1984

System integration & computer graphics aids for VLSI MOS circuits layout design.

Mahmoud Ahmad. Abu-Nasr
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30. Please read the authorization forms which accompany this thesis.

THIS DISSERTATION HAS BEEN MICROFILMED EXACTLY AS RECEIVED

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30. Veuillez prendre connaissance des formules d'autorisation qui accompagnent cette thèse.

LA THÈSE A ÉTÉ MICROFILMÉE TELLE QUE NOUS L'AVONS ŒUVRE
SYSTEM INTEGRATION

&

COMPUTER GRAPHICS AIDS FOR VLSI MOS CIRCUITS

LAYOUT DESIGN

by

Mahmoud Ahmad Abu-Nasr

A thesis

presented to the University of Windsor in partial fulfillment of the requirements for the degree of

MASTEr OF APPLIED SCIENCE in

Electrical Engineering Department

Windsor, Ontario, 1984

(C) Mahmoud Ahmad Abu-Nasr, 1984
ABSTRACT

In this thesis a prototype CAD facility, supporting a standard cell design approach, for the layout of VLSI MOS circuits, was implemented as an investigation of the capabilities of the system existing in the Electrical Engineering Department, University of Windsor.

The hardware elements used in this implementation are: the SEL 32/27 mini-computer, the Aydin graphics computer and display monitor, and a touch screen. These elements are also a part of an image processing station containing an NTT convolver and a colour plotter.

A general scheme for system integration, driving the system by a series of menus, and incorporating both the image processing, and the VLSI layout design applications with a unified user friendly method of access was also developed through this work.
ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisor, Dr. G.A. Jullien, and to Dr. W.C. Miller, for many valuable discussions and constructive criticism. I am also thankful to Dr. M. Sid-Ahmed, for his constant encouragement throughout the study period. My gratitude is extended to all my professors in the Electrical Engineering Department, University of Windsor, Dr. H. Shridhar, Dr. J. J. Soltis, and Dr. M. Ahmadi.

I would like also to express my sincerest love, appreciation and gratitude to my parents and my brother, without their encouragement, inspiration and love, this work would not have started.
# CONTENTS

ABSTRACT ........................................ iv

ACKNOWLEDGEMENTS .................................... v

Chapter page

I. INTRODUCTION ................................ 1

THE NEED FOR VLSI DESIGN TOOLS TO COPE WITH THE
ADVANCES IN VLSI MANUFACTURING TECHNOLOGY ........ 1

CAD DESIGN STATIONS ................................ 2

FACILITIES AVAILABLE IN THE ELECTRICAL
ENGINEERING DEPARTMENT ......................... 3

THESIS OBJECTIVES ................................ 4

THESIS ORGANIZATION .............................. 5

II. VLSI, AND COMPUTER AIDED DESIGN .................. 7

Very Large Scale Integration, (VLSI) ................. 7

Integration Size .................................. 7

Fabrication of IC Packages ....................... 9

The Entire Sequence of Design and Fabrication
of a VLSI Chip .................................. 13

Recent VLSI Related Techniques ................... 17

COMPUTER AIDED DESIGN OF VLSI SYSTEMS .......... 19

Objectives of CAD Programs ...................... 20

Use of CAD in the Different Design Stages of
a VLSI System .................................. 21

LAYOUT DESIGN METHODOLOGIES FOR VLSI .......... 25

FUTURE VLSI ISSUES ............................... 28

Future Large System Design ...................... 29

Future VLSI Manufacturing ...................... 30

III. SYSTEM OVERVIEW ................................ 31

OVERVIEW OF THE SYSTEM HARDWARE ................. 31

SSL 32/27 Mini-Computer Architecture .............. 33

Aycon 16 Display Computer Architecture .......... 35

The Touch Screen .................................. 38

TRILOG 100 COLOUR PLOTTER ....................... 40

OVERVIEW OF THE SYSTEM SOFTWARE ................. 43

SSL SOFTWARE .................................... 43

Aydin System Software .......................... 46
IV. SYSTEM INTEGRATION AND COMPUTER GRAPHICS AIDS FOR VLSI MOS CIRCUITS LAYOUT DESIGN

Menu System Hierarchy ........................................... 49
Developing of the Menu System ................................ 56
Investigation of the System Capability for VLSI Layout Design .................................................. 60
Standard Cell Layout Design Approach ......................... 61
Layout of Basic Cells ............................................. 62
Software Implementation of the Basic Cells ................... 66
Software Implementation of the Wiring Technique ........... 68
Design of the Interactive Editor ................................ 69
Layout Capturing .................................................... 71
Software for Implementing the Interactive Editor and the Storage Method ........................................ 72
Masking Software .................................................... 73
The Masking Programs ............................................. 73
Design of the VLSI Menus and Connection to the Main Menu ......................................................... 75
Testing of the Designed System ................................ 79

V. SUMMARY AND CONCLUSIONS .................................. 91

APPENDIX .............................................................................

A. PROGRAMS LISTING .................................................. 93

REFERENCES .............................................................. 128

VITA AUCTORIS .......................................................... 130
LIST OF TABLES

Table                                            page
1. Integration Size                              8
2. Information Needed by Aydin to Construct a Red Rectangle in a NAND Gate. 67
3. Colours Assigned to the Different Layers in the Layout. 68
4. Interactive Sequence for Locating a NAND Gate on the Screen Using the Edit Menu. 71
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Patterning of Silicon Dioxide Layer</td>
<td>12</td>
</tr>
<tr>
<td>2.</td>
<td>The nMOS Fabrication Process</td>
<td>14</td>
</tr>
<tr>
<td>3.</td>
<td>Gate Array</td>
<td>27</td>
</tr>
<tr>
<td>4.</td>
<td>System Organization</td>
<td>32</td>
</tr>
<tr>
<td>5.</td>
<td>SEL Computer Architecture</td>
<td>34</td>
</tr>
<tr>
<td>6.</td>
<td>Aydin Computer Architecture</td>
<td>36</td>
</tr>
<tr>
<td>7.</td>
<td>Generalized Program Structure for the 5216 Display Computer</td>
<td>48</td>
</tr>
<tr>
<td>8.</td>
<td>Menu System Hierarchy</td>
<td>50</td>
</tr>
<tr>
<td>9.</td>
<td>The Main Menu</td>
<td>51</td>
</tr>
<tr>
<td>10.</td>
<td>The Display Menu</td>
<td>52</td>
</tr>
<tr>
<td>11.</td>
<td>The Edit Submenu</td>
<td>53</td>
</tr>
<tr>
<td>12.</td>
<td>A Piston Head Image Positioned and Coloured Using the Edit Menu</td>
<td>54</td>
</tr>
<tr>
<td>13.</td>
<td>Fortran Decision Structure for PAGE 1</td>
<td>58</td>
</tr>
<tr>
<td>14.</td>
<td>Algorithm of an MPX-32 Command File for Page 1</td>
<td>59</td>
</tr>
<tr>
<td>15.</td>
<td>The Layout of a NAND Gate as Realized on Aydin</td>
<td>64</td>
</tr>
<tr>
<td>16.</td>
<td>The Layout of a Not Gate as Realized on Aydin</td>
<td>65</td>
</tr>
<tr>
<td>17.</td>
<td>VLSI Submenu Number 1.</td>
<td>76</td>
</tr>
<tr>
<td>18.</td>
<td>VLSI Submenu Number 2.</td>
<td>77</td>
</tr>
<tr>
<td>19.</td>
<td>VLSI Submenu Number 3.</td>
<td>78</td>
</tr>
<tr>
<td>20.</td>
<td>Logic Diagram of an XOR Gate</td>
<td>80</td>
</tr>
</tbody>
</table>
21. Layout of an XOR Gate. .................................. 81
22. Logic Diagram of a Half-Adder Circuit ................. 82
23. Layout of a Half-Adder Circuit. ....................... 83
    Borders. ............................................. 84
27. Half-Adder Circuit. Mask 4: Contact Windows. ........ 87
29. Logic Diagram of a Binary Decoder. .................... 89
30. Layout of a Binary Decoder Circuit. ..................... 90
Chapter I

INTRODUCTION

1.1 THE NEED FOR VLSI DESIGN TOOLS TO CORRESPOND WITH THE ADVANCES IN VLSI MANUFACTURING TECHNOLOGY

In recent years, LSI and VLSI chips have been incorporated into an increasingly wide spectrum of products, ranging from watches, toys, video games, home appliances and personal computers to large computers.

With the advent of new manufacturing techniques, like direct step on wafer photolithography, electron beam lithography, advanced etching processes with plasma or reactive ions, industry is capable of fabricating chips containing 500,000 transistors. The line width of the minimum feature on these chips is 2-micrometers. It is predicted that the lateral dimension limit of 0.25 micrometer imposed by physical laws will probably be reached by the early 1990's [12]. Even at the current level of complexity (number of components per chip), system design provides many challenges for several reasons:

1. The data needed to specify the current VLSI system is substantially greater than for small systems. This data must be managed carefully so that it is easily accessed, updated and kept consistent.
2. There is a substantial interdependency of subsystem parts. When changing parts of the system care must be taken that the changes do not adversely affect other parts of the system.

3. Design of a large system becomes a group project. This causes communication problems (misunderstandings) which must be solved in a productive manner.

To manage the complexity, two approaches are currently in use: design automation (DA), and computer aided design (CAD). The usual model of DA is that the computer makes most of the design decisions given a high level specification from a human. In the Cad approach, the computer and the human each perform those tasks best suited to their abilities.

1.2 **CAD Design Stations**

The design station concept is not a new idea. For years users of CAD applications packages have dreamed of systems with one common input driving multiple CAD functions. Several systems have been built around this concept using elaborate databases, and manifold applications codes,
providing in varying degrees, a semblance of a unified system. However most of these systems are large, batch oriented systems with non-interactive input processing and require experts to use them. The casual users of the current CAD stations are electronic engineers, designers and technicians, who know engineering but have neither the time nor the inclination to become CAD experts. Therefore the stations should be easy to learn and easy to use. Interactive graphics techniques are employed extensively to capture the design input rather than the tedious and error prone processes of 'input coding or transcription from previously formed drawings.

1.3 FACILITIES AVAILABLE IN THE ELECTRICAL ENGINEERING DEPARTMENT

The main computational element is the SEL 32/27 mini-computer. Computer peripherals include: the Aydin & Controls display computer and colour monitor, Trilog 100 colour plotter and a TSD Display Canada touch screen. For image processing applications a high speed convolutional filter is used for preprocessing of images and is interfaced to the SEL computer via a high-speed data interface. The SEL computer has also a 32 M bytes hard disk drive and a 1.2 M byte floppy diskette drive. A more elaborate description of this system is given in chapter 3 of this thesis.
1.4 **THESIS OBJECTIVES**

The thesis objectives are:

1. To integrate the facilities mentioned above into one system.
2. To unify the user's access to the various computer aids and facilities.
3. To investigate the capability of our system for VLSI layout design.

These objectives were achieved as follows:

1. Developing software for driving the system by a series of menus creating a user friendly interface to the SEL computer and the Aydin display and the colour plotter. The system is expandable and can accommodate future expansions of the system.
2. Developing an interactive scheme for the layout design of VLSI MOS circuits, by which the user can place gates and wires on the monitor screen, and delete unwanted items, by selecting commands from a displayed menu and positioning the cursor on the desired point or points. The designed layout is captured within the computer memory as it is entered on the screen, and afterwards in a file on the hard disk.
3. Developing programs for NAND and NOT gates using MOS technology design rules, as the first elements of a
library of components to be expanded in the future. In developing these programs, the dimensions were expressed as ratios of lengths to the fundamental resolution of the fabrication process, to allow for the rapidly changing technology, and also to enable the display of the gates with different scales.

4. Developing programs for producing the masks necessary to implement the design on silicon during the fabrication process.

1.5 **THESIS ORGANIZATION**

Chapter 2. discusses VLSI in terms of integration size, factors that limit integration size, the technology of fabrication of IC packages, the entire sequence starting from the design stages, to the packaging of a VLSI chip. The chapter also describes some recent VLSI related techniques which help the manufacturer get better yield or reduce the line widths and therefore increase the packing densities. Then the chapter discusses the computer aided design approach for designing VLSI circuits, the role of CAD in the different design stages of VLSI, the different methodologies used for VLSI layout design. The conclusion of the chapter is a discussion of some future issues regarding the design of large systems and the manufacturing of VLSI.
Chapter 3 gives an overview of the hardware and software aspects of the facilities used in this work.

Chapter 4 describes the developed interactive scheme for system integration, and the implemented prototype CAD station for VLSI MOS layout design. The chapter ends with examples of layouts done using the designed station.

Chapter 5 is a summary of the work done and conclusions. The listing of some of the programs developed and used within this work is given in the appendix.
Chapter II

VLSI, AND COMPUTER AIDED DESIGN

2.1 VERY LARGE SCALE INTEGRATION (VLSI)

The enormous progress of integrated circuit (IC) technology in recent years has been changing many things in our daily lives, because digital systems can be manufactured with much lower costs, lower power consumption, higher speeds and smaller sizes. In the following we will define VLSI and give a brief description of VLSI technology.

2.1.1 Integration Size

In table 1, integration size is defined as the number of gates on a chip, but we should take into account that the definitions are sometimes defined differently. Some authors (e.g. Barna and Porat) [1], define the integration size differently for MOS and Bipolar, for example, LSI is defined as 100 gates or more in the case of bipolar transistors and 300 gates or more in the case of MOSFETS.
<table>
<thead>
<tr>
<th>Integration Size</th>
<th>Number of Gates on a Chip</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSI</td>
<td>&lt;10</td>
</tr>
<tr>
<td>MSI</td>
<td>10-100</td>
</tr>
<tr>
<td>LSI</td>
<td>100-100000</td>
</tr>
<tr>
<td>VLSI</td>
<td>&gt;1000000</td>
</tr>
</tbody>
</table>
The integration size is limited by the production yield and heat generation and technology used in lithography. There is also a fundamental limitation on the size of integration imposed by laws of physics, that is the length of the channel should not be less than quarter of a micron [2], otherwise problems of metal migration due to high current densities in the metalization layer, and problems due to the deviation of impurities concentration from statistical uniformity will occur.

2.1.2 Fabrication of IC Packages

Thin round sheets of silicon about 4 or 5 inches in diameter, which are called wafers, are sliced from a single crystal. On a wafer tens to several hundreds of IC chips are produced simultaneously by the patterning of a particular sequence of successive layers. The patterning of silicon dioxide layer involves the steps shown in (fig.1), and described in the following points.

1. Creating the silicon dioxide layer on the surface of a bare polished wafer by exposing it to oxygen in a high temperature furnace.

2. After the wafer is cooled, it is coated with a thin film of organic resist material.
3. A mask containing the desired pattern is brought in close proximity with the surface and is flooded with an intense source of ionizing radiation, such as ultraviolet light.

4. The radiation passes through the transparent areas in the mask into the resist, and make it soluble in organic solvents.

5. Immersing the wafer in a suitable organic solvent, the pattern is transferred to the resist material.

6. Using hydrofluoric acid, the silicon dioxide is etched in places uncovered by a resist material, thus the pattern is transferred to the silicon dioxide layer.

7. The final step in patterning is the removal of the remaining organic resist material by using strong acids such as chromic acid or by using strong organic solvents.

We will now describe the particular sequence of patterned layers to build up nMOS integrated circuits and shown in (fig. 2).

1. The opening in the opaque material of the first mask exposes all areas that will eventually be the diffusion level.
2. The second mask is used to select areas that will be turned into depletion mode transistors. The actual conversion is done by implanting ions of arsenic or antimony into the silicon surface. The resist material will prevent ions from reaching the silicon surface, and only areas free of resist will be implanted with ions. The resist is then removed from the surface of the wafer.

3. A thin layer of silicon dioxide is grown on the surface of the wafer, then the entire surface is coated with a thin layer of polycrystalline silicon, which will eventually form the gates of the transistors and will serve as a second layer of circuit interconnections. A third mask is used to pattern the polycrystalline silicon.

4. Once polycrystalline areas have been defined, the n-type dopants can be diffused into the p-type substrate. The polycrystalline will prevent impurities from diffusing into underlying silicon. This self-alignment between the gate and sources and drains of the transistor has the advantage of reducing the transistor area.

5. The fourth mask is used for defining contact windows, which are the areas where the metal will make the contact with either polycrystalline or diffused areas.

6. The fifth mask is for patterning the metal layer and defining the circuit interconnections.
1. Silicon Wafer Covered with Oxide.

2. Coat with Resist.

3. Mask and Expose.

4. Develop Resist.

5. Etch.

6. Remove Resist.

Figure 1: Patterning of Silicon Dioxide Layer.
2.1.3 The Entire Sequence of Design and Fabrication of a VLSI Chip.

As seen in the previous fabrication process, all the information about the network to be implemented is contained in the masks used during the photoresist processes. In the following, the entire sequence, from mask preparation till packaging of a VLSI chip is described.

a. Mask layouts

The architecture of digital system to be realized on a chip, such as the connection configuration of registers and networks and the size of memories, is designed, and then the logic design of all the networks with gates such as adders and control logic networks is done. Subsequently the logic networks are converted into electronic circuits, and the layouts of electronic circuits are designed. They are drawn on a mylar sheets which can be as large as several feet on each side (depending on the chip size, or on a CRT screen by which layouts are stored digitally on a magnetic tape or disk. In this case we need to prepare many layouts since each layout represents a pattern on each mask to be used during fabrication. As the integration size of chip increases, the layouts become more time consuming, and the entire sequence of all designs require as many as tens of man-months or possibly more. Each step must be examined for possible mistakes and corrected, until there are no more easily detectable mistakes.
1. P-type silicon covered with patterned $\text{SiO}_2$.
2. Vapour deposition of polysilicon.
3. Patterning of polysilicon.
4. Removal of Oxide.
5. Diffusion of $n^+$ impurities.
6. Oxide layer is deposited.
7. Contact windows are opened.
8. Metalization patterns are established.

Figure 2: The nMOS Fabrication Process.
b. Digitizing

After the layouts of the entire chip have been drawn on Mylar sheets, they are converted into digital data by a digitizer with the aid of a computer, and those data are stored digitally on a magnetic tape or disk. Then different portions of each layout are displayed on a CRT one by one and inspected visually for further mistakes.

c. Reticle Preparation

After all corrections have been made (although more mistakes may be discovered in later stages), a reticle, which is a small photographic plate of the layout image, is prepared from each layout stored on the magnetic tape or disk.

d. Master Mask and Working Mask Preparation

The layout image on the reticle is then reproduced on a glass plate covered with chromium or emulsion photographic film by means of optical reduction equipment called a photorepeater. In this case the original layout image is repeated on the glass plate as many times as there are chips on the wafer, with a photoreduction of about 10:1. The glass plate thus obtained is called a master mask and looks like a tile floor where each rectangular tile has the same layout image of the chip. From this master a large number of working masks are reproduced which will be actually used for fabrication.

e. Printing
In the photore sist process, if the mask is directly contacted to the wafer, then the printing is called contact printing, and if the layout is projected optically onto the wafer, it is called projection printing. In the latter case the mask can be used a large number of times substantially reducing the mask costs per wafer.

f. Probe Testing

When the entire lithography process is over, the chips are tested by probes, because usually there are many faulty chips, after such highly complicated fabrication steps. A probe test is automatically carried out by contacting the pads of every chip with microelectrode probes, and electric signals from some pads are analyzed by computer. If faulty chips are found they are marked out and thrown away later.

g. Scribing and Breaking

The entire wafer is broken into individual chips by a laser scriber (or a diamond saw).

h. Bonding

Only good chips are mounted on containers often ceramic, and then the pads of the chips are connected to the terminals, that is pins of the container, with gold or aluminum wires.

i. Packaging

Each IC package is tested with its external terminals by feeding electric signals to its input pins and analyzing those at its output pins by a computer.
2.1.4 Recent VLSI Related Techniques.

In the following we will discuss some of the current trends in VLSI chip fabrication.

a. Direct Step on Wafer Projection

When the line width (e.g. the width of the metal connection on a chip is reduced in order to increase the packing density of devices on a chip, or when larger wafers are processed, the contact or projection printing does not work well. If a wafer is distorted or warped by the heat cycle during processing it will throw layer to layer registration of the masks on the wafer out of required accuracy, lowering the yield. In order to alleviate this problem, direct step-on-wafer (DSW) projectors or wafer steppers have been developed, which work as follows. The layout image on a reticle is directly projected onto the surface of a photore sist-covered wafer by an optical lens equipment with image reduction (10:1, 5:1, or 1:1). Only a small portion of the wafer, consisting of one chip or at most several chips using a reticle that many layouts, is exposed. Then the wafer moves one step further to a new position for the next exposure, being positioned accurately by a laser interferometer. The registration of the image can be done at each exposure, and also by using a smaller lens aperture, the resolution is higher than for printing methods described before, which project a large entire mask onto the entire wafer. On the
other hand, repetition of the exposures over the entire wafer takes more time than the methods with one exposure [3],[4].

b. Line Width Reduction and Speed Improvement by Deep-Ultraviolet, Electron-Beam, Ion Beam, or X-Ray Lithography.

When the line width is reduced, the chip size is reduced by the improved packing density. However, as long as ultraviolet lithography of 3000 Angstroms wavelength or longer is used in IC fabrication, the line width is limited to 1 micrometer, because if we try to make a fine line, the line becomes blurred due to optical phenomena. If we want to reduce the line width, we need to use a shorter wavelength to improve the resolution. By using deep ultraviolet lithography of 2000 to 3000 Angstrom, the line width can be reduced to about 0.7 micrometer. Thus the deep ultraviolet lithography is becoming popular [5].

By directly writing network patterns on a wafer by electron beam without masks, the line width can be reduced to about 0.5 micrometer. The on-off operation of the electron beam is controlled by an electric field, which is in turn controlled by a computer program. The entire wafer must be scanned by an electron beam, so the exposure is time-consuming.
By using X-ray lithography of 5 to 50 Angstrom, the line width can be further reduced to about 0.02 micrometer. Since it is difficult to form a very sharp X-ray beam and deflect it, masks are prepared with electron-beam lithography, and then each mask pattern is transferred to a wafer by the flood exposure of X-rays. Ion beam lithography (IBL) is also in exploration. Ion beams do not scatter as much as electron beams, and unlike X-rays, collimated sources are available. IBL has shown submicron (0.5 micrometer) resolution, however, presently they have throughput rates much lower than those of electron-beam systems. Some studies indicate that these machines will be in place on commercial IC production lines by 1994 [7].

2.2 COMPUTER AIDED DESIGN OF VLSI SYSTEMS

Computers have been extensively used in all stages of the design and development of an LSI/VLSI system, starting from system specifications to test of prototypes. The design approach is called design automation "DA", when the computer makes all the design decisions given a high level description of what the circuit is supposed to do. The programs which implement this design approach are called "Silicon Compilers". Many of these silicon compilers are currently in use like Macpitts developed at MIT, Bristle Blocks, developed at Caltech, and Model developed at the
university of Edinburgh, in Scotland. When the computers are used with the designer intervention, providing him with tools varying from interactive graphics, to individual programs for circuit or logic simulation, the design approach is called computer aided design (CAD).

2.2.1 Objectives of CAD Programs

The objectives of CAD programs are:

1. To minimize design mistakes and facilitate design changes.

2. To shorten the design and development time of LSI/VLSI chips as well as systems built with this chips, and also to shorten the time for design verification and testing.
2.2.2 Use of CAD in the Different Design Stages of a VLSI System

CAD programs can be either of great value or little value, depending on the tasks in the design for which CAD programs are used. In the following we will check the different stages in VLSI chip design and the use of CAD programs in each stage.


When we want to design a digital system, we need to specify the system performance. This is called system specification. Then the system must be broken down into subunits, and the functional design specifies the relationship among the subunits. Then we need the logic design of the networks which constitute subunits or registers. System specification functional design and logic design are still the most difficult stages for the use of CAD, due to their complexity, throughout the entire design sequence. As the integration size increases, the need for logic design tools increases, and also for computer languages to facilitate system specification, functional design, and in particular control logic [8].
b. Logic Simulation

When a system architecture or logic networks are designed, performance and errors are checked by CAD programs. This is called logic simulation, since CAD programs check by simulation whether the designed systems or networks are realized as desired. In this case the CAD programs usually check whether the networks contain hazards. Some logic simulation programs are available to the public or can be purchased. Currently logic networks of up to tens of thousands of gates can be simulated, but if larger networks are to be simulated, logic simulation is extremely expensive. Logic simulation is important because any logic design errors discovered after realizing chips are time consuming, and expensive to correct, due to the repetition of electronic circuit design, layout, and prototype wafer fabrication [9].

c. Electronic Circuit Simulation and Analysis

Logic circuits have to be converted into electronic circuits when designers specify electronic circuit requirements, such as speed, power supply voltage, types of logic operations, and signal level tolerances. It is desirable to have CAD programs which automatically design electronic circuits meeting all requirements, and specify parameters such
as dimensions of transistors and magnitudes of currents. This is difficult, and the current practice is as follows. Before layout, designers design electronic circuits temporarily, assuming the sizes of some circuit parameters, analyze the circuits by CAD programs, and then modify the circuits based on the analysis. After having obtained more specific values for circuit parameters from actual layout, designers finalize the design of the electronic circuits, analyzing and simulating the laid-out circuits by CAD programs. For this electronic circuit simulation and analysis, CAD programs perform complex numerical analysis calculations of nonlinear differential equations which characterize electronic circuits. The CAD programs usually yield the analysis of transient behaviour, direct current performance, stationary alternating current performance, signal distortion, noise interference, sensitivity, and parameter optimization of the electronic circuits.

After CAD programs such as TAP, RCAP, and NEP1 had been developed for SSI and MSI during the period from 1964 to 1972, more powerful CAD programs for LSI, such as ASTAP by IBM, and SPICE by Nagel and Pederson, 1973, have been developed. Currently, many CAD programs with different features are available, such as CIRCUS 2, SPICE 2, ISPICE, ITAP, NEPII, and UCCAP. Most of them typically can handle electronic circuits of up to a few hundred transistors. In particular, SPICE2 and its modified versions are used extensive-
ly throughout the industry. Also a CAD program MOTIS, was developed by BELL labs in 1975. It has high speed by being tailored to the transient analysis of MOSFETS at the sacrifice of accuracy. MOTIS can handle up to 1000 cells and MOTIS-C can handle up to 2500 cells [8].

d. Layout

The layout for random-logic networks is the most time-consuming stage throughout the entire sequence of LSI/VLSI chip design, though it is done mostly by hand. Other design approaches, such as those based on gate arrays and cell libraries, to shorten layout time, have been conceived in addition to random-logic networks, and designers must choose one based on tradeoff between layout time and quality (i.e., size and performance) of the finished chips. After having finished the layout, designers usually check by CAD programs whether the layout conforms to the layout rules.
2.3 **Layout Design Methodologies for VLSI**

Even at the current level of complexity, chip designers are faced with a task of horrendous proportions. Designing a complex logic chip without automated design tools reportedly costs in the neighborhood of 100 man-years or $10 million [10]. Moreover, as the complexity of the chip increases, design costs will also increase. So finding new design methodologies became important. The methodologies that evolved are bounded on one end by unconstrained, or totally customized, design approaches, and on the other end by constrained approaches, the most restrictive of which is the gate array.

In an unconstrained approach the designer has essentially total freedom to optimize the design to meet the requirements of the application. The only restriction is that the design meet the technology ground rules. The design process typically consists of drawing each individual shape on a CRT terminal. Software design tools enable the designer to group shapes or circuit elements into higher level functions, to replicate shapes or group shapes, to rotate shapes, and so on. In addition, many powerful checking analysis tools can be run to increase designer productivity and accuracy. The advantages of this design approach are high density and high performance or improved operating speed for
the resulting chips. However, these advantages must be weighed against longer design times and higher design costs.

On the other end of the spectrum is the gate array, a method used by most existing design systems. In a gate array, a predefined pattern of transistors is fabricated in an area of silicon called the cell. The pattern is then repeated so that almost the entire chip is covered with identical cells. In a library, or file, the design system has stored all of the circuit designs that the chip designer is allowed to use. These circuits consist of metalization patterns that interconnect the transistors in the individual cells. The design process consists of selecting circuits from the library needed to implement the desired logic function, and placing them in an optimal configuration on the chip image. Circuits are placed in cells so that related circuits are near one another. Once the circuits have been placed, they are interconnected by metalization patterns between the cells. With this design methodology, the designer is restricted to a predefined image of cells, containing predefined transistors and to a predesigned library of circuits.
Figure 3: Gate Array
Because of these restrictions, gate arrays have lower densities and lower performance level than custom designed chips. However the advantage of this approach is a very rapid low-cost design. These savings in time and money come about because most shapes on the chip are transistors and transistor interconnections. These shapes are constructed ahead of time, and the remaining shapes, used to interconnect the circuits are designed automatically. Consequently the designer does not draw any shapes by hand.

2.4 FUTURE VLSI ISSUES

When the integration size increases we have to face a new formidable problem in developing LSI/VLSI chips. We have to integrate, without conflict, different scientific and engineering disciplines such as processing technology, software, algorithms, architecture, logic/electronic circuit layout designs, CAD, design verification, testing, and packaging. In the following we will investigate some of the key future issues regarding VLSI.
2.4.1 Future Large System Design

As the integration size increases, there are probably two different paths in future system design. The first one is to pack cpus, control logic, and memories in each chip as much as possible. This will provide the most cost effective systems in high volume production. Computers designed with these VLSI chips must be able to process the existing software, from marketing point of view. The second path is to design a digital system as an aggregate of dedicated processor chips, probably mixed with general purpose processor chips. Each chip is dedicated to advanced mathematical function computation, language compilers, data-base management, other key portions of conventional operating systems, sorting, input/output channel control, maintenance, application programs and many others. Systems with very high performance can be realized using these dedicated processor chips. Computers, based on nonconventional architecture, such as data-flow computers which are digital systems of parallel processing with multiprocessors, can also be realized with dedicated processor chips.
2.4.2 Future VLSI Manufacturing

The structure of the manufacturing industry is gradually changing. Computer manufacturers, watch manufacturers, toy manufacturers are picking up LSI/VLSI technology know-how, at the same time, semiconductor manufacturers are picking up system design capabilities and supporting software development capabilities. Also, manufacturing methods are drastically changing. Assembly workers are replaced by chip designers aided by CAD systems [11]. CAD systems are inevitable for essentially all manufacturers, requiring huge investments for many years. CRT terminals are distributed throughout a company so that designers can conveniently design LSI/VLSI chips by using a large number of CAD programs and a data base stored in central computers. If we realize that the initial investment for chip development is mostly salaries we can see the increasing importance of effective CAD systems to manufacturers of VLSI chips in the future.

SUMMARY

In this chapter, VLSI was defined and the factors limiting integration size were discussed. The entire sequence of design and fabrication of a VLSI circuits and some of the recent technologies used in fabrication were also discussed. The chapter ends with an overview of the design automation and the CAD approaches for designing VLSI circuits, and a discussion of some future issues regarding the design of large systems.
Chapter III
SYSTEM OVERVIEW.

This chapter describes the system existing in the Electrical Engineering Department, University of Windsor, some of the software features, and the menu driven executive system, which allows the user to interact and fully utilize the system, without necessarily going through the various manuals of its components.

3.1 OVERVIEW OF THE SYSTEM HARDWARE

The system is a mix of hardware and software. The hardware environment consists of the SEL 32/27 computer, the AYDIN & CONTROLS display computer and graphics monitor, the TRILOG 100 colour plotter, and TSD DISPLAY CANADA touch screen. Other peripherals include a high speed convolutional filter, designed and implemented in the department, and an EPROM programmer. The SEL computer can communicate with the NOVA 840 computer in the same laboratory in our department, which allows us to transfer images, algorithms previously developed, on the NOVA to the SEL. The organization of the system is shown in Figure 4, and in the following some hardware and software details about components of the system used in this research are given.
Figure 4: System Organization.
3.1.1 SEL 32/27 Mini-Computer Architecture

a. Selbus

The Selbus is a high-speed synchronous bus that can transfer data at the rate of 26.67 million bytes per second (Fig. 5). Each module on the bus is assigned one of 23 Selbus priority lines by jumper settings.

b. Multipurpose bus

The multipurpose (MP) bus is a medium speed asynchronous bus that can transfer I/O data at a rate of 1.5 million bytes per second. Up to 16 device controllers can exist on the MP bus at a time.

c. CPU

The 32/27 contains a 32-bit CPU that interfaces with the Selbus. The CPU uses instruction lookahead for fast execution of instructions. Instruction fetches are made concurrently with instruction decode and execution.

d. Registers

The SEL computer has a set of eight high-speed, general purpose registers for use by the programmer in arithmetic, logical and shift operations.

e. Memory
Figure 5: SEL Computer Architecture.
Memory in the SRL computer interfaces directly to the Selbus. The Integrated Memory Module is a high density memory subsystem implemented on a single board including 256 KB of dynamic MOS RAM, memory controller, error correction logic, and refresh circuitry. The IMM can overlap two read cycles, allowing reads to be initiated every 300 nanoseconds to a single IMM. A write can be initiated every 300 nanoseconds to a single IMM.

f. Input/output processor

The IOP is a powerful multiplexing channel for I/O operations. The IOP operates independently of and in parallel with the CPU. The result is increased CPU availability and improved system performance.

3.1.2 Aycon 16 Display Computer Architecture

The Aycon 16 derives its function versatility from a modular, bus-oriented architecture, shown in block diagram form in Figure 6. A minimum system configuration consists of four basic module types: processor, memory bus controller/sync, refresh memory and video module. The system processor bus, designated AYBUS, and the memory bus, designated MEMBUS, form dual-bus communication paths which link all elements of the Aycon 16 circuitry.

a. Processor Module
Figure 6: Aydin Computer Architecture.
The primary component of the Aycon 16 circuitry is the processor module, which incorporates an Intel 8086 high performance 16-bit microprocessor as the control and arithmetic element.

b. Memorybus Controller/Sync Module

The link between the AYBUS and the MEMBUS is provided by the memory bus Controller/Sync module circuitry. This module, as the primary master of the MEMBUS, generates all timing and control signals necessary to maintain refresh of the CRT display by utilizing picture information stored in the refresh memory.

c. Refresh Memory Module

An aycon 16 system may incorporate as many as 16 refresh memory modules, each of which, depending upon the required storage capacity, contains 64 of 16K dynamic Random Access Memories (RAMS). The refresh memory module is logically subdivided into 4 memory planes (channels), with 4-bit Z-axis definition of each pixel and 512 x 512 resolution. Data can be loaded into or read from the refresh memory in either a 16-bit word format or a pixel format.

d. The Video Module
It couples the refresh memory outputs to the CRT display via look-up tables (LUTS), digital to analog converters, and video amplifiers which mix synchronization and video signals to generate a composite video signal.

3.1.3 The Touch Screen

The model TF Touch Screen Digitizer enables personnel to gain access to computer data by simply touching a computer display with a finger. It consists of a thin transparent curved panel, which mounts in front of a standard CRT display monitor, and an electronic board connected to the panel with a cable. When the panel is touched, the location of the touch is measured and sent to the host computer as a RS232C message or as parallel data. The digitizer acts as a data entry device and can replace a keyboard or graphic tablet in many applications. The TF series digitizers measure the touch position by measuring the voltage distribution across a transparent conductive film. Two thin transparent conductive sheets are mounted in front of the CRT but kept separated by an insulating spacer at the edges. When the sandwich is touched, one conductive layer is forced into the other, yielding an output voltage proportional to the touch position. This voltage is converted to a binary number combined with data from the other axis, filtered, formatted into ASCII characters and transmitted as a serial RS232C message.
Operating Modes of the Touch Screen

There are 5 operating modes implemented on the model TP Touch Screen.

Mode 0 Inactive

The screen is completely inactive and produces no output when the screen is touched. The touch indicator LED also does not respond to a touch.

Mode 1 Continuous Output

The screen outputs touch position data continuously whenever the screen is touched. The serial output rate is set by the selected baud rate or by 60 coordinate pairs per second, whichever is less.

Mode 2 Output on Initial Touch Only

The screen produces output only on initial touch of the screen. The screen touch must be released before any other output is produced. This feature is useful for menu selection, and it is the mode currently used in this work.

Mode 3 ADDS Regent 40 Alignment
This mode can be used to align the touch film to an ADDS Regent 40 terminal. The output string will position the cursor to the current touch location and write an asterisk at the touch point.

Mode 4 Fixed Array of 80 Touch Pads

This mode reduces the load on the host computer to a minimum, because the screen is divided into 80 fixed touch pads and only a single ASCII character is transmitted for each pad.

3.1.4 TRILOG 100 Colour Plotter

The TRILOG 100 colour plotter is capable of providing high quality low cost colour printing and plotting. Through the use of a raster matrix impact technology and TRILOG's proprietary multicolour ribbon drive, plotting either multicolour or black copies is possible on plain paper. Multicolour printing or plotting is achieved by interspersing dots of the standard process colours: yellow, red and blue. Different hues tones are achieved by programming dot density. For example, green is produced by interspersing yellow and blue dots; the tones (shades) of green (light and dark) depend upon the density of interspersed dots of each of the two ribbon colours used to produce green. The TRILOG 100
uses a unique ribbon made up of three serial colour zones, end to end. The system is typically programmed so that all yellow dots are first plotted on the form. The form is then reversed to the starting position and again advanced as all red dots are plotted. Then, the form is reversed a second time and then advanced to plot all blue dots on the form. The resulting image is a true multicolour plot. TRILOG's proprietary COLORPLOT bidirectional paper drive system assures dot registration accuracy to be within plus or minus 0.005 inches. By installing a plain black ribbon and selecting SINGLE on the SINGLE/MULTI switch, TRILOG may be operated as a standard black and white printer and plotter. TRILOG prints the standard 96 character ASCII set at 150 lines per minute with text quality characters, or at 250 lines per minute with standard dot matrix characters.

A dot is impacted by electromagnetically releasing a leaf spring normally held under tension by a permanent magnet. Printing is done by a bank of 44 leaf spring hammers positioned horizontally at every third character position. The hammers are mounted on a shuttle which sweeps the hammers across three character positions over a 0.3-inch movement. As the shuttle sweeps across, the hammers are activated at each position in the dot row at which a dot is required. While the shuttle is slowed at the end of each movement, and then accelerated in the reverse direction, the paper feed moves the paper upwards one vertical data row.
The shuttle then sweeps across in the reverse direction again distributing dots in that row.

**Plotting**

When plotting, data is interpreted with each bit in a 6-bit character representing a discrete dot on the plot. A transfer of up to 220 six-bit character may define dots in each dot position in the horizontal dot row. A total of 1320 dot positions are available on each line (dot row) to be plotted, when using full-width paper. A line feed command causes paper to advance one dot row instead of one character line as in the print mode. The ribbon drive system initializes by moving the ribbon through a colour boundary crossing, reversing direction and crossing the boundary a second time. During the second pass, the ribbon coding is read, allowing the printer to set logic to determine what colour zone is in front of the print station, and which direction the ribbon must be moved to find the other two colour zones. When the program specifies the colour to be printed, the unit searches for the required colour zone on the three colour ribbon. Printing proceeds when the required colour zone becomes available at the print station. In multicolour plotting, when the second and third colours are required, paper is reversed to the beginning of the plot.
3.2 OVERVIEW OF THE SYSTEM SOFTWARE

In this section we will concentrate on some of the software features of the system, that were utilized in developing the menu driven SEL EXECUTIVE SYSTEM. We shall begin with SEL MPX-32 operating system and some of the software modules in the MPX-32 package like the Terminal Service Manager (TSN) System Generator (SYSGEN), and utilities such as the File Manager. Then we discuss some of the software features of the Aydin graphics system.

3.2.1 SEL SOFTWARE

a. SEL MPX-32 Operating System

MPX-32 provides effective support for real time processing by responding quickly to external events and offering a broad selection of task activation methods. Also, MPX-32 permits up to 64 interactive terminal user to concurrently create, debug and execute programs. Additional features include interterminal communications and the ability to execute console operator functions from any terminal.
b. Interactive Terminal Support

The Terminal Services Manager (TSN) is an integral part of MPX-32 and provides the capability of supporting up to 64 interactive terminals. TSN allows the terminal user to:

- log on to MPX-32
- access any MPX-32 processor
- communicate with other terminals
- run tasks in any processing environment (real-time, interactive, or batch)
- return the interactive environment on exit from another processor
- log off MPX-32

Ease of use is increased by TSN's ability to accept commands from a TSN command file in lieu of terminal input. Any TSN command or TSN macro command can be used to make assignments, define options, and perform other run time operations associated with activating a processor or other task. Additionally, command files may be chained to access other command files. This feature was of great benefit in creating the menu-driven SEL EXECUTIVE SYSTEM.

c. System Generation (SYSGEN)
SYSGEN is a system utility of MPX-32 that can be executed either in batch or interactively. SYSGEN reads a set of configuration directives and creates a permanent file containing the installation specific MPX-32 system in memory image absolute format. A resident system image is formed and subsequently written to the dynamically acquired disc file (System Image File), and after SYSGEN is complete, the user has a permanent file containing the installation-specific MPX-32 system.

d. File Management

In the MPX-32 operating environment, files are used in several ways. Permanent files are created for user programs, user data and system programs. Residing on disk storage, they are defined as system or user files by entries in the System Master Directory (SMD) which specifies each file's name, device address, size, type and system flags. The permanency of these files stems from the fact that all SMD entries are stored on the disk and may be deleted only when specifically directed by the user. All permanent files are referenced by name, and any number of tasks may access any permanent file for both input and output.
3.2.2 Aydin System Software

The Aydin 5216 display computer processing ability is built around the powerful Intel 8086 16 bit microprocessor. The standard firmware provided with the 5216 display computer has an instruction set of largely alphanumeric and graphic instructions which accepts and processes instructions for display. The Aydin 5216 in our system cannot work as a standalone computer, since it is not provided with its own operating system. The operating system is supplied by the SEL computer which communicates with the 5216 as if it were a peripheral device. The standard firmware instruction accepts codes from computer and then executes all the necessary code to generate alphanumeric or graphic data on the display monitor. Alphanumeric characters can be displayed in three sizes, and graphics capabilities include lines, circles, fill, cursor movement, roll, scroll, zoom, and copy. Colour displayed by 5216 is determined in two ways: first, by the data the programmer has stored in refresh memory, and second, by the colour value which the display hardware assigns to each stored data value via a programmable lookup table (LUT) located on the video display card. The LUT on the vid-4 card in our system contains 256 colours.
Generalized Program Structure for the 5216 Display Computer

Typical instruction sequences are indicated in fig. (7). The mode selection determines whether alphanumerical or graphic data is to be subsequently entered, and determines also the memory input mode (OR 1's, Replace, Erase 1's). If pixel mode is selected, the foreground and background pixels should be loaded. Once the channels are selected and the limits specified, scrolling or clearing can be performed. The remaining sequences show the parameters which must be specified, before vectors/conics and characters or graphics may be written.

**SUMMARY**

In this chapter an overview of the hardware and software aspects of the facilities existing in the Electrical Engineering Department was given, with emphasis on the elements used in this work.
Figure 7: Generalized Program Structure for the 5216 Display Computer.
Chapter IV

SYSTEM INTEGRATION AND COMPUTER GRAPHICS AIDS
FOR VLSI MOS CIRCUITS LAYOUT DESIGN.

In chapter 3, we have defined the environment, the SEL computer and its peripherals. In the next sections we will describe the menu driven Integration Scheme developed within this work. The modular structure of the software allows for future system expansion, with minimum alterations of the first graphic page and the decision structure related to it.

4.1 MENU SYSTEM HIERARCHY

In fig. (8) the hierarchical structure of the menu system is shown. The dotted lines show possible future extensions of the menu. The first layer of the hierarchy contains the main menu, which has display/plot, VLSI layout and future extensions are convolver mode for interacting with the convolver, and camera mode, dealing with a video camera and a digitizer when they are interfaced to the system. A provision for exiting the executive system is provided in this layer also. The second layer contains submenus. For the display/plot mode, the first submenu contains the following:
Figure 9: The Main Menu.
Figure 10: The Display Menu.
Figure 11: The Edit Submenu.
Figure 12: A Piston Head Image Positioned and Coloured Using the Edit Menu.
a. Image Directory

These switches allow the user to choose the image to be displayed by touching the name of the image.

b. Plot

It allows the user to plot the image on the plotter.

c. Main Menu

This switch allows the user to get back to the main menu for another choice.

d. Edit

This switch gets the user into the third layer of the hierarchy.

The third layer of the hierarchy contains the edit submenu, which allows the user to perform some editing functions on the displayed image, like Bolling, Scrolling, Zoom-
ing. A provision for going back to the previous page is also given, for making another choice from the directory.

The VLSI layout menu will be discussed in the following sections.

4.1.1 Developing of the Menu System

In developing the menu system a top down design approach was used. First, a study of the system as a whole and the capabilities of each component was made. Then the basic requirements from this system were translated into the hierarchy discussed before. The next step was to actually design the software which will implement these functions. The programs which implement these functions has two parts: the graphics display part which will display the soft switches on the screen, and the other part consists of the programs which translate the physical touch of the screen into a desired action.

As an example, we take page 1. First a program for graphical page 1 containing the items of the main directory: Display/Plot, VLSI Layout, Exit, is developed. Then, a decision structure for that page is developed. The decision structure is composed of two parts. The first is a Fortran program which reads the touch position measured by the touch screen, the second is a file of MPX-32 commands, which
branches to different actions according to the touch positions from the Fortran program.

After designing the first page corresponding to the first menu, we proceed for the 2nd page then the third page and so on.

Within one page the decision structures may consist of one load module, or an MPX-32 command file, conditionally activating several load modules. The one module decision structure has the advantage of speed of execution. The MPX-32 decision structure has the advantage of being able to deal with large applications, without violation of the maximum load module size constraint imposed by the system (load module size + operating system size = 128k words). In this design the MPX-32 command file was chosen as the general scheme of integration, but within some pages like the editor of the VLSI layout system the decision structure and the actions are implemented as one load module.
Figure 13: Fortran Decision Structure for PAGE 1
Figure 14: Algorithm of an MPX-32 Command File for PAGE 1
4.2 INVESTIGATION OF THE SYSTEM CAPABILITY FOR VLSI LAYOUT DESIGN.

As an investigation of the system capability for VLSI layout design, a prototype CAD facility was designed and implemented, with the following basic requirements in mind.

1. The designed facility has to facilitate the layout procedures.
2. It has to capture the designed layout within the computer memory by the end of the design session.
3. It has also to produce the masks necessary to implement the designed layout on silicon.

To achieve these goals, work has been done in the following order:

1. Choosing a suitable design methodology.
2. Design of the modules or building blocks within the design methodology.
3. Software implementation of the modules and of the wiring technique.
4. Design of the interactive editor.
5. Design of a storage method for capturing the layout within the computer memory.
6. Design of the software for producing masks out of the designed layout.
7. Design of the menus for driving the designed software and connecting these menus to the main menu.

4.3 **STANDARD CELL LAYOUT DESIGN APPROACH**

In this approach, compact layouts of basic gates and standard networks are designed with time-consuming efforts and are stored on the computer disk. The designer can call up instances of these gates on the screen of the graphics monitor, and by arranging them and adding connections among them, he can make the entire layout of the network. When the layout is complete, masks are automatically prepared by the computer. All the cells have the same height, but may have different widths. Cells are arranged in pairs of rows, back to back, to facilitate sharing of the same VDD and GROUND lines, with channels separating them, in which routing takes place. By convention, horizontal wiring is done by metal, and vertical wiring is done by polysilicon in another level. The cells are oriented with the input/output locations adjacent to the wiring bay. A NAND and an INVERTER gates were chosen as the first elements in the library, but other modules could be added in the future.
4.4 LAYOUT OF BASIC CELLS

The layout of NAND and NOT gates used in the cell library, are shown at the end of this section. Both of the cells have the same height and width, and the ground and VDD lines are at the same level with respect to the top and bottom of the cell. Mead and Conway have presented a set of design rules, to ensure that the resulting layout is correct by construction, which is one of the fundamentals of their approach to the design of VLSI circuits. These design rules are in a dimensionless form, and represent constraints on the allowable ratios of certain distances to a basic length unit. This basic length unit is equal to the fundamental resolution of the manufacturing process, and was denoted 'Lambda'. Throughout this work, and in the developed programs, this length unit is denoted 'L', to facilitate typing.

The following design rules were used in the design of basic cells:
1. Minimum width of a diffused region is 2L.
2. Minimum spacing between two electrically separate diffused regions is 3L.
3. Minimum width of polysilicon lines is 2L.
4. Minimum spacing between two polysilicon lines is 2L.
5. Minimum separation between ion implantation region and adjacent enhancement mode transistor region is 1.5L.

6. Contacts between metal layers and polysilicon layers or diffusion layers should be at least 2L long and 2L wide.

7. Each level involved in a given contact, should extend beyond the outer boundary of the contact cut by L at all points.

8. Minimum width of a metal line is 3L.

9. Minimum separation between two metal lines is 3L.
Figure 15: The Layout of a NAND Gate as Realized on Aydin.
Figure 16: The Layout of a Not Gate as Realized on Aydin.
4.5 SOFTWARE IMPLEMENTATION OF THE BASIC CELLS

Two FORTRAN 77 routines are used in implementing the basic gates. These routines are also capable of mirroring the gates, translating and locating them anywhere within predefined rectangular limits on the screen. The first routine called COD, calculates the rectangular limits of all the rectangles in the different layers composing the gate. It also assigns different colours to the different layers. All the information pertinent to one rectangle (i.e. the left rectangular limit, the right rectangular limit, the upper rectangular limit, the lower rectangular limit and the colour of the rectangle) are kept as one record in a temporary file on the disk. The GATE routine opens this file, and reads the different records and sends the information to the Aydin computer to execute the required patterns. All the dimensions are taken with respect to the top left corner of the cell, which allows for translation, and in units of a variable 'L', which allows for scaling. The number of rectangles in a NAND gate is 56, and the number of rectangles in a NOT gate is 50.
<table>
<thead>
<tr>
<th>LRL</th>
<th>X+11*L</th>
</tr>
</thead>
<tbody>
<tr>
<td>RRL</td>
<td>X+15*L</td>
</tr>
<tr>
<td>URL</td>
<td>Y+23*L</td>
</tr>
<tr>
<td>BRL</td>
<td>Y+25*L</td>
</tr>
<tr>
<td>COLOUR</td>
<td>4</td>
</tr>
<tr>
<td>XP</td>
<td>LRL+1</td>
</tr>
<tr>
<td>YP</td>
<td>URL+1</td>
</tr>
</tbody>
</table>

**LRL** is the left rectangular limit.
**RRL** is the right rectangular limit.
**URL** is the upper rectangular limit.
**BRL** is the bottom rectangular limit.
Colour is the number of the colour in the recent look-up table programming.
**L** is the scale variable chosen by the designer.
**XP**, **YP** are the X and Y coordinates of any pixel within the rectangle.
TABLE 3

Colours Assigned to the Different Layers in the Layout.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Colour</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diffusion Layer</td>
<td>Green</td>
</tr>
<tr>
<td>Ion Implantation Layer</td>
<td>Yellow</td>
</tr>
<tr>
<td>Polysilicon Layer</td>
<td>Red</td>
</tr>
<tr>
<td>Contact Windows</td>
<td>White</td>
</tr>
<tr>
<td>Metal Layer</td>
<td>Cyan</td>
</tr>
</tbody>
</table>

4.6 SOFTWARE IMPLEMENTATION OF THE WIRING TECHNIQUE

A FORTRAN 77 routine called wire takes the coordinates of the beginning and the end of the wire and sends the information to the Aydin computer. The Aydin then executes vector operations and connects the desired points with the desired colour, representing either metal or polysilicon. The routine can also delete an existing wire between the beginning and end points.
4.7 **DESIGN OF THE INTERACTIVE EDITOR.**

The screen layout is shown in fig. (18). The Aydin screen is divided into two regions, the edit window, where the designs are displayed and edited, and the menu window from which the user can select actions. The menu has two parts, the upper one is concerned with execution, and the lower part is concerned with cursor control and cursor position registration. The execution menu, has the names of the modules and their mirrors in the library, METAL and POLY for wiring with metal or polysilicon, DELETE CELL for deleting unwanted or wrongly placed cells, ERASE METAL and ERASE POLY for erasing unwanted wires or correcting wiring mistakes. The cursor control and position registration menu has provisions for moving the cursor right and left, up and down, with two cursor speeds per touch, fast and slow. At the required cell position, touching the locate cell switch would transfer control to the execution menu and register the cursor X and Y coordinates in two temporary variables in the program, to be used by the COD and GATE routines when one of the gate switches is touched. For wiring, two positions have to be registered, the beginning and the end of the wire, by touching the corresponding switches. Control would not be transferred to the execution menu unless either the LOCATE CELL or the END METAL or the END POLY switches is
touched. Flags are set with each cursor registration so that only one type of program (i.e. GATE or WIRE) could be executed, to avoid mistakenly locating a cell and touching METAL or POLY and vice versa. The CURSOR ENABLE switch would transfer control to the cursor menu and will initialize the cursor counters in the program.
TABLE 4

Interactive Sequence for Locating a NAND Gate on the Screen Using the Edit Menu.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Program's Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Touch cursor enable.</td>
<td>Cursor counters are set to zero. Control is transferred to the cursor menu.</td>
</tr>
<tr>
<td>Touch any of the cursor</td>
<td>Cursor moves accordingly.</td>
</tr>
<tr>
<td>right, left, up or down</td>
<td>Control is transferred to the execution menu, cursor location is read in the</td>
</tr>
<tr>
<td>switches.</td>
<td>program.</td>
</tr>
<tr>
<td>At the desired location of</td>
<td>A NAND gate is drawn at the cursor position.</td>
</tr>
<tr>
<td>the cursor, touch the locate</td>
<td>The cursor position is registered in an array.</td>
</tr>
<tr>
<td>cell switch.</td>
<td></td>
</tr>
<tr>
<td>Touch the NAND Switch.</td>
<td></td>
</tr>
</tbody>
</table>

4.8 **LAYOUT CAPTURING.**

As the layout process proceeds, the coordinates of each cell located or deleted, and the coordinates of the beginning and end of each type of wire existing or erased, are stored in their respective array within the program. When the designer decides to stop the design operation by touch-
ing the stop switch, all these coordinates are stored in a temporary file for this session. This file will contain all the information regarding the design, like the number of gates of each type and number of wires and their locations. Recalling that a NAND gate 56 rectangles of different colours within it, and that it is stored at the end of the design session as one point, we can see the efficiency of the storage method and the efficiency of the module design.

4.9 SOFTWARE FOR IMPLEMENTING THE INTERACTIVE EDITOR AND THE STORAGE METHOD.

A FORTRAN 77 program called CAD, has within it the COD, GATE, WIRE and DELETE routines, is implementing the CAD functions described in the previous section. The program reads the touch screen messages and branches to different actions by a series of IF THEN ELSEB blocks. It also reads the 'L' design scale variable once at the beginning of the program. Once the scale variable 'L' is chosen at the beginning of the design session, it cannot be changed during the entire design time through the interactive editor menu.
One of the built-in features of this program is that the cursor advance per touch in the fast mode would automatically adjust itself to the cell dimensions regardless of the 'L' scale variable chosen. Another built-in feature is that the program automatically adjusts the rectangular limits of the editor display window according to the scale chosen. If the cursor is just enabled, but not moved, the program defaults to zeros for the X and Y coordinates of the cursor position.

4.10 MASKING SOFTWARE

After an editing session, all the information regarding the design is kept in a file on the disk. This information is in the sort of records of coordinate points. The file containing the design information can be accessed through another submenu, from which the designer can select to display the whole design again, or to display any of the fabrication process patterning masks.

4.10.1 The Masking Programs
These programs open the file containing the design information, and store the design data in respective arrays, initialize the Aydin, program the look-up table in the same way it was programmed before in the editor mode. By using the data points in the arrays and calling the corresponding subroutines, the respective level patterns representing the mask patterns, are drawn, whether they are within a cell or a wiring level. Each Maskn program calls a specific Codn subroutine for calculating the coordinates of the rectangles in a certain level of the gates present in the layout. Inputs to these subroutines are the X and Y coordinates of the top left coordinate of the rectangle and two other parameters or flags, indicating to the subroutine the type of the cell and whether it is mirrored or not. The Mask4 program also call the Window and Dwindow subroutines for implementing the contact windows between the metal and polysilicon wires in the layout.
4.11 DESIGN OF THE VLSI MENUS AND CONNECTION TO THE MAIN MENU

The VLSI menu is composed of 3 submenus, shown in figures (17, 18, 19). Getting access to the first menu is by touching the VLSI switch in the main menu designed within the integration scheme. The first submenu is for entering the design scale variable 'L', and for getting access to the editor menu. Touching the stop switch in the editor menu will automatically call for the third submenu, from which the display of masks or the whole layout is possible. A provision is given in this menu for going back to the main menu. Connecting these menus to the main menu is done within the general scheme designed before, using the first MPX-32 command file within the decision structure of the first menu.
Figure 17: VLSI Submenu Number 1.
Figure 18: VLSI Submenu Number 2.
Figure 19: VLSI Submenu Number 3.
4.12 TESTING OF THE DESIGNED SYSTEM

In order to test the system, the layouts of three logical circuits were implemented, using the designed computer tools. The first circuit is an exclusive or gate, the second one is a binary half-adder circuit, and the third is a decoder circuit. The logical diagrams and the resulting layouts of these circuits are given at the end of this chapter.

SUMMARY

In this chapter, the interactive scheme for system integration, and the implemented prototype CAD station for VLSI MOS circuits layout design were described. Examples of the layouts done using the implemented CAD station were given at the end of the chapter.
Figure 20: Logic Diagram of an XOR Gate
Figure 21: Layout of an XOR Gate.
\[ S_i = X_i (\overline{X_i+Y_i}) \]
\[ + \]
\[ Y_i (\overline{X_i+Y_i}) \]

\[ C_i = X_i Y_i \]

\[ \overline{C_i} = \overline{X_i Y_i} \]

Figure 22: Logic Diagram of a Half-Adder Circuit
Figure 23: Layout of a Half-Adder circuit.
Figure 24: Half-Adder Circuit. Mask1: Diffusion Layers Borders.
Figure 25: Half-Adder Circuit. Mask2: Ion Implantation Layers.
Figure 26: Half-Adder Circuit. Mask3: Polysilicon Layers.
Figure 27: Half-Adder Circuit. Mask4: Contact Windows.
Figure 28: Half-Adder Circuit. Mask5: Metalization Layer.
Figure 29: Logic Diagram of a Binary Decoder.
Figure 30: Layout of a Binary Decoder Circuit.
Chapter V
SUMMARY AND CONCLUSIONS.

A system integration scheme and a system for layout design of VLSI MOS circuits, were provided through this work. Both of the integration scheme and the layout design system are based on the graphics capabilities of the Aydin graphics computer, enhanced by Fortran 77 programs developed through this work. The integration scheme is also based on the powerful MPX-32 operating system of the SEL 32/27 computer, and the interaction capabilities provided by a touch screen.

The system capability for VLSI layout design has been investigated in this work by implementing a prototype CAD station for this purpose. The station supports a Standard-Cell design approach, and has as first elements in its library, the basic NAND and INVERTER gates. The station helps the designer in the layout process by providing him with editing functions like locating gates and wires, and facilities to modify the design by deleting and adding cells and wires on the screen by positioning the cursor on the desired point or points. The station also stores the design as it is drawn on the screen, and is capable of producing the
masks necessary to implement the design on silicon during the fabrication process.

The integration scheme is driven by a series of menus for both the image processing applications and the VLSI layout design application. These menus can communicate with the display computer, and access the image files on the disk, provide facilities for editing the displayed image and printing them on the colour plotter. The integration scheme is flexible and can accommodate other menus and other future applications.
Appendix A

PROGRAMS LISTING.

The following CAD program monitors the touch screen, and according to the touch position, it branches to different actions like locating or deleting cells, connecting or deleting wires of metal or polysilicon, and locating or deleting contact windows. It also controls the cursor movement and reads its position.
DUFFER(20)=4200F8
DUFFER(21)=4200F8
DUFFER(22)=4200F8
DUFFER(23)=4200F8
DUFFER(24)=4200F8
DUFFER(25)=4200F8
DUFFER(26)=4200F8
DUFFER(27)=4200F8
DUFFER(28)=4200F8
DUFFER(29)=4200F8
DUFFER(30)=4200F8
DUFFER(31)=4200F8
DUFFER(32)=4200F8
DUFFER(33)=4200F8
DUFFER(34)=4200F8
CALL AINRITE (FCB, LOCBUF, 68, *103, SNSBUF)
TYPE = "LUT DONE"
DUFFER(1)=4200F8
CALL AINRITE (FCB, LOCBUF, 2, *103, SNSBUF)

SET MODE CONTROL AND DEFINE RECTANGULAR LIMITS

DUFFER(1)=424043 !SET NODE CONTROL NORD
CALL AINRITE (FCB, LOCBUF, 2, *103, SNSBUF)
DUFFER(1)=425000
DUFFER(2)=425000+100
DUFFER(3)=425000
DUFFER(4)=427000+511
CALL AINRITE (FCB, LOCBUF, 8, *103, SNSBUF)

INITIALIZE GLOBAL VARIABLES

II=0; J=0; K=0; LAL=0; JJ=0; N=0; II=0; LL=0; MJ=0; JMK=0
KK=0; NNN=0
LOC=0; LV=0
OPEN (UNIT=1, FORM='FORMATED', FILE='SCALE')
READ (1, IMM7) L
CLOSE (1)
XLIM=400 244; YLIM=511 481

READ TOUCH POSITION

READ (7, 4016)(MK(I), I=1, 7)

4210 FORMAT (7X)
50 WRITE (0, 17) (MK(I), I=1, 7)
17 FORMAT ('0', 2X, 7X)
GO 15 I=2, 7
15 MK(I)=MK(I)*'36'
MK(2)=MK(2)*100; MK(3)*10+MK(4)
MK(5)=MK(5)*100; MK(6)*10+MK(7)
IF (0.01, 100) THEN
GO TO 22
ELSE
GO TO 1
END IF
20 IF(V, LT, 23) THEN
GO TO 41
ELSE
GO TO 40
END IF
40 IF(V, GE, 20) THEN
GO TO 50
ELSE
GO TO 1
END IF
50 IF(V, LT, 20) THEN
GO TO 60
ELSE
GO TO 70
END IF
60 IF(REF/, GE, 0) GO TO 1
III=III+1
IA(N)(III)=XPOS
IA(N)(III)=YPOS
NIR=2
GO TO 2003
70 IF(V, LT, 40) THEN
GO TO 60
ELSE
GO TO 30
END IF
80 IF(REF/, GE, 0) GO TO 1
J=J+1
I(N)(J)=XPOS
I(N)(J)=YPOS
NIR=2
GO TO 2003
90 IF(V, LT, GO) THEN
GO TO 100
ELSE
GO TO 110
END IF
100 IF(REF/, GE, 0) GO TO 1
K=K+1
DEL(X)=XPOS
DEL(X)=YPOS
GO TO 2081
110 IF(V, LT, GO) THEN
GO TO 120
ELSE
GO TO 130
END IF
120 IF(REF/, GE, 0) GO TO 1
L=L+1
DEH(R)(L)=XPOS
DEH(R)(L)=YPOS
DEH(R)(L)=XPOS
DEH(R)(L)=YPOS
DEH(R)(L)=YPOS
LON=3
POL=3
GO TO 2005
130 IF(V, LT, 130) THEN
GO TO 140
ELSE
GO TO 150
END IF

STORE THE DESIGN DATA

140  OPEN UNIT=1, FORM='FORMATED', FILE='KHAZN')
WRITE (L1.1) III, J.K.LAL, JJ, N.II, L, MM, NAM, KKK
IF (III.EQ.3) GO TO 7001
WRITE (L1.1) <XNAND(NM), NM=L III>
WRITE (L1.1) <YNAND(NM), NM=L III>
7001 IF (J.EQ.3) GO TO 7002
WRITE (L1.1) <XNAND(NM), NM=L JJ>
WRITE (L1.1) <YNAND(NM), NM=L JJ>
7002 IF (J.EQ.3) GO TO 7003
WRITE (L1.1) <XNOR(NM), NM=L J>
WRITE (L1.1) <YNOR(NM), NM=L J>
7003 IF (J.EQ.3) GO TO 7004
WRITE (L1.1) <XNOR(NM), NM=L NA>
WRITE (L1.1) <YNOR(NM), NM=L NA>
7004 IF (J.EQ.3) GO TO 7005
WRITE (L1.1) <DEL(NM), NM=L K>
WRITE (L1.1) <YDEL(NM), NM=L K>
7005 IF (III.EQ.3) GO TO 7006
WRITE (L1.1) <CNIREX(NM), NM=L II>
WRITE (L1.1) <CNIREY(NM), NM=L II>
WRITE (L1.1) <CNIREX(NM), NM=L II>
WRITE (L1.1) <CNIREY(NM), NM=L II>
7006 IF (LAL.EQ.3) GO TO 7007
WRITE (L1.1) <DENIREX(NM), NM=L LAL>
WRITE (L1.1) <DENIREY(NM), NM=L LAL>
WRITE (L1.1) <DENIREX(NM), NM=L LAL>
WRITE (L1.1) <DENIREY(NM), NM=L LAL>
7007 IF (L.EQ.3) GO TO 7008
WRITE (L1.1) <DPOLYX(NM), NM=L L L LL>
WRITE (L1.1) <DPOLYX(NM), NM=L L L LL>
WRITE (L1.1) <DPOLYX(NM), NM=L L L LL>
WRITE (L1.1) <DPOLYX(NM), NM=L L L LL>
7008 IF (MN.EQ.3) GO TO 7009
WRITE (L1.1) <EPOLYX(NM), NM=L MM>
WRITE (L1.1) <EPOLYX(NM), NM=L MM>
WRITE (L1.1) <EPOLYX(NM), NM=L MM>
WRITE (L1.1) <EPOLYX(NM), NM=L MM>
7009 IF (MN.EQ.3) GO TO 7010
WRITE (L1.1) <YDSHOB(NM), NM=L NNN>
WRITE (L1.1) <YDSHOB(NM), NM=L NNN>
7010 IF (K.EQ.3) GO TO 7011
WRITE (L1.1) <XSHOB(NM), NM=L KKK)
WRITE (L1.1) <XSHOB(NM), NM=L KKK)
7011 CONTINUE
21 FORMAT(1111)
CLOSE (1)
GO TO 200
150 IF (LT.128) THEN
GO TO 2002
ELSE

41  IF (XGT 0) THEN
    GO TO 51
  ELSE
    GO TO 1
  END IF
51  IF (YLT 20) THEN
    GO TO 61
  ELSE
    GO TO 71
  END IF
61  IF (XPOS.EQ.0) GO TO 1
    JJ=JJ+1
    XPOS(K,JJ)=XPOS
    YPOS(K,JJ)=YPOS
    NR=NR+1
    GO TO 2605
71  IF (YLT 40) THEN
    GO TO 61
  ELSE
    GO TO 71
  END IF
81  IF (POS.EQ.0) GO TO 1
    NN=NN+1
    XPOS(NN)=XPOS
    YPOS(NN)=YPOS
    NR=NR+1
    GO TO 2605
91  IF (YLT 60) THEN
    GO TO 81
  ELSE
    GO TO 111
  END IF
111  IF (TACH.EQ.0) GO TO 1
    II=II+1
    XPOS(K,II)=XPOS
    YPOS(K,II)=YPOS
    EWIREX(K,II)=XPOS
    EWIREY(K,II)=YPOS
    LON=16; FOL=0
    GO TO 2605
121  IF (YLT 80) THEN
    GO TO 121
  ELSE
    GO TO 131
  END IF
131  IF (CASP.EQ.0) GO TO 1
    LL=LL+1
    DRPOLYX(LL)=XPOS
    DRPOLYY(LL)=YPOS
    DEPOLYX(LL)=XPOS
    DEPOLYY(LL)=YPOS
    LON=0; FOL=1
    GO TO 2605
141  IF (YLT 100) THEN
    GO TO 141
  ELSE
    GO TO 151
  END IF
151  IF (CASP.EQ.0) GO TO 1
    MN=MN+1
    DRPOLYX(MN)=XPOS
    DRPOLYY(MN)=YPOS
    DEPOLYX(MN)=XPOS
    DEPOLYY(MN)=YPOS
    GO TO 2605
CALLED BY TOUCHING ENABLE CURSOR IN THE SECOND PAGE OF THE
VSDL MENU.

RETURN: XPOS, YPOS. UPPER LEFT COORDINATED OF A CELL
XPOS, YPOS, XPOS, YPOS BEGINNING & END COORD. OF A
WIRE. PASS, PNSP, PNSP ARE FLAGS SET TO ENABLE CALLING
SWITCH IS A VARIABLE SET WITHIN THE PROGRAM BY TOUCHING
THE FAST/SLOW AREA IN THE SCREEN, TO CONTROL THE SPEED
OF CURSOR MOVEMENT.

RECTANGULAR LIMITS

2032
CONTINUE
BUFTER(1)=425000
BUFTER(2)=425000+512
BUFTER(3)=42000
BUFTER(4)=42000+512
BUFTER(5)=425400
BUFTER(6)=425400+512
BUFTER(7)=426000
BUFTER(8)=426000+512
CALL AWRIITE(FCB, LOCBUF, 16, *103, SNSBUF)
SWITCH=1
LCX=0
LCY=0
BUFTER(1)=427000+LCX
BUFTER(2)=427000+LCY
CALL AWRIITE(FCB, LOCBUF, 1, *103, SNSBUF)
RCND(7,4016)(NK1), I=1,7)
3018
WRITE(8,17)(NK1), I=1,7)
DC 3015 I=2,7
3015
NK1=MK1-X-30'
X=MK2(100+MK3)*10+MK4
Y=MK5(100+MK6)*10+MK7
IF(X GT. 120) THEN
GO TO 3020
ELSE
GO TO 3001
END IF
IF (X GT 200) THEN
GO TO 3041
ELSE
GO TO 3030
END IF

IF (Y GE 120) THEN
GO TO 3050
ELSE
GO TO 3001
END IF

IF (Y LT 140) THEN
GO TO 3060
ELSE
GO TO 3070
END IF

IF (SWITCH GT 0) GO TO 3600
LCH=LCH+1
IF (LCH, GE, XLIM) GO TO 3001
BUFFER(1)=427000+LCH
CALL AWRITE(FCB, LOCBUF, 2, *103, SNSBUF)
GO TO 3001

LCH=LCH+25+L
IF (LCH, GE, XLIM) GO TO 3001
BUFFER(1)=427000+LCH
CALL AWRITE(FCB, LOCBUF, 2, *103, SNSBUF)
GO TO 3001

IF (Y LT 160) THEN
GO TO 3060
ELSE
GO TO 3090
END IF

IF (SWITCH GT 0) GO TO 3799
LOC=LOC+1
IF (LOC, GT, YLIM) GO TO 3001
BUFFER(1)=427000+LOC
CALL AWRITE(FCB, LOCBUF, 2, *103, SNSBUF)
GO TO 3001

LOC=LOC+48+L
IF (LOC, GE, YLIM) GO TO 3001
BUFFER(1)=427000+LOC
CALL AWRITE(FCB, LOCBUF, 2, *103, SNSBUF)
GO TO 3001

IF (Y LT 180) THEN
GO TO 3100
ELSE
GO TO 3110
END IF

XPOS=LCH
YPOS=LOC
GO TO 3001

IF (Y LT 200) THEN
GO TO 3120
ELSE
GO TO 3130
END IF

XPOS=LCH
YPOS=LOC
GO TO 3001

IF (Y LT 220) THEN
GO TO 3140
ELSE
GO TO 3150
END IF

XPOS=LCH
YPOS=LOC
PASS=1
GO TO 3209
IF(V.LT.240) THEN
GO TO 3100
ELSE
GO TO 3470
END IF
3468
WINN=LCX
WINN=LCY
WINN=I
GO TO 3209
GO TO 3201
3241
IF(V.GT.120) THEN
GO TO 3351
ELSE
GO TO 3201
END IF
3281
IF(V.LT.140) THEN
GO TO 32861
ELSE
GO TO 32871
END IF
3281
IF(SWITCH GT.0) GO TO 3281
LCV=LCX-1
IF(LCX LT.0) GO TO 3201
BUFFER(1)=I427890+LCX
CALL ASWRITE(FCB, LOCBUF, I2, *103, SNSBUF)
GO TO 3201
3201
LCX=LCY 26+I
IF(LCX LT.0) GO TO 3201
BUFFER(1)=I427890+LCX
3253
CALL ASWRITE(FCB, LOCBUF, I2, *103, SNSBUF)
GO TO 3201
3271
IF(V.LT.150) THEN
GO TO 3204
ELSE
GO TO 3281
END IF
3301
IF(SWITCH GT.0) GO TO 3301
LCV=LCY 1
IF(LCY LT.0) GO TO 3201
BUFFER(1)=I427890+LCY
CALL ASWRITE(FCB, LOCBUF, I2, *103, SNSBUF)
GO TO 3201
3301
LCV=LCY 40+I
IF(LCY LT.0) GO TO 3201
BUFFER(1)=I427890+LCY
3252
CALL ASWRITE(FCB, LOCBUF, I2, *103, SNSBUF)
GO TO 3201
3301
IF(V.LT.100) THEN
GO TO 3101
ELSE
GO TO 3111
END IF
3101
MPOS=LCX
MPOS=LCY
MACH=1
GO TO 3202
3111
IF(V.LT.2601) THEN
GO TO 3121
ELSE
GO TO 3111
END IF
3121
MPOS=LCX
MPOS=LCY
MACH=1
GO TO 3202
ROUTINE DEL TO DELETE CELLS ON THE SCREEN

2001 CONTINUE
LRL=WPOS
RRL=WPOS+COLCHL
URL=WPOS
BRL=WPOS+COLCHL
BUFFER(1)=DBLE300+LRL
BUFFER(2)=DBLE300+RRL
BUFFER(3)=DBLE300+URL
BUFFER(4)=DBLE300+BRL
BUFFER(5)=DBLE300
CALL NWRITE(POS, LOCBUF, 16, 103, SNBUF)
GO TO 1

ROUTINE CDD FOR CALCULATING GATE COORDINATES

2032 CONTINUE
WPOS=POS
XPOS=POS+X
YPOS=POS+Y
FORMAT(14)
OPEN (UNIT=4, FORM='FORMATTED', FILE='NAGCO')
FORMAT(TM)
RECTANGLE 1 GREEN
LRL=W110+L
RRL=W110+H
URL=Y100+L
BRL=Y100+H
COLOR=1
IF(NH(R,0)) GO TO 4001
URL=Y*(40-38)*L; BRL=Y*(40-38)*H

4001 XP=LRL+1
YP=URL+1
WRITE (L, 15) LRL, RRL, URL, BRL, XP, YP, COLOR
RECTANGLE 1D8R
LRL=W110+L
RRL=W110+H
URL=Y200+L; BRL=Y434+H
COLOR=1
IF(NH(R,0)) GO TO 4002
URL=Y*(40-34)*L; BRL=Y*(40-34)*H

4002 XP=LRL+1
YP=URL+1
WRITE (L, 15) LRL, RRL, URL, BRL, XP, YP, COLOR
RECTANGLE 2 GREEN
LRL=X+11+L
RRL=X+22+L
URL=Y+1+L
BRL=Y+10+L
COLOR=1
IF (XIR.CR.0) GO TO 4003
URL=Y+(40 10)+L
BRL=Y+(40 10)+L
4003 XP=LRL+1
YP=URL+1
WRITE (L,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 3 GREEN
LRL=X+20+L
RRL=X+32+L
URL=Y+10+L
BRL=Y+30+L
COLOR=1
IF (XIR.CR.0) GO TO 4004
URL=Y+(40 10)+L
BRL=Y+(40 10)+L
4004 XP=LRL+1
YP=URL+1
WRITE (L,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 3DAR
LRL=X+11+L
RRL=X+23+L
URL=Y+30+L
BRL=Y+30+L
COLOR=1
IF (XIR.CR.0) GO TO 4005
URL=Y+(40 20)+L; BRL=Y+(40-36)+L
4005 XP=LRL+1
YP=URL+1
WRITE (L,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 4 GREEN
LRL=X+12+L
RRL=X+14+L
URL=Y+14+L; BRL=Y+16+L
COLOR=1
IF (XIR.CR.0) GO TO 4006
URL=Y+(40 10)+L; BRL=Y+(40-1)+L
4006 XP=LRL+1
YP=URL+1
WRITE (L,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 4DAR
LRL=X+11+L
RRL=X+12+L
URL=Y+1+L
IF (LR, EQ. 1) GO TO 4828
LRL=X: 125=1L
RRL=X: 137=1L
URL=Y: 133=1L
BRL=Y: 125=1L
COLOR=4
IF (MIR, EQ. 0) GO TO 4160
URL=Y: (48 - 23)*L; BRL=Y: (48 - 23)*L
4160 XP=URL: 11
VP=URL: 11
WRITE (L, 1C) LRL, RRL, URL, BRL, XP, VP, COLOR
RECTANGLE 3 RED
4828 LRL=X
RRL=X: 137=1L
URL=Y: 133=1L
BRL=Y: 130=1L
COLOR=4
IF (MIR, EQ. 0) GO TO 4170
URL=Y: (48 - 23)*L; BRL=Y: (48 - 19)*L
4170 XP=URL: 11
VP=URL: 11
WRITE (1, 1C) LRL, RRL, URL, BRL, XP, VP, COLOR
RECTANGLE 30 RED
4818 LRL=X
RRL=X: 14=1L
URL=Y: 133=1L
BRL=Y: 130=1L
COLOR=4
IF (MIR, EQ. 0) GO TO 4818
URL=Y: (48 - 23)*L; BRL=Y: (48 - 36)*L
4818 XP=URL: 11
VP=URL: 11
WRITE (1, 1C) LRL, RRL, URL, BRL, XP, VP, COLOR
RECTANGLE 2 RED
4819 LRL=X: 12=1L
RRL=X: 14=1L
URL=Y: 133=1L
BRL=Y: 21=1L
COLOR=4
IF (MIR, EQ. 0) GO TO 4819
URL=Y: (48 - 21)*L; BRL=Y: (48 - 19)*L
4819 XP=URL: 11
VP=URL: 11
WRITE (1, 1C) LRL, RRL, URL, BRL, XP, VP, COLOR
RECTANGLE 20 RED
4784 LRL=X: 11=1L
RRL=X: 15=1L
108

WRITE (1,10) LRL, RRL, URL, BRL, XP, YP, COLOR}

RECTANGLE 11A BLUE

LRL=X+11*L; RRL=X+12*L; URL=Y+1*L; BRL=Y+4*L;
COLOR=1C
IF (HNR. EQ. 0) GO TO 4025
URL=Y+(40 4)*L; BRL=Y+(40-1)*L;
4025
XP=LRL+1
YP=URL+1
WRITE (1,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 11B BLUE

LRL=X+12*L; RRL=X+14*L; URL=Y+1*L; BRL=Y+1*L;
COLOR=1C
IF (HNR. EQ. 0) GO TO 4026
URL=Y+(40 4)*L; BRL=Y+(40-1)*L;
4026
XP=LRL+1
YP=URL+1
WRITE (1,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 11C BLUE

LRL=X+14*L; RRL=X+15*L; URL=Y+1*L; BRL=Y+4*L;
COLOR=1C
IF (HNR. EQ. 0) GO TO 4027
URL=Y+(40 4)*L; BRL=Y+(40-1)*L;
4027
XP=LRL+1
YP=URL+1
WRITE (1,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 11D BLUE

LRL=X+15*L; RRL=X+26*L; URL=Y+1*L; BRL=Y+4*L;
COLOR=1C
IF (HNR. EQ. 0) GO TO 4028
URL=Y+(40 4)*L; BRL=Y+(40-1)*L;
4028
XP=LRL+1
YP=URL+1
WRITE (1,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 12 BLUE

LRL=X; RRL=X+2*L; URL=Y+30*L; BRL=Y+34*L;
COLOR=1C
IF (HNR. EQ. 0) GO TO 4030
URL=Y+(40 34)*L; BRL=Y+(40-30)*L;
4030
XP=LRL+1
YP=URL+1
WRITE (1,10) LRL, RRL, URL, BRL, XP, YP, COLOR

RECTANGLE 12A BLUE

LRL=X; RRL=X+1*L; URL=Y+30*L; BRL=Y+34*L;
COLOR=1C
IF (HNR. EQ. 0) GO TO 4031
URL=Y+(40 34)*L; BRL=Y+(40-30)*L;
4031
XP=LRL+1
YP=URL+1
WRITE (1,10) LRL, RRL, URL, BRL, XP, YP, COLOR
RECTANGLE 12 BLUE
URL=X+14+1; RRL=X+12+1; URL=Y+14+1; BRL=Y+14+1;
COLOR=16
IF(MR, EQ. 0) GO TO 4042
URL=Y; (14-12)=L; BRL=Y+(40-12)=L
4042 XP=URL:1
YP=URL:1
WRITE (L, 16) LRL, RRL, URL, BRL, XP, YP, COLOR
RECTANGLE 13 BLUE
URL=X+11+1; RRL=X+12+1; URL=Y+14+1; BRL=Y+15+1;
COLOR=16
IF(MR, EQ. 0) GO TO 4043
URL=Y; (40-12)=L; BRL=Y+(40-12)=L
4043 XP=URL:1
YP=URL:1
WRITE (L, 16) LRL, RRL, URL, BRL, XP, YP, COLOR
RECTANGLE 14 BLUE
URL=X+11+1; RRL=X+12+1; URL=Y+14+1; BRL=Y+15+1;
COLOR=16
IF(MR, EQ. 0) GO TO 4044
URL=Y; (40-12)=L; BRL=Y+(40-12)=L
4044 XP=URL:1
YP=URL:1
WRITE (L, 16) LRL, RRL, URL, BRL, XP, YP, COLOR
RECTANGLE 15 BLUE
URL=X+11+1; RRL=X+12+1; URL=Y+14+1; BRL=Y+15+1;
COLOR=16
IF(MR, EQ. 0) GO TO 4045
URL=Y; (40-12)=L; BRL=Y+(40-12)=L
4045 XP=URL:1
YP=URL:1
WRITE (L, 16) LRL, RRL, URL, BRL, XP, YP, COLOR
RECTANGLE 16 BLUE
URL=X+11+1; RRL=X+12+1; URL=Y+14+1; BRL=Y+15+1;
COLOR=16
IF(MR, EQ. 0) GO TO 4046
URL=Y; (40-12)=L; BRL=Y+(40-12)=L
4046 XP=URL:1
YP=URL:1
WRITE (L, 16) LRL, RRL, URL, BRL, XP, YP, COLOR
1173. 000
1174. 000
1175. 000
1176. 000
1177. 000
1179. 000
1180. 000
1181. 000
1182. 000
1183. 000
1184. 000
1185. 000
1186. 000
1187. 000
1188. 000
1189. 000
1190. 000
1192. 000
1193. 000
1194. 000
1195. 000
1196. 000
1197. 000
1198. 000
1199. 000
1200. 000
1201. 000
1202. 000
1203. 000
1204. 000
1205. 000
1206. 000
1207. 000
1208. 000
1209. 000
1210. 000
1211. 000
1212. 000
1213. 000
1214. 000
1215. 000
1216. 000
1217. 000
1218. 000
1219. 000
1220. 000
1221. 000
1222. 000
1223. 000
1224. 000
1225. 000
1226. 000
1227. 000
1228. 000
1229. 000
1230. 000
1231. 000
1232. 000
1233. 000
1234. 000
1235. 000
1236. 000
1237. 000
1238. 000
1239. 000
1240. 000
1241. 000
1242. 000
1243. 000
1244. 000

ROUTINE GATE
ICGU-50 : NUMBER OF RECTANGLES IN A NAND GATE
SET RECTANGULAR AND CONIC LIMITS TO THE DIMENSIONS OF SCREEN
BUFF(0)=420000+LIXL;BUFF(1)=420100+L1Y1
BUFF(2)=420000
CALL RWRITE,(FCB,LOCBUF,20,*103,SNBUF)
GO TO 1
1000 CALL X:EXIT
200 CONTINUE
7 TYPEC = 'BREAK ERROR=', IERR
   GO TO 200
100 TYPEC = 'WRITE ERROR=', SNBUF
   MESG=8
   GO TO 200
200 TYPEC = 'OPEN ERROR=', IERR
200 CONTINUE
BUFF(1)=420045
CALL RWRITE,(FCB,LOCBUF,2,*,103,SNBUF)
STOP
END

*****************************************************************************

======== SUBROUTINE HINDOW

*****************************************************************************
 SUBROUTINE HINDOW(WINX,WINY,BUFFER,L)
 IMPLICIT INTEGER*2 (A-Z)
 DIMENSION BUFFER(35)
 BUFF(1)=425000+(WINX+L)
 BUFF(2)=425000+(WINX+2*L)
 BUFF(3)=425000+(WNY+L)
 BUFF(4)=425000+(WINY+2*L)
 BUFF(5)=427000+(WINX+L+1)
 BUFF(6)=427000+(WINX+2*L+1)
 DUBBER(7)=425000! LOAD FOREGROUND PIXEL VALUE
 DUBBER(8)=425000+8
 DUBBER(9)=427000! FILL
 RETURN
END

*****************************************************************************

======== SUBROUTINE DWINDOW

*****************************************************************************
 SUBROUTINE DWINDOW(WINX,WINY,BUFFER,L)
 IMPLICIT INTEGER*2 (A-Z)
 DIMENSION BUFFER(35)
 BUFF(1)=425000+(WINX+L)
 BUFF(2)=425000+(WINX+2*L)
 BUFF(3)=425000+(WINY+L)
 BUFF(4)=425000+(WINY+2*L)
 BUFF(5)=427000

==========================================================================
The listing of the program Mask4, which produces the fourth mask containing the contact windows, is given in the following pages.
IMPLICIT INTEGER*2 (A-Z)

INTEGER LOCBUF, IERR, ERR, FCB(16), DEVICE, TIME, SNSBUF
INTEGER LRL, RRL, URL, ERL, XP, YP, COLOR
DIMENSION BUFFER(35)
INTEGER XAND(40), YAND(40), XNAND(40), YNAND(40)
INTEGER XNOT(40), YNOT(40), XNOTM(40), YNOTM(40)
INTEGER XDEL(40), YDEL(40), EWIREFIX(80), EWIRIFY(80)
INTEGER EWIRIFY(80), EWIRIFY(80), DEWIREFIX(80)
INTEGER DEWIRIFY(80), DEWIRIFY(80), DEWIRIFY(80)
INTEGER BPOLEX(80), BPOLEY(80), EPOLEX(80), EPOLEY(80)
INTEGER DEPOLEX(80), DEPOLEY(80), DEPOLEX(80), DEPOLEY(80)
INTEGER XDISHOB(80), YDISHOB(80), XDISHOB(80), XDISHOB(80)
EQUIVALENCE (BUFFER(1), LOCBUF)
DATA DEVICE /"'7ER0'"
DATA FCB /"'22441141', 8, X'02000000', 1300000000000/ 
TIME = 10
CALL AVOPEN (DEVICE, FCB, *800, IERR)
TYPE = 'AVOPEN SUCCESSFUL'
CALL X:DMX (X'0000', ERR = 200)
CALL AVORR (FCB)

PROGRAM LOOK-UP TABLE

BUFFER(1) = 'X'0C68'; BUFFER(2) = '42001F'
BUFFER(3) = '420000'
BUFFER(4) = '420038'
BUFFER(5) = '42003F'
BUFFER(6) = '42003F'
BUFFER(7) = '420077'
BUFFER(8) = '420077'
BUFFER(9) = '420077'
BUFFER(10) = '420077'
BUFFER(11) = '420077'
BUFFER(12) = '420077'
BUFFER(13) = '4200FF'
BUFFER(14) = '4200FF'
BUFFER(15) = '4200FF'
BUFFER(16) = '4200FF'
BUFFER(17) = '4200FF'
BUFFER(18) = '4200FF'
BUFFER(19) = '4200FF'
BUFFER(20) = '4200FF'
BUFFER(21) = '4200FF'
BUFFER(22) = '4200FF'
BUFFER(23) = '4200FF'
BUFFER(24) = '4200FF'
BUFFER(25) = '4200FF'
BUFFER(26) = '4200FF'
BUFFER(27) = '4200FF'
BUFFER(28) = '4200FF'
BUFFER(29) = '4200FF'
BUFFER(30) = '4200FF'
BUFFER(31) = '4200FF'
BUFFER(32) = '4200FF'
BUFFER(33) = '4200FF'
BUFFER(34) = '4200FF'
BUFFER(35) = '4200FF'
CALL AWRITE (FCB, LOCBUF, 68, *103, SNBUF)
TYPE = ‘4UT DONE’
DUFFER(4) = 4200F8
CALL AWRITE (FCB, LOCBUF, 2, *103, SNBUF)

CCT MODE CONTROL AND DEFINE RECTANGULAR LIMITS

DUFFER(4) = 4200F8 ! SET MODE CONTROL WORD
CALL AWRITE (FCB, LOCBUF, 2, *103, SNBUF)
DUFFER(1) = 4200F0
DUFFER(2) = 4200FC+100
DUFFER(3) = 4200F8
DUFFER(4) = 4200FC+51.1
CALL AWRITE (FCB, LOCBUF, 8, *103, SNBUF)
OPEN (UNIT=1, FORM='FORMATTED', FILE='SCALE')
READ (L.4017) L

4017 FORMAT(14)
CLOSE (1)

L10
OPEN(UNIT=1, FORM='FORMATTED', FILE='KHAIN')
READ (L.21) III, J, K, LAL, JJ, N, II, LL, MM, NNR, KKK
WRITE(G.21) III, J, K, LAL, JJ, N, II, LL, MM, NNR, KKK
IF (III.EQ.0) GO TO 7001
READ (L.21) (XAND(NM), NM=1, III)
READ (L.21) (YAND(NM), NM=1, III)
WRITE(G.21) (XAND(NM), NM=1, III)
WRITE(G.21) (YAND(NM), NM=1, III)

7001 IF (J.EQ.0) GO TO 7002
READ (L.21) (XNAM(NM), NM=1, J)
READ (L.21) (YNAM(NM), NM=1, J)
IF (J.EQ.0) GO TO 7003
READ (L.21) (XNAM(NM), NM=1, J)
READ (L.21) (YNAM(NM), NM=1, J)
7003 IF (N.EQ.0) GO TO 7004
READ (L.21) (XNOT(NM), NM=1, N)
READ (L.21) (YNOT(NM), NM=1, N)
WRITE(G.21) (XNOT(NM), NM=1, N)
WRITE(G.21) (YNOT(NM), NM=1, N)

7004 IF (K.EQ.0) GO TO 7005
READ (L.21) (XDEL(NM), NM=1, K)
READ (L.21) (YDEL(NM), NM=1, K)
IF (K.EQ.0) GO TO 7006
READ (L.21) (XDEL(NM), NM=1, K)
READ (L.21) (YDEL(NM), NM=1, K)
7005 IF (II.EQ.0) GO TO 7006
READ (L.21) (XIREX(NM), NM=1, II)
READ (L.21) (YIREX(NM), NM=1, II)
READ (L.21) (XIREX(NM), NM=1, II)
READ (L.21) (YIREX(NM), NM=1, II)
7006 IF (LAL.EQ.0) GO TO 7007
READ (L.21) (XIREX(NM), NM=1, LAL)
CLOSE (1)
CONTINUE.
IF(K.EQ.0) GO TO 3005
DO 3005 NN=1,K
URL=VDC1(NM)
RRL=RL;L+0.4L
URL=VDC1(NM)
DRL=URL+0.4L
BUFFER(1)=425000+URL
BUFFER(2)=425000+RRL
BUFFER(3)=425000+DRL
BUFFER(4)=425000+BRL
BUFFER(5)=425002
CALL NVICUTE(FCD, LOGbuf, 10, +163, SNSBUF)
CONTINUE
IF(KK=K.EQ.0) GO TO 3006
DO 3006 NN=1,KK
WIND = VXHOD(NEW)
WINY= VXVOD(NEW)
CALL WINDOW(WIND, WINY, BUFFER, L).
CALL NVICUTE(FCD, LOGbuf, 18, +103, SNSBUF)
CONTINUE
IF(NNN.EQ.0) GO TO 3007
DO 3007 NN=1,NN
WIND = VXHOD(NEW)
WINY= VXVOD(NEW)
CALL WINDOW(WIND, WINY, BUFFER, L).
CALL NVICUTE(FCD, LOGbuf, 18, +103, SNSBUF)
CONTINUE
GO TO 300
100 CALL N:PRKICT
205 CONTINUE.
TYPE='BREK ERROR=', ERR
GO TO 300
103 TYPE='WRITE ERROR=', SNSBUF
GO TO 300
000 TYPE='OPEN ERROR=', IERR
900 CONTINUE.
BUFFER(17)=428048
CALL NVICUTE (FCD, LOGbuf, 2, +103, SNSBUF)
STOP
END
**************************************************************
SUBROUTINE C0D: FOR CALCULATING GATE COORDINATES
**************************************************************
**************************************************************
SUBROUTINE C0D (XPOS, YPOS, MIR, L, L, L)
IN81 INTT INTTIP (A-7)
0356 000
0357 000
0358 000
0359 000
0360 000
0361 000
0362 000
0363 000
0364 000
0365 000
0366 000
0367 000
0368 000
0369 000
0370 000
0371 000
0372 000
0373 000
0374 000
0375 000
0376 000
0377 000
0378 000
0379 000
0380 000
0381 000
0382 000
0383 000
0384 000
0385 000
0386 000
0387 000
0388 000
0389 000
0390 000
0391 000
0392 000
0393 000
0394 000
0395 000
0396 000
0397 000
0398 000
0399 000
0400 000
0401 000
0402 000
0403 000
0404 000
0405 000
0406 000
0407 000
0408 000
0409 000
0410 000
0411 000
0412 000
0413 000
0414 000
0415 000
0416 000
0417 000
0418 000
0419 000
0420 000
0421 000
(REM) COO, WCOC

TYPE = "OK COO A"

4047 FORMAT(14)
OPEN (UNIT=1, FORM='FORMATTED', FILE='WACO')
10 FORMAT(FI13)

RECTANGLE 14, WHITE

1000

LRL=X+10+L; URL=Y+2+L; BRL=Y+3+L
COLOR= O,
IF(MNR, CO. 0) GO TO 4049
URL=Y+(40-2)+L
4049 XP=URL+1
VP=URL+1
WRITE (1,1G) LRL, RRL, URL, BRL, XP, VP, COLOR,

RECTANGLE 15 WHITE

LRL=X+12+L; RRL=X+1+L; URL=Y+13+L; BRL=Y+14+L
COLOR= O
IF(MNR, CO. 0) GO TO 4050
URL=Y+(40-13)+L
4050 XP=URL+1
VP=URL+1
WRITE (1,1G) LRL, RRL, URL, BRL, XP, VP, COLOR,

RECTANGLE 16 WHITE

LRL=X+10+L; RRL=X+14+L; URL=Y+14+L; BRL=Y+15+L
COLOR= O
IF(MNR, CO. 0) GO TO 4051
URL=Y+(40-14)+L
4051 XP=URL+1
VP=URL+1
WRITE (1,1G) LRL, RRL, URL, BRL, XP, VP, COLOR,

RECTANGLE 17 WHITE

LRL=X+10+L; RRL=X+14+L; URL=Y+15+L; BRL=Y+16+L
COLOR= O
IF(MNR, CO. 0) GO TO 4052
URL=Y+(40-15)+L
4052 XP=URL+1
VP=URL+1
WRITE (1,1G) LRL, RRL, URL, BRL, XP, VP, COLOR,

RECTANGLE 18 WHITE

LRL=X+11+L; RRL=X+15+L; URL=Y+31+L; BRL=Y+33+L
COLOR= O
IF(MNR, CO. 0) GO TO 4053
URL=Y+(40-31)+L
4053 XP=URL+1
VP=URL+1
WRITE (1,1G) LRL, RRL, URL, BRL, XP, VP, COLOR,

RECTANGLE 19 WHITE

LRL=X+10+L; RRL=X+3+L; URL=Y+37+L; BRL=Y+39+L
COLOR= O
IF(MNR, CO. 0) GO TO 4054
URL=Y+(40-37)+L
4054 XP=URL+1
VP=URL+1
WRITE (1,1G) LRL, RRL, URL, BRL, XP, VP, COLOR,
IF(LN. EQ. 1) GO TO 4899
LRL(X) = LRL(Y) RRL(X+9) URL(Y+37) FLR(LRL(Y)+33) L
COLOR = 0
IF(NFIR. EQ. 0) GO TO 4855
URL(Y) = Y(48 30) BRL = Y(48 37) + L
4855 X' = LRL(X) Y' = URL(Y)
WRITE (L, 1G) LRL, RRL, URL, BRL, XP', Y', COLOR
RECTANGLE 12 WHITE
4865 CONTINUE
LRL(X) = LRL(X) RRL(X+9) URL(Y+37) BRL = Y(48 38) + L
COLOR = 0
IF(NFIR. EQ. 0) GO TO 4856
URL(Y) = Y(48 30) BRL = Y(48 37) + L
4856 X' = LRL(X) Y' = URL(Y)
WRITE (L, 1G) LRL, RRL, URL, BRL, XP', Y', COLOR
CLOSE (L)
RETURN
' END

SUBROUTINE WINDOW
SUBROUTINE WINDOW(WINK, WINY, BUFFER, L)
IMPLICIT INTEGER (A-Z)
DIMENSION BUFFER(35)
BUFFER(1) = 425550 + (WINK+L)
BUFFER(2) = 425550 + (WINK+L)
BUFFER(3) = 425550 + (WINY+L)
BUFFER(4) = 425550 + (WINK+L)
BUFFER(5) = 425550 + (WINK+L+1)
BUFFER(6) = 425550 + (WINY+L+1)
BUFFER(7) = 425550 ! LOAD FOREGROUND PIXEL VALUE
BUFFER(8) = 425550 + 8
BUFFER(9) = 425550 ! FILL
TYPE = 'OK WINDOW'
RETURN
END

SUBROUTINE DWINDOW
SUBROUTINE DWINDOW(WINK, WINY, BUFFER, L)
IMPLICIT INTEGER (A-Z)

SUBROUTINE DWINDOW(WINK, WINY, BUFFER, L)
IMPLICIT INTEGER (A-Z)
DIMENSION BUFFER(35)
BUFFER(1) = 125000 + (WINX + L)
BUFFER(2) = 125000 + (WINX + 2 + L)
BUFFER(3) = 125000 + (WINY + L)
BUFFER(4) = 125000 + (WINY + 2 + L)
BUFFER(5) = 125000
RETURN
END
Decision Structures for Page 1 of the Menu System.
C
C
C
************
TSI
************
DIMENSION TEM(3)
INTEGER M(10)
DATA TEM /1,2,3/
READ(7,16) (M(I),I=1,7)
FORMAT (7A1)
16
30 WRITE (8,17) (M(I),I=1,7)
17 FORMAT ('D',2X,7A1)
DO 15 I=2,7
15 M(I)=M(I)-X'30'
X=M(2)*100+M(3)*10+M(4)
Y=M(5)*100+M(6)*10+M(7)
IF (X.GT.200) THEN
GOTO 40
ELSE
GO TO 999
END IF
40 IF (Y.GE.05) THEN
GO TO 50
ELSE
GO TO 999
END IF
50 IF (Y.LT.050) THEN
GO TO 60
ELSE
GO TO 69
END IF
60 OPEN(UNIT=1,FORM='UNFORMATTED',FILE='NA1')
WRITE (1) (TEM(I),I=1,3)
CLOSE(UNIT=1)
GO TO 1000
69 IF (Y.LT.060) THEN
GO TO 999
ELSE
GO TO 70
END IF
70 IF (Y.LT.105) THEN
GO TO 80
ELSE
GO TO 89
END IF
80 OPEN(UNIT=1,FORM='UNFORMATTED',FILE='NA2')
WRITE(1) (TEM(I),I=1,3)
CLOSE (UNIT=1)
GO TO 1000
Fortran Decision Structure for Page 1 of the Menu System.
$JOB MENU, ABU.SLOF=DUMMY
$RUN GWINDSOR(ABU)TS1.SU
$IFT FILE NA1 B1
$IFT FILE NA2 B2
$IFT FILE NA3 B3
$IFT FILE NA100 B100
$GOTO END %B1
$DELETE NA1
$SELECT MENUDSP
$GOTO END %B2
$DELETE NA2
$SELECT MENUULSI
$GOTO END %B100
$DELETE NA100
$SELECT MENU %B3
$DELETE NA3
$GOTO END %END
$EOJ

MPX-32 Command File for Page 1 of the Menu System.
REFERENCES


VITA AUCTORIS

1955  Born on August 24th, Alexandria, Egypt.


1977  Graduated with a B.Sc. degree in Electrical Engineering, from the University of Alexandria, Alexandria, Egypt.

1978  Joined the Communications & Signals branch in the Egyptian Navy (Compulsory Service).

1979  Joined Panco company, Alexandria, Egypt as a Chief Electronic Engineer on board of Al-Esraa Ship.

1980  Joined Samed pipeline company, Alexandria, Egypt as an Instrumentation Engineer.

1984  Candidate for the M.A.Sc. degree from the Electrical Engineering Department, University of Windsor, Ontario, Canada.