LASER LINK CROSSBAR AND OMEGA NETWORK SWITCHING DESIGN AND COMPARISON FOR FAULT-TOLERANCE PURPOSE

by

Kewei Fang

B.S.C.S. Peking University, Beijing, China, 1984

M.S.C.S. Peking University, Beijing, China, 1989

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

in the School
of
Engineering Science

SIMON FRASER UNIVERSITY
June 1995

APPROVAL

Name: Kewei Fang

Degree: Master of Applied Science

Title of Thesis: **Laser Link Crossbar and Omega Network Switching Design and Comparison for Fault-Tolerance Purpose**

Examining Committee: Dr. Jamal Deen, Chairman

Dr. Glenn H. Chapman, Senior Supervisor

Dr. Marek Syrzycki, Supervisor

Dr. Rick Hobson, Examiner

Date Approved: June 20, 1995

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its usrs. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

"Laser Link Crossbar and Omega Network Switching Design and Comparison for Fault-Tolerance Purpose"

Author: _____
(signature)

Kewei Fang
(name)

June 20, 95
(date)

Abstract

Signal routing for defect avoidance and redundant cell interconnection is an essential part of Large Area or Wafer Scale Integrated systems. This thesis studies the tradeoffs between two wafer scale defect avoidance methods; permanent physical connections such as laser linking and active electronic switches. Laser links use the power of a laser to melt the gap between two doped regions creating a connection. To investigate laser connection techniques an 8 x 8 crossbar network was designed and fabricated in CMC 3 micron CMOS process. Restructuring was done using the laser table system at SFU. Linking parameters established by experiment were laser pulses of 2.6 W, 300 microsec, with 6 linking points per device, generating a connection of about 76 Ω. Aluminum lines 5 microns wide were cut with 1.2 W pulses. To minimize design effort a program was written to automatically generate laser bus systems, and tested on the transducer array chip. For active switches a basic 8 x 8 omega network has been designed and implemented with CMC 3 micron process. The transfer block has been tested and the results match the HSPICE simulation results. The system resistance is around 800 Ω, the rise time of 19.59 ns and the falling time is 12.61 ns. The area, power consumption, speed and in-service flexibility of the two laser linking and active switches have been discussed and compared. The area of omega network is 6 times that of equivalent laser crossbar network, and 57 times when the control circuitry is included. The omega switch power consumption is about 100 times that of

the laser crossbar network. Two new kinds of network models which are combinations of the laser linking and omega switches are proposed based on the above discussion and comparison.

To My Family

To My Friends

ACKNOWLEDGMENTS

I would like to thank my senior supervisor, Dr. Glenn Chapman for his encouragement and guidance throughout the research. I also would like to thank Dr. Jamal Deen, Dr. Marek Syrzycki and Dr. Rick Hobson for their useful comments. I wish to thank Mr. Gary Houghton, Mr. Bill Woods and Darren E. Bergen for their help during this research. Especially I want to express my thanks to Brigitte Rabold for her help during my graduate study. In addition, I also wish to thank Canadian Microelectronics Corporation for providing CMOS3 and MITEL 1.5 micron processes for the fabrication of various circuits.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Wafer Scale Integration

Wafer Scale Integration (WSI) is defined as a technology that enables the fabrication of monolithic chips larger than a single chip (15 x 15 mm) and ranging up to the maximum wafer diameter in commercial manufacturing (currently 200 mm) [1]. A characteristic but not exclusive feature of a wafer scale system is the use of redundancy and a strategy to solve the problem of manufacturing defects which inherently destroy the operation of significant fractions of a WSI circuit.

Wafer Scale Integration is the culmination of investigation and development aimed at the fabrication of larger integrated complete systems on a single substrate, replacing whole printed circuit board (PCB) of regular chips. In VLSI, chips were developed by fabricating a wafer with hundreds of identical circuits, testing these circuits, dicing the wafer, and packaging the good dice. Complete systems usually require the

interconnection of chips on a PCB. Since yield of IC's decreases rapidly with increasing area, there is a limit to the complexity of chips which can be built with a given transistor geometry. Even DRAMs, the highest density devices, can only be built using spare memory columns/rows [2]. In contrast in WSI, a wafer is fabricated with several types of circuit blocks (generally referred to as cells) and multiple instances of each cell type. These cells are also tested, and good cells are interconnected to realize a system on a wafer. In addition, VLSI technology has speed restrictions and significant power consumption due to the large area of driver transistors needed to drive the inductance, capacitance and resistance between packaged chips. Since most WSI signal lines stay on the wafer, stray capacitance is lower so that high speeds are achieved and consequently lower power consumption. Finally by integrating all devices on a single substrate considerable volume and mass savings occur relative to PCB systems. Using the same technology as VLSI, WSI implementation has been estimated in the literature to have the potential to be a factor of five times faster, to dissipate ten times less the power, and require one hundredth to one thousandth the volume [3].

The development of WSI began in the 1960s. After that there were several important WSI developments. The first paper to treat WSI issues was by Sack, Lyman, and Chang at Westinghouse in 1964 [4]. They proposed ideas about WSI which are widely used now. In the late 60s, Petritz at Texas Instruments investigated a discretionary wiring technique as a means for interconnecting circuits of about 200 gates complexity on 1.25 inch diameter wafers [5]. This technique involved fabricating the wafers, identifying the errors, and depositing and patterning layers to bypass the errors. Calhoun at Hughes Aircraft company refined and simplified the discretionary wiring approach. Hughes reduced the number of discretionary masks and tested the devices at a point

closer to the end of processing. This technique uses low cost means to accomplish the discretionary levels, but must control defects introduced by processing and handling after the probe test [6].

In the 1970s, several approaches emerged that fabricated the complete circuit before testing and reconfiguration. Hsia, Chang and Erwin at McDonnell Douglas proposed a scheme called "Adaptive WSI" [7]. The approach used a system of circuit modules, buses, and nonvolatile NMOS memory switches to isolate defective regions and to connect working modules.

During the 1980s, MIT Lincoln Laboratories introduced a Restructurable Very Large Scale Integration (RVLSI) concept [3, 8]. This successful program proved the technical feasibility of WSI by designing and fabricating many different complex processor circuits. A key feature of RVLSI is the use of a laser processing of the circuit to connect and/or cut multi-level metal patterns on the wafer. Both the design and fabrication of RVLSI circuits are supported by extensive computer aids.

Peltzer at Trilogy Systems Corporation reported a large, high speed computer based on WSI circuit modules. They selected the emitter coupled logic (ECL) as the basic technology [9]. However, Trilogy's attempt failed badly due to an insufficiently robust defect-avoidance scheme. Meanwhile, Inova Microelectronics Corporation was founded with the objective of developing a simple, manufacturable approach to WSI to produce large static memory modules. The Inova technique used laser cutting to restructure circuits and had the advantage of requiring only standard processing [10].

Approaches in WSI include developing defect avoidance theories, new post-fabrication processes, special packaging technique, realizing interconnection networks and their

use in defect avoidance to increase system yield. Several of these techniques have found application in regular VLSI circuits, such as laser cutting for error correction in DRAM's and built in self test circuits. In addition, WSI can assimilate the technological progress from regular integrated circuits such as reduced device geometries. WSI can also be based on many different materials and/or circuit technologies. For example, WSI can be fabricated with either silicon or gallium arsenide substrate. Within silicon, either CMOS or BiCMOS can be used.

1.2 Redundancy Methods in WSI

An essential technique which influences the success of WSI is the reconfiguration technique which dominates interconnection networks for redundancy purposes. There are three variations of this technique which can be employed:

1. Physical restructuring through which interconnections are physically altered, by cutting or adding.

2. Electronic switching of interconnections through use of programmable switches to bypass faulty circuits.

3. Functional fault avoidance for specific functions, such as large memory arrays, by simply ignoring the faulty portion of the circuit (e.g. by declaring a bad portion of memory as faulty and externally avoiding addresses to that block in much the same manner that hard disks avoid defective regions).

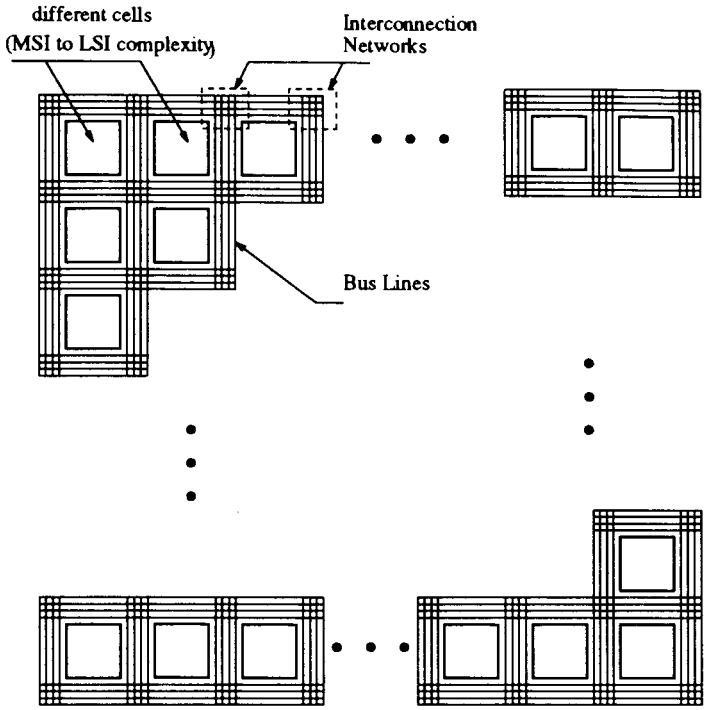


Figure 1.1: Structure of wafer scale integration. (Adapted from [38].)

Interconnection networks are employed in WSI for defect-avoidance and fault-tolerance purposes. To be successful in implementing technology using wafer scale integration, the target system must first be portioned into modules (referred to as cells) of a size which can guarantee a reasonable yield. The diagram in Figure 1.1 shows the structure of such a system.

After fabrication, these modules must then be tested and electrically connected on the wafer. With an interconnection network, different cells are connected to form the function of the target system. If some of these cells fail, the interconnection networks should connect redundant cells. According to the repair techniques mentioned in the last paragraph, interconnection networks may be classified according to whether or not they are permanent, volatile, or electronically programmed [11]. The classification

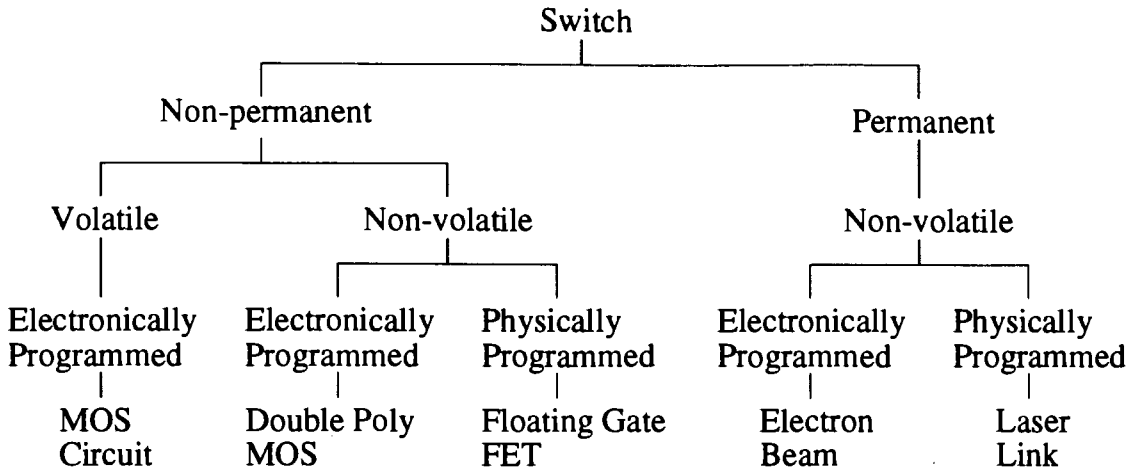


Figure 1.2: A classification of switch implementations.

ranges for implementations are given in Figure 1.2.

1.2.1 Physical Restructuring

Physical restructuring techniques cut lines and make connections using laser-assisted chemical processes, focussed ion beams, electron-beam and laser link processes. Among these, the laser link method of cut and connection restructuring is most successful [3].

After a wafer has been fabricated using standard processes, the laser-link physical restructuring technique uses the power of a laser beam to link or cut bus lines. The choice of connecting or segmenting the buses changes the signal paths of WSI system to avoid defects. This technique was first proposed by J. Raffel in 1979 [12]. At MIT Lincoln Laboratory, the restructurable VLSI project has developed a design methodology, new technology, and CAD tools for wafer scale integration. J. Raffel, A. Anderson and G. Chapman have developed laser restructurable technology and design [3]. As an effective approach they proposed a methodology for laser link physical

restructuring WSI. The important features of this methodology are:

- Additive and subtractive restructuring of wafer system interconnects.
- Isolating the cells from the wafer power and signal buses during fabrication.
- Complete testing of circuits before laser restructuring.
- Complete testing of wafer system interconnects before restructuring.
- Automating signal routing and control of the laser restructuring equipment to create the signal paths on the wafer.
- Testing the signal paths and partially building the system during the restructuring process.

Their research covered a post-fabrication process using laser link and requiring new equipment, basic laser-link structures, and system design. Cohen and Chapman reported detailed mechanisms for silicon substrate reaction after laser linking/cutting [10]. R. Frankel and G. Young created the RVLSI CAD tool, SLASH, which aids the user in carrying out the four-part RVLSI layout/restructuring task [13]. SLASH helps in :

1. Creating a wafer scale layout.
2. Analyzing capacitance test data to determine where defects are located on a particular wafer.
3. Mapping (placement and routing) of a logical system onto the usable resources of a particular wafer.

4. An incremental restructuring and testing cycle during which laser operation implements the prescribed layout.

This physical restructuring technique can remove the existing defects, creating a high probability of building working WSI systems. The disadvantage of this technique is that it needs an extra post processing to do the connecting/disconnecting.

1.2.2 Electronic Switch Restructuring

A programmable electronic reconfiguration switch network is also a desirable technique for interconnecting WSI modules for defect-avoidance and fault-tolerance. This technique requires no extra steps beyond those needed for a standard process. It uses a programmable interconnection network (the structures of interconnection networks are the same as networks used in parallel processor systems) and the routes can be changed during testing and while in service to achieve a self-healing capacity. The programmable signals emitted from control processors decide the connections of different cells. Also in this case redundant cells are required for defect-avoidance and fault-tolerance purposes [14]. Fundamental factors related to programmable electronic switches are:

- Switch network selection and design.
- Switch control circuit design.
- How to set the defect avoidance routing of signals through the switches to harvest the working cells.

- Storage of the switch settings to create automatic start up conditions which set the previously determined switch configurations.
- The signal delay or skews imposed by the switches on the data propagating down the reconfigured paths of various lengths.
- Programmable logic to control the interconnection routes.
- Test strategies for the interconnection networks and systems.
- Procedures for bypassing failed switches.

There has been substantial difficulty implementing programmable electronic switching networks in a WSI system due to the switch yield problem. Thus considerable theoretical, but few practical implementations, have been published on this subject. Below is a summary of the important practical literature on implementations.

W. Chen and P. B. Denyer et. al. at Edinburgh University have developed a fault-tolerant architecture for wafer scale integration which uses a large crossbar network as a reconfigurable interconnect network for serial signals [15]. Their research of the crossbar network was already performed on bit-serial architecture previously used for their FIRST silicon compiler. The FIRST (for Fast Implementation of Real-time Signal Transforms) silicon compiler is a tool which takes a high-level function description as input and produces a detailed chip mask geometry instead of the machine code used by a conventional compiler. Chen et.al. established a dynamic matrix and a static matrix. For a 32 x 32 dynamic matrix using 2.5 micron, double metal CMOS process, the size is 5.2 x 3.1 mm^2; for 32 x 64 static matrix (they have same interconnection capabilities) with same process, the size is 7 x 2.7 mm^2. Their research results show

that the crossbar network consumes a large area of the design relative to the bit serial processors, thus an electronic switch crossbar network takes considerable fraction of the area.

M. Slimane-Kadi, A. Boubekeur and G. Saucier developed a programmable switch to implement interconnection networks to communicate between processors and between processors and memory modules [16]. They discussed using the crossbar, omega and Benes networks to interconnect parallel processors and covered network reliability of a normal 8 x 8 x 8 - bit omega network and a fault-tolerant omega network. They proposed a flexible programmable switch which is very useful in realizing interconnection networks for parallel processors and which also can be employed to research general interconnections in wafer scale integration. The advantages of programmable electronic switch networks are that they have in-service flexibility and can be fabricated using standard processes. In addition, the signals can be routed to switch in spare blocks creating a self healing system. However this flexibility is gained at the cost of the large area consumed by the switches.

1.3 Scope of This Thesis

The interconnection networks are important for wafer scale integration and worthy of detailed study. The subject of the research for this thesis is a comparison of two typical techniques in WSI: a laser link physically restructurable network and an electronic switch network. A crossbar network is a typical physical switch network which is widely discussed and which possesses distinct advantages such as simplicity of structure and high flexibility. For electronic switches an omega network is one

of the simplest multistage interconnection networks with efficient data transmission ability. It consumes a smaller area compared with other switch networks. Based on the above considerations, we designed and fabricated a crossbar network for our physical restructuring technique and a test omega interconnection network for the programmable electronic switch technique. The circuits were fabricated with a 3-micron CMOS process at CMC. A CAD tool was developed for laser link crossbar networks, reducing the layout effort for their design. Physical restructuring was undertaken in the laboratory of the School of Engineering Science at Simon Fraser University. By comparing these two kinds of networks, we have obtained parameters for the tradeoff between features of these two kinds of networks. We propose a new interconnection network which combines the two techniques. This new network has the advantages of saving area, saving transmission time, and possessing in-service flexibility.

In Chapter 2, the theory of interconnection networks is introduced and discussed, including the general classification of topologies of interconnection networks and an introduction to the structure of the omega and crossbar networks. In Chapter 3, the circuit of the laser link crossbar network is introduced, including the CAD layout of crossbar network, the laser linking and cutting process, and the measurement results of laser link. In Chapter 4, interconnection structures with omega network are described, including interconnection structures for wafer scale integration, the layout of the omega network, HSPICE simulation, and measurement results. In Chapter 5, these two kinds of networks are compared and the tradeoff of these two kinds of networks are discussed. A new kind of interconnection network is also proposed. Chapter 6 summarizes the results of this thesis project.

CHAPTER 2

The Theory of Interconnection Networks for Defect-Avoidance and Fault-Tolerance

As mentioned in Chapter 1, interconnection networks for defect-avoidance are essential parts of a WSI system. During the fabrication of wafers, defects created on a wafer affect electronic characteristics of devices and interconnections. Defects cause a low yield of functional blocks (cells) and also cause some damages to the interconnection lines of a WSI system. An interconnection network offers an effective defect-avoidance method to solve this problem. Interconnection networks also have fault-tolerant ability and are widely used in redundant systems.

The theory of interconnection networks has been developing since the 1970's [17],

including design decisions, topologies and communication protocols. A brief discussion of interconnection networks related to defect-avoidance and fault-tolerance will be presented in section 2.1 of this chapter. This work concentrates on two typical networks: the crossbar network and the omega network.

2.1 Classification of Interconnection Networks

2.1.1 General Classification

The study of interconnection networks includes their operation modes, control strategy, switching methodology and network topology [17].

The operation modes are identified as synchronous communication, asynchronous communication or a combination of both. Synchronous communication is required for processing in which communication paths are established synchronously for either a data manipulating function or a data/instruction broadcast. Asynchronous communication is needed for multiprocessing in which connection requests are issued dynamically [18]. The physical restructuring technique is a defect-avoidance method which employs interconnection networks in a synchronous mode. Electronic switch interconnection networks can employ either synchronous or asynchronous modes, depending on the purposes of the WSI systems. Generally, an interconnection network in a WSI system can be designed to facilitate both synchronous and asynchronous processing, which is referred to as the combined mode.

The control strategy is related to the control of interconnection networks. A

typical interconnection network consists of a number of switching elements and interconnecting links. Interconnection functions are realized by properly setting the controls of these switching elements. The control-setting function can be managed by a centralized controller or by an individual switching element. The first strategy is called centralized control. For the physical restructuring technique, the control is accomplished by the physical routing of the signal paths. For an electronic switch interconnection network used in a WSI system, centralized control is employed.

Two major switching methodologies are circuit switching and packet switching. In circuit switching, a physical path is actually established between a source and a destination. In packet switching, data is put in a packet and routed through the interconnection network without establishing a physical connection path. In general, circuit switching is much more suitable for bulk data transmission and packet switching is more efficient for short data messages [17]. In our study, we employed the circuit switching methodology for both the physical restructuring technique and the electronic switches technique.

A network topology is a graph in which nodes represent switching points and edges represent communications. Most topologies are regular and can be grouped into two categories: static and dynamic. In a static topology, links between two processors are passive and dedicated buses cannot be reconfigured for direct connection to other processors. On the other hand, links in the dynamic category can be reconfigured by setting the network's active switching elements. In a WSI system, an interconnection network must be dynamic to accomplish defect-avoidance and fault-tolerant tasks [18]. The setting of the switch states will be based on the distribution of the defective and working cells and the signal paths between them.

2.1.2 Network Topology

Network topology is a key factor in determining a suitable architectural structure. Many such topologies have been developed, and those available for interconnection networks are shown in Figure 2.1. Topologies in the static category can be classified according to dimensions required for layout, specifically one-dimensional, two-dimensional, three-dimensional, and hypercube . We do not discuss static topologies here because of the limitation of their usefulness in WSI systems.

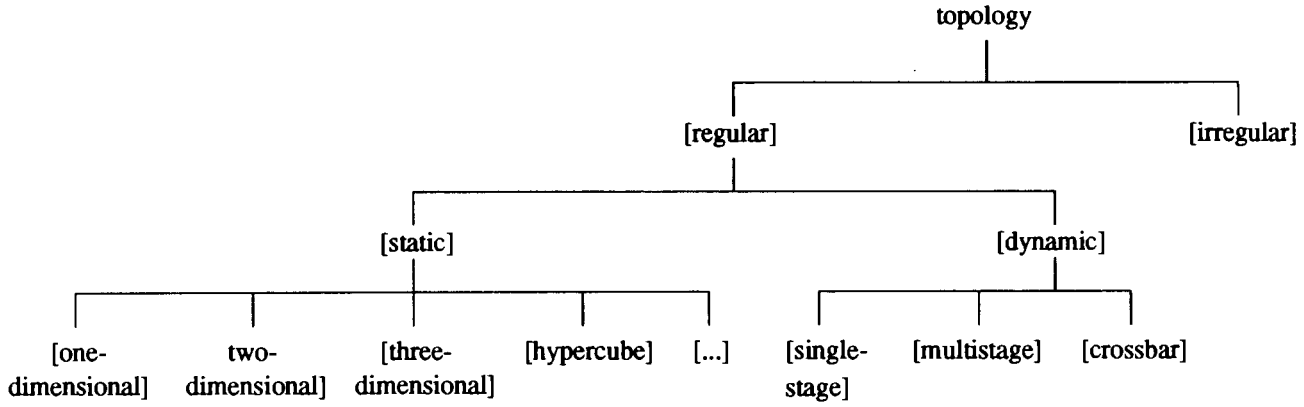


Figure 2.1: Topologies of interconnection networks. (Adapted from [17], © by IEEE Computer Society Press.)

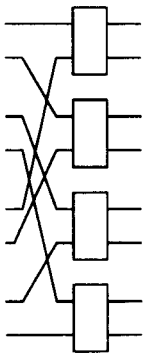
For the dynamic category, topologies can be classified according to their characteristics, which are the number of stages of switching elements, input/output arrangement and functional capability [18]. In terms of stages of switching elements, dynamic topologies can be grouped into three categories: single-stage, multistage and crossbar. In terms of assigning input/output, there are three groups: folded, one-sided and two-sided. In terms of functionality, there are also three categories: blocking, rearrangeable and nonblocking. Some of these concepts, for example, one-sided and two-sided crossbar networks, will be discussed later. At this point we concentrate on

the stages of switching elements.

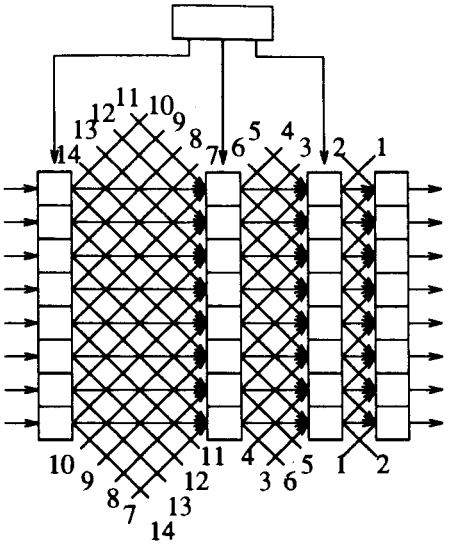
Typical network switch topologies are shown in Figure 2.2. A single-stage network is composed of a stage of switching elements cascaded to a link connection pattern. The shuffle-exchange is a single-stage network based on a perfect-shuffle connection cascaded to a stage of switching elements as shown in Figure 2.2a. The single-stage network is also called a recirculating network because data items may have to recirculate through the single stage several times before reaching their final destination. A multistage network consists of more than one stage of switching elements and is usually capable of connecting an arbitrary input terminal to an arbitrary output terminal. Multistage networks include the data manipulator, banyan, flip, baseline, omega and indirect binary n-cube. Analysis of the topologies of these networks is complex and beyond the scope of this thesis. We selected the most commonly used one, the topology of the omega network, to discuss in detail later on. Even the topology of the omega network is simply a repeat of the single-stage perfect shuffle network, its logic function is the same as other single-path multistage networks. And so the omega network is chosen as our first implementation multistage network [18, 11].

Multistage networks can be one-sided or two-sided. The one-sided networks, also called full switches, have input and output ports on the same side. The two-sided multistage networks usually have an input side and an output side.

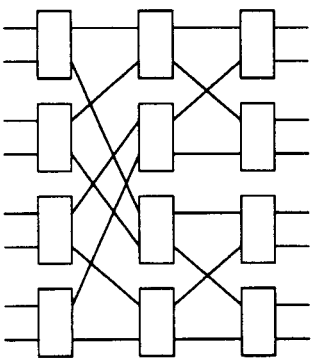
In blocking networks, simultaneous connections of more than one terminal pair may result in conflicts in the use of network communication links. They are all topologically equivalent. A network is called a rearrangeable nonblocking network if it can perform all possible connections between inputs and outputs by rearranging its existing connections so that a connection path for a new input-output pair can



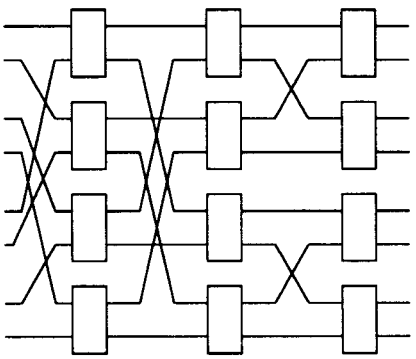
a) 8 x 8 shuffle-exchange



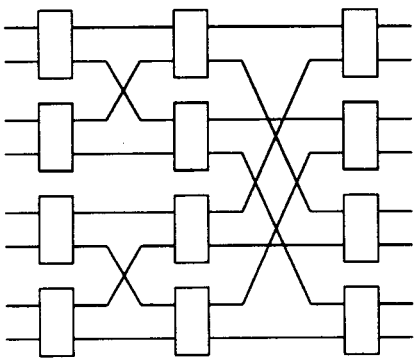
b) 8 x 8 data manipulator



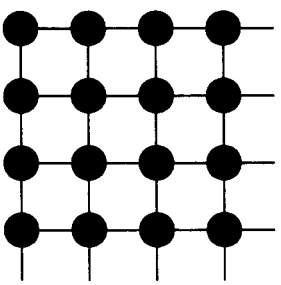
c) 8 x 8 baseline



d) 8 x 8 cube



e) 8 x 8 banyan



f) 4 x 4 crossbar

Figure 2.2: Dynamic network switch topologies. a) Single Stage, b-e) Multistage, f) Crossbar.

always be established. A well-known network, the Benes network, belongs to this class. A network which can handle all possible connections without blocking is called a nonblocking network. In the Clos network a one-to-one connection is made between an input and an output [20]. The other possible case is a one-to-many connections [18].

Classification is a better way to discuss network topologies. In our study, the crossbar network and the omega network are used as defect-avoidance and fault-tolerant interconnection networks. The omega is a multistage, two-sided and blocking network and the crossbar is a two-sided, nonblocking network.

2.2 Crossbar Network

The crossbar network can be used in a restructuring technique for defect-avoidance purposes. Its regular switching element array makes it very suitable to the laser link/cut process [17, 18, 19, 21].

The crossbar network topology is very simple. Figure 2.3 shows the topology of an 8 x 8 two-sided crossbar network. In its general form, a two-sided crossbar switch with N input terminals and M output terminals consists of N x M crosspoints organized as a matrix with N rows and M columns. A connection between input i and output j is established by activating a crosspoint switch placed at the intersection of row i and column j. An important property of a two-sided crossbar switch is the existence of a unique path between every pair of input and output terminals. In traditional designs with electronic switches, the unique path property makes them vulnerable to faults. The failure of any crosspoint in a two-sided matrix disconnects one input-output

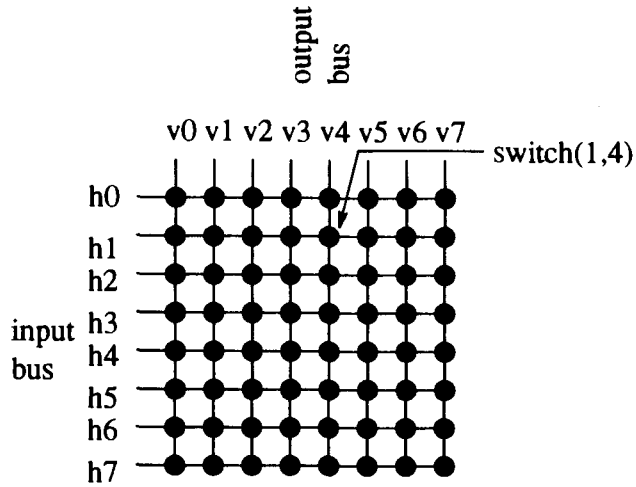


Figure 2.3: The 8 x 8 crossbar network.

pair. An alternate design of crosspoint networks, namely the one-sided crosspoint networks, affords improved fault-tolerance by providing multiple paths for setting up a connection between any two terminals.

Because of the characteristics described above, crossbar networks can be used for the defect-avoidance purposes. In a defect-avoidance system, crossbar networks are placed between intersections of the buses which route signals to different function blocks (cells). Consider two intersecting bus lines, with lines h0 to h7 running horizontally and lines v0 to v7 running vertically. The lines route signals between cells using crossbar interconnection networks. Switches exist at the intersection points. To route a signal from horizontal line h1 to vertical line v2, only switch (1,2) would be activated (see Figure 2.3). To change the signal to bus line v7, switch (1,8) would be turned on. In this manner, the crossbar switch enables horizontal line to be connected to any vertical lines. The crossbar network requires a switch at each intersection point, e.g. 64 switches for the 8 x 8 possible line connections. Hence the crossbar switch offers maximum flexibility at the cost of larger switch area.

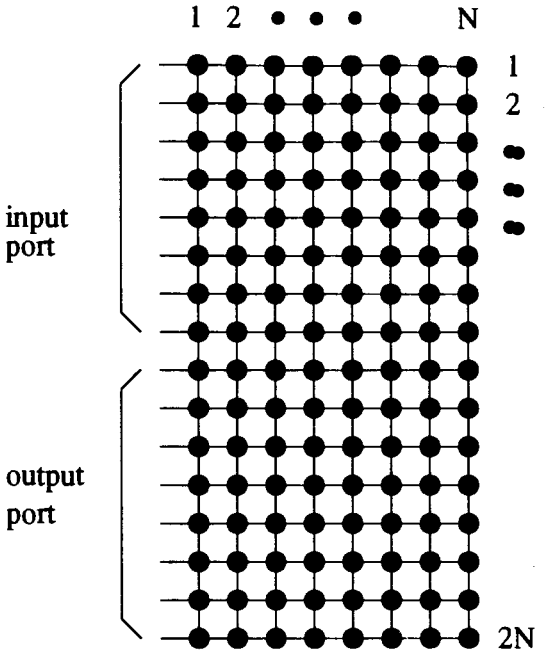


Figure 2.4: 8 x 8 one-side crossbar network.

A one-sided crossbar matrix consists of a set of input and output port lines on the same side and a set of bus lines placed perpendicularly to each other, with switching cells placed at the points of intersection [22, 23]. This is illustrated schematically in Figure 2.4. A connection between two ports is set up by selecting one of the vertical buses and activating the two crosspoints at the intersection of the corresponding port-lines with the selected bus. Thus, any connection can be established through any of the buses, provided that the selected bus is not being used at that time to realize another connection. Since any unused bus lines can be selected to make a connection between the two ports in the matrix, this matrix provides better fault-tolerance as compared to a two-sided matrix. A nonblocking one-sided switch with N input and N output lines can be obtained by means of a matrix with 2N horizontal lines and N vertical buses. The one-sided crossbar matrix in Figure 2.4 with $N = 8$ has 16 input/output port lines and 8 internal vertical bus lines.

The laser link switches used for physical restructuring have high reliability so we do not need to worry about the defect avoidance ability of the crossbar network itself. The crossbar enable maximum flexibility in switching signals to spare lines. Hence we employed two-sided crossbar networks in our research.

2.3 Omega Network

The omega network is among those unique-path multistage networks which have the simplest structure. The omega network occupies relatively little area because of its simple interconnection lines and small number of switching elements. It can be employed for defect-avoidance and fault-tolerant purposes [16, 17, 18, 24].

A N x N omega network consists of $l = log_2 N$ identical stages. Each stage consists of a perfect shuffle interconnection followed by N/2 switching elements as shown in Figure 2.5. The full switch is employed as the switching element which can have one of the four states shown in Figure 2.6. The total number of switches is $l \times 2^{l-1}$ or 12 for an 8 x 8 omega. By convention in the literature, the two input lines are N, W, while the output lines are E, S. However these switches are usually represented by the more compact form of Figure 2.6, rather than a full 4 direction figure. It may either make no connection, send its inputs straight through, swap the inputs, or broadcast one of the inputs to both outputs. However one output of the full switch cannot be connected to both inputs. The perfect shuffle connection has the property of taking an input at a position whose binary representation is $s_1 s_2 \cdots s_l$, and moving it to position $s_2 s_3 \cdots s_l s_1$, The switch can then move the output to $s_2 s_3 \cdots s_l 0$ or $s_2 s_3 \cdots s_l 1$. In order to switch data through the network, each element in the network

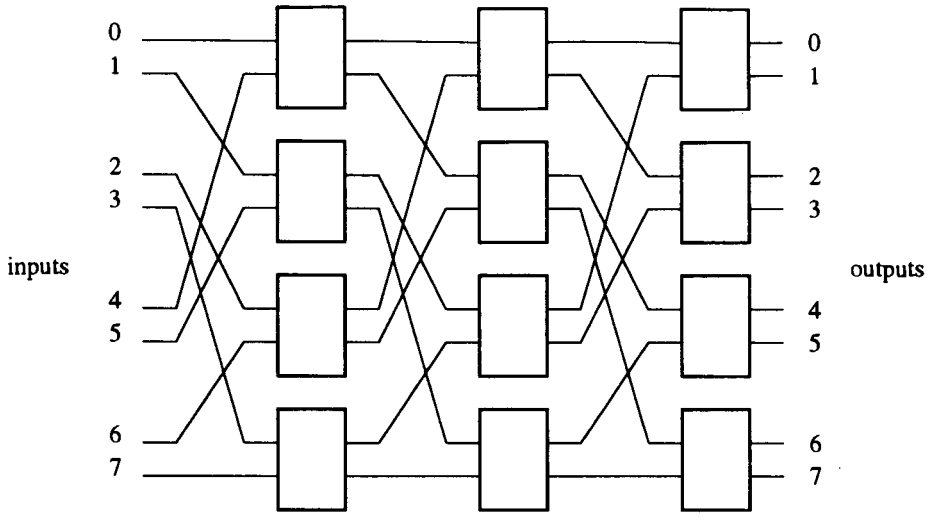


Figure 2.5: 8 x 8 omega network.

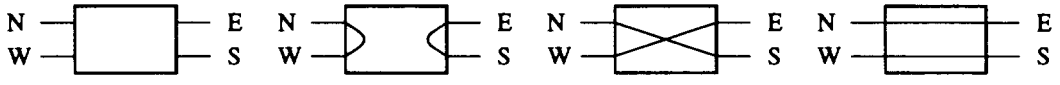


Figure 2.6: Four States of the full switch.

is set in one of the above four states (not necessarily all in the same one) and then the data are allowed to pass from the network inputs to the network outputs. This causes a one-to-one or one-to-many mapping of inputs to outputs. In order to switch data through the network, each element in the network is set in one of the above four statement and then the data is allowed to pass from the network inputs to the network outputs. This causes a one-to-one or one-to-many mapping of outputs.

There is an efficient algorithm for setting the states of the omega network. First, consider switching input number S to output number D. By examining Figure 2.5, it should be easy to see that there is one and only one path between any given input/output pair. Let $D = d_1 d_2 \cdots d_l$ be the binary representation of the output number. Starting at input S, the first switch to which S is connected is set to switch input, S, to the upper output if $d_1 = 0$ or the lower output if $d_1 = 1$. An example is

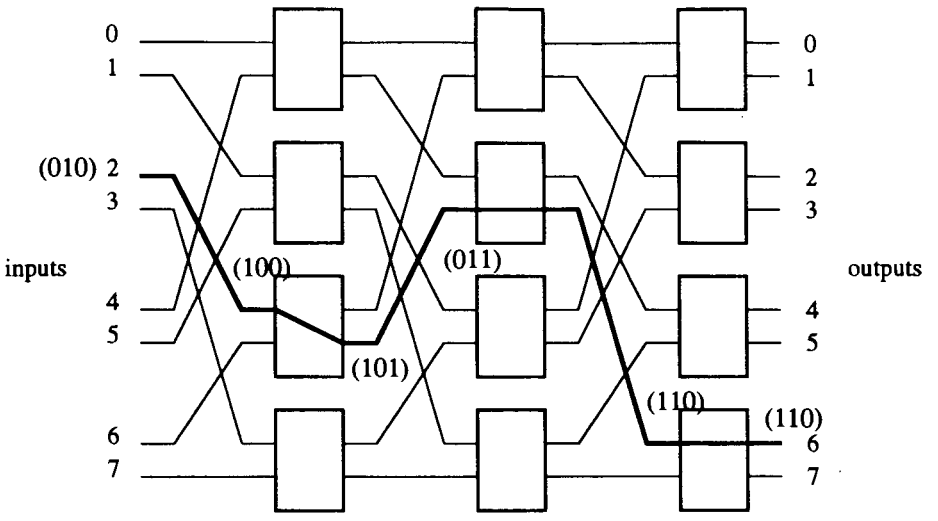


Figure 2.7: The connection of input(010) to output (110).

shown in Figure 2.7 for $S = 010$, $D = 110$. Following the data through to a switch in the next stage, we again switch the input to the upper output if $d_2 = 0$, or to the lower output if $d_2 = 1$. We continue in this manner, switching on d_i at each stage i, until we get to the proper output. It is easy to see that during any given stage i , an input which has been switched to position $s_i s_{i+1} \cdots s_l d_1 d_2 \cdots d_{i-1}$, goes through the perfect shuffle and ends up in position $s_{i+1} s_{i+2} \cdots s_l d_1 d_2 \cdots d_{i-1} s_i$, and is then switched into position $s_{i+1} s_{i+2} \cdots s_l d_1 d_2 \cdots d_{i-1} d_i$. Thus, after $l = log_2 N$ operations, the original input must be connected to output $d_1 d_2 \cdots d_l$. A similar algorithm works by starting at output $d_1 d_2 \cdots d_l$ and working backwards through the network, switching the element at stage i according to s_i. In order to set up a particular mapping of inputs to outputs, we simply follow the above procedure simultaneously for all inputs or all outputs. It remains for us to show that this procedure is complete in the sense that it can set up any mapping of which the network is capable. The algorithm will specify paths for any mapping of inputs to outputs.

Since there is one and only one path between a pair of input/output, the set of paths for a given mapping must be unique, and this must be the set determined by the algorithm. As discussed for the crossbar network, an omega network can be used for defect-avoidance because of its ability to establish a unique path between any input and output. Further more, the electronic switch omega network also has the fault-tolerant ability because its changing of routes is programmable to bring in redundant cells to substitute for failed ones.

2.4 Summary

In this chapter, we have reviewed the classifications of interconnection network topologies, analyzed the topologies of crossbar and omega networks. From this discussion the crossbar is seen to offer the most flexibility at the cost of significant numbers of switches. The omega network offers sufficient switching to interchange every line for a cost of less switches, but does not allow a single input to be connected to multiple outputs. The comparison of the network properties and tradeoffs will be investigated in Chapter 3 and Chapter 4.

CHAPTER 3

A Laser Link Crossbar Network

The Laser Link Physical Restructuring Technique (LLPRT) for WSI mentioned in Chapter 1 has been implemented and tested at Simon Fraser University. Our study used LLPRT to work on a wafer scale system which used the crossbar networks as interconnection networks for defect-avoidance purposes. The research covered in this chapter consists of two parts, designing a program to create automatically a bus system using crossbar networks for interconnections and establishing the parameters for post-fabricating process, i.e. laser linking/cutting process to make routes to connect/disconnect function blocks. We use C Design Language (CDL) to design the bus system and crossbar networks. These designs are implemented using 3-micron CMOS process and 1.5 micron CMOS process. The laser linking/cutting process which was carried on the system included selecting a basic link element and a series of experiments on laser linking and cutting.

3.1 The General Structure of the System Using Crossbar Network

One successful design for wafer scale consists of fabricating a wafer using redundant circuitry and interconnections as shown in Figure 3.1 [2]. The "cells" are function blocks with many copies of the identical blocks for redundancy. These function blocks are interconnected by the bus system to fulfill the wafer scale system task.

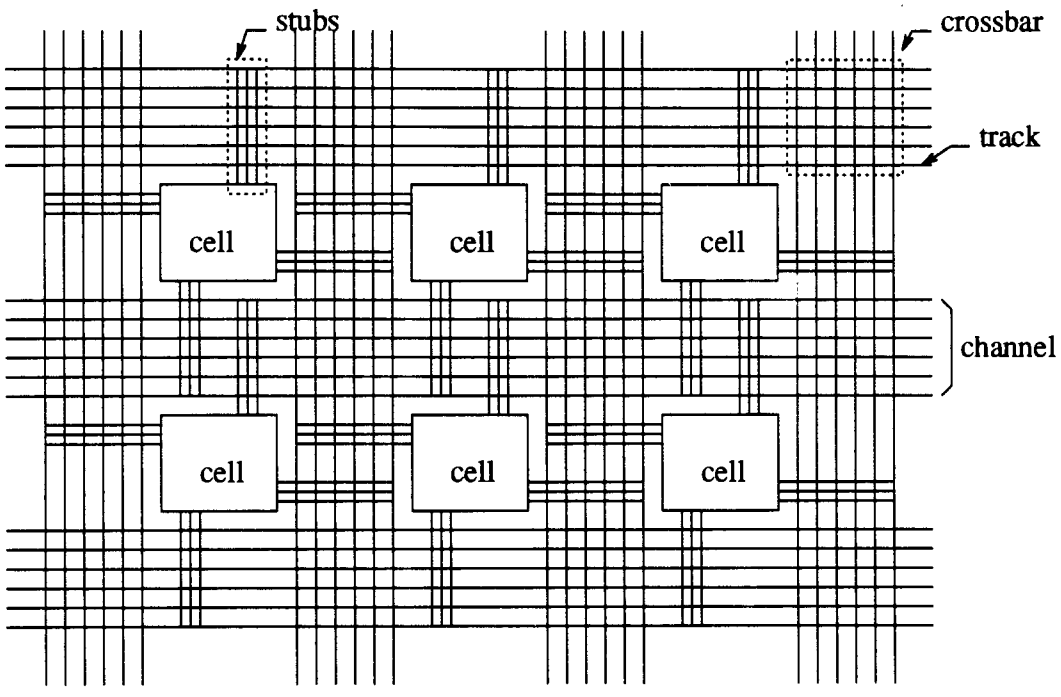


Figure 3.1: The bus system using laser link crossbar network.

These cells are connected to bus tracks with stubs. All kinds of signals such as digital signals, analog signals and power supply are transferred through these stubs. These stubs can be connected or disconnected to bus tracks through link elements which we called link switches. These link switches are specifically designed for LLPRT and will be described in detail. A stub is designed to connect a cell's I/O to several

bus tracks for redundancy purposes. Crossbar networks are put at the intersections of the bus tracks which are parts of the bus system. Communications between cells and between cells and input/output pins are established by these networks. The number of bus tracks in a channel and dimensions of interconnection networks are determined by the number of signals and redundancy considerations. In general, the number of tracks per channel ranges from 5 to several hundred, the number of cells ranges from 10 to several hundred. The number of I/O pins can range from 5 to several hundred [25].

The test strategy of a network is a very complicated problem which was not explored in this thesis. Generally in a WSI system, bus systems are tested first, then each function block (cell) is tested. At the time of fabrication and before the beginning of testing and restructuring, all active devices are isolated from the power bus. The cells are powered up later on using special oversized links provided to connect the higher currents which are associated with power distribution. This is done because a large fraction of the failed cells have power shorts. The routes of the cells can be changed during testing. For some designs, it was possible to observe cell outputs on the package pins. For other designs, a wafer scale test track was connected to an output line, a test was performed, and the test connection was removed.

3.2 A Physical Restructuring Technique

3.2.1 Laser Restructuring Technology

Laser-Link Physical Restructuring Technique (LLPRT) uses a laser beam to change interconnection routes of wafer scale integration systems. After a WSI chip was fabricated by a standard CMOS process, it is put on a X-Y position table. The chip can be moved in x, y and z directions with the accuracy of 0.1 micron. The interconnection networks on the chip consist of laser link switches which can change the route of bus systems to connect or disconnect function blocks. A laser beam is employed to link or cut parts of these link switches to fulfill the route changing task.

The laser linking/cutting process needs specialized equipment to position the chips and to link/cut connections to cells. This apparatus will be introduced briefly. Link switches are specially designed for the laser linking/cutting technique. The linking/cutting procedures of this process are also reviewed.

3.2.2 Laser Link Switches

The laser linking/cutting on a wafer is performed on a special kind of structure: the laser link switch. There are four kinds of laser link switches [3] which were first developed in MIT/Lincoln Laboratory. The four different structures are a vertical link in which a diffused metallic interconnection is formed between two metal layers separated by an insulator, a lateral diffused link in which a diffused region is formed between two implanted areas, a lateral link made by diffusing aluminum into polysilicon, and

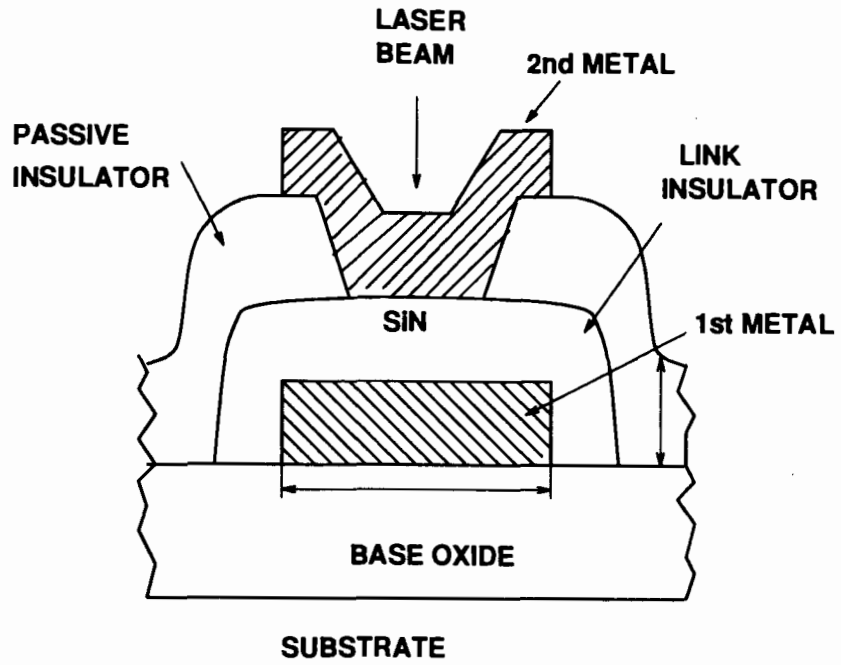


Figure 3.2: Structure of the vertical link.

a lateral intermetal connection using the laser carbonization of polyimide [3].

3.2.3 Vertical Link

The structure of the vertical link is shown in Figure 3.2. It consists of the first metal, the second metal and a link insulator. To form a vertical link, a laser pulse melts both metal and insulator materials to form a weld between these two metal films. The two metal layers are thin film alloys ($Al : Si1\% : Cu4\%$). The link insulator is a film of amorphous silicon (hydrogenated to produce an extremely high resistivity) [2] or silicon nitride. A one-micron layer of insulating material provides the intermetal isolation at other locations.

A connection formed from the vertical link has a low resistance which ranges from

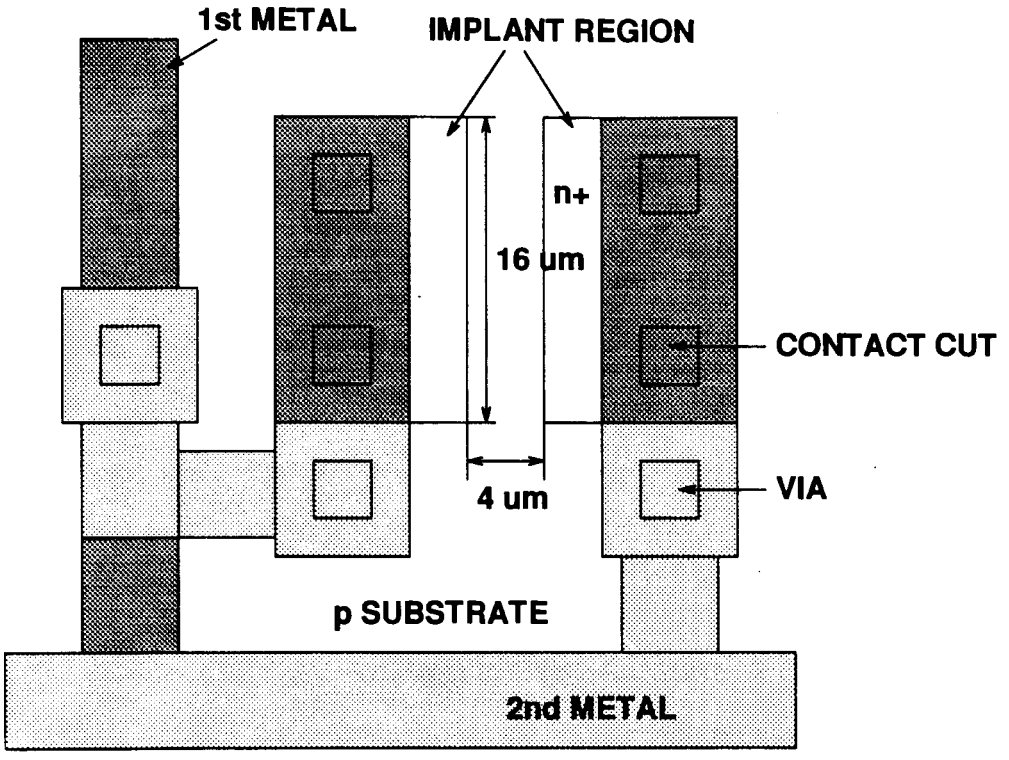


Figure 3.3: Structure of the diffused link switch.

1 to 10 Ω. Connections can be made with high yield and are reliable (99.999%). A disadvantage of the amorphous-silicon link is that a nonstandard material in CMOS is required. Because of its extensive application and the fact that no research was conducted on this kind of link, it is only briefly introduced.

3.2.4 Diffused Link

The diffused link can be fabricated in a standard MOS process (see Figure 3.3). It consists of two diodes formed by implantation into the substrate or a CMOS tub and connected to metal buses by contact and via cuts. The two back to back diodes, separated by a gap equal to 4 μm (allowed diffusion separation) in the CMC 3 micron

CMOS, have the same impedance as that of a reverse biased diode. When the gap is exposed to an argon laser pulse focused to a width which ideally would be about that of the gap, the melting causes dopant diffusion into the gap and forms a connection with low resistance. The typical value of formed link resistance is 75 Ω. Link resistance is dependent on the implant sheet resistance. Hence links made with n implants into p tubs have lower impedance due to the lower n+ sheet resistance. The major visually observable effect of the laser pulse is damage to the passivation and intermetal oxides, which is strongly dependent on the glass used by the CMOS process. Optimum laser pulse parameters vary for wafers built in different fabrication processes, and will be discussed later in the thesis.

The resistance of this link also depends on the width of the diffusion area. A model of the laser melting of Si has been developed [25], in which the heat diffusion equation is solved with a Gaussian source function. Some features of this model include thermal conductivity and reflectivity that vary with temperature, and a beam penetration which, for silicon , may be expressed as an exponential form for the long wavelength or a delta function for the short wavelength. For an argon ion laser running all lines, the 1/e penetration depth in solid Si is approximately 1 μm, much less than the 10 μm of a NdYAG laser at its fundamental 1.064-μm wavelength. Thus using the Argon laser results in a greatly increased stability of the melting process.

In the Engineering Laboratory at Simon Fraser University, we use the diffused link switch to construct the crossbar network and the WSI bus system. The diffused link switches have been successfully linked and cut with our laser equipments.

3.2.5 Laser Table and Operational Procedures

The set up used for laser linking consists of a linear-induction motor X-Y table with laser interferometry position control, an argon ion laser generator, a function generator to control the duration of laser beams, a videocamera to view the wafer on the table and a computer to control the system. The system is shown in Figure 3.4.

When a packaged wafer or chip is put on the table for physical restructuring, its pins are plugged into a socket which is connected by cable so that all of the electronic signals can be measured. This socket can be moved in three directions X, Y and Z (vertical). The movement in X-Y must position the focused laser link switches exactly under the laser beam for the laser linking/cutting operation. The movement in Z direction raises the wafer to allow focusing of the lens system of the laser beam and videocamera (which also focuses at the same point). The light beam from the laser passes through an electro-optic shutter which , under computer control sets the duration (from 2 μs to 100 ms). The energy in a laser pulse is used to form the laser connection or to segment a metal line.

The laser is a 5 W argon ion laser (Model 305 of INNOVA 300) [28] as shown in Figure 3.5. The laser is connected to a synthesized function generator (Stanford Research Systems Model DS 345) which sets the duration of the laser beam. The laser beam is directed with dielectric mirrors through a microscope system. The beam is focused via a 50x objective lens with a 0.6 numerical aperture on to the wafer surface where the spot diameter is typically focussed to 1.2 microns full width at half maximum. In the literature the place where the laser beam melts the material is called the "zap" point. A videocamera views the wafer through the same optics and

the video signal is processed by an autofocus controller that moves the stage vertically to maximize contrast on the horizontal scan lines.

The laser table is controlled by the Intelligent Digital Axis Controller (IDAC) which was fabricated by ANORAD Corporation [29]. The hardware parts of the system consist of a vibration isolated floating table, a control processor, a position measurement laser interferometry system and a 386 PC controlling the system. The program controlling the system was coded by L. Nathawad and D. Bergen who were undergraduate students in the School of Engineering at Simon Fraser University. The program written in "C++" in the MS Windows environment operates the controller through a serial port. This program also controls the function generator for the electro-optic shutter and the Z-axis control.

3.3 Measurement Results

The advantage of physical restructurable technique is that it has no power consuming elements in the bus system. The only component that is different from normal metal line is the diffused link switch. Hence the setting of the signal path using laser linking/cutting of the bus system must be considered.

To obtain high speed operation the resistance must be minimized. During the laser linking/cutting processing many factors influence the resistance of the laser link switch in a given process: laser power, pulse duration, number of zap points and separation of the zap points. In this section we have optimized the values for CMC's 3 micron CMOS and summarized the results with some typical values through a series of experiments.

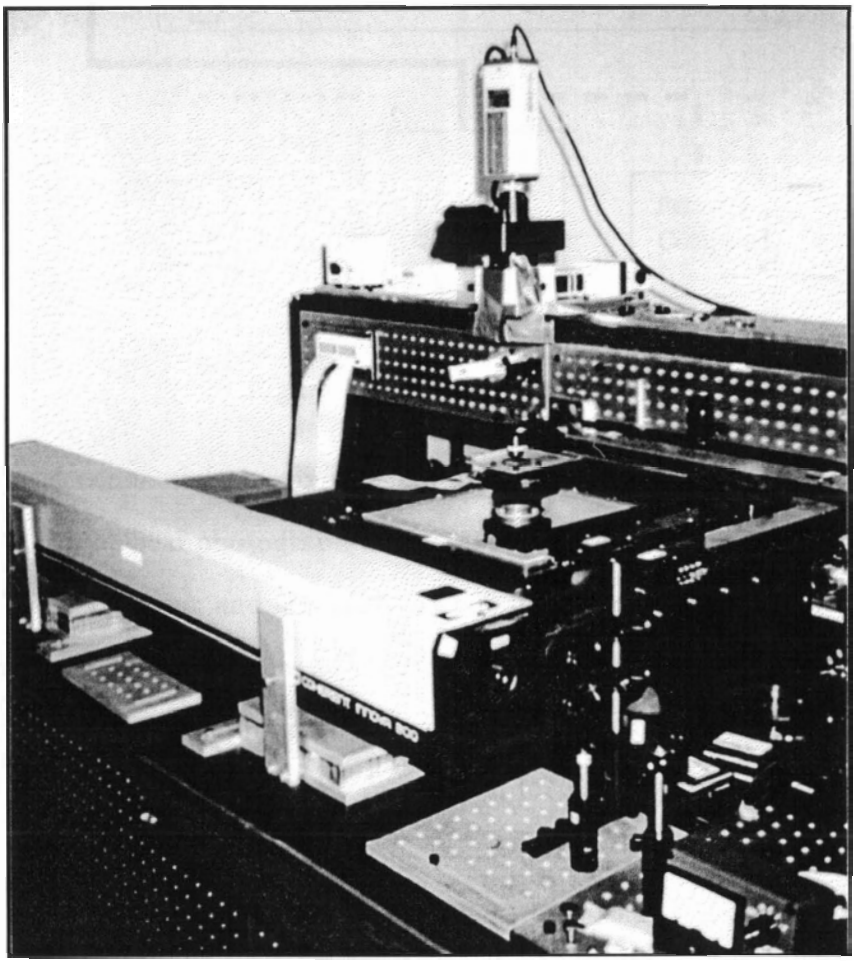


Figure 3.4: Laser and micropositioning system.

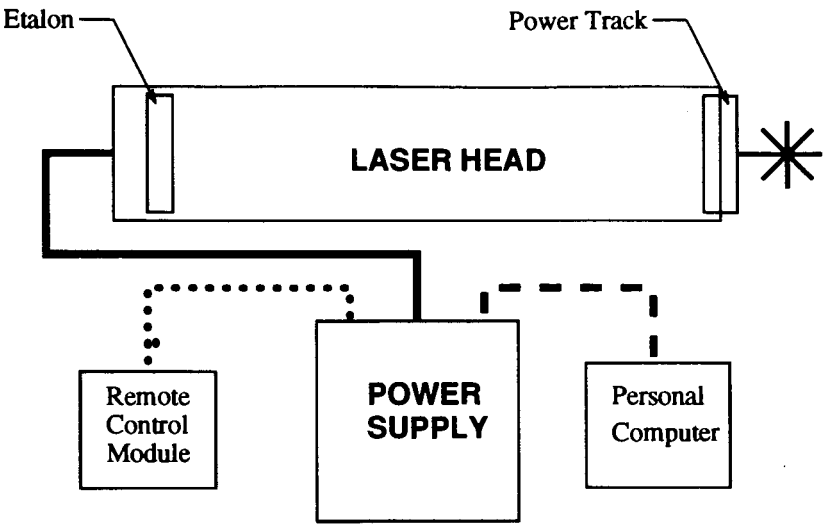


Figure 3.5: Argon laser system.

To form a connection, a group of points are zapped by a laser beam as shown in Figure 3.6. An optical photograph of the laser links is shown in Figure 3.7. The laser can also be used to cut and segment the bus lines (see Figure 3.6) using parameters similar to those of laser linking. Figure 3.8 shows a laser cut on a second metal line. The following section summarizes a series of experiments, optimizing the amount of power, duration of the pulse and zap position for the CMOS 3 process.

3.3.1 Laser Power

It is vital to minimize the resistance when the link is formed. Meanwhile it is also important to use a smaller power to form the connection, because the lower the laser power, the less chance of damage to surrounding areas. For industrial development it is necessary to keep high yield while using physical restructurable technique. Figure 3.9 shows the curve of resistance versus laser power. The measurement results were obtained by using 6 laser zap points and 300 μs laser time pulse duration. From 2.2

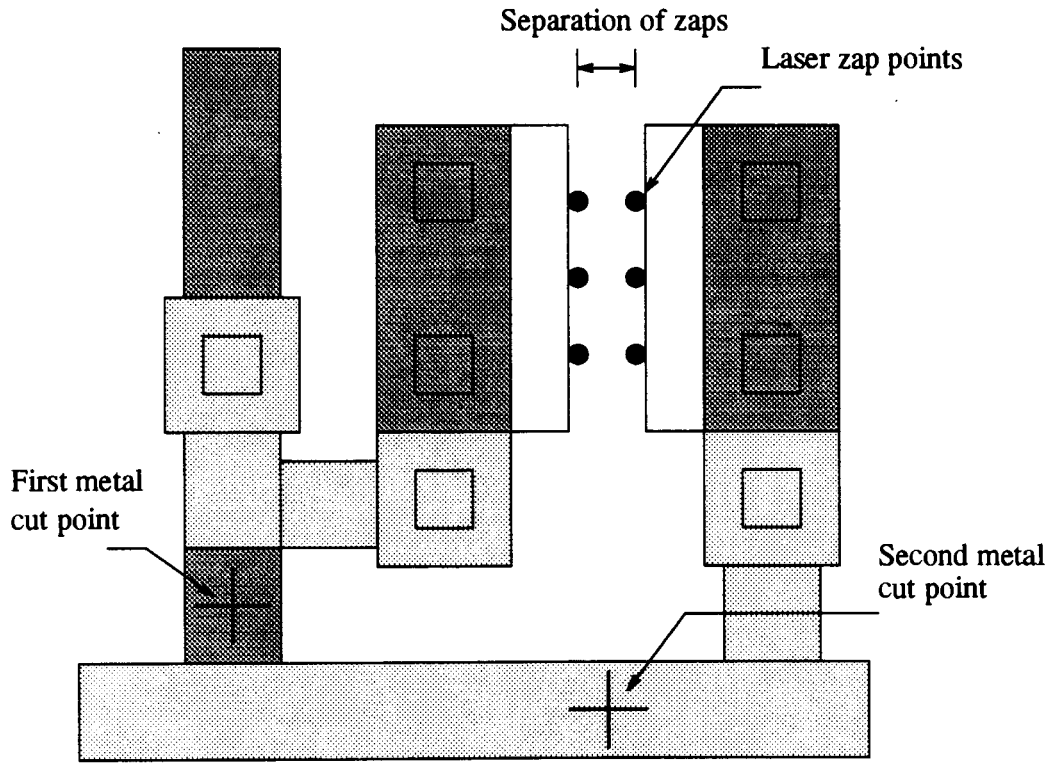


Figure 3.6: Laser linking/cutting positions.

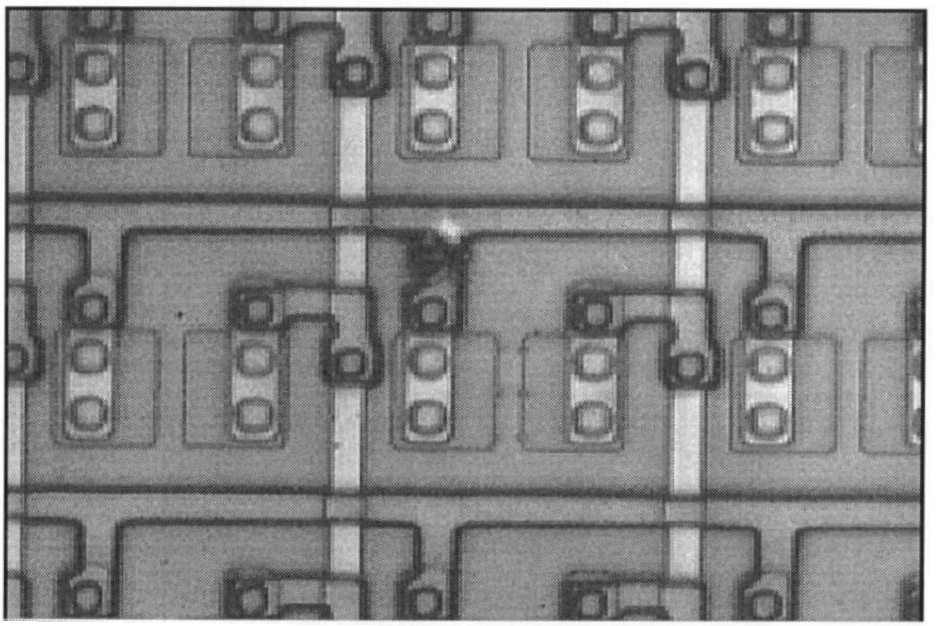


Figure 3.7: CMOS 3 microns laser link switches, with a connected link shown in the centre.

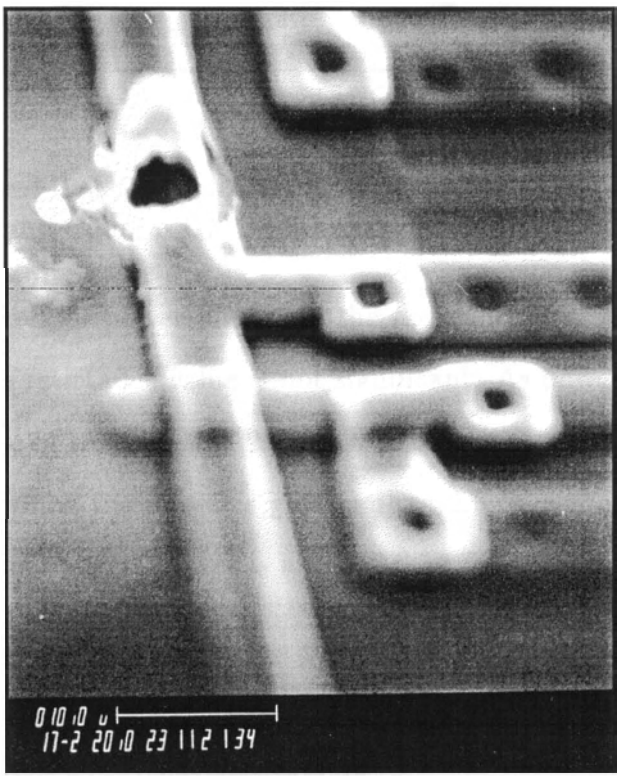


Figure 3.8: SEM photomicrography of laser cutting on a Metal 2 line fabricated by CMC CMOS 3 process.

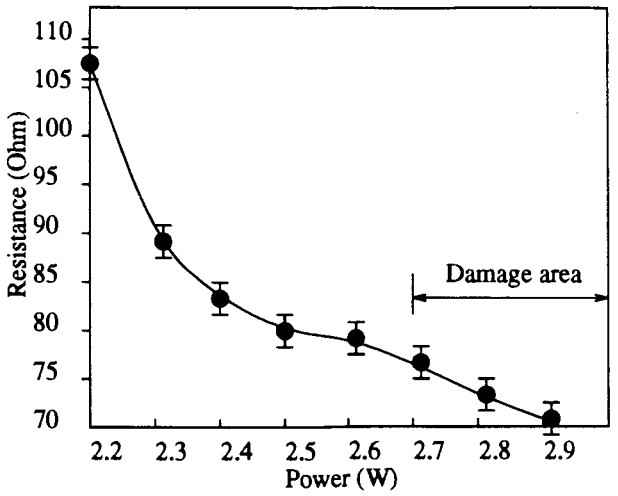


Figure 3.9: Measured link resistance vs. laser power for 6 laser zap points at diffusion edge, 300 μs pulse duration on CMOS 3 links.

W to 2.8 W, as the laser power increases, the resistance decreases. But after 2.6 W, the variation of the resistance value is not significant. As power varies from 2.2 W to 2.6 W, the decrease of resistance is 29.4 Ω, while from 2.5 W to 2.8 W, the decrease is 6.7 Ω. However when the power is greater than 2.6 W some damage occurs in the link area. Hence the optimal power is about 2.6 W.

3.3.2 Pulse Duration

We tested the influence of the duration of the laser beam on the laser link resistance. The experimental results are shown in Figure 3.10. These are obtained with optimal 2.6 W laser power and 6 laser zap points of a given pattern. Typically, when the duration is 20 μs the resistance is 190.2 Ω. The resistance is 112.15 Ω when the duration increases to 100 μs . The resistance saturates after 200 μs and is nearly optimal at 300μs with a resistance of 78.1 Ω. If the duration is too long the accumulation of heat

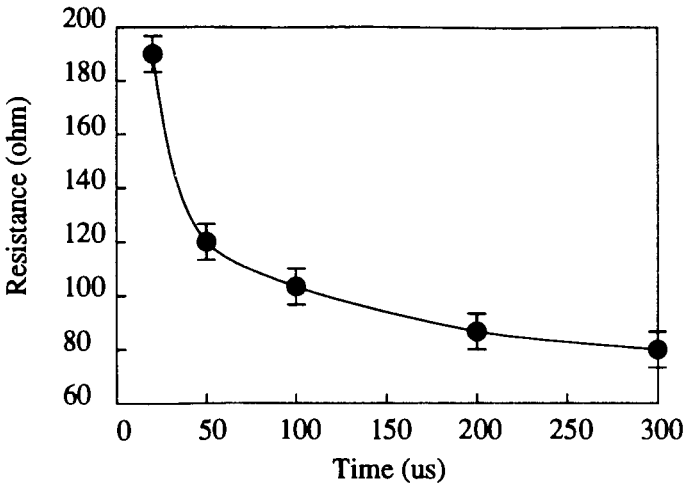


Figure 3.10: Measured resistance vs. laser pulse duration. Results occurred at 6 zap points at diffusion edge, 2.6 W on CMOS 3 links.

caused by the laser beam was observed to damage the surrounding area. Hence the optimal duration is $300\mu s$.

3.3.3 Separation between Laser Zaps

While it is possible to control the laser spot size, one cannot do so without losing considerable laser power given the requirement for at least a 50x objective to position visually the spot with sufficient accuracy. With a given dopant gap width (4 μm for the CMOS 3 process) that is larger than the laser spot size (1.2 μm), it was found best to have 2 rows of zaps separated by a distance centered about the midline of the gap. The separation between laser beam zap sites strongly affects the link resistance. From Figure 3.11, we can see that when zap sites are close to the edge of the diffused area, the resistance is small, however, when the sites close to the midline of the link switch, the resistance increases. When sites are on the midline, the resistance is infinite, which means the link has failed. Such a situation can be explained according to the

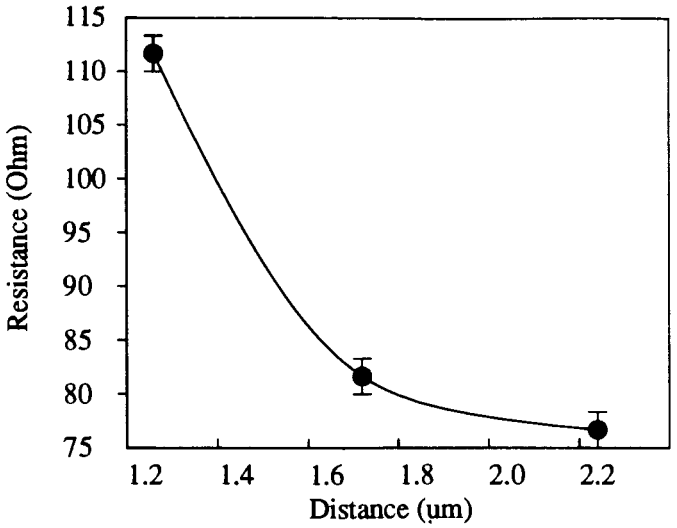


Figure 3.11: Measured resistance vs. separation distance of laser zaps (6 laser zap points, 300 μs pulse duration) for a 4 μm implant gap, and a 1.2 μm laser spot size.

mechanism of the diffused link. When laser beam zap points are on the midline, the silicon in the diffused area does not reach sufficiently high temperature for the silicon to melt and the dopant to diffuse into the gap area and so a conductive channel cannot form. When the laser beam sites are near the edge of the diffused area, the doped silicon will melt and dopants diffuse into the unimplanted area throughout the gap width. The closer the laser beam sites are to the diffusion edge, the better the effect of laser driven diffusing. However if the zaps are at the diffusion edge, damage can occur to the link area. Thus it is necessary to assign the laser beam sites close to the edge of the dopant area, but not touching that implanted region.

3.3.4 Number of Laser Zap Points

Changes in the value of resistance versus the number of laser zap points are shown in Table 3.1. Under a given distance (2.2 μm) and a given power (2.6 W), the value of

Table 3.1: Measured link resistance vs. number of zap points in the gap. Links were at the diffusion edge. (300 μs pulse duration at 2.6 W power)

Number of Holes	Resistance (Ω)	Error (Ω)
4	133.27	3.8
6	78.1	3.4
8	65.35	3.6
10	66.77	3.4
12	65.94	3.5

the link resistance decreases as the number of laser zap points increases. This occurs because there are more dopants diffused into the area. On the other hand, it takes more time to make connection points. When the number of laser zap points is four, the resistance is about 134 Ω. When the number is six, resistance decreases to 78 Ω. When the zap numbers are 8, 10, 12, the resistances are around 66 Ω. The resistances do not change much when the number changes from 6 to 12. It is best to restrict the number of laser zap points to between 6 and 8.

3.3.5 Laser Line Cutting

The physically restructurable technique includes disconnecting or cutting the bus lines. The laser link switch not only provides an area for linking, but also permits bus lines to be cut at both the first and second metal lines. These cuts tend to be quite smooth with damage confined to the cut area (see Figure 3.8) and no splatter of aluminum beyond the cut points. When cut was made, metal flowed away from the cut point and piled up at the edge of the cut. The power needed to cut the metal line was also tested. As the above discussion indicates, it is better to use small power

to reduce the damage to the oxide layer. When the laser power is as low as 0.8 W the laser beam cuts metal effectively. When the power is lower than 0.8 W, it can not fully cut the metal line. To provide a safety margin in power, 1.2 W was used as working power. We have tested more than 50 samples using the 1.2 W power, all of cuts were successful. We have cut the metal lines for both first and second metal layers using the same power and pulse duration with no failures. Cut resistances exceed 1 MΩ in all cases.

3.4 Layout Tools for Crossbar Networks and a Laser Link Bus System

Designing a bus system for a WSI system is difficult because the post-fabrication process requires the accurate positioning of the laser links. As we described in previous sections, the laser linking/cutting is an operation associated with particular areas of a link switch. We need to put the laser beam in the gap between diffused regions for linking or put it at the centre of the metal bus for cutting. With the precision and time needed, it is impossible to process a WSI system using manual operations. In practical linking systems, the linking/cutting positions and laser settings are all set automatically. In this automatic process, the data on link switch positions must be supplied to the computer.

A wide range of CAD programs exists to design digital and analog circuits from simple layout tools like KIC to complex tools like Cadence. However such design tools do not take into account the difference between regular circuit design and laser links.

Unlike regular circuit devices, the actual physical position of the laser links is very important, and the exact location or distribution of the links affects the interconnection process. For example, the distance from link point to link point has a significant impact on the duration of the linking process. Also the links can be thought of as an ordered array of switches, so that the existence of a laser link on one line will require the location of corresponding connections at several other points where the row and column channels intersect. In addition there is a natural advantage to have a CAD tool, that is able to automatically generate the laser link distribution for a wide range of large-area systems. This significantly reduces the time necessary to go from the design of the processor blocks, or cells, to the full wafer layout. These factors have directed this thesis project towards the creation of an algorithmic based floor planner to generate both the bus lines and link distribution [26]. This CAD tool uses the C Design Language (CDL) [39] as its basis. CDL creates complex layout designs using a hierarchical layout technique and module generators to help make large chip layouts more manageable.

3.4.1 Advantages of Algorithmic Bus Design

The availability of the full C programming language within CDL enables a number of calculations which greatly expand the capabilities of the CAD tool over that of simply generating tracks and links. Many calculations are most usefully done as part of the design program. For example, wire capacitance and resistance can be generated directly for each line. As each link contributes a finite amount of capacitance to the line, such factors are important in designing to maximize for system speed. In some situations several sizes of links may be useful, each with its own capacitance and

resulting resistance when connected. Such values are simple parameters transferred from the link design. Such variable link type designs are especially true in Wafer Scale Transducer Arrays, where significant amounts of current may be conducted from a driver circuit to one of many redundant transducer cells [27]. In addition the wafer-scale power grid is best generated using these CAD tools, and it requires significantly different links than the signal laser links.

One very important factor is the actual area taken up by the laser link bus channels. Reports that automatically generate the bus channel area aid in rapid design improvements. Potentially, optimization procedures can automatically shift the wire positions to minimize the area devoted to links, and thus reduce wafer fraction taken up by signal busing. Another important factor of the reporting ability is that the exact physical location and orientation of all laser link structures can be directly output to a data base. Such data bases are needed for assigning and routing CAD tool that controls the defect avoidance signal paths created on each wafer by the laser system. An algorithmic generation routine can insure against subtle shifts in link position which may create problems in actual structures. For example, the misalignment of fabrication layers and variations in laser link orientation require very stringent accuracy in laser beam positioning.

One interesting possibility is that the wire size required for long distance lines (about 5 microns) is much greater than is needed within the laser link itself. With programming it is easy to have wafer length lines that narrow down at the laser link site. Such narrow lines actually enhance the ability to cut bus lines.

Simple parameter changes, and choice of appropriate laser link base designs are all that is needed to change the target design technology. This becomes very important

because the designs are implemented in three CMOS technologies: CMC 1.2 and 3 micron, and Mitel 1.5 micron. This design feature also enables design rule checking to be done on the bus floor plan separate from that of the individual cells, giving quicker results.

As with any program, once a procedure is developed for one design, it can be employed with little effort in future work. Many of these procedures generate choices in various bus parameters which can significantly affect the bus/link area. The general CAD tool developed for this thesis was designed to take in a file consisting of a basic description of the target system, such as number of rows and columns of cells/buses, numbers of lines for each row/column and which buses must make connection to a given type. These input files can be rapidly changed to satisfy the equipment of the target system. The ability to rapidly generate wafer-scale floor plans, visualize them, get reports on the resulting parameters (area, capacitance, resistance, etc.), and modify an entire design at once by adjusting some inputs, greatly aids the design process.

In this project, we first designed two simple crossbar networks to test the CDL tool. Then we designed a bus system using the new CDL based CAD tool for a Wafer Scale Transducer Arrays [27].

3.4.2 The Layout of Crossbar Networks

The 8 x 8 crossbar network is a typical network and the basic part of our system. An 8 x 8 two-sided crossbar and an 8 x 8 one-sided crossbar network were designed to test themselves and the laser linking/cutting process.

The layout of the two-sided crossbar network is shown in Figure 3.12. This is an 8 x 8 link switch array. To take advantage of the CMOS process, we use eight Metal 1 lines and eight Metal 2 lines. The Metal 1 lines run vertically and Metal 2 lines run horizontally. Link switches are put at all of the intersection points. The structure of the one-sided crossbar network is similar to that of the two-sided crossbar network. The structure of the link switch is shown in Figure 3.3. The two metal stubs are designed to connected to Metal 1 and Metal 2 tracks. They are separately connected to two implanted areas. The gap between these two areas are left for the laser beam to connect.

The program to generate the two-sided crossbar network is given in the Appendix B. The link switch was first coded as a function in CDL. When generating the two-sided crossbar network, we called the function 64 times using loops with row and column positions. It is also easy to generate the one-sided crossbar network. An 8 x 8 one-sided crossbar network is actually an 8 x 16 switches array. To generate the one-sided crossbar network, we simply need to change several parameters instead of changing the program. These circuits were fabricated using the CMC CMOS 3 process. The fabricated 8 x 8 two-sided crossbar network is shown in Figure 3.13 and the 8 x 8 one-sided crossbar network is shown in Figure 3.14.

3.4.3 Lplatform CAD Tool For The Layout of the Bus System

During design of the layout of the bus system, we developed an advanced algorithmic tool called Lplatform. Because the bus system was complex, we divided it into several

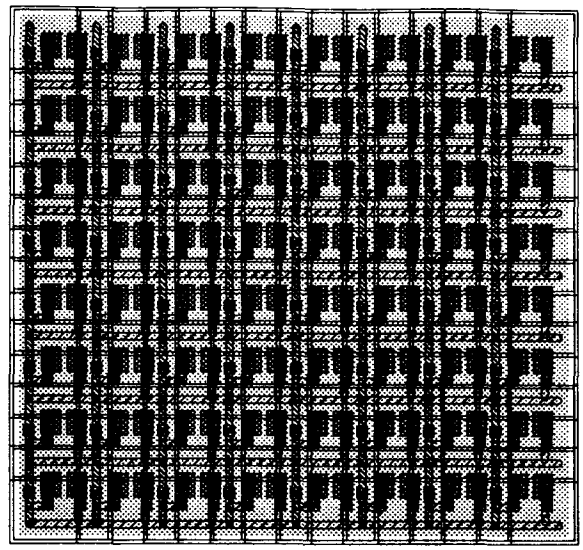


Figure 3.12: CMOS 3 micron layout of the 8 x 8 two sided crossbar network for laser link technique (365x346 $microns^(2)$).

hierarchical modules (which are equivalent to instances in the KIC layout tool). These modules were coded separately in different functions and then were integrated into the main function.

The application of a crossbar network in wafer scale bus system is shown in Figure 3.1. Horizontal and Vertical bus lines are made of Metal 1 and Metal 2. Crossbar networks are placed at intersections of horizontal and vertical bus lines. A crossbar network consists of link switches. Each link switch can connect or disconnect the Metal 1 and Metal 2 tracks that cross it. Function blocks (cells) are separated and connected by the bus system, each signal is connected to one bus line. For every I/O signal, related bus lines are connected to package pins. For lines between cells, several bus lines are connected to test those signals.

The bus system was tested using a real circuit for a WSI system, a Visual to

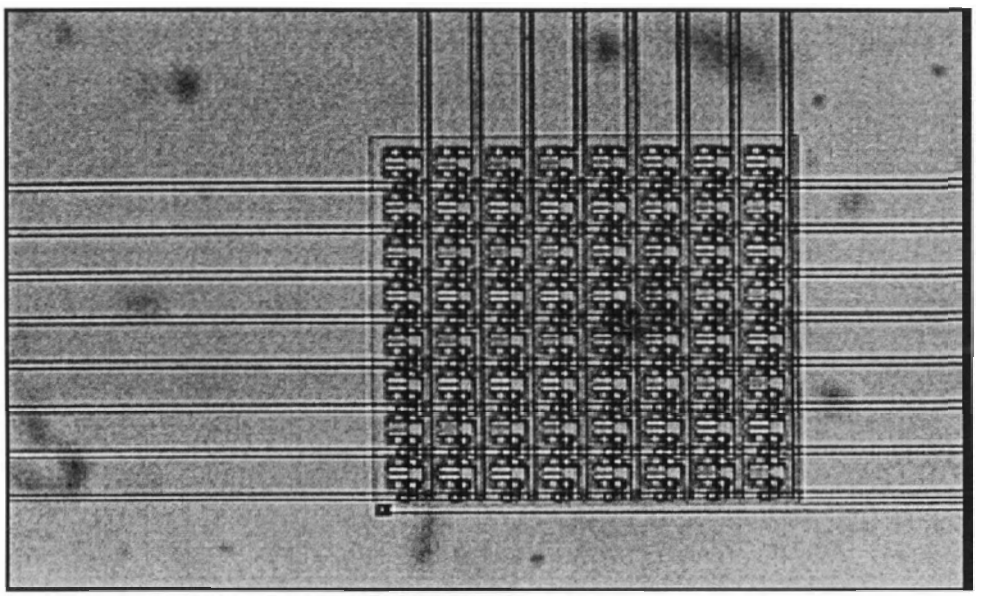


Figure 3.13: Optical photomicrography of CMOS 3 micron 8x8 two sided crossbar network

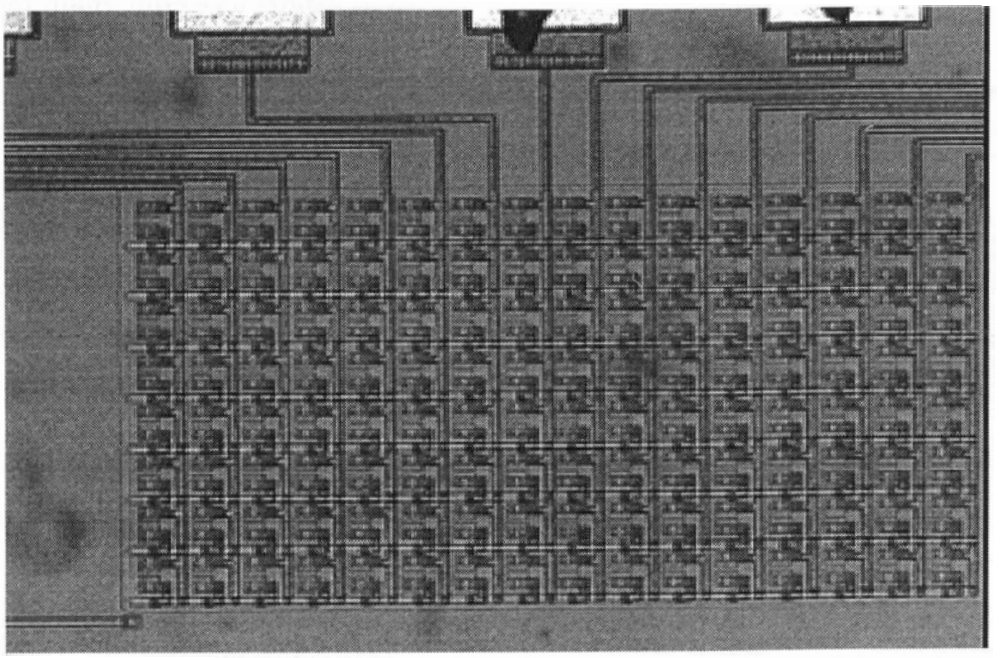


Figure 3.14: Optical photomicrography of CMOS 3 micron 8x8 one sided crossbar network

Thermal Converter for which a bus had been previously designed using KIC [27]. The Converter has 24 I/O signals. The signals from cells or bonding pads are connected to bus tracks with stubs. There are several link switches at the intersections of the stubs and tracks. There are bus tracks for all of the signals. The thermal pixel arrays are to be inserted with KIC. The thermal pixel array is a simple case because it does not need many inter-cell connections.

We use CDL to design the system, making full use of its flexibility. Because the bus system needs to be changed in size to meet different requirements, different wafer systems have different number of circuit blocks and full WSI systems involve hundreds of cells. For illustrative purposes, we give examples with relatively small number. For example, if a system needs the bus system to connect 9 different circuit blocks, the bus system should be 3 x 3. If there are 20 circuit blocks in a wafer, the bus system could be designed in the 4 x 5 scale. Examples of these layouts are shown individually in Figure 3.15 and Figure 3.16.

We designed two different systems. One is a single track system and the other is a double-track system. The distances between bus lines are equal in a single-track system. The distance is the same as the size of the link switch. The link switches can be put at each side of the bus lines. In a double-track system there are two different distances between bus lines. One distance is the minimum spacing between metal layers according to the design rules. While the other is the size of the link switch. The link switches in a double-track system can not be put at both sides of a bus line. This improvement saves one 30% to 40% of the bus area required by a single-track system (see Figure 3-15 for the single-track system and Figure 3-17 for the double-track system).

Although the single-track system consumes more area it has more crosspoint link sites available and thus more flexibility. In the design, we used several functions programmed in CDL to generate the different parts of the system. The function 'column' generates a column of bus lines. The function 'row' generates a row of bus lines. The number of bus lines in a column or in a row can be adjusted by changing a function parameter. The number of columns and rows are determined in the main function. For example, if we need a 3 columns and 3 rows system, we can call the function 'column' and 'row' three times in a loop. The function 'rlcross' places link switches at the interconnection of the system except the left side. The function 'lcross' places link switches at the interconnection of left side column. The function 'ccell' creates the stubs and link switches which are used to connect the input/output signals to bus lines. The length of a stub and the number of link switches on a stub, the tracks they connected to, and the number of columns and tracks are determined by the data array in the program. In future this data will be loaded in as an input file.

To generate a double-track system, we add a switch and several more functions in the program. If the value of the switch is set at 0, the program can generate a single-track system. If the value of the switch is set to 1, the program can generate a double-track system. The function 'srow' creates a row of double-track bus lines. The function 'scolumn' creates a column of double-track bus lines. The function 'rlcross' places link switches at the intersection of columns and rows (except the left side column) in a double-track system. The function 'lcross' puts link switches at the left side column. These functions are all activated using a simple defined data switch.

The program is listed in the Appendix C. Using this program, four different layouts have been generated: Figure 3.15 shows a 3 x 3 single track bus system, Figure 3.16

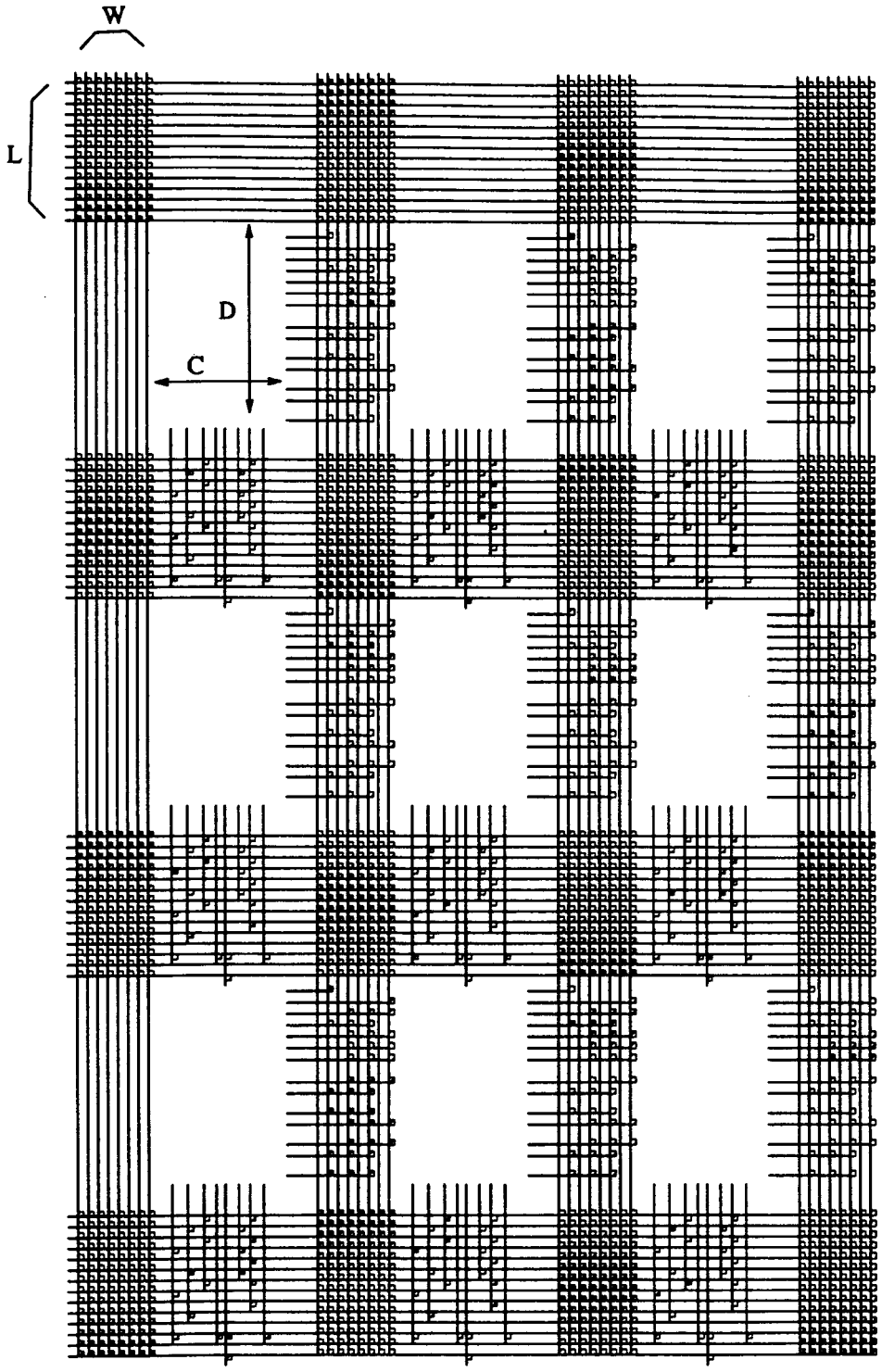


Figure 3.15: A 3 x 3 single track bus system using crossbar network generated by Lplatform.

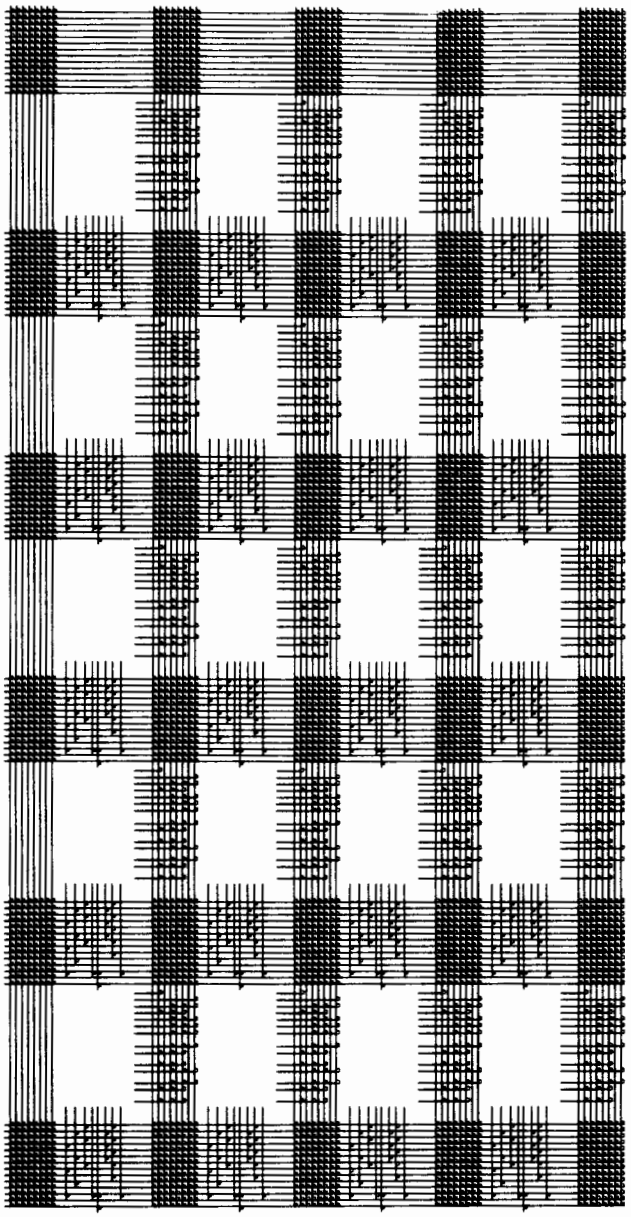


Figure 3.16: A 4 x 5 single track bus system using crossbar network generated by Lplatform.

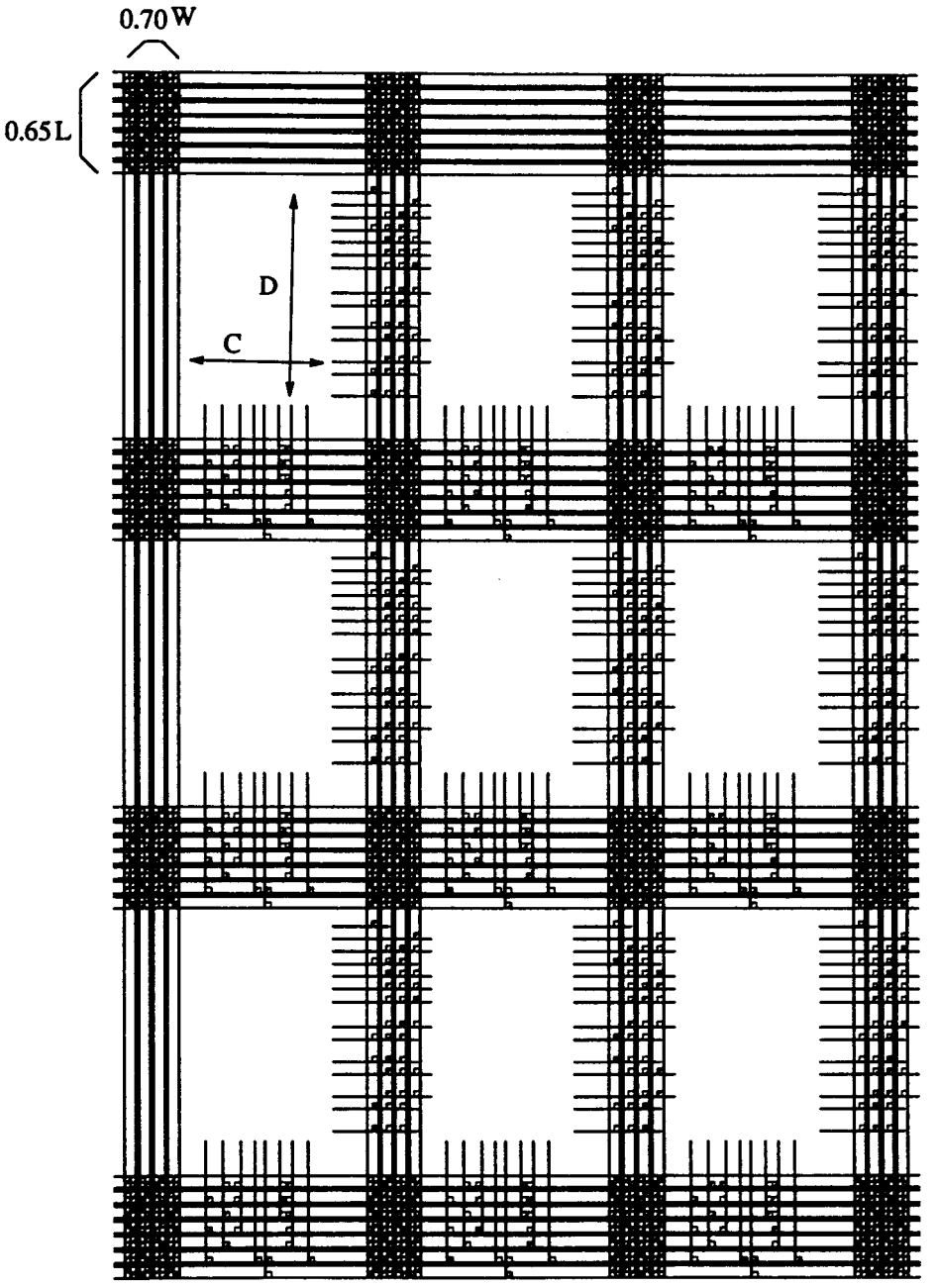


Figure 3.17: A 3 x 3 double track bus system using crossbar network generated by Lplatform.

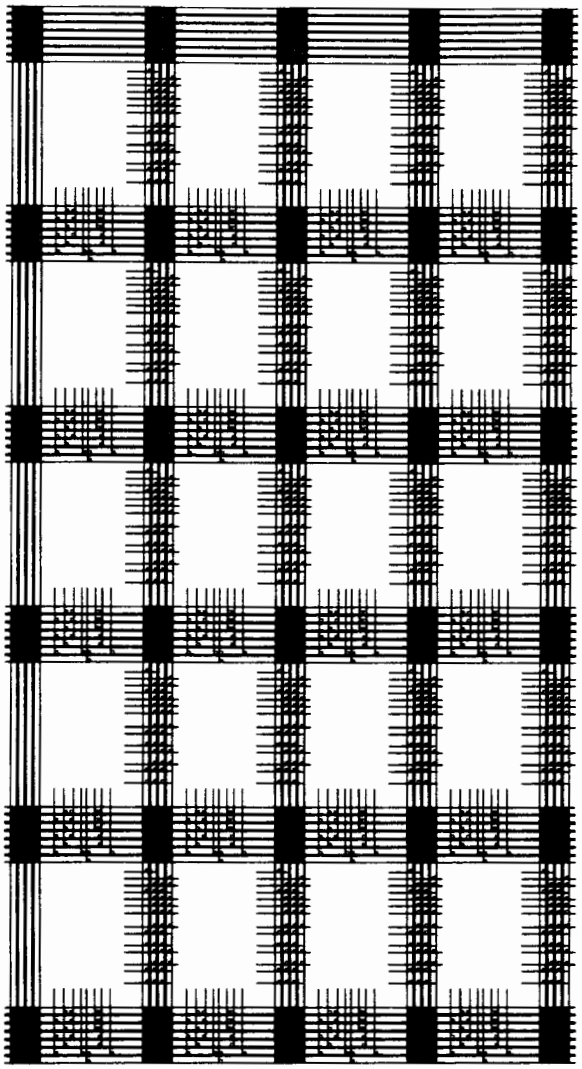


Figure 3.18: A 4 x 5 double track bus system using crossbar network generated by Lplatform.

shows a 4 x 5 single bus system, Figure 3.17 shows a 3 x 3 double-track system and Figure 3.18 shows a 4 x 5 double-track system. All four designs were created by simple changes in the data file. Similarly, as noted, the distribution of the stub links and tracks can be changed by altering the data. It is common in WSI to try several designs with different track combination to minimize area and optimize the other. Lplatform generates a summary file which shows the important factors such as bus area, line capacitance (calculated from the line length and link capacitance) etc., for optimization purposes.

3.5 Conclusion

We have reported the experimental results of the 3 μm CMOS Laser Links, of the design of a bus system using crossbar networks as interconnection networks, and, of using LLPRT as restructuring technique. Small area and high reliability of these laser links make them well suited to the crossbar network. In the next chapter, we develop another interconnection network, an electronic switch omega network, for the WSI bus system.

CHAPTER 4

The Electronic Switch Omega Network

As noted in Section 2.3 constructing a bus system using an electronic switch omega network is another way to achieve the defect-avoidance and the fault-tolerance goals in a WSI system. In this chapter, we present the research on an electronic switch version of the omega network, and a bus system using an omega network. First we describe the structure of a defect-avoidance bus using omega switch. Next the fundamental unit of the omega network, the full switch was designed. Then the full switches were combined into the transfer block of 8 x 8 omega network. The transfer block is used to transfer data in an omega network, which was simulated using HSPICE, fabricated in CMC CMOS 3 process and measured. The control block of the omega network which is used to control the data transfer direction was designed and is now in process. The purpose of these designs is to obtain parameters such as area, power consumption, delay times, etc. for comparison to the laser link network.

4.1 The Structure of a Bus System Using Omega Network

There are more theoretical discussion reports than implementation reports about using electronic switch networks as WSI interconnection networks [30,31,32]. There are probably two reasons. First, the structure of the crossbar network is not suitable to be designed using electronic switch technique. Designing an electronic switch crossbar network is not difficult, however the huge area it occupies prevents from its application in a real WSI system. By comparison, the single path multistage networks using electronic switches are preferable because they have a small number of switches compared with the crossbar network. The multistage network was originally used in the communication of parallel processors. The combination of electronic switches and WSI, such as using single path multistage interconnection networks (for example, an omega network) in a WSI bus system has only recently been discussed [16]. As noted in Section 1.2.2, few other attempts to actually build single path multistage interconnection networks have been undertaken in a WSI bus system.

An 8 x 8 crossbar network and an 8 x 8 omega network provide the same flexibility for signal transfer, i.e. each input port can be connected to any output port. An 8 x 8 crossbar network needs 64 switches while an 8 x 8 omega network needs only 12 switches. Thus the number of switches is reduced 5 times. At the very beginning in this research field, we choose the omega network as the interconnection network in a WSI bus system.

The bus-system structure using electronic switch omega networks is shown in Figure 4.1. The bus system is formed with omega networks in a string form, function

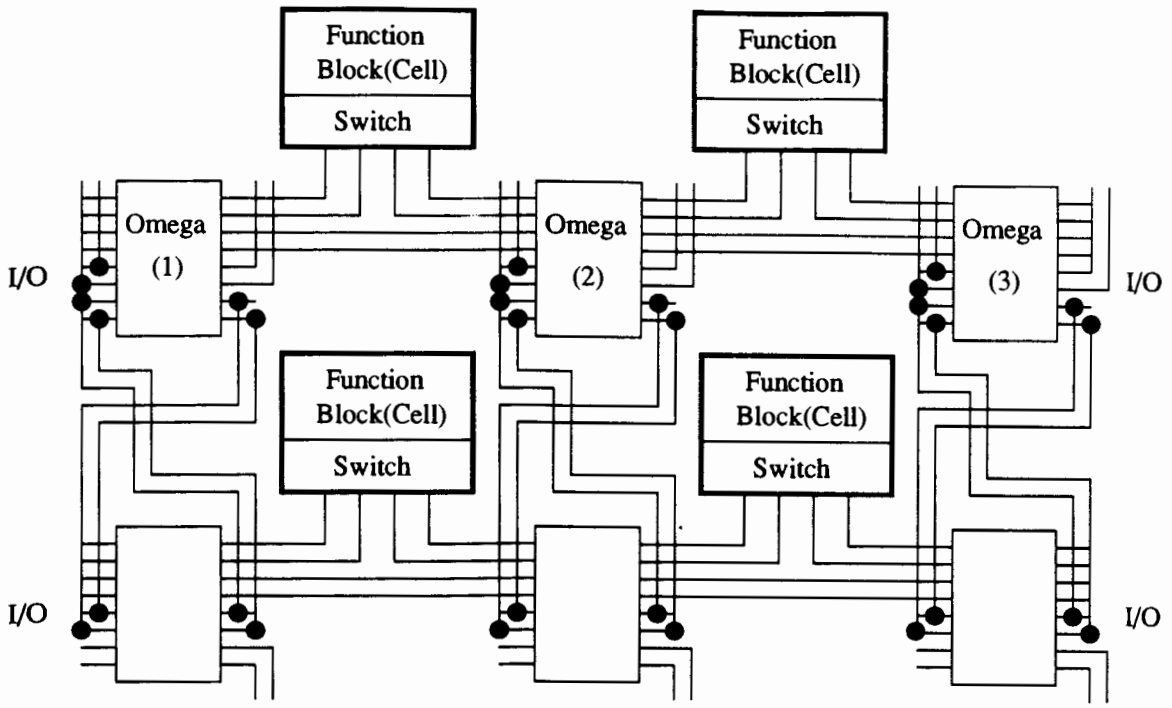


Figure 4.1: A WSI bus system using omega networks

blocks are put between every two omega networks. For example, a function block is put between omega 1 and omega 2, the signals of the function block are connected to the bus lines between omega 1 and omega 2. There are extra lines between these two omega networks for other data transfer. The exchanges of signals are performed with the shuffle property of each omega network at its nodes. A string of omega networks forms a row, with many rows on a wafer. For redundancy purposes each row contains several spare lines, and connection between the rows to enable vertical running signals. These lines are also used to verify the bus system itself. How many lines are actually needed to form the connection between different rows remains to be studied. Input/output signals are carried through omega networks to connect to I/O pins.

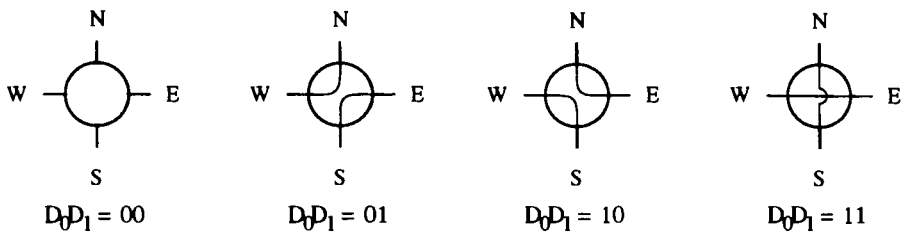


Figure 4.2: The states of the full switch.

The bus lines are connected to a function block through a switch circuit which can be a row of full switches. If the cell works, the omega network will be programmed to integrate this cell into the WSI system. If the cell fails, the switch circuit simply passes the signals to the next omega network and then to the redundant cells. The bus system should be tested before the function blocks are tested.

4.2 The Design of the Full Switch

A full switch is the basic structure in all kinds of communication interconnection networks. A diagram of a full switch is shown in Figure 4.2. This switch can be programmed in four states according to the values of two steering variables D_0 and D_1. For $D_0D_1 = 00$, all four directions are disconnected; for $D_0D_1 = 01$, North and West, South and East are connected; for $D_0D_1 = 10$, West and South, North and East are connected; for $D_0D_1 = 11$, North and South, West and East are connected. There are also several other forms of connection. For example, West can be isolated, while the three other directions are connected together, or all four directions can be connected together, etc..

A full switch can also be enlarged to a n-dimension interconnection network [33].

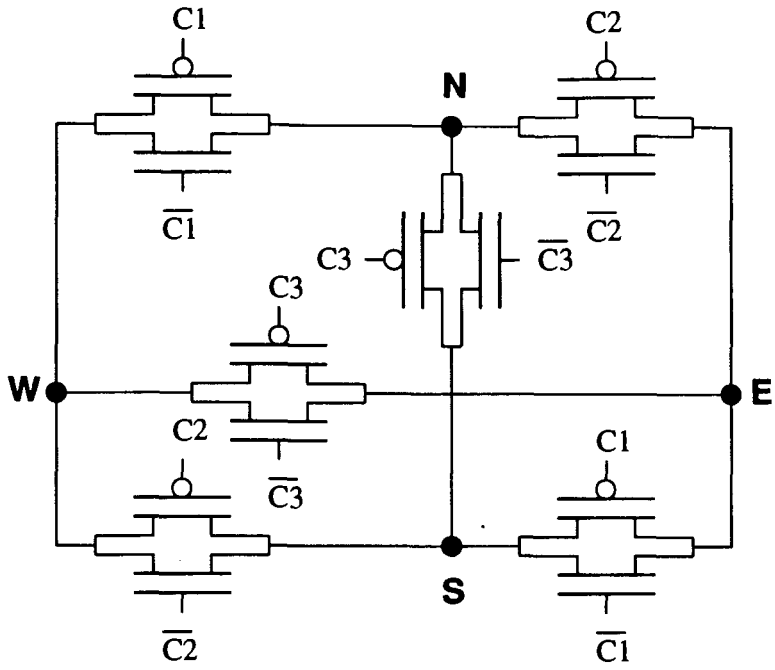


Figure 4.3: The circuit of 6-transmission gates full switch.

A n-dimension interconnection network is defined as having 2 x n terminals, the possibility for one-to-one connection between all pairs of terminals, and no distinction made between inputs and outputs. A full switch is more general in its switching capabilities than a crossbar switch. Gecsei made a theoretical analysis, introduced mathematical formulae and designed a diagram with 8 terminals [33].

The full switch circuit consists of transmission gates which have the bidirectional transmission capability. The circuit diagram of the full switch is shown in Figure 4.3. There are 6 transmission gates which are put between four terminals. Every terminal is connected to three other terminals through a transmission gate. The individual transmission gates are set to either ON (closed) or OFF (open) states to create one of the four configuration states shown in Figure 4.2. The change of states are controlled by three control signals which come from the control block (The control

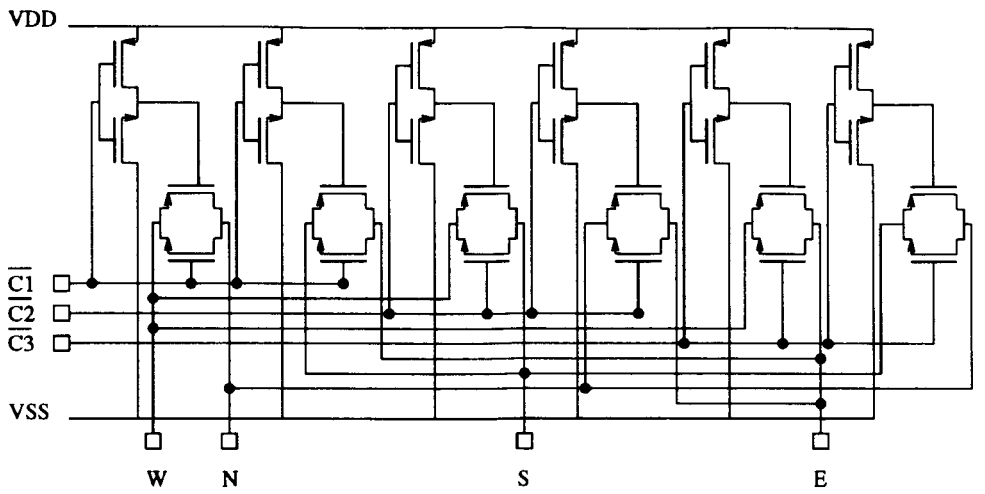


Figure 4.4: The circuit of the full switch in gate level.

block will be discussed later in another section). Figure 4.4 shows the circuit diagram of the full switch: W,N,S,E are four terminals; C1,C2 and C3 are 3 control signals from a control block; VDD is connected to a voltage source; VSS is connected to ground. A test circuit has been designed and tested using the CMC CMOS 3-micron process. Our implementation is shown in Figure 4.5. Design parameters and electronic characteristic will be discussed together with the transfer block of the omega network in Section 4.3. Figure 4.6 shows the photomicrograph of the full switch.

Figure 4.4 above is a possible simple design which requires only 1.5 transmission gates per port of the switch. In addition to the simplicity of the transmission gate structures, these switches are intrinsically bidirectional.

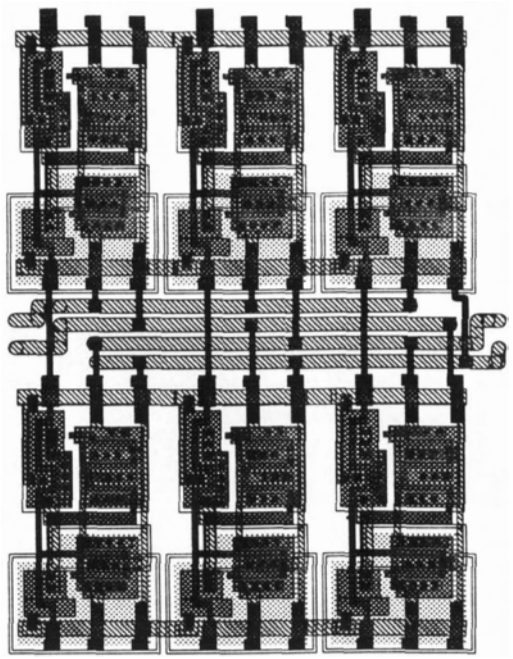


Figure 4.5: Layout of the full switch in 3-micron CMOS (269x349 $microns^2$).

4.3 Simulation of Omega Network Transfer Block

Before being designed, the transfer block of the omega network was first simulated using HSPICE for the CMC CMOS 3-micron process before being designed. The simulation includes time delay, current and power versus time. The HSPICE simulation input file is in Appendix A. The simulation results are as follows.

Time delay is very important in the switch because it influences the speed of the WSI circuit. Figure 4.7 shows the model of a cell output which drives a line having the resistance $R1 = 10\Omega$ and the capacitance $C1 = 1pF$ (representing a typical 1 cm length of Metal 1 line between cells) which leads into the 8 x 8 omega network. The capacitances and resistances of polysilicon, metal lines and contacts between full switches are estimated from their length as $R2 = 320\Omega$ and $C2 = 0.05pF$. For a

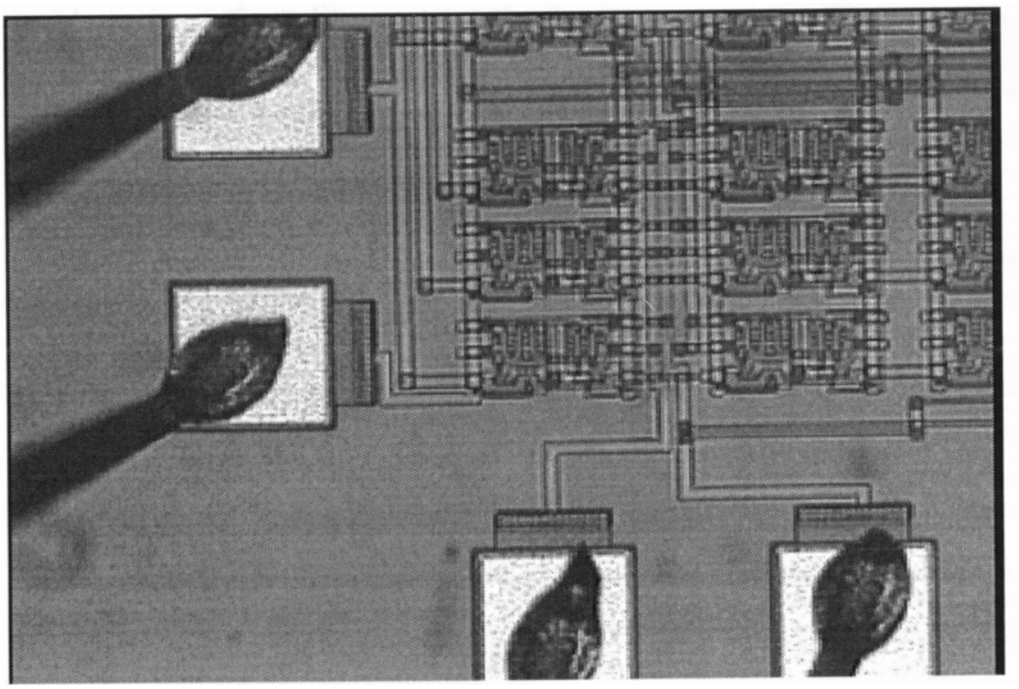


Figure 4.6: Photomicrography of the full switch in 3-micron CMOS

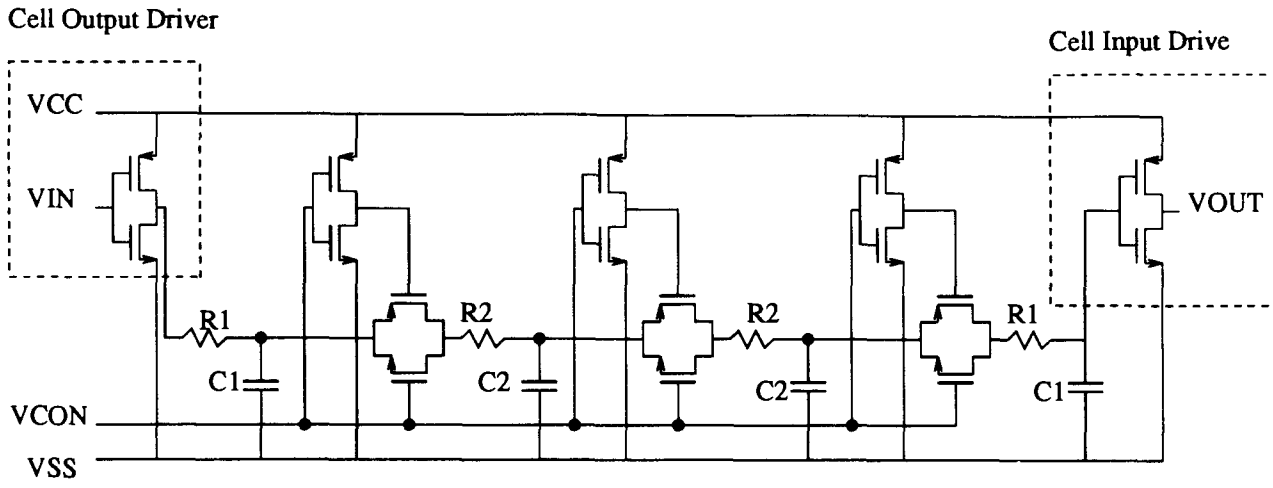


Figure 4.7: The equivalent circuit of 8 x 8 omega transfer block for simulation. R represents the line resistance and the C represent the line Capacitance [solid: input; dashed: output.].

signal to pass an 8 x 8 omega network, it needs to pass three stages transmission gates. Figure 4.8 shows the time delay simulation results. The output of the omega network then drives the lines which have the same length to the input of the next cell. With the 3-micron process, rise time is about 11 ns. For each gate, the delay time is about 3.67 ns. The fall time is about 8 ns. For each gate, fall time is about 2.67 ns.

As seen in the model of Figure 4.7, when switching occurs the input driver must supply current to charge the capacitors of the bus line while the output voltage rises. Figure 4.9 shows the power dissipation versus time. The total power supplied by the driver is 1.52×10^{-10} W per cycle (The period of the cycle is 60 NS and covers the transit changes).

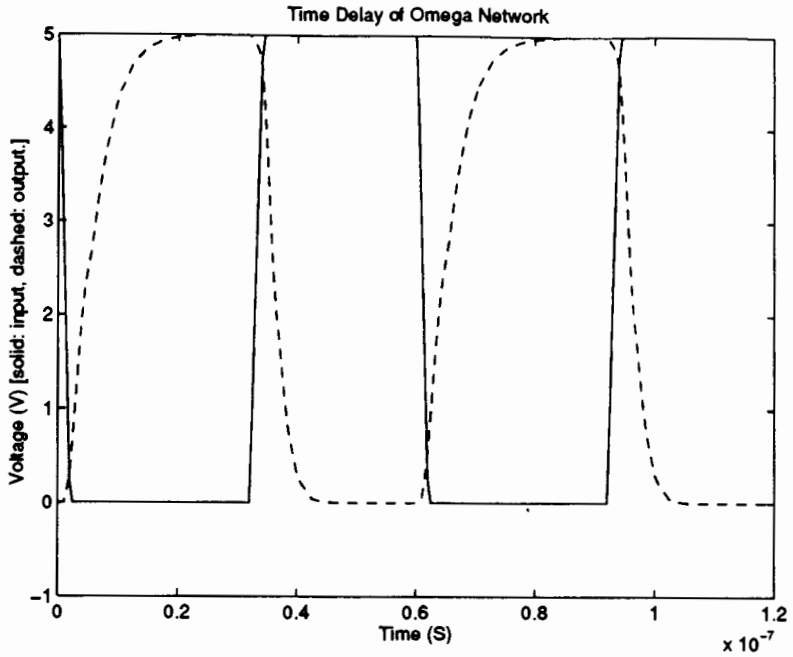


Figure 4.8: HSPICE time delay simulation results of 8 x 8 omega network [solid: input; dashed: output.].

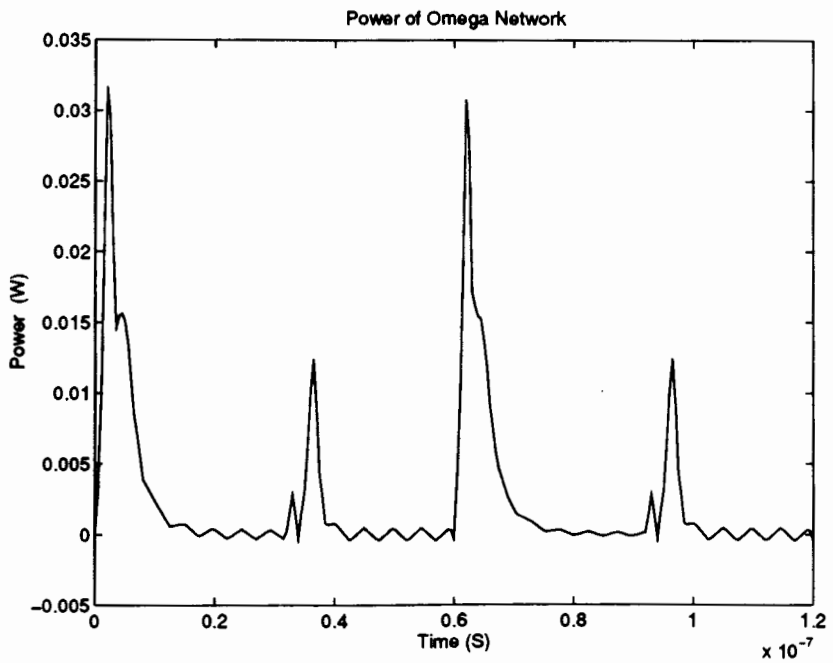


Figure 4.9: HSPICE Simulated Power of 8 x 8 Omega Network.

4.4 The Layout of the Transfer Block the Omega Network

The circuit of an omega network can be divided into two parts: the transfer block and the control block. The transfer block is used to transfer signals between function blocks and between function blocks and input/output pads, while the control block is used to used to control the direction of the transfer block. Figure 4.10 shows the transfer block diagram of the 8 x 8 omega network. There are 12 full switches in the 8 x 8 omega transfer block. The 12 switches are aligned in a 3 x 4 array, i.e. they are connected according to the omega network structure. The circuit was designed using KIC and the design was implemented using CMC CMOS 3-micron process. The layout is shown in Figure 4.11. Note that much of the space between full switches in this omega network is left for the running of the control signals. The area of the transfer block is 923 x 1478 $microns^2$. Figure 4.12 shows the circuit on the CMC chip.

4.5 Measurement of Omega Network Transfer Block

The fabricated transfer block of the omega network was measured using a test circuit, a 4 MHz high frequency function generator, a 300 MHz oscilloscope and a semiconductor parameter analyzer (SPA). The test circuit is shown in Figure 4.13(a). To measure the time delay of the transfer block, we used a function generator and an oscilloscope to observe the input and output waveform. There is a delay between the input and output signals. The rise time is the time delay between the rising edge of the input

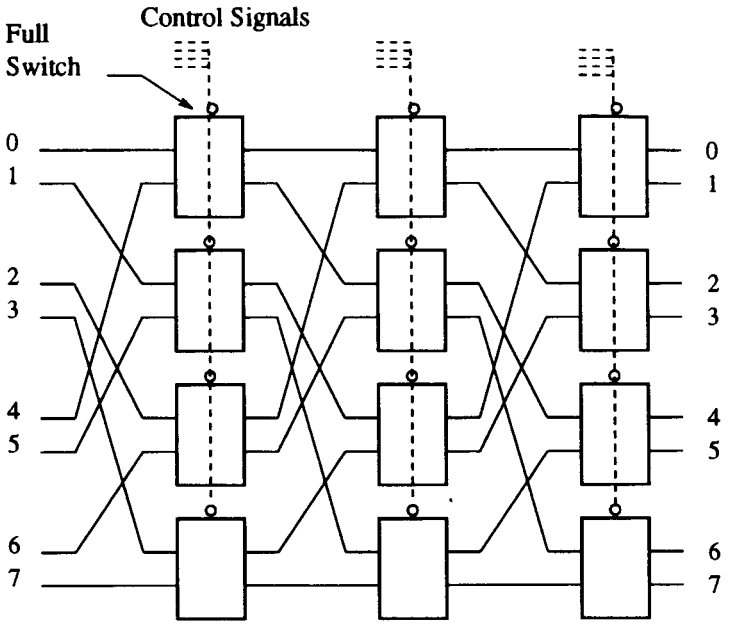


Figure 4.10: The transfer block of the 8 x 8 omega network.

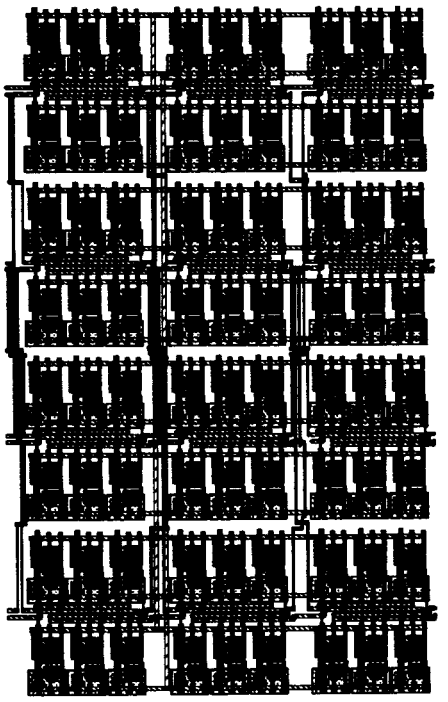


Figure 4.11: Layout of 8x8 transfer block in 3-micron CMOS (920x1477 $microns^2$).

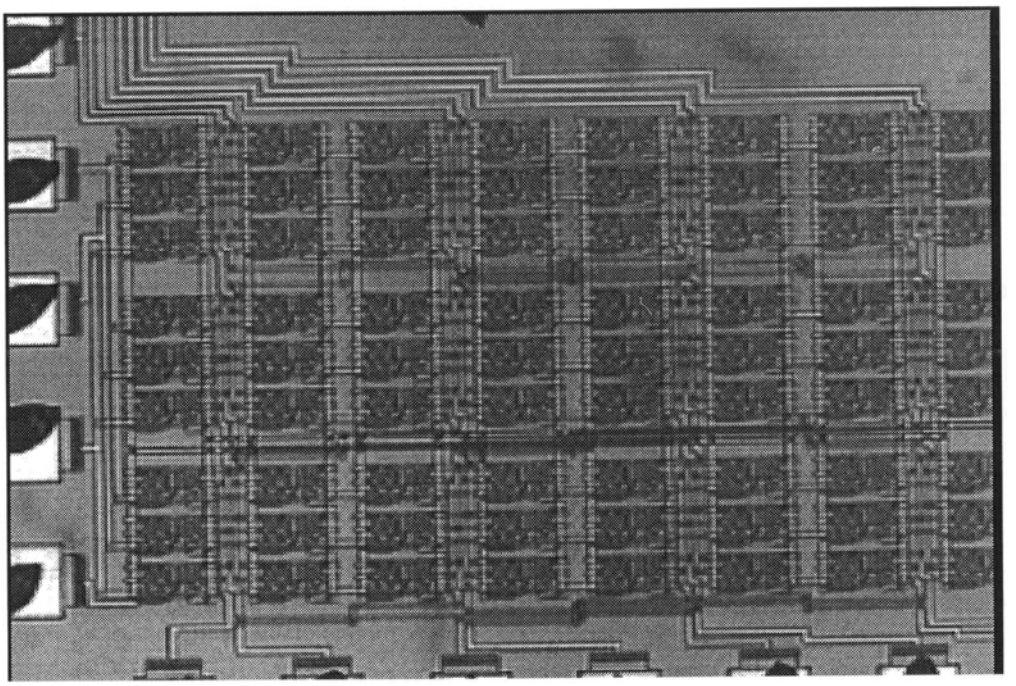


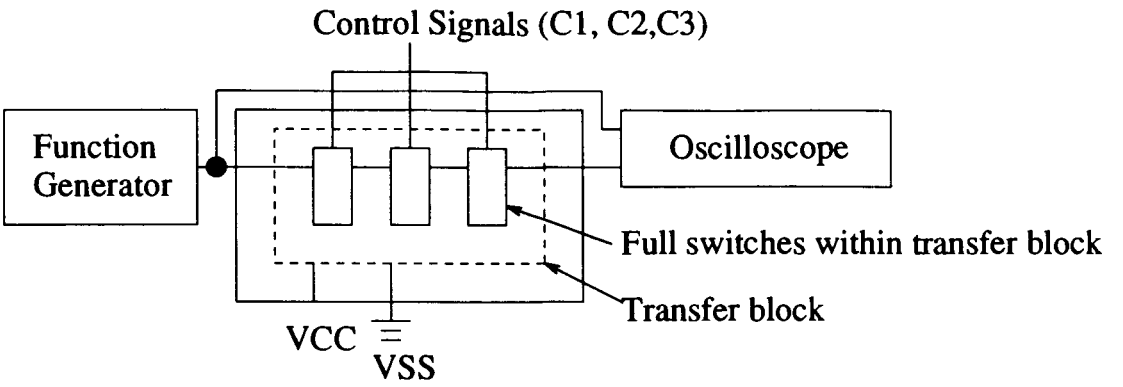
Figure 4.12: Photomicrography of the transfer block in 3-micron CMOS.

Table 4.1: Measurement of Transfer Block Time Delay

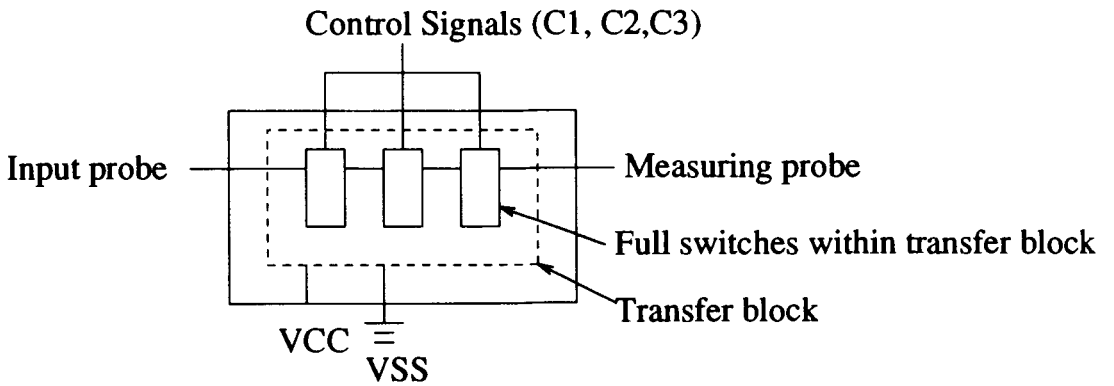
Times	Rise Time (ns)	Fall Time (ns)
1	19.2	12.6
2	19.8	12.9
3	19.7	12.4
average	19.6	12.6
σ	1.3%	1.4%
simulation	11	6

and output signals and the fall time is the time delay of falling edges between the two signals. The rise time was measured from the rising edge of the input signal to the 90% peak value of the output signal. The fall time was measured from the fall edge of the input signal to the 10% peak value of the output signal. The recorded results an shown in Table 4.1. The average rise time is 19.58 ns and the average fall time is 12.61 ns at room temperature. From the Table we can see that the measured times are longer than the simulated results in Figure 4.8, but with a similar difference between the rise time and fall times. And the differences between simulation and measurement are small.

The transfer block was also measured using the HP 4145A Semiconductor Parameter Analyzer and the test circuit is shown in Figure 4.13(b). The I versus V measurements from 0 to 5 V showed a linear plot with a slope yielding an equivalent resistance of about 837 Ω, as shown in Table 4.2. This is considerably higher than the 76 Ω of the laser link switch.



a) Test circuit to measure the time delay.



b) Test circuit to measure the I-V characteristics using HP 4145A SPA.

Figure 4.13: Test circuit to measure the transfer block of the omega network, (a) time delay, (b) I-V Characteristic.

Table 4.2: Measurement of Resistance of the Transfer Block

Times	Resistance(Ω)
1	840.60
2	836.70
3	833.80
average	837.03
σ	0.2%

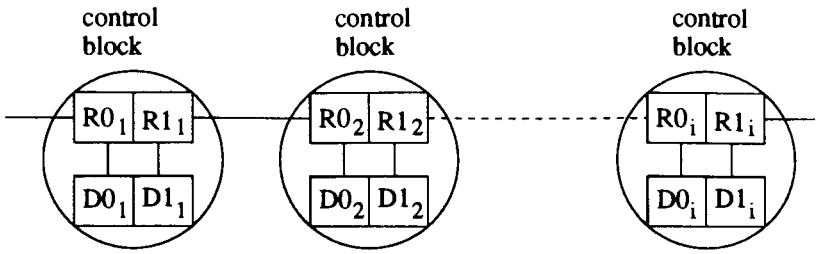


Figure 4.14: Transfer of control signals.

4.6 Control Block

The control block of the omega network is used to control the directions of transfers. The control signals are transferred independently from data transfer. Figure 4.14 shows a string of switching nodes which length is i. The control signals is a sequence of data $(D0_1, D1_1)$, $(D0_2, D1_2)$, ..., $(D0_i, D1_i)$ which is transferred using the shift registers $(R0_1, R1_1)$, $(R0_2, R1_2)$, ..., $(R0_i, R1_i)$. Then at the end, the data is latched into a series of flip-flops $(D0_1, D1_1)$, $(D0_2, D1_2)$, ..., $(D0_i, D1_i)$. This data is decoded by a decoder to generate a serial of control signals for the transfer blocks.

Figure 4.15 shows the structure of the control block. The control block consists of four parts: select block, shift register, 2-bit steering flip-flop, and decoder. The control logic of the omega network (which is also the control logic of the full switch) is shown in Table 4.3.

The select block is used to choose the switch node where related data need to be latched in the registers. It is also a shifting circuit which controls the flip-flop for data to be dumped from the shift register. However, three other parts fulfill the control tasks. Since the select block was more related to the control of the system that was deciding which direction to transfer the signal, the select block was ignored in our

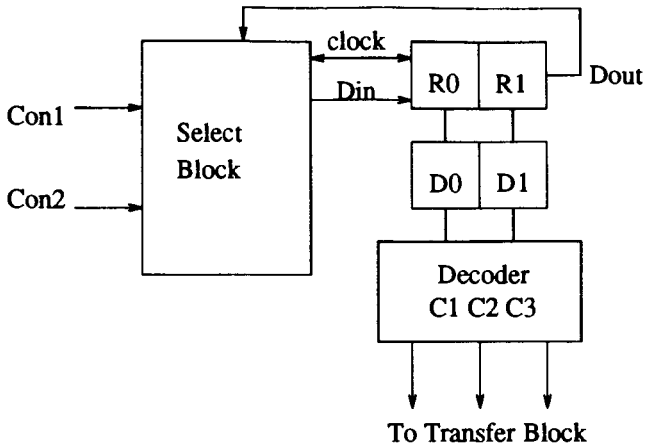


Figure 4.15: Control block of the omega network.

Table 4.3: Control Logic of the full switch

C1	C2	C3	Logic State
1	1	1	all off
0	1	1	W-N S-E on
1	0	1	W-S N-E on
1	1	0	W-E N-S on

first version of control block design.

The shift register is used to receive control data from the input and to output this data to the flip-flop. It was designed as a left/right, serial/parallel shift register because it also needed to shift a string of data for other switches [34]. Figure 4.16 shows the two-bit shift register block diagram. The operating modes of the register are refresh loop mode, parallel load mode, shift right mode, shift left mode and parallel output mode. In the register clock 1 allows loading, shifting, and refreshing to occur while $\overline{clock1}$ isolates the two inverters so that the cells may be loaded. In our design, there can be two input modes, a parallel load mode and a left shift mode. In parallel mode, the inputs 'dp' (see Figure 4.16 and Figure 4.19, 'dp' will be either signal 'D0'

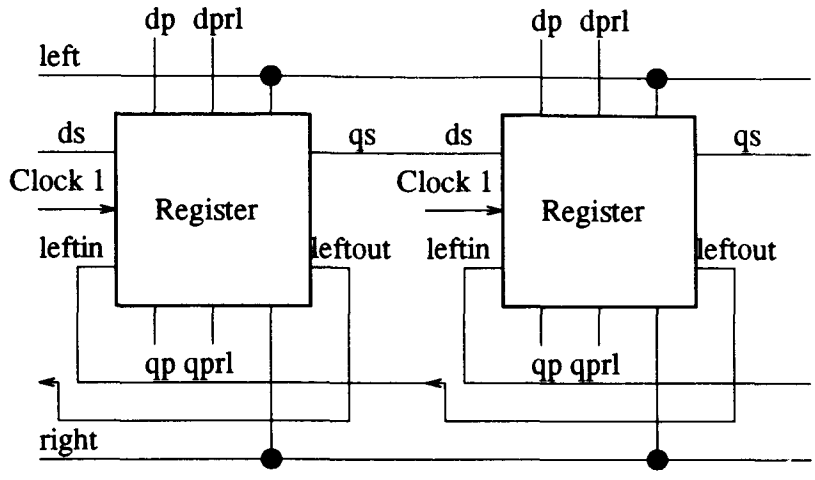


Figure 4.16: Two bits shift register.

or 'D1' in Figure 4.19) and control 'dprl' are used to load the registers in parallel. Asserting 'dprl' when clock 1 is at logic level 1 will cause the input of the first inverter to assume the state of 'dp'. At this time $\overline{clock1}$ is at logic level 0 and the inverter are isolated. Subsequently $\overline{clock1}$ is at logic level 1 and the second inverter output assumes the state of 'dp' which is stored dynamically at the first inverter input. In shift mode, asserting the left line when clock 1 is at level 1 effectively loads the previous register with 'qs' via the feedback line 'leftout'. The register cell to the right of the current one has its 'leftout' connected through a transmission gate to 'leftin' of the present cell. Hence the cell is loaded in the same manner as with a parallel load but the data input comes from the adjoining cell to the right.

The 2-bit flip-flop is used to store the control data and output them to the decoder. We use master-slave D-type flip-flop to realize the function shown in Figure 4.17 [34]. Since the circuit is made up of cascaded latches, the latency time depends on the frequency of the clock pulse.

The decoder is used to decode the two bits input data and outputs the three

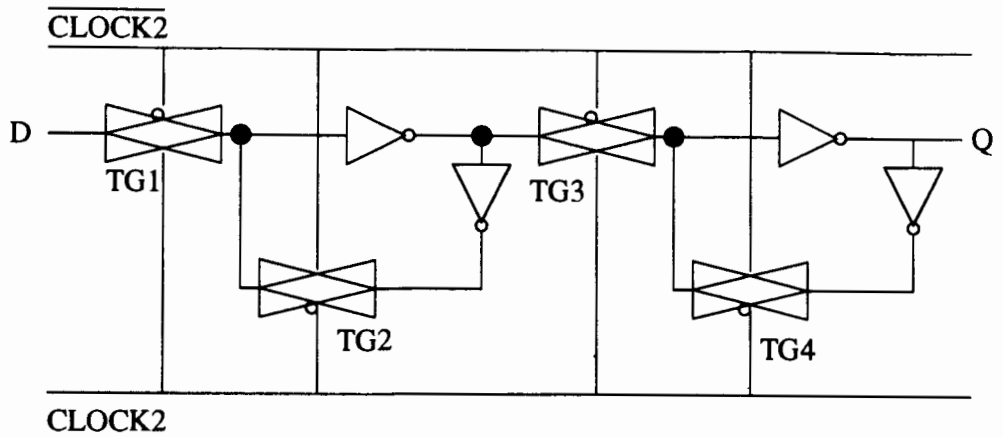


Figure 4.17: The flip-flop.

Table 4.4: Decoder Logic

D0	D1	C1	C2	C3
0	0	0	0	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

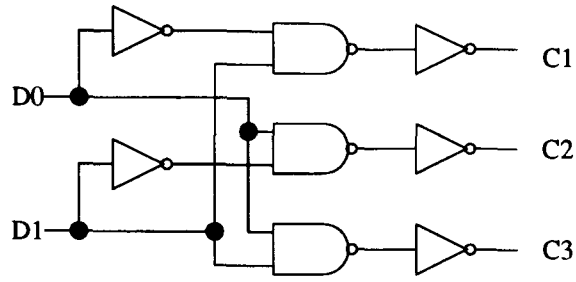


Figure 4.18: Control signal decoder.

control signals to the transfer block. Four states correspond to the four directions. I designed the decoder according to the logic shown in Table 4.4, and the circuit is shown in Figure 4.18.

The two control signals are finally decoded into three control signals which can be used to control one full switch or several full switches. This depends on circuit requirements. The full circuit in gate level is shown in Figure 4.19 and the layout is shown in Figure 4.20. The circuit has been fabricated in the CMC 3-micron CMOS process and Figure 4.21 shows a photomicrograph of the real circuit which is 434 x 1312 $microns^2$ in area. The aim at this point was to obtain area and other design parameters for the work in Chapter 5.

4.7 Conclusion

The implementation of the 8 x 8 omega network was divided into two parts: the transfer block and the control block. The transfer block of the omega network was successfully implemented in the CMC 3-micron CMOS process. Several important parameters, area, time delays and resistances, were measured. The measured switch parameters show the same trend as the simulations. Second, we designed the first

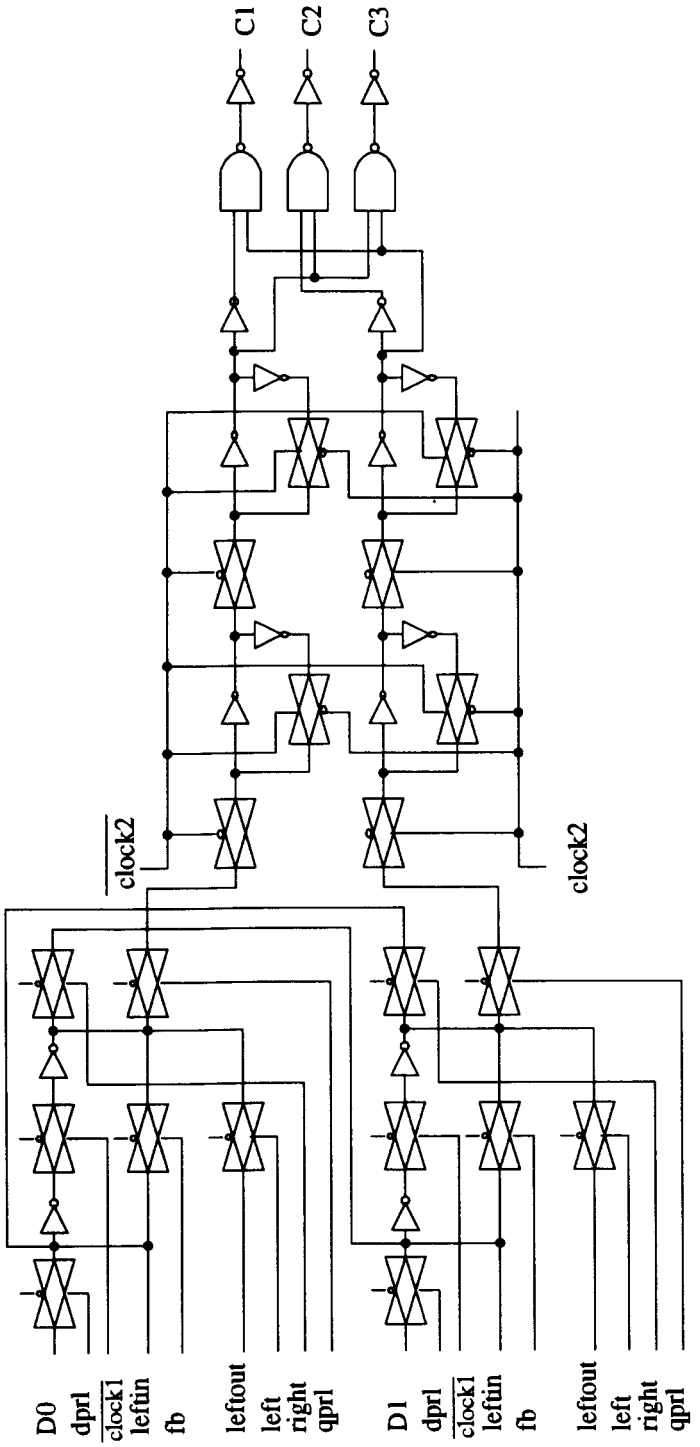


Figure 4.19: A gate-level circuit diagram of the control block.

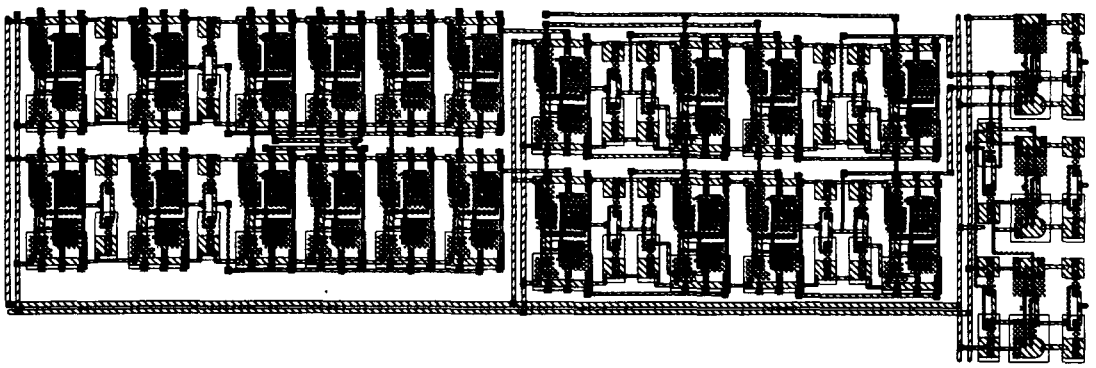


Figure 4.20: Layout of control block in 3-micron CMOS (434x1312 $microns^2$).

version of the control block of omega network and added it to the transfer block. The next version of control block will be designed on this version because this version of control block can fulfill the basic control task. The measurement and further development of the control block will be done in future. The next chapter will take the parameters obtained in this section and use them to compare the laser link crossbar and omega networks.

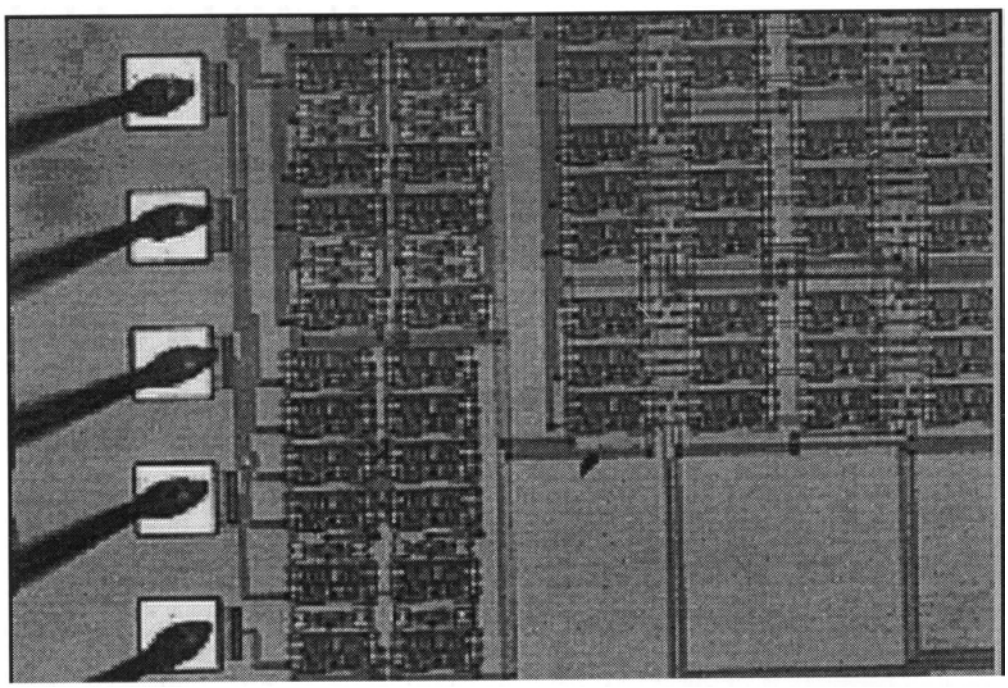


Figure 4.21: Photomicrography of the control block in 3-micron CMOS.

CHAPTER 5

Comparison and Tradeoff of Two Networks

5.1 Comparison of these two networks

We have discussed two different kinds of networks, the laser-link physical restructuring crossbar network and the electronic switch omega network, which can both work very well as interconnection networks in a wafer scale integration system. There are many differences between them in terms of fabrication, electrical characteristics, application areas, etc. [35]. This chapter addresses some of the tradeoffs in choosing which scheme to use. In addition by comparing these two techniques, we proposed a different kind of network which maximizes the advantages and minimizes the disadvantages of these two kinds of networks. This work focuses on several key factors that influence the implementation of these networks: area, power and time delay.

5.2 Area

The area of an interconnection network is a significant factor in WSI design. Due to the area taken by the switching system, a large fraction of a WSI system may be taken up by the signal routing between the active circuit elements. The area taken by the switches and the corresponding control circuitry becomes a significant factor which influences the parameters such as the useful fraction of processing circuitry in the system, number of spare processing blocks available, and the signal propagation delays. Basically the switch area becomes an overhead which limits the complexity of the systems which can be built.

Physical switches typically have a smaller area overhead than programmable electronic switches because of the smaller size of the actual switching point. For physical switching, the area of a laser link is very small compared to the full switch which is the basic cell of electronic switch networks. In our design, the area of the laser link 8 x 8 crossbar network is 1.26 x 10^5 $microns^2$, while the transfer block of the 8 x 8 omega network is 1.36 x 10^6 $microns^2$ (see Figure 5.1). The reason for comparing two 8 x 8 networks is that they have the same logic function. Thus the transfer block section of the omega network alone, without the control block circuitry, requires 9.5 times more area than the laser link crossbar. A single control block would take 5.69 x 10^5 $microns^2$ (see Fig 4.20). The exact size of the full control section would depend on the level of switching needed. For maximum flexibility each full switch would require its own control block, using 12 such blocks in a 3 x 4 array needs 6.83 x 10^6 $microns^2$. The combined 8 x 8 omega switch plus control sections would then take 57 times more area than is required for the 8 x 8 crossbar network. While these omega switch/control cells are not fully optimized for minimum area, clearly the area saving

Table 5.1: Area of different circuits

Circuit	Area ($microns^2$)	Relative area
8x8 Crossbar Network	1.26 x 10^5	1.00
Full Switch	0.94 x 10^5	0.74
8x8 Omega Transfer Block	13.59 x 10^5	10.79
Omega Control Block	5.69 x 10^5	4.52
Transfer + 12 Control Blocks	81.87 x 10^5	64.98

will be of this order in many designs. The area comparison is shown in Table 5.1. Figure 5.1 shows the 8 x 8 crossbar network and the 8 x 8 omega network transfer block in one scale. Figure 5.2 shows a single control block and a single full switch at a scale. Furthermore, the omega network's larger area involves more complex devices. Thus the yield for the omega switch should be significantly lower than that of the laser link crossbar. For example just on the basis of an area alone, if the yield would vary as Y^a, where Y is the yield per unit area and a is the number of units. Then if $Y = 0.995$ for laser links, it would be 0.954 for the one transfer block of omega plus one control block and 0.75 for the transfer block plus 12 control control block. It should be noted that the omega switch area could be optimized and thus the area reduced to improve the ratios. A full design of the omega transfer and control block would shrink area by merging tubs and minimizing spacing, but would consume some area in the more complex control signal routing. However the transfer/control area combination would still be much larger than the laser link crossbar.

The area consumption comparison shows that the laser-link crossbar network has great advantage compared to the electronic switch omega network. The interconnection bus system using crossbar network is different from the bus system using the electronic switch omega network. The number of crossbar networks which the former

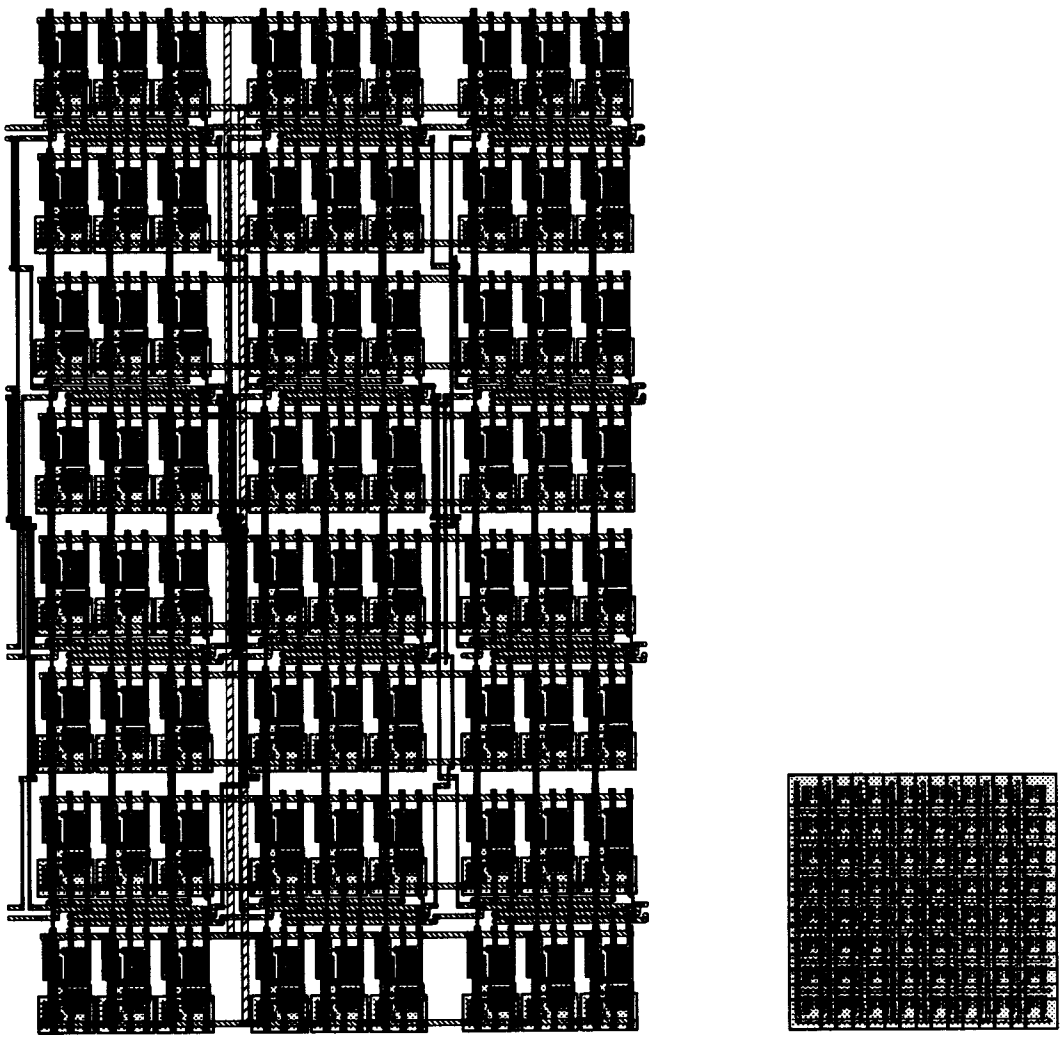


Figure 5.1: 8 x 8 omega transfer block (1534 x 2461 $microns^2$) and 8 x 8 crossbar network (684 x 616 $microns^2$) in the same scale in 3 micron CMOS

.

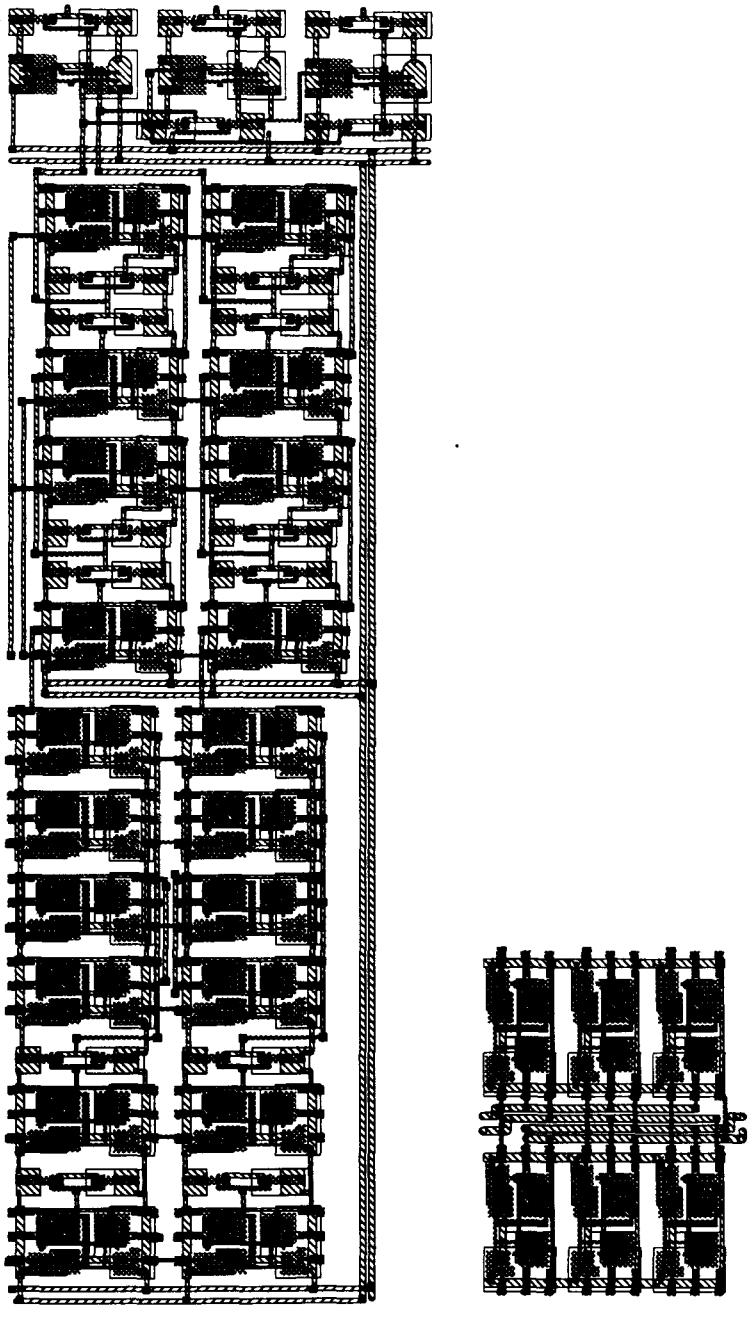


Figure 5.2: An omega control block (724 x 2186 $microns^2$) and a full switch (448 x 581 $microns^2$) in the same scale in 3 micron CMOS.

bus system needs is 1.5 times that of the omega network numbers which the later bus system needs. Even so, the area consumption of the former bus system is much less than that of the latter.

5.3 Laser Process

Compared to the electronic switch omega network, the physically restructuring cross-bar network needs an extra process, laser linking and cutting for defect avoidance, which requires specialized equipment and processing time.

The major time cost result from the interval taken by moving the focused laser spot to the correction site which is the small time (few microseconds) taken to perform the operation. An additional factor is the time taken to move the wafer elevation (z axis) to maintain laser focus at the linking points. In the lab of Simon Fraser University, the laser link/cut system operates more slowly than a commercial system in order to get a good random access of the link points. For a random collection of physical locations within a few hundred $microns^2$ the limits really correspond to the time the table system/computer takes to do short accelerations and decelerations. This typically sets the limits at about 10 connections per second. For very far distances (multiple centimeters) transit times can be about 1 to 2 seconds. However operations can be significantly speeded up if care is taken to physically align all the laser links. For example, if a number of connections or cuts are to be made in a row, with some laser tables, the laser can make connections while the table is moving, resulting in about 4000 connections per second. The initial time for the physical alignment, leveling and orientation of a wafer, which may take minutes, is important for very small systems,

but less so for large area/wafer scale systems.

While slower than active defect avoidance, laser restructuring is permanent. And a wafer scale system typically only requires about 1000 connections or cuts. By comparison active devices may require a restructure initialization of the wafer. Furthermore, most of the complexity and difficulty in active switching is in the detecting or bypassing of fabrication defects. This suggests the need for laser linking to remove the actual power and bus line defects just after fabrication. Active switches would route the signals to the working sections.

5.4 Influence on Speed

The time delay is the factor which influences the speed of networks. The networks having short time delay are faster than those having longer time delay. The HSPICE simulated time delays are used to compare the network speed. Four kinds of circuits are simulated all with 1 cm of line length ($R1 = 10\Omega$, $C1 = 1pF$) assumed between the drivers and the switches. They are direct connection between drivers, laser link crossbar network, electronic switch omega network and omega network with output buffer. Figure 5.3 is the equivalent circuit for laser link crossbar network and an direct connection between two drivers. The direct connection between drivers is simulated using a 100 μm metal line to replace the laser link switch, creating a similiar connection of the input to output drivers. The differences between the two circuits are R2 and C2. For the laser link crossbar network, $R2 = 76\Omega$ and $C2 = 30fF$; For the direct connection between drivers, $R2 = 0.001\Omega$ and $C2 = 0.01fF$. The equivalent circuit of the omega network is shown in the Figure 4.7. Figure 5.4 is the equivalent circuit

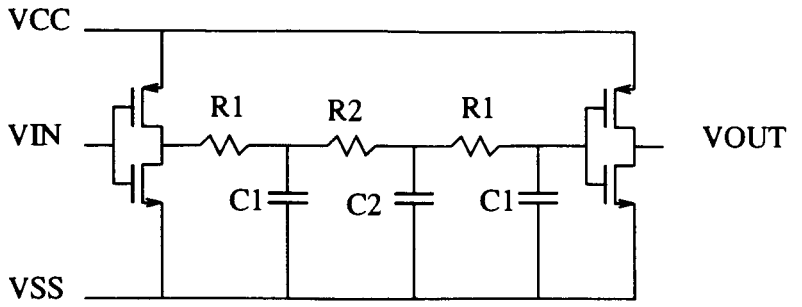


Figure 5.3: The equivalent circuit of laser link crossbar and direct connection between drivers.

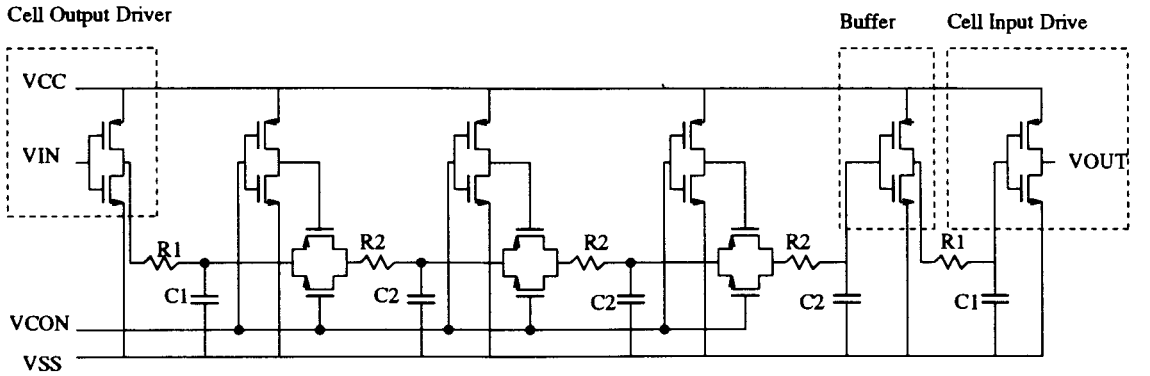


Figure 5.4: The equivalent circuit of omega network with output buffer.

of omega network with output buffer.

The HSPICE simulated time delay of the 100 μm is shown in Figure 5.5. The time delay of laser link crossbar network is shown in Figure 5.6. The time delay of electronic omega network with buffer is shown in Figure 5.7. The time delay of omega network has been disscussed in Section 4.3. Table 5.2 compares the different time delays of different networks. The cycle time delay (rise time + fall time) of laser link crossbar network is 4 ns, which is not much different from the 3 ns of a piece of 100 μm metal line. By comparison the time delay of the omega network is 17 ns which is almost 4 times that of laser link crossbar network. Thus the speed of the laser link crossbar network is faster than the electronic omega network.

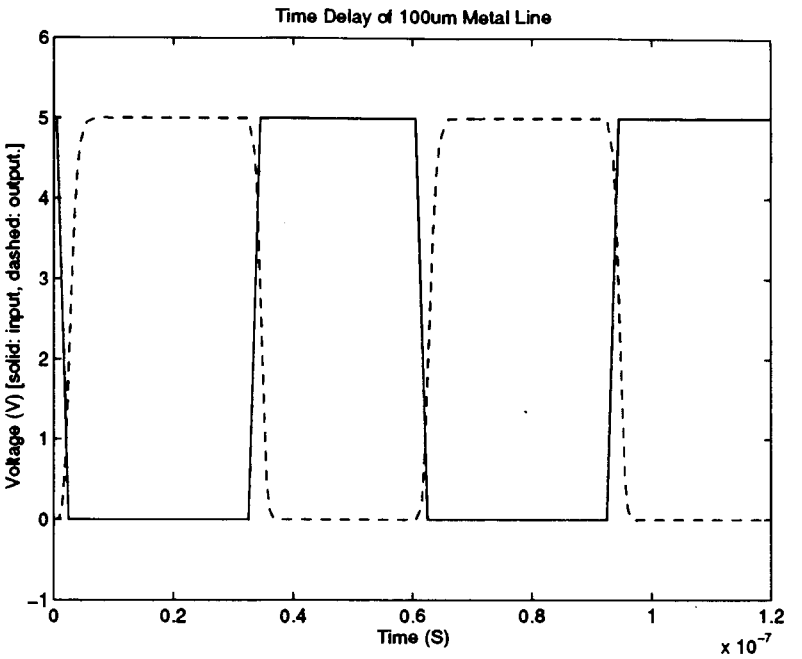


Figure 5.5: HSPICE simulated time delay of a piece of 100 μm metal line [solid: input; dashed: output.].

Table 5.2: The Time Delays of Four Kinds of Circuits

Circuits	Network Rise T(NS)	Network Fall T(NS)	Switch Rise T(NS)	Switch Fall T(NS)
Direct Connection	2	1		
Laser Link Crossbar	3	1	1	0
8 x 8 Omega Network	11	6	9	5
Omega with output Buffer	6	4	4	3

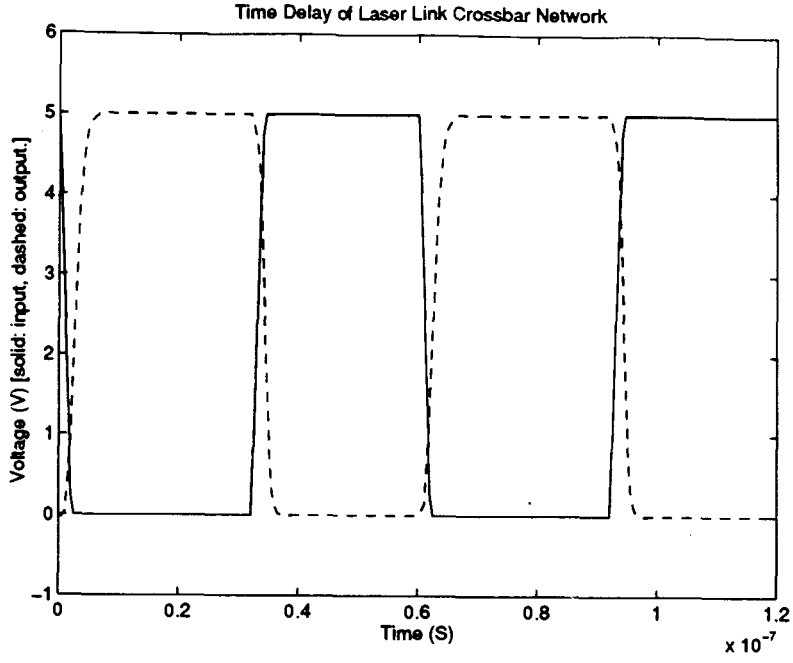


Figure 5.6: HSPICE simulated time delay of the laser link crossbar network [solid: input; dashed: output.].

5.5 Power Consumption

Calculating the power dissipation of CMOS systems is not easy because the majority of the power dissipated in a CMOS system is a function of signal switching rates which depend on the exact nature of the circuits [36]. Meanwhile the quiescent or static power dissipation of most CMOS devices is insignificant. System dynamic power dissipation, on the other hand, is difficult to calculate directly since dynamic power is a function of node frequency, capacitance, and signal voltage swing. Thus the operating conditions of each system signal are needed to calculate the system dynamic power.

A physical restructuring crossbar network does not need any dc bias itself and there

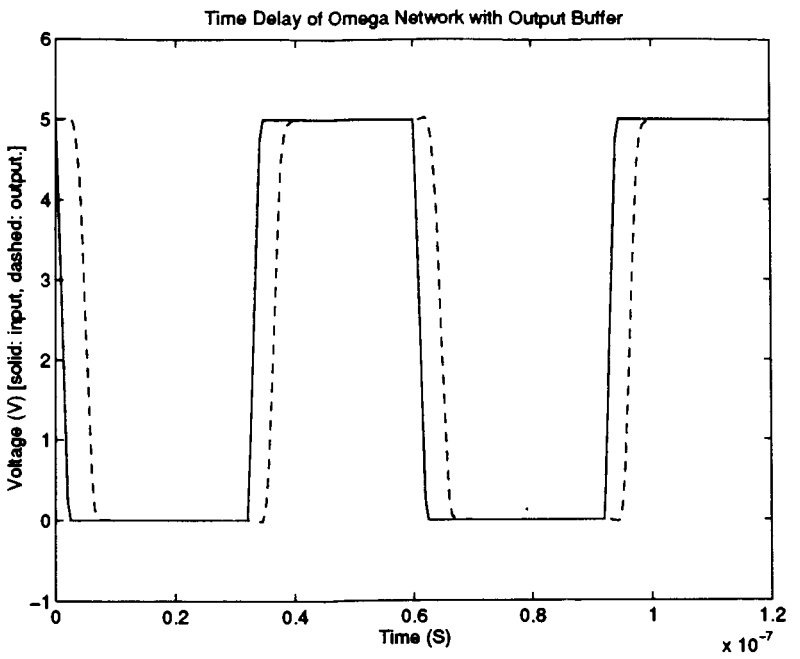


Figure 5.7: HSPICE simulated time delay of the electronic omega network with buffer [solid: input; dashed: output.].

is no dc power dissipation. An electronic switch omega network needs a 5 V voltage source for the transmission gate and control block. Considering the complementary character of the CMOS gate, only the leakage current contributes to the quiescent power dissipation. For an 8 x 8 omega network, there are 12 full switches, about 72 transmission gates, plus control blocks (the number can only be counted with a specific circuit).

The equation for calculating dynamic power P_d is:

$$P_d = (C_L + C_{pd})(\Delta V_s)^2 F \tag{5.1}$$

Where:

C_L = line and device load (both input and output) capacitance

C_{pd} = internal device capacitance

ΔV_s = signal swing ($\approx V_{cc}$ for CMOS devices with CMOS output levels)

F = node toggle frequency

Signal voltage swing is easily established and C_{pd}, which represents internal device capacitance, is provided on general data sheets, but line and load capacitance and the frequency of each node are usually difficult to estimate. For this reason we chose to estimate the power consumption from the HSPICE model's power supply draw. Since the HSPICE model includes all the capacitances, and the voltage swings, the model's power per cycle will correspond to the CV^2 terms of equation 5.1.

The introduction of the defect avoidance circuitry must introduce resistance and capacitance that inherently affect the power consumption of the system. The substantial difference in the area of the interconnect and the transistor/diffusion connections for the two systems will result in a difference in effective resistance and the capacitance to the substrate.

Table 5.3: The Power Comparison of Four Kinds of Circuits

Circuits	Average (W)	Cycle (W)
Direct Connection	1.57×10^{-3}	9.42×10^{-11}
Laser Link Crossbar	1.65×10^{-3}	9.88×10^{-11}
8 x 8 Omega Network	2.54×10^{-3}	1.52×10^{-10}
Omega network with output Buffer	2.73×10^{-3}	1.64×10^{-10}

To analyze the power of different networks, we use the same equivalent circuits as in last section for HSPICE simulation. Figure 5.8 shows the power of input driver of the switches replaced with the 100 μm metal line. Figure 5.9 shows the power of input driver of the laser link crossbar network. Figure 5.10 shows the power of input

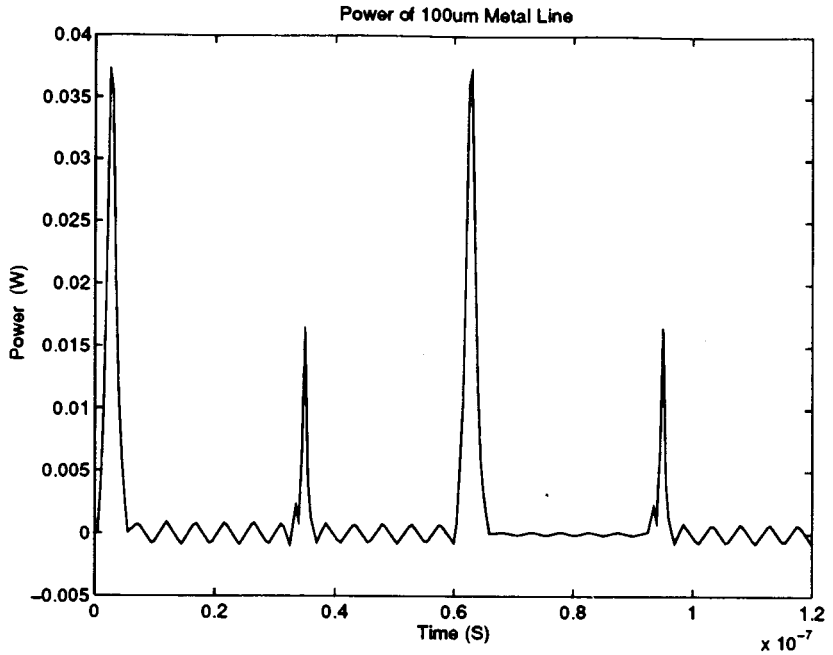


Figure 5.8: HSPICE simulated power of a piece of 100 μm metal line.

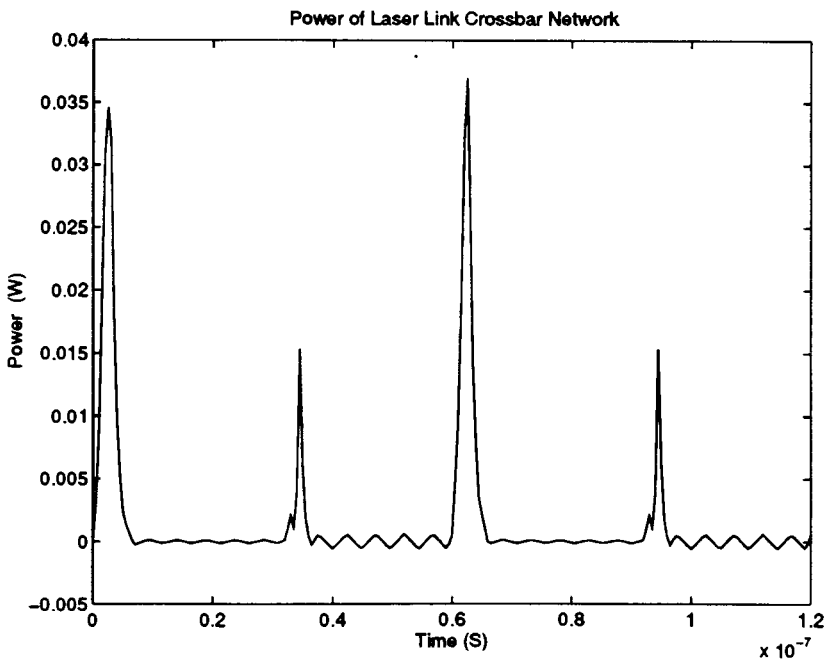


Figure 5.9: HSPICE simulated power of the laser link crossbar network.

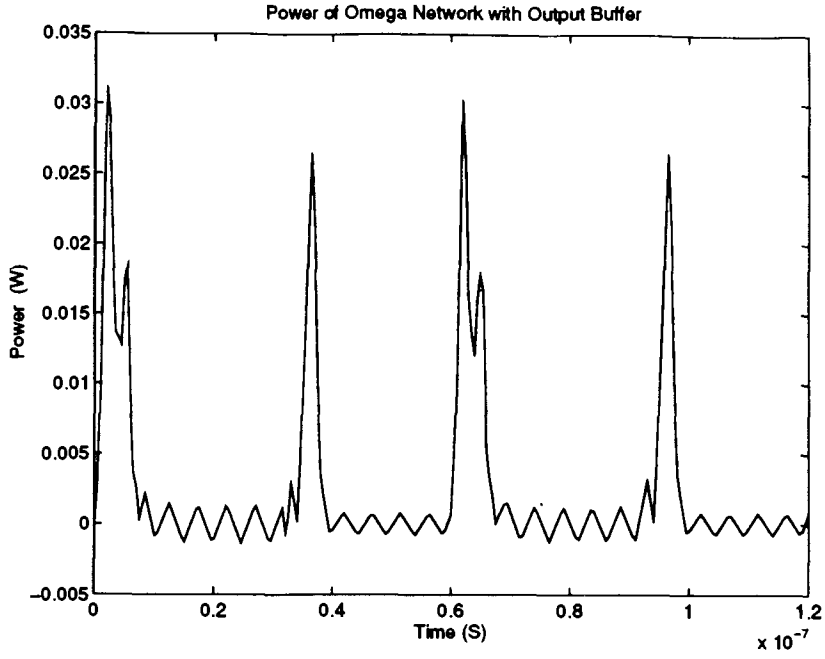


Figure 5.10: HSPICE simulated power of the electronic omega network with buffer.

driver of the omega network with buffer. The simulated power of omega network is in Section 4.3. The comparisons are listed in Table 5.3. The cycle power is the power consumed during a single cycle (the period is 60 ns in our simulation) and the average power is the total power divided by the period of 16.7 MHz. The period is long enough to cover the changes of the signals. The power of the laser link crossbar network is only 5% larger than that required by a direct connection. However the power of omega network is 62% larger than that of the direct connection. The omega network with output buffer is 74% larger than that of the direct connection. These simulations indicate that while the laser links take negiligible power over a direct connection the omegas will consume in the order of 1.6 more power due to the higher impedance of the transistors. The buffered omega network takes only 7% more power than a regular omega, substancially reduces the delay caused by an omega (see section

5.4). The results will clearly depend on the size of the drivers and length of the lines.

5.6 In-Service Flexibility

There are two main advantages to providing a programmable electronic switch network within a wafer scale system. The first is to provide a user customization capability, that is the ability to change the interconnection arrangement of the circuit blocks. The second advantage of a programmable electronic switch network is principally the capability of correcting in-service faults and the versatility of combining the yield enhancement re-configuration with interconnection adaptation to a particular problem. Since the control of omega network interconnection is programmable, the routes can be changed for different purposes. When chips are tested, routes can be changed to test the network itself or test a particular cell and finally the whole system; when in-service, the routes can be changed to connect redundant cells to avoid system failure. The in-service failure is important in such a wafer scale system because of the difficulty of repairing a wafer-level system. Conventional systems using packaged VLSI circuits can be repaired by replacing defective IC's. While wafer scale systems could involve a simple module replacement, the WSI package would probably be expensive. Also for high reliability systems (aircraft, spacecraft, medical, military) it may not be possible to allow the system to fail and be replaced. Self healing systems are needed in these cases.

A pure physical restructuring network can not be changed when in service. The sequence to finish a WSI chip with physical restructuring interconnection networks is:

1. Test interconnection networks

2. Connect and test function blocks

3. Remove failed function blocks and form a whole system

After restructuring and packaging, the whole system needs to return to the laser table for repair. However, in some cases, we do not need to change connection in-service routes and then we can make full use of the advantages of the physical restructuring technique. Depending on the system architecture, it may be possible to use simple bypass circuits which would driver the cells input to the output. In this form all the working cells are initially laser link connected to the network, and the required ones activates the cell with a download select bit, which sets a switch to load the data into a cell or divert the signal past the cell for processing or diverts the signal past it. Spare cells are added by changing the select bits. This system has been successfully tested at Lincoln Lab [14].

5.7 Proposals of New Networks

The discussion in the previous sections compares the two kinds of networks. For a physical restructuring crossbar network, the main advantage is saving area and the most significant disadvantage is the lack of the in-service flexibility. For the electronic switch omega network, the advantage of high flexibility is balanced by increased area and the enhanced possibility of switch failure. This suggests a network which combines both technique to gain reduced area with electronic switching.

The new network consumes a reasonable area which is larger than that of the

crossbar network but smaller than that of the omega network, and it also has in-service flexibility. The design idea is to use an electronic switch omega network as the interconnection network and integrate the physical restructuring technique into it. With the integration of the physical restructuring technique, a lot of area can be saved.

5.7.1 Redundant Lines

When fabricating a WSI system, a fraction of the tracks between two omega networks frequently cannot be used because of defects. The bus lines affected by defects is between 10% and 20% of the total lines [3]. This means that 10 bus lines should be designed to transfer 8 signals, with 2 spare lines for redundancy purposes. As noted in Section 2.3 the omega networks handle 2^l lines, limiting them to 2, 4, 8, 16 etc lines. The area scales as $l \times 2^{l-1}$.Consider the 8 x 8 omega network for redundancy itself, if there are 6 signals transferred through the network, the network has 2 redundant lines. If the number of signals is 7, the number of redundant lines is 1. If the network transfers 8 signals, it does not have any redundant lines and does not have any fault-tolerance. See the analysis in Table 5.4. For fault-tolerance, we should use two 8 x 8 omega networks to transfer 8 signals. Considering the defect is only 10% to 20% of the total lines, the 50% redundancy is more than required.

If we combine the laser linking/cutting technique with the electronic switch technique, we can design the system with two extra lines which can be connected or disconnected by laser linking/cutting. As shown in Figure 5.11 (b), two extra lines stay only between two omega networks, and there are several columns of link switches

Table 5.4: Analysis of the fault-tolerance ability of the 8 x 8 omega network

Transferred Signals	Redundant Lines	Redundancy Percentage
4	4	50%
6	2	25%
7	1	12.5%
8	0	0

connected to these tracks. When some tracks fail, the redundant lines are connected by the laser link technique. This design increases the redundancy from zero to 20 percent when 8 signals are transferred.

We can save a lot of area using this practical technique. In particular, when 8 signals are transferred, we save the area of a 8 x 8 omega network (transfer block and control block). In exchange, we only add the area of 40 link switches which is significantly less than the whole omega network.

5.7.2 Combined Omega/Laser Link Redundancy

The failure of a single omega network would influence the whole bus system. If an omega network contains a failure of just one full switch in the network it can render that omega network useless. Also as noted earlier, 50% of all circuit failure are from power shorts which would require the removal of that omega network from operation. In earlier case the signals cannot pass that interconnection node. This creates a single point failure for all lines on that bus. The sensitivity to the errors of electronic switches provided significant difficulties in the practical implementation of that type of defect avoidance. By comparison the laser links have shown significant resistance

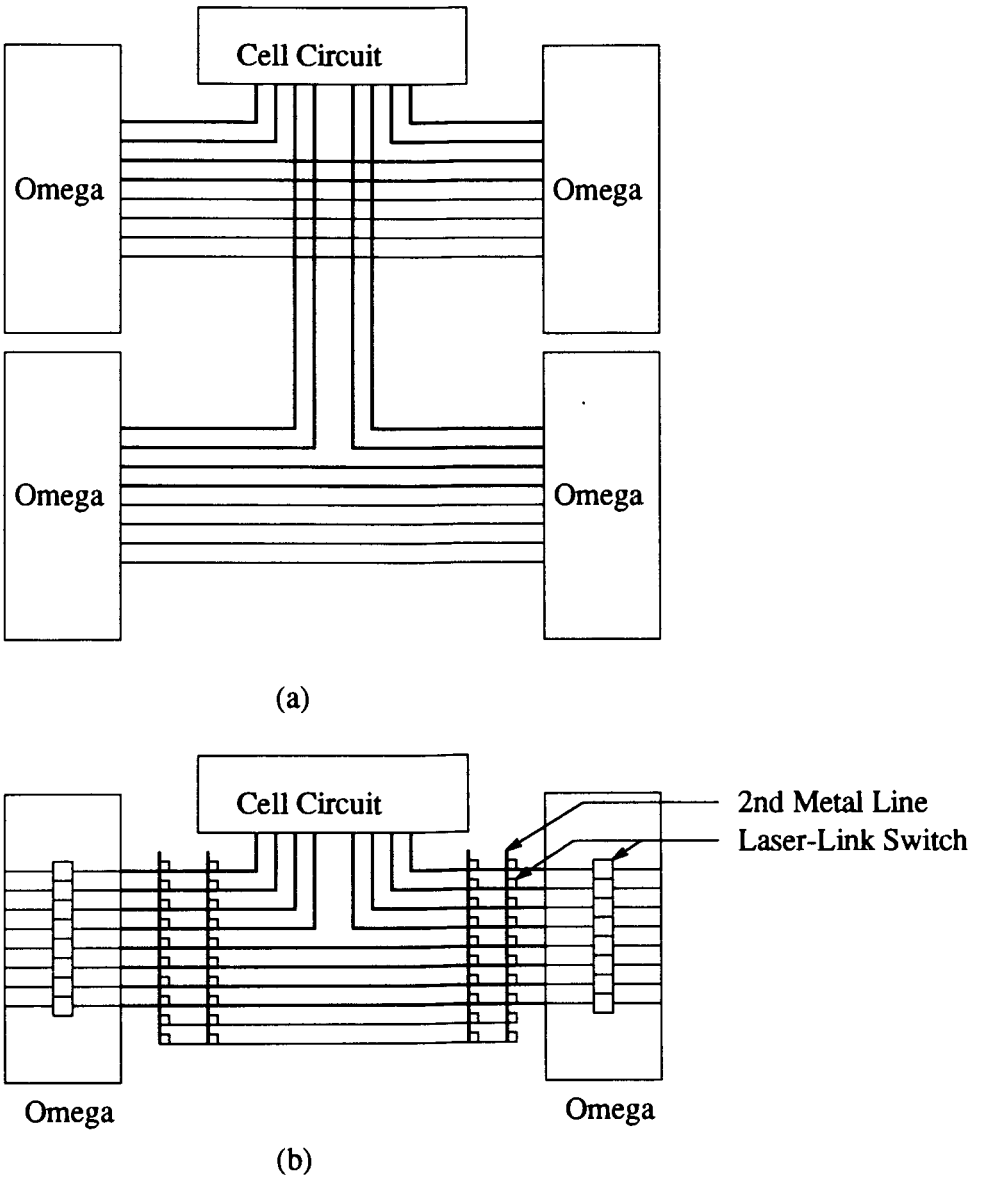


Figure 5.11: (a) Using two 8 x 8 omega networks to transfer 8 signals for the defect-avoidance of the system itself. (b) A new design to integrate the link switches into the 8 x 8 omega network to get the defect-avoidance ability.

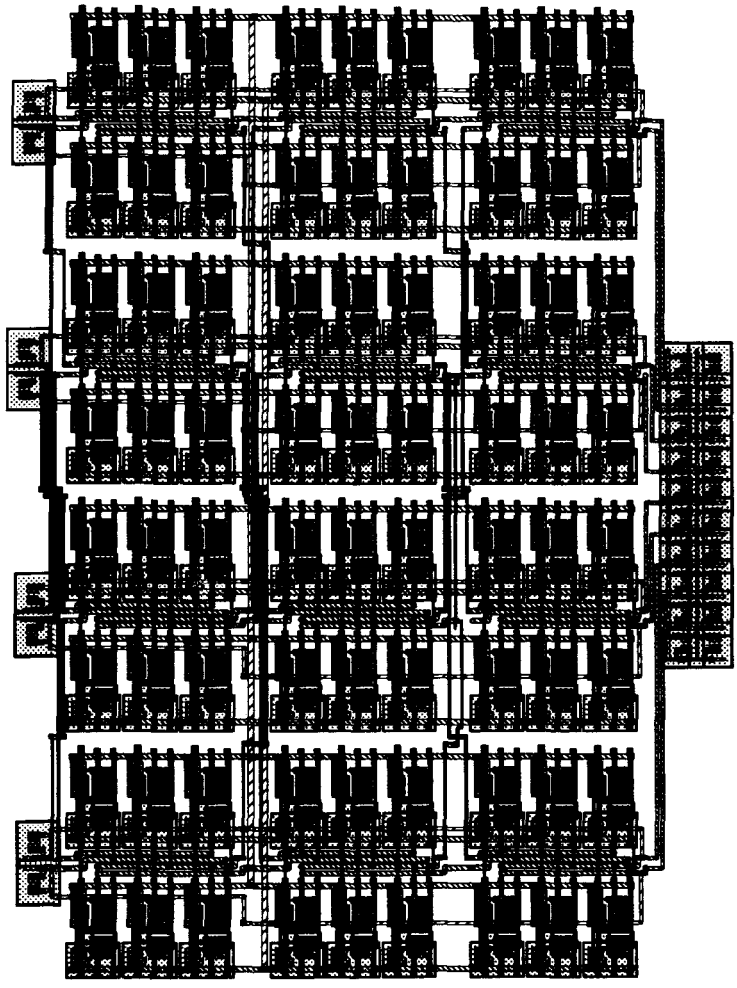


Figure 5.12: Test circuit of combined 8x8 omega network and laser link switches.

to this type of failure. To avoid electronic switch failures, Dr. Chapman proposes to embed the laser link switches into the omega network. As shown in Figure 5.11 (b), this approach places a laser link switch between each pair of input and output lines of an omega network. Originally the laser link switches are unconnected. If tests shows that an omega network fails, these laser link switches will be connected to make the signals bypass the omega network. Figure 5.12 shows a test design implementation of the combined switch, which has not yet been fabricated.

This network model shows that combining the two techniques (laser link and electronic switching) can save considerable area, while building the omega network's flexible defect-avoidance ability for self healing into the bus itself. At the same time it removes the sensitivity to the failure of the switches during the fabrication.

5.7.3 Two Level Bus System

In a complex WSI system, using two-level bus systems can also save a lot of area. The structure of a two-level bus system is shown in Figure 5.13 (a). Figure 5.13 (b) shows the structure of first level system. The first level deals with the function blocks. In this level, the interconnection networks are crossbar networks using the physical restructuring technique. The system in this level can finish part of the function of the whole system and have the defect-avoidance properties, just as we discussed in Chapter 3. The second level deals with the sub-bus system in the first level. In the second level, the electronic switch interconnection networks are used as interconnection networks. At this level, the system has both defect-avoidance and fault-tolerance properties.

A WSI system using this structure has several advantages compared to a system with pure electronic switch omega networks. Because the first-level bus system uses laser link, that section has faster speed and smaller area. On the other side, because it uses electronic switch omega networks on a smaller scale, it is also fault-tolerant.

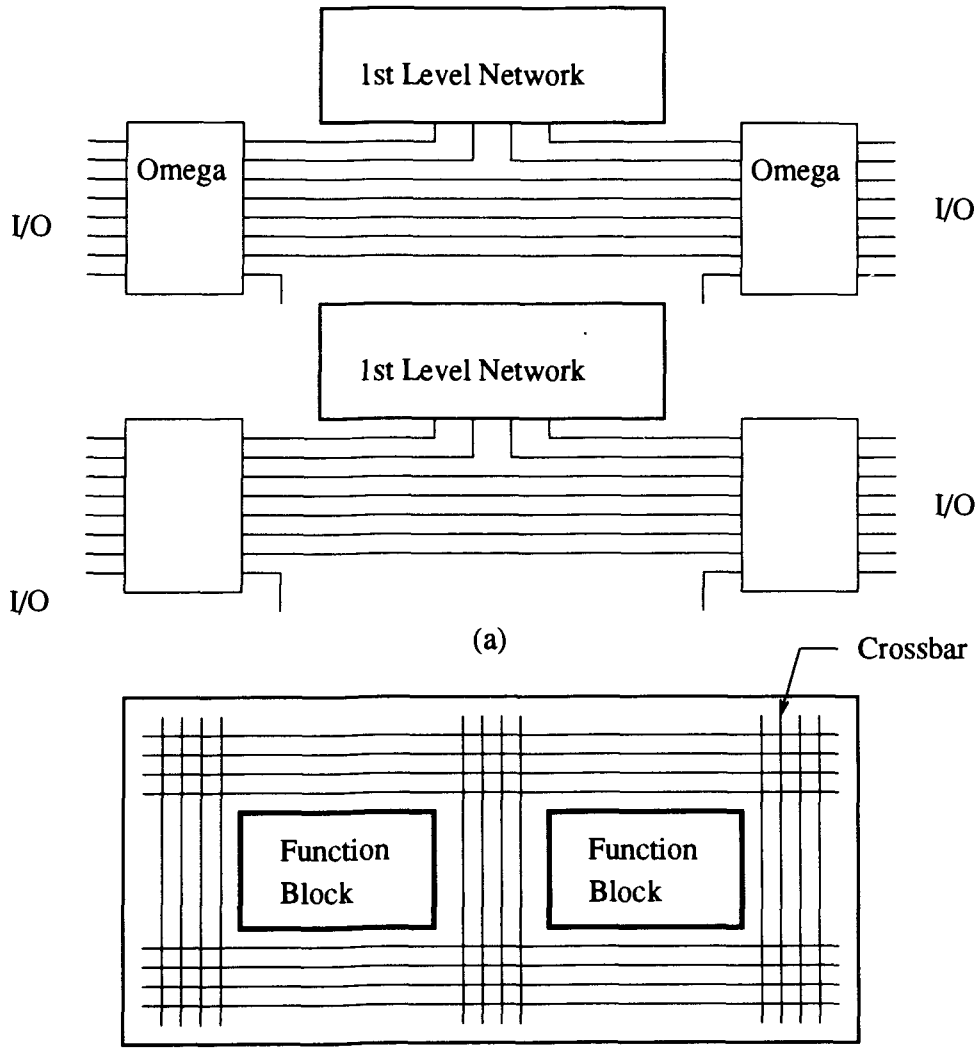


Figure 5.13: (a) The second level bus system; (b) The first level bus system.

5.8 Conclusions

We have established some parameters to compare the advantages and disadvantages of an omega network using the electronic switches technique and a crossbar network using the laser-link physical restructuring technique. In general the laser links have been shown to take less area, require less power from cell drivers for the bus lines, and thus allow faster system operation. The laser method's main disadvantage comes from the lack of flexibility to change the signal path after the links are set. The electronic omega switch takes considerably more area, and forces a slower operating speed. However the omega has a significant capability for multiple changes in the signal paths. Combining these two produces the advantage of both methods.

CHAPTER 6

Conclusions

This thesis studies two kinds of interconnection networks which are used in the bus system of Wafer Scale Integration (WSI) for defect-avoidance and fault-tolerance purposes.

An 8 x 8 crossbar network using laser link switches was designed using C Design Language (CDL) in the CMC CMOS3 process. Laser linking was employed as a post-fabrication process to accomplish defect-avoidance. A series of experiments were carried out to determine the laser linking/cutting parameters. For laser linking switches fabricated using the CMOS3 process, the best linking effects result from using laser power at 2.6 Watts, pulse duration at 300 μs with 6 zap points along the edges of diffused area of link switches. Using the laser beam at 1.2 Watts power for 300 μs can result best cutting effects on metal lines. The CAD tool Lplatform was developed to automatically generate the laser link crossbar type buses for systems. Lplatform was employed to generate a bus system using the crossbar networks for

Thermal Pixel Array.

The transfer block of the 8 x 8 electronic switch omega network was constructed using transmission gates. The electrical characteristic of the transfer block was designed and measured and the experimental results follow the expected trends shown by the HSPICE simulation results. The rising delay of the block is 19.59 ns and the falling delay of the block is 12.61 ns. The average transfer resistance of the block is 837 Ω. The control block of the omega network has been designed. There are four parts in the control block: select block, shift register, 2-bits flip-flop and decoder.

The advantages and disadvantages of laser link crossbar and electronic switch omega networks have been discussed and compared. The advantages of the bus system using the laser link restructuring crossbar network are reduced area, less power and faster circuit speed. Its disadvantages are that it requires a post-fabrication process and does not have in-service flexibility. The advantage of the bus system using electronic switch omega networks is that it possesses both defect-avoidance and fault-tolerance abilities at the cost of larger area, slower speeds and a sensitivity to failures in the switches themselves. We have proposed two models to combine these two techniques. In the first model, the laser link is integrated into an electronic switch omega network, which can save much area compared to electronic switch omega networks. The second model is a two-level model which uses physical restructuring crossbar networks as the first level and electronic switch omega networks as the second level. This combination can save area, increase circuit speed and most importantly, have in-service flexibility.

There are several areas requiring further study. The laser linking is a successful technique which has been used on the CMC 3-micron chips. More experiments to find

process parameters of the laser linking for CMOS 1.5, 1.2 and 0.8 micron processes chips need to be done to follow the update of the process. As to the electronic switch omega network, the future work is to combine it with a real WSI system.

Additional work is needed to realize the two models proposed in Chapter 5. The first model offers a simple and practical way to combine these two techniques. The combination can save much area for the omega network, and increase the yield of the bus system. This could be implemented in a simple test structure. Because the second model is complex, the research for this model includes aligning first level and second level bus and arranging redundant cells. More theoretical study needs to be done before design and implementation of the system are complete.

APPENDIX A

HSPICE Simulation Input File of Omega Network

```
-
* THIS CIRCUIT IS FOR OMEGA NETWORK
* BY KEWEI FANG.
* MODEL NTNMOS AND NTPMOS ARE COPIED FROM CMC.

.OPTION PROBE POST=2 ITL5=15000
*
.MODEL NTNMOS NMOS(LEVEL=2 VTO=0.9 PB=0.7
+ KP=30E-6     GAMMA=1.3     PHI=0.7      LAMBDA=0.01
+ CGSO=4.0E-10  CGDO=4.0E-10  CGBO=2.0E-10  RSH=15.0
+ CJ=4.0E-4     MJ=2.0        CJSW=8.0E-10  MJSW=2.0
+ JS=1.0E-6     TOX=8.5E-8    NSUB=1.0E16   XJ=1.0E-6
+ LD=0.7E-6     UO=750        VMAX=5.0E4    UCRIT=5.0E4
+ UEXP=0.14)
*
.MODEL NTPMOS PMOS(LEVEL=2 VTO=-0.9 PB=0.7
+ KP=9.5E-6     GAMMA=0.6     PHI=0.6      LAMBDA=0.03
+ CGSO=4.0E-10  CGDO=4.0E-10  CGBO=2.0E-10  RSH=75.0
+ CJ=1.8E-4     MJ=2.0        CJSW=6.0E-10  MJSW=2.0
+ JS=1.0E-6     TOX=8.5E-8    NSUB=2.0E15   XJ=0.9E-6
+ LD=0.6E-6     UO=250        VMAX=3.0E4    UCRIT=1.0E4
+ UEXP=0.03)
*
* CIRCUIT DESCRIPTION
* All PMOS substrates are connected to VDD.
```

```
* All NMOS substrates are connected to Ground.
MP1 4 3 20 1 NTPMOS L=3U W=224U   AD=1140P AS=1140P PD=224U PS=224U
MN2 4 3 0  0  NTNMOS L=3U W=138U  AD=344P AS=660P PD=108U  PS=216U
*
MP3 6 2 19 1 NTPMOS L=3U W=61U  AD=345P AS=345P PD=107U PS=107U
MN4 6 2 0  0  NTNMOS L=3U W=35U  AD=161P AS=468P PD=60U  PS=136U
MP5 7 6 5 1   NTPMOS L=3U W=112U AD=570P AS=570P PD=112U PS=112U
MN6 7 2 5 0   NTNMOS L=3U W=69U  AD=172P AS=172P PD=54U  PS=54U
*
MP7  9 2 18 1 NTPMOS L=3U W=61U  AD=345P AS=345P PD=107U PS=107U
MN8  9 2 0  0  NTNMOS L=3U W=35U  AD=161P AS=468P PD=60U  PS=136U
MP9  10 9 8 1   NTPMOS L=3U W=112U AD=570P AS=570P PD=112U PS=112U
MN10 10 2 8 0   NTNMOS L=3U W=69U  AD=172P AS=172P PD=54U  PS=54U
*
MP11 12 2 17 1 NTPMOS L=3U W=61U  AD=345P AS=345P PD=107U PS=107U
MN12 12 2  0  0 NTNMOS L=3U W=35U  AD=161P AS=468P PD=60U  PS=136U
MP13 13 12 11 1   NTPMOS L=3U W=112U AD=570P AS=570P PD=112U PS=112U
MN14 13 2 11 0   NTNMOS L=3U W=69U  AD=172P AS=172P PD=54U  PS=54U
*
MP15 15 14 16 1 NTPMOS L=3U W=8U   AD=24P AS=24P PD=22U PS=22U
MN16 15 14 0 0  NTNMOS L=3U W=6U   AD=18P AS=18P PD=18U  PS=18U
*
R1 22 21 10
C1 21 0 1PF
R2 7 8 320
C2 8 0 0.05PF
R3 10 11 320
C3 11 0 0.05PF
R4 13 14 10
C4 14 0 1PF
*
*SOURCE
VCC  1 0 DC 5
VCON 2 0 DC 5
VSIG 3 0 PULSE(5 0 .1NS 2NS 2NS 30NS 60NS)

*
* VOLTAGE SOURCE FOR MEASUREMENT
V16  1 16 DC 0
V17  1 17 DC 0
V18  1 18 DC 0
V19  1 19 DC 0
V20  1 20 DC 0
V21  21 5 DC 0
v22  4 22 DC 0

.OP

.TRAN 0.5NS 120NS
*.PRINT TRAN V(3) V(4) V(14) V(5) V(7) V(20) I(VCC) I(V20) I(V21) I(v22) I(V19) *I(V18) I(V17)
*.PLOT  TRAN V(3) V(4) V(14) V(5) V(7) V(20) I(VCC) I(V20) I(V21) I(v22) I(V19) *I(V18) I(V17)
```

```
*.PRINT DC POWER P(VCC)
*.PLOT  DC POWER P(VCC)

.PRINT TRAN V(3) V(14)
*.PLOT  TRAN V(3) V(14)

.PRINT TRAN P(VCC) POWER
*.PLOT  TRAN P(VCC) POWER

*.MEAS TRAN AVGPOW AVG P(VCC) FROM=0NS TO=60NS

*
.END
```

APPENDIX B

CDL Program to Generate the Crossbar Network

```
/* switch.c uses ldlink.c as connection cell to make a crossbar net-
work */
/* designed by Kewei Fang, March 31, 1991 */

#ifdef JLS
#define FUTURE
#endif

#include ¡scdl.h¿
#include ¡stdio.h¿

#define DIMENSION 8 /* the dimension is for the number of net-
work */
#define M2WIDTH 40 /* width = 8 dsm = 5 um */
#define M1WIDTH 40 /* width = 8 dsm = 5 um */

main()
{

int     count;
int     i;
int     j;
int     linkdx;     /* x spacing between links */
int     linkdy;     /* y spacing between links */
int     x;
int     y;
int     xbegin;
int     ybegin;
int     xend;
int     yend;

start_cell("switch1");


/* linkdx = 500; */
/* linkdy = 400; */
wherexy( "ldlink", "ld_dxdy", &linkdx, &linkdy);
printf("linkdx = %d , linkdy = %d\n", linkdx, linkdy);

xbegin = 0;
ybegin = 0;
xend = DIMENSION * linkdx;
yend = DIMENSION * linkdy;

/* make metal2 and metal tracks */

for (count=0; count¡(DIMENSION-1); count++)

 layer(metal2);
 width(M2WIDTH);
 wire(xbegin , count*linkdy , xend , count*linkdy, END);

 layer(metal);
 width(M1WIDTH);
 wire(count*linkdx, ybegin, count*linkdx, yend, END);


/* place ldlink */

for (i=0; i¡(DIMENSION); i++)
{
 for (j=0; j¡(DIMENSION); j++)
 {
  place("ldlink", i*linkdx, j*linkdy);


end_cell();
exit(0);
}


/* program to generate the link switch */

#ifdef JLS
#define FUTURE
#endif

#include ¡scdl.h¿
#include ¡stdio.h¿

#define llength 140 /* length of link channel = 28 DSM = 15.6 um */
#define llwidth  35 /* length of link channel = 7 DSM = 4.2 um */
#define linewidth 40 /* width of metal lines  = 8 DSM = 5 um  */
#define m2tongue 65 /* metal 2 tongue length = 13 DSM = 7.8 um */
#define m2twidth  40
     /* metal 2 tongue length width = 8 DSM = 5 um */
#define m1tongue  55
     /* metal 1 tongue length = 11 DSM = 6.6 um  */
#define m1twidth  40
     /* metal 1 tongue length width = 8 DSM = 5 um */
#define  xlink   200  /* link corner to centre of gap = 11 um */
#define  ylink   220  /* link corner to centre of gap = 11 um */
#define  mspace  45  /* space between diffusion and metal  */
#define  m1line  360  /* length of 1nd metal line to half point  */
#define  m2line  380  /* length of 2nd metal line to half point  */
#define  beam2e  40  /* 2*1/e value of laser = 8 DSM = 5 um */
#define  LL3sp  25  /* metal 2 spacing for via3cmc in CMOS3 */
#define  L3box  60  /* Size of metal 1 for via3cmc in CMOS3 */
#define  L3box2 30  /* half Size of metal 1 for via3cmc in CMOS3 */
#define  A3Vsp  25  /* half Size of metal 1 for via3cmc in CMOS3 */

/* Mbox = 60 metal box    */
/* Ccc  = 70 contact to contact spacing  */
/* Lbox = 70 large metal box for m1 to m2 vias */
/* Lbox2 = 35 half metal box for m1 to m2 vias */
/* AVsp = 30 space active to via          */
/* Vbox2 = 15 half of via                 */
/* Vbox  = 30 full width of via           */
/* MMsp = 40 metal1 to metal1 space       */
/* LLsp = 40 metal2 to metal2 space       */
/* WAen = 40 Pwell enclosure of Active    */
/* IAen = 25 Nplus enclosure of Active    */
/* JAen = 25 Pplus enclosure of Active    */
/* MaWcen = 75 Pwell enclosure of Active contact */

main()

int   leftdiff;   /* left edge of the diffusion relative to gap centre */
int   rightdiff;    /* right edge of the diffusion relative to gap cen-
tre */
int     topdiff;     /* top edge of the diffusion relative to gap cen-
tre */
int  bottomdiff;     /* top edge of the diffusion relative to gap cen-
tre */
int       x;     /* x coordinate */
int       x0;     /* x coordinate */
int       y;     /* y coordinate */
int       y0;     /* y coordinate */
int       x1;     /* x coordinate */
int       y1;     /* y coordinate */
int  contactgap;     /* difference between the contacts and the link length */
int     xouter;     /* inner x of link at contact center */
int   xouteract;     /* inner x of link active region */
int    ybottom;     /* bottom y of link */
int     xinner;     /* furtheres x of link */
int       ytop;     /* top y of link x  */
int       xvia;     /* x position of via */
int       yvia;     /* x position of via */
int    diffhalf;     /* half of diffusion pt */
int       xlink;     /* link corner to gap centre x displacement */
int       ylink;     /* link corner to gap centre y displacement */
int    dxlink1;     /* via to via limit on link x spacing */
int    dxlink2;     /* contact metal to line limit on link x spacing */
int     dylink;     /* diffusion to line limit on link y spacing */

start_cell("ldlink");

x = 0;
y = 0;
ylink = A3Vsp + Vbox2 + L3box2 + m2tongue + 5*(((llength + linewidth)/5)/2);
xlink = mspace + L3box + m1tongue + 5*((linewidth+llwidth)/5)/2);
printf("xlink = %d  ylink = %d, link length = %d\n", xlink, ylink, llength);
printf("A3Vsp = %d\n", A3Vsp);
xouter = x + xlink;
ybottom = y + ylink;
diffhalf = 5*((llwidth/5)%2);
printf("offset from center %d\n", diffhalf);
rightdiff = 5*((llwidth/5)/2);
leftdiff = rightdiff + mspace + Mbox2;
topdiff = llength/2;
bottomdiff = -llength/2;

/* set up gap diffusions left side, via first and tongue*/
xouter = x + xlink - leftdiff;
ybottom = y + ylink + bottomdiff;
xinner = x + xlink - rightdiff;
ytop = y + ylink +topdiff;
xvia = xouter;
yvia = ybottom - A3Vsp - Vbox2;
xouteract = xouter - Abox2;
makegap(xouteract,ybottom,xinner,ytop,xouter);
place("via3cmc", xvia,yvia);
printf("via position %d %d\n", xvia,yvia);
layer(metal2);
//* 1st metal (left) tongue */
 width(m2twidth);
 y0 = yvia + (m2twidth - L3box)/2;
 y1 = y0 + m2twidth/2 + LL3sp + L3box2;
 wire(xvia,y0, x,y0, x,y1, END);
 printf("left 2nd metal tongue %d %d %d %d %d %d\n",
       xvia,y0, x,y0, x,y1);
 place("via3cmc", x,y1);
 printf("metal 2 via on line %d %d\n", x, y1);

 /*right diffusion and via */
 xouter = y + xlink + leftdiff + diffhalf;
 xinner = y + xlink + rightdiff + diffhalf;
 xvia = xouter;
```

```
yvia = ybottom - A3Vsp - Vbox2;
xouteract = xouter + Abox2;
makegap(xouteract,ybottom,xinner,ytop,xouter);
place("via3cmc", xvia,yvia);
printf("via position %d %d"n", xvia,yvia);

/* calculate some limits on spaceings for links */
dylink = ytop + beam2e + linewidth/2;
dxlink1 = xvia + L3box + LL3sp;
dxlink2 = xvia + L3box2 + beam2e + linewidth/2;
printf("Via to via limit on link x spacing          = %d"n", dxlink1);
printf("Contact metal to line limit on link x spacing  = %d"n", dxlink2);
printf("Link diffusion to line limit on link y spacing = %d"n", dylink);
/* set dxlink, dylink, for use in other progms */
defn("ld'dxdy", m2line, m1line);
printf("linkdx (2nd metal line) = %d linkdy (1st metal line) = %d"n",
      m2line, m1line);

/*making metal 2 tongue */
layer(metal2);
width(m2twidth);
x0 = xouter  + (m2twidth - Mbox)/2;
wire(x0,yvia, x0,y, END);
printf("right side tongue %d %d %d %d"n", x0,yvia, x0,y);
contactgap = lllength-2*Ccc;
if(contactgap ;0)

   printf("Warning - length of link too short"n");

layer(metal);
x0 = x-linewidth/2;
y0 = y-linewidth/2;
y1 = y+m1line;
width(linewidth);
wire(x,y0, x,y1, END);
printf("metal 1 line %d %d %d %d"n", x0,y0, x,y1);
layer(metal2);
x1 = x+m2line;
wire(x0,y, x1,y, END);
printf("metal 2 line %d %d %d %d"n", x0,y0, x1,y0);

/* adding p well and nplus */

layer(pwell);
box(x0-WAen,y0-WAen,x1+WAen,y1+WAen);
printf("pwell locations, %d %d %d %d"n", x0-WAen,y0-WAen,x1+WAen,y1+WAen);

layer(nplus);
box(x0-IAen,y0-IAen,x1+IAen,y1+IAen);
printf("nplus locations, %d %d %d %d"n", x0-IAen,y0-IAen,x1+IAen,y1+IAen);

end'cell();
exit(0);
                        /* end of main program */

makegap(xinner,ybottom, xouter,ytop, xcontact)
int    xouter;     /* x bottom left corner  */
int    xinner;     /* y top right corner    */
int    ybottom;    /* x bottom right corner */
int    ytop;       /* y top left corner     */
int xcontact;  /* x position of center of contacts */

_
layer( active );
box(xouter,ybottom, xinner, ytop);
Ma(xcontact,ytop-Ccc/2);
Ma(xcontact,ybottom+Ccc/2);
printf("active layer %d %d %d %d"n", xouter, ybottom, xinner, ytop);
    layer(metal);
width(Mbox);
ybottom = ybottom - A3Vsp - Vbox2;
wire(xcontact,ybottom, xcontact,ytop, END);
printf("link metal 1, %d, %d, %d, %d"n",xcontact,ybottom, xcontact,ytop);
```

APPENDIX C

CDL Program to Generate the Bus System

```
#ifdef JLS
#define FUTURE
#endif

#include ¡scdl.h¿
#include ¡stdio.h¿

#define bif        5   /* bif per dsm */
#define xstart     0   /* x start point for frame */
#define ystart     0   /* y start point for frame */
#define nrows      3   /* number of rows of chips */
#define ncols      3   /* number of columns of chips */

#define NOROW      8   /* number lines per row of chips */
#define NOCOL      14  /* number lines per column of chips */
#define NOROS      10
#define NOCOS      16

#define MAXI       50
#define STUBL      6
#define STUBR      2
#define STUBB      14
#define STUBT      4
#define NOSIG      10

#define xcell      7000  /* x size of cell */
#define ycell      4500  /* y size of cell */
#define nstub      9   /* total number of stubs */
#define STUBLl       12  /* number of stub links in matrix */
#define stlink     2   /* start point of link in link matrix */
#define sprint     1  /* set printing cell level, 0 = all, 1 = only links */
#define linewidth 40   /* width of metal lines    = 8 DSM = 5 um    */
#define beam2e  40    /* 2*1/e value of laser = 8 DSM = 5 um */
#define  LL3sp  25   /* metal 2 spacing for via3cmc in CMOS3 */
#define  L3box  60   /* Size of metal 1 for via3cmc in CMOS3 */
#define  L3box2 30   /* half Size of metal 1 for via3cmc in CMOS3 */
#define  A3Vsp  25   /* half Size of metal 1 for via3cmc in CMOS3 */

#define  resist 0.000125   /* resistivity 1 ohm/mm = 1.25e-4
                 ohm/bif */
#define capacity  50    /* c 50 fF per link */
#define linec    0.0125      /* 0.1fF/um =0.0125fF/bif */

int dtrack[4][2] = – – 0, 400˝, –0, -1100˝, –400, 0˝, —1100,0˝ ˝;
                /* offsets for tracks */

int dou[3][2] = – –1,1˝, –1,1˝, –1,1˝ ˝;
                /*switch for 2-tracks, left, incolumn, right */
                /* bottom ,inrow ,top */

int ntrack[3][2] =  – –14,8˝, –14,8˝, –14,8˝ ˝;
               /* number of lines,left, incolumn, right */
               /* bottom ,inrow ,top */

void ccell();
void area();
void linknumc();
void linknumr();

main()

int         x;       /* x coordinate of first link */
int         y;       /* y coordinate of first link */
int     linkdx;      /* x spacing between links */
int     linkdy;      /* y spacing between links */
int     slinkdx;     /* for 2-tracks, short distance between
                        column lines */
int    slinkdy;   /* for 2-tracks, short distance between row lines */
int      coldx1;      /* x width of column */
int      rowdy1;      /* y width of row */
int      coldx2;      /* x width of column  2-tracks*/
int      rowdy2;      /* y width of row  2-tracks*/
int      lmcold;
int      mcold;        /* middle value for caculation */
int      rmcold;
int      bmrowd;
int      mrowd;         /* middle value for caculation */
int      tmrowd;
int     dxcell1;      /* x displacement of cell */
int     dycell1;      /* y displacement of cell */
int     dxcell2;      /* x displacement of cell */
int     dycell2;      /* y displacement of cell */
int       xend;      /* x end point for frame in bif */
int       yend;      /* y end point for frame in bif */
int         x0;      /* x coordinate */
int         y0;      /* y coordinate */
int         x1;      /* x coordinate */
int       . y1;      /* y coordinate */
int         x2;      /* x coordinate */
int         y2;      /* y coordinate */
int        xrig;      /* x right column width */
int        ytop;      /* y top row width */
int          i;      /* link 2x2 block counter   */
int          j;      /* row pair counter   */
int      xrcol;      /* x right col start for frame in bif */
int      ytrow;      /* y top row start for frame in bif */
int      linewidth2;

int lcell[NOROW][NOCOL], ltbedge[NOROW][NOCOL];
int lrledge[NOROW][NOCOL], lcorner[NOROW][NOCOL];
int lestubs[STUBL][NOCOS], ristubs[STUBR][NOCOS];
int bostubs[STUBB][NOROS], tostubs[STUBT][NOCOS];

/* signal column array */
static int sigcol[10][NOCOL] = –
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1˝, ˝;
       /* signal row array */

static int sigrow[10][NOROW] = –
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝,
                  –1, 1, 1, 1, 1, 1, 1, 1˝, ˝;


static int lestubd[STUBL][2] = –
                  –  0, 1300˝,
       –  0, 1705˝,
       –  0, 2175˝,
                  –  0, 2945˝,
                  –  0, 3515˝,
                  –  0, 4065˝, ˝;
                  /* matrix of cell stubs */

static int ristubd[STUBR][2] = –
                  –7000, 1300˝,
                  –7000, 1760˝, ˝;
                  /* matrix of cell stubs */

static int bostubd[STUBB][2] = –
                  –300,   0˝,
       –1000,  0˝,
                  –1400,  0˝,
                  –2100,  0˝,
                  –2500,  0˝,
                  –3200,  0˝,
                  –3600,  0˝,
```

```
-4400, 0",
-4825, 0",
-5225, 0",
-5625, 0",
-6025, 0",
-6425, 0",
-6825, 0", ";
/* matrix of cell stubs */

    -0, 4500",
    -0, 4500",
    -0, 4500",
    -0, 4500", ";
/* matrix of cell stubs */

/* left signal stubs */

static int siglero[NOSIG][STUBL] = -
    -1, 0, 0, 0, 0, 0",
    -1, 0, 0, 0, 0, 0",
    -0, 1, 0, 0, 0, 0",
    -0, 0, 1, 0, 0, 0",
    -0, 0, 0, 1, 0, 0",
    -0, 0, 0, 1, 0, 0",
    -0, 0, 0, 0, 1, 0",
    -0, 0, 0, 0, 1, 0",
    -0, 0, 0, 0, 0, 1",
    -0, 0, 0, 0, 0, 0", ";
static int sigleco[NOSIG][NOCOL] = -
    -0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1",
    -0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0",
    -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1",
    -0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0",
    -0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0", ";

/* right signal */

static int sigriro[NOSIG][STUBR] = -
    -1, 0",
    -0, 1",
    -1, 0",
    -0, 0",
    -0, 0",
    -0, 0",
    -0, 0",
    -0, 0",
    -0, 0",
    -0, 0", ";

static int sigrico[NOSIG][NOCOL] = -
    -0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
/* bottom signal */

static int sigboro[NOSIG][STUBB] = -
    -1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1", ";

static int sigboco[NOSIG][NOROW] = -
    -0, 0, 1, 0, 1, 0, 1, 0",
    -1, 0, 1, 0, 1, 0, 0, 0",
    -0, 0, 1, 0, 1, 0, 1, 0",
    -0, 0, 1, 0, 1, 0, 1, 0",
    -1, 0, 1, 0, 1, 0, 0, 0",
    -1, 0, 1, 0, 1, 0, 0, 0",
    -0, 1, 0, 1, 0, 1, 0, 0",
    -0, 0, 1, 0, 1, 0, 1, 0",
    -1, 0, 1, 0, 1, 0, 0, 0",
    -1, 0, 0, 0, 0, 0, 0, 1", "
/* top signal */

static int sigtoro[NOSIG][STUBT] = -
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0",
    -0, 0, 0, 0", ";
static int tostubd[STUBT][2] = -
    static int sigtoco[NOSIG][NOROW] = -
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",
    -0, 0, 0, 0, 0, 0, 0, 0",


start cell("dgrid3c");

/* form crosspart */


for (i=0; i¡NOROW; i++)

  for (j=0; j¡NOCOL; j++)
    lcell[i][j] = 0;

formc(NOSIG,NOROW,NOCOL,sigrow,sigcol,lcell);
for (i=0; i¡NOROW; i++)
-
  for (j=0; j¡NOCOL; j++)
  -
    ltbedge[i][j] = lcell[i][j];
    lrledge[i][j] = lcell[i][j];
    lcorner[i][j] = lcell[i][j];

    printf(""n lcell[%d][%d] = %d "n", i,j,lcell[i][j]);


/* form left stub */
for (i=0; i¡STUBL; i++)
-
  lestubs[i][0] = lestubd[i][0];
  lestubs[i][1] = lestubd[i][1];
  for (j=2; j¡NOCOS; j++)
    lestubs[i][j] = 0;

forml(NOSIG,STUBL,NOCOL,siglero,sigleco,lestubs);

printf(""n LEFT STUB ARRAY"n");
for (i=0; i¡STUBL; i++)
-
  for (j=2; j¡NOCOS; j++)
    printf(""n lestubs[%d][%d] = %d "n",i,j,lestubs[i][j]);
  "

/* form right stub */
for (i=0; i¡STUBR; i++)
-
  ristubs[i][0] = ristubd[i][0];
  ristubs[i][1] = ristubd[i][1];
  for (j=2; j¡NOCOS; j++)
    ristubs[i][j] = 0;

formr(NOSIG,STUBR,NOCOL,sigriro,sigrico,ristubs);

printf(""n RIGHT STUB ARRAY "n");
for (i=0; i¡STUBR; i++)

  for (j=2; j¡NOCOS; j++)
    printf(""n ristubs[%d][%d] = %d "n",i,j,ristubs[i][j]);
  "

/*form bottom stub */
for (i=0; i¡STUBB; i++)
-
  bostubs[i][0] = bostubd[i][0];
  bostubs[i][1] = bostubd[i][1];
  for (j=2; j¡NOROS; j++)
    bostubs[i][j] = 0;
  "
formr(NOSIG,STUBB,NOROW,sigboro,sigboco,bostubs);

printf(""n BOTTOM STUB ARRAY "n");
for (i=0; i¡STUBB; i++)
-
  for (j=2; j¡NOROS; j++)
    printf(""n bostubs[%d][%d] = %d "n",i,j,bostubs[i][j]);
  "
```

```
/* form top stub */
for (i=0; i;STUBT; i++)

   tostubs[i][0] = tostubd[i][0];
   tostubs[i][1] = tostubd[i][1];
   for (j=2; j;NOROS; j++)
    bostubs[i][j] = 0;

formr(NOSIG,STUBT,NOROW,sigtoro,sigtoco,tostubs);

printf(""n TOP STUB ARRAY "n");
for (i=0; i;STUBT; i++)

   for (j=2; j;NOROS; j++)
    printf(""n tostubs[%d][%d] = %d "n",i,j,tostubs[i][j]);


/* input and form grid */

x = 0;
y = 0;
linewidth2 = 5*((linewidth/5)/2);

wherexy( "ldlink", "ld'dxdy", &linkdx, &linkdy);
printf("linkdx = %d , linkdy = %d"n", linkdx, linkdy);

slinkdx = linkdx/4;
slinkdy = linkdy/4;
printf("slinkdx = %d, slinkdy = %d"n",slinkdx,slinkdy);

/* caculation of distance */

coldx1 = (ntrack[1][0]-1)*linkdx - dtrack[3][0] + dtrack[2][0];
rowdy1 = (ntrack[1][1]-1)*linkdy - dtrack[1][1] + dtrack[0][1];

if (ntrack[0][0]%2 == 0)
  lmcold = (ntrack[0][0]/2)*linkdx + ((ntrack[0][0]-2)/2)*slinkdx
else if (ntrack[0][0]%2 != 0 )
  lmcold =  ((ntrack[0][0]-1)/2)*(linkdx+slinkdx);

if (ntrack[1][0]%2 == 0)
  mcold = (ntrack[1][0]/2)*linkdx + ((ntrack[1][0]-2)/2)*slinkdx
else if (ntrack[1][0]%2 != 0 )
  mcold =  ((ntrack[1][0]-1)/2)*(linkdx+slinkdx);

if (ntrack[2][0]%2 == 0)
  rmcold = (ntrack[1][0]/2)*linkdx + ((ntrack[2][0]-2)/2)*slinkdx ;
else if (ntrack[2][0]%2 != 0 )
  rmcold =  ((ntrack[2][0]-1)/2)*(linkdx+slinkdx);

if (ntrack[0][1]%2 == 0)
  bmrowd = (ntrack[0][1]/2)*linkdy + ((ntrack[0][1]-2)/2)*slinkdy
else if (ntrack[0][1]%2 != 0 )
   bmrowd =  ((ntrack[0][1]-1)/2)*(linkdy+slinkdy);

if (ntrack[1][1]%2 == 0)
  mrowd = (ntrack[1][1]/2)*linkdy + ((ntrack[1][1]-2)/2)*slinkdy
else if (ntrack[1][1]%2 != 0 )
   mrowd =  ((ntrack[1][1]-1)/2)*(linkdy+slinkdy);

if (ntrack[2][1]%2 == 0)
  tmrowd = (ntrack[2][1]/2)*linkdy + ((ntrack[2][1]-2)/2)*slinkdy ;
else if (ntrack[2][1]%2 != 0 )
   tmrowd =  ((ntrack[2][1]-1)/2)*(linkdy+slinkdy);


coldx2 =  mcold - dtrack[3][0] + dtrack[2][0];
rowdy2 =  mrowd - dtrack[1][1] + dtrack[0][1];

dxcell1 = xcell + coldx1;
dycell1 = ycell + rowdy1;
dxcell2 = xcell + coldx2;
dycell2 = ycell + rowdy2;

if (dou[0][0]==1)
 x0=x - lmcold + dtrack[3][0];
else
 x0 = x -(ntrack[0][0]-1)*linkdx + dtrack[3][0];
if (dou[0][1]==1)
 y0=y - bmrowd + dtrack[1][1];
else
 y0 = y -(ntrack[0][1]-1)*linkdy + dtrack[1][1];

if ( dou[1][0] ==1 )
xrcol = x + (ncols-1)*dxcell2 + xcell +dtrack[2][0];
else
xrcol = x + (ncols-1)*dxcell1 + xcell +dtrack[2][0];
if (dou[1][1] ==1)
ytrow = y + dycell2*(nrows-1) + ycell + dtrack[0][1];
else
ytrow = y + dycell1*(nrows-1) + ycell + dtrack[0][1];

if (dou[2][0]==1)
xrig = rmcold;
else
xrig = (ntrack[2][0]-1)*linkdx;
if (dou[2][1]==1)
ytop = tmrowd;
else
ytop = (ntrack[2][1]-1)*linkdy;

xend = xrcol +xrig;
yend = ytrow + ytop;
/* end of distance */

/* row */
if (dou[0][1]==1)
 srow(0, x0,y0,xend,linkdy,slinkdy,ntrack[0][1]);
else
 row(0,x0,y0,xend,linkdy,ntrack[0][1]);

for (i = 1; i ; nrows; i++)
 {
  if (dou[1][1] ==1 )
  {
  y1 = y + dycell2*i;
  y2 = y1 - mrowd + dtrack[1][1] ;
  srow(nrows,x0,y2,xend,linkdy,slinkdy,ntrack[1][1]) ;
  }
  else
  {
  y1 = y + dycell1*i;
  y2 = y1 - (ntrack[1][1] -1)*linkdy + dtrack[1][1] ;
  row(i, x0, y2, xend, linkdy,ntrack[1][1]);
  }


if (dou[2][1]==1)
 srow(0, x0,ytrow,xend,linkdy,slinkdy,ntrack[2][1]);
else
 row(nrows,x0,ytrow,xend,linkdy,ntrack[2][1]);
/* end of row */

/* column */
if (dou[0][0]==1)
 scolumn(0, x0, y0, yend,linkdx,slinkdx,ntrack[0][0]);
else
 column(0, x0, y0, yend, linkdx,ntrack[0][0]);

for ( j = 1; j ; ncols; j++)
 {
  if (dou[1][0] ==1 )
  {
  x1 = x + dxcell2*j;
  x2 = x1 - mcold + dtrack[3][0];
  scolumn(j,x2,y0,yend,linkdx,slinkdx,ntrack[1][0]);
  }
  else
  {
  x1 = x + dxcell1*j;
  x2 = x1 - (ntrack[1][0] -1)*linkdx + dtrack[3][0];
  column(j,x2,y0,yend,linkdx,ntrack[1][0]);
  }


if (dou[2][0]==1)
 scolumn(ncols, xrcol, y0, yend,linkdx,slinkdx,ntrack[2][0]);
else
 column(ncols, xrcol, y0, yend, linkdx,ntrack[2][0]);
/* end of column */

/* cross */
/* bottom left */
if ( (dou[0][0]==1) && (dou[0][1]==0))
   clcross(x0, y0,linkdx,slinkdx,linkdy,lcorner);
else if ( (dou[0][0]==0) && (dou[0][1]==1))
   rlcross(x0, y0,linkdx,linkdy,slinkdy,lcorner);
else if ( (dou[0][0]==0) && (dou[0][1]==0))
   lcross(x0, y0,linkdx,linkdy,lcorner);
else
    dlcross(x0, y0,linkdx,slinkdx,linkdy,slinkdy,lcorner);

/* top left */
if ( (dou[0][0]==1) && (dou[2][1]==0))
   clcross(x0, ytrow,linkdx,slinkdx,linkdy,lcorner);
else if ( (dou[0][0]==0) && (dou[2][1]==1))
   rlcross(x0, ytrow,linkdx,linkdy,slinkdy,lcorner);
else if ( (dou[0][0]==0) && (dou[2][1]==0))
   lcross(x0, ytrow,linkdx,linkdy,lcorner);
else
   dlcross(x0, ytrow,linkdx,slinkdx,linkdy,slinkdy,lcorner);

/* bottom right */
if ( (dou[2][0]==1) && (dou[0][1]==0))
   clcross(xrcol, y0,linkdx,slinkdx,linkdy,lcorner);
else if ( (dou[2][0]==0) && (dou[0][1]==1))
   rlcross(xrcol, y0,linkdx,linkdy,slinkdy,lcorner);
else if ( (dou[2][0]==0) && (dou[0][1]==0))
   lcross(xrcol, y0,linkdx,linkdy,lcorner);
```

```
else
   dlcross(xrcol, y0,linkdx,slinkdx,linkdy,slinkdy,lcorner);

/* top right */
if ( (dou[2][0]==1) && (dou[2][1]==0))
   clcross(xrcol, ytrow,linkdx,slinkdx,linkdy,lcorner);
else if ( (dou[2][0]==0) && (dou[2][1]==1))
   rlcross(xrcol, ytrow,linkdx,linkdy,slinkdy,lcorner);
else if ( (dou[2][0]==0) && (dou[2][1]==0))
   lcross(xrcol, ytrow,linkdx,linkdy,lcorner);
else
    dlcross(xrcol, ytrow,linkdx,slinkdx,linkdy,slinkdy,lcorner);

/* inter cross */
for (i = 0; i ¡ nrows; i++)
  {
if ( i ¿ 0 )
   {

/* left side col cross points */
   printf(""nRow & col cross links for cell i=%2d j=%2d at
        %5d %5d"n", i, 0, x0, y2);
   if (dou[1][1] ==1 )
  {
   y1 = y + dycell2*i;
   y2 = y1 - mrowd + dtrack[1][1] ;

   else
  {
   y1 = y + dycell1*i;
   y2 = y1 - (ntrack[1][1] -1)*linkdy + dtrack[1][1] ;
  "
   if ( (dou[0][0]==1) && (dou[1][1]==0))
   clcross(x0, y2,linkdx,slinkdx,linkdy,ltbedge);
else if ( (dou[0][0]==0) && (dou[1][1]==1))
   rlcross(x0, y2,linkdx,linkdy,slinkdy,ltbedge);
else if ( (dou[0][0]==0) && (dou[1][1]==0))
   lcross(x0, y2,linkdx,linkdy,ltbedge);
else
    dlcross(x0, y2,linkdx,slinkdx,linkdy,slinkdy,ltbedge);


/* right side col cross points */
   printf(""nRow & col cross links for cell i=%2d j=%2d
        at %5d %5d"n", i, 0, xrcol, y2);
if ( (dou[2][0]==1) && (dou[1][1]==0))
   clcross(xrcol, y2,linkdx,slinkdx,linkdy,ltbedge);
else if ( (dou[2][0]==0) && (dou[1][1]==1))
   rlcross(xrcol, y2,linkdx,linkdy,slinkdy,ltbedge);
else if ( (dou[2][0]==0) && (dou[1][1]==0))
   lcross(xrcol, y2,linkdx,linkdy,ltbedge);
else
   dlcross(xrcol, y2,linkdx,slinkdx,linkdy,slinkdy,ltbedge);
  " /* end of i¿0 */

  for ( j = 0; j ¡ ncols; j++)
  {
  if (dou[1][0] ==1 )
  {
  x1 = x + dxcell2*j;
  x2 = x1 - mcold + dtrack[3][0];

  else
  {
  x1 = x + dxcell1*j;
  x2 = x1 - (ntrack[1][0] -1)*linkdx + dtrack[3][0];
  "

/* links for bottom row crosspoint excluding corners */
     if ( (i == 0) && (j ¿ 0) && ( j ¡ ncols) )
       {
       printf(""nBottom Row & col cross links: cell i=%2d j=%2d
            at %5d %5d"n", i, j, x2, y2);

if ( (dou[0][1]==1) && (dou[1][0]==0))
   rlcross(x2, y0,linkdx,linkdy,slinkdy,ltbedge);
else if ( (dou[0][1]==0) && (dou[1][0]==1))
   clcross(x2, y0,linkdx,slinkdx,linkdy,ltbedge);
else if ( (dou[0][1]==0) && (dou[1][0]==0))
   lcross(x2, y0,linkdx,linkdy,ltbedge);
else
   dlcross(x2, y0,linkdx,slinkdx,linkdy,slinkdy,ltbedge);
        " /*end of bottom row crosspoint */

/* making links for top row col crosspoint excluding corners */
     if ( (i == nrows-1) && (j ¿ 0)  && ( j ¡ ncols) )
       {
       printf(""nRow & col cross links for cell i=%2d j=%2d at
       %5d %5d"n".
       ncols, j, x2, ytrow);
if ( (dou[0][1]==1) && (dou[1][0]==0))
   rlcross(x2, ytrow,linkdx,linkdy,slinkdy,ltbedge);
else if ( (dou[0][1]==0) && (dou[1][0]==1))
   clcross(x2, ytrow,linkdx,slinkdx,linkdy,ltbedge);
else if ( (dou[0][1]==0) && (dou[1][0]==0))
   lcross(x2, ytrow,linkdx,linkdy,ltbedge);
else
   dlcross(x2, ytrow,linkdx,slinkdx,linkdy,slinkdy,ltbedge);
        " /* end of top row crosspoint */
     if ( (i ¿ 0) && (j ¿ 0) )
       {
       printf(""nRow & col cross links for cell i=%2d j=%2d
            at %5d %5d"n",
            i, j, x2, y2);

if ( (dou[0][1]==1) && (dou[1][0]==0))
   rlcross(x2, y2,linkdx,linkdy,slinkdy,lcell);
else if ( (dou[0][1]==0) && (dou[1][0]==1))
   clcross(x2, y2,linkdx,slinkdx,linkdy,lcell);
else if ( (dou[0][1]==0) && (dou[1][0]==0))
   lcross(x2, y2,linkdx,linkdy,lcell);
else
    dlcross(x2, y2,linkdx,slinkdx,linkdy,slinkdy,lcell);

        " /*  end of inter crosspoint */

     "     /* end j loop */
   "    /* end i loop */

/* insert stubs */
for (i = 0; i ¡ nrows; i++)
  {
   for ( j = 0; j ¡ ncols; j++)

     if ((dou[1][0] == 0) && (dou[1][1] == 0))
       {
        x1 = x + dxcell1*j;
        y1 = y + dycell1*i;
        ccell(x1,y1, linkdx, slinkdx, linkdy, slinkdy,
        dou[1][0], dou[1][0], dou[1][1], dou[1][1]) ;
        "
      else if  ((dou[1][0] == 1) && (dou[1][1] == 0))

        x1 = x + dxcell2*j;
        y1 = y + dycell1*i;
        ccell(x1,y1, linkdx, slinkdx, linkdy, slinkdy,
        dou[1][0], dou[1][0], dou[1][1], dou[1][1]) ;
        "
       else if  ((dou[1][0] == 0) && (dou[1][1] == 1))
       {
        x1 = x + dxcell1*j;
        y1 = y + dycell2*i;
        ccell(x1,y1, linkdx, slinkdx, linkdy, slinkdy,
        dou[1][0], dou[1][0], dou[1][1], dou[1][1]) ;
        "
       else if  ((dou[1][0] == 1) && (dou[1][1] == 1))
       {
        x1 = x + dxcell2*j;
        y1 = y + dycell2*i;
        ccell(x1,y1, linkdx, slinkdx, linkdy, slinkdy,
        dou[1][0], dou[1][0], dou[1][1], dou[1][1]) ;
        "
     "



/* report */
area(y0, yend, linkdx, slinkdx, linkdy, slinkdy);

linknumc(ntrack[1][0],ntrack[1][1],STUBL,STUBR,lcell,
      ltbedge,lrledge,lcorner,lestubs,ristubs,y0,yend,linkdy);
linknumr (ntrack[1][0],ntrack[1][1],STUBT,STUBB,lcell,ltbedge,
      lrledge,lcorner,lestubs,ristubs,x0,xend,linkdx);

end"cell();
exit(0);
" /* end of main program */

row(rowno, x,y, xend, linkdy, nline )

int       rowno;  /* row number */
int          x;  /* x start of row */
int          y;  /* y start of row */
int       linkdy;     /* y spacing between links */
int         xend;     /* x end point for frame in bif */
int       nline;      /* n of track */
{
int       i; /* counter */
int      x0; /* link x location */
int      y0; /* link y location */

    layer(metal2);
    width(linewidth);
printf(""nRow channel %2d"n", rowno);
for ( i = 0; i ¡ nline; i++)
  {
  y0 = y + i*linkdy;
  printf("row %2d, track %2d; %5d, %5d; %5d %5d"n",
       rowno, i, x,y0, xend, y0);
  wire(x, y0, xend, y0, END);
```

```
"    /* end row routine */

column(colno, x,y, yend, linkdx, nline)

int       colno; /* col number */
int          x; /* x start of row */
int          y; /* y start of row */
int       yend;     /* y end point for frame in bif */
int      linkdx;     /* x spacing between links */
int      nline;     /* n of tracks */
{
int      i; /* counter */
int     x0; /* link x location */
int     y0; /* link y location */

     layer(metal);
     width(linewidth);
printf(""nColumn channel %2d"n", colno);
for ( i = 0; i ; nline; i++)
  {
  x0 = x + i*linkdx;
  printf("col %2d, track %2d; %5d, %5d; %5d %5d"n",
       colno, i, x0,y, x0, yend);
  wire(x0, y, x0, yend, END);
  }
"    /* end column routine */

srow(rowno,x,y,xend,linkdy,slinkdy,nline)
int rowno;
int x;
int y;
int xend;
int linkdy;
int slinkdy;
int nline;     /* n of tracks */
{
int i;
int x0;
int y1;
int y0;

layer(metal2);
width(linewidth);
y1 = y ;
wire(x,y1,xend,y1,END);
printf(""nRow channel %2d"n", rowno);
for (i=1; i;nline; i++)
  {
if (i % 2 != 0 )
  y0 = y1 + linkdy  ;
else
  y0 = y1 + slinkdy ;
wire(x, y0, xend, y0, END);
y1 = y0 ;
" /* end of for */
" /* end of srow */

scolumn(colno,x,y,yend,linkdx,slinkdx,nline)
/* for 2-tracks by kewei */
int colno;
int x;
int y;
int yend;
int linkdx;
int slinkdx;
int nline;     /* n of tracks */
{
int i;
int x0;
int x1;
int y0;

layer(metal);
width(linewidth);
x1 = x ;
wire(x1,y,x1,yend,END);
printf(""nSColumn channel %2d"n", colno);

for (i=1; i;nline; i++)
  {
if (i % 2 != 0 )
  x0 = x1 + linkdx  ;
else
  x0 = x1 + slinkdx ;
wire(x0, y, x0, yend, END);
x1 = x0 ;
" /* end of for */
" /* end of scolumn */

lcross(x,y, linkdx, linkdy, links)

int          x; /* x start of row */
int          y; /* y start of row */
int      linkdx; /* x spacing between links */
int      linkdy; /* y spacing between links */
int   links[NOROW][NOCOL]; /* Array of link points */

int      i; /* counter */
int      j; /* counter */
int     x0; /* link x location */
int     y0; /* link y location */


for (i=0; i ; NOROW; ++i)

  for (j=0; j ; NOCOL; ++j)

    if (sprint == 0)
      printf("testing %d %d link %d"n", i, j, links[i][j]);
    y0 = y + linkdy*i;
    if (links[i][j] ¿ 0 )
      {
      x0 = x + linkdx*j;
      place("ldlink",x0,y0);
      printf("link tracks: row %2d col %2d: link type
            %2d at %5d %5d"n".
          i, j, links[i][j], x0, y0);
"    /* end if statement */
          /* end j loop */
          /* end i loop */
          /* end lcross function */

clcross(x,y,linkdx,slinkdx,linkdy,links)
/* for 2-tracks, added by kewei */
int x;
int y;
int linkdx;
int slinkdx;
int linkdy;
int links[NOROW][NOCOL];
{
int i;
int j;
int x0;
int y0;

for (i=0; i ; NOROW; ++i)
  {

  for (j=0; j ; NOCOL; ++j)

      if (sprint == 0)
       printf("testing %d %d link %d "n", i,j, links[i][j]);
      y0 = y + linkdy*i;
       if ( links[i][j]¿0 )
         {
           if (j%2 == 0)

             x0 = x + (j/2)*linkdx +(j/2)*slinkdx ;
             place("ldlink", x0, y0);
             " /* end of i%2 == 0 */
           else

             x0 = x + ((j+1)/2)*linkdx + ((j-1)/2)*slinkdx ;
             place("ldlink,MY", x0,y0);
            " /* end of else */
         " /* end of links(i,j)¿0 */
    "  /* end of j loop */
  " /* end of i loop */
" /* end of  clcross */


rlcross(x,y,linkdx,linkdy,slinkdy,links)
int x;
int y;
int linkdx;
int slinkdy;
int linkdy;
int links[NOROW][NOCOL];
{
int i;
int j;
int x0;
int y0;

for (i=0; i ; NOROW; ++i)


  for (j=0; j ; NOCOL; ++j)
    {
      if (sprint == 0)
       printf("testing %d %d link %d "n", i,j, links[i][j]);
       x0 =x + linkdx*j;
       if ( links[i][j]¿0 )
         {
           if (i%2 == 0)
             {
             y0 = y + (i/2)*linkdy +(i/2)*slinkdy ;
             place("ldlink", x0, y0);
```

```
            " /* end of i%2 == 0 */
          else
            -
            y0 = y + ((i+1)/2)*linkdy + ((i-1)/2)*slinkdy
            place("ldlink,MX", x0,y0);
              /* end of else */
        " /* end of links(i,j)¿0 */
    " /* end of j loop */
  " /* end of i loop */
" /* end of  rlcross */

dlcross(x,y,linkdx,slinkdx,linkdy,slinkdy,links)
int x;
int y;
int linkdx;
int slinkdx;
int slinkdy;
int linkdy;
int links[NOROW][NOCOL];
-
 nt i;
 nt j;
 nt x0;
 nt y0;

for (i=0; i ¡ NOROW; ++i)
  -

  for (j=0; j ¡ NOCOL; ++j)
    -
      if (sprint == 0)
       printf("testing %d %d link %d "n", i,j, links[i][j]);

       if ( links[i][j]¿0 )
        -
          if ((i%2 == 0) && (j%2 == 0))
            -
            x0 = x + (j/2)*linkdx +(j/2)*slinkdx ;
            y0 = y + (i/2)*linkdy +(i/2)*slinkdy ;
            place("ldlink", x0, y0);
              " /* end of i%2 == 0 */
          else if ((i%2 == 0) && (j%2 != 0))
            -
             x0 = x + ((j+1)/2)*linkdx +((j-1)/2)*slinkdx ;
             y0 = y + (i/2)*linkdy + (i/2)*slinkdy ;
             place("ldlink,MX", x0,y0);
            " /* end of else */
          else if ((i%2 != 0) && (j%2 == 0))
            -
              x0 = x + (j/2)*linkdx + (j/2)*slinkdx ;
              y0 = y + ((i+1)/2)*linkdy +((i-1)/2)*slinkdy ;
              place("ldlink,MY", x0,y0);
            "
           else
            -
              x0 = x + ((j+1)/2)*linkdx + ((j-1)/2)*slinkdx ;
              y0 = y + ((i+1)/2)*linkdy + ((i-1)/2)*slinkdy ;
              place("ldlink,MX,MY", x0,y0);
            "
        " /* end of links(i,j)¿0 */
    " /* end of j loop */
  " /* end of i loop */
 /* end of  rlcross */


void ccell(x,y, linkdx, slinkdx, linkdy,slinkdy,lef,rig,bot,top)

int          x;  /* x start of cell */
int          y;  /* y start of link */
int      linkdx;     /* x spacing between links */
int      slinkdx;
int      linkdy;     /* y spacing between links */
int      slinkdy;
int      lef;
int      rig;
int      bot;
int      top;

-
static in  lestubs[6][16] = -
                -   0, 1300, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1",
    -   0, 1705, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0",
    -   0, 2175, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
                -   0, 2945, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1",
                -   0, 3515, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0",
              -  0, 4065, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0", ";
              /* matrix of cell stubs */
static int ristubs[2][16] = -
              -7000, 1300, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0",
             -7000, 1760, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0", ";
              /* matrix of cell stubs */
static int bostubs[14][10] = -
              -300,   0, 0, 0, 1, 0, 1, 0, 1, 0",
    -1000,  0, 0, 0, 1, 0, 1, 0, 1, 0",
              -1400,  0, 1, 0, 1, 0, 1, 0, 0, 0",
              -2100,  0, 1, 0, 1, 0, 1, 0, 0, 0",
              -2500,  0, 0, 0, 1, 0, 1, 0, 1, 0",
              -3200,  0, 0, 0, 1, 0, 1, 0, 1, 0",
              -3600,  0, 1, 0, 1, 0, 1, 0, 0, 0",
              -4400,  0, 1, 0, 1, 0, 1, 0, 0, 0",
              -4825,  0, 0, 1, 0, 1, 0, 1, 0, 0",
              -5225,  0, 1, 0, 1, 0, 1, 0, 0, 0",
              -5625,  0, 0, 0, 1, 0, 1, 0, 1, 0",
              -6025,  0, 1, 0, 1, 0, 1, 0, 0, 0",
              -6425,  0, 1, 0, 0, 0, 0, 0, 0, 1",
              -6825,  0, 1, 0, 0, 0, 0, 0, 0, 1", ";
              /* matrix of cell  tubs */
                                                    static int tostubs[4][1
                -0,  4500,  0,  0, 0, 0, 0, 0, 0, 0",
                -0,  4500,  0,  0, 0, 0, 0, 0, 0, 0",
                -0,  4500,  0,  0, 0, 0, 0, 0, 0, 0",
                -0,  4500,  0,  0, 0, 0, 0, 0, 0, 0", ";
              /* matrix of cell  tubs */


int      i; /* counter */
int      j; /* counter */
int      x00; /* link x location */
int      y00; /* link y location */
int      x01; /* link x location */
int      y01; /* link y location */
int      x1; /* x coordinate */
int      y1; /* y coordinate */
int      x2; /* x coordinate */
int      y2; /* y coordinate */
int      x3; /* x coordinate */
int      y3; /* y coordinate */
int      x4; /* x coordinate */
int      y4; /* y coordinate */
int      xl1; /* x coordinate */
int      yl1; /* y coordinate */
int      xr2; /* x coordinate */
int      yr2; /* y coordinate */
int      xb3; /* x coordinate */
int      yb3; /* y coordinate */
int      xt4; /* x coordinate */
int      yt4; /* y coordinate */
int      dx1; /* x displacement for stubs */
int      dx2;
int      dx3;
int      dx4;
int      dy1; /* y displacement for stubs */
int      dy2;
int      dy3;
int      dy4;
int      colline;
int      rowline;
int      trackl;
int      trackr;
int      trackb;
int      trackt;

colline = ntrack[1][0]+2;
rowline = ntrack[1][1]+2;

/*left stubs */

 for (i = 0; i¡STUBL; i++)
  -
   for (j = 2; j¡(ntrack[1][0]+2); j++)
   -
     trackl = j-2;
     if ( lestubs[i][j] ¿ 0 )
     -
       y1 = y + lestubs[i][1] ;
       if (lef==1)
       -
        if (ntrack[1][0]%2 == 0)
       x00 = -(ntrack[1][0]/2)*linkdx - ((ntrack[1][0]-2)/2)*slinkdx
        else if (ntrack[1][0]%2 != 0 )
          x00 =  -((ntrack[1][0]-1)/2)*(linkdx+slinkdx);
       "
       if (trackl%2 == 0)
         -
         dx1 = (trackl/2)*linkdx + (trackl/2)*slinkdx  ;
          x1 = x + x00 + dx1 + lestubs[i][0] + dtrack[3][0] ;
          place("ldlink",x1,y1);
         "
       else if (trackl%2 != 0 )
         -
       dx1 =  ((trackl-1)/2)*(linkdx+slinkdx);
       x1 = x + x00 + dx1 + lestubs[i][0] + dtrack[3][0];
       place("ldlink,MY",x1,y1);
         "
       "
       else if (lef==0)
        -
         dx1 = trackl*linkdx ;
         x00 = - (ntrack[1][0]-1)*linkdx;
```

```
        x1 = x + x00 + dx1 + lestubs[i][0] + dtrack[3][0];
        place("ldlink,MX",x1,y1);


        layer(metal2);
        width(linewidth);
        xl1 = x + lestubs[i][0];
        yl1 = y + lestubs[i][1];
        wire( xl1, yl1, x1, y1, END);
    }
  }
/*right stubs */

  for (i = 0; i;STUBR; i++)
  {
   for (j = 2; j;colline; j++)
   {
     trackr = j-2;
     if (ristubs[i][j]¿0)
     {
       y2 = y + ristubs[i][1] ;
       if (rig==1)
       {
        x01 = 0;

        if (trackr%2 == 0)
        {
          dx2 = (trackr/2)*linkdx + (trackr/2)*slinkdx  ;
          x2 = x + x01 + dx2 + ristubs[i][0] + dtrack[2][0] ;
          place("ldlink",x2,y2);
        }
        else if (trackr%2 != 0 )
        {
          dx2 =  ((trackr-1)/2)*(linkdx+slinkdx);
          x2 = x + x01 + dx2 + ristubs[i][0] + dtrack[2][0];
          place("ldlink,MY",x2,y2);
        }

        else if (rig==0)
        {
          dx2 = trackl*linkdx ;
          x01 = 0;
          x2 = x + x01 + dx2 + ristubs[i][0] + dtrack[2][0];
          place("ldlink",x2,y2);


          layer(metal2);
          width(linewidth);
          xr2 = x + ristubs[i][0];
          yr2 = y + ristubs[i][1];
          wire( xr2, yr2, x2, y2, END);
        }
      }
    }

/* bottom stubs */
   for (i = 0; i;STUBB; i++)
   {
    for (j = 2; j;rowline; j++)

     trackb = j-2;
     if (bostubs[i][j]¿0)
     {
       x3 = x + bostubs[i][0] ;

       if (bot==1)
       {
        if (ntrack[1][1]%2 == 0)
       y00 = -(ntrack[1][1]/2)*linkdy - ((ntrack[1][1]-2)/2)*slinkdy  ;
        else if (ntrack[1][1]%2 != 0 )
          y00 =  -((ntrack[1][1]-1)/2)*(linkdy+slinkdy);
        if (trackb%2 == 0)
        {
          dy3 = (trackb/2)*linkdy + (trackb/2)*slinkdy  ;
          y3 = y + y00 + dy3 + bostubs[i][1] + dtrack[1][1] ;
          place("ldlink,MX",x3,y3);
        }
        else if (trackb%2 != 0 )
        {
          dy3 =  ((trackb-1)/2)*(linkdy+slinkdy);
          y3 = y + y00 + dy3 + bostubs[i][1] + dtrack[1][1];
          place("ldlink",x3,y3);
        }
       else if (bot==0)
        {
          dy3 = trackb*linkdy ;
          y00 =  -(ntrack[1][1]-1)*linkdy;
          y3 = y + y00 + dy3 + bostubs[i][1] + dtrack[1][1];
          place("ldlink",x3,y3);

        layer(metal);
        width(linewidth);
        xb3 = x + bostubs[i][0];
        yb3 = y + bostubs[i][1];
        wire( xb3, yb3, x3, y3, END);



/* top stubs */
  for (i = 0; i;STUBT; i++)
  {
   for (j = 2; j;rowline; j++)
   {
    trackt = j-2;
     if (tostubs[i][j]¿0)

       x4 = x + tostubs[i][0] ;

       if (top==1)
       {

         y01 =  0;
        if (trackb%2 == 0)
        {
         dy4 = (trackt/2)*linkdy + (trackt/2)*slinkdy  ;
          y4 = y + y01 + dy4 + tostubs[i][1] + dtrack[0][1] ;
          place("ldlink",x4,y4);
        }
       else if (trackt%2 != 0 )
        {
         dy4 =  ((trackt-1)/2)*(linkdy+slinkdy);
         y4 = y + y01 + dy4 + tostubs[i][1] + dtrack[0][1];
         place("ldlink,MX",x4,y4);
        }

       else if (top==0)
        {
         dy4 = trackt*linkdy ;
         y01 = 0;
         y4 = y + y01 + dy4 + tostubs[i][1] + dtrack[0][1];
         place("ldlink",x4,y4);


        layer(metal);
         width(linewidth);
         xt4 = x + tostubs[i][0];
         yt4 = y + tostubs[i][1];
         wire( xt4, yt4, x4, y4, END);
      }
    }
  }

/* area caculation, assuming track numbers of column of left
,incolumn, */
/* right are the same ; so is the row */

void area(y0, yend, linkdx, slinkdx, linkdy, slinkdy)
int y0, yend;
int linkdx, slinkdx, linkdy, slinkdy;
{
int rarea, carea;
int tarea;
int sarea, aarea;
float ratio;
if (dou[1][0]==0)
 carea = (ntrack[1][0] - 1)*linkdx*(yend - y0)*(ncols + 1);
else
 {
  if (ntrack[1][0]%2 == 0)
  carea = (ntrack[1][0]/2*linkdx + (ntrack[1][0]/2 - 1)*
        slinkdx)*(yend - y0)*(ncols + 1);
  else if  (ntrack[1][0]%2 != 0)
  carea = (ntrack[1][0]/2 - 1)*(linkdx + slinkdx)*(yend -
        y0)*(ncols + 1);


 if (dou[1][1] == 0)
  rarea = (ntrack[1][1] - 1)*linkdy*(xcell + dtrack[2][0]
        -dtrack[3][0]) * ncols*(nrows+1);
 else
 {
  if (ntrack[1][1]%2 == 0)
   rarea = (ntrack[1][1]/2*linkdy + (ntrack[1][0]/2 - 1)*
         slinkdy)*(xcell + dtrack[2][0] -dtrack[3][0])
         * ncols*(nrows+1);
  else if (ntrack[1][1]%2 != 0)
   rarea = (ntrack[1][1]/2 - 1)*(linkdy + slinkdy)*(xcell
         + dtrack[2][0] -dtrack[3][0]) * ncols*(nrows+1);
 }
 tarea = (rarea + carea)/64; /* 1 square um = 64 square bif */
 sarea = nrows*ncols*(xcell+dtrack[2][0]-dtrack[3][0])*(ycell+
        dtrack[0][1-dtrack[1][1])/64;
 aarea = tarea + sarea;
 ratio = (float)tarea/(float)aarea;
```

```
  printf(""nThe  area of tracks = %d square um"n", tarea);
  printf(""n(area of tracks)/(total area) = %6.4f "n", ratio);

/* end of area */

/* caculation of resistance */
float rres(x0, xend)
int x0, xend;

float rresist;
rresist = resist*(xend - x0);
return(rresist);


float cres(y0,yend)
int y0,yend;

float cresist;
cresist = resist*(yend - y0);
return(cresist);


/* end of resistance */

/* link number caculation*/
/* counter for matrix */

int countx(nx,ny,l)   /* counter for x axis of matrix l */
int  nx,ny,l[][50];

int i,x;

x = 0;
for (i=0;i¡nx; i++)

 if ((l[i][ny]¿0) && (l[i][ny]¡=50))
 x = x++;

return(x);


int county(nx,ny,l)   /* counter for y axis of matrix l */
int  nx,ny,l[][50];

int i,y;
y = 0;
for (i=0; i¡ny; i++)

  if ((l[nx][i]¿0) && (l[nx][i]¡=50))
 y = y++;

return(y);


/*first consider column channel, left, middle, right */
/* for each channel, consider cross,
left stub, right stub */

void linknumc(x,y,yl,yr,lcell,ltbedge,lrledge,lcorner,
          lestubs,ristubs,y0,yend,linkdy)
int x, y;
int yl, yr;
int lcell[][50], ltbedge[][50], lrledge[][50], lcorner[][50];
int lestubs[][50], ristubs[][50];
int y0,yend;
int linkdy;

int lef, mid, rig;
int i, j;
float clef, cmid, crig;
lef = 0;
mid = 0;
rig = 0;
/* lef channel */
printf(""n"n The Number of Link in Left Column Channel "n");
for (i=0; i¡x; ++i)

  lef = lef + 2*county(i,y,lcorner);
  lef = lef + (nrows-1)*county(i,y,lrledge);
  lef = lef + nrows*county(i+2,yl,lestubs);
  clef = lef * capacity + (yend-y0-lef*linkdy) * linec;

  printf(""n Track No = %2d, Number of Links = %2d,
        R = %7.1f ohm, C = %7.1f fF", i,lef,cres(y0,yend),clef);
  lef = 0;

/* middle channel */
printf(""n"n The Number of Link in Middle Column Channel "n");
for (i=0; i¡x; ++i)

  mid = mid + 2*county(i,y,ltbedge);
  mid = mid + (nrows-1)*county(i,y,lcell);
  mid = mid + nrows*county(i+2,yl,lestubs);
  mid = mid + nrows*county(i+2,yr,ristubs);
  cmid = mid * capacity + (yend-y0-mid*linkdy) * linec;
  printf(""n Track No = %2d, Number of Links = %2d,
        R = %7.1f ohm, C = %7.1f fF", i,mid,cres(y0,yend),cmid);
  mid = 0;

/* right channel */
printf(""n"n The Number of Link in Right Column Channel "n");
for (i=0; i¡x; ++i)

  rig = rig + 2*county(i,y,lcorner);
  rig = rig + (nrows-1)*county(i,y,lrledge);
  rig = rig + nrows*county(i+2,yr,ristubs);
  crig = rig * capacity + (yend-y0-rig*linkdy) * linec;
  printf(""n Track No = %2d, Number of Links = %2d,
        R = %7.1f ohm, C = %7.1f fF", i,rig,cres(y0,yend),crig);
  rig = 0;


/*then  consider row channel, top, middle, bottom */
/* for each channel, consider cross, top stub, bottom stub */
void linknumr(x,y,xt,xb,lcell,ltbedge,lrledge,lcorner,
          tostubs,bostubs,x0,xend,linkdx)
int x, y;
int xt, xb;
int lcell[][50], ltbedge[][50], lrledge[][50], lcorner[][50];
int tostubs[][50], bostubs[][50];
int x0,xend;
int linkdx;

int top, mid, bot;
int i, j;
float ctop, cmid, cbot;

top = 0;
mid = 0;
bot = 0;
/* top channel */
printf(""n"n The Number of Link in Top Row Channel "n");
for (i=0; i¡y; ++i)

  top = top + 2*countx(x,i,lcorner);
  top = top + (nrows-1)*countx(x,i,ltbedge);
  top = top + nrows*county(i+2,xt,tostubs);
  ctop = top*capacity + (xend-x0-top*linkdx)*linec;
  printf(""n Track No = %2d, Number of links = %2d,
        R = %7.1f ohm, C = %7.1f fF", i,top,rres(x0,xend),ctop);
  top = 0;

/* middle channel */
printf(""n"n The Number of Link in Middle Row Channel "n");
for (i=0; i¡x; ++i)

  mid = mid + 2*countx(x,i,ltbedge);
  mid = mid + (nrows-1)*countx(x,i,lcell);
  mid = mid + nrows*county(i+2,xt,tostubs);
  mid = mid + nrows*county(i+2,xb,bostubs);
  cmid = mid*capacity + (xend-x0-mid*linkdx)*linec;
 printf(""n Track No = %2d, Number of Links = %2d, R = %7.1f ohm
        C = %7.1f fF", i,mid,rres(x0,xend),cmid);
  mid = 0;

/* bottom channel */
  printf(""n"n The Number of Link in Bottom Row Channel "n");
for (i=0; i¡x; ++i)

  bot = bot + 2*countx(x,i,lcorner);
  bot = bot + (nrows-1)*countx(x,i,ltbedge);
  bot = bot + nrows*county(i+2,xb,bostubs);
  cbot = bot*capacity + (xend-x0-bot*linkdx)*linec;
  printf(""n  Track No = %2d, Number of links = %2d,
        R = %7.1f ohm, C = %7.1f fF ", i,bot,rres(x0,xend),cbot);
  bot = 0;

printf(""n");


/* formc function is to form cross array */

formc(signal, trackr, trackc, arow, acol, across)
int signal, trackr, trackc;
int acol[][NOCOL], arow[][NOROW], across[NOROW][NOCOL];

int i, j, k;

 for(i=0; i¡trackr; i++)

   for (j=0; j¡trackc; j++)

     for (k=0; k¡signal; k++)

       if (across[i][j] == 0)
         across[i][j] = acol[k][j]*arow[k][i] ;
```

```
/* form left stub */
forml(signal, trackr, trackc, arow, acol, across)
int signal, trackr, trackc;
int  arow[][STUBL],acol[][NOCOL], across[STUBL][NOCOS];

int i, j, k, l;

 for(i=0; i<trackr; i++)

    for (j=0; j<trackc; j++)

      for (k=0; k<signal; k++)

        l = j + 2;
        if (across[i][l] == 0)
          across[i][l] = acol[k][j]*arow[k][i] ;



/* form right stub */
formr(signal, trackr, trackc, arow, acol, across)
int signal, trackr, trackc;
int  arow[][STUBR],acol[][NOCOL], across[STUBR][NOCOS];

int i, j, k, l;

 for(i=0; i<trackr; i++)

    for (j=0; j<trackc; j++)

      for (k=0; k<signal; k++)

        l = j + 2;
        if (across[i][l] == 0)
          across[i][l] = acol[k][j]*arow[k][i] ;



/*form bottom stub */
formb(signal, trackr, trackc, arow, acol, across)
int signal, trackr, trackc;
int  arow[][STUBB],acol[][NOROW], across[STUBB][NOROS];

int i, j, k, l;

 for(i=0; i<trackr; i++)

    for (j=0; j<trackc; j++)

      for (k=0; k<signal; k++)

        l = j + 2;
        if (across[i][l] == 0)
          across[i][l] = acol[k][j]*arow[k][i] ;



/* form top stub */
formt(signal, trackr, trackc, arow, acol, across)
int signal, trackr, trackc;
int  arow[][STUBT],acol[][NOROW], across[STUBT][NOROS];

int i, j, k, l;

 for(i=0; i<trackr; i++)

    for (j=0; j<trackc; j++)

      for (k=0; k<signal; k++)

        l = j + 2;
        if (across[i][l] == 0)
          across[i][l] = acol[k][j]*arow[k][i] ;


```

REFERENCES

[1] R. K. Cavin, J. R. Key, **'Wafer Scale Integration: An Assessment'**, *Semiconductor Research Cooperation Workshop*, pp89-94,Sept., 1984.

[2] G. H. Chapman, **'Laser Processes for Defect Correction in Large Area VLSI Systems'**, *International Workshop on Defect and Fault Tolerance in VLSi systems.*, pp106-113, 1994.

[3] E. E. Swartzlander, **'Wafer Scale Integration'**, Kluwer Academic Publishers, 1989.

[4] E. A. Sack, R. C. Lyman and G. Y. Chang, **'Evolution of the Concept of a Computer on a slice'**, *Proceedings of the IEEE* , Vol. 52, pp1713-1720, 1964.

[5] R. L. Petritz, **'Current Status of Large Scale Integration'**, *IEEE Iournal of Solid state Circuits* , Vol. 4 , pp130-146, 1967.

[6] D. F. Calhoun, **'The Pad Relocation Technique for Interconnection LSI Arrays of Imperfect Yield'**, *Fall Joint Computer Conference Proceedings* , Vol. 35 , pp99-109, 1969.

[7] Y. Hsia, G. C. Chang, and F. D. Erwin, **'Adaptive Wafer Scale Integration'**, *Proceedings of the IEEE 1979 International Conference on Solid State Devices* , Vol. 19 , pp193-202, 1979.

[8] D. L. Pelzer, **'Wafer Scale Integration: The Limits of VLSI'**, *VLSI Design*, pp43-47, Sept., 1983.

[9] R. Varshney, **'Wafer Level Integration technique'**, *U. S. Patent 4703436* , issued Oct. 27, 1987.

[10] S. S. Cohen, G. H. Chapman, **'Laser Beam Processing and Wafer-Scale Integration'**, *VLSI Electronics: Microstructure Science* , Vol. 21, chapter 2, pp19-111 1989.

[11] D. C. Opferman, N. T. Tsao-Wu, **'On a Class of Rearrangeable Switching Networks'**, *The Bell System Technical Journal*, pp1579-1600, May-June 1971.

[12] A. J.Rushton, C. R. Jesshope, **'The Reconfigurable Processor Array'**, in *Wafer Scale Integration* ed. C. Jesshope and W. Moore, Adam Hilger Press, Bristol, pp149-158, 1986.

[13] R.Frankel, J. J. Hunt, M. V. Alstyne and G. Young, **'SLASH – An RVLSI CAD System'**, *Proc. IEEE 1989 International Conference on Wafer Scale Integration*, 1989.

[14] S. K. Tewksbury, **'Wafer-level Integrated Systems : Implementation Issues'**, Kluwer Acadamic Publishers, 1989.

[15] W. Chen, P. B. Denyer, J Mavor and D. Renshaw, **'Fault-Tolerant Wafer Scale Architecture Using Large Crossbar'**, in *Wafer Scale Integration* ed. C. Jesshope and W. Moore, Adam Hilger Press, Bristol, pp112-124, 1986.

[16] M. Slimane-Kadi, A. Boubekeur and G. Saucier, **'Interconnection Networks with Fault-Tolerance Properties'**, *Proc. IEEE 1993 International Conference on Wafer Scale Integration*, , 1993, San Franscisco, pp213-222, 1993.

[17] C. L. Wu, T. Y. Feng, **'Tutorial: Interconnection Networks for Parallel and Distributed Processing'**, IEEE Computer Society Press, 1984.

[18] H. J. Siegel, **'Interconnection Networks For Large-Scale Parallel Processing'**, McGraw-Hill Publishing Company, 2nd edition, 1989.

[19] C. D. Thompson, **'Generalized Connection Networks for Parallel Processor Intercommunication'**, *IEEE Trans. on Computers*, C-27, pp1119-1125, Dec. 1978.

[20] C. L. Wu, T. Y. Feng, **'On a Class of Multistage Interconnection Networks'**, *IEEE Trans. on Computers*, Vol. C-29, No.8, pp694-702, Aug. 1980.

[21] A. Varma, S. Chalasani, **'Fault-Tolerance Analysis of One-Sided Crosspoint Switching Networks'**, *IEEE Trans. on Computers*, Vol.41, No.2, pp143-158, Feb. 1992.

[22] T. Y. Feng, **'Data Manipulating Functions in Parallel Processors and Their Implementations'**, *IEEE Trans. on Computers*, Vol. C-23, No.3, pp309-318, Mar. 1974.

[23] D. H. Lawrie, **'Access and Aligment in an Array Processor'**, *IEEE Trans. on Computers*, Vol. C-24, No.12, pp1145-1155, Dec. 1975.

[24] T. Lang, **'Interconnections Between Processors and Memory Modules Using the Shuffle-Exchange Network'**, *IEEE Trans. on Computers*, Vol. C-25, No.5, pp496-503, May 1976.

[25] P. W. Wyatt, J. I. Raffel, **'Restructurable VLSI – A Demonstrated Wafer Scale Technology'**, *Proc. IEEE 1989 International Conference on Wafer Scale Integration*, pp186-194, 1989.

[26] G. H.Chapman, R.F. Hobson, **'Algorithmic Bus and Circuit Layout for Wafer-Scale Integration and Multichip Modules'**, Proc. Fifth Int. Conf. on Wafer Scale Integration, pp137-146, 1993.

[27] G.H.Chapman, L.Carr, M.J.Syrzycki and B.Dufort. **'Test Vehicle for a Wafer-Scale Thermal Pixel Scene Simulator'**, *IEEE Trans. on Components, Hybrids and Manuf. Technology.*, pp334-341, 1994.

[28] **'Operator's Manual: The Coherent INNOVA 300 Series Ion Laser'**, Coherent Coperation, 1991.

[29] **'IDAC Intelligent Digital Axis Controller Programming and Hardware Manual'**, Anorad Corporation, 1992.

[30] J. Gecsei, **'Interconnection Networks from Three-State Cells'**, *IEEE Trans. on Computers*, Vol. C 26, No.8, pp705-711, Aug. 1977.

[31] H. S. Stone, **'Parallel Processing with the Perfect Shuffle'**, *IEEE Trans. on Computers*, Vol. C-20, No.2, pp153-161, Feb. 1971.

[32] M. K. Franklin, **'VLSI Performance of Banyan and Crossbar Communication Networks'**, *IEEE Trans. on Computers*, Vol. C-30, No.4, pp776-784, Apr. 1981.

[33] J. Gecsei, **'Interconnection Networks from Three-State Cells'**, *IEEE Trans. on Computers*, Vol. C 26, No.8, pp705-711, Aug. 1977.

[34] M. Annaratone, **'Digital CMOS Circuit Design'**, Kluwer Academic Publishers, 1986.

[35] G. H. Chapman, K. Fang, **'Comparison of Laser Link crossbar and Omega Network Switching for Wafer-Scale Integration Defect Avoidance'**, *Proceedings IEEE Int. Conf. on WSI*, pp352-361, 1994.

[36] J. Buchanan, **'BiCMOS/CMOS System Design'**, McGraw-Hill Inc., 1991.

[37] G. Chevalier and G. Saucier, **'A Programmable Switch Matrix For The Wafer Scale Integration of a Processor Array'**, in *Wafer Scale Integration* ed. C. Jesshope and W. Moore, Adam Hilger Press, Bristol, pp93-100, 1986.

[38] D. V. Heinbuch , **'CMOS3 Cell Library'**, Addison-Wesley Publishing Company, 1988.

[39] R.F. Hobson, D. Smith, **'CDL: A Tool for Algorithmic VLSI Design '**, Dept. of Computer Science, Simon Fraser University, 1994.