

Parallel Architectures for
Database Systems

A. R. HURSON

Computer Engineering Program
Department of Electrical Engineering
Pennsylvania State University
University Park, Pennsylvania

L. L. MILLER

Department of Computer Science
Iowa State University
Ames, Iowa

S. H. PAKZAD

Computer Engineering Program
Department of Electrical Engineering
Pennsylvania State University
University Park, Pennsylvania

M. H. EICH
B. SHIRAZI

Department of Computer Science
Southern Methodist University
Dallas, Texas

1. Introduction 108
2. Classification of Database Machines 110
2.1 Rosenthal's Classification (1977) 112
2.2 Berra's Classification (1977) 113
2.3 Champine's Classification (1978) 113
2.4 Hsiao's Classification (1980) 113
2.5 Su's Classification (1980) 114
2.6 Bray and Freeman's Classification (1979) 114
2.7 Song's Classification (1983) 114
2.8 Qadah's Classification (1985) 116
2.9 Boral and Redfield's Classification (1985) 116
2.10 A New Classification 116

3. Database Machines	119
3.1 Low VLSI-Compatible Database Machines	120
3.2 Semi VLSI-Compatible Database Machines	123
3.3 High VLSI-Compatible Database Machines	130
3.4 Related Efforts	143
4. Conclusion and Future Directions	144
References	146

1. Introduction

Since the early 1970s the complexity of conventional database management systems has gradually increased by *i*) the number and size of databases, *ii*) the number and type of application programs, and *iii*) the number and type of on-line users. For example, among the estimated 20,000 U.S. Government databases, the patent office has a database of 65 billion characters and a query retrieval of one item every 24 seconds (Hsiao, 1983). As another example, the Ballistic Missile Defense System requires a distributed and dynamic database which can maintain and upgrade itself in a complex and rapidly evolving battle ground. The system should be able to perform about 60–120 million operations per second to verify, trace, classify and eliminate a threat (Fletcher, 1984). It is projected that by 1995 the Defense Mapping Agency will have a database of 10^{19} bits supporting 1000 on-line users needing 10^{14} bits each (Mahoney, 1981).

Conventional systems using typical software approaches fail to meet the requirements of the aforementioned applications. A software implementation of direct search on an IBM 370/158 can process approximately 100,000 characters per second. But even if this speed could be increased tenfold, it would take 18 hours to search the 65 billion characters in the U.S. patent office's data file system.

To avoid the need for an exhaustive search, most existing software systems are based on an indexing mechanism. An indexed system improves performance by means of a sophisticated software system and additional redundancy of data. Nevertheless, this has created some additional problems: the indexed structures require extra storage for the indices. It has been estimated that in a fully inverted file, the space needed to store the index ranges from 50% to 300% of the size of the database itself (Bird *et al.*, 1977; Haskin, 1978). In addition, the use of a directory creates complexity in the search, update, and delete algorithms. In fact, indexing merely shifts the processing burden from searching to merging of pointer lists, and so it offers only a partial solution in terms of the efficiency of database operations.

The inefficiency of conventional systems in handling large databases can be associated to the existing semantic gap, computation gap and size gap. These gaps stem from the fact that *i*) conventional systems by their very nature are

sequential machines, *ii*) a conventional ALU is structured for numeric computations (e.g., the CRAY-1 is able to perform 250 million floating point operations per second, while it can not handle more than 15 million characters per second), and finally *iii*) the memory hierarchy has a passive role in the organization and hence there is a massive amount of data movement between processing elements and storage units. Therefore, there is a great need to design and develop new approaches specialized to the demand of a database environment.

Since the mid 1970s a great deal of effort has been directed towards the design of special-purpose architectures for efficient handling of large database systems, namely Data Base Machines (DBMs). Hsiao (1980) examined the motivation for the move to an architectural approach and Demurjian *et al.* (1986) looked at the issue of performance evaluation of such machines. Recent advances in technology have forced drastic changes in the architectural aspects of these machines. According to the current state of the art, it is possible to have a complexity of 10^5 transistors together on a chip, and it is anticipated that by 1990, this complexity will be improved by an order of magnitude. Nevertheless, with the exception of memory organization, the great potential of such a development has not yet been fully exploited. Such a gap between theory and practice is partially due to the lack of suitable architectures and algorithms for hardware implementation. The hardware algorithms should reduce the communication requirements, as well as the computation throughout the chip by replication of a few basic operations in time or space. This implies simplicity, regularity and modularity in the design of an architecture.

Recent database machines proposed in Bonuccelli *et al.* (1985), Eich (1987), Gajski *et al.* (1984), Garcia-Molina *et al.* (1984), Hurson (1981), Kung and Lehman (1980), Lehman (1986), Oflazer and Ozkarahan (1980), and Song (1980) are attempts to design systems based on the constraints imposed by the technology. These systems are highly regular and simple in nature and hence suitable for VLSI implementation. However, adaptability and performance of these models for dynamic large databases have to be studied in more detail.

In this paper, our primary goal is to examine the impact of current technology on the design of special-purpose database machines. To accomplish this end, we present a new classification scheme that incorporates the suitability of the design for use with current technology. We initiate the discussion by looking at what makes a good classification scheme and by briefly discussing several of the schemes that have appeared in the literature.

Finally, we use our classification scheme to provide a survey of the DBMs that have appeared in the literature. The survey focuses on the adaptability of the designs to current technology. Earlier surveys of DBMs centered around what we will call "historical machines" (Ambarder, 1985; Qadah, 1985; Smith

and Smith, 1979). These machines are of historical significance, but in light of current technology seem to be of little value as potential future designs. However for the sake of completeness they will be briefly discussed. Our primary emphasis in this paper is on designs more suited to current technology.

In the next section, we examine previously proposed classification techniques, discuss their weaknesses, and propose a new scheme which we feel is more appropriate for examining novel architectures. Section 3 covers the DBMs based on our proposed classification.

2. Classification of Database Machines

Since the late 1960s, different researchers have attempted to classify various proposed or designed computer architectures (Feng, 1976; Flynn, 1966; Händler, 1982). Flynn (1966) has classified the concurrent space according to the multiplicity of instruction and data streams. According to Flynn's classification the computer architectures fall into four groups: SISD (Single Instruction stream—Single Data stream), SIMD (Single Instruction stream—Multiple Data stream), MISD (Multiple Instruction stream—Single Data stream) and MIMD (Multiple Instruction stream—Multiple Data stream). Unfortunately, Flynn's classification does not address the interactions among the processing modules and the methods in which processing modules in a concurrent system are controlled. As a result one can classify a pipeline computer as an SISD machine, since both instructions and data are provided sequentially. Händler (1982) has defined the concurrent space as a three-dimensional space $t = (k, d, w)$ in which

- k is the number of control units interpreting a program.
- d is the number of arithmetic and logic units (ALUs) controlled by a control unit.
- w is the word length or number of bits handled in one of the ALUs.

According to this classification a von Neumann machine with serial or parallel ALUs is represented as $(1, 1, 1)$ or $(1, 1, M)$, respectively. To represent pipelining at different levels (e.g., macro pipeline, instruction pipeline and arithmetic pipeline), and to illustrate the diversity, sequentiality and flexibility or adaptability of an organization, the above triplet has been extended by three variables (e.g., k', d', w') and three operators (e.g., $+$, $*$, $\vee$) where

- k' represents the macro pipeline—the number of control units interpreting tasks of a program, where the data flow through them is sequential.

- d' represents instruction pipeline—the number of functional units managed by one control unit and working on one data stream.
- w' represents arithmetic pipe—the number of stages.
- $+$ represents diversity—existence of more than one structure.
- $*$ represents sequentiality—for sequentially ordered structures.
- $\vee$ represents flexibility or adaptability—for reconfigurable organization.

According to this extension to Händler's notation, the CDC 7600 and DAP are represented as

$(15 * 1, 1 * 1, 12 * 1) * (1 * 1, 1 * 9, 60 * 1)$
 and $(1 * 1, 1 * 1, 32 * 1) * [(1 * 1, 128 * 1, 32 * 1) \vee (1 * 1, 4096 * 1, 1 * 1)]$,
 respectively.

These classifications suffer from the fact that either they do not uniquely identify a specific organization, or they cannot thoroughly determine the interrelationships among different modules in an organization.

In general a classification scheme should

- (i) Categorize all existing as well as foreseeable computer designs.
- (ii) Differentiate essential processing elements.
- (iii) Assign an architecture to a unique class.

As one can discuss, these efforts are directed to *i*) generalize and identify the characteristics of different designs, *ii*) formulate a systematic mechanism by which different designs can be analyzed and compared against each other, and *iii*) define a systematic mechanism to transform the solutions from one design to other designs.

The research on design and development of database machines during the past two decades has resulted in a large body of database machines with different architectures, characteristics, specifications, and target applications. These database machines are interrelated in many ways and share common characteristics. Through a classification and taxonomy of database machines, one can provide grounds for a basic understanding and a unified view of the body of research on database machines. Thus, by knowing the class in which a particular machine belongs, one can make intelligent decisions and deductions with regard to that machine without knowing its specific characteristics. Within this framework several classifications of database machines along with their general characteristics are discussed.

Hsiao (1980) classified databases into two general categories, namely *formatted* and *unformatted* structures. Formatted databases are mainly time-variant entities and are subject to extensive alteration as well as search operations. Unformatted databases (bibliographic or full text) are archival in nature and are processed by searching for a pattern or a combination of

patterns. As a result, operations on the formatted databases are based on the contents of the attribute values of the records, while in the unformatted databases the patterns are unpredictable combinations of terms and words. The major theme of this paper centers around the formatted databases. Hence, we use the terms database and database machines to refer to the formatted database as the underlying data structure. The literature has addressed several classifications of database machines. In the following, these classifications along with their general characteristics are discussed.

2.1 Rosenthal's Classification (1977)

This classification categorizes the database machines according to their functions in a computing system into three groups: *Smart peripheral systems*, *large backend*, and *distributed network data node* (Rosenthal, 1977). A smart peripheral system at the low end is an *intelligent controller* for conventional rotating storage, which mainly moves some of the access method functions from the mainframe to the controller. At the high end it represents a controller capable of handling database operations at the secondary storage level. Such an architecture can be organized to perform database operations on a unit of data (i.e., data on the fly) or on several units of data (i.e., logic per track). This concept is a solution for the data communication problem, which eliminates one aspect of the 90–10% rule.

A large backend system is a semi-autonomous special-purpose unit, which is closely coupled to a general-purpose host machine, and has access to the databases. Its function is to provide database management services for the host machine. Higher performance due to overlapping between operations in the backend and frontend machines, reducing the semantic gap due to the hardware implementation of the database primitive operations at the backend module, and reducing data transportation are the major thrusts of this architecture. However, one has to pay a penalty for lower reliability and performance due to the added processing module(s) and inter-communication among them.

A distributed network data node is a loosely coupled autonomous processor which is an element in a network consisting of both general- and special-purpose processors. This topology is an extension of the backend approach where a database processor is shared with a group of processors in the network. Such an architecture bears the advantages of the backend design with a better utilization of the database processor. However, it enforces a complicated intercommunication protocol among the network nodes and a more complex and sophisticated mechanism to distribute and share the processing power of the database processor among different network elements.

2.2 Berra's Classification (1977)

Berra has classified the research on database machines into *backend systems*, *logic in memory*, and *large associative processors* (Berra, 1977). A backend system in this classification has the same configuration and characteristics as discussed by Rosenthal. A logic-in-memory system essentially matches the "high end" of the smart peripheral system class in the previous classification. It is based on the logic-per-track concept proposed by Slotnick (1970).

A large associative processor performs database operations in associative fashion, acting as a backend processor or as a stand-alone unit. High performance, flexibility, and elimination of the name mapping resolution problem (due to the embedded parallelism in the architecture and content addressability of the data) are major advantages of this approach. However, the existing size gap between size of databases and the capacity of associative memories enforces the use of a partitioned data set mechanism. In addition, high-speed associative processing requires parallel I/O operations and a sophisticated buffering scheme between back-up storage and the associative processor.

2.3 Champine's Classification (1978)

According to this classification, database machines are grouped as *backend processor for a host*, *intelligent peripheral control unit*, *network node*, and *storage hierarchy* (Champine, 1978). The first three groups have the same definitions as the three groups discussed by Rosenthal. Storage hierarchy is based on the old concept of memory hierarchy and virtual memory, which has been proposed to overcome the existing size gap and access gap found in conventional systems. It should be noted that the efficiency of this approach is heavily due to locality of references in the database, which can be achieved at the expense of higher data dependence (i.e., partitioned data set).

2.4 Hsiao's Classification (1980)

Hsiao (1980) has classified database machines into three groups, namely the *cellular logic approach*, the *associative array approach*, and the *functionally specialized approach*. The cellular logic approach falls at the high end of the intelligent controller systems category. In this approach, a cell is the basic building block with some storage capacity and ability to perform database operations on its data set. The associative array approach is the same as the large associative processor discussed before. The functionally specialized approach is a cluster of specialized units, each with considerably different processing speeds and memory capacity requirements. This approach allows the construction of a relatively well balanced computer to overcome the

existing bottlenecks by providing each component with the right amount of processing power and memory capacity.

2.5 Su's Classification (1980)

This classification groups database machines into four classes *cellular systems*, *high-speed associative systems*, *integrated database machines*, and *backend computers* (Su *et al.*, 1980). The first three are similar to Hsiao's definition and the last one is the same as the backend systems in previous classifications.

These classifications are similar in that for the most part they are based on how the database machines in a computing environment will be used, but they reveal nothing about the characteristics or architecture of such machines. Bray and Freeman (1979) have proposed a two-dimensional database space based on the degree of parallelism and the location where the data is searched.

2.6 Bray and Freeman's Classification (1979)

According to this classification database machines are grouped into five categories: *single processor indirect search*, *single processor direct search*, *multiple processor direct search*, *multiple processor indirect search*, and *multiple processor combined search* (Bray and Freeman, 1979). Direct search processing implies the ability to search database at the secondary storage level, while indirect search represents the fact that data needs to be transferred to an intermediate storage media before the search can be conducted.

Single processor indirect search (SPIS) represents the conventional von Neumann type architecture with no degree of parallelism. Naturally, such an architecture bears all the aforementioned deficiencies in handling large databases. Single processor direct search (SPDS) represents a conventional system enhanced by searching capability at the secondary storage. As a result, only the desired records or their specified parts are sent to the host. Multiple processor direct/indirect search (MPDS/MPIS) machines represent parallel versions of SPDS/SPIS organizations. A MultiProcessor Combined Search (MPCS) is a combination of the MPDS and MPIS organizations, in which search is performed on the data loaded into intermediate storage, while multiple processing units are assigned to the blocks of intermediate storages. It is clear that this classification is general enough to cover the previous classifications. Table I summarizes the relationships between the classifications.

2.7 Song's Classification (1983)

Song (1983) has defined database machines as computer systems that are enhanced by special-purpose logic for handling database operations. With

TABLE I
RELATIONSHIPS AMONG DIFFERENT CLASSIFICATIONS OF DATABASE MACHINES

Bray & Freeman (1980)	Rosenthal (1977)	Berra (1977)	Champine (1978)	Su <i>et al.</i> (1980)	Hsiao (1980)	Song (1983)	Qadah (1985)
SPDS	Smart peripheral system	—	Intelligent peripheral control unit	—	—	Logic at secondary storage with static allocation (sequential operation)	SOSD with relation indexing on the disk search
SPIS	Large backend processor	Backend system	Backend processor	—	—	Logic at primary storage with static allocation (sequential operation)	SOSD with relation or page indexing off disk search
MPDS	Smart peripheral system	Logic in memory	Intelligent peripheral control unit	Cellular logic	Cellular logic	Logic at secondary storage with static/dynamic allocation (Parallel Operation)	MOMD/SOMD with relation indexing on disk search
MPIS	Large backend processor	Backend system/ large associative processor	Backend processor	High-speed associative system/ backend system	Associative array	Logic at primary memory with static/dynamic allocation (Parallel Operations)	MOMD/SOMD with relation or page indexing off disk search
MPCS	Distributed net- work data node	—	Network node	Integrated database machine	Functionally specialized system	Logic at primary/ secondary memory with static/dynamic allocation. Medium to high level of parallelism	MOMD/SOMD with relation or page indexing on/off/ hybrid search

such a view, he has classified these machines according to three parameters:

- (i) The place where hardware logic is applied. This could be either at (close to) secondary storage or at primary memory.
- (ii) Allocation of logic to storage unit. This could be static or dynamic. Naturally, a dynamic allocation offers better resource utilization.
- (iii) Degree of logic distribution, which defines the number of storage elements associated with each processing unit. This parameter represents the degree of parallelism, and hence it directly affects the performance.

2.8 Qadah's Classification (1985)

Qadah (1985) has extended Bray and Freeman's database space by a third dimension, namely indexing level. The coordinates of this database space are the indexing level, the query processing place, and the processing multiplicity. The indexing coordinate represents the smallest addressable unit of data. Along this coordinate database machines can be grouped into database indexing level, relation indexing level and page indexing level. The query processing place determines the location where data is searched. This could be off the secondary storage, on the secondary storage, or a hybrid of both. The third coordinate represents the degree of parallelism. Along this coordinate we can group database machines into single operation stream-single data stream (SOSD), single operation stream-multiple data stream (SOMD), and multiple operation stream-multiple data stream (MOMD).

2.9 Boral and Redfield's Classification (1985)

Boral and Redfield (1985) have proposed a classification based on two components: a catalog and an anatomy. The catalog describes and discriminates among the DBMs using several parameters. It can describe a DBM and provide an overview of its working mechanisms. The anatomy systematically describes the architecture and hardware organization of a DBM. Due to the fact that we view the scheme more as means of describing the hardware than as a general classification, we have chosen not to attempt to integrate it into the remainder of the discussion.

2.10 A New Classification

During the past 35 years, the transition from vacuum tubes to VLSI has increased the processor speed by four orders of magnitude, and has reduced the logic circuit size and memory cell size by factors of 500 and 6400, respectively (Block and Galage, 1978; Klass, 1984). For example, the first

electronic computer, ENIAC, consisted of 18,000 vacuum tubes and 15,000 relays. The U-shaped computer was 100 feet long and nearly 9 feet high and weighed over 30 tons. ENIAC was able to perform merely 5000 additions or subtractions per second. By comparison, with the strong emergence of VLSI technology, today an entire 32-bit microprocessor or a 1M-bit RAM memory can be incorporated into a single chip. The reduction in the gate switching delay and wire length on the one hand and miniaturization of circuits on the other hand has a direct effect on the clock rate and hence in closing the computation gap. In addition, such a trend has increased the sophistication of the supporting software and the migration of the software functions into hardware.

In the late 1960s Moore predicted that component density on a chip would quadruple every three or four years (Mukherjee, 1986). This is partially due to the development of high-resolution lithographic techniques, increases in the size of the silicon wafer, and growth of the accumulated circuits and layout design issues leading to an improved architecture capable of exploiting the technology. However, as improvements in technology approach the limit (the speed of light), Moore's law is no longer applicable and the emphasis is shifted in the direction of advances in system architecture and organization.

Migration of software functions into hardware, combined with advances in the technology's intrinsic speed, have reduced the computation gap. However, there has always been the need for much more computer performance than is feasible with a simple straightforward design. For example, there currently exist several projects with requirements of 10^9 instructions per second (e.g., 1 ns per instruction) balanced against technologies that are approaching the speed-of-light transmission limitation (e.g. 30 cm/ns). To overcome these limitations, computer designers have long been attracted to designs in which the computer hardware is simultaneously processing more than one basic operation—i.e., concurrent systems. Within this general category are several well recognized techniques: parallelism, pipelining, multiprocessing and distributed processing. However, a discouraging aspect of these activities is the lack of extensive published knowledge regarding the rationale for various designs, or comparing the results achieved by various approaches.

The aforementioned classifications of database machines do not address the effect of the advances in technology on the architecture and adaptability of a specific architecture for the current and foreseeable technology. We believe such a parameter should be used as a coordinate of the database space. Recent developments in technology have influenced the architecture of database machines in two directions:

- (i) Reconfiguration and reevaluation of the previously designed database architectures according to the constraints imposed by the technology.

The evolution of RAP (Ofizer and Ozkarahan, 1980; Schuster *et al.*, 1979) demonstrates the validity of this discussion.

- (ii) The design of new architectures based on the constraints imposed by the technology (Bonuccelli *et al.*, 1985; Eich, 1987; Gajski *et al.*, 1984; Hurson, 1981; Kung and Lehman, 1980; Song, 1980).

The classification of database machines based on their architectural characteristics and technology allows one to

- (i) Group the machines with common architectural characteristics into one class. Thus, a solution for a particular fundamental problem based on a specific machine or technology can be easily extended and applied to the other machines within the class. For example, a practical solution to the chip I/O pin limitation problem can be extended to other VLSI-based database machines;
- (ii) Pinpoint the weak and strong points of each machine architecture based on comparing it to similar (intra-class) and different (inter-class) machines. Such a study results in a good set of parameters and tools for evaluating existing and future machines. For example, the tree-structured database machines are especially suited for sorting and searching operations, but are not necessarily efficient for performing the join operation. On the other hand, associative-based architectures are proven to be efficient for many database operations, but are prohibitively expensive to develop. Therefore, the tree-structured database machines should be compared and evaluated based on their efficiency in handling the join operation. The associative-based database machines should be evaluated based on how effectively they can handle large files using associative memories. In addition, an architectural classification of database machines allows the designers of new database machines first to match their specifications and hardware requirements with one of the existing classes and then to proceed with their specialized architecture if a perfect match is not found;
- (iii) Anticipate the future trends and the new developments in the field of database machine design. We witnessed the trend in the use of associative memory technology in the design of database machines in the 1970s. The impact of current technology on the design of database machines in the 1980s is quite obvious by the attempts to use systems with systolic array characteristics.

For our architectural classification of database machines we propose adding a fourth dimension to Qadah's classification, creating a database space

of the following four coordinates: technology adaptability, degree of parallelism, query processing place, and indexing level. According to these parameters one can characterize an architecture based on its ability to handle *i*) computation gap, *ii*) semantic gap, *iii*) size gap, *iv*) data transmission, and *v*) name mapping resolution.

The indexing level determines the smallest accessible unit of data. Such a coordinate determines name mapping resolution and the proper protocol which one should take to enforce the security and guarantee the system's integrity. The query processing coordinate illustrates the capability of the design in handling the data transmission problem. It is characterized by the following four classes: searching at the secondary storage, searching close to the secondary storage, indirect search, or a combination of the previous three. The degree of parallelism describes an architecture's ability to close the computation gap. Along the technology adaptability coordinate, database architectures are characterized as high-, semi- or low-adaptable designs. Due to continued advances in device and storage technology, the adaptability issue is measured with respect to these two parameters.

3. Database Machines

The overall structure and organization of several database machines will be examined with respect to the parameters of the new classification. In addition, in our evaluation we will consider the flexibility of a design for different database operations. In other words, we will look at whether an architecture requires different modules for basic database operations or a module can be used to carry out different basic operations. It should be noted that in this study we concentrate on the architectures that utilize the relational data model, and hence basic database operations refer to the relational algebra operations.

This section is divided into three subsections, based on the adaptability of database machines to current technology. The first subsection examines some of the "low adaptable" organizations. These machines were generally reported before the widespread application of the current advances in technology. In addition, the literature has not examined their potential for the current technology. The second subsection addresses those machines that have not been designed according to the constraints imposed by the technology. However, some efforts have been made to modify and adapt their designs for current technology. Finally, the last subsection addresses the machines that are designed for current technology — i.e., their topologies bear the constraints imposed by technology.

3.1 Low VLSI-Compatible Database Machines

The machines in this class represent the early DBM architectures. These are architectures that were primarily concerned with establishing the conceptual feasibility of DBMs. However, we should note that some of these designs are inherently adaptable to the current advances in device technology. The general trend of these designs was primarily based on associative operations. They can be classified as fully parallel or block-oriented associative organizations.

3.1.1 Fully Parallel Associative Architecture

The first special-purpose processor for handling non-numeric operations was designed by Lee and Paull (1963). This bit-parallel word-serial hardware machine is composed of an array of identical cells. Each cell is a small finite-state machine which can communicate with its neighbors. The array is controlled via a set of programming commands broadcast among all the cells by a controller. Each cell includes a set of bistable devices called cell elements. Cell elements are divided into cell state elements and cell symbol elements. Cell symbol elements hold a bit pattern corresponding to a character in the alphabet. There also exists a matching circuit which matches the bit pattern broadcast by the controller and the bit patterns stored in each cell. Data is organized as a single string of symbols divided into substrings of arbitrary lengths by delimiters. The system is suitable for text retrieval operations, but because of the hardware cost of the memory, the memory size could not be large enough for handling databases. In order to overcome the propagation timing problems, Gaines and Lee (1965) redesigned the logic circuitry. The new design was able to perform simultaneous operations of shifting and marking strings, but the cost was high for practical implementation. In the early 1970s some new hardware for handling databases using associative processors was designed (Defiore and Berra, 1973; Linde *et al.*, 1973; Moulder, 1973). These systems are able to perform the operations in "bit slice" fashion. That is, parallel operations are performed on one bit of all words at a time. In each approach, a critical assumption has to be made that the entire data file can fit into the associative memory. Comparisons between an associative-processor-based architecture and a similar von Neumann architecture have shown the superiority of associative processors over the von Neumann design with respect to retrieval, update and storage operations (Berra, 1974). Defiore and Berra (1973) have shown that associative processors need to use 3 to 15 times less storage compared to a database system using inverted list organization. Moreover, the response time is 10 times faster.

The system implemented by Moulder (1973) is a typical design of a database system based on an associative processor. The system is restricted to moderate size databases (6×10^7 bits). This system, is composed of a general-purpose

computer (Sigma 5) augmented by a four-array STARAN computer (Rudolph, 1972). The database is subdivided into a fixed size sector. A sector is read in and will be searched. Then the system reads in another sector and performs the same search, continuing in this fashion until the entire database is searched.

The system designed by Linde *et al.* (1973) is an integrated associative computer. This system is composed of two associative processing units with 2K, 256-bit elements, which are linked to an IBM 370/145.

It is worthwhile to mention that, besides the above special-purpose designs based on associative processors, there are some general-purpose associative processors. STARAN is an example of such a system (Davis, 1974, 1983; Rudolph, 1972). It is composed of an associative array processor (one to 32 modular associative processor) with an interface (custom interface unit) to the users. It also has a conventionally addressed control memory for program storage and data buffering. Control signals generated by the control logic unit are fed to the processing elements in parallel, and all processing elements execute the instruction simultaneously. The STARAN symbolic assembler language APPLE provides a flexible and convenient assembler for programming without the complex and costly indexing, nested loops, and data manipulation constructions required in conventional systems.

3.1.2 Block-Oriented Database Architecture

In the late 1960s, Slotnick (1970) proposed the "Logic per Track" concept. This idea, which has served as the guideline for a large number of proposed hardware designs, is simply based on assigning a read/write head to each track of a disk. Later on, this concept was enhanced by adding more logic to each read/write head (cell) and hence the cellular organization. The general philosophy behind the proposed architectures based on Slotnick's idea is the selection of tuples at the secondary device level. These cellular organizations are composed of a set of identical cells supported by a general-purpose computer. Each cell acts as a small computer, capable of performing some basic operations on the data residing in its memory. As a result, just a subset of data is selected and transferred to the frontend machine (e.g., the solution to the data transmission problem). Figure 1 shows the general diagram of a cellular organization, in which N processors ($P_1, P_2, \dots, P_N$) are controlled by a controller and to each processor P_i a memory M_i is assigned ($1 \leq i \leq N$).

A *Logic Per Track Retrieval System* is a fixed-head disk with a logic chip attached directly to each read/write head (Parker, 1971). Each head is allowed to search for a fixed key, use garbage collection on the track, and insert, and/or delete a record and/or a key. Since there is no communication path between heads, the maximum length of each record is logically restricted to the size of a

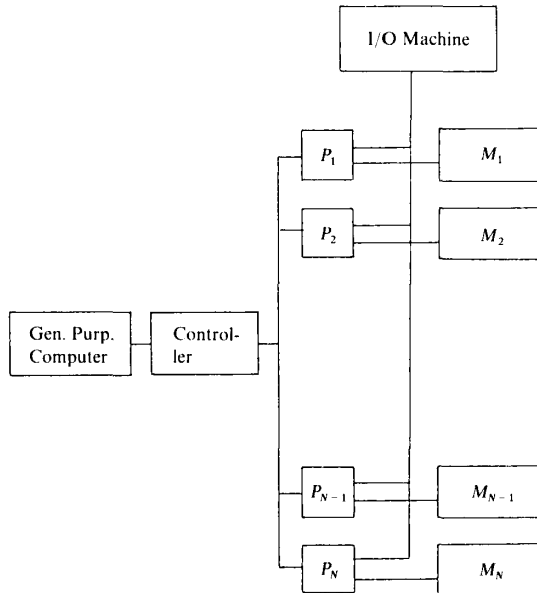


FIG. 1. General structure of a cellular organization.

track. Information on the tracks is divided into three groups: holes, keys, and data record. Moreover, each record should be preceded by a set of mark bits for more complex instructions which require more than one disk rotation.

RAPID (Rotating Associative Processing for Information Dissemination) is an array of identical cells controlled by a control unit (Parhami, 1972). *RAPID* is based on Slotnick's idea and Lee's machine (Lee and Paull, 1963). This combination will reduce the cost of memory (compared to Lee's memory) at the expense of execution time. Information on the tracks is in bit-parallel word-serial format. The strings of data, which are stored on the secondary memory, are read into the cell storage one character at a time, processed and stored back. Since data is handled one character at a time, each character should have some control storage associated with it to store the temporary result, similar to the one proposed in Lee's design.

CASSS (Context Associated Segment Sequential Storage) is another non-numeric-oriented system based on a cellular organization, augmented by a fixed-head disk (Healy *et al.*, 1972). Data files are partitioned into segments of equal length, one segment per track. Data is organized in bit-serial word-serial

fashion. Each record is of variable length, and is preceded by a set of control bits for storing the intermediate results. CASSS is also capable of performing non-numeric operations, which includes automatic garbage collection.

CASSM (Context Addressed Segment Sequential Memory) is a database machine capable of handling the hierarchical data model, as well as the relational and network data models (Su, 1979). Data is stored in a bit-serial word-serial fashion. One particular feature of CASSM is that each processing element can directly communicate with its adjacent neighbors.

RARES (Rotating Associative RELational Store) uses a very different organization from CASSM and RAP (Lin *et al.*, 1976). Tuples are laid out on the secondary storage device across the tracks, rather than along the tracks (bit-serial byte-parallel fashion). Each set of tracks used to store a relation in this fashion is called a band. The number of tracks in a band is a function of the size of tuples in the relation. In this design a cell will be assigned to each band, which can overcome the inefficiency of previously mentioned cellular organizations, at the expense of slower execution time and additional hardware for association of cells to bands of different sizes. Another special feature in this system is the elimination of mark bits associated to each tuple stored on the secondary storage, since mark bits are implemented in the cells (search modules) rather than on the secondary storage.

According to Qadah's classification the aforementioned machines can be classified as SOMD architectures. However, the Search Processor for database management systems proposed by Leilich *et al.* (1977) belongs to the class of MOSD machines.

A slightly different design than the logic-per-track DBMs, namely CAFS (Content Addressable File Store) (Babb, 1979), was proposed in the late 1970s. CAFS utilizes a data filter between the disk storage device and the host computer, which screens the data before sending it to the host. Its hardware reads out up to 12 tracks simultaneously and then multiplexes the data as input to multiple processing elements each executing a different operation.

3.2 Semi VLSI-Compatible Database Machines

3.2.1 RAP—Relational Associative Processor

The Architecture of RAP was proposed in 1975 (Ozkarahan *et al.*, 1975). However since then, in order to increase its compatibility with the advances in technology, it has undergone several revisions (Ofizer and Ozkarahan, 1980; Schuster *et al.*, 1979). Originally RAP was based on the concept of logic per track proposed by Slotnick (1970). It is a collection of identical and autonomous processors (cells) communicating with a general-purpose frontend machine through a controller (Fig. 2). As the memory rotates under the read/

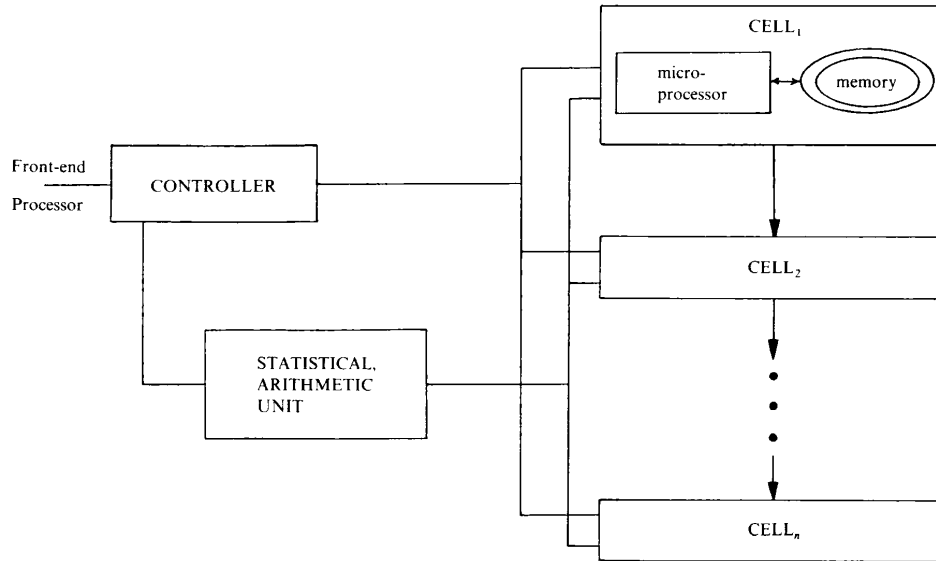


FIG. 2. Organization of RAP architecture.

write heads, tuples are read into the buffer and circulated through the cell's logic and written back on the track after a delay. As a result, memory tracks are searched in parallel in associative fashion, and hence the execution time is independent of the size of the database and is a function of the rotation time and the number of rotations required to execute a query.

The development of the rotating memory technology and the complexity of the controller to synchronize and coordinate the operations within the cells have shown that such a simple concept is not economically feasible. In addition, it has been shown that this model has very low performance in handling inter-relational operations such as the join (Ofizer, 1983). These deficiencies have evolved the RAP architecture into RAP-2 and RAP-3, where system dependence on a specific secondary technology has been reduced. In addition, the functionality and flexibility of each cell has been upgraded through the concept of microprogramming. Finally, special algorithms have been devised for efficient handling of "hard" database operations (Ofizer, 1983). However, the compatibility of the system to the recent advances in technology has to be studied in detail.

3.2.2 A Database Computer (DBC)

The database computer (Banerjee *et al.*, 1978, 1979) is a specialized backend system capable of managing $10^9 - 10^{10}$ bytes of data represented in different data models. In addition, it has a built-in support for security, and different relational operations.

DBC uses two forms of parallelism. First, an entire cylinder is processed in parallel. Second, the system performs queries in a pipeline fashion by separate units around two rings (Fig. 3): the structure loop and the data loop. The function of the structure loop is to translate predicates in the query into physical addresses of the minimal addressable units (cylinders).

Along the structure loop the keyword transformation unit transforms the request into a series of indices to the structure memory. The structure memory is an inversion table for the database; it produces the logical address of data required to answer a request. The structure memory information processor then uses the output of the structure memory to perform boolean operations on these logical addresses. The index translation unit then translates the logical addresses into physical addresses. These physical addresses are used to access the data residing on the mass memory. Finally, the security filter processor is used to enforce the security specifications.

VLSI compatibility of the join operation has been evaluated and discussed by Hsiao (1983). The major drawback of this system is its performance for large databases. The performance analysis of this system in supporting relational databases shows that a general-purpose machine can perform better

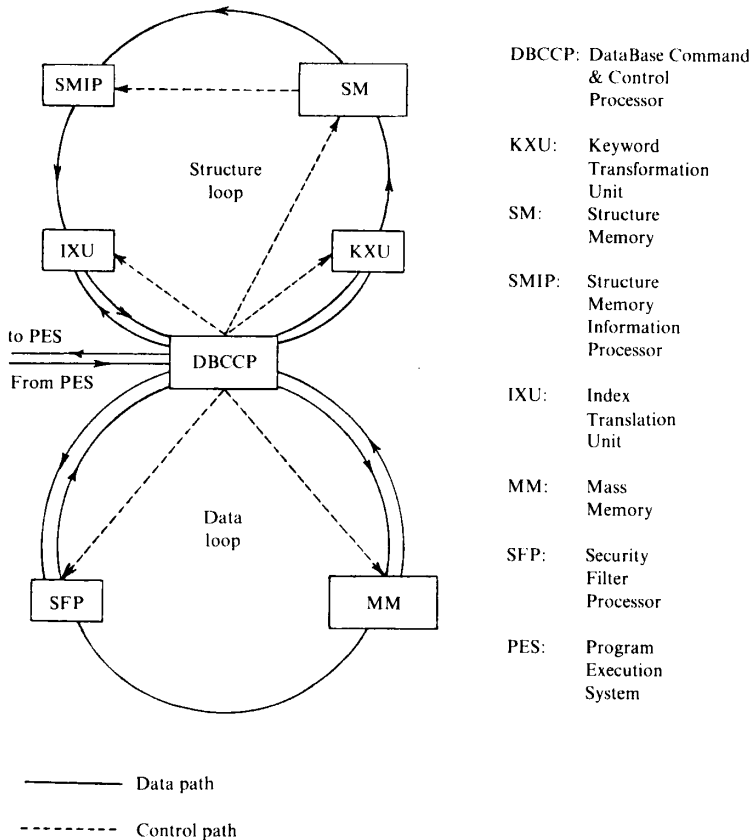


FIG. 3. DBC organization.

than DBC (Ofizer and Ozkarahan, 1980). Furthermore, since the system relies on the concept of index processing, the same complexity of algorithms and data dependence in conventional systems will exist in DBC, which could affect the performance.

3.2.3 *DIRECT—A Multiprocessor Backend Database Machine*

Specific features which distinguish this architecture from the other database machines are based on the fact that *i*) DIRECT is an MIMD organization, and

ii) it can dynamically allocate resources to the queries. As a result, it can simultaneously support intra-query and inter-query concurrency. Since the late 1970s its architecture has been under study and evaluation (Bitton *et al.*, 1983; Boral *et al.*, 1982; Boral and DeWitt 1981; DeWitt 1979; DeWitt and Hawthorn, 1981).

DIRECT is a collection of query processing elements and memory modules (Fig. 4) interconnected through a cross-bar switch network. The network provides access paths between any pair of query processor and memory module. Memory modules are shared disk caches. Allocation and deallocation of memory modules to the database pages is similar to the concept of paging scheme in the conventional systems. The backend controller is responsible for determining the number of query processors and memory modules that should be allocated to a query. This is done based on the complexity of the query, size of the database(s) and the system's contention. Simultaneous execution of different queries on a database enforces a locking protocol which guarantees the data consistency. This has been achieved by enforcing the locking scheme at the relation level.

The design offers better load balancing and allows the processors to be shared among the storage modules. Furthermore, expansion is easy and modular. However, this design suffers from the data transmission problem. As a result, a lot of data is transferred from one module to another. This could be reduced through the concept of presearching and selection of the related tuples before pages are allocated to a memory module. In addition, due to the nature of the organization (e.g., MIMD) a more elaborate security mechanism is needed to enforce the security and privacy access control. Finally, the adaptability and suitability of the query processing elements according to the constraints imposed by the current technology need to be studied in depth.

3.2.4 DIALOG—*Distributed Associative LOGic Database Machine*

DIALOG (Wah and Yao, 1980) is a collection of clusters of identical modules organized in a hierarchy. Each cluster is under the control of a controller, which provides communication paths between its nodes and other clusters on the one hand, and the frontend machine on the other hand (Fig. 5). The processing units (e.g., data modules) are designed to store a part of a database and perform the relational operations on them. Data modules are based on the concept of preprocessing. In other words, the data file is searched close to secondary storage and the valid tuples are sent to the system.

DIALOG is designed for handling a large on-line database system. Organization of the system is based on heterogeneous storage devices which allows the system to upgrade itself according to the future advances in secondary storage technology. In addition, similarity of the overall structure

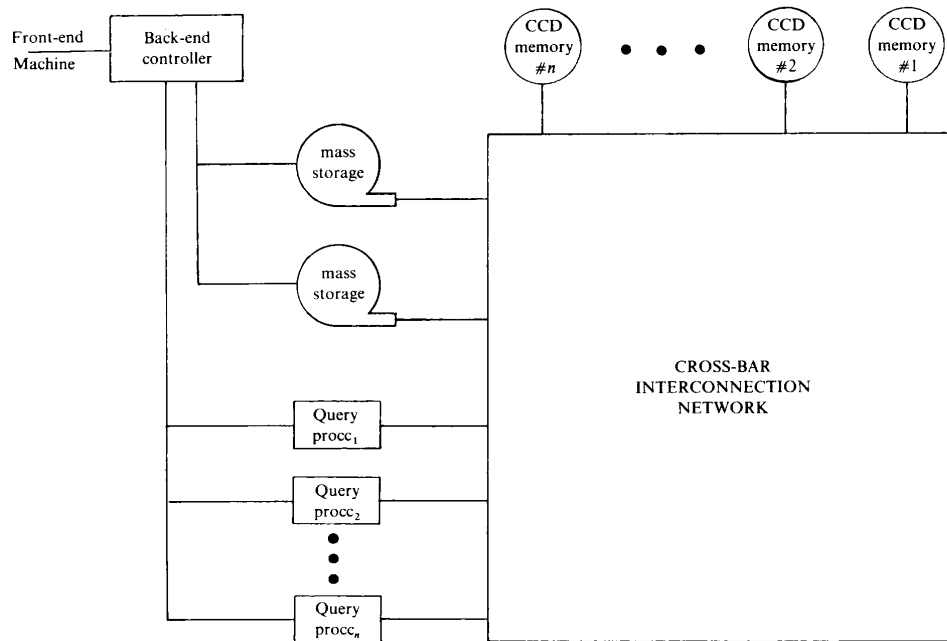


FIG. 4. DIRECT organization.

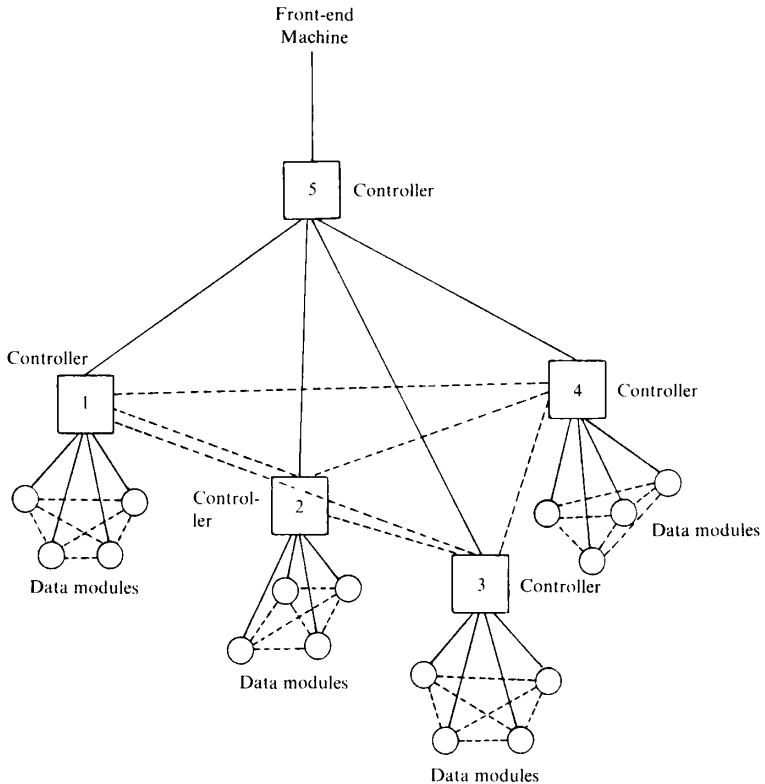


FIG. 5. DIALOG organization.

of the clusters and the uniformity of the sequence of operations within the clusters could simplify the overall design phase and fabrication. However, the simplicity of each data module and its adaptability for VLSI technology should be studied in more detail. Moreover, the computational capability of the system for handling a complete set of database operations and aggregate functions in a multi-user distributed environment should be enhanced. Otherwise, the bottleneck at the controller level will degrade the overall performance of the system. Finally, one has to study the overall performance of the system extensively.

3.3 High VLSI-Compatible Database Machines

The systems in this class are basically designed based on the constraints imposed by technology. In general, they are highly parallel with regular and simple architectures.

3.3.1 Systolic Organization

The systolic model proposed by Kung and Lehman (1980) is a collection of basic cells replicated in a two-dimensional space. Each cell has the organization as depicted in Fig. 6; a_{in} , b_{in} , and t_{in} represent the inputs and a_{out} , b_{out} , and t_{out} represent the outputs from a cell. A cell performs the following sequence of operations:

$$t_{out} \leftarrow t_{in} \wedge (a_{in} \theta b_{in})$$

$$a_{out} \leftarrow a_{in}$$

$$b_{out} \leftarrow b_{in},$$

where $\theta \in \{=, \neq, <, >, \leq, \geq\}$.

Tuples from the source and target relations move along the a and b paths and the result of the comparison within each cell is accumulated along the t path. As tuples are pipelined to the system, each pair of the tuples (i, j) in the

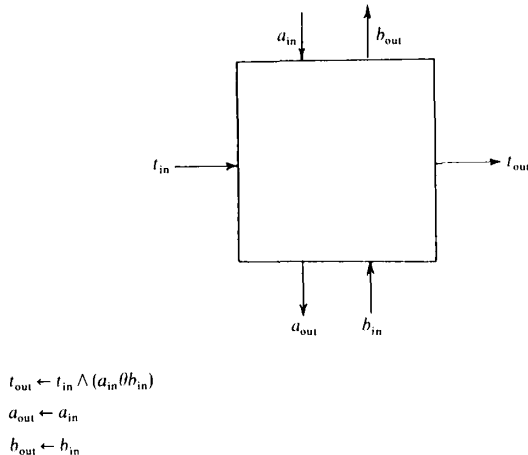


FIG. 6. A systolic cell.

source and target relations meet each other in some row. The final result of the comparisons within the rows are accumulated in a binary matrix T , where

$$T_{ij}^{final} = \begin{cases} \text{True} & \text{if } T_{ij}^{initial} = \text{true and } \forall_k a_{ik} \theta b_{jk} \\ \text{False} & \text{otherwise.} \end{cases}$$

Then, scanning the bit patterns in T determines the tuples that participate in the operation. Figure 7 shows the sequence of the operations and the formation of T for "join." As can be seen, the operations within cells should be synchronized. In addition, for proper execution of the basic operations, data should be delivered in a special pattern. The potential of this simple organization for handling database operations and file processing operations has been discussed by Lehman (1981).

The simplicity of the basic cell and the overall regularity of the architecture has provided a suitable ground for VLSI implementation of this architecture. On the negative part, each relation should be scanned twice. First they should be pipelined to the array to determine interrelationships between each pair of tuples, and then they should be scanned again to extract relevant tuples. The investigation of the resultant matrix (e.g., T) is another issue that should be studied in detail. According to the flow of the operations and data, the array should be large enough to accommodate the relations. This means that the feasibility of this design for handling large databases should be investigated in depth. Finally, as can be seen in Fig. 7, the staggered processing of data means that half of the cells will be idle during a clock period. This inefficiency, as discussed by Kung and Lehman (1980), can be removed by preallocation of a relation (say source) in the array and crossing the target tuples along the b path (or a path).

3.3.2 Tree Machines

The tree machine proposed by Bentley and Kung (1979) is a collection of three basic cells, namely the $\circ$ -node, $\square$ -node and $\triangle$ -node, organized as two-mirror-image binary trees sharing the leaf nodes (Fig. 8). The $\circ$ -nodes are used to broadcast the data and instructions to the leaf nodes. Each $\circ$ -node has a limited storage capacity enhanced by a processor to perform increment/decrement and simple selection operations. The $\square$ -nodes store records and perform the various broadcast operations in response to a user query. The $\triangle$ -nodes are used to combine the results of the operations in the $\square$ -nodes to compute the final result.

Song (1980) has shown that such a model can be used to perform relational operations. By assuming that duplicate tuples are not allowed within a relation, he simplified the original model proposed by Bentley and Kung

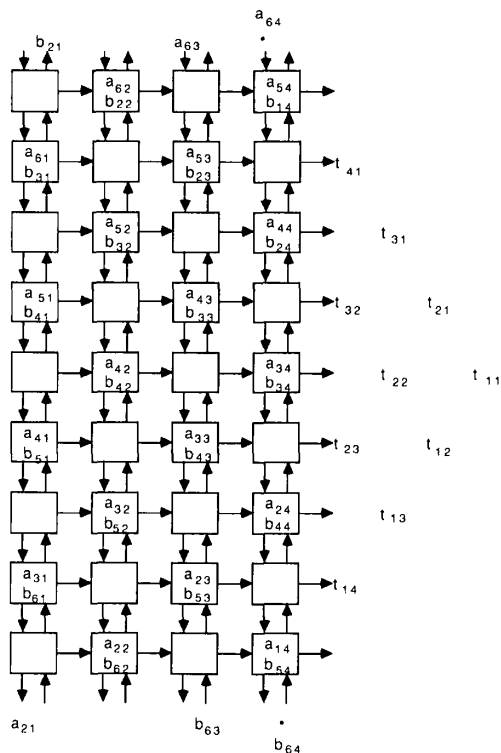


FIG. 7. Flow of data in systolic join.

(1979). The system performs the operations based on the principle of pipelining. Hence, it provides a smooth flow of data with a high throughput. However, due to the existing bottleneck at the root of the Δ -nodes, the pipeline might be forced to be turned off.

The regularity of the model and the simplicity of the connections among the cells, combined with the efficient chip layout format of a binary tree structure, provide an attractive model for VLSI implementation of the system. By unmirroring the binary trees (Fig. 9), the system can be constructed utilizing two types of chips, namely internal and leaf chips. An internal chip contains a combination of the $\circ$ -nodes and Δ -nodes while the leaf chip contains the $\square$ -nodes.

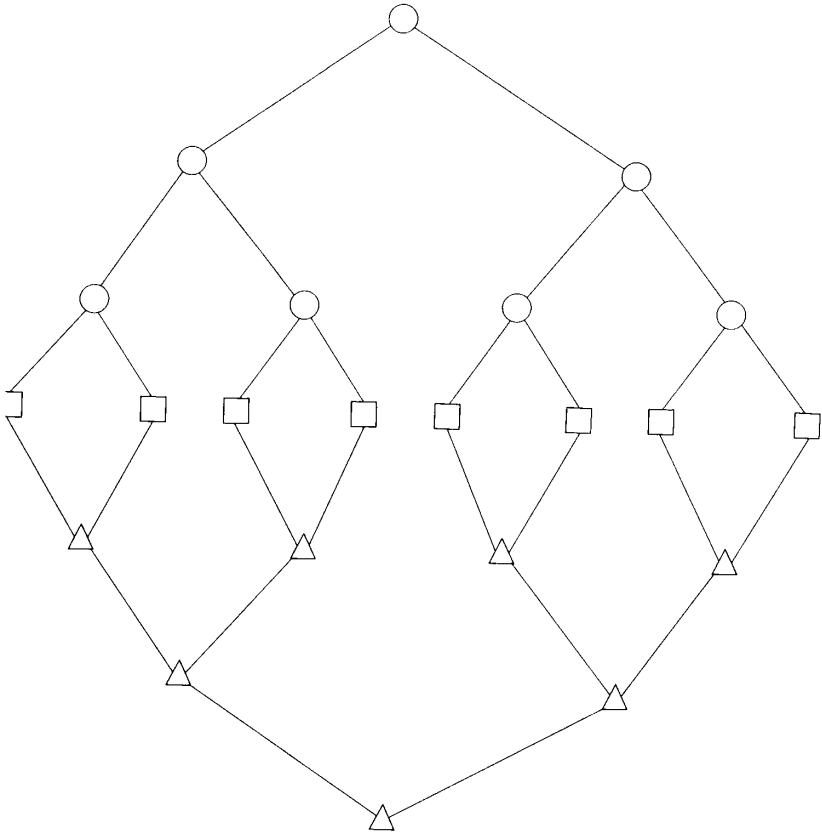


FIG. 8. Tree organization.

Data transmission among the cells is the major bottleneck of the model. As a result, in many instances the pipeline cannot have its smooth and uniform data flow. This will enforce some complexities at the control level. I/O pin limitation is another drawback which has to be addressed extensively. It is possible to provide a direct parallel I/O facility at the leaf chip. However, problems such as pin limitation have to be discussed. For some relational operations such as join (Song, 1980), resource utilization of the system is low. Finally, capability of this model for handling large databases is a serious problem which should be investigated extensively.

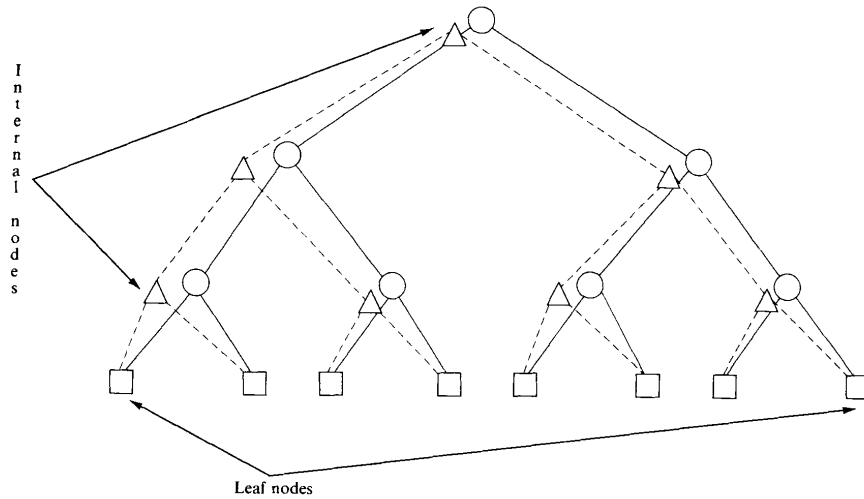


FIG. 9. Unmirror tree of Fig. 8.

It should be noted that the tree organization proposed by Bonuccelli *et al* (1985) is based on the same philosophy. However, nodes have uniform organization, each capable of performing the bookkeeping, communication, and database operations. In addition, the representation of tuples and the sequence of the operations are totally different than the one discussed by Song (1980).

3.3.3 ASLM—An Associative Search Language Machine

ASLM (Associative Search Language Machine) is a backend database machine (Hurson, 1981, 1983). The system is composed of a collection of general-purpose frontend machines supported by ASLM (Fig. 10). Security validation of the user and the user's query, translation of the user's query into ASLM primitives, and transmission of the final results to the user are the major functions of the frontend systems.

ASLM is composed of four modules: *i*) controller, *ii*) secondary storage interface, *iii*) an array of preprocessors, and *iv*) a database processor.

Controller. The controller *i*) stores the ASL microinstructions generated by the ASL compiler, *ii*) decodes the ASL microinstructions, *iii*) propagates the control sequences to the appropriate modules in the backend, and *iv*) manages the sharing and distribution of the resources among different concurrent users' requests. The simplicity of the controller is primarily due to the use of associative hardware for the execution of the ASL primitives.

Secondary Storage Interface. The secondary storage interface is a collection of random access memory modules, augmented by some hardware facilities. This module is an interface between secondary storage and ASLM. It accesses blocks of data from secondary storage and distributes them among the preprocessors.

Preprocessors. The preprocessors act as a filter which screens the data, selecting the valid data. The valid data is then placed in an associative module in the database processor. In addition, the preprocessors perform a projection over the relevant attributes.

Database Processor. The database processor is composed of a set of associative modules enhanced by some hardware capabilities for direct implementation of the relational operators. The result of the operations of the array of preprocessors on a relation r with relation scheme R is a relation r_s with relation scheme R_s , which is stored in an associative memory.

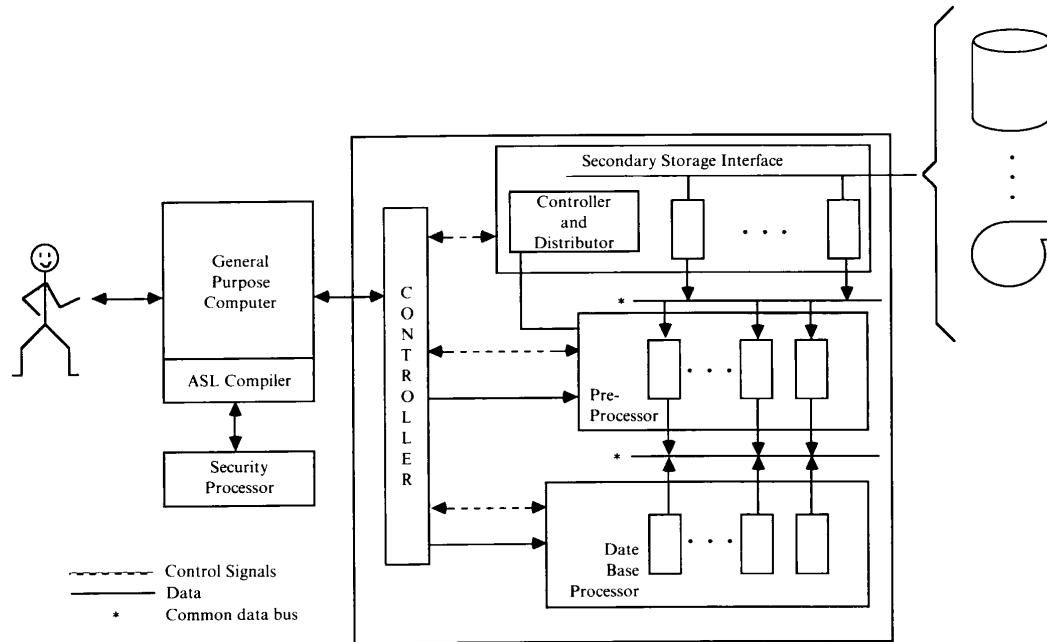


FIG. 10. ASLM architecture.

This unit is a set of identical and independent associative memories of size $w * d$ (where w is the width and d is the depth) augmented by some hardware circuits. The associative modules can be linked together to make a memory size $(K * w) * d$ ($1 \leq K \leq n$, where n is the number of associative modules) capable of holding d tuples of size $K * w$ bits, or they could be linked to form a memory of size $w * (K * d)$ capable of holding $K * d$ tuples of size w bits. The independence of the associative modules enhances the modularity and as such the fault tolerance of the system. An enhanced join module capable of performing high-level operations on null values has been incorporated in the design of the database processor (Miller and Hurson, 1986).

VLSI time/space complexity of the basic cells of an associative memory for handling basic set and relational operations has been discussed by Hurson (1986). Based on such a calculation, the search time would be $13\Delta t + 3(n - 1)\Delta t$ where Δt and n stand for average delay time of an inverter and the word length, respectively. In addition, the geometry area of a cell has been estimated at $40\lambda * 20\lambda$.

The Non-Von DBM project, under investigation since the early 1980s (Shaw, 1980), is composed of two components: a primary processing subsystem (PPS) and a secondary processing subsystem (SPS). The SPS is the major repository of the data and consists of conventional movable-head disks with processing logic per head. The logic is capable of examining arbitrary attributes in each tuple and performing a hash function on them. PPS processing includes associative retrieval, arithmetic comparisons, logical manipulation, and I/O functions.

3.3.4 Parallel Pipelined Query Processor

The pipelined database machine (Gajski *et al.*, 1984) is a collection of four modules, namely single-relation query pipe, aggregation pipe, join pipe, and sorting pipe. These pipes share a group of FIFO memories through a cross-bar switch network (Fig. 11). Like ASLM (Hurson, 1981), the system is based on the principle of preprocessing.

A single-relation query pipe is a collection of 32 chips each composed of two parts: a where-evaluation part and a select-evaluation part. The where-evaluation part is used to check the validity of tuples according to the search arguments defined in the user query. The select-evaluation part projects tuples on the attributes defined in the output set. In addition, it can perform some simple arithmetic operations on the attributes specified in the output set. These two parts perform their operations in parallel on the designated tuple.

An aggregation pipe is composed of several units (e.g., squeezer, aggregation unit, duplicate checker, and arithmetic/comparator unit) for handling operations such as select unique, Count, Sum, Max, Min and Avg.

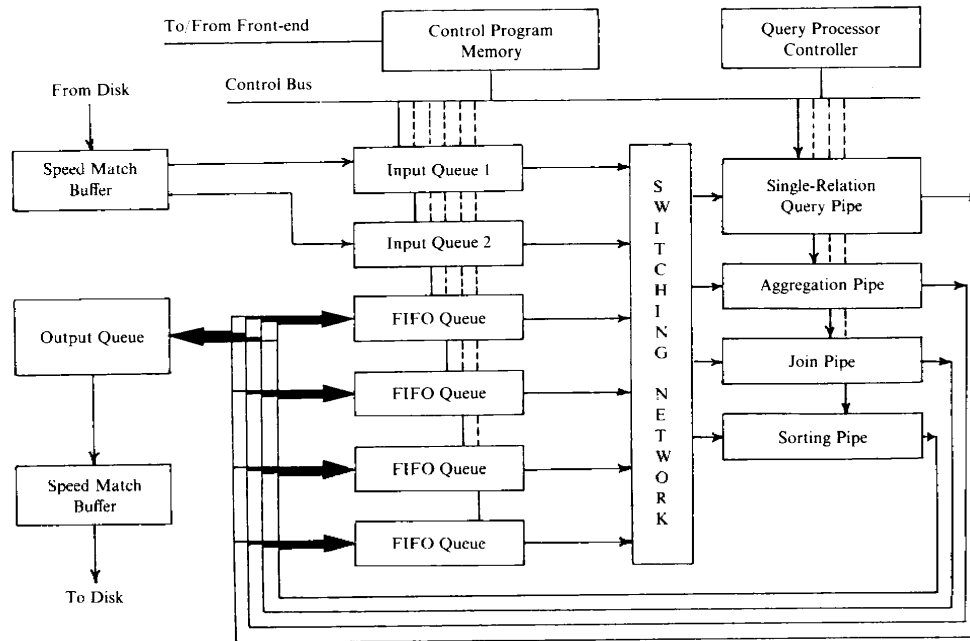


FIG. 11. Query processor organization.

The join pipe, consisting of 32 identical join chips, performs the operation as a sequence of two steps: first, similar to ASLM, source and target tuples are selected and projected, and then the join is performed on the explicitly generated reduced relations.

The sorting pipe is organized as eight chips of 16×16 sorting modules. Figure 12 depicts the overall organization of a 4×4 sorting module. It is a

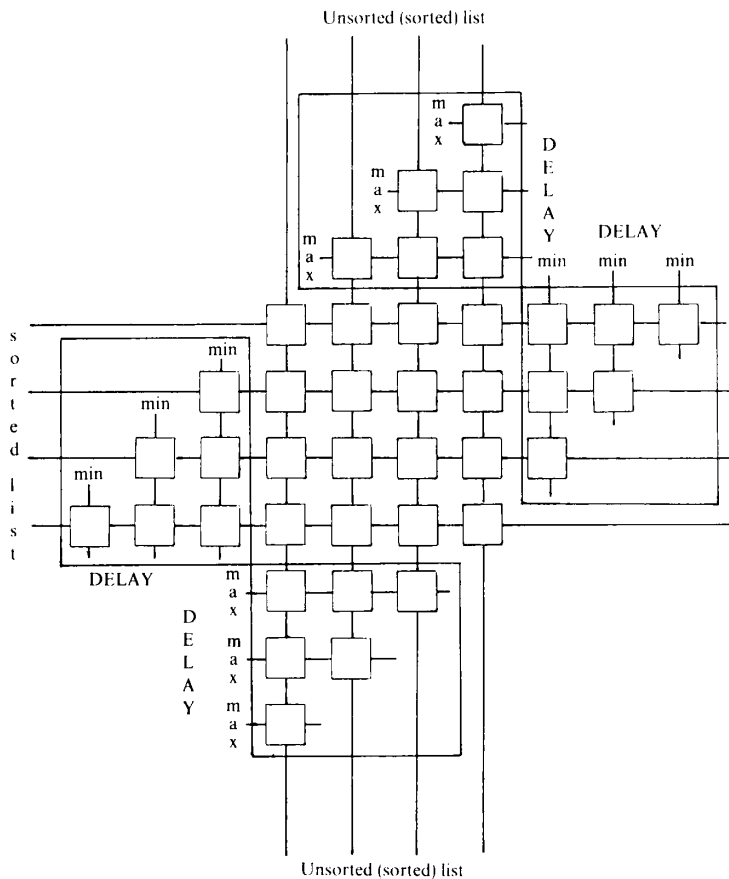


FIG. 12. A 4×4 sorting pipe.

matrix of comparator/switches. The delay elements as depicted in Fig. 12 are used to synchronize the arrival and departure of tuples to and from the 4×4 matrix at the center.

In general, the system is highly adaptable for VLSI implementation and is topologically similar to CRAY. As discussed by Gajski *et al.* (1984), some of the chips are under implementation. However, due to the fact that the modules are not general, data transmission within the system is high, which could degrade performance.

3.3.5 GRACE

The GRACE relational DBM currently being developed at the University of Tokyo is aimed at achieving high performance by adapting processing algorithms based on hash and sort operations (Fushimi *et al.*, 1986). The general philosophy behind the query processing in GRACE is based on clustering the databases via hashing, a uniform distribution of clusters among several temporary storage units, and parallel execution of processing units on distributed clusters. For example, in a join operation the hash clustering feature of GRACE allows the source and target relations to be partitioned into disjoint sets of smaller relations (subrelation), where just source and target subrelations stored in a memory module need to be joined together. Figure 13 depicts the overall architecture of GRACE. The system is composed of two basic rings (e.g., processing ring and staging ring) and four fundamental modules (e.g., processing module (PM), memory module (MM), disk module (DM), and control module (CM)). The processing module performs relational operations on stored data in the memory module in a pipeline fashion. The disk module is a small processor capable of filtering the data according to the user query and distributing the selected tuples among the memory modules. Finally, the control module monitors the flow of data and operations within different modules in the system.

3.3.6 Delta

The Delta database machine (Hiroshi *et al.*, 1984, Kakuta *et al.*, 1985) is currently being studied at ICOT (Institute for New Generation Computer Technology). It is planned to be connected to a set of inference machines as part of Japan's fifth-generation project. The architecture of Delta is composed of five components (Fig. 14):

- (a) *Interface Processor (IP)* is an interface between Delta and inference machines. This interaction is through a local area network (LAN).
- (b) *Control Processor (CP)* provides database management functions to manage the resources and control the flow of data and operations.

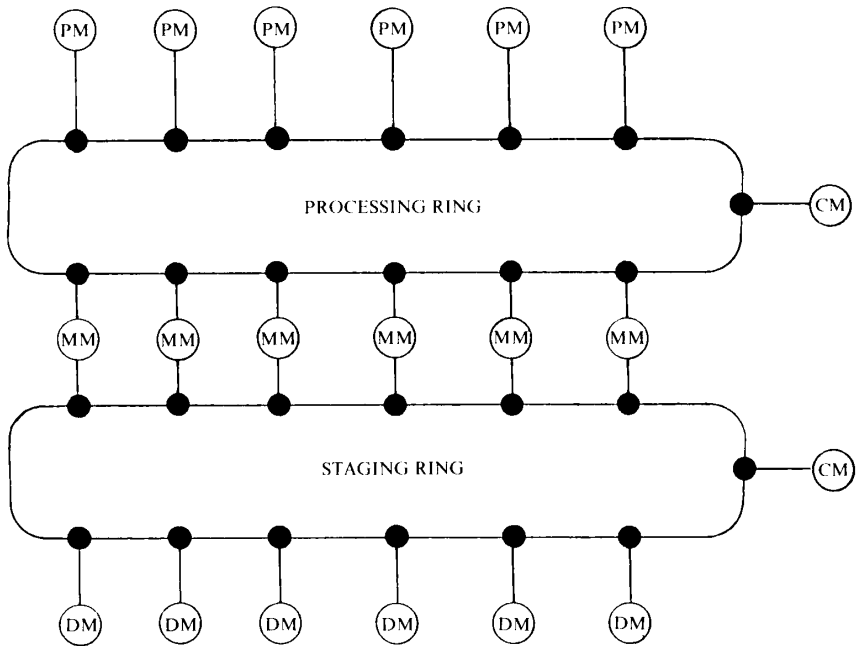


FIG. 13. The overall organization of GRACE. PM = Processing Module; MM = Memory Module; DM = Disk Module; CM = Control Module

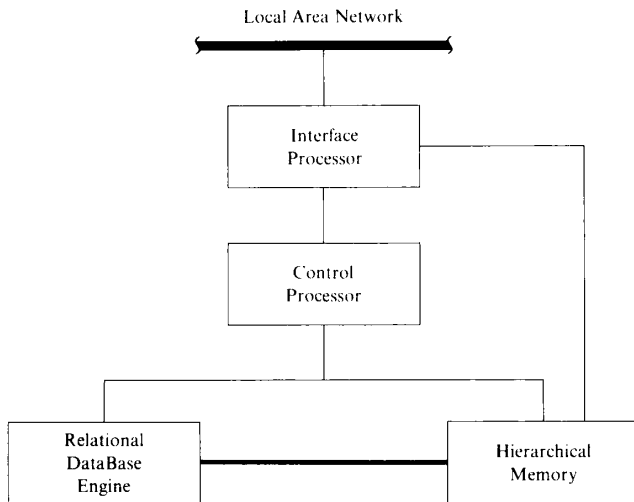


FIG. 14. Functional architecture of Delta.

- (c) *Relational DataBase Engine* (RDBE) is the key component for processing relational operations in Delta, based on sorting algorithms.
- (d) *Maintenance Processor* (MP), provides functions to enhance Delta's reliability and serviceability.
- (e) *Hierarchical Memory* (HM) provides functions such as sorting, accessing, clustering and maintaining relations. This component is composed of a general-purpose process as controller and a hierarchy of memory units consisting of a semiconductor memory and large-capacity moving-head disks.

The query processing in Delta is as follows: The Interface Processor receives the translated user's commands (e.g., Delta Command) from a host connected to the LAN, and sends them to the Central Processor. The Central Processor translates these commands into a sequence of internal subcommands which are then issued to the Relational DataBase Engine and Hierarchical Memory. After execution of the commands, the IP transfers the result to the host. The Relational DataBase Engine as the key component for handling relational operations is composed of a general-purpose CPU, a sorting pipe of 12 stages, and a merger. A combination of sorting pipe and merger is used to perform a sort-merge join operation.

As mentioned before, Delta functions as a database server to Personal Sequential Inference (PSI) machines in a local area network. The PSIs use a logic programming language which must interface with Delta. To avoid the possible bottleneck, Delta does not offer a wide range of database functions.

3.3.7 *MIRDM—Michigan Relational Database Machine*

The MIRDM (Michigan Relational Database Machine) is a backend relational machine suitable for supporting concurrent on-line very large databases (Qadah and Irani, 1985). To reduce the data transportation the system maintains a series of index tables to allow data granularity at the page level. However, this reduction is achieved at the expense of more complex operations and extra space overhead.

MIRDM architecture can be classified as a loosely coupled MSIMD (Multiple Single Instruction-stream, Multiple Data-stream) architecture. As depicted in Fig. 15, it is composed of four basic components:

- (a) *Master Backend Controller* (MBC). Like any other backend system, MBC is an interface between frontend machine and the backend database system. It translates the user queries, schedules and monitors the query execution, manages and controls the different components of MIRDM and provides security checking and integrity mechanisms.

- (b) *Mass Storage Subsystem (MSS)*. MSS is a two-level memory hierarchy consisting of a Mass Memory (MM) organized as a set of moving-head disks, and parallel buffer (PB) organized as a set of blocks.
- (c) *Processing Cluster Subsystem (PCS)*. This is the major component to perform relational algebra operations on the data. Each processor in a cluster is composed of three modules: local memory, controller, and a processing unit.
- (d) *Interconnection Network Subsystem (INS)*. This unit is designed to allow simultaneous and dynamic interactions among the PCSs and PBs on one hand and PBs and MM on the other hand.

As reported by Qadah and Irani (1985), a prototype of MIRDMM consisting of a processing cluster with eight modules is under development.

3.4 Related Efforts

Several database machines have been proposed that fall outside of the focus of this paper, but are significant enough to deserve mention. The concept of

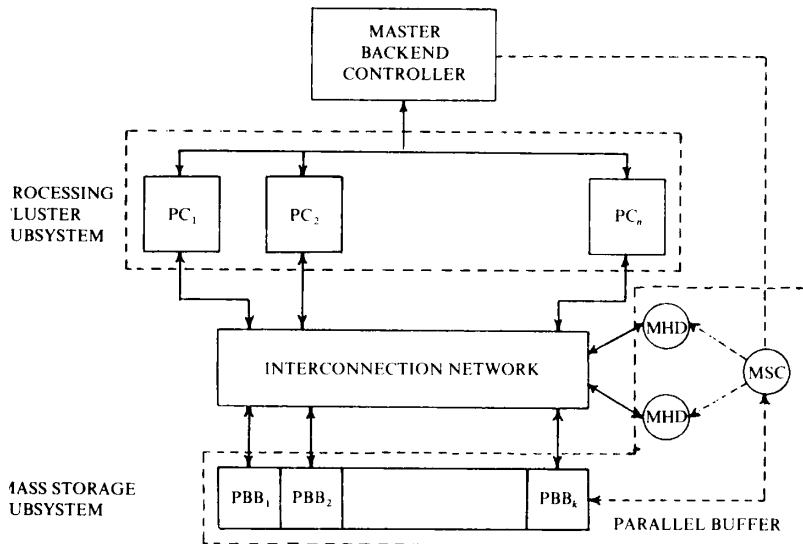


FIG. 15. The organization of the Michigan Database Machine.

data flow has been incorporated into several recent DBM proposals. GAMMA (DeWitt *et al.*, 1987), AGM (Bic and Hartmann, 1985), and FLASH (Boral and DeWitt, 1980) are some examples. In addition, a great deal of effort has recently been directed towards developing massive memory DBMs. New architectures in this area include MARS (Eich, 1987), MM-DBS (Lehman, 1986), and MMM (Garcia-Molina *et al.*, 1984).

4. Conclusion and Future Directions

The purpose of this paper was twofold: *i*) to study the architectural aspects of DBMs, and *ii*) to investigate the impact of technology on DBM design. We also intended to put technology-oriented DBMs into a perspective. To accomplish this we have proposed a new classification scheme that emphasizes the adaptability of a design to current technology. As a result, several classification schemes of database machines have been overviewed and compared.

The new classification scheme provides an extension to Qadah's database space (Qadah, 1985). The new coordinate studies the database machines according to their compatibility for current advances in technology. This coordinate classifies the DBMs as low-, semi-, and high-compatible architectures. Several machines which were discussed are summarized according to our database space in Table II. One of our goals in doing this survey was to develop an understanding of what architectural aspects of the existing designs make them "practical" with respect to current technology. It is our feeling that the new classification scheme aids the reader in making similar judgments.

Obviously, we did not intend to provide a critique of database machines. However, one should briefly address the lack of widespread use of DBMs after two decades of research and development. One major problem plaguing the DBMs is the I/O bottleneck since little attention has been given to exploiting mechanisms for increasing I/O bandwidth. Boral and DeWitt (1985) have devised two solutions for this purpose. The basic idea in the first solution is due to the use of a customized disk controller that permits simultaneous data transfer between the controller and conventional disk drives. The second solution is based on the utilization of an I/O cache.

Most database machine designers have failed to address issues central to the future success of their designs. In particular, issues such as how updates are performed have been ignored in many cases. In addition, designers have generally failed to address recent developments in database theory (Hurson and Miller, 1987; Miller and Hurson, 1986). The importance of such developments in software systems means that designers of practical DBMs cannot continue to ignore such issues.

TABLE II
CLASSIFICATION OF DATABASE MACHINES

Machine	Technology compatibility	Indexing level	Processing location	Parallelism
PPQP* (Gajski <i>et al.</i> 1984)	high	relation	off disk	MOMD
ASLM (Hurson 1981, 1983)	high	relation	off disk	SOMD
Systolic (Kung and Lehman, 1980; Lehman, 1981)	high	relation	off disk	SOMD
RAP3 (Ofazer and Ozkarahan, 1980)	semi	relation	off disk	MOMD
DIALOG (Wah and Yao, 1980)	semi	relation	hybrid	MOMD
Tree Machine (Bentley and Kung, 1979; Song 1980; Bonuccelli <i>et al.</i> , 1985)	high	relation	off disk	SOMD
Delta (Hiroshi <i>et al.</i> , 1984; Kakuta <i>et al.</i> , 1985)	high	relation	off disk	SOMD
GRACE (Fushimi <i>et al.</i> , 1986)	high	relation	hybrid	MOMD
MIRDM (Qadah, 1985)	high	page	off disk	MOMD
DBC (Banerjee <i>et al.</i> , 1978, 1979)	semi	page	hybrid	SOMD
DIRECT (Bitton <i>et al.</i> , 1983; Boral and DeWitt, 1981; Boral <i>et al.</i> , 1982; DeWitt, 1979; DeWitt and Hawthorn, 1981)	semi	relation	off disk	MOMD
RELACS (Oliver, 1979)	semi	relation	off disk	MOMD
RAP2 (Schuster 1979)	semi	relation	off disk	MOMD
CASSM (Su, 1979)	low	database	on disk	SOMD
Search Processor (Leilich <i>et al.</i> , 1977)	low	database	on disk	MOSD
RARES (Lin <i>et al.</i> , 1976)	low	database	on disk	SOMD
RAP1 (Ozkarahan <i>et al.</i> , 1975)	semi	database	on disk	SOMD
Moulder (Moulder, 1973)	low	database	off disk	SOMD

(continues)

TABLE II (Continued)

Machine	Technology compatibility	Indexing level	Processing location	Parallelism
Linde (Linde <i>et al.</i> , 1973)	low	database	off disk	SOMD
Defiore (Defiore and Berra, 1973)	low	database	off disk	SOMD
CASSS (Healy <i>et al.</i> , 1972)	low	database	off disk	SOMD
RAPID (Parhami, 1972)	low	database	off disk	SOMD
Logic Per Track (Parker, 1971)	low	database	on disk	SOMD
Gaines (Gaines 1965)	low	database	off disk	SOMD
Lee (Lee and Paull, 1963)	low	database	off disk	SOMD

* PPQP: Parallel Pipelined Query Processing

As proposed by Boral and Redfield (1985), the future database machine designers should concentrate on two important issues: *i*) the investigation of the effect and benefit of a specialized database operating system on their DBM designs, and *ii*) the optimization of the system throughput rather than improving the response time of a single request.

REFERENCES

- Ambardar, V. (1985). Data base machines. *Proc. 18th Annu. Hawaii Int. Conf. System Sciences* **1**, pp. 352–372.
- Arora, S. K., and Dumpala, S. R. (1981). WCRC: An ANSI SPARC machine architecture for data base management. *8th Annu. Symp. Computer Architecture*, pp. 373–387.
- Babb, E. (1979). Implementing a relational data base by means of specialized hardware. *ACM Trans. Data Base Systems* **4**, 1–29.
- Banerjee, J. Hsiao, D. K., and Baum R. J. (1978). Concepts and capabilities of a data base computer. *ACM Trans. Data Base Systems* **3**, 347–384.
- Banerjee, J., Hsiao, D. K., and Kannan, (1979). DBC—A database computer for very large data bases. *IEEE Trans. Computer* **28**, 414–429.
- Bentley, J. L., and Kung H. T. (1979). A tree machine for searching problems. *IEEE Int. Conf. Parallel Processing*, pp. 257–266.
- Berra, B. P. (1974). Some problems in associative processor applications to database management. *Nat. Computer Conf.*, pp. 1–5.
- Berra, B. P. (1977). Database machines. *ACM SIGIR Forum* **13**, 4–23.
- Berra, B., and Oliver, E. (1979). The role of associative array in data base machine architecture. *IEEE Computer* **12**, 53–61.
- Bic, L., and Hartmann, R. L. (1985). Hither hundreds of processors in a database machine. *Database Machine 4th Int. Workshop*, pp. 153–168.

- Bird, R. M., Tu, J. C., and Worthy, R. M. (1977). Associative/parallel processors for searching very large textual data bases. *Proc. 3rd Non-Numeric Workshop*, pp. 1–9.
- Bitton, D. et al. (1983). Parallel algorithms for the execution of relational data base operations. *ACM Trans. Data Base Systems* **8**, 324–353.
- Block E., and Galage, D. (1978). Component progress; its effect on high speed computer architecture and machine organization. *Computer* **11**, 64–76.
- Bonuccelli, M. A., Lodi, E., Luccio F., Maestrini, P., and Pagli, L. (1985). VLSI algorithms and architecture for relational operations. *Estratto da Calcolo* **XXII**, 63–90.
- Boral, H. L., and DeWitt, D. J. (1980). Design consideration for dataflow database machines. *Proc. ACM SIGMOD Int. Conf. Management of Data*, pp. 95–104.
- Boral, H. L., and DeWitt, D. J. (1981). Processor allocation strategies for multiprocessor database machines. *ACM Trans. Database Systems* **6**, 227–254.
- Boral, H. L., and DeWitt, D. J. (1985). Database machines: an idea whose time has passed? a critique of the future of database machines. "Database Machines", pp. 166–187, Springer Verlag.
- Boral, H. L., and Redfield, S. (1985). Database machine morphology. *Proc. Very Large Databases*, pp. 59–71.
- Boral, H. L., DeWitt, D. J., Friedland, D., Jarrel, N. F., and Wilkinson, K. W. (1982). Implementation of the database machine direct. *IEEE Trans. Software Engineering* **SE-8**, 533–543.
- Boyce, R. F., Chamberlin, D. D., King, W. F., III, and Hammer, M. M. (1975). Specifying queries as relational expression: the SQUARE data sublanguage. *Comm. ACM* **18**, 621–628.
- Bray, H. O., and Freeman, H. A. (1979). "Database Computers." Lexington Books, Massachusetts.
- Bray, H. O., and Thurber, K. J. (1979). What's happening with data base processors? *Datamation*, 146–156.
- Cardenas, A. F. (1975). Analysis and performance of inverted database structures. *Comm. ACM* **18**, 253–273.
- Chamberlin, D. D. (1976). Relational data base management systems. *Computing Surveys* **8**, 43–59.
- Chamberlin, D. D., Gray, J. N., and Traiger, I. L. (1975). Views, authorization and locking in a relational data base system. *Nat. Computer Conf.* **44**, pp. 425–430.
- Champine, G. A. (1978). Four approaches to a data base computer. *Datamation*, 101–106.
- Champine, G. A. (1979). Current trends in data base systems. *Computer* **12**, 27–41.
- Davis, E. W. (1974). STARAN parallel processor system software. *AFIPS Nat. Computer Conf.* **43**, pp. 17–22.
- Davis, E. W. (1983). Application of the massively parallel processor to database management systems. *Proc. AFIPS Nat. Computer Conf.*, pp. 299–307.
- Defiore, C. R., and Berra, P. B. (1973). A data management system utilizing an associative memory. *Nat. Computer Conf.* **42**, pp. 181–185.
- Demurjian, S. A., Hsiao D. K., and Strawser P. R. (1986). Design analysis and performance evaluation methodologies for database computers. In "Advances in Computers" **25** (M. C. Yovits, ed.), pp. 101–214.
- DeWitt, D. J. (1979). DIRECT—A multiprocessor organization for supporting relational database management systems. *IEEE Trans. Computers* **C-28**, 395–406.
- DeWitt, D. J., and Hawthorn, P. B. (1981). A performance evaluation of database machine architecture. *Proc. 7th Int. Conf. Very Large Data Bases*, pp. 199–213.
- DeWitt, D. J., et al. (1987). A Single User Evaluation of a Gamma Database Machine. In "Database Machines and Knowledge Base Machines" (M. Kitsuregawa and H. Tanaka, eds.), pp. 370–386. Kluwer, Academic Publishers.
- Eich, M. H. (1987). MARS: the design of a main memory database machine. In "Database

- Machines and Knowledge Based Machines", (M. Kitsuregawa and H. Tanaka, eds.), pp. 325–338. Kluwer, Academic Publishers.
- Feng, T. Y. (1976). Some considerations in multiprocessor architecture. In "Multiprocessor Systems" (C. H. White, ed.), pp. 277–286. Infotech state of the art report, England.
- Finnila, C. A., and Love, H. H., Jr. (1977). The associative linear array processor. *IEEE Trans. Electron Computers* **16**, 112–142.
- Fletcher, J. C. (1984). Report of the study of eliminating the threat posed by nuclear ballistic missiles. MDA 90384 Coo31, Task T-3-191.
- Flynn, M. J. (1966). Very high speed computing systems. *Proc. IEEE* **54**, 1901–1909.
- Fushimi, S., Kitsuregawa, M., and Tanaka, H. (1986). An overview of the system software of a parallel relational database machine GRACE. *Proc. Int. Conf. Very Large Databases*, pp. 209–219.
- Gaines, R. S., and Lee, C. Y. (1965). An improved cell memory. *IEEE Trans. Electron Computers* **C-14**, 72–75.
- Gajski, D., Kim, W., and Fushimi, S. (1984). A parallel pipelined relational query processor: an architectural overview. *Proc. 11th Annu. Symp. Computer Architecture*, pp. 134–141.
- Garcia-Molina, H., Lipton, R. J., and Valdes, J. (1984). A massive memory machine. *IEEE Trans. Computers* **C-33**, 391–399.
- Gonzalez-Rubio, R., and Rohmer, J. (1984). The schuss filter: a processor for non-numerical data processing. *Proc. 11th Annu. Symp. Computer Architecture*, pp. 393–395.
- Händler, W. (1982). Innovative computer architecture to increase parallelism but not complexity. In "Parallel Processing Systems" (D. J. Evans, ed.), pp. 1–41. Cambridge University Press, London.
- Haskin, R. (1978). Hardware for searching very large text databases. Ph.D. Dissertation, University of Illinois, Urbana, Illinois.
- Haskin, R., and Hollaar, L. A. (1983). Operational characteristics of a hardware-based pattern matcher. *ACM Trans. Database Systems* **8**, 15–40.
- Hawthorn, P. (1980). Panel on database machines. *Proc. 6th Int. Conf. Very Large Data Bases*, pp. 393–395.
- Healy, L. D., Lipovski, G. J., and Doty, K. L. (1972). The architecture of a context addressed segment-sequential storage. *Fall Joint Computer Conf.* **41**, 691–701.
- Held, G. D., Stonebraker, M. R., and Wong, E. (1975). INGRES—A relational data base system. *Nat. Computer Conf.* **44**, 409–416.
- Hikita, S., Kawakami, S., and Haniuda, H. (1985). Database machine FRENDD. *Database Machines 4th Int. Workshop*, pp. 190–207.
- Hiroshi, S., *et al.* (1984). Design and implementation of the relational data base engine. *Proc. Int. Conf. Fifth Generation Computer Systems*, pp. 419–426.
- Hong, Y. C. (1985). Efficient computing of relational algebraic primitives in a data base machine architecture. *IEEE Trans. Computers* **C-34**, 588–595.
- Hsiao, D. K. (1980). Data base computers, In "Advances in Computers" **19** (M. C. Yovits, ed.), pp. 1–64. Academic Press, New York.
- Hsiao, D. K. (1983). "Advanced Database Machine Architecture." Prentice Hall.
- Hsiao, D. K. (1983). Cost-effective ways of improving database computer performance. *Nat. Computer Conf.* **52**, pp. 292–298.
- Hsiao, D. K., Kannan, K., and Kerr, D. S. (1977). Structure memory designs for a database computer. *AFIPS Proc. Nat. Computer Conf.*, pp. 343–350.
- Hurson, A. R. (1981). An associative backend machine for database management. *Proc. 6th Workshop on Computer Architecture for Non-Numeric Processor*.
- Hurson, A. R. (1983). A special purpose mini computer for relational database management. *ISMM J. Microcomputer Applications* **2**, 9–12.

- Hurson, A. R. (1984). A VLSI design for the parallel finite state automaton and its performance evaluation as a hardware scanner. *Int. J. Computer and Information Sciences* **13**, 491–508.
- Hurson, A. R. (1985). A mini processor for database security enforcement. *ISMM J. Micro-computer Applications* **6**, 16–19.
- Hurson, A. R. (1986). VLSI time/space complexities of an associative join module. *Proc. Int. Conf. Parallel Processing*, pp. 379–386.
- Hurson, A. R., and Miller, L. L. (1982). A non-numeric processor. *AICA Annu. Conf. Italian Computer Society*, pp. 845–852.
- Hurson, A. R., and Miller, L. L. (1983). Security issues in backend database machines. *Proc. 17th Annu. Conf. Information Science and Systems*, pp. 690–696.
- Hurson, A. R., and Miller, L. L. (1987). A database machine architecture for supporting incomplete information. *J. Computer Science and Systems Engineering* **2**, 107–116.
- Kakuta, T., Miyazaki, N., Shibayama, S., Yokota, H., and Murakami, K. (1985). The design and implementation of relational database machine delta. *Database Machine 4th Int. Workshop*, pp. 13–34.
- Kambayashi, Y. (1984). A Database Machine Based on the Data Distribution Approach. *Nat. Computer Conf.* **53**, pp. 613–625.
- Kambayashi, N., and Seo, K. (1982). SPIRIT-III: an advanced relational database machine introducing a novel data-staging architecture with tuple stream filters to preprocess relational algebra. *Nat. Computer Conf.* **51**, pp. 605–616.
- Kannan, K. (1978). The design of a mass memory for a database computer. *Proc. 5th Annu. Symp. Computer Architecture*, pp. 44–50.
- Kerr, D. S. (1979). Data base machines with large content addressable blocks and structural information processors. *Computer* **12**, 64–81.
- Kim, Won, (1979). Relational database systems. *Computing Surveys* **11**, 185–211.
- Klass, P. J. (1984). New microcircuits challenge silicon use. *Aviation Week of Space Technology*, 179–184.
- Kung, H. T., and Lehman, P. L. (1980). Systolic (VLSI) arrays for relational database operations. *ACM/SIGMOD Int. Conf. Management of Data*, pp. 105–116.
- Lamb, S. (1978). An ADD-in recognition memory for S-100 bus micro computers. *Computer Design* **17**, 162–168.
- Lamb, S., and Vanderslice, R. (1978). Recognition memory: low cost content addressable parallel processor for speech data manipulation. *Joint Meeting of the Acoustical Society of America and Acoustical Society of Japan*.
- Langdon, G. G., Jr. (1978). A note on associative processors for data management. *ACM Trans. Data Base Systems* **3**, 148–158.
- Langdon, G. G., Jr. (1979). Database machines: an introduction. *IEEE Trans. Computers* **C-28**, 381–383.
- Lea, R. M. (1976). "Associative Processing of Non-numerical Information." D. Reider Publishing Company, Dordrecht, Holland.
- Lee, C. Y., and Paull, M. C. (1963). A content addressable distributed logic memory with applications to information retrieval. *Proc. IEEE* **51**, 924–932.
- Lee, S. Y., and Chang, H. (1979). Associative search bubble devices for content addressable memory and array logic, *IEEE Trans. Computers* **C-28**, 627–636.
- Lehman, P. L. (1981). A systolic (VLSI) array for processing simple relational queries. In "VLSI Systems and Computations" (Kung et al., eds.), pp. 285–295. Computer Science Press.
- Lehman, T. J. (1986). Design and performance evaluation of a main memory relational database system. Ph.D. Dissertation, University of Wisconsin-Madison.
- Leilich, H. O., Steige, G., and Zeidler, H. C. (1977). A search processor for data base management systems. Technical University of Braunschweig, Braunschweig, Germany.

- Lin, C. S., Smith, D., and Smith, J. M. (1976). The design of a rotating associative memory for relational database applications. *ACM Trans. Database Systems* **1**, 53–65.
- Linde, R. R., Gates, R., and Peng, T. (1973). Associative processor applications to real-time data management. *Nat. Computer Conf.* **42**, pp. 187–195.
- Lipovski, S. J. (1978). Architectural features of CASSM: a context addressed segment sequential memory. *SIGARCH Newsletter* **6**, 31–38.
- Liuzzi, R. A., and Berra, P. B. (1982). A methodology for the development of special-purpose function architectures. *Nat. Computer Conf.* **51**, 125–134.
- Mahoney, W. C. (1981). Defense mapping agency (DMA) overview of mapping, charting, and geodesy (MC&G) applications of digital image pattern recognition. *SPIE* **281**, Techniques and Applications of Image Understanding, 11–23.
- Maller, V. A. (1979). The content addressable file store—CAFS. *ICL Tech. J.* **2**, 265–279.
- Maryanski, F. J. (1980). Backend database systems. *ACM Computing Surveys* **12**, 3–25.
- Miller, L. L. and Hurson, A. R. (1983). Performance evaluation of database machines based on the database relationships. *IEEE Workshop on Computer Architecture for Pattern Analysis and Image Database Management*, pp. 187–192.
- Miller, L. L., and Hurson, A. R. (1986). Maybe algebra operators in database machine architecture. *Proc. Fall Joint Computer Conf.*, pp. 1210–1218.
- Minsky, N. (1972). Rotating storage devices as partially associative memories. *Proc. AFIPS Fall Joint Computer Conf.*, pp. 587–595.
- Moulder, R. (1973). An implementation of a data management system on an associative processor. *Nat. Computer Conf.*, pp. 171–176.
- Mukherjee, A. (1986). "Introduction to nMOS and CMOS VLSI Systems Design." Prentice-Hall, New Jersey.
- Mukhopadhyay, A. (1979). Hardware algorithms for non-numeric computation. *IEEE Trans. Computers* **C-28**, 384–394.
- Mukhopadhyay, A., and Hurson, A. R. (1979). An associative search language for data management. *Nat. Computer Conf.*, pp. 727–732.
- Muraszkiewicz, M. (1981). Concepts of sorting and projection in a cellular array. *Proc. 7th Int. Conf. Very Large Data Bases*, pp. 76–79.
- Myers, G. J. (1978). "Advances in Computer Architecture." Wiley, New York.
- Oflazer, K. (1983). A reconfigurable VLSI architecture for a database processor. *Nat. Computer Conf.*, pp. 271–281.
- Oflazer, K., and Ozkarahan, A. E. (1980). RAP3—A multi-microprocessor cell architecture for the RAP database machine. *Proc. Int. Workshop on High Level Language Computer Architecture*, pp. 108–119.
- Oliver, E. J. (1979). RELACS, an associative computer architecture for a relational data model. Ph.D. Dissertation, Syracuse University.
- Ozkarahan, E. A., Schuster, S. A., and Smith, K. C. (1974). A data base processor. Technical Report, University of Toronto.
- Ozkarahan, E. A., Schuster, S. A., and Smith, K. C. (1975). RAP—an associative processor for data base management. *Nat. Computer Conf.*, pp. 379–387.
- Ozkarahan, E. A., Schuster, S. A., and Smith, K. C. (1976). A high level machine-oriented assembler language for a database machine. Technical Report, University of Toronto.
- Parhami, B. (1972). A highly parallel computing system for information retrieval. *Fall Joint Computer Conf.*, pp. 681–690.
- Parker, J. L. (1971). A logic per track retrieval system. Technical Report, University of British Columbia.
- Qadah, G. Z. (1985). Database machines: a survey. *Nat Computer Conf.*, **54**, 211–223.
- Qadah, G. Z., and Irani, K. B. (1985). A database machine for very large relational databases. *IEEE Trans. Computers* **C-34** 1015–1025.

- Rosenthal, R. S. (1977). The data management machine, A classification. *Third Computer Architecture for Non-numeric Processing*, pp. 35–39.
- Rudolph, J. A. (1972). A production implementation of an associative array processor—STARAN. *Fall Joint Computer Conf.*, pp. 229–241.
- Schuster, S. A., Nguyen, H. B., Ozkarahan, E. A., and Smith, K. C. (1979). RAP2—an associative processor for databases and its application. *IEEE Trans. Computers C-28*, 446–458.
- Shaw, E. (1980). Relational operator. *Minnowbrook Workshop on Data Base Machine*.
- Slotnick, D. L. (1970). Logic per track devices. In “Advances in Computers” 10 (F. L. Alt. and M. Rubenoff, eds.). Academic Press, New York.
- Smith, D. C., and Smith, J. M. (1979). Relational database machines. *Computer* 12, 28–38.
- Song, S. W. (1980). A highly concurrent tree machine for database applications. *Proc. Int. Conf. Parallel Processing*, pp. 259–268.
- Song, S. W. (1983). A survey and taxonomy of database machines. In “Database Engineering” (W. Kim, D. Batory, A. Herner, R. Katz and D. Reiner, eds.), pp. 5–15.
- Su, S. Y. W. (1979). Cellular logic devices concepts and application. *Computer* 12, 11–25.
- Su, S. Y. W., and Emam, A. (1980). CASDAL: CASSM’s Data Language. *ACM Trans. Database Systems* 3, 57–91.
- Su, S. Y. W., Nguyen, H. B., Emam, A., and Lipovski, G. J. (1979). The architectural features and implementation techniques of the multi cell CASSM. *IEEE Trans. Computers C-28*, 430–445.
- Su, S. Y. W., et al. (1980). Database machines and some issues on DBMS standards. *Nat. Computer Conf.*, pp. 191–208.
- Tanaka, K., and Kambayashi, Y. (1981). Logical integration of locally independent relational databases into a distributed database. *Proc. 7th Int. Conf. Very Large Data Bases*, pp. 131–141.
- Tong, F., and Yao, S. B. (1982). Performance analysis of database join processors. *Nat. Computer Conf.* 51, pp. 627–637.
- Valduriez, P., and Gardarin, G. (1984). Join and semijoin algorithms for a multiprocessor data base machine. *ACM Trans. Database Systems* 9, 133–161.
- Wah, B. W., and Yao, B. S. (1980). DIALOG—A distributed processor organization for database machine. *Nat. Computer Conf.* 49, 243–253.