

[26] B. W. Wah, A Comparative Study of Resource Sharing on Multiprocessors," Proc. of 10th Annual Symposium on Computer Architecture, June 1983, pp. 301-308.

[27] K. S. Fu, Syntactic Pattern Recognition and Applications, Prentice Hall, 1982.

[28] S. Hanaki, and T. Temma, "Template Controlled Image Processor (TIP) Project" in Multicomputers and Image Processing, (Preston and Uhr, Editors), Academic Press, New York, 1982, pp. 343-352.