"AN OPTICAL REFLECTIVE CONNECTION APPROACH

FOR MULTIPLE PROCESSOR ARRAYS"

by

JONATHAN F. GALLO

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master of Science in Engineering

in the

Electrical Engineering

Program

Prof. Samuel J. Skarote 3/9/94

Advisor Date

3/14/94

Dean of the Graduate School Date

YOUNGSTOWN STATE UNIVERSITY

March, 1994

# ABSTRACT

"AN OPTICAL REFLECTIVE CONNECTION APPROACH
FOR MULTIPLE PROCESSOR ARRAYS"

Jonathan F. Gallo
Master of Science in Engineering
Youngstown State University, 1994

The purpose of this thesis is to describe a microcomputer array interconnection system, configurable through optical links, such that computing power of a matrix of cells can be distributed optimally for changes in computational demands. Detail is given for the arrangement of the reflectors for free-space switching. Requirements for dealing with errors in construction of such a grid are discussed. A partial treatment is given for actual creation of the hardware, and an ideal device is described.

# ACKNOWLEDGEMENTS

I express my sincerest thanks to the following persons for their varied assistances through which I would not have been able to finally complete this paper: K.P. Moy, M.W. Allender, M. Pietros, P. Cetrone, P.N. Crook, A.F. Gallo, Dr. & Mrs. Gallo, and others who lent much needed support. I would especially like to thank my advisor, Professor Samuel J. Skarote, for his extensive patience and advice in this undertaking.

# TABLE OF CONTENTS

# LIST OF SYMBOLS

| SYMBOL | MEANING |
|---|---|
| $d, d_2, d_3$ | node-to-node distances |
| $f, f_i, f_r$ | plane reflection angles |
| $h, h_2$ | reflector heights |
| $x, y$ | array position values |
| {A,B,C,D,E} | node address matrixes |
| NOT,XOR,AND | combinational logic functions |
| $\alpha, \alpha_2, \alpha_3$ | reflector angles |
| $\beta$ | residual of error reflection |
| $\varepsilon, \varepsilon_1, \varepsilon_2$ | beam error angles |
| A, -A | unit-length reflection angles |
| B, -B | 2x-unit-length reflection angles |
| X, -X | 4x-unit-length reflection angles |

# LIST OF FIGURES

# LIST OF TABLES

# CHAPTER I

## INTRODUCTION

As clock speeds of computing systems increase, higher rates of information transfer are needed. Increasing amounts of data are subjected to more varied and complex types of processing. This is evident in the nature of current microcomputers. Where these used to only contain a central computing unit, now they have a main processing unit coupled with an associated numeric processor. Some systems have separate video computation as well as processor-based print technology. Communication channels and audio circuits can now perform sophisticated data manipulation with minimal reliance on the main processor. Evolution of such systems point to multiprocessor based computing, even within the context of the main processing section.

Parallelism of computing power and distributed functionality of processing both require communication amongst many processing cells. Throughput of such systems is dependent on the speed by which data gets transferred from point to point. Speed and density of point-to-point connection systems are limited by the amount of crosstalk or coupling between separate links. Optically connected electronics are attracting more interest, because light interconnection exhibits less electromagnetic interference than electronic methods.

Most small computer systems incorporate one or more paralleled-wire busses to transfer data between a single processing unit, its memory, and external devices. Multiprocessor designs typically utilize crossbar circuits to dynamically interconnect computing cells' inputs and outputs. Crossbar circuits are switch matrices typically implemented in physical wiring, integrated form, and, in some new designs, optical gates. As stated above, the fact that optical connections are more electromagnetically quiet than electronic ones gives hope that very high-speed and low-interference switching devices can be fabricated [1].

Basic optical connections consist of a transmitter coupled to a receiver through a transmission medium such as optical fiber or free space. A gating medium is also needed when using more than one connection, to link certain inputs with certain outputs. This can be accomplished through an optical coupling circuit, which routes some or all optical energy (or signal content) in one fiber to another. By using free-space links, wiring constraints associated with physical waveguides can be avoided, thus allowing a high degree of interconnection in computational grids. Free-space links imply point-to-point connections; thus some means for redirecting these vectors is needed, to provide for the physical positions of individual cells. Free-space routing can be accomplished through reflection or refraction by an intervening redirecting device, such as a lens, prism, refractive grating, reflector, or hologram.

This thesis explores one possible free-space routing device - a planar reflector array - by which such free-space redirection can be implemented. Chapter II discusses the basic structure of a computational array, and the resulting requirements for an associated reflector array. It assumes the computing array is organized in a square and planar configuration of processing cells, with parallel light beams orthogonal to its surface. This then requires an overhead array composed of reflective surfaces which perform the optical redirection.

The amount of optical redirection depends on the manner in which the computing cells are connected. Chapter III compares three interconnection methods, ring, full and N-cube. The N-cube structure is chosen for implementation in the thesis, due to its low maximum path distance, and its reasonable number of links. Chapter IV explores N-cube addressing and details its implementation in the context of the square array to be implemented.

Some errors associated with beam reflection, such as receiver/reflector shift and beam misalignment, are covered in Chapter V, while Chapter VI reveals details of the example reflective array actually built. Chapter VII describes an ideal system and some future work. Computer programs used in development of the hardware are listed in the Appendixes.

# CHAPTER II

## HARDWARE OVERVIEW

Figure 1, on page 7, shows one possible arrangement of a multiprocessor system, that of a planar array of identical computing cells. Each consists of a processor, local memory, and input/output section. The processors and memory are either optical or standard electronic circuits. Constructing an array from identical computing and memory modules reduces system complexity. The input section for each cell consists of a number of isolated light-receiving circuits (i.e. solar cell, phototransistor, or light junction), while the output section contains one or more light-transmitting circuits (diode, laser diode, laser, or light port.) All the output beams as well as the receiving areas are oriented normal to the computing array, so as to be pointed straight up into the face of an overhead reflecting surface. This reflecting surface could be constructed in several ways:

i) milled to redirect the incoming beams to their destinations;

ii) coated with a variable thickness of optically refractive material;

iii) or a planar reflector needing self output beam redirection.

These reflector constructions are shown in Figure 2. One further approach would have the reflector implemented as a

semiconductor hologram, possibly time-variant, to provide dynamic redirection of the output beams [2].

The function of the reflecting array is to provide switching between the output light beams and the input photosensors for each computing cell. In two of these configurations, the optical paths would be frozen into predetermined path links, implying the reflector surface would have to be physically changed to reconfigure the array. Both the planar reflector and the time-changing hologram might be able to dynamically reconfigure the array without direct physical manipulation of the structure. A planar reflector would require beam directors at the individual nodes, while the hologram approach would dynamically change its own reflective characteristics. This thesis does not discuss implementation of the holographic array plane, whose basic structure is outlined in [3].

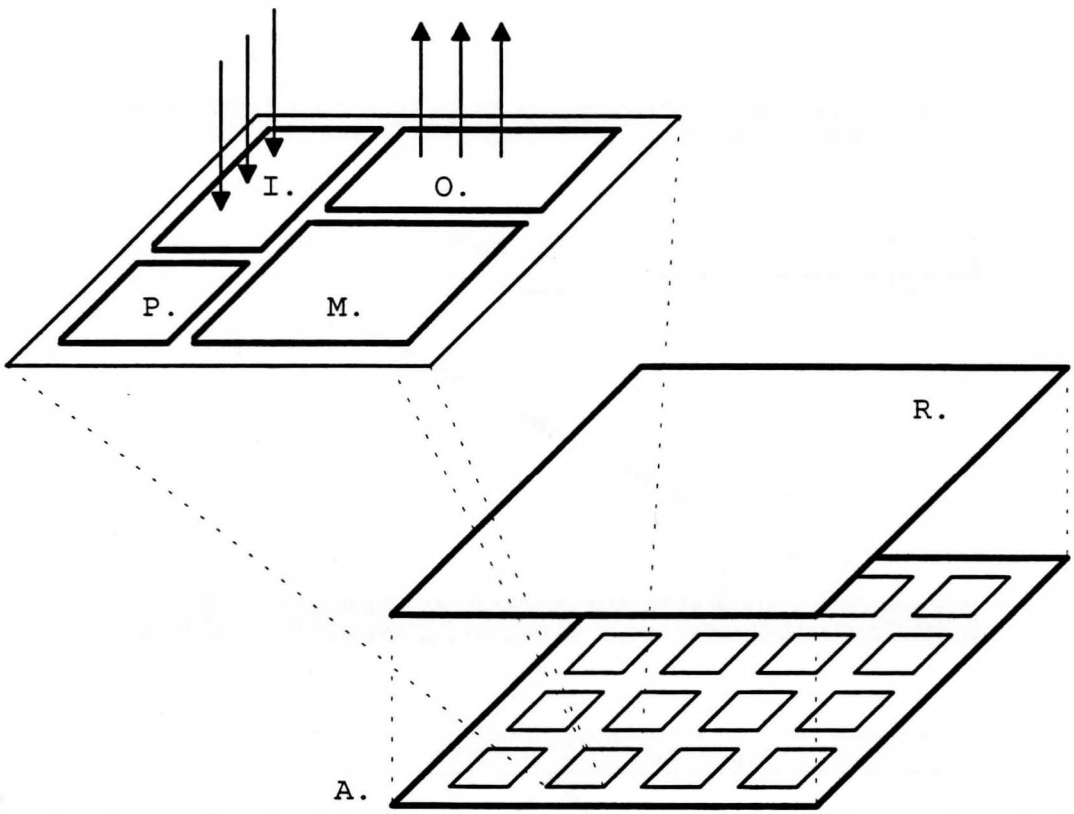
With a synchronized array, timing could be provided through the power supply leads, thus reducing the overall complexity of the system. If light gates are used, a common laser (diode) source could provide power [3], timing, and communication beams for the entire array. Uniformity in configuration simplifies node construction and programming.

This thesis assumes a milled reflector approach. Simplification of the system was considered a primary objective. Steered beams were viewed as making the beam hardware too complex, while the holgramatic approach would shift complexity to the reflector construction. However,

the basic concepts of addressing and layout are applicable to these reflector constructions.

One method of cell placement is to randomly place the cells on the 2-dimensional cell plane, storing the x-y coordinates of the transmitters and receivers for each cell. The method used by the software is to place the cells in an organized manner. This simplifies the mapping algorithm by reducing the number of reflecting plane angles involved. A drawback to symmetric cell placement is rigidity, as it does not easily allow for failure of individual elements, as does random placement.

A further software requirement, not implemented, is adjustment of the reflector's surface characteristic to correct for angular beam spreading and maximize optical energy at the corresponding receptor. Indeed, with the array implemented in a wafer structure, surface real estate available for sensors would be limited, thus requiring very accurate and adjusted beam redirection. This could be accomplished or aided by alteration of the individual transmitters to modify their beam characteristics, not limited to micromachined lenses for each transmitter, or corrective layers of refractive material. Binary step milling [4] holds the promise of very large numbers of tiny lens elements, for transmitter/receiver augmentation, and for direct lithography of the reflective surface itself.




A. Processor Array Plane.
I. Input Photosensors.
M. Node Memory.
O. Output Transmitters.
P. Node Processors.
R. Reflector Array Plane.

Figure 1. Basic Structure of Micro Array

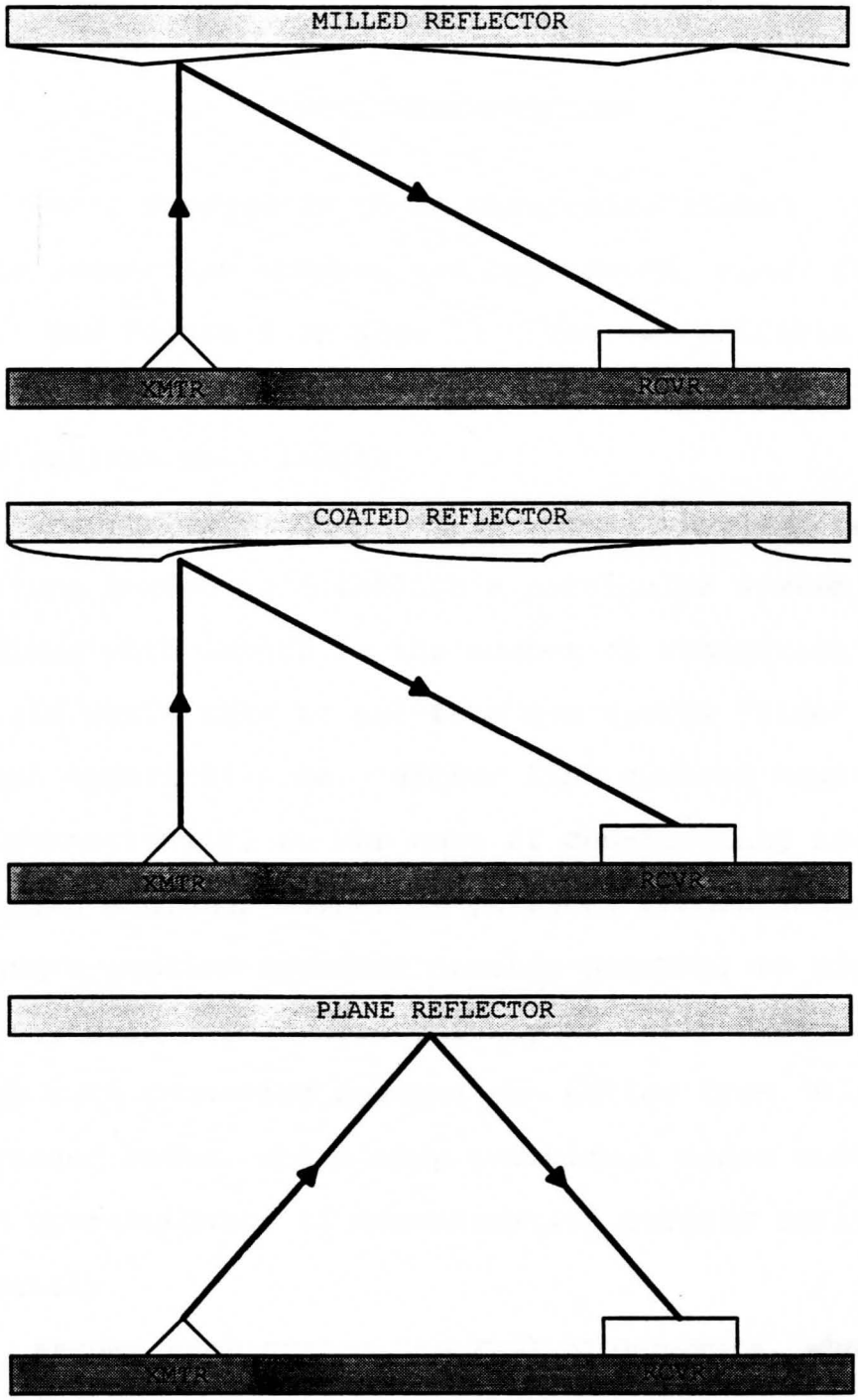



Figure 2. Various Reflector Array Constructions

# CHAPTER III

## CONNECTION OVERVIEW

For P processors to be physically linked, three possible connection schemes are considered; ring, full, and N-cube. See Figure 3 on page 11. The two criteria of connection comparison considered are the number of links, and the maximum path length.

The number of links is the total physical data connections needed to establish a particular system, while the maximum path length is the number of connection steps a data token would take to get from one system "side" to the "farthest opposite" side. Higher link numbers equate to higher connectivity, at the cost of constructing and maintaining a higher number of physical connections. Lower path length implies speedier message passing, as signal packages should arrive more quickly to their destinations, although such connected systems can suffer from "distracted" or overtaxed nodes, where many individual nodes must deal with an overabundance of communication packets arriving concurrently.

Assume each system has $P=2^{N-1}$ processors, where $N=\{1,2,3...\}$ as shown in Figure 3 on page 11.

A ring connection is just what its name implies - it connects all nodes into a giant ring, where each node is connected to only two other nodes. This scheme requires only

P links and has a maximum path length of P/2, this length having no back tracking of a packet. For large numbers of nodes, the path length becomes very long, as there is only one path (with possibly two directions on it).

The full connection ties each node to every other node. With high number of nodes, this becomes a daunting task. Although it has the smallest maximum path length of just 1, it needs

$$(P-1)+(P-2)+\ldots+2+1 = P(P-1)/2 \qquad (3.1)$$

links. As can be seen, as the number of processors rises, the maximum path length of the ring becomes extreme, while the number of links for the full case becomes excessive.

TABLE 1

COMPARISON OF THREE ARRANGEMENTS

| N | P | Number of Links | | | Maximum Path Delay | | |
|---|---|---|---|---|---|---|---|
| | | Ring | Full | N-cube | Ring | Full | N-cube |
| 1 | 2 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 4 | 6 | 4 | 2 | 1 | 2 |
| 3 | 8 | 8 | 28 | 12 | 4 | 1 | 3 |
| 4 | 16 | 16 | 120 | 32 | 8 | 1 | 4 |
| 5 | 32 | 32 | 496 | 80 | 16 | 1 | 5 |
| 6 | 64 | 64 | 2016 | 192 | 32 | 1 | 6 |
| 7 | 128 | 128 | 8128 | 448 | 64 | 1 | 7 |
| . | . | . | . | . | . | . | . |
| . | . | . | . | . | . | . | . |
| N | $P=2^N$ | P | P(P-1)/2 | N*P | P/2 | 1 | N |

The N-cube, or hypercube architecture [5] strikes a balance between these two. It starts from a single node and

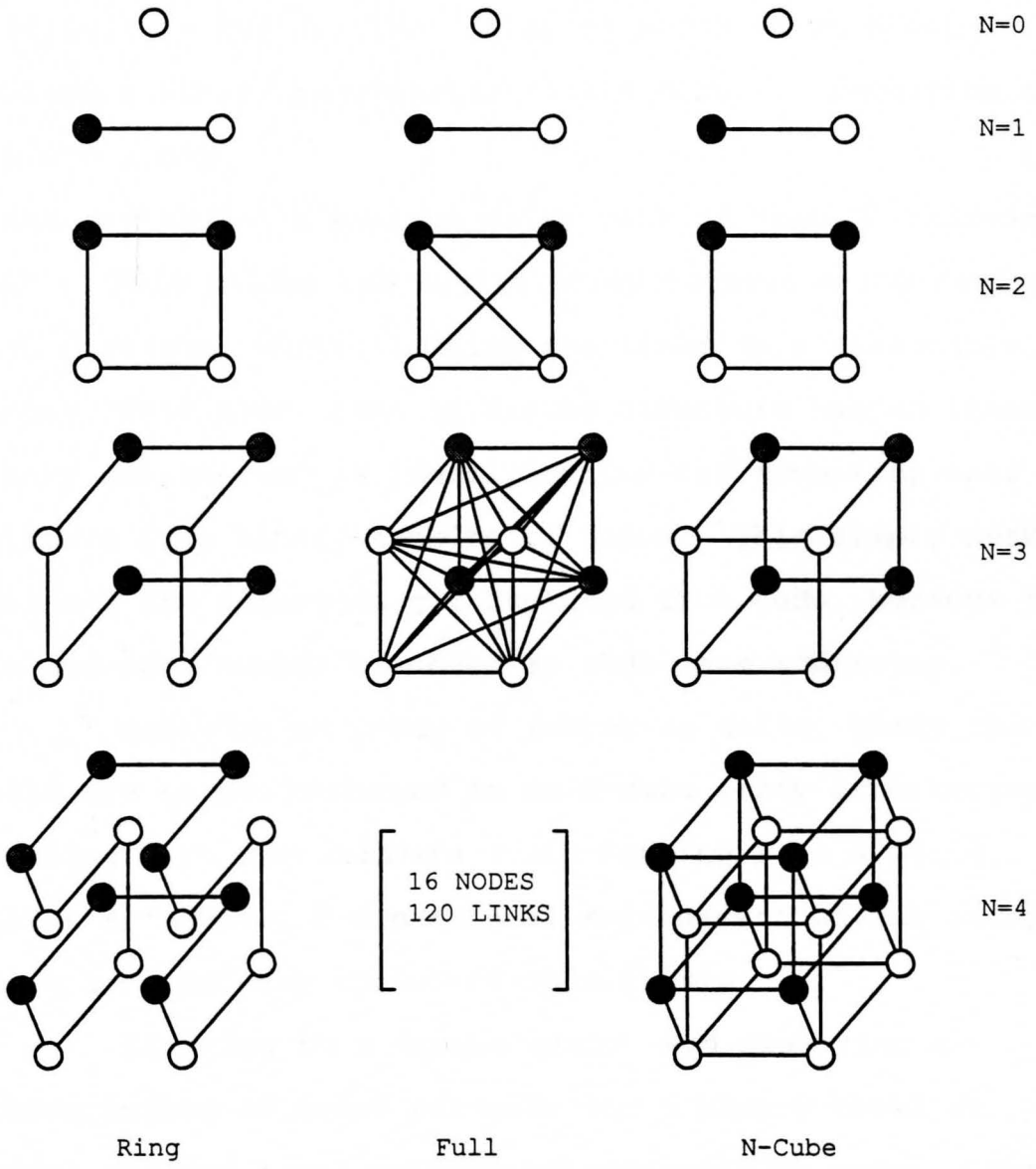



Figure 3. Ring, Full, and N-cube Constructions

simply replicates and connects itself to obtain higher form. As shown in Figure 3, each N-cube is a doubling of its previous size. Also shown in Figure 3 are ring and full connections for certain sized sets of nodes. One can see that the number of connections for the full system rapidly

becomes undrawable. In a wired system, it would be very difficult to build. The number of nodes in an N-cube follows a binary progression (1,2,4,8,16...) requiring only

$$N*2^{N-1} = N*P \quad (3.2)$$

links and having a maximum delay path of just N, assuming $P=2^{N-1}$. This allows the entire array to have a low maximal path distance, while limiting the links to a reasonable number. Note also, that an N-cube structure has an inherent binary nature, and is ideally suited for computing uses - it allows a full binary mapping of nodes. This simply means that all the addresses get assigned to a node, because there are the same number of nodes as there are addresses.

Assuming an array of computing cells, where the cells are to be connected in an N-cube array architecture, limiting N to even numbers (thus setting P to 4,16,64...) allows a "square" 2-dimensional arrangement of the array, which is a primary thrust of this thesis.

Limiting to a square array, and requiring a binary number of nodes per side (or a binary total of nodes), allows total mapping of addresses to nodes. Such limiting is chosen due to the 2-dimensional nature of the physical placement of this paper's device. This will be discussed in the following chapter. However, it should be noted that such limiting need not be utilized. Indeed, if there is not maximal mapping of addresses to nodes, open addresses might be used for other companion or control nodes.

# CHAPTER IV

## MAPPING CONSIDERATIONS

To demonstrate, assume a 2-cube of 4 processing cells in a square arrangement as shown in Figure 4. Each cell needs 2 transmitters and 2 receivers. Assume modular cells, where each transmitter and receiver have unique positions in the cell, but have identical positions from cell to cell. Assume further that the cells are addressed in a gray-code manner, meaning that neighboring nodes differ in address by only one bit. Physical mapping is found by assigning half of the cell address to the x-coordinate of the plane, and the other half to the y-coordinate, as shown in Table 2.

Since there are only 4 processors, there need only be 2 bits in each cell address, and hence only one bit affecting either physical dimension. Inverting one bit of a cell's address produces the address of an adjacent cell on the N-cube. (For this simple case, it so happens that all the N-cube adjacent cells are also adjacent physically on the coordinate plane.)

As can be seen in Figure 4, each cell is unit step from its two neighbors, thus (with good placement of its components), each receiver-transmitter pair is the same length. Thus, for a common height reflector, the angle of reflection will be the same for all links.

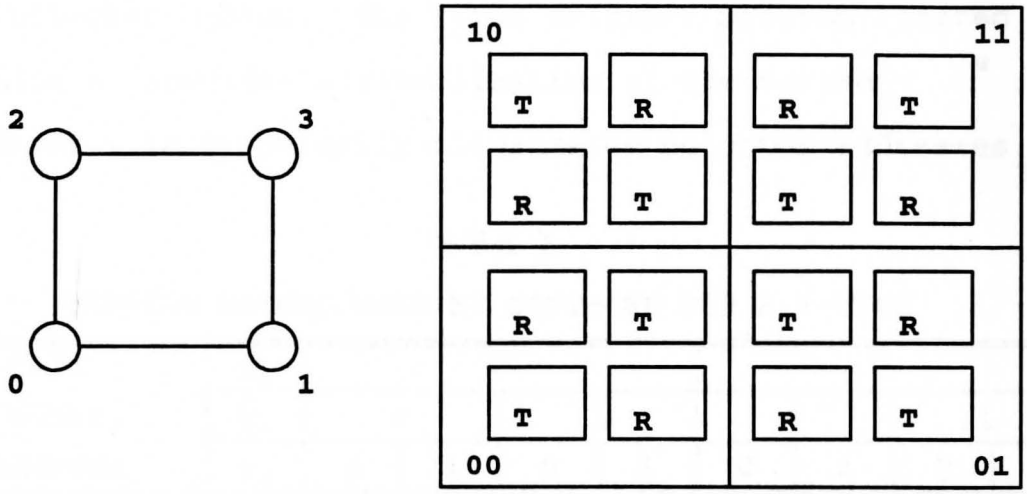



Figure 4. Mapping of a 2-cube

TABLE 2

CELL MAPPING INFORMATION FOR A 2-CUBE

| Cell | Cell Position | | Connected to: | |
|---|---|---|---|---|
| # | X | Y | cell# & cell# | |
| 0 | 0 | 0 | 1 | 2 |
| 1 | 0 | 1 | 0 | 3 |
| 2 | 1 | 0 | 0 | 3 |
| 3 | 1 | 1 | 1 | 2 |

Table 3 shows the physical mapping method used for the first program developed for this paper. Note that for the 6-cube ($2^6 = 64 = 8*8$) example, each cell has 6 nearest neighbors, with a 2-dimensional placement of all 64 cells. Note the nature of the nearest neighbors; each is still (in either the x or y direction), in the same column or row as the origin of that beam. This means that only the x or y

axis of the reflector has to be changed, thus simplifying the reflector design. The first software program, listed in Appendix A, provides a visualization of the 6-cube discussed. It graphically illustrates neighbor addresses.

TABLE 3

MAPPING METHOD USED BY SOFTWARE FOR A 6-CUBE

| Cell 6-Bit Address | | | $y_2$ | 0 | | | | 1 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $y_1$ | 0 | | 1 | | 0 | | 1 | |
| | | | $y_0$ | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| $x_2$ | $x_1$ | $x_0$ | | | | | | | | | |
| 1 | 1 | 1 | | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 |
| | | 0 | | 48 | 49 | 50 | 51 | 52 | 53 | 54 | 55 |
| | 0 | 1 | | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| | | 0 | | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| 0 | 1 | 1 | | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| | | 0 | | 16 | 17 | [illegible] | 19 | 20 | 21 | 22 | 23 |
| | 0 | 1 | | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| | | 0 | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

Inverting one bit of the address$\{y_2y_1y_0x_2x_1x_0\}$ of node 18{010010} produces six neighbors; 19{010011}, 16{010000}, 22{010110}, 26{011010}, 2{000010}, and 50{110010}. Angular displacement left-to-right is similar to the right-to-left displacement. Thus 18-to-19 reflection is the same as 18-to-26; 18-to-2 is the same as 18-to-16; and 18-to-22 is the same as 18-to-50. As derived in the next chapter, with a reflector height of h, and a lateral (x or y) displacement of d, see Figure 5:

$$A = \{\tan^{-1}[d/h]\}/2 \qquad (4.1)$$

In Tables 4 and 5, and Figure 5, A is the angle for left-to-right redirection of one cell length, while -A is the angle for right-to-left. B and -B refer to redirection of 2 unit lengths, while X and -X redirect 4 unit lengths.

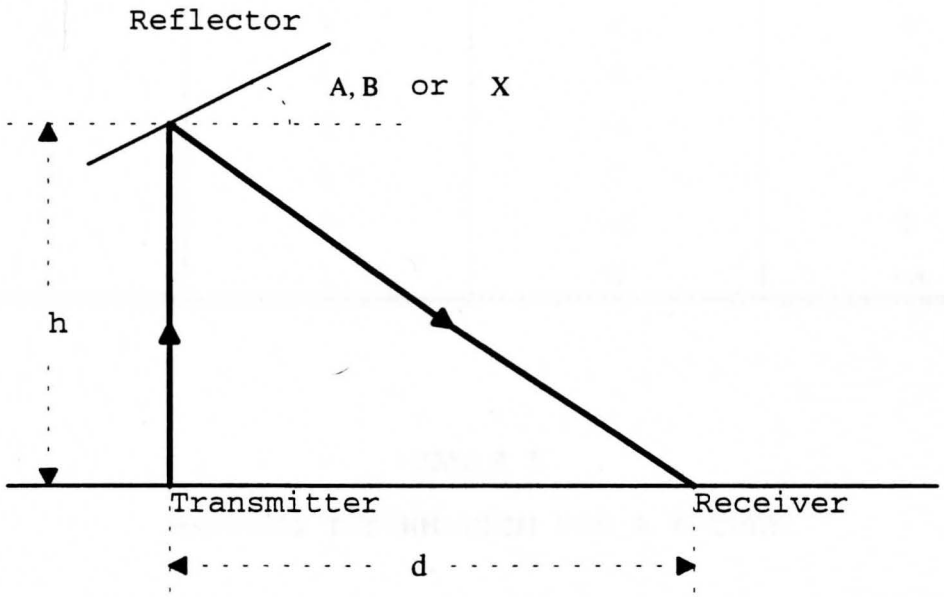



Figure 5. Reflector Angle Determination

With odd placement of any of the components, there will be instances when the simple case of reflection will not work. Figure 5 displays a side view, thus not indicating that the planar reflective area might need to be tilted in the other dimension (into or out of the page) to accommodate a two-axis reflection, occurring where there exists a conflict of assignment for two or more devices.

TABLE 4

REDIRECTION ANGLES FOR A 2-CUBE

| Cell | Receiver | A1(X-Axis) | A2(Y-Axis) |
|---|---|---|---|
| 0 | 0 | 0 | A |
| 0 | 1 | A | 0 |
| 1 | 0 | A | 0 |
| 1 | 1 | 0 | -A |
| 2 | 0 | -A | 0 |
| 2 | 1 | 0 | A |
| 3 | 0 | -A | 0 |
| 3 | 1 | 0 | -A |

TABLE 5

MAPPING INFORMATION FOR A 6-CUBE

| # | X-ADDRESS | | | Y-ADDRESS | | | N1 | N2 | N3 | N4 | N5 | N6 | A | B | X | A | B | X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 32 | 16 | 8 | 4 | 2 | 1 | + | + | + | + | + | + |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 33 | 17 | 9 | 5 | 3 | 0 | + | + | + | + | + | - |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 | 34 | 18 | 10 | 6 | 0 | 3 | + | + | + | + | - | + |
| 3 | 0 | 0 | 0 | 0 | 1 | 1 | 35 | 19 | 11 | 7 | 1 | 2 | + | + | + | + | - | - |
| 4 | 0 | 0 | 0 | 1 | 0 | 0 | 36 | 20 | 12 | 0 | 6 | 5 | + | + | + | - | + | + |
| 5 | 0 | 0 | 0 | 1 | 0 | 1 | 37 | 21 | 13 | 1 | 7 | 4 | + | + | + | - | + | - |
| 6 | 0 | 0 | 0 | 1 | 1 | 0 | 38 | 22 | 14 | 2 | 4 | 7 | + | + | + | - | - | + |
| 7 | 0 | 0 | 0 | 1 | 1 | 1 | 39 | 23 | 15 | 3 | 5 | 6 | + | + | + | - | - | - |
| 8 | 0 | 0 | 1 | 0 | 0 | 0 | 40 | 24 | 0 | 12 | 10 | 9 | + | + | - | + | + | + |
| 9 | 0 | 0 | 1 | 0 | 0 | 1 | 41 | 25 | 1 | 13 | 11 | 8 | + | + | - | + | + | - |
| 10 | 0 | 0 | 1 | 0 | 1 | 0 | 42 | 26 | 2 | 14 | 8 | 11 | + | + | - | + | - | + |
| 11 | 0 | 0 | 1 | 0 | 1 | 1 | 43 | 27 | 3 | 15 | 9 | 10 | + | + | - | + | - | - |
| 12 | 0 | 0 | 1 | 1 | 0 | 0 | 44 | 28 | 4 | 8 | 14 | 13 | + | + | - | - | + | + |
| 13 | 0 | 0 | 1 | 1 | 0 | 1 | 45 | 29 | 5 | 9 | 15 | 12 | + | + | - | - | + | - |

TABLE 5 (Continued)

| # | X-ADDRESS | | | Y-ADDRESS | | | N1 | N2 | N3 | N4 | N5 | N6 | A | B | X | A | B | X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 14 | 0 | 0 | 1 | 1 | 1 | 0 | 46 | 30 | 6 | 10 | 12 | 15 | + | + | − | − | − | + |
| 15 | 0 | 0 | 1 | 1 | 1 | 1 | 47 | 31 | 7 | 11 | 13 | 14 | + | + | − | − | − | − |
| 16 | 0 | 1 | 0 | 0 | 0 | 0 | 48 | 0 | 24 | 20 | 18 | 17 | + | − | + | + | + | + |
| 17 | 0 | 1 | 0 | 0 | 0 | 1 | 49 | 1 | 25 | 21 | 19 | 16 | + | − | + | + | + | − |
| 18 | 0 | 1 | 0 | 0 | 1 | 0 | 50 | 2 | 26 | 22 | 16 | 19 | + | − | + | + | − | + |
| 19 | 0 | 1 | 0 | 0 | 1 | 1 | 51 | 3 | 27 | 23 | 17 | 18 | + | − | + | + | − | − |
| 20 | 0 | 1 | 0 | 1 | 0 | 0 | 52 | 4 | 28 | 16 | 22 | 21 | + | − | + | − | + | + |
| 21 | 0 | 1 | 0 | 1 | 0 | 1 | 53 | 5 | 29 | 17 | 23 | 20 | + | − | + | − | + | − |
| 22 | 0 | 1 | 0 | 1 | 1 | 0 | 54 | 6 | 30 | 18 | 20 | 23 | + | − | + | − | − | + |
| 23 | 0 | 1 | 0 | 1 | 1 | 1 | 55 | 7 | 31 | 19 | 21 | 22 | + | − | + | − | − | − |
| 24 | 0 | 1 | 1 | 0 | 0 | 0 | 56 | 8 | 16 | 28 | 26 | 25 | + | − | − | + | + | + |
| 25 | 0 | 1 | 1 | 0 | 0 | 1 | 57 | 9 | 17 | 29 | 27 | 24 | + | − | − | + | + | − |
| 26 | 0 | 1 | 1 | 0 | 1 | 0 | 58 | 10 | 18 | 30 | 24 | 27 | + | − | − | + | − | + |
| 27 | 0 | 1 | 1 | 0 | 1 | 1 | 59 | 11 | 19 | 31 | 25 | 26 | + | − | − | + | − | − |
| 28 | 0 | 1 | 1 | 1 | 0 | 0 | 60 | 12 | 20 | 24 | 30 | 29 | + | − | − | − | + | + |
| 29 | 0 | 1 | 1 | 1 | 0 | 1 | 61 | 13 | 21 | 25 | 31 | 28 | + | − | − | − | + | − |
| 30 | 0 | 1 | 1 | 1 | 1 | 0 | 62 | 14 | 22 | 26 | 28 | 31 | + | − | − | − | − | + |
| 31 | 0 | 1 | 1 | 1 | 1 | 1 | 63 | 15 | 23 | 27 | 29 | 30 | + | − | − | − | − | − |
| 32 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 48 | 40 | 36 | 34 | 33 | − | + | + | + | + | + |
| 33 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 49 | 41 | 37 | 35 | 32 | − | + | + | + | + | − |
| 34 | 1 | 0 | 0 | 0 | 1 | 0 | 2 | 50 | 42 | 38 | 32 | 35 | − | + | + | + | − | + |
| 35 | 1 | 0 | 0 | 0 | 1 | 1 | 3 | 51 | 43 | 39 | 34 | 34 | − | + | + | + | − | − |
| 36 | 1 | 0 | 0 | 1 | 0 | 0 | 4 | 52 | 44 | 32 | 38 | 37 | − | + | + | − | + | + |
| 37 | 1 | 0 | 0 | 1 | 0 | 1 | 5 | 53 | 45 | 33 | 39 | 36 | − | + | + | − | + | − |
| 38 | 1 | 0 | 0 | 1 | 1 | 0 | 6 | 54 | 46 | 34 | 36 | 39 | − | + | + | − | − | + |
| 39 | 1 | 0 | 0 | 1 | 1 | 1 | 7 | 55 | 47 | 35 | 37 | 38 | − | + | + | − | − | − |
| 40 | 1 | 0 | 1 | 0 | 0 | 0 | 8 | 56 | 32 | 44 | 42 | 41 | − | + | − | + | + | + |
| 41 | 1 | 0 | 1 | 0 | 0 | 1 | 9 | 57 | 33 | 45 | 43 | 40 | − | + | − | + | + | − |
| 42 | 1 | 0 | 1 | 0 | 1 | 0 | 10 | 58 | 34 | 46 | 40 | 43 | − | + | − | + | − | + |
| 43 | 1 | 0 | 1 | 0 | 1 | 1 | 11 | 59 | 35 | 47 | 41 | 42 | − | + | − | + | − | − |
| 44 | 1 | 0 | 1 | 1 | 0 | 0 | 12 | 60 | 36 | 40 | 46 | 45 | − | + | − | − | + | + |

TABLE 5 (Continued)

| # | X-ADDRESS | | | Y-ADDRESS | | | N1 | N2 | N3 | N4 | N5 | N6 | A | B | X | A | B | X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 45 | 1 | 0 | 1 | 1 | 0 | 1 | 13 | 61 | 37 | 41 | 47 | 44 | - | + | - | - | + | - |
| 46 | 1 | 0 | 1 | 1 | 1 | 0 | 14 | 62 | 38 | 42 | 44 | 47 | - | + | - | - | - | + |
| 47 | 1 | 0 | 1 | 1 | 1 | 1 | 15 | 63 | 39 | 43 | 45 | 46 | - | + | - | - | - | - |
| 48 | 1 | 1 | 0 | 0 | 0 | 0 | 16 | 32 | 56 | 52 | 50 | 49 | - | - | + | + | + | + |
| 49 | 1 | 1 | 0 | 0 | 0 | 1 | 17 | 33 | 57 | 53 | 51 | 48 | - | - | + | + | + | - |
| 50 | 1 | 1 | 0 | 0 | 1 | 0 | 18 | 34 | 58 | 54 | 48 | 51 | - | - | + | + | - | + |
| 51 | 1 | 1 | 0 | 0 | 1 | 1 | 19 | 35 | 59 | 55 | 49 | 50 | - | - | + | + | - | - |
| 52 | 1 | 1 | 0 | 1 | 0 | 0 | 20 | 36 | 60 | 48 | 54 | 53 | - | - | + | - | + | + |
| 53 | 1 | 1 | 0 | 1 | 0 | 1 | 21 | 37 | 61 | 49 | 55 | 52 | - | - | + | - | + | - |
| 54 | 1 | 1 | 0 | 1 | 1 | 0 | 22 | 38 | 62 | 50 | 52 | 55 | - | - | + | - | - | + |
| 55 | 1 | 1 | 0 | 1 | 1 | 1 | 23 | 39 | 63 | 51 | 53 | 54 | - | - | + | - | - | - |
| 56 | 1 | 1 | 1 | 0 | 0 | 0 | 24 | 40 | 48 | 60 | 58 | 57 | - | - | - | + | + | + |
| 57 | 1 | 1 | 1 | 0 | 0 | 1 | 25 | 41 | 49 | 61 | 59 | 56 | - | - | - | + | + | - |
| 58 | 1 | 1 | 1 | 0 | 1 | 0 | 26 | 42 | 50 | 62 | 56 | 59 | - | - | - | + | - | + |
| 59 | 1 | 1 | 1 | 0 | 1 | 1 | 27 | 43 | 51 | 63 | 57 | 58 | - | - | - | + | - | - |
| 60 | 1 | 1 | 1 | 1 | 0 | 0 | 28 | 44 | 52 | 56 | 62 | 61 | - | - | - | - | + | + |
| 61 | 1 | 1 | 1 | 1 | 0 | 1 | 29 | 45 | 53 | 57 | 63 | 60 | - | - | - | - | + | - |
| 62 | 1 | 1 | 1 | 1 | 1 | 0 | 30 | 46 | 54 | 58 | 60 | 63 | - | - | - | - | - | + |
| 63 | 1 | 1 | 1 | 1 | 1 | 1 | 31 | 47 | 55 | 59 | 61 | 62 | - | - | - | - | - | - |

The second computer program listed in Appendix B, computes the above data for the computing cells in the 8x8 arrangement of the 6-cube of Table 3. For the reflection information, each beam redirection needs two angular displacements, one for each dimension of the array. The software finds the x and y values of displacement. Each step in either the x or y direction is unit distance from the regular arrangement of the grid. This means that each reflection is a combination of the inter-cell spacing and

the spacing between transmitter and receiver. In such an arrangement, the address of each node contains:

i) that node's position (x,y) on the plane array.

ii) the addresses of the node's nearest neighbors.

iii) direction information to reach each neighbor.

As in the case of the 2-cube, it can be seen that the first half of the address is the x position of that cell, while the other half of the address is the y position of the cell.

Again, to find the nearest neighbors of any cell, one bit of its address is inverted to find one neighbor. This works if a gray-code addressing scheme is used for adjacent node addresses. As in the case of the 2-cube, there will only be 2 directions for reflector orientation. This is due to the 2-dimensionality in physical layout. However, now there will be **3** possible angles for the reflecting surface. The software shows two of each angle A, B, and X for each node, one in a vertical and one in a horizontal direction. It can be seen that the direction (+/- for increasing/decreasing x or y), will mimic the bits of the node address, where a 1 bit of address corresponds to a negative reflection angle. From a binary perspective, this makes absolute sense, as it can be seen that by inverting only one bit from a one to a zero changes the address negatively by some power of 2. A zero in the address will cause a positive address change by some power of 2, thus requiring a positive reflection angle. Since

half the bits (in this case, 3 bits) of an address control the x-axis placement of the node, there will be three addresses in the same row as the original node, each differing in address by 1, 2, and 4. Correspondingly, the other half of the address affects the y-axis location, thus the other three neighbor nodes occur in the same column, differing from the original address by 8, 16, and 32. Referring to Table 3, note the orientations of the highlighted cells.

Thus, for the 6-cube discussed, the address is of the form:

$$\{y_2 y_1 y_0 x_2 x_1 x_0\} \tag{4.2}$$

giving the square as shown in Table 3, and in general, for an N-cube of the square type:

$$\{y_{(N/2)-1}\, y_{(N/2)-2} \cdots y_2\, y_1\, y_0\, x_{(N/2)-1}\, x_{(N/2)-2} \cdots x_2\, x_1\, x_0\} \tag{4.3}$$

where N=2,4,6,8...

In a general form, {A} is substituted for the {yx} form of the address, implying a source address for a node. The destination address {D} of the same form is supplied by the sending node. Exclusively OR'ing them will produce the intermediate product {B}:

$$\{B\} = \{A\} \text{ XOR } \{D\} \tag{4.4}$$

which is used to modify another term {E}, which starts out as zeroed array (, {E} = {..000..}.) {B} is scanned in some predetermined order to find a digit one value (1). At the point that the first 1 is found, the corresponding bit in {E} must be asserted, and scanning of {B} stopped:

$$\{E\} = f(\{B\}) = f(\{A\},\{D\}) \quad (4.5)$$

This changed {E} term is then XOR'ed with the original source address, to produce the node address which the packet should be sent to next:

$$\{C\} = \{E\} \text{ XOR } \{A\} \quad (4.6)$$

which allows any node to send along a message packet in the correct direction within the array. The final function for the address {C} is dependent on the manner in which scanning is performed on the intermediate term {B}. Scanning from right to left will produce the following logic:

$$\text{IF } B_0 = 1 \text{ then } \{E\} = \{..001\} \quad (4.7)$$
$$\text{ELSE IF } B_1 = 1 \text{ then } \{E\} = \{..010\} \quad (4.8)$$
$$\text{ELSE IF } B_2 = 1 \text{ then } \{E\} = \{..100\} \quad (4.9)$$
$$\ldots\ldots$$
$$\text{OTHERWISE } \{E\} = \{..000\} \quad (4.10)$$

By placing the terms into a Karnaugh map, the terms for {E} can be found to be:

$$E_0 = B_0 \quad (4.11)$$
$$E_1 = \text{NOT}(B_0)B_1 \quad (4.12)$$
$$E_2 = \text{NOT}(B_0)\text{NOT}(B_1)B_2 \quad (4.13)$$
$$\ldots\ldots$$
$$E_N = \text{NOT}(B_0)\text{NOT}(B_1)\ldots\text{NOT}(B_{N-1})B_N \quad (4.14)$$

This form of {E} can be manipulated into a simpler form by recognizing that:

$$\{B\} = \{..B_3B_2B_1B_0\} \quad (4.15)$$
$$2\{B\} = \{..B_2B_1B_00\} \quad (4.16)$$
$$4\{B\} = \{..B_1B_000\} \quad (4.17)$$
$$\ldots\ldots$$
$$2^n\{B\} = \{..B_1B_0000000 \text{<-- n zeros --}>00000\} \quad (4.18)$$

which only means that by NOT'ing all but the first of the above functions will produce necessary terms which can be AND'ed together to form {E}. Of course, there will exist higher valued terms outside the range of bits needed, as

multiplication (even by 2,) will produce numbers larger than the initial value. These terms can be eliminated by simply AND'ing the final product with $2^n$, where n is the number of bits of resolution in the source address {A}.

The third computer program, listed in Appendix C, produces passing addresses by using the procedure just described. The experimental program generates random addresses for a 4-cube array, stacking simulated data tokens for each node. No output of the program is shown or discussed in this paper.

The above discussion gave the procedure to follow if software is utilized to compute next-neighbor-to-pass-to addresses for token passing. If hardware is built, the equations (4.4) through (4.6) can be implemented through the use of combinational logic gates, as shown in Figure 6.

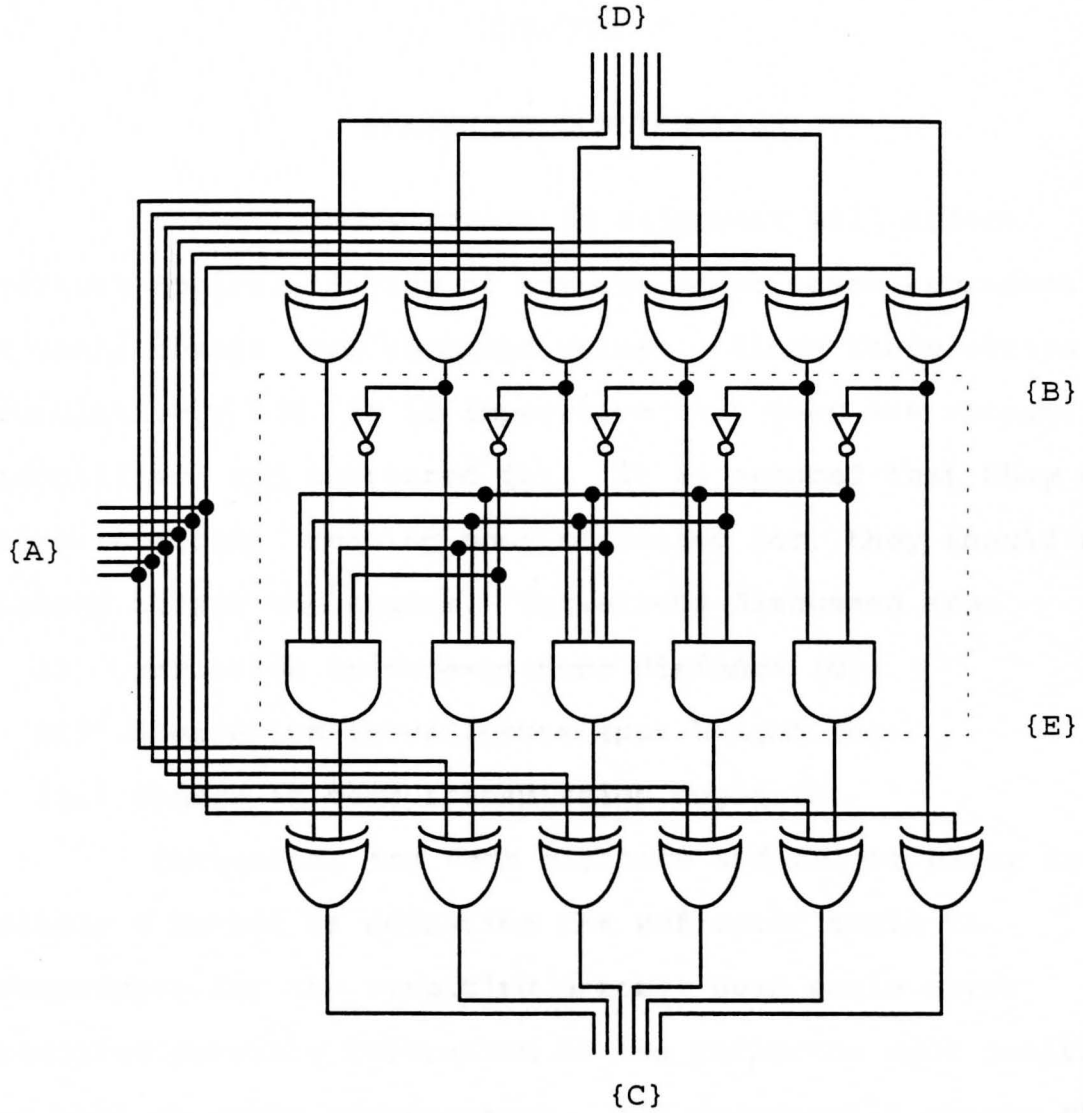



Figure 6. Combinational Logic Circuit

# CHAPTER V

## ERRORS IN CONSTRUCTION

Three primary errors in alignment will affect correct optical linking of any two nodes, each representing a small change from computed values. Since these errors cumulatively add, it is imperative that they are recognized, identified, and corrected for. It is assumed that they are time-invariant, meaning once corrected for, they should no longer affect the system. Variations discussed are:

i) variation in node-to-node distance (d);

ii) aberration in reflector spot height (h);

iii) change in beam transmission angle ($\varepsilon$).

Correcting for both distance and height error is simply a matter of adjusting the reflector angle to compensate for the resulting error. Beam angle error requires possible relocation of the reflector spot position as well as angle compensation. All errors will change the spot size of the beam at the receiver, so dispersion correction (or beam angle error correction) is necessary if the receiver area or sensitivity are critically small.

For the following, refer to Figure 7. Assuming the reflective spot a linear reflector, the incident and reflected angles will be equal ($f_i = f_r = f$). From the diagram, for the calculation of the reflector angle:

$$f = \varepsilon + \alpha \qquad (5.1)$$

$$2f = \varepsilon + \beta \tag{5.2}$$

and by solving for $\alpha$:

$$\alpha = (\beta - \varepsilon)/2 \quad = \quad \beta/2 - \varepsilon/2 \tag{5.3}$$

or:

$$\alpha = \{\tan^{-1}[(d-x)/h] - \varepsilon\}/2 \tag{5.4}$$

When $x = 0$ and $\varepsilon = 0$:

$$\alpha = \{\tan^{-1}[d/h]\}/2 \tag{5.5}$$

as was indicated in equation 4.1

With a vertical beam, symmetric changes in both d and h imply no change in reflector angle. Thus, a 10% change in d and h keeps the angle the same.

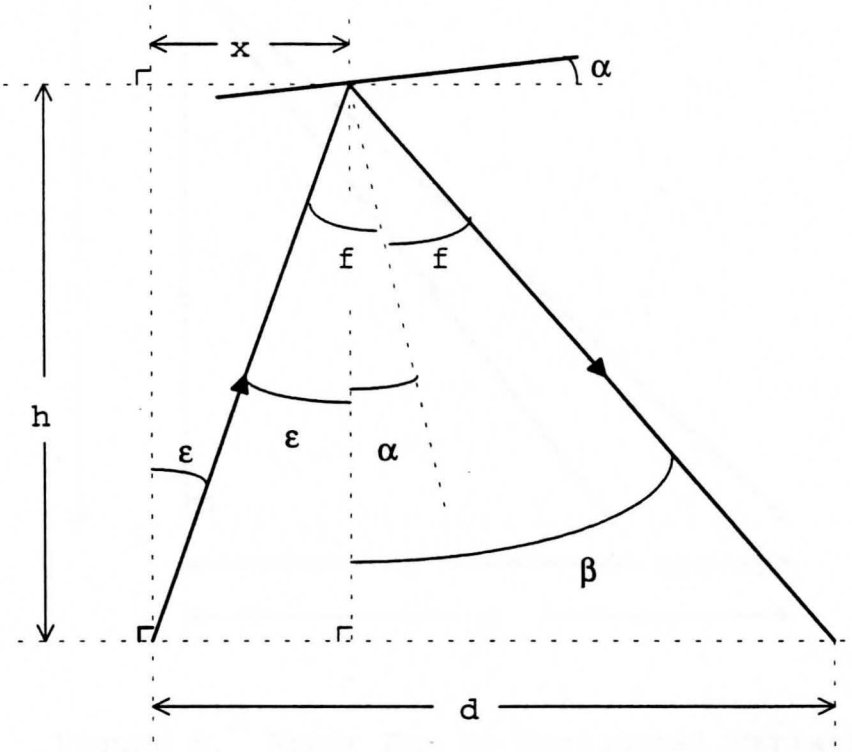



Figure 7. Detail of Reflector Angle Determination

## 5.1 NODE-TO-NODE DISTANCE ERROR

As indicated in Figure 8, a variation in distance between nodes will cause the receiver position to deviate from the intended spot landing area, thus corrupting or impeding signal reception. A small change in distance, err(d) will require either relocation of the receiver to its original space, or correction of the reflective angle:

$$d_2 = err(d) + d \tag{5.6}$$

$$\alpha_2 = \{\tan^{-1}[d_2/h]\}/2 \tag{5.7}$$

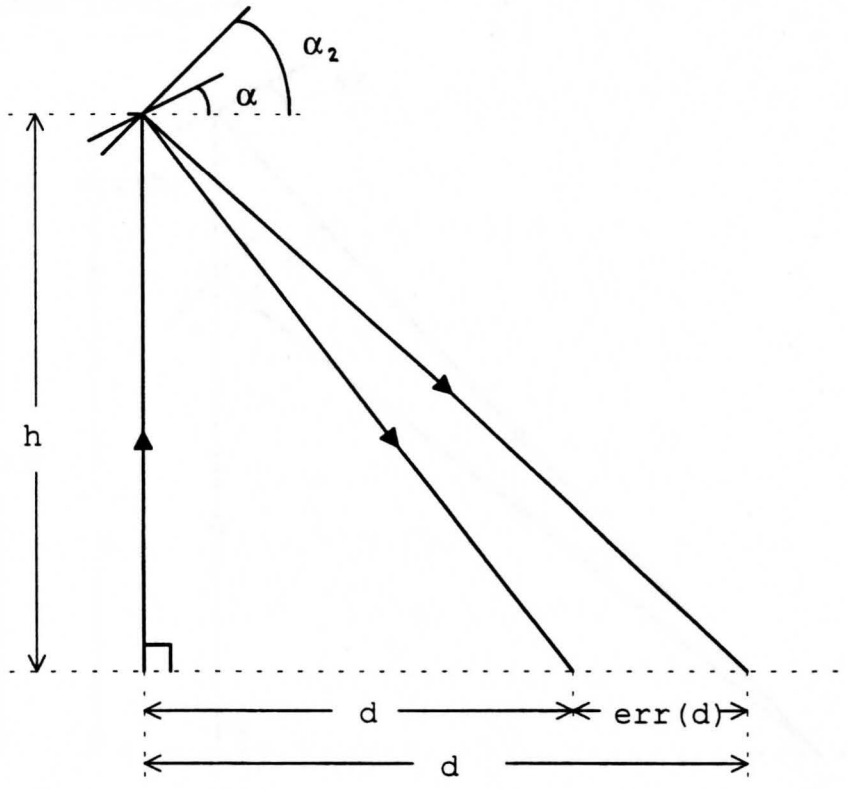



Figure 8. Error Due to Horizontal Variation

## 5.2 REFLECTOR HEIGHT ERROR

A deviation in reflector height would be caused by bending, warpage, or some local surface discontinuity of the reflective surface. Although not anticipated to occur, such an error in height, err(h), would result in overshooting or undershooting of the intended receiver. Compensating for a change in surface-to-reflector height requires corrective milling of the reflective spot such that its angle is decreased for an increase in height, as per Fig. 9:

$$h_2 = err(h) + h \tag{5.8}$$

$$\alpha_2 = \{\tan^{-1}[d/h_2]\}/2 \tag{5.9}$$

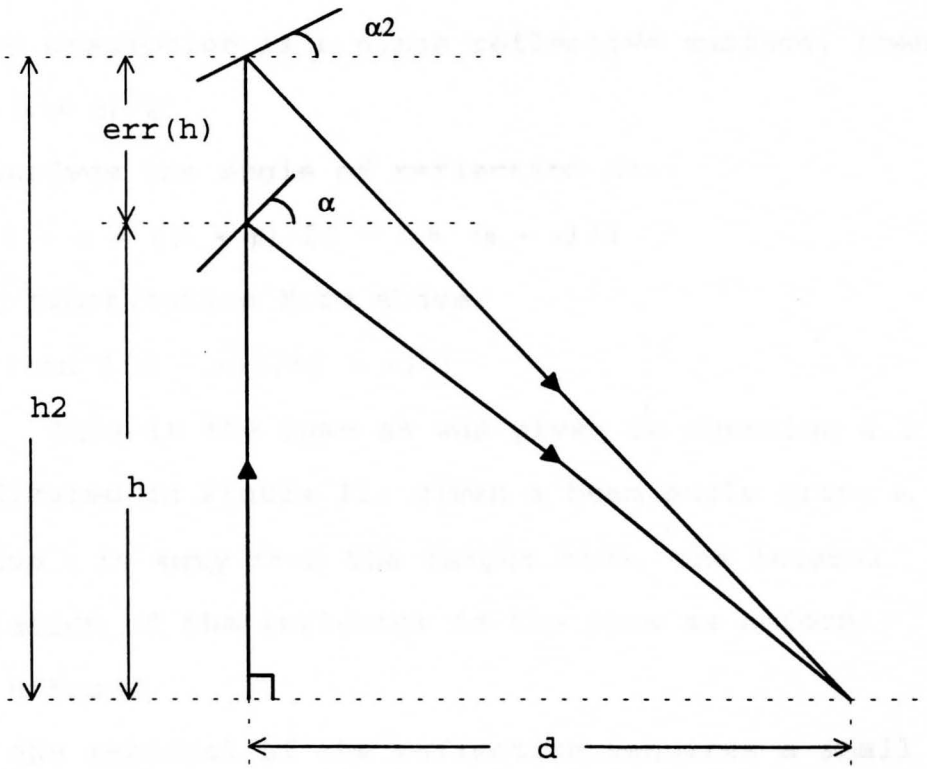



Figure 9. Error Due to Vertical Variation

## 5.3 BEAM ANGLE ERROR

It is expected that if the transmitters located at each node are discrete devices, there will exist the likelihood of the beams not being entirely normal to the surface of the plane.

As shown in Figure 10, given a beam angle error $\varepsilon$, where positive $\varepsilon$ is towards the target node, the lateral translation of the reflector is:

$$x = h*\tan(\varepsilon) \tag{5.10}$$

while the residual of the reflection is:

$$\beta = \tan^{-1}((d - x)/h) \tag{5.11}$$

and the assumption is a plane reflective surface, then:

$$f = (\varepsilon + \beta)/2 \tag{5.12}$$

This derives the angle of reflection as:

$$\alpha = f - \varepsilon = ((\varepsilon + \beta)/2) - \varepsilon = (\beta - \varepsilon)/2 \tag{5.13}$$

or, by substituting from above:

$$\alpha = \{\tan^{-1}[(d - x))/h] - \varepsilon\}/2 \tag{5.14}$$

This is the same as was given in equation 4.1 Now, as indicated in Figure 11, given a beam angle error $\varepsilon$, where positive $\varepsilon$ is **away** from the target node, the lateral translation of the reflector is the same as before:

$$x = h*\tan(\varepsilon) \tag{5.15}$$

while the residual of the reflection requires a small change from subtraction to addition:

$$\beta = \tan^{-1}((d + x)/h) \tag{5.16}$$

while again, the assumption is a plane reflective surface,

there is another change in summation:

$$f = (\beta - \varepsilon)/2 \tag{5.17}$$

deriving the angle of reflection to:

$$\alpha = f + \varepsilon = ((\beta - \varepsilon)/2) + \varepsilon = (\beta + \varepsilon)/2 \tag{5.18}$$

or, by substituting from above:

$$\alpha = \{\tan^{-1}[(d + x)/h] + \varepsilon\}/2 \tag{5.19}$$

The general form now becomes, where positive $\varepsilon$ is towards, and negative $\varepsilon$ is away from the target node:

$$\alpha = [\tan^{-1}\{(d - h*\tan(\varepsilon))/h\} + \varepsilon]/2 \tag{5.20}$$

Note that with zero beam error, the beam is normal to horizontal:

$$\varepsilon = 0 \tag{5.21}$$

and the angle simplifies to that in equation 5.4:

$$\alpha = [\tan^{-1}\{d/h\}]/2 \tag{5.22}$$

Dynamic errors in the values just discussed are more difficult to compensate for, as they will corrupt more than one node-to-node connection linkage. Small tilts, shifts, or vibrations of the reflector will affect almost all communicating nodes. Thermal changes cause expansion and contraction of both the reflector supporting structure, as well as the circuit substrate, thus introducing the above errors. Gross tiltage or movement of the entire reflective array must not occur, as this will result in massive disruption of almost all connections.

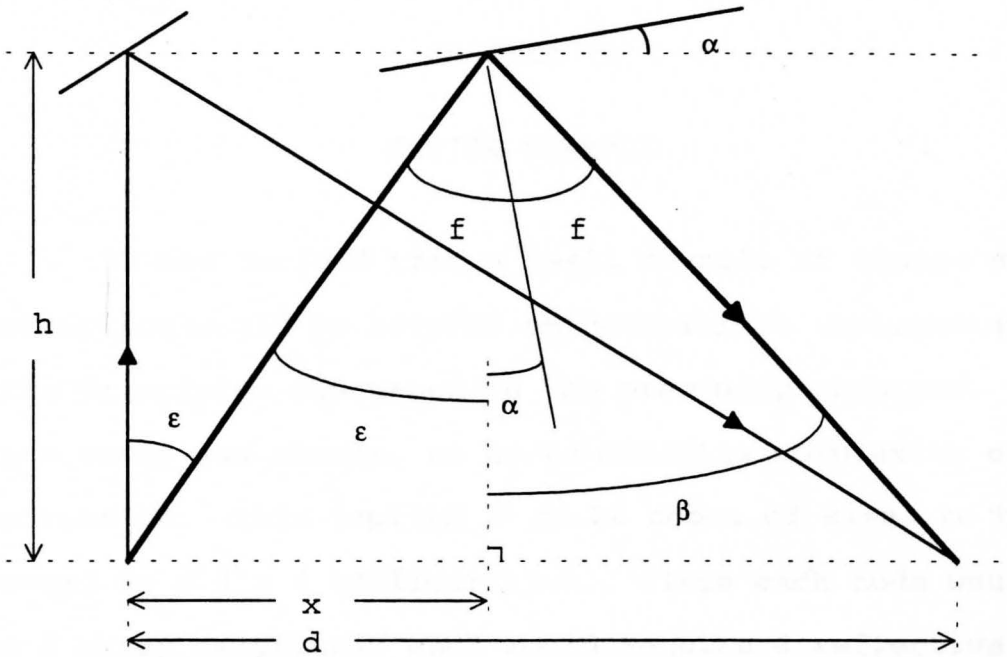



Figure 10. Error - Positive Beam Angle

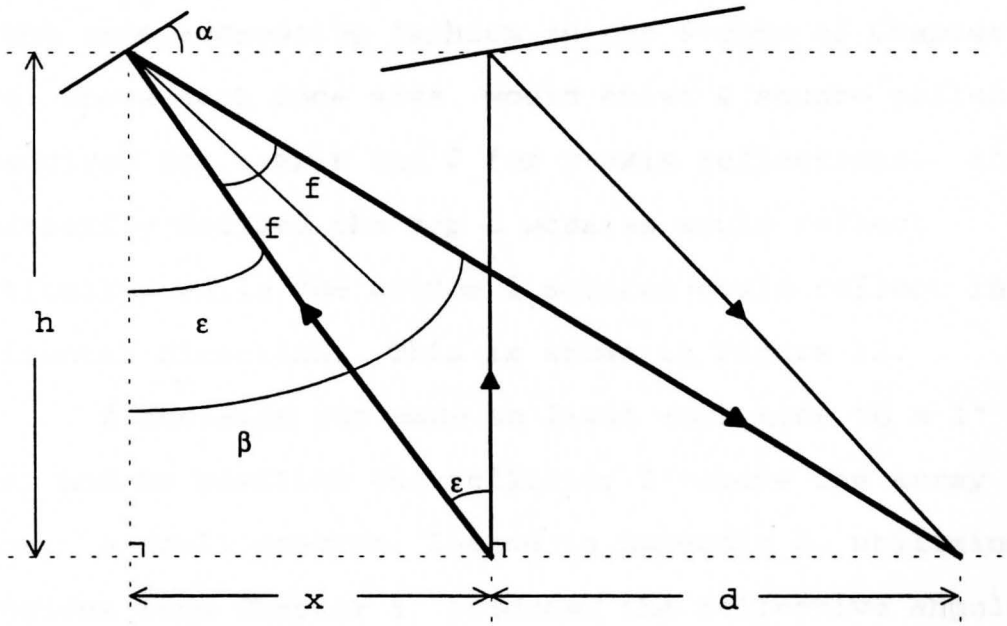



Figure 11. Error - Negative Beam Angle

# CHAPTER VI

## DESIGN EXAMPLE

It was decided that a small example of design and construction would be helpful to demonstrate implementation of the principles discussed in the preceding chapters. A 4-cube array was chosen, so as to minimize complexity of construction. This implied $2^4$ or 16 nodes of size, to be arranged in a 4 x 4 configuration. Since each node would have 4 array neighbors, each would require 4 reflective surfaces above it, thus requiring a total of 64 reflectors, oriented in a chessboard 8 x 8 arrangement, again arranged in the same addressing fashion as the 6-cube of Chapter 4. Here, above each node area, would exist 4 square reflectors, 2 required for x-axis and 2 for y-axis reflections. It was arbitrarily decided the top 2 squares would reflect vertically, while the bottom 2 squares would reflect in a horizontal direction. This is shown in Figure 12.

A decision was made to limit each node to a 1" x 1" area, and to position the reflector 2" above the array plane. A small program, listed in Appendix D, utilizing the equations from Chapter 4, produced the reflective angular data shown in Table 6. Notice only two distinct reflective angles are necessary for the entire array, as placement and orientation determine their beams' directions. Note also,

the alternation in Figure 11 of the two angles' placements.

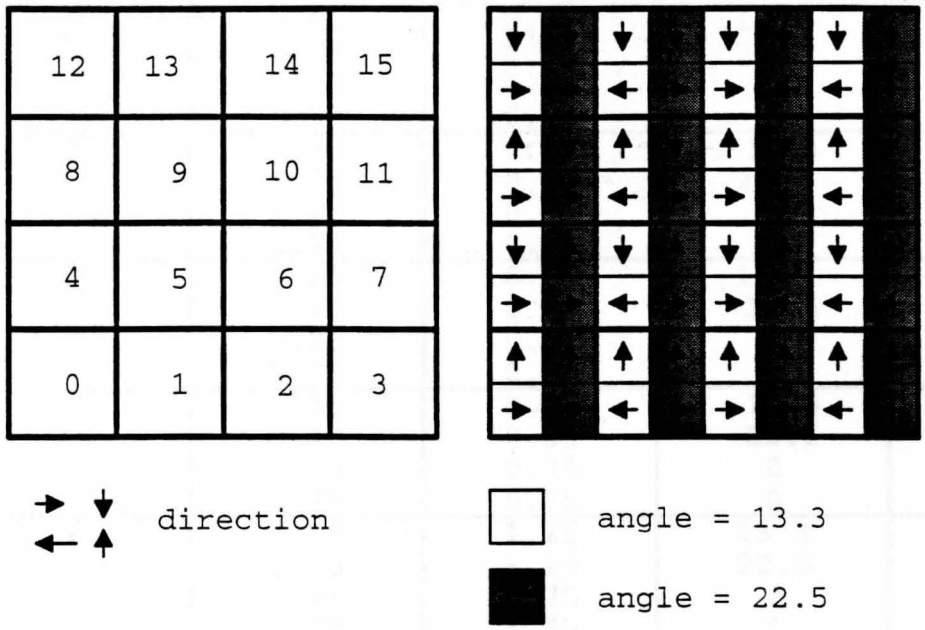



Figure 12. 4-cube Reflector Placement

To implement the reflective surface, square stock metal rods 1/2" x 1/2" were cut to the required height and ground to the correct angles; 32 were ground to an angle of 22.5°, while the other 32 were ground to 13.3°. The height of 2" separation was to be from the array surface to the center of each reflective spot.

Pictures of the hardware to be described is shown in Figures 13 and 14, while a mechanical drawing of the base plate is shown in Figure 15. Figure 16 shows the circuit board used as the array plane.

TABLE 6

REDIRECTION ANGLES FOR A 4-CUBE

| node# | x-posn | y-posn | horiz. | vert. |
|---|---|---|---|---|
| 0 | 0.25 | 0.25 | 13.3 | 0 |
| | 0.75 | 0.25 | 22.5 | 0 |
| | 0.25 | 0.75 | 0 | 13.3 |
| | 0.75 | 0.75 | 0 | 22.5 |
| 1 | 1.25 | 0.25 | -13.3 | 0 |
| | 1.75 | 0.25 | 22.5 | 0 |
| | 1.25 | 0.75 | 0 | 13.3 |
| | 1.75 | 0.75 | 0 | 22.5 |
| 2 | 2.25 | 0.25 | 13.3 | 0 |
| | 2.75 | 0.25 | -22.5 | 0 |
| | 2.25 | 0.75 | 0 | 13.3 |
| | 2.75 | 0.75 | 0 | 22.5 |
| 3 | 3.25 | 0.25 | -13.3 | 0 |
| | 3.75 | 0.25 | -22.5 | 0 |
| | 3.25 | 0.75 | 0 | 13.3 |
| | 3.75 | 0.75 | 0 | 22.5 |
| 4 | 0.25 | 1.25 | 13.3 | 0 |
| | 0.75 | 1.25 | 22.5 | 0 |
| | 0.25 | 1.75 | 0 | -13.3 |
| | 0.75 | 1.75 | 0 | 22.5 |
| 5 | 1.25 | 1.25 | -13.3 | 0 |
| | 1.75 | 1.25 | 22.5 | 0 |
| | 1.25 | 1.75 | 0 | -13.3 |
| | 1.75 | 1.75 | 0 | 22.5 |
| 6 | 2.25 | 1.25 | 13.3 | 0 |
| | 2.75 | 1.25 | -22.5 | 0 |
| | 2.25 | 1.75 | 0 | -13.3 |
| | 2.75 | 1.75 | 0 | 22.5 |
| 7 | 3.25 | 1.25 | -13.3 | 0 |
| | 3.75 | 1.25 | -22.5 | 0 |
| | 3.25 | 1.75 | 0 | -13.3 |
| | 3.75 | 1.75 | 0 | 22.5 |
| 8 | 0.25 | 2.25 | 13.3 | 0 |
| | 0.75 | 2.25 | 22.5 | 0 |
| | 0.25 | 2.75 | 0 | 13.3 |
| | 0.75 | 2.75 | 0 | -22.5 |
| 9 | 1.25 | 2.25 | -13.3 | 0 |
| | 1.75 | 2.25 | 22.5 | 0 |
| | 1.25 | 2.75 | 0 | 13.3 |
| | 1.75 | 2.75 | 0 | -22.5 |
| 10 | 2.25 | 2.25 | 13.3 | 0 |
| | 2.75 | 2.25 | -22.5 | 0 |
| | 2.25 | 2.75 | 0 | 13.3 |
| | 2.75 | 2.75 | 0 | -22.5 |

TABLE 6 (Continued)

| node# | x-posn | y-posn | horiz. | vert. |
|---|---|---|---|---|
| 11 | 3.25 | 2.25 | -13.3 | 0 |
| | 3.75 | 2.25 | -22.5 | 0 |
| | 3.25 | 2.75 | 0 | 13.3 |
| | 3.75 | 2.75 | 0 | -22.5 |
| 12 | 0.25 | 3.25 | 13.3 | 0 |
| | 0.75 | 3.25 | 22.5 | 0 |
| | 0.25 | 3.75 | 0 | -13.3 |
| | 0.75 | 3.75 | 0 | -22.5 |
| 13 | 1.25 | 3.25 | -13.3 | 0 |
| | 1.75 | 3.75 | 22.5 | 0 |
| | 1.25 | 3.25 | 0 | -13.3 |
| | 1.75 | 3.75 | 0 | -22.5 |
| 14 | 2.25 | 3.25 | 13.3 | 0 |
| | 2.75 | 3.75 | -22.5 | 0 |
| | 2.25 | 3.25 | 0 | -13.3 |
| | 2.75 | 3.75 | 0 | -22.5 |
| 15 | 3.25 | 3.25 | -13.3 | 0 |
| | 3.75 | 3.75 | -22.5 | 0 |
| | 3.25 | 3.25 | 0 | -13.3 |
| | 3.75 | 3.75 | 0 | -22.5 |

Since the reflectors' ends were ground down from a set size, there was a small difference in height which had to be added to those having the larger angle, in order to bring their height up, so that their centers would be at the same level as those reflectors with the smaller angle.

Although steel stock was selected for the reflectors, a highly polished finish was found to be too time intensive to achieve, so that after they were made, a way had to be found to increase their reflectance. This was achieved through an adhesive-backed reflective film applied after grinding a reasonably smooth top to each rod. Had this reflective arrangement been decided on in advance, the

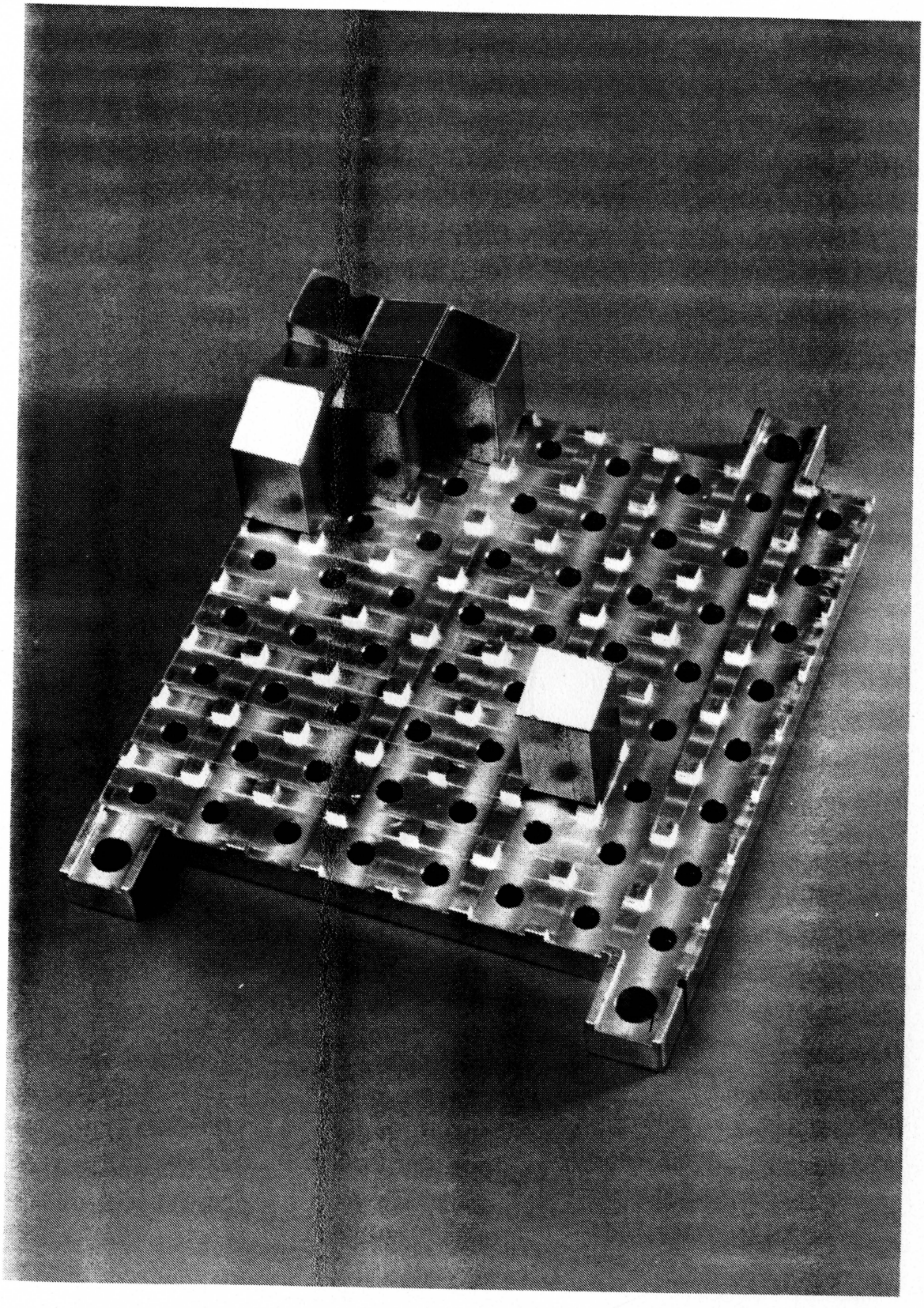


Figure 13. Partially Assembled Reflector

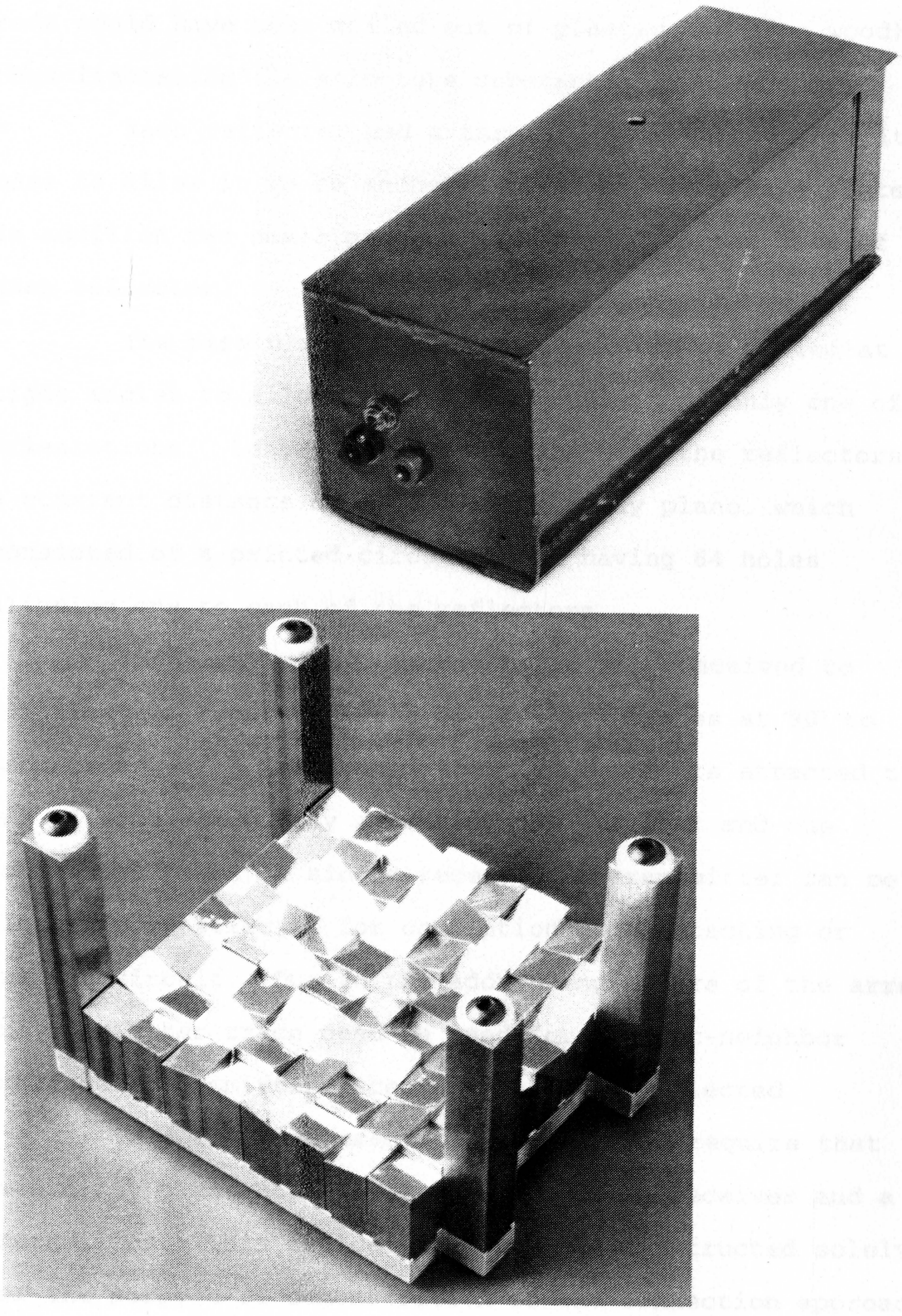


Figure 14. Assembled Reflector and Laser

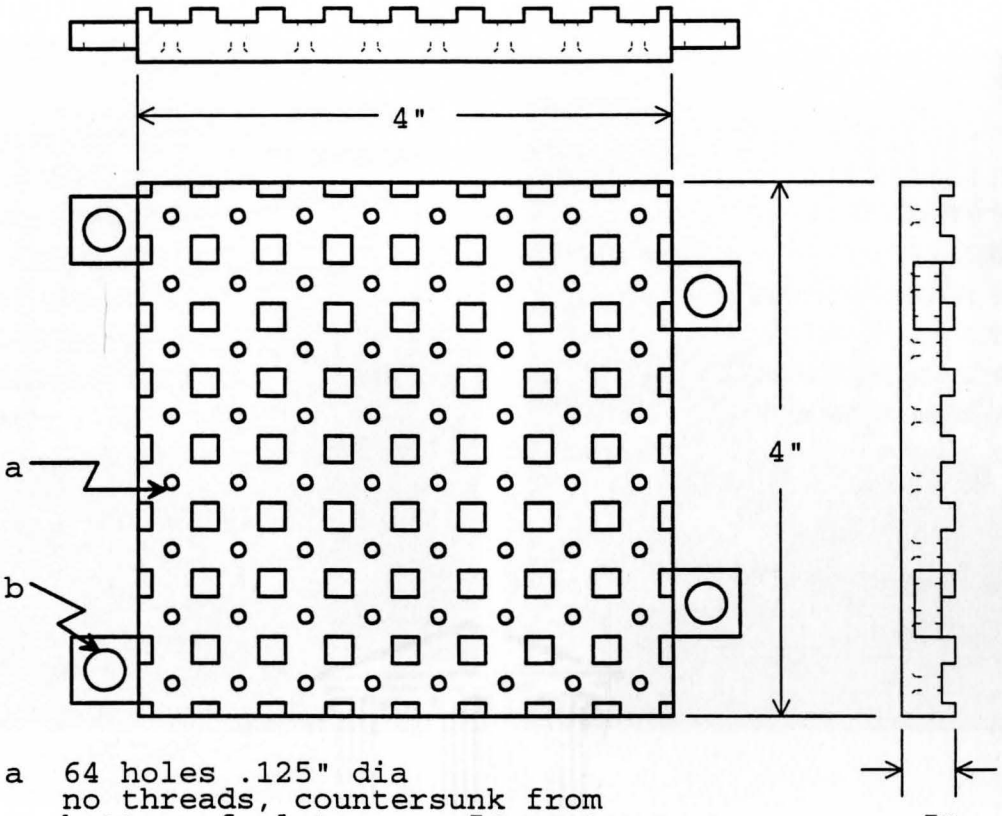
rods could have been milled out of plastic (or even wood), thus lightening the structure substantially.

Each reflector had a threaded hole tapped into its base to allow it to be anchored to an aluminum base plate. In addition two small grooves were ground on the base of each reflector.

The base plate had rectangular grooves milled at right angles to allow any reflector to sit at only one of 4 orientations. Longer threaded shafts held the reflectors at a constant distance of 2" above the array plane, which consisted of a printed-circuit board having 64 holes aligning one to each of the reflectors.

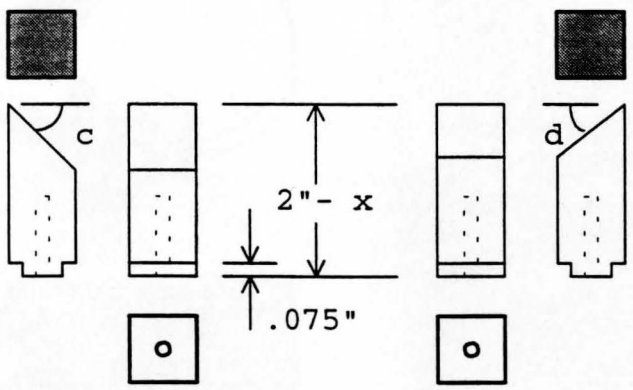
The array plane circuit board was conceived to comprise two separated sets of parallel traces at 90° to each other, with the transmitters or receivers attached to one of each trace. By selecting one vertical and one horizontal trace, a single receiver or transmitter can be electrically selected for connection to a detecting or driving circuit. Due to the addressing nature of the array, only one other trace need be used for nearest-neighbor addressed transmitter/receiver of the one selected.

In a full implementation, one would require that each array plane hole would contain both a receiver and a transmitter. This device, however, was constructed solely for the purpose of demonstration of the connection approach, and not for complete implementation of a computing device.




a 64 holes .125" dia
no threads, countersunk from
bottom of plate, on .5" centers.

b 4 holes .35" dia
no threads, tapped from top,
one per flange as shown.




c 32 rods .5" x .5" base, with centered, threaded hole from bottom. Angle = 22.5 degrees from horizontal, x = 0"

d same as c except, angle = 13.3, x = .045"

Figure 15. Drawing of Reflector Components

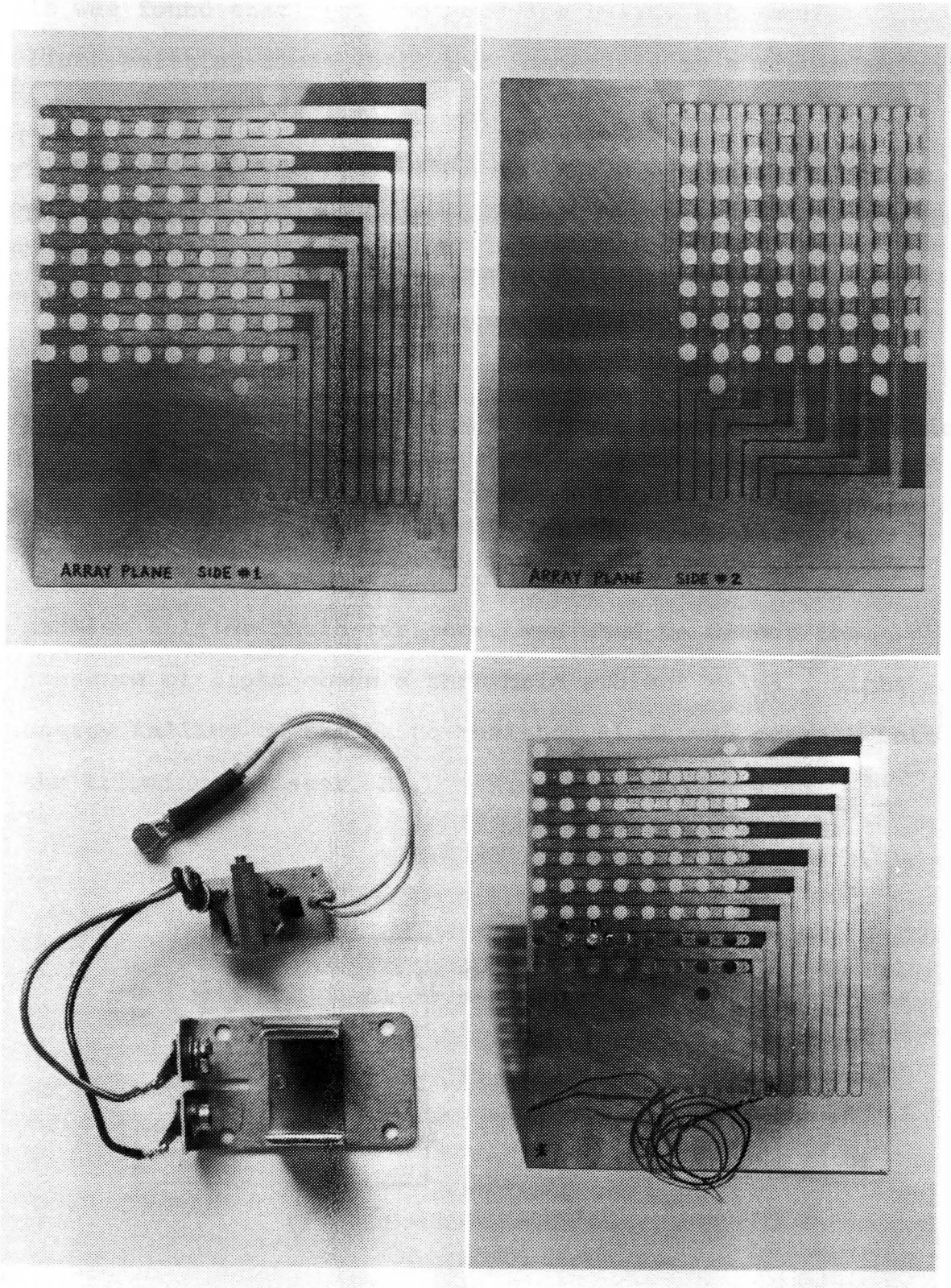
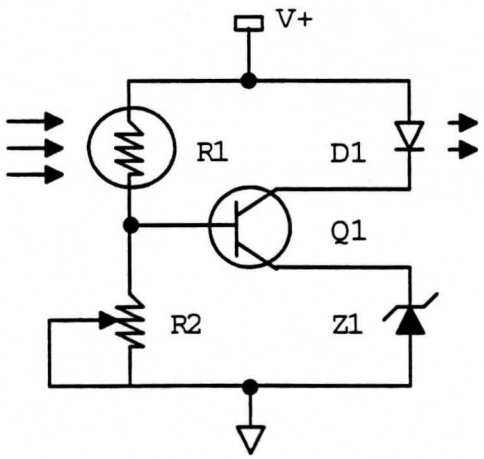



Figure 16. Array Plane Circuit Board

It was found that, for the hardware built, a common light-emitting-diode (LED) was inadequate as a transmission source, due to its limited optical output and wide angular dispersion. A narrower beam, higher output LED was found, but then rejected for cost reasons. Collimation of multiple sources was also ruled out, again to minimize cost.

For demonstration purposes, a Helium-Neon laser source was chosen. Its marginal power output of less than 1 milliwatt was not high enough to justify developing an optic splitting device; only one surface reflection can be demonstrated at a time.

A simple receiver circuit, centered around a cadmium sulfide photo-resistor, was used to detect the presence of light above a threshold ambient value. Light energy falling on the photo-resistor R1 drives current into the trimming resistor, R2.




V+ = 5 - 13 volts
R1 = phototransistor
R2 = 2K trim pot
Q1 = NPN darlington
D1 = L. E. D.
Z1 = 3.6 zener diode

Figure 17. Generic Receiver Circuit

When the voltage of R2 is higher than the zener voltage plus that of the base-to-emitter voltage, the Darlington conducts, and the resulting current through its collector lights the LED indicator. Trimming resistor R2 allows for varying the threshold value to compensate for ambient light conditions. Refer to the circuit shown in Figure 17. A picture of the completed detector is shown in Figure 16.

# CHAPTER VII

## IDEAL SYSTEM DESCRIPTION

An ideal system would have one beam director at each array node, all nodes being supplied their beams from a common pump source. By utilizing beam directors at the transmitters instead of reflective redirection, each node would be responsible for steering its own beam to its destination. Advantages for such a system would be:

a) transmitter compensation of the above inconsistencies
b) transmitter compensation for surface irregularities
c) reduction to a simple smooth (flat) reflector
d) dynamic structure reconnection
e) accommodation of random node placement
f) simplified construction with single node controllers

With identical node structures, a simple reflective plane, one common pump source, and one beam per node, the physical structure would be relatively non-complex for high numbers of array elements. Such a system is depicted in Figure 18.

The direction element would consist of two stacked refractive elements (such as piezo-optic crystals), each responsible for bending the beam in their respective axis. To communicate with another element, a node would provide x and y bending voltages to the redirector to thus point its beam in the correct direction [2]. With higher beam

directivity, dynamic changes in the overall structure, such as linear distortion due to thermal expansion, could be compensated for. To handle random node placement, each node would energize its beam in turn, while every other node would monitor for reflection, announcing through a "manager" node when its receiver had been hit by the beam. This direction information, stored in a lookup table, allows a node to "dial up" a connection, similar to a phone system. As this could allow full interconnection of an array, there would be a need for node identification as well as connection and timing protocols.

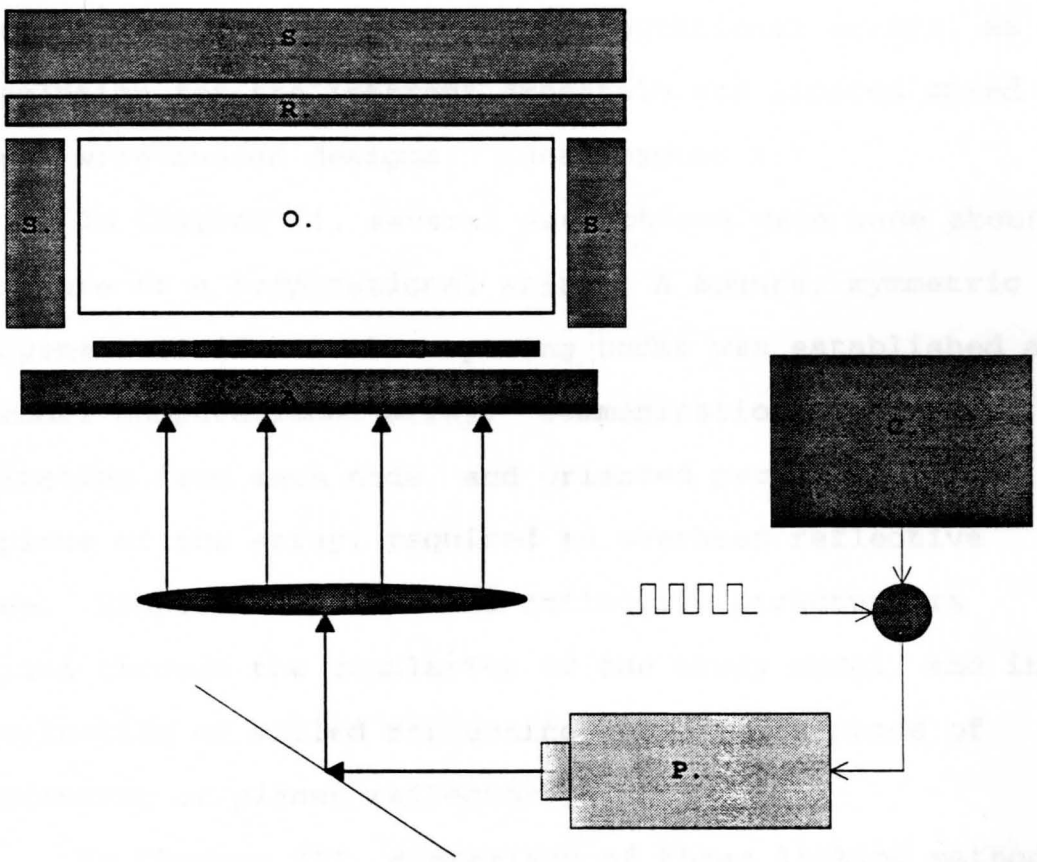
Higher link flexibility requires rules for managing an increased rate of data collision, although this would be offset by the capability of the system to dynamically restructure itself into alternate connection configurations, indeed even allowing fracturing of the main system into computational subsystems. This would translate into capability to handle varying degrees of computational complexity to match that required by the incoming applications.

Structural support of the reflector could be implemented by utilizing a solid, optically transparent material (glass/plastic/diamond) in place of free space. This could be aided by implementing direct point-to-point connection by eliminating the reflective plane and placing the array nodes in a circular, cylindrical, or spherical orientation, although for construction purposes, a planar

configuration is preferred. Both of these would decrease the propagation time of the light beams, but would impose power constraints due to optical attenuation through the bending devices and the associated heat and stress dissipation.

The entire paper has implied common frequency light sources, to minimize system complexity. However, this need not be the case. Provided that its receiving area is complementary, each node need not be limited to the same optical frequency. This would reduce optical crosstalk from multiple reflections.

Overall system speed would depend primarily on the base speeds of the nodes, augmented by using supercooled, superconducting, or high-speed node logic circuitry.




A. Array Plane with Optic-Gate Substrate.
C. Control System for Pump and Reflector.
O. Optical Media.
P. Beam Pump Source.
R. Reflector Plane.
S. Structural/Cooling Framework.

Figure 18. Ideal System Description

# CHAPTER VIII

## SUMMARY

This paper has covered some of the aspects of a reflective connection device for computational arrays, as one solution for the inherent crosstalk and limited speed of current wire-bussed designs. (See Chapter I.)

In Chapter II, several assumptions were made about the nature of a computational array. A square, symmetric arrangement of identical computing nodes was established as the model computational array. Communication light beams originating from each node, and oriented perpendicular to the plane of the array, required an overhead reflective device. Simplification of the reflective structure is obtained through the regularity of the array model, and in the selection of milled reflecting elements in place of hologramatic or planar reflectors.

In Chapter III, comparison of three linking methods (ring, full, and N-cube), led to the selection of the N-cube for the computational array. The N-cube architecture is a binary structure containing $2^N$ nodes. Primarily, the low maximum path distance and low number of physical links of the N-cube minimized the number of reflections implemented. Reflector simplification was achieved by positioning the array nodes in a regular manner.

As further explained in Chapter IV, addressing and positioning of the N-cube had impact on the nature of the reflector design. In addressing the N-cube in a gray-code manner, and requiring the array nodes to be systematically arranged on a 2-dimensional plane, the reflector element requirements are reduced to single axis reflections. This later eased the milling requirements.

Possible errors (receiver shift, reflector shift, and beam misalignment) were considered, and the adjustments to individual surfaces' angles was given. These errors, as derived in Chapter V, can be compensated for when they are time-invariant. No mention of measurement for these errors was discussed.

Construction of a prototype reflective device was documented in Chapter VI. The reflector, consisting of 64 milled reflective rods attached to a base plate, is suited for a 4-cube of 16 processors. By utilizing the layout considerations of Chapter IV, there are only 2 different angled reflectors used (32 of each). A helium-neon laser and small detector circuit were assembled for use in demonstration of the reflection elements.

Chapter VII gave a brief description of a possible ideal system, including some ideas which would allow dynamic reconfiguration of the reflector plane (common pump beam, optical gates, transmitter beam steering, etc.)

Further research into this type of connective system might include many details of the aforementioned

ideal system description (as described in Chapter VII). Microlithography of an entire wafer-scale device would be the next logical step, as technology always demands smaller size, reduced power, reduced weight, higher computing power, and the like. An exploration of other physical placement algorithms could yield a more effective (speed, real-estate, cost) design. In addition, actual construction of an entire computing array design, complete with interfacing hardware and configuration software, would be most desired.

# APPENDIX A

```
'NAME: ARRAY01.BAS
'DATE: 02-92
'NOTE: This program displays a planar array of 64 cells,
'assumed connected in a 6-CUBE architecture.  Neighbors
'(grey) are indicated for one selected (red) element.
'u$ are data for the address characters, del is pitch
'between digits.  (x(n),y(n)) is beginning point while
'(x(n+1),y(n+1)) is ending point for numeral line segment
'where 0 < n < 8.

SCREEN 9: WINDOW (-1.5, -2.5)-(9, 8)
u$(0) = "0111111"
u$(1) = "0001100": x(1) = .3: y(1) = .2
u$(2) = "1011011": x(2) = .2: y(2) = .2
u$(3) = "1011110": x(3) = .2: y(3) = .3
u$(4) = "1101100": x(4) = .3: y(4) = .3
u$(5) = "1110110": x(5) = .3: y(5) = .2
u$(6) = "1110111": x(6) = .3: y(6) = .1
u$(7) = "0011100": x(7) = .2: y(7) = .1
u$(8) = "1111111": x(8) = .2: y(8) = .2
u$(9) = "1111100"
del = .16

'grid draw routine
FOR i = 0 TO 7
 FOR j = 0 TO 7
  LINE (i, j)-(i + .9, j + .9), , B
  numb = j * 8 + i
  t = INT(numb / 10)
  o = numb - 10 * t
  FOR k = 1 TO 7
   t1$ = MID$(u$(t), k, 1)
   t2$ = MID$(u$(o), k, 1)
  IF t1$ = "1" THEN LINE
(i+x(k),j+y(k))-(i+x(k+1),j+y(k+1))
  IF t2$ = "1" THEN LINE
     (i+x(k)+del,j+y(k))-(i+x(k+1)+del,j +y(k+1))
  NEXT k
 NEXT j
NEXT i
LOCATE 22, 10: PRINT "Arrows to move.       y address = "
LOCATE 23, 10: PRINT "<esc> for exit.       x address = "
'keyboard interpretation routine
get.key: a$ = INKEY$: IF a$ = "" THEN GOTO get.key

IF LEN(a$) = 2 THEN b = ASC(MID$(a$, 2, 1))
IF ASC(a$) = 27 THEN END
IF b = 72 THEN y = y + 1
```

```
IF b = 80 THEN y = y - 1
IF b = 77 THEN x = x + 1
IF b = 75 THEN x = x - 1
x = ABS(x + 8) MOD 8
y = ABS(y + 8) MOD 8

'...erases the old mappings...
PAINT (old.x + .5, old.y + .5), 0, 15

FOR i = 0 TO 2
 PAINT ((old.x XOR 2 ^ i) + .5, old.y + .5), 0, 15
 PAINT (old.x + .5, (old.y XOR 2 ^ i) + .5), 0, 15
NEXT i

'...and maps the adjacent addresses...
PAINT (x + .5, y + .5), 4, 15

FOR i = 0 TO 2
 PAINT ((x XOR 2 ^ i) + .5, y + .5), 7, 15
 PAINT (x + .5, (y XOR 2 ^ i) + .5), 7, 15
NEXT i

FOR i = 0 TO 2
 LOCATE 22, 47 - 2 * i
 PRINT (y AND 2 ^ i) / 2 ^ i;
 LOCATE 23, 47 - 2 * i
 PRINT (x AND 2 ^ i) / 2 ^ i;
NEXT i

'...then saves this new position...
old.x = x: old.y = y

'...and goes back to get another key-press.
GOTO get.key

END
```

# APPENDIX B

```
'NAME: ARRAY02.BAS
'DATE: 02-92
'PROG: This program calculates nearest-neighbor data for a
'planar array of 64 cells, configured in a 6-CUBE
'architecture.  Note:  The 6-bit address of each node
'contains the following data:
'(a) The addresses of each of the 6 adjacent nodes.
'(b) The direction(physical) towards each adjacent node.
'(c) The number distance to each of the adjacent nodes.

'OPEN "b:\stuff.out" FOR OUTPUT AS #1

OPEN "con" FOR OUTPUT AS #1

FOR  z  = 0 to 63
 IF z MOD 16 = 0 THEN
  PRINT #1, ""
  PRINT #1, " --address--  ---neighbor-addresses---
  ----------------"
  PRINT #1, "  z     y    x     (32) (16) (8)  (4)  (2)
  (1)    A B C A B C"
  ELSE
 END IF

 y = INT( z / 8 )
 x = z - 8 * y

 PRINT #1, USING " ##   "; z;

 FOR i = 2 TO 0 STEP -1
  PRINT #1, RIGHT$(STR$((y AND 2 ^ i) / 2 ^ i), 1);
 NEXT i

 PRINT #1, " ";

 FOR i = 2 TO 0 STEP -1
  PRINT #1, RIGHT$(STR$((x AND 2 ^ i) / 2 ^ i), 1);
 NEXT i

 PRINT #1, " ";

 FOR i = 5 TO 0 STEP -1
  PRINT #1, USING " ##  "; z XOR 2 ^ i;
 NEXT i

 PRINT #1, " ";
```

```
 FOR i = 5 TO 0 STEP -1
  IF (z AND 2 ^ i) / (2 ^ i) THEN PRINT #1, "-"; ELSE PRINT
  #1, "+";
  PRINT #1, "    ";
 NEXT i
 PRINT #1, ""

NEXT z

END
```

# APPENDIX C

```
'NAME: ARRAY03.BAS
'DATE: 06-93
'PROG: This program will generate random addresses for each
'node in a 4-cube, displaying the current address for each
'node stack as well as the percentage used by the four node
'paths.  An indication is given for any excessive node stack
'size.

DIM z(16,4,190),y(16),x(16)
'z is the node stack array.  z(,,0) is the node stack size.
'y array counts the number of hits used by the 8,4,2,1 bits.
'x array counts the number of unique values in 8,4,2,1 bits.

SCREEN 0: CLS

FOR a = 0 TO 15
 LOCATE 5 + a, 10
 PRINT USING "##"; a;
 PRINT "              .           .           .          ."
NEXT a

FOR y = 0 TO 299
 FOR a = 0 TO 15
  LOCATE 1, 1: PRINT y, a
  'wait for a keypress to continue.
  'keypress: a$ = INKEY$: IF a$ = "" THEN GOTO keypress
  'd is the destination address, while a is the current node
  'address.
  d = INT(15 * RND(1))
  b = a XOR d
  'generates f(B)
  b = 15 AND b AND (NOT (b * 2))
  b = b AND (NOT (b * 4)) AND (NOT (b * 8))
  IF b = 8 THEN c = 0
  IF b = 4 THEN c = 1
  IF b = 2 THEN c = 2
  IF b = 1 THEN c = 3
  IF b = 0 THEN GOTO around
  'PRINT "sending to node: "; b XOR d
  b = a XOR b
  'b is now the new address to send the token to.
  'c is the bit that was changed.
  LOCATE 5 + b, 20 + 12 * c
  z(b, c, 0) = z(b, c, 0) + 1
  'if b = d then the token has reached its destination.
  '  IF b = d THEN GOTO around
  'if there are more than 40 stack elements then print "E"
  IF  z(b, c, 0) > 40 THEN PRINT "*"; ELSE PRINT " ";
```

```
   z(b, c, z(b, c, 0)) = d
   PRINT USING "(###)"; z(b, c, 0);
   PRINT USING " ##   "; z(b, c, z(b, c, 0))
   y(c) = y(c) + 1
   'at the bottom, print the percentages.
   FOR e = 0 TO 3
    LOCATE 22, 23 + 12 * e
    temp = y(0) + y(1) + y(2) + y(3)
    PRINT USING "##.#  "; 100 * y(e) / temp
   NEXT e
around:

   'FOR b = 0 TO 3
   'IF z(a, y(b), 0) = 0 THEN GOTO around2 ELSE z(a, y(b), 0)
   '= z(a, y(b), 0) + 1
   'around2:
   NEXT b
 NEXT a
NEXT y

   'THIS ROUTINE SHOULD LIST THE NUMBER OF UNIQUE ADDRESSES
   'WHICH PASSED THROUGH EACH OF THE NODES
   'FOR a = 0 TO 15
   ' PRINT
   ' FOR b = 0 TO 3
   '  FOR c = 0 TO 15
   '   x(c) = 0
   '  NEXT c
   '  sum = 0
   '  FOR c = 1 TO z(a, b, 0)
   '   IF x(z(a, b, c) = 0 THEN sum = sum + 1
   '   IF x(z(a, b, c) = 0 THEN PRINT z(a, b, c); " ";
   '   x(z(a, b, c)) = 1
   '  NEXT c
   '  LOCATE 5 + a, 20 + 7 * b
   '  PRINT TAB(50); , ;
   '  PRINT USING "#### "; sum
   ' NEXT b
   'NEXT a

END
```

# APPENDIX D

```
'NAME: ARRAY04.BAS
'DATE: 6-93
'PROG: This program generates nearest-neighbor node angles
'of reflection for a 4-cube.  Data is outputted for each of
'the four sub areas within each node.  It is assumed that
'the position of each sub area is the same with respect to
'each node.

OPTION BASE 0
DIM X(16), Y(16), o(4,2,2), ang(16,4,2), used(16,4)
CONST pi = 3.141592654#
CLS
h = 2                                          'height
a = 4                                          'array length
b = 4                                          'array width
cell.a = a / 4                                 'cell length
cell.b = b / 4                                 'cell width
PRINT "h="; h; "  a="; a; "  b="; b

'Generates cell offset positions.
FOR j = 0 TO 3
 FOR i = 0 TO 3
  n = i + j * 4
  X(n) = a * i / 4
  Y(n) = b * j / 4
 NEXT i
NEXT j

'Generate xmtr/rcvr offsets within cell, o(a,b,c) where a is
'the sub-cell(0-3), b is xmtr/rcvr (0/1), c is x/y (0/1)
'Assumes xmtr and rcvr offsets are the same.

FOR k = 0 TO 1
 FOR l = 0 TO 1
  m = l + k * 2
  o(m,0,0) = cell.a / 4 + cell.a * l / 2
  o(m,0,1) = cell.b / 4 + cell.b * k / 2
  o(m,1,0) = cell.a / 4 + cell.a * l / 2
  o(m,1,1) = cell.b / 4 + cell.b * k / 2
 NEXT l
NEXT k

'Calculates nearest-neighbor node numbers.
'Uses a used node matrix used(a,b) where a is the node
'number (0-15), 'and b is the xmtr number (0-3).
'Also computes reflection angles using invtan(d/h)
```

```
FOR i = 0 TO 15
 FOR j = 0 TO 1
  IF used(i,j) = 1 THEN PRINT "  hit"; : GOTO around
  used(i,j) = 1
  k = i XOR 2 ^ j
  d = X(k) + o(j,1,0) - X(i) - o(j,0,0)
  ang(i,j,0) = 28.6479 * ATN(d/h)
  IF ang(i,j,0) > 90 THEN ang(i,j,0) = ang(i,j,0) - 180
 NEXT j

 FOR j = 2 TO 3
 IF used(i,j) = 1 THEN PRINT "  hit"; : GOTO around
 used(i,j) = 1
 k = i XOR 2 ^ j
 d = Y(k) + o(j,1,1) - Y(i) - o(j,0,1)
 ang(i,j,1) = 28.6479 * ATN(d/h)
 IF ang(i,j,1) > 90 THEN ang(i,j,1) = ang(i,j,1) - 180
around:
 NEXT j

NEXT i

'Prints out the center positions (x,y) and angles (x,y)
'for the reflector patches.

FOR i = 0 TO 15
 PRINT "("; X(i); ","; Y(i); ")";
 FOR j = 0 TO 3
  PRINT , "      at (x,y)";
  PRINT USING " ###.###"; X(i) + o(j,1,0); Y(i) + o(j,1,1);
  PRINT "  angle (x,y)";
  PRINT USING " ###.###"; ang(i,j,0); ang(i,j,1)
 NEXT j
 PRINT
NEXT i

END
```

# REFERENCES

[1] T. E. Bell, "Optical Computing: a field in flux.," IEEE Spectrum, 1986, 23#8:34-57.

[2] C. D. Howe and B. Moxon, "How to program parallel processors," IEEE Spectrum, 1987, 24#9:36-41.

[3] L. D. Hutcheson and P. Haugen, "Optical interconnects replace hardware," IEEE Spectrum, 1987, 24#3:30-35.

[4] M. Leonard, "The World of Communications is Moving To Fiber Optics," Electronic Design, 1992, 40#1:73-80.

[5] P. Wiley, "A parallel architechture comes of age at last," IEEE Spectrum, 1987, 24#6:46-50.

# BIBLIOGRAPHY

Drummond, T. J. et. al. 1988.
"Quantum-tailored solid-state devices,"
IEEE Spectrum, 25#6:33-37.

Goutzoulis A. P. and Abramovitz I. J. 1988.
"Digital Electronics meets its match,"
IEEE Spectrum, 25#8:25.

Katzman, M., 1987.
Laser Satellite Communications
N. J. Englewood Cliffs: Prentice Hall, Inc.

Shibata J. and Kauiwara T. 1989.
"Optics and electronics are living together,"
IEEE Spectrum, 26#2:34-38.

Turabian, K. L. 1973.
A Manual for Writers of Term Papers, Theses, and Dissertations
Chicago: University of Chicago Press.

Verdeyen, J. T., 1981.
Laser Electronics
N. J. Englewood Cliffs: Prentice Hall, Inc.