1981 IEEE Computer Society workshop on Computer Architecture for Pattern Analysis and Image Database Management, Hot Springs, VA, November 11-13, 1981.

PUMPS Architecture for Pattern Analysis and Image Database Management†

Faye A. Briggs, Kai Hwang, K. S. Fu, and Benjamin W. Wah

School of Electrical Engineering
Purdue University
West Lafayette, IN 47907

Abstract

The PUMPS architecture consists of N Task Processing Units (TPU) which share a pool of special peripheral processors and VLSI functional units and a common two dimensional main memory (SM) via a block transfer oriented interconnection network. A shared cache is provided between the TPUs and SM for efficient MIMD interprocessor communication. The SM is also connected via a Backend Database Management Network (BDMN) with distributed control to the File Memories (FM), which are disk-based database storage devices.

A Front End Communication Processor (FECP) is used to switch the control of I/O terminals. The FECP and the BDMN manage the relationally-structured image databases in addition to providing the efficient processing capabilities for classical file manipulative primitives. The FECP and BDMN have unique features such as a language interface for relational image database management using Query-by-Picture Example (QPE) and spatial operators.

The architecture of a Database Machine for Image Processing (DMIP) is discussed. Specral hardware for selection and histogramming are presented.

1. Introduction

Image analysis tasks require a wide variety of processing techniques and mathematical tools. Computational processes required for both numerical and combinatorial algorithms are summarized in Table 1. The table reveals the computation-intensive properties in solving LSE, ordered search and sorting, and constrained optimizations that are frequently needed in image processing. This table is not meant to be exhaustive.

In most machine intelligence systems, large single processor computers are employed to process pictorial information. These computers are often designed for general applications and are not tailored for the special needs of pattern analysis. Many image processing tasks require only repetitive Boolean operations or simple arithmetic operations defined over extremely large arrays of picture elements. Hence the use of a large general purpose computer results in intolerable waste of resources. Moreover, a rigidly structured architecture does not lend itself to the flexibility required to process a wide spectrum of PRIP algorithms. Various computer architectures have been proposed in the past for image analysis and a survey of some of these has been given in [FUK 78]. The existing image understanding system architectures consists of the array-structured machines such as ILLIAC IV, BSP and STARAN to the multi-mini/microprocessor systems, such as C.mmp and C.m* [JON 79]. Other image processing systems are the real-time cellular-logic based CLIP-4 and the Toshiba TOSPICS which has a large image memory designed to reduce data transfer time [FUK 78]. Most of the above machines are too rigid in their configurations. Yet other systems, such as the PM^4 [BRI 79] and PASM [SIE 78] which limit their degree of reconfiguration to SIMD and MIMD have been proposed.

These architectures do not satisfactorily address the bottleneck problems of manipulating large image databases especially in a real-time environment. In a study on the effect of computer architecture on algorithm decomposition, a conclusion was reached that cost-effective solutions to many applications such as image analysis problems require some form of functional specialization in the computer architecture [HON 77]. A general-purpose multiprocessor system which is designed basically for a time-sharing environment is not tailored to efficient implementation of specific applicative domain which requires the execution of special classes of algorithms.

It is well known that certain algorithms are more suited to SIMD array processing, some to MIMD multiprocessing. However, there are also some classes of image algorithms in which there is a need to update a common database (such as the same copy of a histogram), or the need to efficiently process a moving image, in which pipelining is most adequate. In order to meet the processing requirements of the explosive amount of pictorial information, concurrency must be exploited maximally at all levels of a computation. At the instruction preprocessing level, pipelining can be implemented within the processor. At the arithmetic and logic operation level, pipelining and parallelism can be implemented in VLSI or special attached processing units [MEA 80]. These units, when integrated with an efficient muttiprocessing system which is organized at the control level in a relatively tightly coupled system, enable configurations of parallelism (synchronized or asynchronous) [KUN 78] or of macropipelines [HAN 73]. In addition, an efficient database system which is harmoniously integrated into the reconfigurable

†This research was supported by NSF-Grant MCS 78-18906.

multiprocessor system will alleviate most of the inefficiencies exhibited by the rigid architectures. The reconfigurability implies that for each given parallel algorithm, an optimum path may be selected through the structure to efficiently execute the set of task. Moreover, the system utilization and throughput may be increased by having several algorithms in execution concurrently in different partitions of the system.

The PUMPS architecture described in this paper exploits the new emerging VLSI computing structures and special peripheral processors such as the attached array processor AP120B [FLP 76] and the image database machine. The PUMPS configuration also resulted from the need to relax the processing modes defined in the PM^4 architecture [BRI 79]. PUMPS organization introduces features such as distributed operating system and cost-effective resource pool configuration for a given applicative domain and workload. The PUMPS architecture can thus be easily reconfigured to suit a specially-chosen application area.

The emphasis of application of the architecture in this paper is in pattern analysis and image database management. A general block diagram of an image analysis system, which has a raw image as input, is shown in Figure 1. There are essentially five processing stages. In the preprocessing stage, image enhancement operations are performed to restore the noisy blurred images. The enhanced imagery data are then segmented according to feature regions. The classification stage recognizes the imagery pattern by indicating its membership in the pattern classes. Finally, structure analysis is performed to produce description of the pattern information.

The image analysis system above is often applied to a continuous sequence of raw images. The processing time of such a repetitive task can be reduced by partitioning the task into a sequence of s processes which can be executed on a macropipeline [HAN 73] configuration of the PUMPS. Each process can be executed in an autonomous unit, called a segment, which operates concurrently with other segments. In the PUMPS, a segment can be an SISD, SIMD, MIMD, pipeline [FLY 72], or specially dedicated VLSI processors. Hence a macropipeline configuration of the PUMPS is composed of segments which are logically arranged so that consecutive processes of a task can be assigned to distinct segments of the macropipeline for processing. In general, the concept of macropipelining is of great importance for the flexibility and efficiency of PUMPS for processing PRIP algorithm.

In the next section, we outline the architectural features of PUMPS. The design issues in the configuration of macropipelining have been enumerated in [BRI81b]. In section 3, the design of a database machine for image processing is discussed. In section 4, a database machine designed for selection and histogramming is presented. Section 5 provides some concluding remarks and directions for future studies.

2. System Architecture of PUMPS

An integrated image analysis and image database processing system is shown in Figure 2. It consists of three closely cooperating subsystems, namely, the man-machine interface using a command interpreter, the image processing and analysis subsystem and the database management subsystem. The PUMPS architecture integrates these subsystems into an efficient multiprocessor system.

The PUMPS is a high-performance multiple microprocessor computer operating with SISD/MIMD task processors and a set of shared PPVUs. A block diagram showing the major components in PUMPs is given in Fig. 3. In general, there are P Task Processing Units (TPUs) in the system, each of which is multiprogrammed. They also can operate in an interactive fashion through the shared PPVUs and shared memory systems. The PTUs can communicate with each other via the Task Processor Communications (TPC) Bus. This intercommunication medium is very effective in passing interrupts, synchronization and other control signals. All the TPUs are connected to the shared-resource pool of special peripheral processors and VLSI units (PPVUs) via a Special Resource Arbitration Network (SRAN). This network provides connections between each TPU and the desired PPVU. The SRAN is a crossbar-like switch with increased connectivity. Besides connecting any TPU to any PPVU, it must provide for arbitrary inter-PPVU paths. As VLSI technology develops, modular crossbar switches [FRA 79] will become more cost-effective because of their regular and local connections [FRA 81]. In case several TPUs 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 TPU's 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, the PUMPS has a dynamically reconfigurable structure. Different applicative environment may be equipped with different functional units. The remaining system resources such as TPU's and shared memories, are designed for general-purpose MIMD computations.

Basic VLSI arithmetic modules for local L-U decomposition, local matrix inversion, and matrix multiplications have been proposed by Hwang and Cheng [HWA80,HWA81b]. VLSI matrix manipulation networks were developed by Hwang and Su [HWA81c]. The objective is to develop a hardware pattern classifier for real-time or high-speed PRIP applications. This class of VLSI pattern classification networks form a part of the dedicated Peripheral and VLSI functional Units (PPVUs) needed in PUMPS for solving PRIP problems. VLSI array/pipeline structures have been also assessed in references [HWA81a,HWA82].

The TPUs perform three basic types of functions: (i) dispatching and initiating PPVU tasks, (ii) executing purely sequential tasks, (iii) participating in MIMD processes and running the operating system. In order to perform the first type of function, a local Task Memory (TM) is provided within each TPU as depicted in Fig. 4. The TM is partitioned into several segments. These consists 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 is shared between the Task Processor (TP) and the PPVUs. The PPVUs are generally passive and the TPU, acting as a controller, must provide the PPVU with a continuous flow of data when the PPVU is processing a task. A Resource Controller and Data Channels (RCDC) in the TPU is used to format and channel the data between the TM and the PPVU.

To match the speed of the TP, a local Task Cache (TC), is provided between the TP and TM. The TC stores the most recently used private instructions and data of the active processes assigned to the TPU. Due to the locality property of programs [DEN 75], most of the references can be made in the cache. A multiprocessor system with private caches may encounter data coherence problem in which several copies of the same block of shared data may exist in different caches at any given time. Basically, two methods have been proposed to solve this coherence problem. In the first method, whenever a processor attempts to update a variable in a TC, the possible copies of the variable in other caches must be modified accordingly before the process execution proceeds. This requires maintaining a central copy of the directories of each cache [TAN 76] or selectively tagging cache blocks with common variables [CEN 78]. This method although efficient may be cost-prohibitive. The second method was proposed for the C.mmp [SAT 80], in which shared data is non-cacheable and reside in shared memory. In the PUMPS architecture a compromise solution was sought to the cache consistency problem by tagging the data as private (P-data) or shared (S-data). The shared data are referenced in a unique cache (SC) shared by all the processors. An analysis of the performance of the shared cache concept shows its effectiveness [DUB 81].

The PUMPS has a distributed memory organization using virtual memory addressing based on paged segments. Within the TC, misses are serviced by the TMMU which initiates the block transfer from the TM to the TC, provided the block exists in the TM. If the block does not reside in the TM but is known to the process, a TM page fault occurs which is also serviced by the page-fault handler that resides in the TMMu. The occurrence of a TM page fault causes the current active process in the TP to be blocked, whereupon a task switch is made to a runnable process also residing in the TPU. By distributing the memory management functions over the entire system, the TPUs are relieved of performing memory management functions and thereby increase their effectiveness in performing useful computations. Such a memory management scheme was proposed in the PM^4 system.

The interleaved TM in TPU serves as a high-spaced buffer between the TPU and the Shared Memory (SMs). The SMs are semiconductor memories organized with multiported ℓ lines and m memory modules per each line as described in [BRI 77] to permit efficient block transfers of information. A block-transfer oriented Processor-Memory Interconnection Network (PMIN), such as the delta network [PAT 79], is used between the TPUs and the SMs. The shared memories are also connected to the File Memories (FM) via a Backend Image Database Management Network (BDMN), which is designed to handle the data transfers from multiple disks. The FM together with a backend computer and the BDMN comprise the database machine for Image Processing (DMIP).

The architectural design of the database machine for image processing is shown in Fig. 5. It 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 BDMN. The design of an efficient image database system of the PUMPS differs from the conventional database design for alphanumeric information processing, because of the large amounts of imagery data involved. The need to interface with image processing packages [CHA 79] and the necessity of providing a high-level language interface to the users, complicates the design issues. However, the design of the DMIP is based on the identification of the properties of various image data manipulation and retrieval operators. These operators are interpretable via a language interface which permits a logical representation of the images. To permit a high degree of concurrency of accesses to the DMIP, its control is distributed. The concurrent accesses are possible if the image database is deadlock-free serializable and security controllable [WAH 79]. The control mechanism requires the partitioning and replication of images and the placement of these partitions on the secondary storage. The images on the file memory are also dynamically reorganized for efficient retrievals. The design of the DMIP will be discussed in more detail in the next two sections.

A Front-End Communication Processor (FECP) is used to switch the control of I/O terminals and performs preprocessing task such as interactive file editing. The FECP together with the BDMN provide efficient processing capabilities for classical file manipulative primitives in addition to providing the users access to the whole multiprocessor system.

3. Backend Image Database Management System

In the design of PUMPS, imagery data must be staged to the SM before they are accessible by the TPU's. This is efficient only when the same data are repeatedly used. In this case, the overhead of the transfer is balanced by the efficiency in processing. On the other hand, some images processing operations resemble the conventional database processing operations, namely, a large amount of data is processed in a small number of times. In this case, it is more efficient to process the data directly on the storage medium and elminates the congestion in the processor interconnection network (PMIN). Furthermore, it relieves the TPU's to process operations that are more CPU-bound. The Database Machine for Image Processing (DMIP) is the result of an archiectural approach which distributes processing power closer to devices on which data are stored and offloads database processing functions from the main computer [LAN79].

Nearly all the research on database machine are directed towards conventional databases which can be classified into four types. The first type is the backend system which utilizes minicomputers to enhance the database processing of large host com-

puters. The functions of the backend system include access validation, storage management, concurrency control and I/O control. IDM by Britton La, Inc. is an example of this class.

The second type of database machine utilizes the single-instruction-multiple-data-stream (SIMD) principle. This concept is extended from backend machines in which the database processing functions are moved to a lower level. In this class, simpler, less costly processors are dedicated to a small block of data. This concept is exemplified by the Logic-Per-Track device in which processing logic are duplicated for each track of the disk and the keys in different tracks are searched in parallel. Examples of this design are Tape DRUM [HOL56], Slotnick's Logic Per Track disk [SLO70], RAPID [PAR72], CASSM [LIP78,SU79], RAP [OZK77,SCH79], RARES [LIN76], DBC [BAU76,KER79,BAN79] and Chang's Major/Minor Loop Machine [CHA78]. If the replication goes further down to the bit level, an associative memory results. This is exemplified by RELACS [BER79] in which STARAN is used as the associative memory.

The third type of database machine design is based on the multiple-instruction-multiple-data-stream (MIMD) principle. This design is exemplified by DIRECT [DEW79] in which the processing logic are interconnected with an array of CCD memory modules through a cross-bar switch. A variable number of processors can be assigned to process a database query. This design offers more flexibility and better load balancing and allows the processors to be shared among the storage modules.

The last type of database machine design is based on the distributed system principle. Modules which consists of a storage medium with enhanced intelligence, communicate with each other through a communication network. DIALOG is an example of this class [WARI80]. The advantages of this type of database machine are its flexibility and its expandability. Further, heterogeneous storage devices can be accommodated in the system. Techniques developed for the optimization of distributed databases are also applicable for the design of this class of database machines.

4. Database Machine for Image Processing (DMIP)

A database machine for image processing can be designed as one or a combination of the aforementioned classes. The following functional features are identified. 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, 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--REDI [CHA79] and a relational database machine--DIALOG [WAH80]. REDI [CHA79] is designed as an integrated database system interfaced with an image understanding system for the efficient storage and retrieval of images 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) [CHA81] is introduced for manipulating queries regarding spatial relations as well as conventional queries. Conventional database manipulation operators such as selection and join are used to manipulate imagery objects stored in relations.

DIALOG [WAH80] is a relational database machine which facilitates query processing and parallel processing. The system is hierarchically organized so that it is easily extendable. Figure 6 shows the stucture of a 16 module backend network. Communications at higher levels can be reduced by carefully allocating related files to individual clusters so that most of the communications are intra-cluster. Each data module in DIALOG is a storage sub-system such as a disk module with processing capabilities to perform selection and join. Figure 7 shows the architecture of a data module in DIALOG. The join processing module has been described in detail before [WAH80] and will not be repeated here. We describe in this paper the design of the selection processing module which can be used to process selections and histogramming [WAH81].

Figure 8 shows the architecture of the selection processing module. There is a common part of the circuit that is used for both selection and histogramming. This consists of n circulating registers which contain keys to be matched. The details of the associative logic is shown in Figure 9. The timing and control module controls the associative logic which is capable of performing equality match, proximity and threshold operations [RAM78]. A subset of the associative logic are enabled and selected keys are compared with the data output from the disk. The multiple match resolution circuit is used to select the first 1 from a set of responding 1's and is similar to those designed for associative memories [FOS68,RAM78]. The Software Programmable Logic Array (SPAL) computes the output $Z = f(x_1,\cdots,x_n)$ with a user selected function f of n variables. The output Z is used to control the gate which filters off unnecessary information from the disk. Since the selection function does not change throughout the matching of the entire database, the function f can be input before the selection begins. f is represented in disjunctive normal form and each term of f can be input sequentially. If n is larger than the number of possible inputs of a single SPAL, the n inputs can be partitioned into sets and fed into multiple SPAL's. Multiple outputs $Z_1,Z_2,\cdots Z_k$ will be obtained and supplied to a single SPAL to compute Z. The circuit can be extended systematically into a tree of SPAL's.

The selection circuit operates in period in which a period is the length of time for a database record (tuple) to be output from the disk. The database data is fed to all the associative logic. As selected fields of the record is detected, a subset of the associative logic are enabled and the results of the matching are stored in temporary flip flops. After the end of record is detected, the values $X_1,\cdots,X_n$ stored in the temporary flip flops are output to the SPAL which computes the value Z and decides whether the record is to be discarded.

The advantages of this selection circuit are three fold : (a) Generality: The design is more general than the finite state automation approach [HOL78] and it allows arbitrary logic functions (with limitation imposed by the size and number of the registers) to be implemented. It follows the parallel discrete comparators approach of Stellhorn [STE74] except that all the functions are implemented in hardware and is, therefore, much faster. (b) Reconfigurability: The size of a key is not limited to the size of a register. For equality and threshold operations, suppose it is needed to compute $D \geq B$ where D is a field of the database tuple and B is a key. If B and D have a size greater than the size of the register, they can be partitioned into k sub-keys, $B = B_1,\cdots,B_k$, $D = D_1,\cdots,D_k$ and the operation $D \geq B$ is equivalent to $(D_1 > B_1) \vee ((D_1 = B_1) \wedge (D_2 > B_2)) \vee \cdots \vee ((D_1 = B_1) \wedge (D_2 = B_2) \wedge \cdots (D_k > B_k))$. Therefore, the registers can be reconfigured to perform operations on keys longer than the size of a register. (c) Speed: Because the computation of the function f by the SPAL is completely overlapped with the associative matching of the next record, the operation is pipelined and can result in a higher throughput than conventional selection circuits where the function f is computed simultaneously with the matching operations.

To use the circuit to compute the histogram of an image, the registers will be set to the different thresholds and the associative logic are set to perform the less than functions. As each pixel of image is output, all the registers with keys smaller than the value of the pixel is enabled and the value of X is set to 1. The multiple match resolution circuit selects the correct interval where the counter should be incremented. Based on the results of the counters, a second pass of the imagery data would allow thresholding to be performed.

5. Conclusion

The PUMPS architecture generates a number of related research topics that need to be investigated. One basic topic is the development of an appropriate operating system and language that can be used to effectively control and express the processing tasks. Another operating system related task is the memory management method and its implementation. It could be said that Pictorial Image Analysis algorithms cannot be classified as having properties of the more general programs. For example, an algorithm for performing a specific function on a VLSI processor may be well behaved. In which case, parameters such as looping distance and page reference string will be well defined for a given task dimension. With such parameters we can develop an efficient image buffer management policy and determine appropriate buffer size for each VLSI unit. VLSI technology can be explored for fast manipulation and update of relational image data base and permit a modular growth [KUN 79].

Implementation of a processing task on the PUMPS architecture requires the efficient allocation of system resources. In particular, the effective utilization of the PPVUs is important. The match of the set of PPVU configurations with the specific application needs makes the PUMPS very attractive. The unification of the image database management subsystem with the user-oriented multiprocessor system in the PUMPS permits an effective on-line processing of pattern analysis problems and the interactive management of imagery data. Continued research efforts are being conducted in other aspects of the PUMPS.

The design of efficient parallel algorithms for image processing is also very important. In [CHI81], a parallel processing algorithm for distance computation in syntactic pattern recognition is presented. The proposed algorithm has time complexity O(n+m) and requires min(n,m)+1 processors where n and m are length of strings. Three different parallel systems have been discussed, a general SIMD, a dedicated SIMD and a MIMD system. Research is also planned for other algorithms discussed in Table 1.

References

[AGR 80] D. P. Agrawal and R. Jain, "Computer Analysis of Motion Using a Network of Processors," Proceedings of the 5th International Conference on Pattern Recognition, December 1980.

[ARM 79] C. V. Armstrong et al, "An Adaptive Multimicroprocessor Array Computing Structure for Radar Signal Processing Applications," Proceedings of the 6th Annual Symposium on Computer Architecture, 1979.

[BAN 79] Banerjee, J., Hsiao, D. K., and Kannon, K., "DBC -- A Data Base Computer for Very Large Data Bases," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979.

[BAU 76] Baum, R. I. and Hsiao, D. K., "Data Base Computers -- A Step towards Data Utilities," IEEE Trans. on Computers, Vol. C-25, No. 12, Dec. 1976.

[BER 79] Berra, P. B. and Oliver, E., "The Role of Associative Array Processors in Data Base Machine Architecture," IEEE Computer, March 1979.

[BRI 77] F. A. Briggs, "Organization of Semiconductor Memories for Parallel-Pipelined Processors," IEEE Trans. on Comput., Vol. C-26, No. 2, Feb. 1977, pp. 162-169.

[BRI 79] Briggs, F. A., Fu, K. S., Hwang, K. and Patel, J. H., "PM ^{4} - A Reconfigurable Multiprocessor System for Pattern Recognition and Image Processing," Proc. of NCC 1979, pp. 255-265.

[BRI 81a] F. A. Briggs, M. Dubois, and K. Hwang, "Throughput Analysis and Configuration Design of a Shared-Resource Multiprocessor System: PUMPS," Proceedings of the 8th Annual Symposium on Computer Architecture, May 1981.

[BRI 81b] F. A. Briggs, K. Hwang, K. S. Fu and M. Dubois, "PUMPS Architecture for Pattern Analysis and Image Database Management," Proc. of Conf. on Pattern Recognition and Image Processing, Dallas, TX, 1981.

[CEN 78] L. M. Censier and P. Feautrier, "A New Solution to Coherence Problems in Multicache Systems," IEEE Trans. on Computers, Vol. C-27, No. 12, December 1978.

[CHA 78] Chang, H., "On Bubble Memories and Relational Data Base," 4th Int'l. Conf. on Very Large Data Bases, Berlin, Sept. 1978.

[CHA 79] Chang, N. S. and Fu, K. S., "A Relational Data Base System for Images," Report TR-EE 79-28, School of Electrical Engineering, Purdue University, May 1979.

[CHA 80] J. K. Cheng and T. S. Huang, "Algorithms for Matching Relational Structures and Their Applications to Image Processing," Purdue University, Technical Report TR-EE 80-53, December 1980.

[CHI 81] Y. Chiang and K. S. Fu, "Parallel Processing for Distance Computation in Syntactic Pattern Recognition," Proc. IEEE Computer Society, Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Nov. 11-13, 1981.

[DEN 75] P. J. Denning and G. S. Graham, "Multiprogrammed Memory Management," Proc. IEEE 63, 6, June 1975.

[DEW 79] DeWitt, D. J., "DIRECT -- A Multiprocessor Organization for Support Relational Data Base Management Systems," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979.

[DUB 81] M. Dubois, F. A. Briggs, "Efficient Interprocessor Communications for MIMD Multiprocessor Systems," Proc. of 8th Annual Symposium on Computer Architecture," pp. 187-196, 1981.

[FAR 78] E. J. Farrell, "Processing Limitations of Ultrasonic Image Reconstruction," Proceedings of the Conference on Pattern Recognition and Image Processing, June 1978.

[FLP 76] Floating Point System, Inc., AP-120B Processor Handbook, Portland, Oregon, Pub. No. 7259-02.

[FLY 72] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," IEEE Transactions on Computers, Vol. C-21, No. 9, September 1972.

[FOS 68] C. C. Foster, "Determination of Priority in Associative Memories," IEEE Trans. Electronic Comp., Vol. EC-17, pp. 788-789, Aug. 1968.

[FRA 79] M. A. Franklin, S. A. Kahn and M. J. Stucki, "Design Issues in the Development of a Modular Multiprocessor Communications Network," Proceedings of the 6th Annual Symposium on Computer Architecture, 1979.

[FRA 81] M. A. Franklin, "VLSI Performance Comparison of Banyan and Crossbar Communications Networks," IEEE Transactions on Computers, Vol. C-30, No. 4, April 1981.

[FUK 76a] Fu, K. S. and Rosenfeld, A., "Pattern Recognition and Image Processing," IEEE Trans. on Computers, Vol. C-25, No. 12, Dec. 1976, pp. 1336-1346.

[FUK 76b] Fu, K. S., ed., Digital Pattern Recognition, Springer-Verlag, 1976.

[FUK 78] Fu, K. S., "Special Compute Architectures for Pattern Recognition and Image Processing," Proc. 1978 National Comp. Conf., pp. 1003-1013.

[FUK 80] Fu, K. S. and Mui, J. K., "A Survey on Image Segmentation," Pattern Recognition, Vol. 12, 1980.

[HAB 76] A. N. Habermann, Introduction to Operating System Design, Science Research Associates, Inc.

[HAN 73] W. Hansler, "The Concept of Macropipelining with High Availability," Elektronische Rechenanhagan, No. 15, 1973, pp. 269-274.

[HOL 56] Hollander, G. L., "Quasi-Random Access Memory Systems," AFIPS Conf. Proc., EJCC, 1956.

[HOL 78] L. A. Hollar and D. C. Roberts, "Current Research into Specialized Processors for Text Information Retrieval," Proc. of 4th Int'l. Conf. on Very Large Databases," Berlin, 1978, pp. 270-279.

[HON 77] Hon, R. and Reddy, D. R., "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.

[HWA 80] Hwang, K. and Cheng, Y. H., "VLSI Computer Structures for Solving Large-Scale Linear System of Equations," Proc. of the 1980 Int'l. Conf. on Parallel Processing, pp. 217-230.

[HWA 81a] Hwang, K., Su, S. P. and Ni, L. M., "Vector Processing Computer Architecture," in Advances in Computers, Vol. 20, Academic Press, New York 1981. (to appear)

[HWA 81b] Hwang, K. and Cheng, Y. H., "Partitioned Matrix Algorithms and VLSI Structures for Large-Scale Matrix Computations," Proc. of IEEE 5th Symp. on Computer Architecture, May 1981, pp. 222-232.

[HWA 81c] Hwang, K. and Su, S. P., "A Partitioned Matrix Approach to VLSI Pattern Classification," IEEE Computer Society, Workshop on Computer Architecture for Pattern Analysis and Image Database Management, Nov. 11-13, 1981.

[HWA 82] Hwang, K. and Briggs, F. A., Computer Architecture and Parallel Processing, McGraw-Hill Book Co. (in press), to appear in 1982.

[JON 79] A. K. Jones and P. Schwartz, "Experience Using Multiprocessor Systems: A Status Report," Carnegie-Mellon Univ., Technical Report, CMU-CS-79-146, Oct. 1979.

[KER 79] Kerr, D. S., "Data Base Machine with Large Content Addressable Blocks and Structural Information Processors," Computer, Vol. 12, No. 3, March 1979.

[KUN 76] H. T. Kung, "Synchronized and Asynchronous Parallel Algorithms for Multiprocessors," in Algorithms and Complexity: New Directions and Recent Results, J. F. Traub, ed., New York: Academic Press, 1976.

[KUN 79] Kung, H. T. and D. L. Lehman, "VLSI Algorithms for Relational Data Bases," Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Penn., 15213 March 1979.

[LAN 79] Langdon, Jr., G. G., "Data Base Machine, An Introduction," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979.

[LIN 76] Lin, C. S., et al., "The Design of a Rotating Associative Memory for Relational Data Base Applications," ACM Trans. on Data Base Systems, Vol. 1, No. 1.

[LIP 78] Lipovski, G. J., "Architectural Features of CASSM: A Context Addressed Segment Sequential Memory," Proc. 5th Ann. Symp. on Comp. Arch., ACM-SIGARCH.

[MEA 79] Mead, C. A. and Conway, L. A., Introduction to VLSI Systems, Addison Welsey Pub. Co., 1979.

[NIC 79] G. Nicolac and K. H. Hohne, "Multiprocessor System for the Real-Time Digital Processing of Video-Image Series," Elektronische Rechenanhagen, No. 21, 1979.

[NUD 80] R. Nudd, "Image Understanding Architectures," Proceedings of the AFIPS, 1980.

[OZK 77] Ozkarahan, E. A., et al., "Performance Evaluation of a Relational Associative Processor," ACM Trans. on Data Base Systems, Vol. 2, No. 2, June 1977.

[PAR 72] Parhami, B., "A Highly Parallel Computing System for Information Retrieval," AFIPS Conf. Proc., 1972, FJCC, Vol. 41, part II.

[PAT 79] Patel, J. H., "Processor-Memory Interconnection for Multiprocessors," 6th Int'l. Symp. Comp. Arch., Apr. 1979, pp. 168-177.

[PAV 78] Pavlidis, T., Structure Pattern Recognition, Springer-Verlag, 1978.

[PER 77] Persoon, E., and Fu, K. S., "Shape Discrimination Using Towering Descriptors," IEEE Trans. SMC, March 1977.

[PRO 80] Proceedings of 1980 IEEE Picture Data Description and Management Workshop, Aug. 27-29, Asilomar, Calif.

[RAM 78] Ramamoorthy, C. V., Turner, J. L., and Wah, B. W., "A Design of a Cellular Associative Memory for Ordered Retrieval," IEEE Trans. on Computers, Vol. C-27, No. 9, Sept. 1978.

[ROS 76] Rosenfeld, A., and Kak, A. C., Digital Picture Processing, Academic Press, 1976.

[SAT 80] M. Satyanarayanban, Multiprocessors: A Comparative Study, Prentice-Hall, Inc., Englewood Cliff, N.J.

[SCH 79] Schuster, S. A., et al., "RAP.2 -- An Associative Processor for Data Base and its Applications," IEEE Trans. on Comp., Vol. C-28, No. 6, June 1979.

[SIE 78] Siegel, H. J., et al., "Control of Partitionable Multimicroprocessor System," Proc. 1978 Int. Conf. Parallel Processing, pp. 9-17.

[SLO 70] Slotnick, D. L., "Logic Per Track Devices," Advances in Computers, Academic Press, 1970.

[STE 74] W. H. Stelhorn, "A Processor for Direct Scanning of Text," Proc. of 1st Workshop on Comp. Arch. for Non-numeric Processing, Dallas, 1974.

[SU 79] Su, S. Y. W., et al., "The Architectural Features and Implementation Techniques of the Multi-cell CASSM," IEEE Trans. on Computers, Vol. C-28, No. 6, June 1979.

[WAH 79] Wah, B. W., A Systematic Approach to the Management of Data in Distributed Data Bases, Ph.D. Dissertation, University of California, Berkeley, 1979.

[WAH 80] B. W. Wah and S. B. Yao, "DIALOG -- A Distributed Processor Organization for Database Machine," Proc. National Computer Conference, Vol. 49, 1980, pp. 243-253.

[WAH 81] B. W. Wah, "A Selection Circuit for Text Processing and Histogramming," presented at 1981 Syracuse University Minnowbrook Workshop on Database Machine, Minnowbrook, New York, 1981.

[YAS 80] Y. Yasuda, M. Dubois, and T. S. Huang, "Data Compression for Check Processing Machines," Proceedings of the IEEE, Vol. 68, No. 7, July 1980.

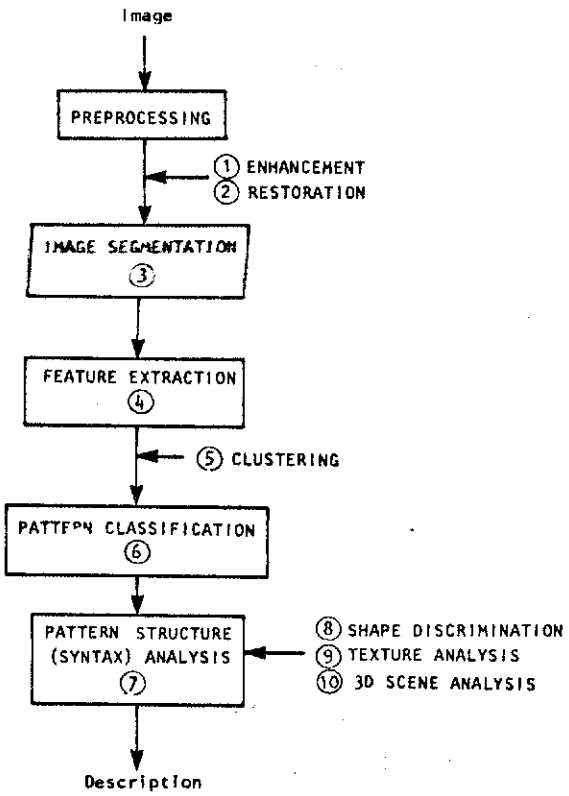

Fig. 1 A General Block Diagram of Image Analysis System

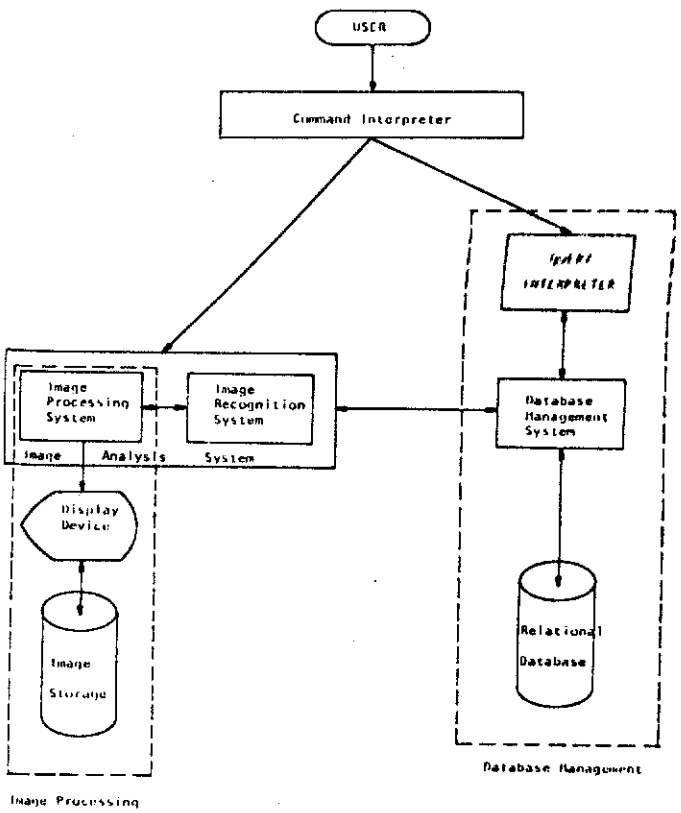

Fig. 2 Integrated Image Analysis and Database Management System

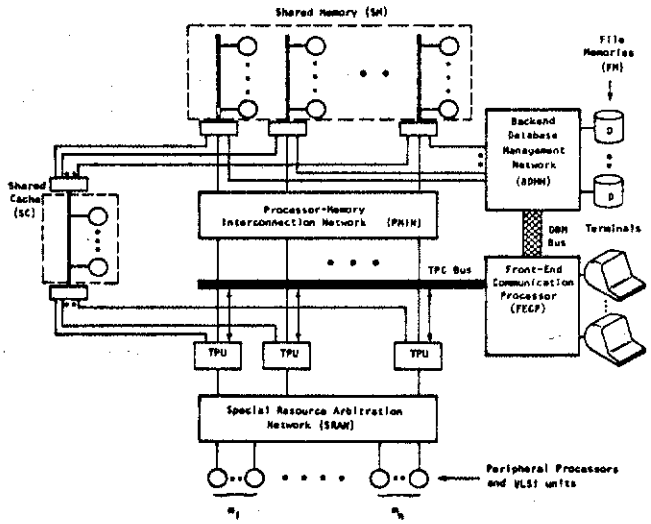

Fig. 3 The System Architecture of PUMPS

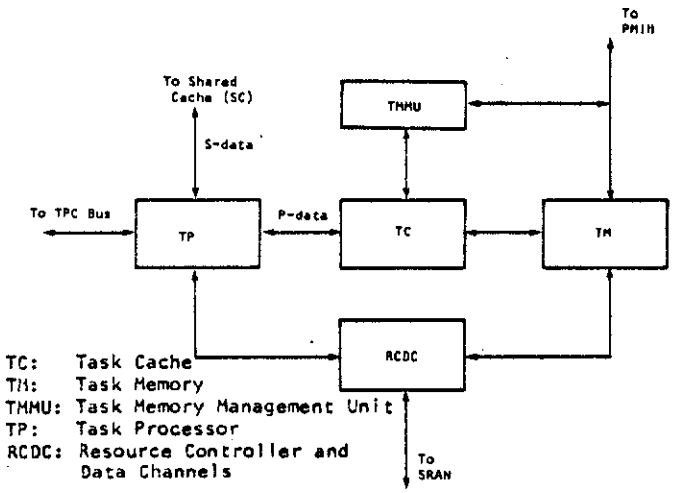

Fig. 4 Architecture of a Task Processing Unit (TPU)

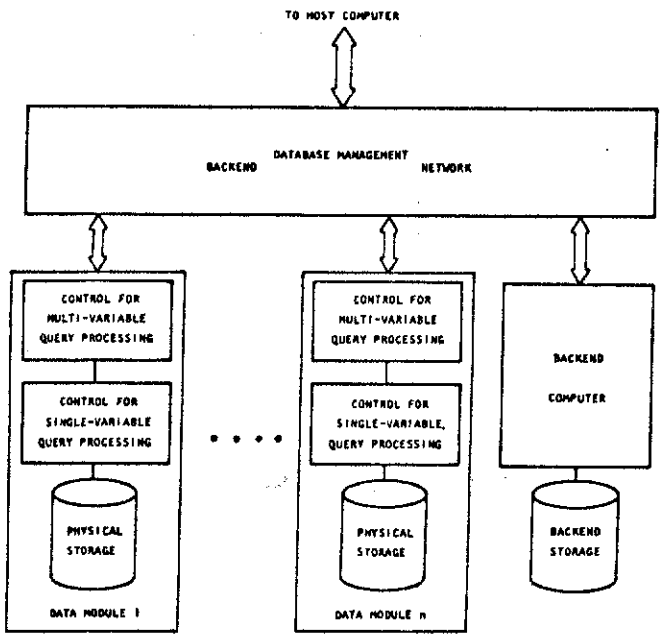

Fig. 5 Database Machine for Image Processing

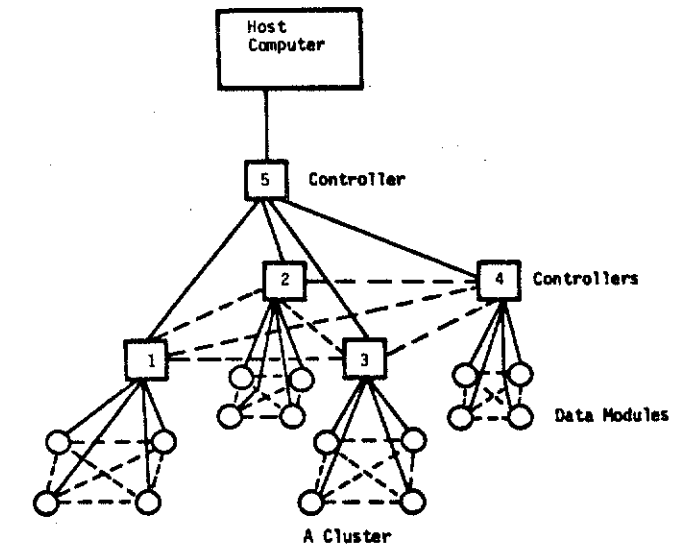

Fig. 6 An Example of the Hierarchical Network in DIALOG.

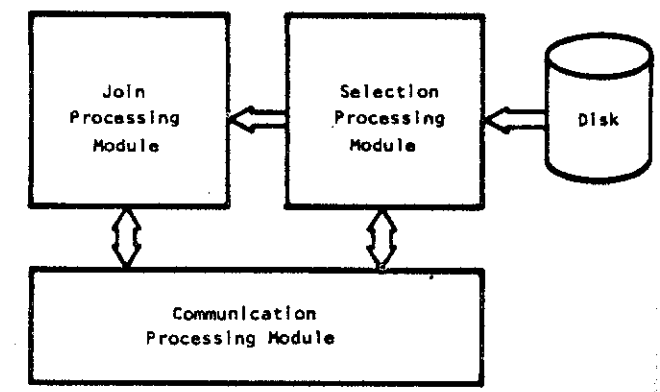

Fig. 7 Architecture of a data module in DIALOG.

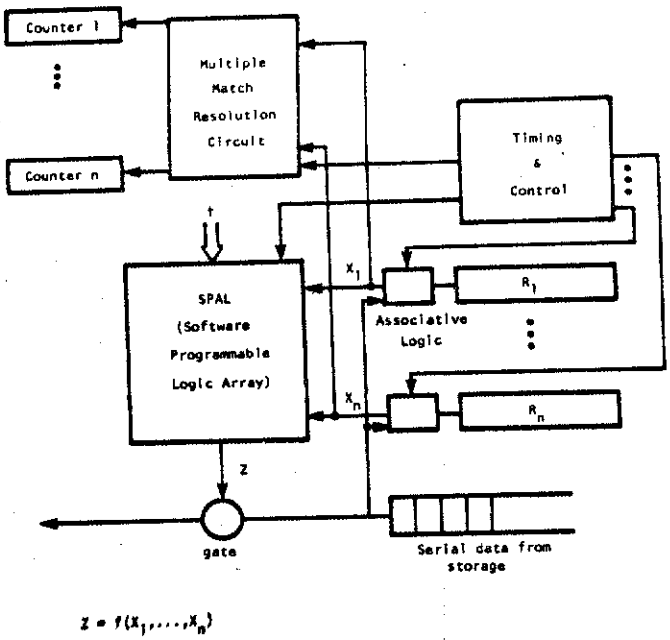

Fig. 8 Architecture of the selection Processing Module for selection, projection and histogramming.

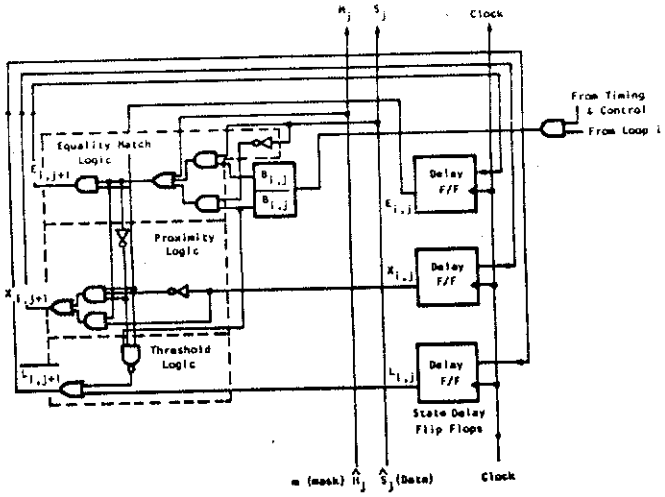

Fig. 9 Associative logic with equality, less than, greater than and proximity searches. [CRAM 78, WAH 79]

Table 1. Image Analysis and Data Base Management Problems and Their Required Computations.

Computational Tasks Required \ Image Analysis and Database Problems	Image Enhancement	Image Restoration	Image Segmentation	Feature Extraction & Selection	Clustering Techniques	Statistical Pattern Classification	Pattern Structure (Syntax) Analysis	Shape Discrimination	Texture Analysis	Scene Analysis	Alphanumeric and Image Database Management	Image Query Language Processing
Iterative relaxation methods	X	X	X		X			X		X		
Constrained and unconstrained optimizations	X	X	X	X		X		X				
Matrix manipulation (Solving Algebraic equations)		X		X	X	X						
Transformation and convolution	X	X		X					X			
Interpolation and functional evaluation		X	X	X				X	X			
Histogramming	X		X	X					X			
Logical operations (boolean)	X	X	X	X		X	X			X	X	X
Similarity retrieval and spatial operations								X			X	
Set Manipulation and relational operations			X		X		X			X		
Ordered search or heuristic search	X		X		X		X	X		X		
Sorting, pattern matching and graph operations					X		X	X		X	X	
Backtracking and syntax parsing							X	X	X	X		X
Language translation and compiling					X		X					X
Image Display Operations									X		X	