Optical content-addressable parallel processor for high-speed database processing

Ahmed Louri and James A. Hatch Jr.

We extend the concept of optical content-addressable parallel processing [Appl. Opt. **31,** 3241 (1992)] to a novel architecture designed specifically for the parallel and high-speed implementation of database operations called optical content-addressable parallel processor for relational database processing (OCAPPRP). An OCAPPRP combines a parallel model of computation, associative processing, with parallel and high-speed technology optics. The architecture is developed to provide optimal support for high-speed parallel equivalence (pattern matching) and relative-magnitude searches (greater than and lesser than). Distinctive features of the proposed architecture include (1) a two-dimensional match–compare unit for two-dimensional pattern matching, (2) constant-time retrieval of database entries, (3) an optical word and bit-parallel relative-magnitude single-step algorithm, and (4) the capability of constant-time sorting. Since relational database operations rely heavily on parallel equivalence or relative-magnitude searches, database processing is an excellent candidate for implementation on an OCAPPRP. The architecture delivers a speedup factor of n over conventional optical database architectures, where n is the number of rows in a database table. We present an overview of the architecture followed by its optical implementation. The representative relational database operations, intersection, and selection are outlined to illustrate the architecture's potential for efficiently supporting high-speed database processing.

Key words: Optical computing, optical content-addressable parallel processor, optical content-addressable memories, optical database machines, relational operations, content-addressable memories.

1. Introduction

The information explosion seen in recent years has stimulated the development of computer-based information systems to assist in the creation, storage, modification, classification, and retrieval of mainly textual data. Some of these systems are known as database management systems. Although much progress has been made in the development of database management systems, they have not yet achieved the required performance.[1,2] This is mostly due to the implementation of databases on machines designed primarily for numerical computations. These machines use a memory scheme in which data items are referenced by their addresses, which implies searching data one word at a time. This requires the use of sophisticated software techniques to reduce the search time.[3] However, resorting to software techniques has resulted in complex, expensive, and even slower systems.

A solution to the problems encountered with these techniques is accomplished by the use of associative processing based on content-addressable memories[4,5] (CAM's). The absolute location of data items in an associative memory has no logical significance. Data items are retrieved by a search of all data entries in parallel at the memory level. However, CAM's are more expensive to build and have lower storage density than conventional address-based memories because of the extra hardware needed for comparison, manipulation, routing, enabling and disabling, and output selection logic. Since data items are searched in parallel, the search string must be broadcast to all cells, which complicates the interconnect design. Furthermore, CAM's require parallel readout because multiple entries may satisfy a given search. The lack of parallel input–output usually offsets the benefits of CAM's.[6] Associative memories may be more efficiently realized if we can implement them with a more inherently parallel technology, such as optics, that is also capable of providing the parallel readout of results. The advantage of optics for providing efficient support for future parallel process-

The authors are with the Department of Electrical and Computer Engineering, The University of Arizona, Tucson, Arizona 85721.

Received 22 October 1993; revised manuscript received 22 June 1994.

0003-6935/94/358153-11$06.00/0.

ing systems have been cited on numerous occasions.[7–9] These advantages include large bandwidth, innate parallelism, and noninterfering propagation along with the capacity for three-dimensional interconnects. The ease with which optical signals can be expanded (which allows for signal broadcasting) and combined (which allows for signal funneling) can also be exploited to solve the interconnect design and alleviate network latency problems. The combination of optics and associative processing has the potential to overcome the above limitations and deliver an efficient overall platform for parallel database processing.

In this paper we extend the concept of optical content-addressable parallel processing[4] to a new architecture called an optical content-addressable parallel processor for relational database processing (OCAPPRP). The architecture is developed to provide optimal support for high-speed parallel equivalence (pattern matching) and relative-magnitude searches. Since relational database queries can be broken down into combinations of these two searches, database processing is an excellent candidate for implementation on an OCAPPRP. Distinctive features of the proposed architecture include (1) an optical two-dimensional (2-D) match–compare unit (MCU) for highly parallel matching, (2) an optical word and bit-parallel algorithm for constant-time relative-magnitude searches[10] (namely, greater-than, lesser-than, and in-between limits, etc.), and (3) constant-time enumeration sorting. We also discuss techniques for implementing an optical output unit (OU) for the parallel readout of selected database items to eliminate the current serial bottleneck. The benefits of the architecture and its features described above are numerous. Most importantly, the optical 2-D MCU searches multiple patterns through a database table simultaneously. This fully exploits the inherent parallelism of relational database processing. Furthermore, our development of an optical bit-parallel relative-magnitude search algorithm eliminates the execution time difference between equivalence and relative-magnitude searches. Until now, optical relative-magnitude searches have been restricted to bit-serial implementations,[11] whereas optical equivalence searches have been bit parallel. The new algorithm also improves sorting. When multiple relative-magnitude searches are performed in parallel, data can be sorted in cosntant time.

The rest of the paper is organized as follows. Section 2 reviews the fundamentals of database processing. Section 3 presents the organizational structure of the OCAPPRP. An ongoing example illustrates the flow of data through the system as each unit is reviewed. Section 4 follows with the optical implementation of an OCAPPRP. Section 5 presents the algorithms for the intersection and selection operations to demonstrate the architecture's potential for high-speed database processing. Section 6 concludes the paper.

2. Fundamentals of Database Processing

A database is a computerized record-keeping system that maintains information and makes it available on demand.[12] Although there are numerous models on which to design a database system, most current systems follow the relational model to a large extent. The foundation of the relational model is a data structure known as a relation, which is similar to a data file. As an example, we demonstrate the manipulation of the employee database illustrated in Fig. 1. An employee relation, represented by relations A and B in the figure, is a table in which each row represents an entire employee record called a tuple. The columns, called attributes, store items of information such as employee name, number, years of employment, age, etc. Data are written to and retrieved from databases through a set of operations, known as relational algebra, which consists of union, intersection, difference, product, selection, projection, and join. To review these operations briefly, we begin with union. The union of two relations, A and B, forms a new relation consisting of tuples residing in either A or B while intersection selects only tuples common to both A and B. Selection retrieves tuples that satisfy a specified conditional expression, from a relation. Meanwhile, projection forms the vertical subset of a relation by selecting specified attribute(s) from it. In Fig. 1, the selection operation searches for all employees of more than five years by selecting all tuples in relation A with a value in the years of employment attribute greater than five. Projection over the employee name attribute of the selected tuples retrieves the employees' names. An explanation of the other oeprations may be found in Ref. 12.

3. Optical Content-Addressable Parallel Processor for Relational Processing

3.A. Overview

In this section we describe the organizational structure of an OCAPPRP. Throughout the discussion,

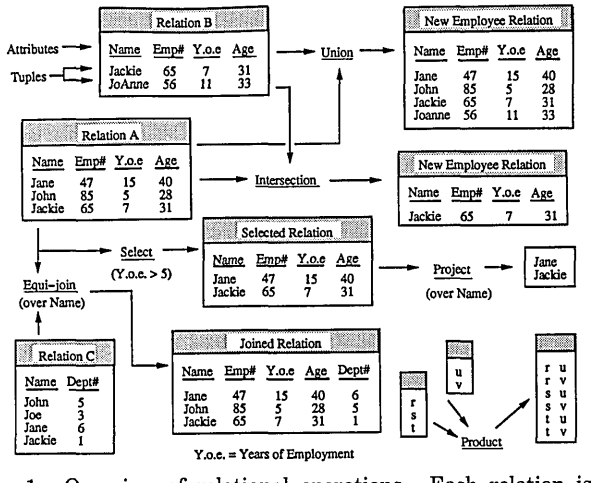

Fig. 1. Overview of relational operations. Each relation is a database table with rows called tuples and columns called attributes.

we show how the architecture provides optimal support for relational database processing by fully exploiting the parallelism of database operations. We begin by discussing 2-D matching, one of OCAPPRP's most important attributes. 2-D matching, which has been previously discussed in Ref 13, is the capability of searching multiple strings through a database table, i.e., the search of n strings through the same database table in parallel. Thus for tables with k entries, the system performs nk word-pair comparisons at the same time. By contrast, one-dimensional (1-D) matching is limited to the parallel search of a single string through a k-entry table, delivering only k simultaneous searches.

In addition to developing 2-D matching, we also developed a new algorithm to execute relative-magnitude searches in constant time, (i.e. a single-step algorithm). Equivalence searches are easily implemented as bit-parallel operations since a mismatch in any bit position of two words indicates their inequality. Step one simply XORs the two words to determine bit-by-bit mismatches, and step two sums across the bit positions of the result to search for one or more mismatches. However, for two words that are not equal, the relative magnitude is not immediately known. Relative-magnitude searches are more complicated because only the first bit position to result in a mismatch, beginning with the most-significant bit, is relevant. The other bit positions must not be able to factor into the result. The problem of isolating this bit position has limited relative-magnitude searches to bit-serial implementations, requiring up to m iterations, where m is the word size. We exploit the interconnection capabilities of optics in our new bit-parallel technique for isolating this bit position and thus perform relative-magnitude comparisons in constant time. Thus both equivalence and relative-magnitude searches now operate in the same time complexity, which eliminates the execution-time discrepancy previously experienced by algorithms that utilize both of these searches. Moreover, the combination of 2-D matching and bit-parallel relative-magnitude searches results in the potential for constant-time sorting, a breakthrough in itself. We discuss each of these innovations and the role of each unit in Subsection 3.B as we illustrate the flow of data for a representative search. The detailed optical implementation of an OCAPPRP follows in Section 4.

3.B. Description of an Optical Content-Addressable Parallel Processor for Relational Database Processing

In Fig. 2, a preliminary organizational structure for an OCAPPRP is illustrated. The architecture consists of a selection unit, a 2-D MCU, an equality unit, a threshold unit, a 2-D OU, and a control unit. The purpose of the selection unit is to enable word and bit slices of an array of search strings (comparands) called the comparand array (CA). The CA consists of n comparands of word size m, organized one per row for the simplicity of the diagram. These comparands may be tuples, from relations, for operations that operate on two relations, such as union. They may also be the arguments of conditional searches, such as the number 5 in the search for all database entries greater than 5. As discussed above, each of the n comparands searches the same database table simultaneously.

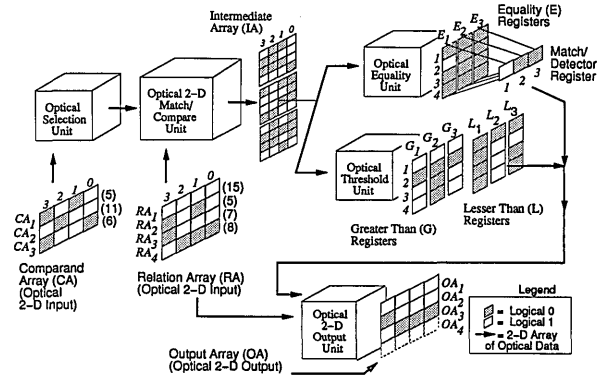

Fig. 2. Structural organization of the OCAPPRP. In the figure, the values written to the CA are simultaneously compared with each of the four values in the RA. The E, G, L, and match/detector registers store the results. Column registers G_h, L_h, and E_h represent the greater-than, lesser-than, and equality searches, respectively, of row CA_h and the RA. As an example, we search for all RA values that are greater than 5. Since the number 5 appears in CA_1, register G_1 indicates that RA_1, RA_2, and RA_4 are larger than 5. These rows are then copied to the output array by the 2-D OU.

The 2-D MCU searches the comparands through a data array known as the relation array (RA), which stores the database table (relation) being searched. Similar to the CA, the RA consists of k tuples of word size m. To explain the operation of the 2-D MCU, we begin with the simple case of one-dimensional searching. The search of a single comparand through the RA begins with the XOR of the two, as in the following expression, where CA = $ca_{(m-1)} \cdots ca_1 ca_0$ is simultaneously compared against all entries of the RA:

$$[ca_{(m-1)} \cdots ca_1 ca_0] \oplus \begin{bmatrix} ra_{1(m-1)} & \cdots & ra_{11} ra_{10} \\ ra_{2(m-1)} & \cdots & ra_{21} ra_{20} \\ \vdots & \vdots & \vdots \\ ra_{k(m-1)} & \cdots & ra_{k1} ra_{k0} \end{bmatrix}$$

$$= \begin{bmatrix} ra_{1(m-1)} \oplus ca_{(m-1)} \cdots ra_{11} \oplus ca_1 ra_{10} \oplus ca_0 \\ ra_{2(m-1)} \oplus ca_{(m-1)} \cdots ra_{21} \oplus ca_1 ra_{20} \oplus ca_0 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ ra_{k(m-1)} \oplus ca_{(m-1)} \cdots ra_{k1} \oplus ca_1 ra_{k0} \oplus ca_0 \end{bmatrix}, \quad (1)$$

where the symbol $\oplus$ represents the XOR operation. From Eq. (1), we see that the 1-D search is actually a vector–matrix XOR of a CA row and the RA. We generalize the search to 2-D by representing the CA as CA_h where $h = 1, 2, \ldots, n$ and $CA_h = ca_{h(m-1)} \cdots$

$ca_{h1}ca_{h0}$. Thus

$$\begin{bmatrix} ca_{1(m-1)} \cdots ca_{11}ca_{10} \\ ca_{2(m-1)} \cdots ca_{21}ca_{20} \\ \vdots \quad\quad \vdots \quad \vdots \\ ca_{n(m-1)} \cdots ca_{n1}ca_{n0} \end{bmatrix} \oplus \begin{bmatrix} ra_{(m-1)} \cdots ra_{11}ra_{10} \\ ra_{2(m-1)} \cdots ra_{21}ra_{20} \\ \vdots \quad\quad \vdots \quad \vdots \\ ra_{k(m-1)} \cdots ra_{k1}ra_{k0} \end{bmatrix}$$

$$= \left[\begin{array}{l} ra_{1(m-1)} \oplus ca_{1(m-1)} \cdots ra_{11} \oplus ca_{11}ra_{10} \oplus ca_{10} \\ ra_{2(m-1)} \oplus ca_{1(m-1)} \cdots ra_{21} \oplus ca_{11}ra_{20} \oplus ca_{10} \\ \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad \vdots \\ ra_{k(m-1)} \oplus ca_{1(m-1)} \cdots ra_{k1} \oplus ca_{11}ra_{k0} \oplus ca_{10} \\ \hline ra_{1(m-1)} \oplus ca_{2(m-1)} \cdots ra_{11} \oplus ca_{21}ra_{10} \oplus ca_{20} \\ ra_{2(m-1)} \oplus ca_{2(m-1)} \cdots ra_{21} \oplus ca_{21}ra_{20} \oplus ca_{20} \\ \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad \vdots \\ ra_{k(m-1)} \oplus ca_{2(m-1)} \cdots ra_{k1} \oplus ca_{21}ra_{k0} \oplus ca_{20} \\ \hline \quad\quad\quad\quad\quad\quad \vdots \\ \hline ra_{1(m-1)} \oplus ca_{n(m-1)} \cdots ra_{11} \oplus ca_{n1}ra_{10} \oplus ca_{n0} \\ ra_{2(m-1)} \oplus ca_{n(m-1)} \cdots ra_{21} \oplus ca_{n1}ra_{20} \oplus ca_{n0} \\ \quad\quad \vdots \quad\quad\quad\quad \vdots \quad\quad\quad \vdots \\ ra_{k(m-1)} \oplus ca_{n(m-1)} \cdots ra_{k1} \oplus ca_{n1}ra_{k0} \oplus ca_{n0} \end{array}\right], \quad (2)$$

where the horizontal lines separate each of the n vector–matrix XOR's. Note that each of these operations is processed in parallel and their results are stacked vertically. The functionality of the 2-D MCU is also demonstrated schematically in Fig. 3.

At this point, the search is not complete. These intermediate results, called the intermediate array (IA) in Fig. 3, merely indicate how the words compare on the bit-by-bit level. Because of the use of the XOR operation, a 0 in the IA indicates the equality of the corresponding bits from the CA and the RA. Meanwhile, a 1 indicates their inequality. To determine the equality–inequality and relative magnitude on the word level, we need to process the IA further in the equality and the threshold units, respectively.

The equality unit uses the IA to determine matches among input pairs. The second step of the equivalence search scans the IA to determine if there is at least one bit position in the comparison of the two words that results in a mismatch. This is accomplished by ORing the bits along the rows of the IA since mismatches are represented by 1's. The result is then inverted, a step that is needed only if the positive logic representation is desired. These operations are demonstrated by the following expression:

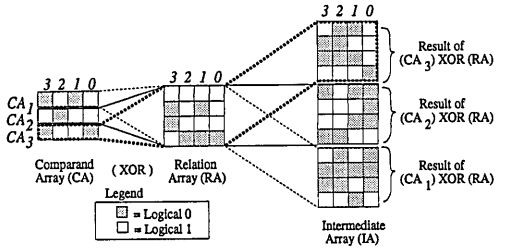

Fig. 3. 2-D MCU compares each row of the CA with the entire RA by XORing the two in a vector–matrix fashion. The results of these parallel vector–matrix operations appear vertically stacked in the intermediate array (IA), which will later be processed by the equality and the threshold units. The result of (CA_1) XOR (RA) appears in the lower block of the IA because of a vertical inversion resulting from the optical implementation.

where the symbols $\vee$ and $(^-)$ denote the logical OR and logical NOT operations, respectively. Equation (3) forms $n \times (k \times 1)$ column vectors known as the equality (E) registers. Each E register is represented as $E_h = [e_{1h}\, e_{2h} \cdots e_{kh}]^T$ where a 1 in element e_{ih} represents the equality of the hth comparand, CA_h, with the ith RA entry, RA_i. The equality unit is also demonstrated schematically in Fig. 4. From the expressions and Fig. 4; we see that the n E registers are formed in parallel. Thus nk word pairs are tested for equality in parallel with an execution time that is independent of word size. From Fig. 4, we also see that element $e_{21} = 1$, indicating that $RA_2 = CA_1$. Additionally, these registers are then vertrically ORed to form the $1 \times n$ match–detector (M/D) register. $M/D_h = 1$ indicates that CA_h matches at least one entry of the RA.

The threshold unit processes a second copy of the IA for the word and the bit-parallel relative-magnitude searches of the CA and the RA in a single step, where the term threshold is synonymous with relative magnitude. The rows of the IA indicate the bit-by-bit equality–inequality of CA_h and the RA. Since we are concerned with only the bit positions of inequality, we disable the bit positions that result in equality, i.e., logical 0's in the IA. By disabling, we mean the elimination of a bit position from further comparison. Throughout this paper, disabled bits will be represented by d. The physical meaning and the implementation of disabling are discussed in Subsection 4.B.

Once we disable the equalities, we determine the bit-by-bit relative magnitude of the two words by XORing each row of the IA with the original comparand. As mentioned above, this is not enough to determine the relative magnitude on the word level. To accom-

$$E_h = \begin{bmatrix} e_{1h} \\ e_{2h} \\ \vdots \\ e_{kh} \end{bmatrix} = \left[\begin{array}{l} \overline{[RA_{1(m-1)} \oplus CA_{h(m-1)}] \vee \cdots \vee [RA_{11} \oplus CA_{h1}] \vee [RA_{10} \oplus CA_{h0}]} \\ \overline{[RA_{2(m-1)} \oplus CA_{h(m-1)}] \vee \cdots \vee [RA_{21} \oplus CA_{h1}] \vee [RA_{20} \oplus CA_{h0}]} \\ \quad\quad\quad\quad\quad\quad \vdots \\ \overline{[RA_{k(m-1)} \oplus CA_{h(m-1)}] \vee \cdots \vee [RA_{k1} \oplus CA_{h1}] \vee [RA_{k0} \oplus CA_{h0}]} \end{array}\right], \quad (3)$$

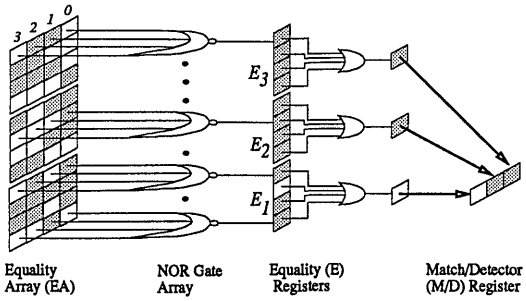

Fig. 4. Equality unit determines the match–mismatch of input pairs from the CA and the RA by detecting the presence of at least one mismatching bit position. Mismatching bit positions, represented as logical ones in the IA, are detected when the bits are NORed across IA rows. In the figure, the second row from the top of register E_1 is set, which means that the word in RA_2 is equal to the word in CA_1. TRhe M/D register reports the matching of the corresponding CA row with at least one row of the RA.

plish this, we must isolate the first bit position, beginning with the most-significant bit, to result in an inequality. The word with a 1 in this bit position is the larger-valued word. For example, the latter of the binary patterns $A = 10100$ and $B = 11011$ is the larger-valued word because it contains a 1 in the highest bit position, denoted by $j = 3$, that results in an inequality. To perform relative magnitude in constant time, the other bit positions, of lower-index j, must be disabled in parallel. Thus a 1 in bit position j, for all j, must disable all bit positions $(j - 1), (j - 2), \ldots, 0$. Further discussion of this operation is delayed until Subsection 4.D since its operation is implementation dependent.

The output of the threshold unit is two sets of registers called the greater-than (G) registers, where $G_h = \{g_{1h}g_{2h} \cdots g_{kh}\}^T$, and the lesser-than (L) registers, $L_h = \{l_{1h}l_{2h} \cdots l_{kh}\}^T$. Similar to the E registers $g_{ih} = 1$ indicates that RA_i is greater than CA_h. Similarly, $l_{ih} = 1$ indicates that RA_i is lesser than CA_h. The contents of these registers are also used to perform constant-time enumeration sorting. In an enumeration sort, each of N data elements is assigned a unique integer, called its rank, which indicates its position in the sorted array. The rank is a sort of pointer for the data set, meaning that the data need not be physically reordered in memory. Let x and y be indices ranging in value from 1 to N for a data set D, where $D = d_1d_2 \ldots d_N$; each element d_x is assigned a rank r_x, where $r_x = 1, 2, \ldots, N$. Thus $r_x < r_y$ if and only if $d_x < d_y$. To translate this into an operation, we first load the data array to be sorted into both the CA and the RA. In effect, we are comparing each data element with all the others, in parallel. Since the G register contains 1's for all rows of the RA that are greater than their corresponding rows in the CA, we can determine the comparand's position in the sorted array simply by summing the logical values in its G register. Since all G registers are available in constant time, the enumeration sort of a data array is also performed in constant time. To this end, the system is able to determine the relationship between the elements of the CA and the relation stored in the RA. Words that are equal are reported in the E registers, words that are greater than are reported in the G registers, and words that are lesser than are reported in the L registers.

The 2-D OU transfers selected tuples of the RA to the optical output array (OA). It dynamically maps nonconsecutive input tuples onto consecutive rows of the OA. In Fig. 3, rows RA_1, RA_3, and RA_4 are mapped onto rows OA_1, OA_2, and OA_3, respectively. The necessity of the 2-D OU is justified by the following argument. Since the 2-D MCU is capable of searching multiple comparands through the same table in parallel, the output unit must be capable of parallel readout. If the output unit of a 2-D system is capable of transferring only single tuples to the OA each cycle (as in the 1-D case), then the system is fundamentally limited by its output facilities rather than by its processing capability. Many authors in the past have argued that the transition to a 2-D operation in the matching unit directly results in a dimensionality of processing improvement. However, they failed to indicate that the OU must also be redesigned, with increased complexity, to support this higher level of processing. In Subsection 4.E, we discuss techniques for performing this function.

As an additional benefit, searches that require some logical operation (AND, OR, NOT) of the E, G, or L registers may be enhanced if the 2-D OU contains some combinational logic. For instance, between-limits searches, such as the search for all RA rows greater than 5 and less than 11, normally perform a greater-than search and feed the reduced data set through a lesser-than search on the second pass. Instead, both comparands can be loaded into the CA during the first pass. The greater-than and the lesser-than searches are simultaneously performed, and the results, which appear in the corresponding G and L registers, are ANDed together, eliminating the overhead of an additional cycle.

In summary, the system is capable of performing the following searches in constant time:

- equivalence (or equality), $\{=, \neq\}$;
- relative magnitude, $\{>, <, \geq, \leq\}$;
- between-limits search; the search for words z between two limits X and Y $(X < Y)$ for the conditions $X \leq z \leq Y$, $X < z \leq Y$, $X \leq z < Y$, and $X < z < Y$;
- outside-limits search, the search for words z outside two limits X and Y $(X < Y)$ for the conditions $X \geq z \geq Y$, $X > z \geq Y$, $X \geq z > Y$, and $X > z > Y$;
- enumeration sorting.

4. Optical Implementation of an Optical Content-Addressable Parallel Processor for Relational Database Processing

In this section, we present the detailed optical implementation for selection, 2-D MCU's, equality, threshold, and 2-D OU's of an OCAPPRP. We begin by discussing the logic-encoding scheme and active-

switching devices used. This is followed by the optical implementation of each unit.

4.A. Logic-Encoding Scheme and Active Devices

Any physical implementation of an optical computing system must provide a means of encoding binary-valued data as optical signals capable of performing logic. For intensity-encoded data, 0's are represented by the absence of light while 1's are indicated by its presence. However, polarization may also be used as a means for optically encoding data.[14] In our system, we use a combination of intensity- and polarization-encoding schemes.[15] Polarization encoding is used primarily for performing comparisons. Internally, 0's are represented by vertically polarized light and 1's are represented by horizontally polarized light. Additionally, the combination of the two encoding schemes temporarily provideds three logic states, which is extensively taken advantage of by the threshold unit.

Polarization-encoded logic is easy to manipulate. For instance, bits are disabled from further comparison when they are masked with a polarizer. To understand how the XOR operation is performed, consider an optical data beam from the selection unit, consisting of vertically and horizontally polarized light, as in Fig. 5. The incident data plane, which represents the CA in this simplified example, is XORed bit by bit with the logical values written to the spatial light modulator (SLM), which stores the RA. From the diagram, we see that the result is the logical XOR of the CA and the RA, which represents the IA. For this example, the SLM is configured such that a logical 0 written to it results in no polarization rotation of the incident light, while a logical 1 results in a 90° polarization rotation.

For our system, we primarily use optically addressable ferroelectric liquid-crystal (FLC) SLM's[16,17] consisting of 2-D arrays of FLC cells. In a typcial FLC cell, a positive voltage applied to the cell results in a 90° polarization rotation of the light whereas a negative applied voltage of the same magnitude results in no rotation. Optically addressed SLM's (OASLM's) contain a photoconductive layer integrated onto one side of the liquid-crystal cell. A 2-D optical wave front impinges on the photoconductive layer of the SLM, which converts the spatial intensity variations

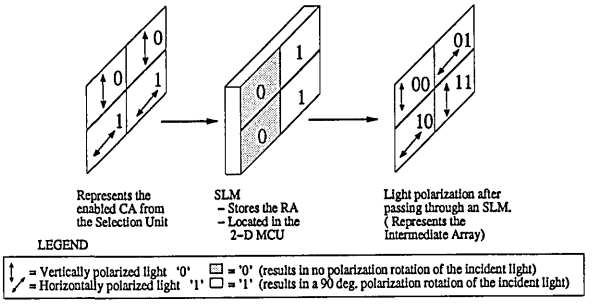

Fig. 5. Polarization-encoding scheme used for XORing the CA and the RA. The enabled CA is XORed with the logical values written to the spatial light modulator (which stores the RA) to form the IA.

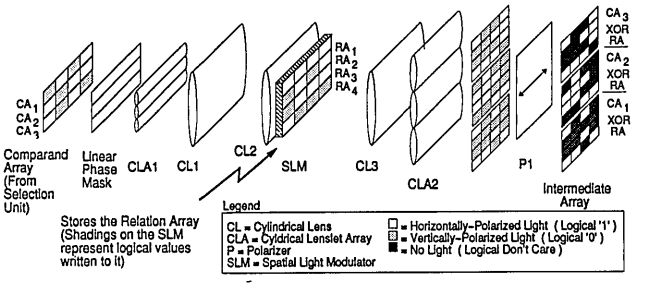

Fig. 6. Optical implementation of the 2-D MCU. Each row of the CA is XORed with the RA and reported in the IA in accordance with Fig. 3.

into electrical control signals for the corresponding FLC cell. An optical readout beam incident upon the other side is modulated on reflection.

4.B. Optical Selection and Two-Dimensional Match–Compare Units

The optical implementation of the selection unit, because of its simplicity, consists of only an SLM and some bulk optical components. To perform the word and bit-slice enabling–disabling of the selection unit, bits of the optical input are disabled from further computation by blocking them. This can be accomplished by a switch of the corresponding SLM pixels from the transparent to the opaque state. In the current example, all the rows of the CA are enabled.

The 2-D MCU, on the other hand, is more involved. It uses a custom-designed matrix–matrix multiplier to achieve 2-D matching. Figure 6 illustrates the optical implementation of the 2-D MCU, and a side view illustrating sample ray traces follows in Fig. 7. The system is based on the matrix–matrix multiplier proposed by Gheen in Ref. 18. In his system, he uses a lienar phase mask inserted into a traditional vector–matrix multiplier to permit multiple-input row vectors to be multiplied by a SLM and to be spatially distinguished as output column vectors. In our system, we also use a phase mask to separate the input

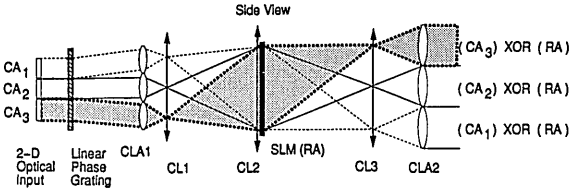

Fig. 7. Side view of Fig. 6, which illustrates the operation of the 2-D MCU. Each row of the CA is deflected by a different angle. A cylindrical lens array focuses them so that they each form individual expanding beams. The cylindrical lens in the focal plane redirects the off axis beams so that they all pass through the SLM. The cylindrical lens–SLM combination performs the XOR operation while focusing the individual beams so that they can be properly scaled and aligned. The realignment step is performed by the third cylindrical lens placed in the beam's focal plane. The following cylindrical lens array collimates the result beams into a vertically stacked arrangement to form the IA. Because only cylindrical lenses are used, the input beams are unaltered along the top view as they propagate through the system. Notation is the same as that of Fig. 6.

vectors by deflecting each of the CA rows by different angles. A cylindrical lens array individually focuses each row so that they each form expanding beams. A cylindrical lens in the focal plane redirects the off-axis beams so that they all pass through the SLM. The cylindrical lens–SLM combination performs the XOR operation while also focusing the individual beams so that they can be properly aligned at the output. For the results to emerge from the system as parallel collimated beams, the third cylindrical lens, which is placed in the focal plane, rotates the off-axis beams so that they are parallel. The cylindrical lens array collimates the result beams and aligns them into vertically stacked data planes to form the IA. Because of the use of only cylindrical lenses, the input beams are unaltered along the top view as they propagate through the system.

Because of the use of polarized logic, both logical values of the IA are represented by light differing only in polarization. The 0's of the IA are disabled by the blocking of any vertically polarized light with a polarizer. Without light, a bit position is no longer capable of participating in computations. This disabling operation is necessary for the equality and the threshold units to isolate mismatches. In addition to the characteristics discussed in Subsection 3.B, disabled bits are to be treated as logical 0's that cannot be set by a device other than an OASLM since it is the only component that responds to intensity differences.

4.C. Optical Equality Unit

The equality unit uses the results stored in the IA to determine the equality of the elements of the CA and those of the RA. The bit positions of the IA, which are indexed by j, are bright in those locations where the bits of the two corresponding input words are not equal. Thus, if at least one bit position of a row is bright, the two corresponding input words are not equal. We detect inequalities by using a cylindrical lens to focus the IA to a vertical line and then inverting the result. As mentioned in Subsection 3.B, the inversion is necessary only if the designer desires the positive logic representation. We form the M/D register by vertically focusing each of the E registers with a cylindrical lens array to form a single vector. These registers are then transferred to the 2-D OU where they participate in the selection of output tuples.

4.D. Optical Threshold Unit

The optical implementation of the threshold unit is illustrated in Figs. 8, and 9. According to the discussion in Subsection 3.B, the first step for constant-time relative-magnitude comparison is the bit-by-bit XOR of each comparand with the IA. Since each comparand is to be XORed with k rows (corresponding to the size of the RA), the CA is scaled (not shown) by a factor of k to form the expanded CA. The XOR is performed by the imaging of the IA onto the reflective side of an OASLM through a beam splitter. The

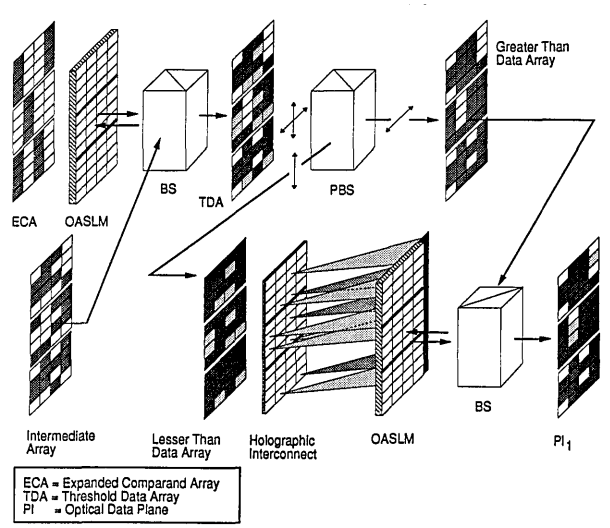

Fig. 8. Optical implementation of the threshold unit. The optical implementation is continued in Fig. 9. A detailed illustration of the holographic interconnect may be found in Ref. 10. BS's, beam splitters; PBS, polarizing beam splitter.

expanded CA simultaneously writes its values to the photoconductive side of the OASLM. This bit-by-bit operation determines where ra_{ij} is greater than or lesser than ca_{hj}. The output of the OASLM is referred to as the threshold data array and consists of horizontally polarized light in bit positions where $ra_{ij} > ca_{hj}$, vertically polarized light in those where $ra_{ij} < ca_{hj}$, and no light where $ra_{ij} = ca_{hj}$.

The next step involves finding the first bit position, beginning with the most-significant bit, where an inequality exists, and then disabling, in parallel, the other bit positions of lower-index j. Using the example in Subsection 3.B to envision the disabling process, consider the creation of two data planes, the greater-than data array (GTDA) and the lesser-than data array (LTDA). The GTDA contains a 1 in each bit position where $B_j > A_j$ and disabled bits elsewhere, while the LTDA contains 1's in bit positions

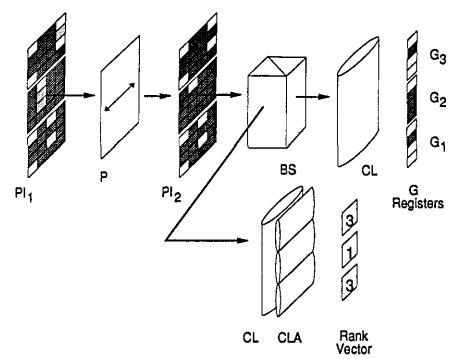

Fig. 9. Optical implementation of the threshold unit (continued). Rotated bits of the greater-than data array resulting from the disabling step appear in PI_1. They are then disabled by a polarizer to form PI_2. One copy of PI_2 is horizontally summed to form the G registers while the other copy is vertically and horizontally summed to form the rank vector, which is used only for sorting. Notation is the same as that of Figs. 6 and 8.

where $B_j < A_j$ and disabled bits elsewhere. In this example, the LTDA is $dd1dd$ while the GTDA is $d1d11$. Logically, each GTDA bit position is ANDed with the NOR of the LTDA bits in higher positions. Thus if a 1 occurs in a higher bit position of the LTDA, then the corresponding bit of the GTDA is forced into the disabled state if it is not there already. Otherwise, it is unaffected. Therefore if the GTDA still has 1's in any of its bit position(s), then $B_j > A_j$ is in a higher bit position than the first $B_j < A_j$, meaning that $B > A$. If all the GTDA bits are disabled, then the LTDA has a 1 in the highest bit position to result in an inequality. However, there may not have been any bright bits in the GTDA to begin with, i.e., the two words are equal. Thus a fully disabled GTDA row indicates only that $A \leq B$. In this example, the 1's in the LTDA force all the GTDA bits in lower positions into the disabled state, forming the vector $d1ddd$, and the result $B > A$.

The optical system of Fig. 8 accomplishes this data plane generation and bit-disabling process. The polarizing beam splitter creates two data planes of separate polarizations: the GTDA containing horizontally polarized light only in positions where $ra_{ij} > ca_{hj}$ and the LTDA containing vertically polarized light only in positions where $RA_{ij} < CA_{hj}$. A space-invariant computer-generated holographic interconnect images bits of the LTDA onto their corresponding lower bit positions of an OASLM. A more detailed illustration of this hologram is published in Ref. 10. The diffracted beams from the hologram impinge on the photoconductive side of the OASLM while a beam splitter images the GTDA onto its reflective side. The electric fields established in the OASLM by the diffracted beams rotate the polarization of the GTDA bits in the corresponding locations to transform the appropriate 1's to 0's. A beam splitter then transmits reflected data plane Pl_1. In Fig. 9, a polarizer disables any rotates bits (0's) to form Pl_2. The cylindrical lens searches for any remaining bright GTDA bits in one copy of Pl_1 by focusing each row to form the G register. A bright G-register bit means that $RA_i > CA_h$ while a dark bit means that $RA_i \leq CA_h$. The G registers are formed in a single pass through the threshold unit. The L registers (not shown) are realized as a logical function of the G and the E registers in order to eliminate the need for duplicate hardware. Additionally, the greater-than-or-equal and lesser-than-or-equal registers (not shown) are formed by the inversion of the L and the G registers, respectively. Meanwhile, the cylindrical lens–cylindrical lens array combination in Fig. 9 forms each G register in another copy of Pl_2 and sums the values in them to form the rank vector. Note that the values in the rank vector are meaningful only when the same table is loaded into both the CA and RA, i.e., sorting is being performed.

4.E. Optical Two-Dimensional Output Unit

In this subsection, we describe the design of the 2-D OU briefly in order to keep within page limitations. The function of mapping nonconsecutive input rows onto consecutive output rows can be implemented in a couple of ways. One possible method uses an acousto-optic (AO) cell placed in the Fourier plane of a 4-f imaging system.[19] The optical setup for this is illustrated in Fig. 10. The AO cell shifts the image in the output plane by redirecting the light in the Fourier plane of the input image. To control each row separately, the lenses of a typical 4-f system are replaced with cylindrical lens arrays and the AO cell is replaced with a 2-D AO cell.

In the current search for all RA rows such that $RA_i > 5$, selected rows of the RA to be mapped onto the OA are indicated by 1's in register G_1. We perform the mapping by deflecting each input row by an angle proportional to the difference in row position between the input and the output data planes. We determine this difference by counting, for each row, the number of preceding rows that did not satisfy the search and are not transferred to the OA. We perform this operation by first inverting and broadcasting register G_1 to form a $k \times k$ matrix. We then disable all below-diagonal entries of this matrix and sum the number of 1's in each column. This operation is shown below:

$$\begin{array}{cccc} 1 & 0\ 0\ 0\ 0 & 0\ 0\ 0\ 0 \\ 0 & 1\ 1\ 1\ 1 & d\ 1\ 1\ 1 \\ 1 \rightarrow & 0\ 0\ 0\ 0 & \rightarrow\ d\ d\ 0\ 0 \\ 1 & 0\ 0\ 0\ 0 & d\ d\ d\ d \\ & & \hline & & 0\ 1\ 1\ 1 \end{array} . \quad (4)$$

The output 0111 modulates the AO cell to perform the required parallel mapping. Rows of the RA that are not transferred to the OA should be disabled so that they are not imaged by the 2-D AO cell. From the dotted lines in Fig. 10, we see that RA_1 is mapped onto OA_1 while RA_3 and RA_4 are mapped onto OA_2 and OA_3, respectively.

To this end, we have presented the optical implementation of the selection, 2-D MCU's, equality, threshold, and 2-D OU's of an OCAPPRP. We are in the process of constructing an experimental prototype of the architecture, the details of which will be

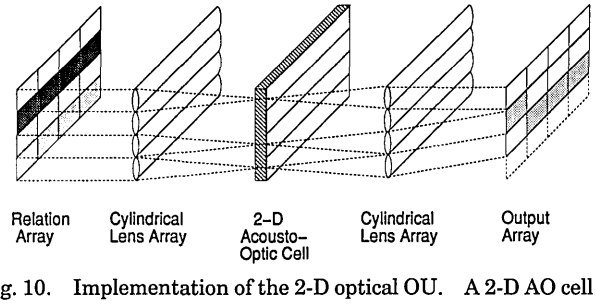

Fig. 10. Implementation of the 2-D optical OU. A 2-D AO cell in the Fourier plane of a 4-f system redirects the RA rows so that they are shifted in the OA. Row RA_2 is disabled since it did not satisfy the search and does not appear in the output.

delayed to a later paper at the recommendation of the reviewers.

5. Algorithms for Performing Relational Operations on an OCAPPRP

In this section we present parallel algorithms for relational operations that are efficiently implemented on an OCAPPRP. Although we have developed parallel algorithms for all relational operations, union, intersection, selection, projection, and join, we present only two representative operations, intersection and multiple-argument selection, for clarity and page limitation purposes.

5.A. Intersection

The itnersection operation forms a relation from tuples that belong to both of two specified relations. It is implemented by the performance of an equivalence search on two relations loaded from optical memory into the processor (as the CA and RA). When the comparison is completed, the M/D register is used by the 2-D OU to copy rows of the CA with matching RA entries to the OA.

Intersection Algorithm

1. Initialize by using the following steps:
   (a) Load the first of two relations into the CA.
   (b) Load the second of two relations into the RA.
   (c) Clear registers E, G, L, and M/D.
2. In the selection unit, enable all rows of the CA.
3. Perform the bit-parallel equivalence comparison of the two arrays.
4. In the 2-D OU, use the M/D register to transfer the selected CA rows onto consecutive rows of the OA.

5.B. Selection

The selection operation forms a subset of a single relation by resolving conditional expressions on selected attribute(s). For instance, the search for all tuples with a value in attribute C of the RA where $C_i > 5$ AND $C_i < 11$ involves the greater-than search of C with the number 5 and a lesser-than search of C with the number 11. Since only attribute C is involved in the operation, the selection unit disables all other nonparticipating attributes. The selection arguments, the numbers 5 and 11, are loaded into CA_1 and CA_2, respectively, while the relation being searched is loaded into the RA. The 2-D OU logically ANDs registers G_1 and L_2. Because of the logical processing capability of the 2-D OU, selections consisting of multiple arguments are processed in a single pass through the optical system instead of sequentially as multiple single-argument operations. Thus, multiple- and single-argument selections have the same time complexity, assuming that the number of arguments is less than n.

Multiple-Argument Selection Algorithm

Search: Find all RA rows where $C_i > 5$ AND $C_i < 11$.

1. Initialize by using the following steps:
   (a) Load rows CA_1 and CA_2 with the constants 5 and 11, respectively.
   (b) Load the relation being searched into the RA.
   (c) Clear registers E, G, L, and M/D.
2. In the selection unit, enable attribute C.
3. Perform the bit-parallel relative-magnitude comparison of the two arrays.
4. In the 2-D OU,
   (a) AND registers G_1 and L_2.
   (b) Transfer the selected rows from the RA onto consecutive rows of the OA.

The above algorithm assumes, for simplicity, that the database tables have less than k tuples. Larger tables must be broken down into segments of k tuples, which must then be cycled through. Since the simplified version of the algorithm requires no looping, both the single- and the multiple-comparand selections of a k-item table occur in constant time. Furthermore, selections involving relative-magnitude searches, such as the one above, execute in the same time as those involving equivalence searches because of the use of the single-step algorithm.

Table 1 summarizes the execution times for each of the relational operations. This is intended to be

Table 1. Summary of the Execution Times for Implementing the Full Set of Relational Operations on an OCAPPRP[a]

Relational Operation	Execution Time on a Serial Processor	Execution Time on OCAPPRP	Speedup
Union	$O(n \log n)$	$O(1)$	$O(n \log n)$
Intersection	$O(n \log n)$	$O(1)$	$O(n \log n)$
Selection single argument	Best case		
	$O(k)$	$O(1)$	$O(k)$
	Worst case		
	$O(w \times m)$	$O(1)$	$O(w \times m)$
Selection multiple arguments	Best case		
	$O(q \times w)$	$O(1)$	$O(q \times w)$
	Worst case		
	$O(q \times w \times m)$	$O(1)$	$O(q \times w \times m)$
Projection	$O(n \log n)$	$O(1)$	$O(n \log n)$
Join	$O(n \log n)$	$O(n)$	$O(\log n)$
Sorting	$O(n \log n)$	$O\left(\frac{n}{p}\right)$	$O(p \log n)$

[a] m, the word size in the CA's and the RA's; n, the number of words held in the CA; k, the number of words held in the RA; q, the number of arguments in a multiple-argument selection; p, the number of parallel vector–matrix multiplications supported by the threshold unit.

merely a quick summary rather than a thorough examination of an OCAPPRP performance. We are currently in the process of preparing a more thorough performance analysis. In the analysis, only the execution time of the optical processing elements is considered. We assume that the setup and the transfer times can either be overlapped with the execution time or may be reduced so that they do not dominate the performance complexity. It is also assumed that both of the relations are capable of being stored in SLM's of currently available sizes. Thus many of the operations can be processed in constant time because of OCAPPRP's 2-D matching capability. Tables of larger size need to be partitioned, in which case iterative terms should be included in the execution times. The indices m, n, and k retain the same meaning as they carried throughout the paper; m is the word size, n is the number of rows in the CA, and k is the number of tuples in the RA. In the execution time of multiple-argument selection, it is assumed that q, the number of arguments in the selection, is less than n, i.e., all the arguments can fit in the CA without partitioning.

The number p represents the parallelism of the threshold unit. Since the 2-D MCU processes n vector–matrix XOR's in parallel, the threshold unit requires hardware of size $nk \times m$ for maximum parallelism. However, as the values of n and k increase, the spatial complexity of the threshold unit becomes large. Thus we may desire to include enough hardware only to process p, where $p < n$, of the vector–matrix results in the IA at a time. The execution time for sorting is therefore $O(n/p)$ where $O(n/p) \rightarrow O(1)$ as $p \rightarrow n$. This factor is ignored in the execution time of selections requiring relative-magnitude searches since it is assumed that $q < p$.

6. Discussion

Current database management systems have not yet achieved the performance levels demanded by applications such as real-time command and control and high-speed information retrieval. Since database searches are highly parallel, associative processing can improve the performance of these systems. In particular, the use of optical CAM's offers the benefits of associative processing along with the advantages of optics.

In this paper, we presented an optical architecture called OCAPPRP for the high-speed processing of database operations based on a 2-D optical CAM. The architecture benefits from improvements resulting from new techniques in hardware and algorithm design. Since relational database operations can be broken down into two types of searches, equivalence and relative magnitude, OCAPPRP performs both of these with maximum parallelism. For instance, the 2-D MCU searches n comparands through the k entries of a database table in constant time. To accomplsih this, we proposed a modified matrix–matrix multiplier. By further processing the intermediate results from the 2-D MCU, we were able to perform both word and bit-parallel equivalence and relative-magnitude searches in constant time. Previously, relative-magnitude searches were limited to bit-serial implementations because of the problem of isolating the first bit position of the XOR of two words to generate an inequality. Our improvement in relative-magnitude searches results from the use of a new single-step algorithm we introduced to eliminate this previous bit-serial limitation. With this algorithm, relative-magnitude searches are enhanced by a maximum factor of m, where m is the word size, and are execution-time matched with equivalence searches. This eliminates the previous execution-time differences experienced by operations that use both types of search. Furthermore, OCAPPRP is also capable of constant-time enumeration sorting because of its potential for 2-D relative magnitude searches.

Along with addressing the improvements in search algorithms, we also address the issue of parallel readout of the CAM. The extension from 1-D to 2-D matching demands the simultaneous transfer of multiple-input tuples to the output. Otherwise the system will be limited by the sequential processing speed of the output facilities. The problem lies not in transferring multiple tuples but in dynamically mapping nonconsecutive input tuples onto consecutive rows of the OA in parallel. To demonstrate the feasibility of OCAPPRP, we then presented its optical implementation with existing optoelectronic components. Because of page limitations, we merely discussed a technique for implementing a 2-D OU capable of transferring k tuples to the optical output array in constant time. (By similar reasoning, at the recommendation of the reviewers, we delay the discussion of our current experimental prototype system until a later publication.) Following this, we introduced the algorithms for constant-time selection and enumeration sorting to represent the capabilities of OCAPPRP for parallel optical data searches.

In summary, symbolic processing employing the use of 2-D optical CAM's and the innovations described in this paper offer great hope for delivering the needed performance enhancements in database–knowledgebase machines. In particular, the continued development of optoelectronic devices for spatial light modulation present the increasing potential for the experimental and, we hope, commercial realization of optical database machines.

This research was supported by National Science Foundation grant MIP 9113688.

References

1. P. B. Berra, K.-H. Brenner, W. T. Cathey, H. J. Caulfield, S. H. Lee, and H. Szu, "Optical database/knowledgebase machines," Appl. Opt. **29,** 195–205 (1990).
2. P. B. Berra, A. Ghafoor, M. Guizani, S. J. Marcinkowski, and P. A. Mitkas, "Optics and supercomputing," Proc. IEEE **77,** 1797–1815 (1989).
3. T. Kohonen, *Content-Addressable Memories* (Springer-Verlag, New York, 1980).

4. A. Louri, "Optical content-addressable parallel processor: architecture, algorithms, and design concepts," Appl. Opt. **31,** 3241–3258 (1992).
5. S. Y. Su, *Database Computers: Principles, Architectures, and Techniques* (McGraw-Hill, New York, 1988).
6. J. R. Ullman, "Fast implementation of relational operations via inverse projections," Comput. J. **31,** 147–154 (1988).
7. A. Louri, "Three-dimensional optical architecture and data-parallel algorithms for massively parallel computing," IEEE Micro. **11,** 24–68 (1991).
8. F. B. McCormick, *Photonics in Switching* (Academic, Boston, 1993), Vol. II, Chap. 4.
9. J. A. Neff, R. A. Athale, and S. H. Lee, "Two-dimensional spatial light modulators: a tutorial," Proc. IEEE **78,** 826–854 (1990).
10. A. Louri and J. A. Hatch Jr., "Optical implementation of a single-iteration thresholding algorithm with applications to parallel data-base/knowledge-base processing," Opt. Lett. **18,** 992–994 (1993).
11. R. A. Athale and M. W. Haney, "Optical implementation of numerical inequality detection and its applications to database machines," Opt. Lett. **17,** 1611–1613 (1992).
12. C. J. Date, *An Introduction to Database Systems* (Addison-Wesley, Reading, Mass., 1986).
13. S. Akyokus and P. B. Berra, "Optical content addressable memories for data/knowledge base processing," presented at the Fifth International Symposium on Parallel Processing, Anaheim, Calif., 1991.
14. A. W. Lohmann, "Polarization and optical logic," Appl. Opt. **25,** 1594–1597 (1986).
15. K. W. Wong, L. M Cheng, and M. C. Poon, "Design of digital-optical processors by using both intensity and polarization-encoding schemes," Appl. Opt. **31,** 3225–3232 (1992).
16. K. M. Johnson and G. Moddel, "Motivations for using ferroelectric liquid crystal spatial light modulators in neurocomputing," Appl. Opt. **28,** 4888–4899 (1989).
17. C. Warde and A. Fisher, "Spatial light modulators: applications and functional capabilities," in *Optical Signal Processing,* J. Horner, ed. (Academic, 1987), New York, pp. 478–524.
18. G. Gheen, "Optical matrix–matrix multiplier," Appl. Opt. **29,** 886–887 (1990).
19. H. J. Caulfield, "Massively parallel optical data base management," in *Digital and Optical Shape Representation and Pattern Recognition,* R. D. Juday, ed., Proc. Soc. Photo-Opt. Instrum. Eng. **938,** 52–54 (1988).