Parallel electro-optical rule-based system for fast execution of expert systems

Ahmed Louri and Jongwhoa Na

The slow execution speed of current rule-based systems has restricted their application areas. Multiprocessor architectures have been proposed to overcome this limitation. However, as the number of processors in a multiprocessor system grows, so does the cost of communication between processors or between processor and memory units. The use of optics for a fast and parallel implementation of rule-based systems is proposed. The proposed optical system is hybrid in nature, using electronics for the user interface and optics for the rule-based inference engine. The proposed system uses two-dimensional planes as basic computational entities and is therefore able to provide concurrent rule processing. Furthermore, it provides highly efficient implementation of the basic operations needed in rule-based systems; namely, matching, selection, and rule firing. The execution speed of the proposed system is theoretically estimated and is shown to be potentially of orders of magnitude faster than current electronic systems.

Key words: Expert system, electro-optical rule-based system, inference engine, optical interconnection networks.

1. Introduction

A key feature of artificial intelligence computing systems is the large amount of irregular communications required between the large number of processing elements (PE's). Unfortunately, current electronics technology is unable to provide adequate bandwidth and connectivity for the large number of PE's involved. Consequently, current symbolic computing tasks, including rule-based systems (RBS's) require an excessive amount of time to produce results. Optics offers the possibility of eliminating the performance bottlenecks associated with communications. Optics has the advantages of higher spatial and temporal bandwidth, no cross talk, larger fan-in and fan-out, and two-dimensional (2-D) data-processing capability. These advantages are useful in exploiting massive parallelism and in achieving higher throughput. Optical systems can provide not only adequate communication support, but also the necessary parallelism to perform the most frequently used and time-consuming operations in RBS's, namely, pattern matching and comparison operations.

This has resulted in major research efforts for exploiting the use of optics in symbolic artificial intelligence computing. Warde and Kottas have proposed two different hybrid optical inference machines. The matched-filter inference machine performs inferencing using a classical Vander-Lugt matched-filter system. The mapped template inference machine matches a template representing the relationship between input and output data to produce related output data with the optical correlograph system of Willshaw and Longuet-Higgins. Jau et al. used a semantic network as a knowledge-representation scheme. Jau showed that a simple query can be answered by using an optical matrix-multiplication technique. Schmidt and Cathey proposed an optical mathematical resolution system, which can be used as an inferencing mechanism in an expert system.

We propose a parallel electro-optical rule-based system (called EORBS), in which rules are used to represent knowledge and an inference engine is used for reasoning. To take advantage of optics properties, we represent rules in 2-D space, and the inference engine is logically transformed into an optical interconnection network that can carry out inferencing optically and hence in a highly parallel fashion.

Section 2 provides a brief background of RBS's. Section 3 describes fundamental concepts of the proposed EORBS. Section 4 discusses the implemen-
2. Background

An expert system can be defined as an intelligent system that can mimic some part of human intelligence. For example, MYCIN is a rule-based expert system that diagnoses bacterial infections. MYCIN uses ~500 rules to diagnose approximately 100 causes of infections. An expert system is composed of (1) a RBS and a knowledge-acquisition facility, (2) an explanation facility, and (3) a user interface, as shown in Fig. 1. The RBS performs inferencing by using the knowledge base and the inference engine. The knowledge-acquisition facility and the user interface act as an interface unit between the RBS and the user. The explanation facility explains to the user how the results have been obtained. The knowledge base is composed of rules representing universal knowledge and facts that represent current knowledge. A rule conjoins condition elements $C_i$ and action elements $A_i$ with the following format:

$$\text{if } (C_1 \land C_2 \land C_3 \cdots \land C_n) \text{ then } (A_1, A_2, \ldots, A_m).$$

The condition elements and action elements are represented by facts, which are the basic units of knowledge in an RBS.

The inference engine uses the modus ponens theorem as an underlying principle. The modus ponens theorem states that if there is an axiom of the form $E_1 \Rightarrow E_2$, and if there is another axiom of the form $E_1$, then $E_2$ logically follows. The inference engine uses modus ponens as an underlying principle as follows:

$$\left\{ \begin{array}{c} \text{current fact} \\
(A \text{ is TRUE}) \land \{(\text{if } A, \text{ then } B)\} \end{array} \right\} \Rightarrow \{(B \text{ is TRUE}) \}$$

In an RBS, facts and rules can be represented as an inference net or as an AND/OR tree. Figure 2 shows an inference net and an AND/OR tree for the following set of rules (R1–R4):

- If $(a \land b)$ then $i$,
- If $(a \land c)$ then $j$,
- If $(d \land i)$ then $u$,
- If $(e \land j)$ then $v$,

where $\land$ is a logical AND operator.

Considering that the typical number of rules in an RBS is large and that each rule has several
condition/action elements, the inference net can be extremely complicated. To traverse this increasing search space effectively, researchers have proposed parallel-processor systems that incorporate sophisticated parallel algorithms.\(^2\) For example, DADO2 (Ref. 1) is a tree-structured multiprocessor system that has 1023 PE's. Each PE is composed of an 8-bit microprocessor, 64 kbytes of random-access memory, and a switch. Owing to the large number of PE's, communication time is a dominating factor in DADO2.\(^2\) Furthermore, since each rule of the knowledge base is unique and requires a different processing time, there is a synchronization problem between rules with a small number of condition elements and rules with a large number of condition elements.

As an alternative to the electronic parallel-processing systems, optical systems such as the matched-filter inference engine (MFIE)\(^7\) and the matrix-multiplication inference engine (MMIE)\(^9\) have been proposed. These systems can represent a large number of related facts, called relations, owing to the multi-dimensionality of optical systems. One characteristic of these systems is that many facts within a relation can be processed concurrently (fact-level parallelism). In the MFIE a relation (set of facts) is compared with a condition element of a rule; therefore, if a problem consists of \(r\) rules, \(l\) condition elements per rule, and \(a\) condition elements per relation, the MFIE must execute \(rl/a\) matched-filtering operations to answer a query. In the MMIE a relation is represented in a 2-D matrix. Then, this relation is loaded into an optical matrix-multiplication unit so that multiple facts in a relation can be compared with a condition element of a rule. Thus, for a problem consisting of \(r\) rules, \(l\) condition elements per rule, and \(a\) condition elements per relation, the MMIE requires \(rl/a\) matrix multiplications to generate a final matrix that can be used to answer a query. Note that the MFIE and MMIE require a multiple optical matched filter or a multiple optical matrix multiplier, respectively, to exploit a condition element-level parallelism and a rule-level parallelism available in the rule-based system. In the following sections we discuss an optical RBS architecture that can exploit a condition element-level parallelism and a rule-level parallelism as well as a fact-level parallelism.

3. Electro-Optical Rule-Based System: Overview
The main objective of the EORBS is the exploitation of maximum parallelism in RBS's. This is achieved by using a 2-D representation of knowledge and multidimensional optical interconnects to speed up the match, select, and act operations of an RBS. EORBS offers a major advantage over previous proposals (including optical ones): simultaneous processing of several rules as opposed to the sequential processing of rules.

In order to process many rules in parallel, we have to ensure that multiple rule firing has no side effects.\(^2\)\(^2\) EORBS guarantees this by employing a monotonic-reasoning methodology\(^2\)\(^2\) and by identifying a set of mutually exclusive rules. The monotonic-reasoning methodology ensures that rule-firing operations only augment the facts without deleting or changing any facts that can be sources of side effects. Therefore, once a fact is asserted, the fact remains true forever. Thus the number of true facts keeps growing until the end of an inference operation. To apply the monotonic-reasoning scheme, we have to restrict a variable to be bound to some value during compilation time so that the data presented at the run time can be represented by propositions. Consider the following examples \(F_1\) and \(F_2\):

\[
72 \leq 3.832,
\]

\[
a \geq b.
\]

For \(F_1\) the value of the expression can be evaluated directly so that the host can assign FALSE to \(F_1\). For \(F_2\), first, the variables \(a\) and \(b\) must be bound to some values. Then, the host can evaluate \(F_2\) and assign TRUE or FALSE to \(F_2\). Therefore, if we can collect all the data in advance and we can infer something out of the given knowledge, the monotonic-reasoning methodology can be applied. The application areas for the monotonic-reasoning methodology include theorem proving, problem-solving systems, diagnostics, and consultation problems.

In order to evaluate rules in parallel, the host must maintain consistency. To maintain consistency, the host must compare a fact element with a group of
related rules and facts. Thus the size of the group can be important in that the comparison time depends on the number of data in the group. One of the major characteristics of the rule-based programming environment is modularity and hierarchy. These stem from the fact that the structure of human knowledge can be modeled in a modular and hierarchical structure such as a tree, in which each node represents a module that is composed of a small number of rules and facts. In this tree (from a high-level viewpoint), checking whether an incoming fact belongs to some mutually exclusive group of facts or not implies that the incoming fact must be compared with a mutually exclusive group. Thus the fact is compared with a specific group that is small in size. Also, note that this checking operation is performed at the compilation time. Therefore the checking time for a given fact for the group should not cause any excessive overhead at run time.

Using a monotonic-reasoning scheme is advantageous in that EORBS does not require conflict resolution. Since EORBS explores all the possible paths in the search space without any side effect, it does not have to select which path to follow. An example of parallel rule firing can be found in the fuzzy RBS used in control.24

A. Knowledge Representation in the Host Computer

In general, to run any high-level programming language on a computer, one must translate the source code into an object code that can be run by a central processing unit. Likewise, RBS's require the rules to be in human-readable form, which is also required for the translated code that is optimized for the hardware. In the RBS the human-readable rules act as an interface between the user and the computer. These rules are used in editing and debugging the knowledge base of a given problem. As a result, RBS's keep their own version of the translated knowledge base. As an example, MYCIN uses a rule compiler for generating a decision tree that is used by the inference engine. For EORBS we developed a translation algorithm that translates the knowledge base into a 2-D array called a proposition table (PT). Figure 3(b) shows a PT that results from the application of the translation algorithm introduced below to the knowledge base of Fig. 3(a).

An entry of the PT is composed of a condition/action element and a proposition. The condition element slot can be divided further into a left-element slot, an operator slot, and a right-element slot. The left-element slot has the name of an object or variable, the right-element slot has a variable or some constant, and the operator slot contains the operation to be performed with the left-element slot and right-element slot. Also, the proposition slot can be divided further into two slots: one for the name (integer identification number) of the proposition and the other for the binary value of the proposition.

Along with the PT, the host computer generates a condition array (CA) and an action array (AA), which are shown in Figs. 3(e) and 3(f), respectively. The CA and the AA represent the condition parts and action parts of all the rules in the knowledge base. The CA has a dimension of \( r \times l \) and the AA has a dimension of \( r \times m \), where \( r \) is the number of rules, \( l \) is the number of condition elements per rule, and \( m \) is the number of action elements per rule.

The host also generates two copies of the 2-D fact array [Figs. 3(c) and 3(d)]. In the first copy, called the reference fact array (RFA), each element of the array represents the name of the fact; therefore the RFA is a name template of the optical fact plane of EORBS. In the second copy, called the data fact array (DFA), each element of the array is the value of the element that is designated by the RFA; therefore the DFA is an actual data plane for EORBS. An
algorithm that interprets a human-readable knowledge base accessed by users to generate the PT, CA, AA, RFA, and DFA is described below.

1. Initialize the PT, RFA, DFA, CA, and AA.
2. For \( i = 1 \) to (the number of rules), find current rule.
3. For \( j = 1 \) to (the number of condition elements of the current rule), do the following:
   a. Find the current condition element.
   b. Find the symbol for the current condition element. (If the current condition element has already used the symbol, then find the used symbol, else create a new symbol.)
4. With the symbol (current condition element), do the following:
   a. Evaluate the value of the symbol. (Fill in a row of the PT.)
   b. Calculate \( \text{PT}[\text{symbol}] = (\text{current element, symbol, symbol value}). \)
   c. Calculate \( \text{CA}[i, j] = \text{symbol}. \)
   d. Calculate \( j = j + 1; \) go to step (3).
5. For \( k = 1 \) to (the number of action elements of the current rule), do the following:
   a. Find the current action element.
   b. Find the symbol for the current action element. (If the current action element has already used the symbol, then find the used symbol or create a new symbol.)
6. With the symbol (current action element), do the following:
   a. Evaluate the value of the symbol. (Fill in a row of the PT.)
   b. Calculate \( \text{PT}[\text{symbol}] = (\text{current element, symbol, symbol value}). \)
   c. Calculate \( \text{AA}[i, k] = \text{symbol}. \)
   d. Calculate \( k = k + 1; \) go to step (5).
7. Calculate \( i = i + 1; \) go to step (2). (Find the next rule.)

B. Knowledge Representation in Electro-optical Rule-Based System

In EORBS, knowledge is represented in a 2-D form. Facts are represented in a 2-D plane called the fact plane. Rules have two parts, a condition plane to represent condition elements and an action plane to represent action elements. These planes represent the basic computational entity in EORBS. Also, to increase the utilization ratio of limited 2-D space, propositional logic is used. In a propositional logic system, each fact, goal, condition, or action is represented by a pixel in the corresponding 2-D plane. Each element of the plane is expressed in a proposition that is position encoded in 2-D space, as shown in Fig. 4. The value of a fact \( F_i \), for example, is represented by light being on/off at the specific location of the fact plane. Similar considerations take place for other planes.

In the fact plane the host allocates a small portion of the fact plane for the goal facts. These goal facts occupy the last row of the fact plane and are used at the end of the inference operation. Thus the host computer must generate the RFA such that the last row of the fact plane has goal facts. If the number of goal facts exceeds one row, the host can add rows in the RFA to accommodate a large number of goal facts. The optimal size for the number of goal facts should be determined after a thorough simulation study.

C. Description of Electro-optical Rule-Based System

The overall role of EORBS is to infer new facts by using the current facts and rules and by comparing the inferred facts to the given goals. Figure 5 shows a block diagram of EORBS. It consists of an electronic host computer, a goal-checking unit (GCU), and an inferencing unit (IU). EORBS receives the necessary facts and rules from the host computer and returns the processed data to the host. The host computer uses a 2-D fact plane to convert electronic data to optical data for optical inferencing. The fact plane is encoded as explained above. The host computes and sets the interconnection pattern between the facts and rules in the IU (explained below).

The IU performs the match and the act operations. The match operation consists of comparing the current facts with the condition elements of the rules. During the act operation, the action parts of the rules with satisfied condition elements are executed. A detailed explanation of the match and the act operations is presented in subsection 3.D.

The role of the GCU is to determine whether or not there is any assertion of goal facts during the inferencing process. If the output from the GCU indicates a match, EORBS stops inferencing, and the host starts its own operation with the processed results from EORBS; otherwise, the inference engine starts the inference operation.
D. Fact-to-Rule and Rule-to-Fact Mapping

Performing the match operation means that the condition elements of rules are evaluated with the current values of facts. This is usually a time-consuming operation in conventional RBS’s, since it is done sequentially. Here we introduce a new method for performing rule matching that can be carried out in parallel, owing to the parallel nature of optical interconnects. The method consists of first providing a mapping called fact-to-rule (F-R) mapping from the facts to the rules. This mapping specifies the interconnection patterns required from the fact plane to the condition plane. Next, the condition templates of each rule are logically AND’ed into a column vector representing the selected rules for the current iteration. This column vector represents the action plane. In general, the size and the dimension of the interconnection network and the AND-gate array representing the action plane depend on the configuration of the condition plane and the number of action elements per rule. In this example, we assume that each rule has only one action, hence the action plane is one dimensional. In general, the action part of the rule may contain more than one action element. In this case the AND array and the action plane are both of dimensions r x m, where r is the number of rules and m is the number of action elements per rule.

In a similar fashion, the act operation must assert the action elements of the triggered rules into the current fact plane to form a new fact plane. Again, this operation is carried out by mapping from the action plane to the fact plane. This mapping, called the rule-to-fact (R-F) mapping, asserts new facts from the newly generated action plane into the previous fact plane. Using the above two rules and the algorithm described below, we can simply find the required interconnection patterns A1-F10 and A2-F20 for the RFMT from the RFA and AA inputs.

```
(1) For i = 1 to (number of elements on the AA), do step (2).
(2) If AA[i] ≠ null, do the following:
   (a) Find the available slot in the row (content of AA[i]) of the RFMT (content of CA[i,j] = symbol identification number in integer).
   (b) Calculate RFMT(CA[i,j]) = (i, j) [save destination (i, j) at the FRMT].
   (c) Calculate j = j + 1; go to (2).
   (d) Calculate i = i + 1; go to (1).
```

Since rules are defined as the conjunction of condition elements, we can set the unused elements (e.g., C1,3 and C1,4 in R1) to a logical 1. Next, all the elements of each row of this condition plane are logically AND’ed together to form a one-dimensional (1-D) column vector representing the selected rules for the current iteration. This column vector represents the action plane. In general, the size and the dimension of the interconnection network and the AND-gate array representing the action plane depend on the configuration of the condition plane and the number of action elements per rule. In this example, we assume that each rule has only one action, hence the action plane is one dimensional. In general, the action part of the rule may contain more than one action element. In this case the AND array and the action plane are both of dimensions r x m, where r is the number of rules and m is the number of action elements per rule.

In a similar fashion, the act operation must assert the action elements of the triggered rules into the current fact plane to form a new fact plane. Again, this operation is carried out by mapping from the action plane to the fact plane. This mapping, called the rule-to-fact (R-F) mapping, asserts new facts from the newly generated action plane into the previous fact plane. Using the above two rules and the algorithm described below, we can simply find the required interconnection patterns A1-F10 and A2-F20 for the RFMT from the RFA and AA inputs.

```
(1) For i = 1 to (number of elements on the AA), do step (2).
(2) If AA[i] ≠ null, do the following:
   (a) Find the available slot in the row (content of AA[i]) of the RFMT (content of AA[i,j] = symbol identification number in integer).
   (b) Calculate RFMT(AA[i]) = (i, j) [save destination (i,j) at the FRMT].
   (c) Calculate j = j + 1; go to (2).
   (d) Calculate i = i + 1; go to (1).
```

The new fact plane contains previous facts as well as newly inferred facts. These facts are then sent to the GCU for further processing. A detailed example of this mapping is described in Section 5.

4. Architecture of Electro-Optical Rule-Based System

A detailed optical architecture for EORBS is shown in Fig. 6. As described in Section 3, the system consists of a GCU, an IU, and 2-D optical source (fact) and detector planes for input–output (I/O) interfacing with the host. A 2-D optical source such as an electrically addressed spatial light modulator (SLM)
can be used for the fact planes. For the electrically addressed SLM, a ferro-electric liquid-crystal SLM, an electrically addressed microchannel SLM, or a silicon/lead lanthanum zirconate titonate SLM can be used. Since EORBS must send results to an electronic host, an optical detector array is needed to convert optical signals into electronic signals. EORBS also needs 2-D optical logic-gate arrays to perform AND and OR logic functions. A possible candidate for the optical logic-gate array is the self-electro-optic-effect-device family of devices.

In what follows, we examine the optical implementation of each unit needed in EORBS.

A. Input-Output Unit
The input–output unit is the interface unit between the electronic host and the optical rule-based system (ORBS). The input–output unit consists of an SLM, an optical OR-gate array O₁, and a 2-D detector array. The SLM is used to load the input fact plane (DFA) to the ORBS. The OR-gate array O₁ receives two input planes, an initial fact plane from the host and a processed fact plane from the ORBS. Thus, by superimposing the two planes, O₁ can generate an updated input fact plane at every cycle for inference operations in the IU. The detector array is used to send the processed fact plane to the host.

B. Inferencing Unit
The IU is composed of an optical AND-gate array A₁ and two interconnection networks, IN₁ and IN₂. The interconnection network IN₁ performs the F-R mapping, and IN₂ performs the R-F mapping, as described in subsection 3.D. IN₁ uses the fact plane as an input and generates the condition plane. The AND-gate array A₁ receives the condition plane from IN₁ and produces a column vector that is used as the action plane. This vector is sent to the IN₂ for asserting the new facts into the fact plane for the next iteration and to the GCU for goal-checking purposes.

Optical Implementation of Fact-to-Rule and Rule-to-Fact Mapping
The optical implementation of the F-R mapping requires a 2-D to 2-D network since we are mapping the 2-D fact plane to the 2-D condition plane. The R-F mapping requires a 1-D to 2-D network that maps an element from the 1-D action plane to some elements of the 2-D fact plane. The required interconnection patterns in the F-R/R-F mappings are any-to-any connections. This means that any element of the input plane should be able to access any element of the output plane. This can be achieved by several techniques. A possible optical implementation that uses space-invariant holograms is proposed below.

Holographic Interconnection Network
As an example, let us consider the following facts and rules:

- facts $F_1, F_2, F_3, F_4$;
- rule 1 is, if $(F_1 \land F_2)$, then $F_3$;
- rule 2 is, if $(F_1 \land F_3)$, then $F_4$;
- rule 3 is, if $(F_2 \land F_3)$, then $F_4$.
The RFMT indicates the following interconnection algorithm, we generate the RFMT shown in Fig. 7(e). Using the same example as above and the RFMT must compare the action elements with the facts. condition elements.

is the number of rules, and is the number of given facts and rules. With a r x I; x I/; fact plane and width product (SBWP) of r x I, is also a multifacet hologram and has the same size as the action plane. Therefore H2 contains three facets, and each facet acts as a one-to-four beam splitter, as shown in Fig. 9.

As in IN1, an SLM is needed to block unwanted connections. Lens L2 is used to demagnify the image generated by H2 and the SLM. The SBWP of hologram H2 is O(r/nl) that of L2 is O(rml), where m is the number of action elements per rule. Note that there are many different ways of achieving these mappings optically.

C. Goal-Checking Unit

The GCU consists of lens L1, spatial filter F1, shutter S1, three beam splitters (BS1, BS2, and BS3), and detector D1. At the output stage of the interconnection network IN2, the fact plane is divided into two fact planes by beam splitter BS1. One is sent to shutter S1 and the other to mirror M3. The fact plane routed to M3 is filtered spatially by F1, so that only the goal fact portion of the fact plane arrives at lens element L1. Then the goal fact plane is collimated by L1 upon detector D1 and by the control input to shutter S1. When the input to D1 is high, D1 signals to the host computer that the goal has been found. At the same time, the high control signal disables the IU from further processing by disabling S1. On a low signal, the low control signal to D1 enables S1, which in turn permits further inferencing in the IU.

5. Detailed Operation of Electro-optical Rule-Based System

For an illustration of the operation of EORBS, consider the example of the family tree of Fig. 10(a).

The current facts consist of two relationships: is-married-to and is-father-of, as shown in Fig. 10(b). The rules for this system are shown in Fig. 10(c).

The goal of this example is to find who is the grandmother of whom (e.g., is-grandmother-of).

Using the rules shown in Fig. 10(c) and the translation algorithm presented in Subsection 3.A, the host computer generates the PT, RFA, CA, and AA [Figs. 10(e) and 10(f)]. Next, the DFA is generated using the RFA of Fig. 10(f). If a fact is currently known, then its corresponding location in the fact plane is set to 1, while others are set to 0. Note that, among the elements of the DFA, the host reserves the last row for the goal facts. The goal of this example is to find a grandmother. Therefore the host assigns the goal facts (F18 and F19) in the last row of the RFA. The resulting DFA is shown in the table of Fig. 11(a).

Then, to provide the means for the F-R mapping and the R-F mapping described in Subsection 3.D, the
Fig. 8. Holographic interconnection network for F-R mapping.

host computes the F-R/R-F mapping and sends it to the IU. Using the RFA and the CA of Fig. 10(f) as inputs, the FRMT algorithm computes the required interconnection patterns. For example, fact $F_1$ is used as $C_{1,1}$ and $C_{2,1}$ of the condition plane. Thus EORBS must be able to connect $F_1$ to $C_{1,1}$ and $C_{2,1}$. The complete F-R mapping is shown in Fig. 12(a).

To generate the R-F mapping, we use the AA and the RFA of Fig. 10(f) as inputs to the RFMT algorithm. For example, the first action element $A_1$ of the action plane is used as the element $F_{10}$ of the fact plane in Fig. 10(e). Hence we need a connection for the first action element $A_1$ to the fact element $F_{10}$. In a similar fashion, the complete R-F mapping is generated [Fig. 12(b)].

Once the DFA and the R-F/F-R mapping information is ready, this complete information is loaded into the optical fact plane and the interconnection networks IN$_1$ and IN$_2$. Then, EORBS starts its operation.

Interferencing unit (F-R mapping). Initially, the fact plane (DFA) is supplied by the host. However, in subsequent interactions, the fact plane includes newly asserted facts from previous iterations by using the optical OR-gate array $O_1$. In this example, each element of the DFA is transferred into a predetermined location of the condition plane by using the FRMT described in Fig. 12(a). Figure 11(b) shows the resulting condition plane. The rightmost two unused columns are set to a logical 1. IN$_1$ produces the column vector of Fig. 11(c). In the column vector, a logical 1 in the $i$th location represents a triggered rule.

Inferencing unit (R-F mapping). If there are rules to be fired, the IU executes the action part of the triggered rules by using the RFMT. The triggered rules are mapped onto the fact plane with IN$_2$. In this example, each triggered rule represented in Fig. 11(c) is mapped onto the new fact plane of Fig. 11(d) with the RFMT shown in Fig. 12(b).

Goal checking. The GCU receives the DFA from the IU. Upon receiving the DFA [see Fig. 11(d)], the goal fact portion of the DFA (shaded pixels of the DFA) is routed to detector $D_1$ and shutter $S_1$ by lens $L_1$. A high signal to $D_1$ and $S_1$ implies that a goal has been found, while a low signal implies that no goal has been found at this time. In this example, the control signal to $D_1$ and $S_1$ is low, which means that no goal fact has been asserted. This ends the first iteration of EORBS.

Fig. 9. Holographic interconnection network for R-F mapping.
Next, using the AA, the IU performs an R-F mapping of Fig. 13(b) row-wise. The DFA [Fig. 11(d)] is combined with the original facts form.

Second iteration. In the second iteration, the DFA [Fig. 11(d)] is combined with the original facts [Fig. 11(a)], which are given by the host. As in the first iteration, the new DFA is routed to the IU to perform F-R mapping. Then, the results of the F-R mapping of Fig. 13(b) are AND'ed row-wise. The result is shown in the column vector of Fig. 13(c). Next, using the AA, the IU performs an R-F mapping and produces the DFA shown in Fig. 13(d).

The GCU routes the goal facts (shaded pixels) of the DFA [Fig. 13(d)] to detector D1. Since there is one pixel (F18) that has a value of 1, the GCU sends a high
control signal to the host and terminates the operation of EORBS by disabling shutter S1. As soon as the host receives the high control signal (goal-found signal), it reads in the last fact plane and converts it to electronic signals by using the RFA shown in Fig. 10(f) to answer the query.

6. Performance Analysis of Electro-optical Rule-Based System

Here we estimate the theoretical performance of the proposed architecture by estimating its execution speed and by comparing it with that of conventional RBS's. In what follows we use the following terms and assumptions for estimating the execution speed:

1. We assume that the response time of optical gate arrays used for logic functions (e.g., optical AND, OR, and optical shutter) is $T_r$.
2. The setup time of the SLM of size $n$ is $T_s$.
3. The processing time of transferring elements of one DFA to another is $T_p$.
4. The access time of the detector plane of the I/O unit is $T_d$.
5. The propagation delay of optical passive devices such as lenses, mirrors, beam splitters, and holograms is negligible.
6. We assume that the number of elements in the fact plane is $n$ and that the number of condition elements per rule is $l$.
7. We ignore the reconfiguration time of IN1 and IN2, since these are reconfigured only at the start of each query. While solving a query, the interconnection networks can be considered fixed.

A. Execution Speed

The execution speed of EORBS can be thought of as being divided into three parts: initialization time $T_{\text{init}}$, execution time $T_{\text{exec}}$, and output fetch time $T_{\text{fetch}}$. In order to estimate the execution time, we consider the longest signal path through EORBS.

To complete one iteration, the optical signal must go through the optical OR-gate array ($O_1$), the IU (one AND-gate array and two SLM's), and one shutter $S_1$. Such a path contains five active devices that require switching; therefore the total time required for one iteration is $5T_r$.

The initialization time includes the setup times for the fact plane and the SLM’s in the interconnection networks $IN_1$ and $IN_2$. However, these units can be accessed simultaneously, and hence $T_{\text{init}}$ becomes $T_s$ (the setup time for a single SLM). The output fetch time includes the time to read in the detector of the GCU. Thus $T_{\text{fetch}}$ is equal to $T_d$.

The data for the input and output planes of EORBS can be formatted in a 2-D array data structure during the knowledge-base compilation time. Since the compiled rules and facts are used throughout the query processing, the sum of $T_{\text{init}}$ and $T_{\text{fetch}}$ becomes the actual time for I/O operations required between the host computer and EORBS. Therefore the overall response time of EORBS, $T_{\text{response}}$, becomes

$$T_{\text{response}} = T_{\text{init}} + T_{\text{exec}} + T_{\text{fetch}} = T_s + x \times 5T_r + T_d,$$

where $x$ represents the number of iterations required to solve a given query.

Here we evaluate the number of rules that can be processed in EORBS. Given the number of condition elements $l$ of a rule, the number of rules that can be represented in EORBS becomes $n/l$. Since the cycle time of EORBS is $5T_r$, the maximum number of rules $R$ that can be processed per second is

$$R = \frac{n}{5T_r}.$$

However, if the size of the problem (the number of rules) is greater than the size of the condition plane of EORBS, we must partition the problem into a set of subproblems whose size fits into the condition plane of EORBS. In this case the cycle time must include the time taken to read in the DFA from EORBS (detector access time $T_d$), the time taken to reconfigure some elements of the DFA into a new DFA' (host processing time $T_p$), and the time to load the input DFA' to the I/O unit of EORBS (SLM setup time $T_s$).

In this study, we assume that the SBWP of the space-invariant hologram of the interconnection network $IN_1$ is $O(rl)$. With this hologram, any fact element in a DFA can access any location in the condition plane. However, there is a high price to pay for such flexibility, namely, a large fan-out factor. This flexibility can be compromised without degrading the performance of EORBS. One method is limiting the fan-out $rl$ to a small integer $\alpha$. This is possible because a single fact element is not used as the condition element of all rules in practical RBS's. As an example, the sample RBS used here requires a maximum fan-out of only 3. Even if there exists an RBS that has a fact $F_1$ that is greater than the permitted fan-out $\alpha$, we can easily duplicate $F_1$ into as many copies as necessary to cover the fan-out requirement.

Limiting the fan-out implies that the number of pixels used of the SLM of the interconnection networks, $IN_1$ and $IN_2$, can be reduced. That is, given $n$,
the number of facts in the fact plane, and given \( n_{rl} \), the number of condition elements in the condition plane, we must use an SLM with a SBWP of \( n_{rl} \) in IN\(_1\). Also, note that the maximum number of rules and facts that can be represented in EORBS is limited by the product \( n_{rl} \). Therefore, if we can reduce the required SBWP of the SLM, we can increase the number of rules and facts that can be represented in EORBS by using the unused parts of the SLM.

To evaluate \( R \), we assume \( y \) represents the ratio of the area that the hologram with the reduced fan-out can cover to the area of the SLM of the interconnection networks. Then, the number of rules that can be represented on EORBS becomes \( n / yl \). Considering this factor, if the size of the problem is greater than the size of EORBS, we see that the maximum number of rules that can be processed per second becomes

\[
R = \frac{n}{yl(5T_r + T_s + T_p + T_d)}. \tag{3}
\]

As an example, if we assume \( n = 10^3 \), \( l = 5 \), \( y = 0.1 \), \( T_r = 10^{-6} \) s, and \( T_s = T_p = T_d = 10^{-3} \) s, EORBS can execute \( \sim 6.65 \times 10^5 \) rules. In order to better appreciate this speed advantage of EORBS, we compare the execution speed of EORBS with the electronic systems. The DADO2 multiprocessor system is estimated to process \( \sim 70 \) rules/s.\(^3\) The non-VON is a multiprocessor system that is composed of 32 powerful large processing elements and 16,000 small processing elements, and it is estimated to process 850 rules/s.\(^4\) With processing capability of \( 6.65 \times 10^5 \) rules/s, EORBS is expected to achieve a system throughput far better than any electronic RBS can achieve.

B. Discussion
The performance of EORBS is primary limited by the availability of optical devices with large SBWP and fast response time. The large SBWP permits the optical RBS to represent an increased number of rules, and the fast response time decreases the time taken for a single iteration in the optical RBS. The limited fan-out concept is introduced to increase the number of rules that EORBS can represent with a given SBWP of an SLM. Along with these requirements, the optical power is also an important problem owing to the fan-out hologram used in the interconnection network. Because of the beam-spreading operation of the fan-out hologram, the fan-out hologram is the major power-consumption factor. However, the limited fan-out concept can also reduce the problem incurred by the beam-spreading operation by controlling the number of necessary output destinations. The limited fan-out can also relieve the device requirements. For example, if we reduce the required target area of a beam, the beam-uniformity condition is greatly relieved. Also, since a small number of pixels of the SLM are routed to a pixel of the condition plane (IN\(_1\)), the contrast ratio of the SLM can be decreased accordingly.

7. Conclusions
While rule-based programming and RBS’s are popular and advantageous for solving problems in the artificial intelligence domain, particularly in expert systems, their slow execution speed has limited their extensive use. Conventional electronic parallel machines contain fast processors, but have limited communication bandwidth, which is critical in parallel processing. Optics can provide the large communication bandwidths and faster execution speeds required for RBS’s.

We have explored an electro-optical rule-based system architecture. The architecture exploits the natural parallelism of optics and the advantages of optical interconnects. A distinctive feature of EORBS is the concurrent rule execution, a feature absent in electronic RBS’s. In addition, EORBS provides an efficient implementation of the basic operations required for a RBS; namely, selection, matching, and rule firing. The concurrent rule execution nature of EORBS, combined with the speed with which the basic operations are implemented in EORBS, can potentially result in a hybrid RBS with significant performance improvements over any existing RBS’s.

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

References