Design of an Optical Content-Addressable Parallel Processor with Applications to Fast Searching and Information Retrieval *

Ahmed Louri
Department of Electrical and Computer Engineering
University of Arizona
Tucson, Arizona 85721

Associative processing based on content-addressable memories has been argued to be the natural solution for non-numerical information processing applications. Unfortunately, the implementation requirements of these architectures using conventional electronic technology have been very cost prohibitive, and therefore associative processors have not been realized. Optics has the capability over electronics for directly supporting associative processing by providing economic and efficient interconnects, massive parallelism, and high-speed processing. This paper presents the principles of designing an optical content-addressable parallel processor called OCAPP, for the efficient support of parallel symbolic computing. The architecture is designed to fully exploit optics advantages in interconnects and high-speed operations, and is potentially very suitable for applications where the number of data sets to be operated on is high. Several search and retrieval algorithms are mapped onto OCAPP to illustrate its ability to support parallel symbolic computing.

1 Introduction

Searching, retrieving, sorting and modifying symbolic data can be significantly improved by the use of content-addressable memory instead of location-addressability. In a content-addressable memory data is addressed by its contents[1]. An associative processor is a parallel processing machine in which the data items are content-addressable with the added capability to write in parallel into words satisfying certain criterion. A well known model of a basic associative architecture[2] which can provide parallel search and update consists of a data memory of n words with m bits per word, word select logic, bit-slice select logic, a control memory for storing programs and control, a response register, multiple match resolver, an output register, and some auxiliary circuits for control. Each cell in the memory contains storage for one data bit and comparison logic hardware. The comparand register, C, is used to hold the key operand being searched for. The mask register, M, is used to enable or disable the bit-slices to be involved in the parallel comparison operations across all the words of the memory array. An interrogating signal (a combination of the comparand register and the mask register) is broadcast to all the cells for comparison. All bit cells perform a comparison between the broadcast signal and their values. The response register R identifies the matching words. The multiple response resolver is used to select one (it may be the first) of the matching words. Although the model is straighforward, it has not been largely used because of the difficulty and high cost of implementing it in conventional electronic technology. This paper argues that optics is, potentially, the ideal medium to implement parallel processors based on the associative model of computing.

Optical systems hold the promise for providing efficient support for future parallel processing systems[3]. These include inherent parallelism, high spatial and temporal bandwidths, and non-interfering communications. For associative processing, optics may be the ideal solution to the fundamental problems faced by electronic implementations, namely cell complexity, interconnects latency, difficulty of implementing information broadcasting and parallel access to the stored data. Optics can alleviate the cell complexity by migrating the implementation of wiring and logic into free-space. The multi-dimensional nature of optical systems allows for data storage and logic to be performed on two-dimensional planes while the third dimension can be used for interconnects. The high degree of connectivity available in free-space space-invariant optical systems (10^6 to 10^8), and 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 interconnects problems[4]. Moreover, optical and electro-optical systems can offer a considerable storage capacity and parallel access than do pure electronic systems.

* This research was supported by an NSF Grant No. MIP - 8909216.

2 Optical Content-Addressable Parallel Processor: OCAPP

In figure 1 an organizational structure for an optical content-addressable parallel processor called OCAPP is proposed. The architecture is organized in a modular fashion, and consists of a *selection unit*, a *match/compare unit*, a *response unit*, an *output unit*, and a *control unit*. The architecture is developed to meet four goals, namely: (1) exploitation of maximum parallelism; (2) amenability to optical implementation with existing devices; (3) modular design in that it can be scalable to bigger problems; and (4) ability to efficiently implement information retrieval, and symbolic computations.

The selection unit is comprised of (1) a storage array of n words, each m bits long (in actuality, the storage array capacity is $n \times 2m$, since each bit position is comprised of a true bit w_{ij} and its complement $\bar{w_{ij}}$); and (2) word and bit-slice enable logic to enable/disable the words and/or the bit-slices that participate in the match operation, and reset the rest. The match/compare unit (shown in fig.2) contains a (1) $1 \times m$ interrogation register I; (2) logic hardware to perform parallel bitwise comparison between the bits of the interrogation register and the enabled bits of the storage array; (3) two $n \times 1$ working registers, G and L, which are used for magnitude comparisons (to be explained later); (4) a $n \times 1$ response register R for displaying the result of the comparison; and (5) a single indicator bit called the match detector MD, which indicates whether or not there is any matching words. This unit allows comparison of a single operand stored in the interrogation register and the words stored in the storage array. The I register is a combination of the comparand register C and the mask register M as shown in table 1. As such, it holds the operand (depending on masking information if any) being searched for or being compared with.

The response unit is responsible for selecting one or several matching words. It comprises several scratch-pad registers and a priority circuit for selecting the first matching word. Depending on program control, the output of the response unit is routed either to the output unit for outputting the result or fed back to the selection unit for further processing of the matching words. All units are under the supervision of a conventional control unit with conventional storage (eg., a local RAM) which stores the program instruction. Its role is to load/unload the storage array, set/rest various registers such as the I, R, G and L of the match/compare unit, enable/disable memory words, perform conditional instructions, monitor the MD bit, and test program termination. In what follows, we describe the implementation of several parallel symbolic as well as numerical algorithms on the OCAPP in order to show its use and processing benefits.

Table 1: **Formulation of the interrogation register**

Search bit c_j	Mask bit m_j	Interrogation bits I_j I_j
0	0	0 1
1	0	1 0
0	1	1 1
1	1	1 1

The entry $I_j\bar{I_j} = 11$ means that no comparison is performed at that bit position for all words.

3 Parallel Search Algorithms on OCAPP

We classify search operations as basic and compound operations. A basic search operation is one which can be completed in one sweep over all the bit-slices of the storage array. It does not involve any feedback processing. A compound search operation requires a feedback from the response unit to the selection unit. As a consequence, it takes more than one sweep over the storage array to complete.

3.1 Parallel Algorithms for Basic Search Operations on OCAPP

In what follows, we denote a memory word as $W_i = (w_{im}w_{im-1}\dots w_{i1})$ where w_{ij} is the jth bit cell of the word W_i. We denote the jth bit-slice by $B_j = (w_{1j}w_{2j}\dots w_{nj})$, which is made up of the jth bit of every word in the storage array. The interrogation and response registers are denoted by $I = I_m I_{m-1}\dots I_1$, and $R = R_1 R_2 \dots R_n$ respectively. The comparand word (search argument) and the mask register words are denoted by $C = (c_m c_{m-1}\dots c_1)$, and $M = (m_m m_{m-1}\dots m_1)$.

3.1.1 Equivalence Search

In this type of search, the memory is partitioned according to the magnitude of the search word C into two sets, namely, words which are equal to C and words which are not. The equality and masked search operations can be implemented by a bitwise match. For equality match all the bits of the search word need to be matched, whereas for the masked search, only a subset of the bits of the search word is compared with the respective bits of the memory words. For $m_j = 0$ means that c_j is not masked, while $m_j = 1$ means c_j is masked. These two search modes can be combined as shown in Table 1.

Given a search word C, a bit match denoted by b_{ij}

on the jth cell of the ith word is given by:

$$b_{ij} = (I_j \wedge w_{ij}) \vee (\bar{I_j} \wedge \bar{w_{ij}}) \qquad \text{(exclusive-and)} \quad (1)$$

where the symbols $\wedge$, $\vee$, and the bar ($^-$) denote the logical AND, logical OR and logical NOT respectively. Now the exact matching of memory word W_i with search argument C requires the logical product of the bits b_{ij} for $j = 1, \ldots, m$, therefore:

$$R_i = \bigwedge_{j=1}^{j=m} b_{ij} = b_{im} \wedge b_{im-1} \wedge \ldots \wedge b_{i1}. \qquad (2)$$

where $\bigwedge$ denotes a logical AND over all bits. The above equation indicates that matching words in memory will be flagged by having their corresponding R bit set to one, and all mismatches will have their R bits set to zero. Equations 1 and 2 are space-invariant and can be implemented in bit-parallel, word-parallel, and therefore, all R_is for $i = 1, \ldots, n$, are computed at the same time with a single access to the storage array.

3.1.2 Threshold Search

This mode of search partitions the memory according to the magnitude of the search word C into three sets, namely words which are equal to C, words which are less than C, and words which are greater than C. The result of the search is stored in the three registers of the response unit, namely R, G and L. Initially, all memory words are made active by making control registers RGL = 100. The memory is scanned from the most significant to the least significant bit position by enabling a single bit-slice at a time. When the comparand bit c_j is one, we select all active memory words with $w_{ij} = 0$ as "less than" by setting their corresponding bit position RGL = 001. These words are then disabled from further comparisons (the disabling process will be explained later). Similarly, when $c_j = 0$, we select all active memory words with $w_{ij} = 1$ as "greater than" by setting their corresponding bit position RGL = 010, and then disable them from further processing. At the end of the last bit position, words still in the state RGL = 100 are equal to the comparand, words in the state RGL = 010 are greater than the comparand, and words in the state RGL = 001 are less than the comparand. It is important to note that, even though we are scanning the memory from most significant bit to least significant bit, the search process can be terminated any time there are no matching words at a given bit position ($R_i = 0$ for all $i = 1, \ldots, n$). Such a condition is easily detectable by checking the MD bit.

3.1.3 Extrema Search

This type of search refers to finding the maximum (or minimum) of a set of (or all) memory words. We consider first the search for maximum.

A. Maximum Search

To find the maximum, we scan memory words from the most to the least significant bit positions. As we scan the bit-slices, we determine if any of the enabled words have a one in the current bit position. If we find some, we disable all those words that do not have a one in this position. If none of the words at the current position possess a one, we do nothing. At any given time, all remaining candidates are equal as far as we have examined them, because for every bit position either everybody had a zero in that bit position, or whenever some words have ones, we disable the ones with zeros. Therefore, at bit position j, enabled words with $w_{ij} = 1$ are larger than enabled words with $w_{ij} = 0$. Since we are seeking the maximum, we disable the ones with $w_{ij} = 0$. This process is repeated until we exhaust all bit positions at which time the maximum word will be indicated by $R_i = 1$.

B. Minimum Search

The search for the minimum is very similar to the search for the maximum except that the I register is initially loaded with zeros and that if any enabled word has a zero in the current bit position (There exists a memory word W_i such that its corresponding $R_i = 1$), we disable the words with a one in the current bit position ($R_i = 0$). These words are bound to be greater than the minimum sought. The process is repeated until we exhaust all bits of the enabled words. The minimum value will also be indicated by a one in register R.

3.2 Parallel Algorithms for Compound Search Operations on OCAPP

3.2.1 Double Limits Search (Between and Outside Limits)

Given two numbers called HIGH and LOW, the double limits search consists of finding those words that are between this limits and/or words that are outside these limits. This gives rise to eight different searches which can be accomplished in a very similar manner. Let us consider the between limit search. Given the two numbers HIGH and LOW, we wish to find those words that are greater than LOW but less than HIGH namely, find all W_i such that $LOW < W_i < HIGH$. We can accomplish this search by using the magnitude comparison search as follows. First, we determine the words that are less than the comparand HIGH. These words will be indicated by a one in the L register. We then disable all other words except the ones that are less than HIGH, and perform another threshold search using the comparand LOW. After the second search, words that are less than HIGH and greater than LOW will be marked with a one in the G register, which could be routed to the output unit for outputting the search result.

3.2.2 Adjacency Search:

To find the word that is next-above the comparand (the smallest word larger than the comparand), we search for all words that are larger than the comparand and then select their minimum. Similarly, to find the word that is next-below the comparand (the largest word smaller than the comparand), we search for all words less than the comparand and select their maximum.

The search for the largest word smaller that the comparand (next-below search) can be carried out by a similar algorithm as the one above. In this case, step two of the next-above algorithm is replaced by a search for words that are less than the comparand, and step four is replaced by a maximum search.

3.2.3 Ordered Retrieval (Sorting)

The sorting or ordered retrieval of a set of data can be achieved by performing the extrema search repeatedly until all the data are retrieved. For the ascending order retrieval, we enable the memory words to be sorted, and determine their minimum (using the minimum search operation). We output the obtained minimum value and disable it from the storage array. We repeat these steps until we retrieve (in ascending order) all the enabled words. For descending order retrieval, we select the maximum value at each step.

4 Optical Implementation

The optical components required to implement OCAPP can be divided into (1) logic elements, (2) storage elements, and (3) information transfer elements (or interconnects). For optical logic and storage, many approaches are being investigated. One approach is the adaptation of the spatial light modulator (SLM) technology to optical logic[5]. Another approach for realizing optical components capable of performing logic, is to optimize the device from the beginning for digital operations. The recent emergence of the quantum-well self-electrooptic effect device (SEED) and its derivatives (S-SEED, T-SEED, D-SEED) is one such a product[6]. The SEED devices can be used to realize both logic operations such as NOR, OR, AND, NAND, etc. as well as for storage such as S-R latches[6]. Optical resonators are another family under this approach intended for optical logic. Two similar bistable devices, etalons, and interference filters both based on the Fabry-Perot resonator are being actively pursued[7]. All data movements and information transfer in OCAPP are space-invariant which may render their implementation easier. Classical optical components such as lenses, mirrors, beam splitters, holographic deflectors, and delay elements are most likely to be used for this purpose[8]. In addition, halfwave plates, shutters, and masks may be used for dynamic routing[9].

4.1 A Modular Implementation of OCAPP

The implementation of this first version will make use of the SEED device operating as a NOR gate for optical logic, and of the S-SEED device operating as a S-R latch for storage[6]. As described earlier, the optical processor can be constructed from several units: the selection unit, the match/compare unit, the response unit, the output unit, and the control unit. In what follows, we describe the optical implementation (architectural rather than experimental setup) of each of these units. Details of routing and imaging have been omitted.

4.2 The Optical Selection Unit

The optical selection unit of Fig.3 is composed of a storage array which consists of a 2-D $n \times m$ array of clocked S-SEED devices (each entry in the array at position i, j has two incoming bits S, R and two outgoing bits w_{ij}, $\bar{w_{ij}}$), a $n \times 1$ word register A which serves at setting and resetting data words in the storage array, a $1 \times (m+1))$ bit-slice loading register B for loading a single bit-slice of the storage array. The first bit B_0 and its complement $\bar{B}_0$ are called set-E and reset-E respectively, since they are used for setting and resetting the $n \times 1$ enable register E which is used for the matching process (to be explained below). Memory words are disabled through the $n \times 1$ NOR gate array, representing the D register. The D register can be loaded from R, G or L registers. In addition, the E register can also be reset from the priority register P of the response unit (to be explained below).

To allow a memory word W_i in participating in the matching process, its corresponding bit E_i in the E register must be set high. Similarly, to disallow a memory word w_i from participating in the matching process, its corresponding bit E_i in the E register must be made low. To enable/disable the entire memory words, the set-E/reset-E bits ($B_0/\bar{B}_0$) are spread out vertically and broadcast to all the set/reset ports of E. To selectively disable memory words whose R, or G or L bits are not asserted (R= 0, or G = 0 or L = 0), requires the routing of the appropriate register (R, G, or L) to the NOR gate array D. The output of D (which represents the complement of the routed register) is imaged onto the reset ports of register E. Thus a low bit R_i of the R register will disable the i-th bit of the E register, which in turn disables memory word W_i from participating in further comparisons.

4.3 The Optical Match/Compare Unit

This unit performs exact match, and magnitude comparison searches between the interrogation register I and words of the storage array. As shown in Fig.4, It contains several SEED arrays operating as NOR gate arrays, and three registers, namely the response register, R, the greater than register G, and the less than register L. Parallel comparison takes place between memory words emanating from the storage array and the interrogation register I. A match at bit w_{ij} is detected by an exclusive-and principle as indicated in Eq.1. For that, register I needs to be spread out vertically so that each bit I_j impinges on one port of the NOR gates of the j-th column of the array, while data bits w_{ij} impinge on the second port of the NOR gates of the same j-th column. Matches between I_j and w_{ij} are reported in bit R_i of the R register. Otherwise, the G and L registers indicate the relative magnitude. The contents of R , G, and L are routed to the response unit as well as fed back to the selection unit. The R register bits are logically ORed to form the Match Detector (MD) bit. The MD flip-flop indicates if there is any match between I and memory words.

4.4 The Optical Response Unit

The response unit, contains a combinational priority circuit, and a priority register P for indicating the first matching word in memory (It may also contain few scratchpad registers for temporary storage). The priority circuit allows only the first responder (the first memory word W_i whose R_i is one) to pass to the priority register P. The priority circuit can be implemented using several stages of the NOR gate arrays in the form of a binary tree with space-invariant interconnections between them. Contents of the P register are routed to the output unit, and also fed back to the selection unit.

4.5 The Optical Output Unit

The output unit outputs memory word whose corresponding bit in the priority register P, R, G or L is set to one (Fig.5). These latter registers are routed to a $n \times 1$ NOR gate array, denoted by N in Fig.5, whose sole purpose is to invert their values. Each bit N_i of N is logically NORed with memory word W_i using a 2-D NOR gate array. Next, each column of the NOR gate array is logically ORed to form output bit O_i of the output register. This latter could be a photosensitive device which only detects the presence of light and outputs electrical signals, or a 1-D array of SEED devices acting as OR gates, and outputting optical signals. It should be noted that parallel readout of selected memory words is also achievable by replacing P with a 2-D output device and eliminating the OR function.

5 Conclusions

This paper presented the principles and initial design concepts of an associative architecture that matches well with optics advantages, and therefore is highly amenable to optical implementation. The architecture relies heavily on the use of space-invariant interconnections, optical signal broadcasting and funneling (combining), and the simultaneous application of the same operation to many data points (SIMD mode of computing). The motivations behind this is the ease with which these operations can be realized with optics. A representative set of search algorithms have been presented to show the use and merits of the architecture. These algorithms are key components which occur in large computing tasks. It is important to note that these fundamental search algorithms are implemented on the optical architecture with an execution time independent of the problem size (the number of words to be processed). This indicates that the architecture would be best suited to applications where the number of data sets to be operated on is high.

References

[1] T. Kohonen, *Content-addressable memories*, Springer-Verlag, 1980.

[2] C. C. Foster, *Content Addressable Parallel Processors*, Nostrand Reinhold, 1976.

[3] A. Louri, "3-D optical architecture and data-parallel algorithms for massively parallel computing," *IEEE MICRO*, April 1991.

[4] P. B. Berra, A. Ghafoor, M. Guizani, S. J. Marcinkowski, and P. A. Mitkas, "Optics and supercomputing," *Proceedings of the IEEE*, vol. 77, pp. 1797 - 1815, Dec. 1989.

[5] J. A. Neff, R. A. Athale, and S. H. Lee, "Two-dimensional spatial light modulators: a tutorial," *Proceedings of the IEEE*, vol. 78, pp. 836 - 855, May 1990.

[6] A. L. Lentine, H. S. Hinton, D. A. B. Miller, J. E. Henry, J. E. Cunningham, and L. M. F. Chirovsky, "Symmetric self-electrooptic effect device: optical set-reset latch, differential logic gate, and differential modulator/detector," *IEEE J. of Quantum Electron.*, vol. 25, pp. 1928 - 1936, Aug. 1989.

[7] J. L. Jewell, M. C. Rushford, and H. M. Gibbs, "Use of a single non-linear Fabry-Perot etalon as optical logic gate," *Appl. Phys. Lett.*, vol. 44, pp. 172 - 174, Jan. 1984.

[8] A. W. Lohmann, "What classical optics can do for the digital optical computer," *Applied Optics*, vol. 25, no. 10, pp. 1543 - 1549, 15 May 1986.

[9] A. Hartmann and S. Redfield, "Design sketches for optical crossbar switches intended for large-scale parallel processing applications," *Optical Engineering*, vol. 28, no. 4, pp. 315 - 328, May 1989.

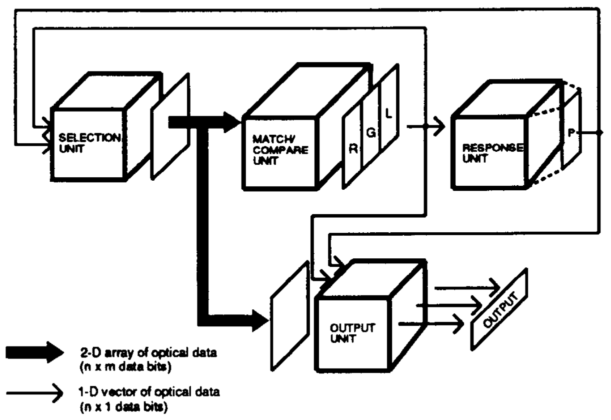
SELECTION UNIT
MATCH/ COMPARE UNIT
R
G
L
RESPONSE UNIT
P
OUTPUT UNIT
OUTPUT
2-D array of optical data (n x m data bits)
1-D vector of optical data (n x 1 data bits)

Fig. 1. A schematic organization of the optical content-addressable parallel processor : OCAPP.

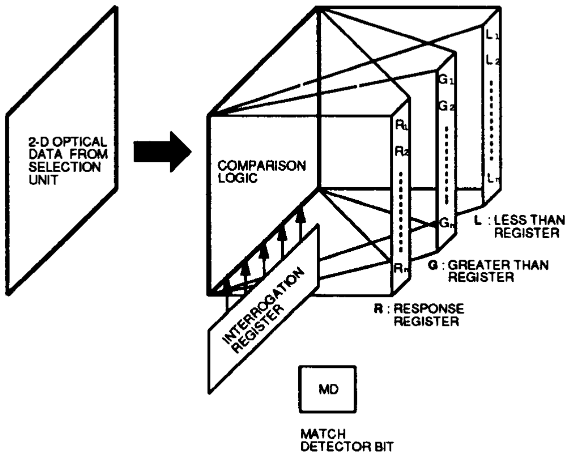

Fig. 2. Organization of the match/compare unit.

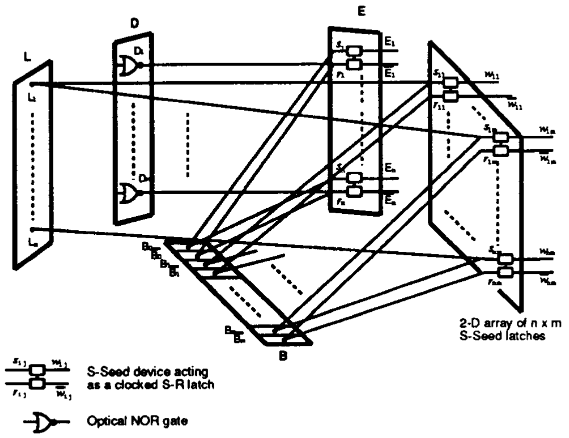
D
E
L
B
2-D array of n x m S-Seed latches
S-Seed device acting as a clocked S-R latch
Optical NOR gate

Fig. 3. Optical implementation of the selection unit.

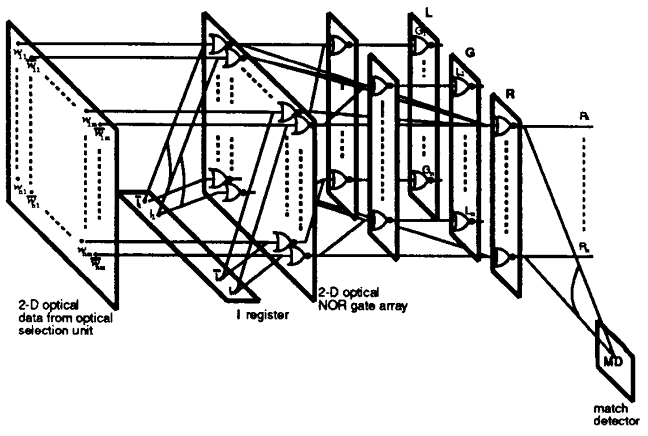
L
G
R
2-D optical data from optical selection unit
I register
2-D optical NOR gate array
MD
match detector

Fig. 4. Optical implementation of the match/compare unit.

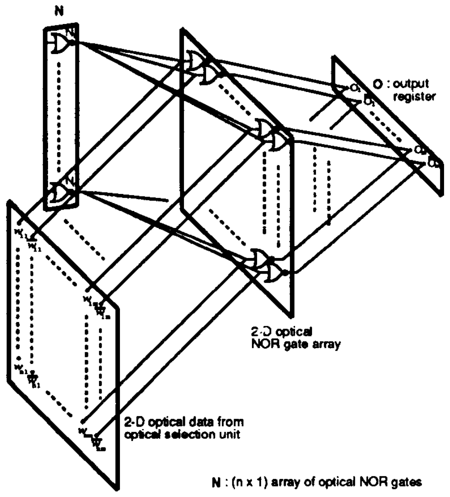
N
O : output register
2-D optical NOR gate array
2-D optical data from optical selection unit
N : (n x 1) array of optical NOR gates

Fig. 5. Optical implementation of the output unit.