Constant-time parallel sorting algorithm and its optical implementation using smart pixels

Ahmed Louri, James A. Hatch, Jr., and Jongwhoa Na

Sorting is a fundamental operation that has important implications in a vast number of areas. For instance, sorting is heavily utilized in applications such as database machines, in which hashing techniques are used to accelerate data-processing algorithms. It is also the basis for interprocessor message routing and has strong implications in video telecommunications. However, high-speed electronic sorting networks are difficult to implement with VLSI technology because of the dense, global connectivity required. Optics eliminates this bottleneck by offering global interconnects, massive parallelism, and noninterfering communications. We present a parallel sorting algorithm and its efficient optical implementation. The algorithm sorts n data elements in few steps, independent of the number of elements to be sorted. Thus it is a constant-time sorting algorithm [i.e., $O(1)$ time]. We also estimate the system's performance to show that the proposed sorting algorithm can provide at least 2 orders of magnitude improvement in execution time over conventional electronic algorithms.

1. Introduction

Sorting is a basic, fundamental operation used for many symbolic, numeric, and artificial intelligence tasks. Some of the applications of sorting include the togetherness problem, i.e., the problem of bringing all identical items in a list or a file together. Sorting can also be used for matching problems, for example, if one is trying to find all matching entries of two files. If both files are first sorted, then all matching entries can be found in one pass through the data. A sort can also be used in database and knowledge-base processing. Sorting algorithms can serve as a basis for performing many common and highly useful operations such as selection, projection, division, and join in the context of relational databases,[1,2] and intersection, union, difference, and Cartesian product in the context of sets. Sorting can be used to simplify searching. In addition to its widespread use in information processing, sorting is also important in communications, in which it serves as the basis for packing routing in networks. Because of its importance, there has been a great deal of work on developing and analyzing sorting algorithms and architectures.[3–6] In general, a sort on a string of n data elements can be done with $O(n \log_2 n)$ comparisons by use of the best serial algorithms.[7] Using a conventional sorting network and taking advantage of parallelism as much as possible permits sorting to be done in $O(\log_2 n)$ time steps. Recently, a theoretical algorithm for electronic $O(1)$ sorting on a parallel processor was proposed.[8] However, the amount of hardware required for its implementation makes it cost prohibitive.

Optical architectures for sorting are worth developing for two reasons: (1) less conventional architectures may permit a sorting operation to be done in far fewer steps, and (2) the fastest conventional sorting networks seem to require fairly dense, globally connected networks that are difficult to implement with conventional electronic technology alone. An example of the latter is a sorting network based on Batcher's bitonic sort,[9] which for a string of 2^k data elements requires $[k(k+1)]/2$ stages. Each stage consists of fast-Fourier-transform-like butterfly interconnections of varying sizes. Even the above-mentioned $O(1)$ algorithm[6] relies on a three-dimensional reconfigurable mesh, which is also difficult to implement. Optical technology with its inherent parallelism and noninterfering communications is well suited for implementing the sorting operation because of its ability to process two-dimensional (2-D) data arrays in parallel. Optical interconnects also permit the efficient and high-speed implementation of the global connection patterns required for sorting algorithms. Most important, optical systems permit

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

Received 15 July 1994; revised manuscript received 8 December 1994.
0003-6935/95/173087-10$06.00/0.

the implementation of sorting with an execution time independent of the number of data elements to be sorted [i.e., $O(1)$ time]; this is in contrast to electronic sorting systems in which the execution time is usually some function of the number of data elements. In addition, optical sorting systems have minimum time skew and may communicate information at optical media bandwidths. Thus the sorting throughput can be quite large and is limited in practice by the response time of the optical active devices used.

This paper explores the problem of sorting and discusses the optical implementation of a highly parallel sorting algorithm that takes into account the unique properties of optics. Several optical sorting algorithms have been proposed in the past.[10–12] Stirk and Athale[11] proposed a parallel-pipelined sorting algorithm using optical compare-and-exchange modules that has a time complexity of $O(\log^2 n)$ steps. Researchers at IBM also proposed an optical enumeration sort[12] by use of an optical system based on phase addition and subtraction (interference) to perform analog algebraic operations. However, these coherent systems are difficult to align. Futhermore, the authors of these optical systems did not address the issue of physically reordering the data elements after their positions in the sorted output have been determined. This restricts the system to pointer-based computing systems. In this paper we propose an optical system capable of both determining the positions of the sorted data elements and physically reordering them in $O(1)$ time steps. It uses photonics for highly parallel interconnects and optoelectronics, in the form of smart pixels, for processing.[13–16] Thus it exploits the advantages of both the optical and the electrical domains. In its current form the system can be configured as a high-speed sorting engine for conventional computers or further developed to function as a stand-alone optical sorting processor.

The paper is organized as follows. Section 2 introduces the parallel sorting algorithm. Section 3 discusses the detailed optical implementation of the algorithm, the devices used, and related issues. Section 4 presents the system layout and estimates the performance that can be expected from the algorithm. We also estimate the power requirement of the system and discuss other related implementation issues. Section 5 concludes the paper.

2. Constant-Time Parallel Sorting Algorithm

Given the ability of optics to process 2-D arrays of data, parallel algorithms that are infeasible on electronic computers because of the requirement of a large number of processors gain new importance. As an example of a highly parallel algorithm, sorting an array of n numbers requires the comparison of every number to every other number. From this the rank (the position in the sorted output) of each data element is computed. In what follows we describe a sorting algorithm that implements exactly the above strategy in constant time [i.e., $O(1)$, independent of the number of words being sorted]. In Section 3 we present the optical implementation of each of the steps.

Before formalizing the algorithm, let us first discuss the conventions that will be used in this example and throughout the rest of the paper. Row vectors will be indicated by lowercase boldface letters, such as $\mathbf{x}$, for example, whereas an uppercase letter indicates a matrix. A subscript indicates the index of the vector or matrix. Thus x_j indicates the jth element of the row vector $\mathbf{x}$, and $A_{i,j}$ indicates the element in the ith row and the jth column of matrix A. On occasion, the notation x_i will be used to indicate the ith element of column vector $\mathbf{x}^T$, where the i illustrates a correspondence to a matrix row. Finally, 2-D data will be referred to as matrices in the context of algorithms and data arrays in the context of optics. This is done to adhere to both mathematical and optical conventions for representing 2-D arrays.

To sort an n-element vector, we must compare each element of the vector against every other element. We perform these comparisons in parallel by broadcasting the data vector $\mathbf{a} = [7\ 8\ 2\ 8\ 5]$ and $\mathbf{a}^T$ n times and performing one comparison operation for every resulting pair of elements. Optics performs this operation easily with a simple imaging system. For our example,

$$A = \begin{bmatrix} 7 & 8 & 2 & 8 & 5 \\ 7 & 8 & 2 & 8 & 5 \\ 7 & 8 & 2 & 8 & 5 \\ 7 & 8 & 2 & 8 & 5 \\ 7 & 8 & 2 & 8 & 5 \end{bmatrix}. \tag{1}$$

Each column of the resulting $n \times n$ data matrix A contains n copies of each data element a_j. In order to compare each column of A to every element of $\mathbf{a}$, we must first transpose $\mathbf{a}$ and then broadcast it horizontally:

$$A^T = \begin{bmatrix} 7 & 7 & 7 & 7 & 7 \\ 8 & 8 & 8 & 8 & 8 \\ 2 & 2 & 2 & 2 & 2 \\ 8 & 8 & 8 & 8 & 8 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}. \tag{2}$$

We perform the comparison operation between every pair of elements of $\mathbf{a}$ by subtracting the two matrices A and A^T, in which the subtraction is represented as the addition of a negative quantity:

$$D = A + (-A^T) = \begin{bmatrix} 0 & 1 & -5 & 1 & -2 \\ -1 & 0 & -6 & 0 & -3 \\ 5 & 6 & 0 & 6 & 3 \\ -1 & 0 & -6 & 0 & -3 \\ 2 & 3 & -3 & 3 & 0 \end{bmatrix}. \tag{3}$$

In the difference matrix D of Eq. (3) each column j represents the comparison of each element a_j with every other element $\mathbf{a}$. To compute the rank of each element for an ascending order sort, we need to sum the number of data elements that are numerically less than element a_j. If a_i represents a row element of $\mathbf{a}^T$ and thus A^T, then elements $D_{i,j}$, where $a_i < a_j$, contribute one to the rank of a_j. Similarly, negative values in D contribute zero to an element's rank because a_i should appear after a_j in the sorted array. Finally, zeros in D represent equivalent data elements. Each column of D contains at least one zero along the matrix diagonal that represents the comparison of a data element to itself. On the other hand, off-diagonal zeros represent nonunique data elements. For now, we let all zeros of D contribute one to a_j's rank. Thus we arrive at the following for the rank matrix R:

$$R = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}. \tag{4}$$

Summing each column j of R, we obtain the non-unique rank for the corresponding element a_j. Thus the ranks of the sorted elements are

7 rank:	3,
8 rank:	5,
2 rank:	1,
8 rank:	5,
5 rank:	2.

Notice that the multiple instances of the number 8 are assigned the same rank. To resolve the non-unique ranks, we compare the locations of the two elements being compared. In our example, the first occurrence of two nonunique numbers in the vector $\mathbf{a}$ will be considered the larger of the two. Hence for every (i, j), such that $i < j$ and $a_i = a_j$, we treat (a_i, a_j) as though $a_i > a_j$. Thus if $D_{i,j} = 0$ and $i < j$, then $D_{i,j}$ should still contribute to the rank of a_j and hence should remain nonnegative. To distinguish elements of D where $D_{i,j} = 0$ and $i > j$, we modify the below-diagonal elements. To do so, we create a new matrix U, where $U_{i,j} = 1$ if $i > j$, and is 0 otherwise:

$$U = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}. \tag{5}$$

Next, we subtract U from D, again by the addition of a negative quantity:

$$D' = D + (-U) = \begin{bmatrix} 0 & 1 & -5 & 1 & -2 \\ -2 & 0 & -6 & 0 & -3 \\ 4 & 5 & 0 & 6 & 3 \\ -2 & -1 & -7 & 0 & -3 \\ 1 & 1 & -4 & 2 & 0 \end{bmatrix}. \tag{6}$$

In this new difference matrix D' the negative elements remain negative and hence contribute nothing to an element's rank. The positive elements remain positive, or become 0, and still contribute one to the rank of an element. The zeros in the above-diagonal portion of the matrix D become negative in D' and no longer contribute to the rank while the other zeros are unaffected. This resolves the ties between nonunique ranks in the manner described above. The new rank matrix is then

$$R' = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}, \tag{7}$$

where $R'_{i,j} = 0$ indicates that a_i appears in the sorted output after a_j, and $R'_{i,j} = 1$ indicates that it does not. Summing each column of R', we get the fully resolved ranks of each element in $\mathbf{a}$:

7 rank:	3,
8 rank:	4,
2 rank:	1,
8 rank:	5,
5 rank:	2.

The result from the algorithm is the generation of the rank vector, $\mathbf{r} = [3\ 4\ 1\ 5\ 2]$, which contains the positions of each of the data elements in the sorted output. Although we demonstrate the algorithm for two identical elements only, it should be noted that it also works for more than two identical elements. This is because the conflict-resolution scheme relies only on a data element's position; as a result, additional identical elements will merely increase the rank of the higher-indexed elements. This algorithm and an accompanying optical system are complete when they are capable of rearranging the input data to the order reported in $\mathbf{r}$. The problem of physically reordering the data reduces to the task of determining, in parallel, which column of $\mathbf{r}$ contains the number 1, the number 2, etc., so that we know which element is first, second, etc., in the sorted output. We accomplish this by comparing each element of $\mathbf{r}$ to the numbers $[1, \ldots, n]$. Mathematically, this is illustrated by spreading the **r** vector n times and subtracting the vector $[1, \ldots, n]^T$, spread horizontally n times, as in the following:

$$S = \begin{bmatrix} 3 & 4 & 1 & 5 & 2 \\ 3 & 4 & 1 & 5 & 2 \\ 3 & 4 & 1 & 5 & 2 \\ 3 & 4 & 1 & 5 & 2 \\ 3 & 4 & 1 & 5 & 2 \end{bmatrix} - \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 \\ 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 3 & \mathbf{0} & 4 & 1 \\ 1 & 2 & -1 & 3 & \mathbf{0} \\ \mathbf{0} & 1 & -2 & 2 & -1 \\ -1 & \mathbf{0} & -3 & 1 & -2 \\ -2 & -1 & -4 & \mathbf{0} & -3 \end{bmatrix}. \qquad (8)$$

In the resulting matrix on the right-hand side of Eq. (8), $S_{i,j} = 0$, highlighted in boldface, indicates in which column j of **r** the value i exists. It tells us that a_j is to be relocated to row i of the sorted output, where we assume the sorted output as a column vector. $S_{i,j} = 0$ for only one element per row of S. If we use S to select or discard elements from a copy of the A matrix, such that elements $S_{i,j} = 0$ select elements $A_{i,j}$ and elements $S_{i,j} \neq 0$ discard elements $A_{i,j}$, we have effectively rearranged the inputs to their positions in the sorted output. This operation is illustrated below, where the selected elements are highlighted in boldface:

$$A = \begin{bmatrix} 7 & 8 & \mathbf{2} & 8 & 5 \\ 7 & 8 & 2 & 8 & \mathbf{5} \\ \mathbf{7} & 8 & 2 & 8 & 5 \\ 7 & \mathbf{8} & 2 & 8 & 5 \\ 7 & 8 & 2 & \mathbf{8} & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 \\ 5 \\ 7 \\ 8 \\ 8 \end{bmatrix}. \qquad (9)$$

Thus the problem of reordering the data reduces to selection of the appropriate element from each row of A because each row of A has a copy of each data element and discards the rest. The discrete steps for constant time sorting are summarized below:

Step 1. Process the input:

(a) Generate matrix A by vertically spreading **a** n times.
(b) Generate matrix A^T by horizontally spreading $\mathbf{a}^T$ n times.

Step 2. Compare every element of **a** with every element of $\mathbf{a}^T$ by computing the difference matrix $D = A + (-A^T)$.
Step 3. Generate the U matrix, where $U_{i,j} = 1$ iff $i > j$.
Step 4. Resolve nonunique ranks by computing matrix $D' = D + (-U)$.
Step 5. Generate R' by thresholding D', where $R_{i,j} = 1$ iff $D'_{i,j} \geq 0$.
Step 6. Form the rank vector, **r**, by summing each column of the matrix R'.
Step 7. Reorder the sorted data:

(a) Compare every element of **r** to every element of $[1, \ldots, n]^T$ by expanding both by n and subtracting the latter from the former to form the S matrix.
(b) Use S to select or discard A, where $S_{i,j} = 0$ indicates that data element $A_{i,j}$ should be transferred to row i in the sorted output.

In what follows we present an efficient and economical optical implementation of the proposed algorithm.

3. Optical Implementation of the Constant-Time Parallel Sorting Algorithm

We will now consider an optical system that implements the above steps and physically reorders the input data in constant time. The system contains a mixture of optics and electronics. Photonics are used for highly parallel, noninterfering interconnects to provide the massive connectivity required by the algorithm. Electronics, integrated into arrays of smart pixels, perform the algebraic operations.

The input to the system is a one-dimensional (1-D) array of electronic data to be sorted. To provide the optical data planes, vertical-cavity surface-emitting lasers[17] (VCSEL's) are used both within the smart pixels and as the system input. Because VCSEL's can be integrated in densities of $>10^6$ microlasers on a single chip, their development is important in the realization of high-density optically interconnected systems. The output of the system is a 1-D array of electronic sorted data. Because both the input and the output are electrical, the system can be configured as a high-speed sorting engine for conventional computers. In the rest of this section we present the optical implementation of this sorting unit. But first, we describe the scheme used for subtracting two optical analog values.

A. Optical Algebraic Operations

In order to implement steps 2, 4, and 7 of the algorithm, we must have a way of subtracting two numbers. Because the intensity of light cannot be less than zero, we use a dual-channel scheme for representing both positive and negative results. For ease of explanation we assume that we are sorting positive values only. Negative values will then occur only as the subtraction result of two positive numbers. The system can be easily modified slightly to accommodate negative data elements.

Positive data numbers are represented by the light-intensity level from a VCSEL array, VC1, and their negated values are represented by an equivalent light-intensity level from a second VCSEL array, VC2. Thus a number is considered negative merely by the fact that it is generated by VC2. The values from the VCSEL's propagate through the system as two side-by-side channels. Because photons do not interact in free space, the actual subtraction of the absolute values generated by VC1 and VC2 is performed electronically in a smart-pixel array. The light from VC1 impinges upon a photodetector while the light from VC2 impinges upon a second photodetector. The positive photodetector outputs, V_1 and V_2, where the notation V_y represents the photodetected voltage corresponding to the light level of channel y, are then fed into the positive and the negative terminals, respectively, of an operational amplifier (op-amp). The op-amp subtracts the absolute values, $|V_1| - |V_2|$, to fulfill step 7.

B. Generating the Rank Vector **r**

1. *Implementation of Step 1 of the Algorithm*

Figure 1 illustrates an optical implementation for the first two steps of the algorithm. In step 1 the 1-D input, **a**, modulates the columns of 2-D laser array VC1 to form the A array. Meanwhile, $\mathbf{a}^T$ modulates the rows of a 2-D laser array VC2 to form the $-A^T$ array (the minus sign is inherent in the use of VC2). Thus the data duplication step is actually performed before the optical generation of the data. This differs slightly from our algorithm, which suggested that the two vectors be vertically and horizontally spread n times with optics. The optical broadcasting of a 1-D array into a 2-D array reduces the optical power of each data element by n. This has a direct consequence in the execution speed of the algorithm. We also avoided beam spreading because the distance between analog levels is reduced by a factor of n. This reduction makes it more difficult to distinguish the analog levels at the detectors. Furthermore, the beam spreading results in cross talk among the two channels.

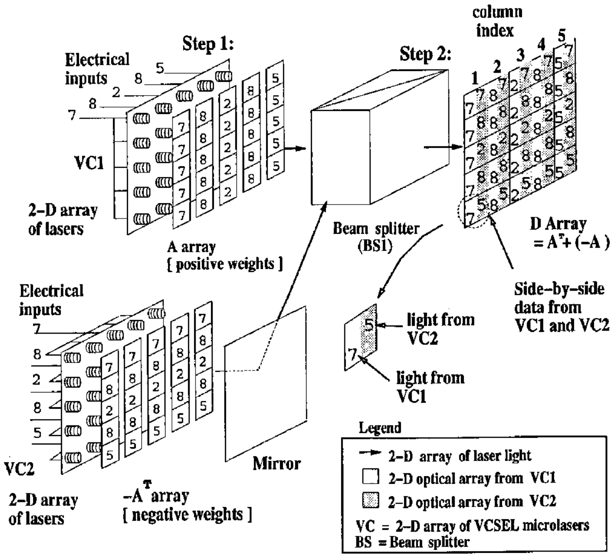

Fig. 1. Optical hardware for implementing steps 1 and 2 of the algorithm. A and $-A^T$ are formed by modulation of 2-D laser arrays by **a** and $\mathbf{a}^T$, respectively. These are summed to form the difference matrix D.

2. *Implementation of Step 2 of the Algorithm*

The difference array D of step 2 is formed by the sum of arrays A and $-A^T$. This is performed optically by merging and interlacing of the optical data planes so that corresponding values are side by side. Recall that at this stage the beams are merely propagated together; the actual subtraction will be performed later. In Fig. 1 each element of the D array contains two numbers that represent the interlacing of the two values. The number in the upper-right-hand corner represents the intensity level of the negative light component, and the number in the lower-left-hand corner represents the positive light component.

3. *Implementation of Steps 3, 4, 5, and 6 of the Algorithm*

Figure 2 illustrates the implementation of step 3 and part of step 4. The $-U$ array of step 3 is formed by modulation of a third VCSEL (not shown). The summation of D and $-U$ in step 4 is performed by the beam splitter BS2 in Fig. 2. Figure 3 illustrates, for a single pixel, the subtraction of the absolute values in D'. Notice the integration of the photodetectors, the modulation electronics, and the surface-emitting laser in this close-up view of a single smart pixel. The two light components from VC1 and VC2 within a pixel of the D' array impinge upon the photodetectors of the smart-pixel array. The op-amp subtracts the photodetected values. The output is thresholded by a complementary metal-oxide semiconductor gate (not shown). The digital output from the thresholding operation of step 5 then modulates the surface-emitting laser for communication to the next stage.

Figure 4 illustrates steps 4, 5, and 6 on a full scale, in which the D' array is being viewed from behind. The output of the electronic subtraction and thresholding of D' by smart-pixel array SP1 modulates the surface-emitting lasers to generate the R' array. Because the lasers are integrated on the same side of the substrate as the photodetectors, the R' array propagates back into the system and passes through half-wave plate HWP1. HWP1 rotates the polarization of the light from the smart-pixel array so that it will be entirely reflected from the polarizing beam

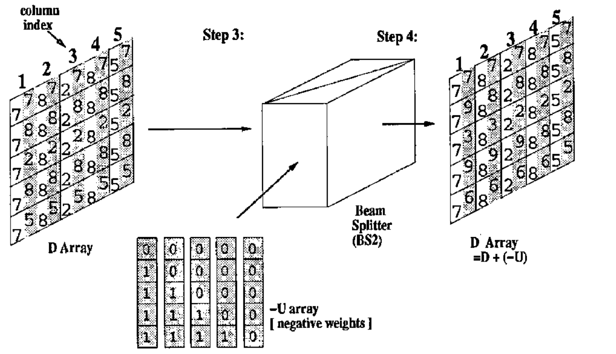

Fig. 2. Optical implementation of step 3 and part of step 4 of the proposed algorithm. To resolve nonunique ranks, we add U to D to form D'; the actual subtraction is performed in the next stage, shown in Fig. 3.

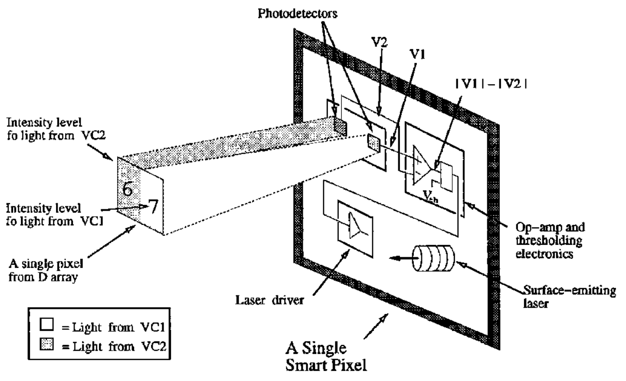
Photodetectors
V2
V1
|V1|-|V2|
Intensity level fo light from VC2
Intensity level fo light from VC1
A single pixel from D array
Op-amp and thresholding electronics
Surface-emitting laser
Laser driver
□ = Light from VC1
▨ = Light from VC2
A Single Smart Pixel

Fig. 3. Optical implementation of the actual subtraction from step 4 and all of step 5. The light components from VC1 and VC2 impinge upon their corresponding photodetectors in the smart pixel to generate V_1 and V_2. The op-amp performs the subtraction $|V_1| - |V_2|$. The output is thresholded, the result of which modulates the laser driver.

splitter PBS1. The polarizing beam splitter reduces the power loss of the system and also prevents backward propagation of light from SP1. The half-wave plate can be eliminated from the system if the lasers on SP1 are orthogonally polarized with respect to VC1 and VC2. The cylindrical lens vertically sums the ones in the R' matrix to form the rank vector, **r**, in accordance with step 6.

Next, we need to reorder the input data according to the rank vector. This additional step may not be required for all applications. For example, if the data elements are sorted in a content-addressable memory, then any data element can be recalled directly by its rank, and if this is what is needed, then physical reordering is not necessary.

C. Physical Reording of the Input Data

In addition to the generation of the rank vector, an equally important aspect of this paper is the physical reordering of the input. This has received little attention previously. The optical system in Fig. 5 performs the final step of the algorithm. Here, we use a second smart-pixel array to select the appropriate element from each row of the A array. The labeling conventions of the A array are reversed in the figure because it is being viewed from behind.

As shown in Fig. 5, each pixel of SP2 consists of a photodetector, comparison logic, laser driver electronics, and a surface-emitting laser. The photodetector receives the optical power from each pixel of the A array. The comparison logic selects the appropriate element from each row of the A array, as outlined in Eq. (8). We spread the **r** vector and the array $[1\ 2\ 3\ 4\ 5]^T$ vertically and horizontally by writing them to the column- and the row-addressing lines of SP2, respectively. The array $[1\ 2\ 3\ 4\ 5]^T$ can be easily implemented by a resistive network integrated onto the device substrate because its values remain the same for each sort unless the order of the sort (ascending or descending) is changed. The **r** vector is imaged onto an integrated 1-D photodetector array that is internally connected to the column-addressing lines. If the two input signals to the comparison logic are identical, corresponding to the condition $S_{i,j} = 0$ in Eq. (8), then the output enables the laser driver so that the optical intensity level detected at the photodetector can be regenerated by the surface-emitting laser. If the two signals are not identical,

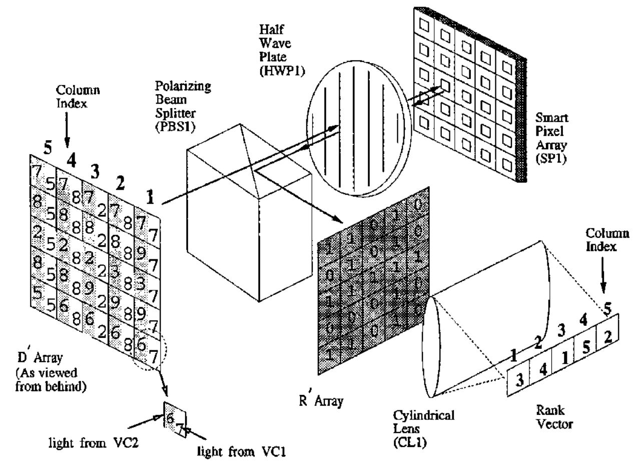

Fig. 4. Optical system for implementing the actual subtraction phase of step 4 along with steps 5 and 6 on a full scale. Each pixel of the D' array is imaged onto a photodetector of the smart-pixel array. The subtraction and the thresholding of steps 4 and 5 are performed by the integrated electronics in the smart-pixel array. The surface-emitting laser writes the result to the R' array, which then reflects off the beam splitter and is then vertically summed by the cylindrical lens to form the rank vector, **r**, in step 6. The labeling conventions on the D' array in this figure have been changed to mirror the fact that we are now viewing it from behind.

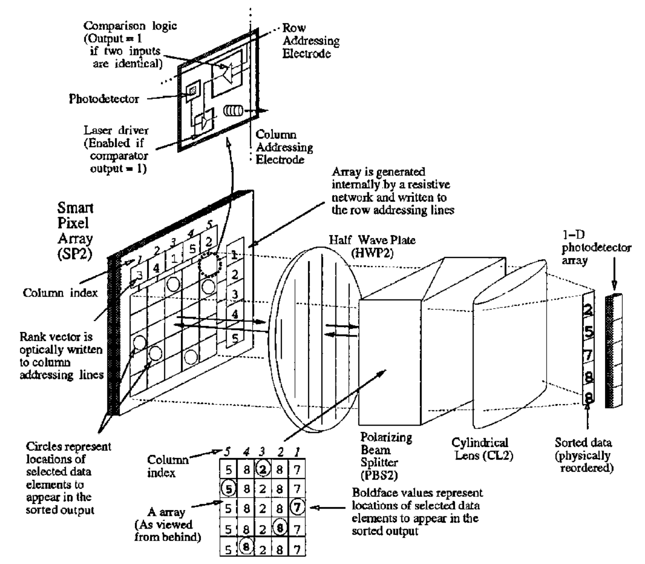
Comparison logic (Output = 1 if two inputs are identical)
Row Addressing Electrode
Photodetector
Laser driver (Enabled if comparator output = 1)
Column Addressing Electrode
Array is generated internally by a resistive network and written to the row addressing lines
Smart Pixel Array (SP2)
Half Wave Plate (HWP2)
1-D photodetector array
Column index
Rank vector is optically written to column addressing lines
Circles represent locations of selected data elements to appear in the sorted output
Column index
A array (As viewed from behind)
Polarizing Beam Splitter (PBS2)
Cylindrical Lens (CL2)
Sorted data (physically reordered)
Boldface values represent locations of selected data elements to appear in the sorted output

Fig. 5. Optical setup for physically reordering the sorted input. The A array reflects off the polarizing beam splitter and is imaged onto the smart-pixel array (SP2). If a smart pixel has identical signals on the column- and row-addressing lines, the optical intensity level detected by the photodetector is regenerated by the surface-emitting laser. On the other hand, if the two signals are different, the smart pixel will generate no light. Therefore only selected elements of A reflect off SP2 and are focused on a column, which is the desired sorted data.

i.e., $S_{i,j} \neq 0$, the output of the comparison logic disables the laser driver so that no light is generated. The selected elements of A are focused to a vertical line at the focal plane of cylindrical lens CL2. Thus we have effectively demonstrated the implementation of Eq. (9), the physical reordering of the sorted data.

4. System Layout and Performance Analysis

A. System Layout

In Fig. 6 we present the physical layout of the proposed optical sorting system. Two vertical-cavity surface-emitting lasers (VCSEL's), VC1 and VC2, provide the 2-D optical input data, as mentioned in Subsection 3.B. In order to reduce the amount of hardware required (a VCSEL and beam splitter BS2) and also reducing the power loss due to beam splitting, we generate both the $-A^T$ and the $-U$ arrays by VC2. Because U is an array of constants, it can easily be added to the A^T array as one of the inputs to the VCSEL's laser drivers. Thus VC2 provides the array $-(A^T + U)$ to be added to A to form the D' array. Two copies of the D' array are generated by beam splitter BS1. One copy is imaged onto SP1 by polarizing beam splitter PBS1 for the generation of the R' array. The rank vector is formed by CL1 and imaged onto the column-addressing photodetectors of SP2. In order to preserve the parity of the array, we can implement this 90° deflection with a pentaprism (not shown) instead of a mirror. The other copy of D' is imaged onto the pixels of SP2 by PBS2 for the reordering step. Because this step requires the A array, a slit filter removes the $(A^T + U)$ elements from the D' array, leaving behind only the A array. After SP2 selects the appropriate elements of A, the sorted data is imaged onto the 1-D detector array, DET.

The layout in Fig. 6 also suggests that the system volume has the potential to be very small. Using 1-in. (2.54-cm) aperture optics (lenses, beam splitter, and half-wave plates) with a modest 1.5-in. (3.8-cm) component separation, we see that the entire system can fit in an area of roughly 20 cm × 20 cm. Although the above measurement does not include the interface electronics or the heat-removal components, the complete optical system should be small enough to be manufactured as an add-on unit for mainframe computers.

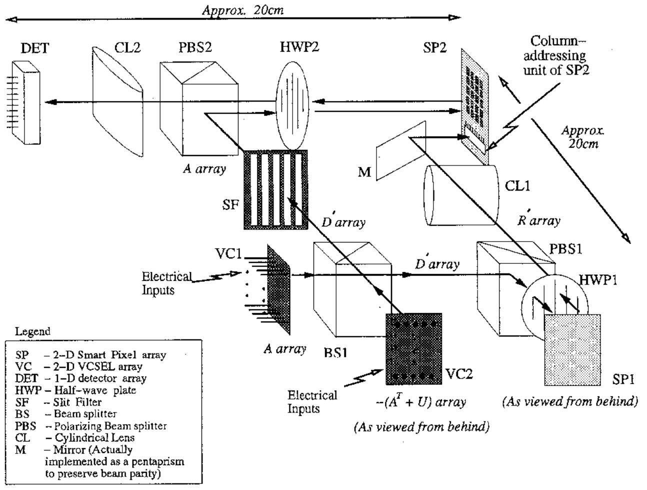

Fig. 6. Layout of the proposed optical sorting system.

B. Execution Speed and Power Requirement

In this section we estimate the system's execution speed and power requirements. For the maximum execution speed we first calculate the maximum rate the detectors can be operated because this parameter will limit the cycle time. The detectors' operating speeds are dependent on the optical power available to them. With current technology a single VCSEL can provide 3 mW of optical power.[18] Because the proposed optical sorting system is analog, the 3-mW value will correspond to the largest analog intensity level that the system can represent. As shown in Fig. 6, the data from VC1 and VC2 are partially reflected or transmitted by BS1 and imaged onto both smart-pixel arrays. We assume that all glass optics result in negligible loss of power because special coatings can be used. Thus the maximum optical power, P_{in}, incident upon each of SP1's (or SP2's) detectors is 1.5 mW. This optical power needs to charge each detector to an energy level, E, of 10 pJ.[19] The energy, E, is linearly proportional to the incident optical power P_{in} by the expression $E = P_{\text{in}}\Delta t$, where Δt is the integration time of the detector. Smaller analog levels will require the same integration time because both E and P_{in} will be reduced proportionally. With the values for P_{in} and E listed above, a single detector of a smart-pixel array will require an integration time of 13.33 ns. Thus it can be operated at ~ 75 MHz for current technology.

For a complete single sorting operation the input optical signal from VC1 and VC2 must propagate through BS1 and PBS1, be modulated by SP1 and SP2, and finally be detected by DET. Thus the overall cycle time to complete the sort, T_{cycle}, becomes the sum of the times through each of these devices. Assuming the propagation delay of optical passive devices such as lenses, beam splitters, and mirrors is negligible, T_{cycle} becomes

$$T_{\text{cycle}} = T_v + T_{\text{sp1}} + T_{\text{sp2}} + T_{\text{det}}, \tag{10}$$

where T_v represents the modulation time of the VCSEL array, T_{sp} represents the response time of the smart-pixel array, and T_{det} represents the response time of the 1-D detector array. T_{sp} can be further decomposed into the sum $T_v + T_e + T_d$, where T_v represents the modulation time of the smart pixel's VCSEL's, T_e represents the response time of the internal electronics, and T_d represents the response time of the smart pixel's detectors. Therefore

$$\begin{aligned} T_{\text{cycle}} &= T_v + (2T_v + 2T_e + 2T_d) + T_{\text{det}} \\ &= 3T_v + 2T_e + 2T_d + T_{\text{det}}, \end{aligned} \tag{11}$$

where $T_v = 100$ ps, $T_e = 10$ ns,[16] and $T_d = T_{\text{det}} = 13.3$ ns, as previously calculated for current technology. Thus T_{cycle} becomes 60.2 ns. If the rank-vector determination is separated from the physical reordering step, which is common[5] and can easily be accomplished with pipelining, Eq. (11) can be reduced to $2T_v + T_e + 2T_d$, resulting in a pipelined cycle time of 36.8 ns.

For the optical power requirement of the system, several factors must be considered. First, the majority of the power requirement will be related to the output of the VCSEL's. Second, the 3-mW output level from each VCSEL corresponds to the maximum analog level that the system can represent. Thus the power requirement theoretically varies with the input data set. In our power estimate we evaluate the worst case by assuming that each VCSEL is required to source 3 mW. Keep in mind that, on the average, actual values should be lower. We also assume that the power requirement of SP1 is substantially less than the other arrays because it has to output only a single level instead of an entire analog range. With the major power contributions coming from VC1, VC2, and SP2, approximately 9 mW of optical power is needed for each channel. For a currently available array size, such as 8×8, the total optical power requirement is ~ 600 mW. As array sizes increase, this power requirement will also grow quite rapidly because it is a function of n^2. We anticipate that the future will bring not only larger array sizes but more power-efficient devices, such as lower-power detectors.

C. Comparison to Electronic-Based Sorting Systems

In addition to the execution time an equally important parameter is the bit capacity of the system, i.e., the word size. Many factors affect the word size. Once such factor is the uniformity of the VCSEL and detector arrays. To ensure that the same value applied to two VCSEL's results in the same output level, one should ensure little variation in the operating characteristics among the individual VCSEL's in any one array. A similar argument holds for the detector arrays. Variations that occur between two separate arrays can potentially be compensated by adjustment of the biasing levels for the device. This issue will of course become more acute as the array size grows. However, we believe that the maturity of semiconductor processing technology should help greatly in minimizing device variations.

Another factor that strongly affects the word size is the dynamic range of the detectors. Currently, light intensity may be produced, controlled, and detected in ~ 128 discrete levels.[20-23] This implies that the sorter has the capability of a 7-bit microprocessor. Obviously this limits the word length of the data that can be sorted in constant time. The number of data elements to be sorted at once is not affected by this limitation. However, this same restriction is also imposed upon electronic sorting systems. For a string sort a typical string might have between 10 and 20 characters in it. Because both optical and electronic sorters can operate on only a subset of this string at any time, a single character for instance, the sorting time is inevitably related linearly to the number of characters. In addition to the word-length restriction it is also prohibitive to build a sorting system that is capable of sorting a sequence length of 10,000 data items without iteration, in which a sequence length is the number of data items to be sorted. In general, the sequence length, m, will be much larger than the number of items, n, that the sorter can hold at any time. Thus the data set will have to be divided into m/n smaller data subsets of sequence length n, which will then be sorted separately and compared with each other during multiple passes through the data set. At first glance it would seem as though one needs to sort each data subset against itself and then against each other subsets. This approach would extend the execution time to $O(m^2/n^2)$ cycles.

Fortunately there are algorithms that can reduce the number of passes through the data. One of the fastest techniques involves a preprocessing step to arrange the data into groups called buckets. These buckets are chosen to match the capacity of the system as closely as possible. Ideally, m/n buckets are chosen and all data items that fall between adjacent limits are placed in the corresponding bucket. As an example of a simple bucket algorithm, consider limits that are dictated by alphabetical order. For instance, all data items beginning with the letter A are placed in the first bucket, all data items beginning with the letter B are placed in a second bucket, etc. This is easily accomplished if the preprocessing step scans the first letter of multiple data items in parallel and transfers the data to the appropriate bucket on the fly. The data within the buckets are unsorted even though the buckets are ordered relative to each other. The preprocessing step can be run on the host computer and overlapped with the execution time of the optical sorting system. Also, as the capacity of the system increases, fewer buckets will be needed and the preprocessing step will execute in less time. With fewer buckets and the potential to overlap the execution time the bucket preprocessing step can substantially improve the overall execution time of the algorithm. After that, all we have to do is internally sort each of the buckets, and then the entire data set will be sorted automatically. Thus the number of steps has been reduced from $O(m^2/n^2)$ to $O(m/n)$.

In practice the latter time complexity must be multiplied by a small constant number of iterations. This occurs when a few of the buckets overflow. A bucket overflows when the number of data items assigned to it exceeds the capacity of the system. For instance, owing to the prevalence of the letter A, more than n data items may appear in the A bucket, whereas very few may appear in the Z bucket. This means that an additional pass may be necessary in order to sort the A bucket. For systems with a large capacity n, the constant factor is ~ 3.[6,24]

A comparison of the proposed optical sorting system was then made based on the execution time of an electronic hardware-based sorter and the Quicksort benchmarks. Quicksort[24,5,6] is a software sorting algorithm that is run on machines to evaluate their sorting capability. The electronic sorter, called the biway sorter, is based on a bidirectional systolic array implemented with 3-μm complementary metal-oxide semiconductor technology, and, to the authors' knowledge, is the fastest physically demonstrated implementation. The biway sorter is 10 bits wide, has a capacity, n, of 20 data items, and an execution time of $O(m)$, where m is the sequence length.

Figure 7 illustrates the execution times of the Quicksort algorithm, the biway sorter, and the proposed optical sorting system versus the sequence length. Line 1 on the graph illustrates the execution time of the Quicksort algorithm running on a Burroughs B6700 computer with 0.5 MIPS (million instructions per second) capacity.[25] For comparison with the Quicksort benchmark the biway sorter was also run at 0.25 MHz, although it has actually been demonstrated at 9 MHz. Its execution time is represented by line 2 in the graph. For the proposed optical system we also calculated the execution time of the system running at 0.25 MHz with a modest smart-pixel array size of 8×8, and we display the results as line 3. To be consistent with the comparison in Ref. 6, we assumed a bucket preprocessing factor of 3, and the physical relocation of the data set was ignored. From these three lines we see that,

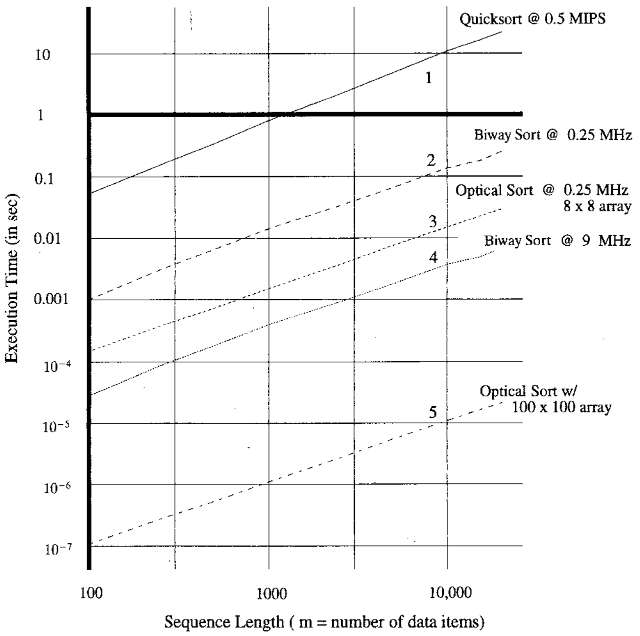

Fig. 7. Execution time versus sequence length for the Quicksort algorithm (line 1), the electronic biway sorter (line 2), and the proposed optical sorting algorithm (line 3). Line 4 illustrates the performance of the biway sorter extended to its current clock rate of 9 MHz, and line 5 illustrates the expected future performance of the optical system with a 100×100 smart-pixel array.

under the same test conditions of low clock rate and even a modest array size, the proposed optical system was almost an order of magnitude faster than the electronic biway sorter and almost 3 orders of magnitude faster than the Quicksort benchmark.

We then expanded the comparison in an attempt to predict the performance that can be expected in the future. For this the clock rate of the biway sorter was extended to its demonstrated 9 MHz, and its performance was redisplayed as line 4 of the graph. The clock period of the optical system was then set to the 36.8-ns value calculated earlier in this section for current technology. Although they have not been currently realized, it is suggested[26] that we can reasonably expect smart-pixel arrays in sizes of 100×100 in the future. With this future capability we illustrate in line 5 that the proposed optical sorting system has the potential to be at least 2 orders of magnitude faster than the biway sorter and between 5 to 6 orders of magnitude faster than the Quicksort algorithm. Obviously, electronic technology has progressed far enough that circuits can run at much faster than 9 MHz. This is not an issue for our system because our speedup is due to the parallelism exploited in the constant-time algorithm and not the clock rate. For similar clock rates the optical system will always deliver a performance improvement of at least n. Major future improvements in the optical system will come from advancements in smart-pixel integration densities rather than increases in clock rate.

5. Conclusion

Advances in the use of optics in computing have opened up new possibilities in several fields related to high-performance computing, high-speed communications, and parallel algorithm design. It is necessary to take into consideration the specific properties of optics, such as massive parallelism and global interconnects, to design algorithms that execute faster.

Sorting is a fundamental operation that has important implications in many areas. In this paper we presented a parallel sorting algorithm and its efficient optical implementation using currently emerging technology. The algorithm sorts n data elements in constant time, i.e., independent of the number of words being sorted. The proposed optical system is capable of both generating the rank vector and physically reordering the sorted data. Previously proposed constant-time optical sorting systems proceeded only as far as generating the rank vector, which is limited to being used as a pointer. Thus the data were not physically rearranged, as desired in many applications.

Our novel system uses smart-pixel arrays for optical data processing. Obviously, continued developments in the integration densities of smart-pixel arrays are needed for the practical realization of constant-time sorting. Nonetheless, the algorithm and its optical implementation presented in this paper are excellent examples of what optics can achieve. With the continued maturing of smart pixels, it is anticipated that we can sort 100 data items in <40 ns. This exhibits a quantum leap (~ 2 orders of magnitude) over its electronic counterparts in execution time. We expect that such an algorithm will have a major effect on sorting applications in the future.

References

1. A. Louri and J. A. Hatch, Jr., "Optical content-addressable parallel processor for high-speed database processing," Appl. Opt. **33,** 8153–8163 (1994).
2. L. Raschid, T. Fei, H. Lam, and S. Y. W. Su, "A special-function unit for sorting and sort-based database operations," IEEE Trans. Comput. **C-35,** 1071–1077 (1986).
3. S. G. Akl and H. Schmeck, "Systolic sorting in a sequential input/output environment," Parallel Comput. **3,** 11–23 (1986).
4. L. E. Winslow and Y. C. Chow, "The analysis and design of some new sorting machines," IEEE Trans. Comput. **C-32,** 677–683 (1983).
5. Y. Yasuura, N. Tagaki, and S. Yajima, "The parallel enumerations sorting scheme for VLSI," IEEE Trans. Comput. **C-31,** 1192–1201 (1982).
6. G. Orton, L. Peppard, and S. Akl, "Biway sorter: a two-dimensional systolic array," IEE Proc. E **139,** 147–155 (1992).
7. F. T. Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays, Trees, Hypercubes* (Morgan Kaufmann, San Mateo, Calif., 1992).
8. Y. C. Chen and W. T. Chen, "Constant time sorting on reconfigurable meshes," IEEE Trans. Comput. **43,** 749–751 (1994).
9. K. E. Batcher, "Sorting networks and their applications," in *Proceedings, 1968 Spring Joint Computer Conference* (American Federation of Information Processing Societies Press, Reston, Va., 1968), Vol. 32, pp. 307–314.
10. Y. B. Karasik, "Fast sorting algorithm based on the massive parallelism of optical computing," in *High Performance Computing. II. Proceedings of the Second Symposium,* M. Durand and F. E. Dabaghi, eds. (Elsevier, New York, 1991), Vol. 6, pp. 83–89.
11. C. W. Stirk and R. A. Athale, "Sorting with optical compare-and-exchange modules," Appl. Opt. **27,** 1721–1726 (1988).
12. K. S. Huang, L. Liu, and J. K. Peir, "Optical facility for parallel enumeration sort," IBM Tech. Discl. Bull. **33**(3A), 16–19 (1990).
13. H. S. Hinton, "Architectural considerations for photonic switching networks," IEEE J. Select Areas Commun. **SAC-6,** 1209–1226 (1988).
14. T. J. Cloonan, G. W. Richards, A. L. Lentine, F. B. McCormick, Jr., H. S. Hinton, and S. J. Hinterlong, "A complexity analysis of smart pixel switching nodes for photonic extended generalized shuffle switching networks," IEEE J. Quantum Electron. 619–633 (1993).
15. J. Cheng, P. Zhou, S. Z. Sun, S. Hersee, D. R. Myers, J. Zolper, and G. A. Vawter, "Surface-emitting laser-based smart pixels for two-dimensional optical logic and reconfigurable optical interconnections," IEEE J. Quantum Electron. **29,** 741–756 (1993).
16. J. Cheng and P. Zhou, "Smart pixels for two-dimensional arrays," IEEE Circuits Devices **9**(2), 19–27 (1993).
17. J. L. Jewell, Y. H. Lee, A. Scherer, S. L. McCall, N. A. Olsson, J. P. Harbison, and L. T. Florez, "Surface-emitting microlasers for photonic switching and interchip connections," Opt. Eng. **29,** 210–214 (1990).
18. J. Jewell and G. Olbright, "Surface-emitting lasers emerge from the laboratory," Laser Focus World, **28,** 217–223 (1992).
19. S. Yu and S. R. Forrest, "Implementations of smart pixels for optoelectronic processors and interconnection systems. I. Optoelectronic gate technology," J. Lightwave Technol. **11,** 1659–1669 (1993).
20. D. Psaltis, "Input/output devices," in *Optical Computing,* J. A. Neff, ed., Proc. Soc. Photo-Opt. Instrum. Eng. **456,** 66–71 (1984).
21. D. Psaltis and R. A. Athale, "High accuracy computation with linear analog optical systems: a critical study," Appl. Opt. **25,** 3071–3077 (1986).
22. E. Pochapsky and D. Casasent, "Linear acousto-optic heterodyning processors for complex-valued data processing," in *Digital Optical Computing,* R. Arrathoon, ed., Proc. Soc. Photo-Opt. Instrum. Eng. **752,** 155–171 (1987).
23. R. Arrathoon, *Optical Computing, Digital and Symbolic* (Dekker, New York, 1989).
24. D. Knuth, *The Art of Computer Programming. III. Sorting and Searching* (Addison-Wesley, Reading, Mass., 1973).
25. S. G. Akl and H. Meijer, "Recent advances in hybrid sorting algorithms," Utilicas Math. **21C,** 325–343 (1982).
26. P. A. Mitkas, L. J. Irakliotis, F. R. Beyette, Jr., S. A. Feld, and C. W. Wilmsen, "Optoelectronic data filter for selection and projection," Appl. Opt. **33,** 1345–1353 (1994).