THE LENS INTERCONNECTION STRATEGY

by

Raphael A. Finkel
Marvin H. Solomon

Computer Science Technical Report #387

April 1980

The Lens Interconnection Strategy

by

Raphael A. Finkel

Marvin H. Solomon

ABSTRACT

In this paper we describe a new family of topologies for interconnecting many identical processors to form an MIMD multiprocessor. It extends to arbitrarily many processors while keeping the number of neighbors of any one processor fixed. We show that this family behaves very well with respect to uniformity of bus load, simplicity of routing algorithms, and distance between processors.

TABLE OF CONTENTS

The Lens Interconnection Strategy

1. Introduction

In this paper we describe a new family of topologies for interconnecting many identical processors to form an MIMD multiprocessor. The purpose of the interconnection is to allow the individual processors to communicate with each other as they cooperate to perform tasks. Special topologies can be designed for particular tasks, but general-purpose topologies are important for networks like Cm* [Jones 77], Micronet [Wittie 78, 79] and Arachne [Solomon 79, Finkel 8Øa]. Such topologies may be judged by five criteria:

1. The diameter of the network should grow slowly with the number of processors.

2. The number of neighbors of any one processor should be independent of the size of the network.

3. Addresses should exist for each processor that permit efficient algorithms for routing messages from any processor to any other.

4. The traffic load on various parts of the network should be uniform.

5. Message pathways should be redundant to provide robustness in the event of component failure.

Previously reported interconnection strategies perform well according to some of these criteria, but not others. For example, the hypercube [Sullivan 77] performs excellently with respect to criteria 1 (logarithmic relationship), 3, 4 (completely uniform), and 5 (very high redundancy). Unfortunately, criterion 2 is violated: As the dimension rises, the number of neighbors of each processor increases. The dense snowflake [Finkel 80b] satisfies criteria 2 (fixed complexity), 3, and 5 (good redundancy), but it does less well with respect to criteria 1 and 4: The diameter grows as a root of the number of processors (rather than a logarithm) and traffic load varies widely. The multi-tree-structured network [Arden 78] satisfies criteria 1 and 2, but is very poor with respect to criterion 3: There is no general addressing and routing scheme. It is hard to evaluate its performance on criteria 4 and 5.

This paper presents a new family of interconnection strategies called the lens. We show that lenses perform well with respect to all the criteria.

2. Background

Many other authors have considered interconnection topologies, although differing assumptions about the underlying technology often lead to solutions that are difficult to compare. A recent survey [Siegel 79] distinguishes three main approaches: _multistage_ networks, in which processors are connected to memories or to each other through a network of lines and switches, _dedicated path_ networks, in which processors are connected to each other with (simplex or duplex) lines, and _shared path_ networks, in which sets of processors share a communications medium such as a bus. The lens structure presented in this paper is, strictly speaking, a shared path network, although its principles may be applied to dedicated path networks, and it has striking similarities to several proposed multistage schemes.

The work described here is descended from a shared path scheme called the Mega-Micro-Computer network [Wittie 76]. In such a network, a large number or processors are connected by busses, with some small fixed number of processors (say p, where $p > 1$) allowed on each bus, and at most two busses attached to each processor. Any processor may send a message directly to all neighboring processors, that is, all those connected by any bus to the originating processor; messages may be relayed to more distant destinations. Local connectivity (criterion 2 above) is dependent only on p, not on the total number of processors.

We have investigated several such structures in detail

[Finkel 80b], deriving exact formulas for average and worst-case interprocessor distances and bus traffic, and presenting distributed algorithms for message routing. One of these structures, the _dense snowflake_, is a direct generalization of Wittie's structure. It is constructed recursively, starting with a cluster of p processors on one bus and building a larger cluster from p smaller ones, connecting processors from the edges of sub-clusters with p-1 new busses. The diameter of the snowflake grows in proportion to a root of the number of processors. We presented the _star_ as an attempt to decrease the diameter. Its construction is similar to the snowflakes, but processors near the centers of the sub-clusters are used to connect them together. The diameter is improved (proportional to the logarithm of the number of processors), but the problem of uneven bus traffic is worse. For example, if a random pair of processors exchange messages, the central bus will be used with probability $(p-1)/p$, whereas the outermost busses will only be used with probability about $2(p-1)/N^2$ (where N is the total number of processors). Moreover, the star has only one path between any pair of processors, so any component failure splits the network into unconnected regions.

In the current paper, we improve on these previous results. We also consider the more general situation in which each processor connects to q or fewer busses (for some fixed $q > 1$).

3. Definition of the Lens

Roughly speaking, the lens is constructed of subclusters, q-1 of which are joined by their edges to form a multi-layered structure. The thickening at the center, which gives the lens its name, is intended to add extra busses at the center where they are needed to share the greater message traffic.

More precisely, assuming at most p processors per bus and q busses per processor, a level-1 cluster consists of q-1 busses sharing each of p-1 processors. The situation in which p = q = 3 is shown in Figure 1a, in which processors and busses are represented as "P" and "B", respectively. Each of the shared processors is <u>deficient</u> in that it is connected to only q-1 of the q busses allowed. The busses are also deficient, since each has only p-1 processors. To create a level-n cluster, take q-1 level-(n-1) clusters, add a bus to each deficient processor, and cement the clusters together by allowing corresponding new busses from the q-1 level-1 subclusters to share p-1 new processors (Figures 1b and 1c).

The lens has a natural addressing scheme. We give a processor an address of the form "w.v" where w is a sequence of base p-1 digits and v is a sequence of base q-1 digits. In a level-n lens, all addresses have n digits. We may define the addresses recursively according to the level of the lens. When a cluster is formed from q-1 sub-clusters, each address in the i^{th} sub-cluster is extended by adding the digit i to its right end. None of these new addresses ends with a dot. Each new processor is connected by new busses to q-1 previously deficient processors, one from each sub-cluster. Those old processors have new addresses of the form "w.a", where w is an n-1 digit string (their old address) and a is one digit (the new digit). All old processors connected to a single new processor agree on w, and disagree on a. We give the new processor an address "wb.", choosing a different digit b for each new processor connected to the same group of old ones. Since each new bus is connected to p-1 new processors, all the possible digits b will be chosen. Figure 1 shows the addresses of processors.

In the special case p = q, the number of deficient processors at the border of a cluster is the same as the number of deficient busses at the center. Such a lens may be completed by connecting these deficient processors and busses together. A processor with address "aw." is connected to any deficient bus adjacent to processor "Ø.w". (Our discussion of routing will explain this decision.) Figure 1c shows a completed level-3 lens in which the added connections are indicated by dotted lines.

If p = q, counting addresses shows that a level-n lens has $n(p-1)^n$ processors. Each processor except for the $(p-1)^n$ ones on the edge was given a unique bus during the construction at some level; those last ones may be connected to unique busses in the center. Thus, the number of busses equals the number of processors.

4. Routing

Processors are adjacent (that is, they share a bus) if their addresses follow either of these patterns:

A. One address is "wa.u" and the other is "w.bu".

B. One address is "wa.u" and the other is "wb.u".

In order to find a direct neighbor of any processor, start with its address. Pattern A implies that the dot may be moved one position to the right or the left; the position that is passed over may be set to any digit. Pattern B implies that the digit before the dot may be changed to any digit. In both cases, the new address names a processor adjacent to the old one.

These patterns lead to an algorithm for determining a path between arbitrary processors. For example, a legal path from Ø21.1Ø to ØØ.22Ø (here, n=5 and p=3) is:

```
Ø 2 1.1 Ø
Ø 2 1 Ø.Ø
Ø 2 1.2 Ø
Ø 2.2 2 Ø
Ø Ø.2 2 Ø
```

Pattern A has been used for the first three steps, and pattern B for the last one. If we restrict ourselves to pattern A alone, that last step requires two:

```
Ø 2.2 2 Ø
Ø.1 2 2 Ø
Ø Ø.2 2 Ø
```

Given two addresses, which we may call the _source_ and _destination_, we wish to find all minimal paths between them. First, define a _region of difference_ as the minimal set of digit positions that satisfies these properties:

1. The positions are contiguous in the range [1..n].
2. The source and destination addresses agree in all positions outside the region of difference.
3. The dot is within the region of difference or on its borders.

In the example above, the region of difference is those positions marked here with 1:

```
Ø111Ø
```

The source and destination may agree in places within the region of difference, but they must disagree on the borders of that region (at least with respect to the position of the dot).

To move from the source to the destination, it is necessary to change all the digits in the region of difference and end up with the dot in the correct place. The algorithm has four steps:

1. The dot moves from its position in the source address to one end of the region of difference; all digits are replaced with arbitrary ones.
2. The dot moves back to the correct position in the destination address; all digits are replaced with correct ones for the destination.
3. The dot continues to the other end of the region of

difference; all digits are replaced with arbitrary ones.

4. The dot moves back to the correct position in the destination address; all digits are replaced with correct ones for the destination.

Any of these steps may be trivial if the source or destination address has its dot at the end of the region of difference.

Moving the dot by one position corresponds to traversing one bus on the path. Let k be the number of positions in the region of difference, and let i and j be the distance of the dot from the left end of the region of difference in the source and destination addresses, respectively. If step 1 moves towards the right end of the region of difference, then the path length will be $k-i$ for step 1, k for steps 2 and 3, and j for step 4; a total of $2k + j - i$ steps. Similarly, if step 1 moves left first, the total distance is $2k + i - j$. Therefore, a shortest path begins by moving the dot right if and only if $j < i$. If $j = i$, both directions yield minimal paths.

The maximum distance between two processors occurs when $i = j$; in this case, the path length is 2k. Since $k \leq n$, the maximum distance between two processors is 2n.

This routing algorithm uses pattern A exclusively. Pattern B is only useful when the dot changes direction from left to right, and this situation only occurs at most once in any path, so using pattern B could only decrease the length of any path that we form by one step.

5. Routing in the Completed Lens

The routing algorithm is somewhat complicated by completion. We still have the two adjacency patterns shown earlier, but now an address of the form "w." is considered to be equivalent to the address ".w"; the two ends of the address have been joined. It is easiest to view an address as a circular list of digits with a dot between some two adjacent digits.

Suppose source and destination addresses have been chosen and we want to find all shortest paths. The dot positions in the source and destination divide the circle of digits into two arcs; let A be one of the arcs, and let a denote its length (see Figure 2b). Let B be a maximal sequence of contiguous positions within A where the source and destination digits are equal. (If there is more than one maximal sequence choose any one as B; different choices may yield different minimal paths. In Figures 2a through 2f, zero denotes a position in which the source and destination agree, and one denotes a position of difference.) Let b be the length of B.

One way to find a path from the source to the destination is to move the dot from its initial position to one end of B, then back around to the other end of B, and finally to its destination position (see Figure 2c). Call paths formed this way <u>class-1</u> paths. The total length of each of these paths is n+a-2b. As in the routing algorithm for the uncompleted lens, when the dot moves past the same position twice, the digit at that position is set to an arbitrary value the first time and to the value from the destination address the second time. Different choices of B and of the arbitrary values yield different paths within the class.

Paths in class 2 move the dot from the source position, through A to the destination position, and then make another complete circuit (Figure 2d). Each of these paths has length $n+a$, so paths in class 2 are only minimal if $b = 0$ (that is, the source and destination addresses differ at all positions within A). In an analogous manner, letting C be a maximal region of agreement outside of A, and letting c be its length, we get classes 3 and 4 of lengths $2n-a-2c$ and $2n-a$, respectively (Figures 2e and 2f).

Minimal paths are found as follows: Find regions A, B, and C and their lengths a, b, and c. (Several choices may be possible.) Compute the lengths of paths in classes 1, 2, 3, and 4 by the formulas given above and eliminate classes that yield non-minimal lengths. Each remaining class yields one or more minimal paths according to the choices of regions B and C and of "arbitrary" digits. If we are only interested in finding one minimal path, not all of them, we may use a simpler algorithm: Choose class 1 whenever $2b - 2c > 2a - n$; otherwise, choose class 3. If A is chosen as the longer of the two "halves" of the circle, $2a - n \geq 0$.

The worst-case path length occurs when $n = 2a$ and $b = c = 0$ (that is, the source differs from the destination in all positions). In that case, all four paths have the same length:

3n/2.

All paths described above rely exclusively on pattern A. Once again, pattern B can only decrease the path length by one (for paths of classes 1 and 3), since the dot only changes direction from counter-clockwise to clockwise at most once in these paths.

6. Symmetry and Loading

In a completed lens, all processors are equivalent in the following sense:

Theorem 1 For any two processors in a completed lens, there exists a graph automorphism on the lens that maps one onto the other.

Proof We build the necessary automorphism out of the following two operations on addresses:

H(x): shift x cyclically to the right.

G(x): increase the rightmost digit of x by 1 (mod p-1).

Each operation corresponds to a graph automorphism. Given two addresses s_1 and s_2, transform s_1 to s_2 as follows: First, repeatedly apply H to bring each digit of s_1 to the right end. If the digit differs from the corresponding digit of s_2 (that is, the digit at the same distance from the dot position), then make it equal by repeated applications of G. Finally, apply H as necessary to make the dot position agree with s_2.

Definitions We say that message traffic is uniform in a network if (source, destination) pairs for messages are randomly and uniformly distributed. We say a routing strategy is regular if the path chosen for a message is always chosen at random from the set of minimal paths, with all minimal paths equally likely. The loading of a processor is the probability that it is used to relay a message between a random (source, destination) pair.

Corollary If message traffic in a completed lens is balanced and routing is regular, then all processors have equal loading.

Proof Given two processors S and D, there is an automorphism f that maps S onto D. Since f extends to a bijection on the set of minimal paths, and since path p goes through S iff f(p) goes through D, the result follows.

Note The hypothesis of regular routing is stronger than necessary in the corollary above. However, the routing strategy must be sufficiently "fair" that automorphisms preserve path probabilities. The routing algorithm given above satisfies this condition.

Definition The dual of a processor graph G is a processor graph G' formed by relabelling the graph, interchanging busses and processors. (p(G') = q(G), and q(G') = p(G)).

Theorem _2_ A completed lens is isomorphic to its dual.

Proof Let G be the completed lens. Label the busses of G as follows: Each bus is adjacent to a set of q-1 processors labelled wi.v (i = Ø ... q-2) and one processor labelled w.av, for some strings w and v and digit a. Assign the label w.av to this bus. Let f be the mapping from G to G' that assigns a processor to the bus whose label is the _reversal_ of the processor's label, and _vice_ _versa_. Clearly, f is one-one and onto. The busses adjacent to processor wa.v are wa.v and w.iv for i = Ø...p-2, and the processors adjacent to bus w.av are w.av and wi.v, for i = Ø...q-2. Hence if p = q, f is an automorphism interchanging processors and busses.

Corollary For any two busses in a completed lens, there exists a graph automorphism on the lens that maps one onto the other.

Proof Let b_1 and b_2 be busses, let f be the isomorphism of Theorem 2, and let g be the automorphism of Theorem 1 that maps $f(b_1)$ onto $f(b_2)$. Then f^{-1} o go f is an automorphism mapping b_1 onto b_2.

Corollary If message traffic in the completed lens is balanced and routing is regular, then bus loading is uniform. (Bus loading is defined analogously to processor loading.)

7. The Lens and Multistage Networks

An alternate picture of the lens indicates a correspondence between shared-path and dedicated-path topologies and also suggests a striking similarity between the lens and several multistage schemes previously proposed. Figure 3 represents the same interconnection schemes as Figure 1 but with the processors and busses arranged differently on the page. Drawn this way, a level-k lens is formed of k rows of "exchange groups", with each row connected to the next by a perfect shuffle interconnection [Stone 71]. Each exchange group consists of p-1 processors and q-1 busses, connected in a complete bipartite graph. Figure 4 shows Figure 3c with each exchange group replaced by a box. The extra connections to complete the lens are shown as dotted lines. For clarity, the bottom row of boxes is repeated at the top.

The graph structure of Figure 4 has been proposed several times as a multistage network, but only in the case where $p = q = 3$. It has been variously called the flip network [Batcher 74], the indirect binary n-cube [Pease 77], the Omega network [Lawrie 75], and the Banyan [Goke 73]. In each case the interpretation of the graph as an interconnection network has been slightly different, and hence different properties of the graph have been of interest.

To the best of our knowledge, all previous papers mentioning the graph of Figure 4 interpret boxes in the interior of the graph as switching elements of some sort. The boxes at the top and bottom are interpreted as processors, processors and memories, or processors and busses. The results of this paper show that a fruitful interconnection topology can be achieved by interpreting the all the boxes as processing elements or as shared-path subnetworks.

Our results also generalize previous work in allowing numbers of neighbors other than 4. When drawn in the style of Figure 4 the lens has p connections from each box "upward" and q connections "downward".

8. Conclusions

In this paper we have presented a new interconnection strategy called the lens that performs well with respect to all the criteria listed in the introduction:

1. The diameter of the completed lens grows slowly with the number of processors. The diameter d is $\lfloor 3n/2 \rfloor$, and the number of processors N is $n(p-1)^n$, so the diameter is

$$d = \left\lfloor \frac{3(\log N - \log n)}{2 \log(p-1)} \right\rfloor,$$

which is logarithmic in N. The uncompleted lens diameter also grows logarithmically with N; in this case, d is 2n.

2. Every processor is connected to q busses, and every bus is connected to p processors, independent of the size of the network.

3. The processor addressing scheme permits efficient algorithms for routing messages from any processor to any other.

Completing the lens does add complexity to the algorithm, but it is still linear in the length of addresses.

4. The traffic load is uniform for all processors and all busses.

5. Multiple minimal paths provide robustness in the event of component failure.

One problem with the complete lens is that it seems to require expansion in large increments, since the number of processors must be an integer of the form $n(p-1)^n$. Structures of other sizes could be built by viewing them as lenses of the next larger size with some processors and busses missing. The characteristics of such "partial lenses" will be investigated in a future paper.

9. References

Arden, B. E. and Lee, H. "A Multi-Tree Structured Network" Princeton University Electrical Engineering and Computer Science Technical Report #239, January 1978.

Batcher, K. E. "The flip network in STARAN," _Proceedings of the National Computer Conference_, pp. 405-410, May 1974.

Finkel, R. A., Solomon, M. H., Tischler, R., _Arachne User Guide, Version 1.2_, University of Wisconsin--Madison Computer Sciences Technical Report #379, February 1980.

Finkel, R. A, and Solomon, M. H. Processor Interconnection Strategies, IEEE Transactions on Computers, May 198Ø.

Goke, L. R. and Lipovski, G. J., "Banyan networks for partitioning multiprocessor systems," First Annual Symposium on Computer Architecture, pp. 21-28, December 1973.

Jones, A. K., Chansler, R. J. Jr., Durham, I., Feiler, P., Schwans, K., "Software Management of Cm* -- A Distributed Multiprocessor", Proceedings of the National Computer Conference, Vol 46, pp. 657-663, AFIPS Press, 1977.

Lawrie, D., "Access and alignment of data in an array processor," IEEE Transactions on Computers, Vol. C-24, No. 12, pp. 1145-1155, December 1975.

Pease, M. C., "The indirect binary n-cube microprocessor array", IEEE Transactions on Computers, Vol. C-26, No. 5, pp. 458-473, May, 1977.

Siegel, H. J., McMillen, R. J., and Mueller, P. T. Jr., "A survey of interconnection methods for reconfigurable parallel processing systems," Proceedings of the National Computer Conference, pp. 529-542, June 1979.

Solomon, M., and Finkel, R., "The Roscoe Distributed Operating System", Proceedings of the Seventh Symposium on Operating Systems Principles, pp. 1Ø8-114, 1Ø-12 December, 1979.

Stone, H. S., "Parallel processing with the perfect shuffle," IEEE Transactions on Computers, Vol. C-2Ø, No. 2, pp. 153-

161, February 1971.

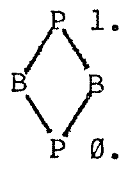
Sullivan, H., and Bashkow, T. R., "A Large Scale, Homogeneous, Fully Distributed Parallel Machine," _Proceedings of the Fourth Symposium on Computer Architecture_, pp. 1Ø5-124, 1977.

Wittie, L. D. "Efficient Message Routing in Mega-Micro-Computer Networks", _Proceedings of the Third Symposium on Computer Architecture_, pp. 136-14Ø, 1976.

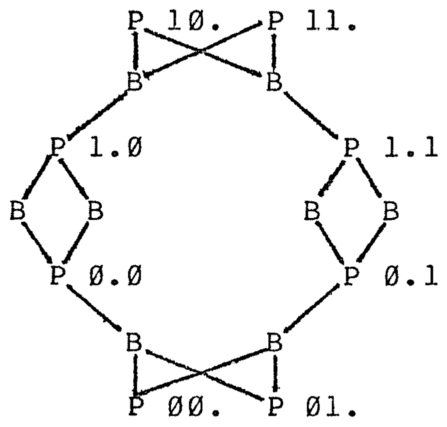
Wittie, L. D., _Micronet: A reconfigurable microcomputer network for distributed systems research_, State University of New York at Buffalo Department of Computer Science Technical Report 143, April, 1978.

Wittie, Larry D., "A Distributed Operating System for a Reconfigurable Network Computer" _Proceedings of the First International Conference on Distributed Computing_, October 1979.

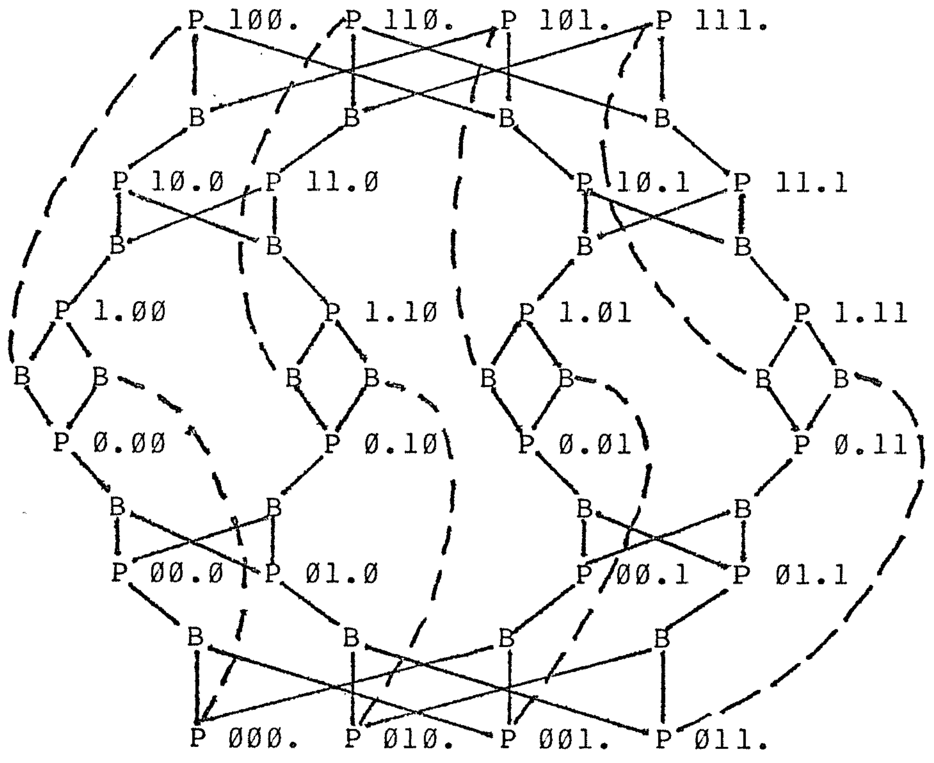
1Ø. Figures



(a) Level 1



(b) Level 2

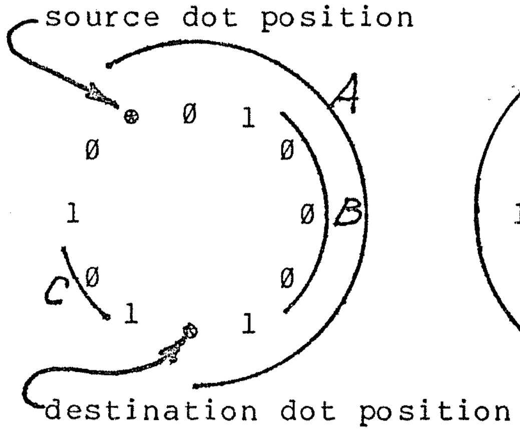


(c) Level 3

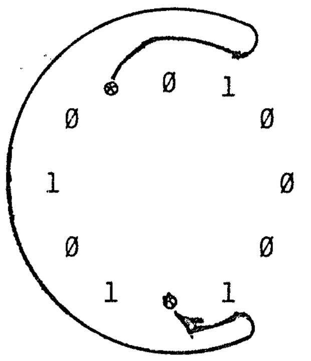
Figure 1
The Lens (p = q = 3)

```
Source address:      Ø 2 1.Ø 2 1 2 Ø 1 2
Destination address: Ø 1 1 Ø Ø 1 Ø.2 1 2
Differences:         Ø 1 Ø.Ø 1 Ø 1.1 Ø Ø
        Source dot position
        Destination dot position
```

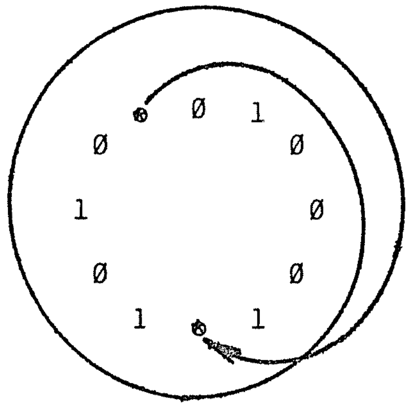
(a) A Sample Routing Problem (n = 1Ø; p = q = 4)



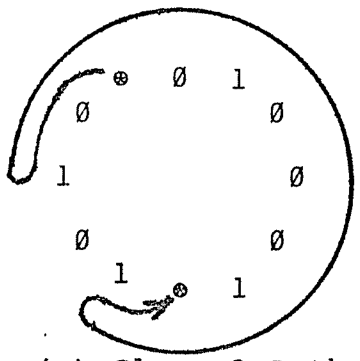
(b) Differences



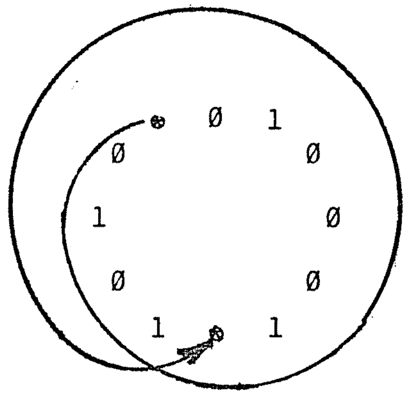
(c) Class-1 Paths



(d) Class-2 Paths



(e) Class-3 Paths



(f) Class-4 Paths

```
Ø 2 1.Ø 2 1 2 Ø 1 2
Ø 2.? Ø 2 1 2 Ø 1 2
Ø.? ? Ø 2 1 2 Ø 1 2
Ø 1.? Ø 2 1 2 Ø 1 2
Ø 1 1.Ø 2 1 2 Ø 1 2
Ø 1 1 Ø.2 1 2 Ø 1 2
Ø 1 1 Ø Ø.1 2 Ø 1 2
Ø 1 1 Ø Ø 1.2 Ø 1 2
Ø 1 1 Ø Ø 1 Ø.Ø 1 2
Ø 1 1 Ø Ø 1 Ø ?.1 2
Ø 1 1 Ø Ø 1 Ø.2 1 2
```

(g) Class-1 Paths

Figure 2

Routing in the Completed Lens

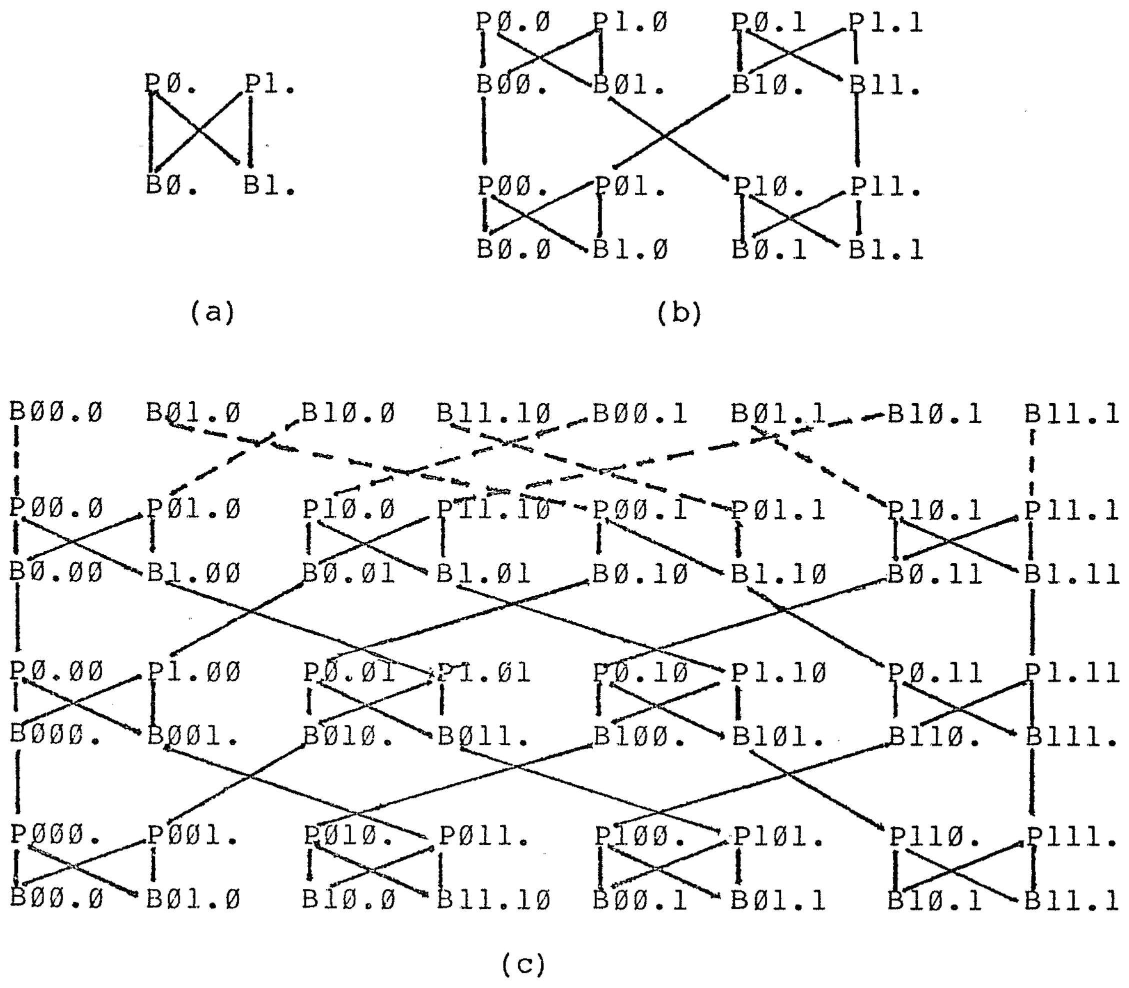


Figure 3

A Different Layout of the Lens

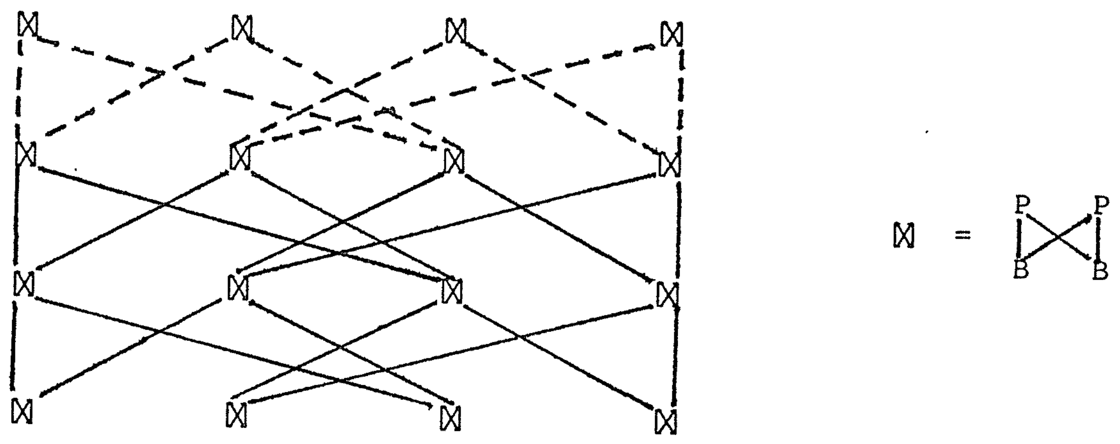


Figure 4

Schematic Version of Figure 3