

To the Graduate Council:

I am submitting herewith a thesis written by Youngsup Kim entitled "Computer-Aided Design of Digital Systems Using Pattern Recognition and Menu Selection Techniques." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

D. W. Bouldin

D. W. Bouldin, Major Professor

We have read this thesis
and recommend its acceptance:

R. C. Gonzalez

R. E. Bolensheimer

Accepted for the Council:

C. W. Munkel

Graduate School

COMPUTER-AIDED DESIGN OF DIGITAL SYSTEMS
USING PATTERN RECOGNITION AND
MENU SELECTION TECHNIQUES

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Youngsup Kim
December 1983

ACKNOWLEDGEMENTS

I wish to express my appreciation to my major advisor, Dr. D. W. Bouldin for his thoughtful guidance. I also extend great appreciation to the other members of my graduate committee, Drs. R. C. Gonzalez, and R. E. Bodenheimer, for their discussions and suggestions on the subject of this thesis.

Special thanks go to Drs. J. M. Googe, and I. Alexeff for their concern and encouragement.

In addition, I am indebted to A. Perez for his enthusiastic help on programming in the TI-990 computer, and to M. Abidi for his discussion on problems involved in pattern recognition techniques.

Thanks also must go to my wife, Higyung, for her encouragement and patience, and to my daughter, Goeun, for her love.

Finally, I appreciate the moral support given by my parents, Mr. Doohwan Kim and Mrs. Eunju Yang.

ABSTRACT

A computer program producing a net list, pin assignment list, check list and list of materials needed for the computer-aided circuit design (CAD) process from a hardware diagram has been designed and implemented in Pascal. In addition, an approach to recognize hand-drawn schematic diagrams has been considered and carried out as one of the very early stages of complete design automation (DA) for very large scale integration (VLSI) circuits. Algorithms for discriminating between connection lines and gate symbols, identifying the type of symbols are presented. Further improvements with these approaches are discussed.

TABLE OF CONTENTS

SECTION	PAGE
1. INTRODUCTION	1
2. USER'S GUIDE FOR IC-MAP	4
2.1 General	4
2.2 IMAP	7
2.3 CMAP	12
2.4 Results	18
3. SOFTWARE DEVELOPMENT	20
3.1 Key Indexed File Structure	20
3.2 Data Structure	23
3.3 Sorting Schemes	26
3.4 Procedures in IC-MAP	31
3.4.1 External Procedures	31
3.4.2 Internal Procedures	34
4. RECOGNITION OF HAND-DRAWN SCHEMATIC DIAGRAM	37
4.1 General	37
4.2 Preprocessing	40
4.3 Filtering	43
4.3.1 Local Properties of Digital Picture	45
4.3.2 Shrinking and Thinning Algorithms	50
4.3.3 Filtering	51
4.4 Feature Extraction and Classification	52
4.4.1 Membership Functions	55
4.4.2 Separation of Lines and Symbols	59
4.4.3 Classification of Symbols	61
5. CONCLUDING REMARKS	67
BIBLIOGRAPHY	70
APPENDICES	74
APPENDIX A	75
APPENDIX B	84
APPENDIX C	88
VITA	111

LIST OF FIGURES

FIGURE	PAGE
2.1. An example wiring diagram	5
2.2. MAIN menu selection	9
2.3. An example display of 74LS00	9
2.4. LIBRARY menu set	10
2.5. Selection of "CRE" in Library mode	10
2.6. TTL menu set	11
2.7. Selection of "NAND" in TTL or MAIN menu	13
2.8. Selection of "RAM" in MAIN menu	13
2.9. Selection of "USER" in MAIN menu	14
2.10. Creation of a device in User mode	15
2.11. VDT display when executing CMAP	19
3.1. Main program and procedures of IMAP	21
3.2. Main program and procedures of CMAP	22
3.3. Some data structures extracted from IC-MAP	25
3.4. Bubble sort with two keys in CMAP	28
3.5. Quicksort implementation in CMAP	30
3.6. Sorting scheme using the linked list	32
4.1. Design process of a circuit	38
4.2. Recognition procedures	39
4.3. Logic symbols to be recognized	41
4.4. A hand-drawn schematic diagram	42

FIGURE	PAGE
4.5. The output binary image after thresholding	44
4.6. Neighborhood of a pixel $P(x,y)$	46
4.7. Several types of connectivity	48
4.8. Some details of the binary image	53
4.9. Membership functions	57
4.10. An example of membership grades	58
4.11. Separation of gates from lines	60
4.12. Parameters in a symbol	63
4.13. Classification of symbols	65
4.14. The final output of the schematic diagram	66
A.1. Interconnection between devices	77
A.2. Primary selection	79
A.3. Secondary selection with MENU 1	81
A.4. Secondary selection with MENU 2	81
A.5. Some experimental output figures	83
B.1. Output documents : net list	85
B.2. Output documents : list of material	86
B.3. Output documents : check list	86
B.4. Output documents : pin assignment list	87

1. INTRODUCTION

Computer-aided design (CAD) techniques developed over the past decade are now finding widespread use in the design, manufacture and test of electronic assemblies. In the electronic industries, many computer-aided design tools have been developed to help engineers who need to draw schematic diagrams and write the corresponding net lists, lists of materials, etc. Actually, a schematic drawing is only a by-product, but obviously the visual geometry of the schematic is a useful aid to the engineer's intuition and ability to predict the circuit's performance. In practice, the usual process to produce such a diagram is to have the engineer draw the sketches needed and forward them to a draftsman who might redraw them manually or use a digitizing tablet to enable a computer to drive a plotter. It should be noted that the information needed to implement the given diagrams must be extracted from hand-drawn schematics of the design engineer has to list them.

Initially, the author developed the Fortran program KPATH (refer to Appendix A for details) which aids the designer in drawing schematic circuit diagrams employing a user-oriented CAD system. The program KPATH, however, does not have the capability of generating the net lists, lists of materials, checking lists, etc., needed for production or for automatic placement and routing. In essence, a complete CAD software package must consist not only of the capability of aiding design at the functional block and

schematic diagram level, but also of providing other design documents that organize the design information into appropriate formats for further processing in various production departments.

Considering paractical applications, KPATH leaves many tools to be desired such as (1) a library containing part outline templates available for schematic drawing, (2) a symbol-creation subroutine with which the engineer may create special symbols needed, (3) a window function that contarcts or expands the dispaly such that the active drawing fits the screen, (4) documentation generation routines which extract information from the data files captured in the schematic design step and produces all the documents needed in the production phase, and (5) tools providing for automatic palcement and routing of chips, logic simulation, and timing analysis.

Using a library of parts provides a powerful aid to the designer. Since the dimension and pin numbers of a chip or component are standardized, a template library can contain the exact template of a chip with the fields (pin connections, signal names, circuit designation and part numbers) ready to be filled in by the engineer. Since the program KPATH is impractical and time-consuming, an approach to eliminate these problems has been undertaken on a TI-990 computer system which does not support a graphics environment. Hence, a computer program, named Integrated Circuit Map (IC-MAP), producing a net list, check list, pin assignment list, and statistics needed for the automatic design

process from the hardware diagram level has been designed and implemented in Pascal. The output lists of IC-MAP not only enable engineers to check their design at a glance, but also can be used for further processing such as logic simulation, automatic placement, layout, wiring, timing analysis, and other design needs.

An additional software package has been developed by the author to recognize hand-drawn schematic. Since the schematic level diagram itself provides a great deal of information concerning circuit function in a very simple form and has been the easiest way represent the circuit of a designer's ideas, an approach to recognize a hand-drawn schematic diagram has been considered and carried out as an early stage at the fully automated CAD or design automation (DA) for VLSI circuitry. The procedures presented recognize a hand-drawn schematic diagram by discriminating between connection lines and gate symbols and identifying the gate type such as AND, OR, NAND, NOR, and INVERTER.

The following sections describe in detail; (1) a user's guide for IC-MAP, (2) software description of IC-MAP, (3) recognition algorithm for a hand-drawn schematic diagram and (4) concluding remarks include suggestions for further improvements of IC-MAP and a discussion of various problems confronted with the computer recognition of hand-drawn circuit diagrams.

2. USER'S GUIDE FOR IC-MAP

2.1 General

The Pascal program IC-MAP to be executed with Texas Instrument Pascal (TIP) on the TI-990 computer system consists of two separate modules. The first, named IMAP, not only enables a user to create and edit the integrated circuit specifications, but also allows creation of an output key indexed file which can be used as an input file to produce all other output files. The other module called CMAP generates a net list, pin assignment list, check list, and list of materials corresponding to the key indexed file generated by IMAP.

To use IMAP and CMAP, first of all the user has to prepare a pin-assignment wiring diagram. An example is shown in Figure 2.1. After obtaining this diagram, the following steps should be taken at least once. Note that .RES.RES999 is an arbitrary user ID, <CR> means a carriage return, and * indicates the prompt of the computer.

(1) Assign synonyms as shown below for your convenience.

```
[ ] AS
  SYNONYM: DIR
  VALUE: .RES.RES999
  SYNONYM: SOURCE
  VALUE: DIR.SOURCE
  SYNONYM: OBJ
  VALUE: DIR.OBJECT
  SYNONYM: LIST
  VALUE: DIR.LIST
  SYNONYM: LINK
  VALUE: DIR.LINK
  SYNONYM: OUTFILE
  VALUE: DIR.OUTFILE
```

(2) Create file directories for SOURCE, OBJECT, LIST, LINK,

Program images as an example shown below.

```
[] CFDIR
                                PATHNAME: DIR
                                MAX ENTRIES: 10
                                DEFAULT PHYSICAL RECORD SIZE: <CR>
```

```
[] CFDIR
                                PATHNAME: SOURCE
                                MAX ENTRIES: 30
                                DEFAULT PHYSICAL RECORD SIZE: <CR>
```

Repeat for OBJECT, LIST and LINK.

```
[] CFPRO
                                PATHNAME: DIR.PRO
                                MAX NUMBER OF TASKS: 255
                                MAX NUMBER OF PROCEDURES: 255
                                MAX NUMBER OF OVERLAYS: 255
                                INITIAL ALLOCATION: 85
                                SECONDARY ALLOCATION: <CR>
                                EXPANDABLE?: YES
```

(3) Copy source programs in .RES.RES549.

```
[] CC
  INPUT ACCESS NAME(S): .RES.RES549.SOURCE.IMAP
  OUTPUT ACCESS NAME: SOURCE.IMAP
  REPLACE?: NO
  MAXIMUM RECORD LENGTH: <CR>
```

```
[] CC
  INPUT ACCESS NAME(S): .RES.RES549.LINK.IMAP
  OUTPUT ACCESS NAME: LINK.IMAP
  REPLACE?: NO
  MAXIMUM RECORD LENGTH: <CR>
```

Repeat the same steps for CMAP.

(4) Compile source programs.

```
[] XTIP
  SOURCE: SOURCE.IMAP
  OBJECT: OBJ.IMAP
  LISTING: LIST.IMAP
  MESSAGES: <CR>
  OPTIONS: <CR>
  MEM1: <CR>
  MEM2: <CR>
  MEM3: <CR>
```

Repeat the same steps of XTIP for CMAP.

(5) Link object programs.

```
[ ] XLE
      CONTROL ACCESS NAME: LINK.IMAP
      LINKED OUTPUT ACCESS NAME: DIR.PRO
      LISTING ACCESS NAME: LIST.IMAP
      PRINT WIDTH: 80
```

Repeat the same steps for CMAP.

2.2 IMAP

The simple circuit of Figure 2.1 is used to illustrate the execution of IC-MAP. The user should take the following steps:

(1) Create a key indexed file by using a command CFKEY. This file written by IMAP will be an input file for CMAP. An example is shown below.

```
[ ] CFKEY
      PATHNAME: OUTFILE.TESTMAP
      LOGICAL RECORD LENGTH: 300
      PHYSICAL RECORD LENGTH: 768
      INITIAL ALLOCATION: <CR>
      SECONDARY ALLOCATION: <CR>
      MAXIMUM SIZE: 100
      START POSITION: 1
      KEY LENGTH: 4
      DUPLICATES?: N
      MODIFIABLE?: N
      ANY MORE KEYS?: NO
```

Here, the pathname format should be like
<.RES.RES999.OUTFILE.filename>.

(2) Now IMAP can be executed.

```
[ ] XPT
      PROGRAM FILE: DIR.PRO
      TASK NAME OR ID: IMAP
      INPUT: <CR>
      OUTPUT: <CR>
```

```
MESSAGES: <CR>
MODE (F,B,D): FOREGROUND
MEMORY: 10,1
```

(3) Enter a key indexed file name when prompted from the video display terminal (VDT). Note that this key indexed file must be created with CFKEY command before executing IMAP.

[] Enter a output key filename to create/edit your IC design.

```
This file will be input file for CMAP (eg: TESTMAP):
TESTMAP
KEY FILE PATHNAME: .RES.RES999.OUTFILE.TESTMAP
```

(4) THE VDT then displays the MAIN menu selection as shown in Figure 2.2. Note that any item in the menu sets can be chosen by the first two characters regardless of the length of string in the item. Type a device name you want to use in order to make sure the device exists in the Library (e.g.: 74LS00). If it exists, the VDT shows the configurations (Figure 2.3), otherwise, it displays some instructions. If it is necessary to correct or create a specification or configuration of that chip in the Library, enter "LIB" to get in the Library mode. Then the VDT will show the Library menu set on the screen as shown in Figure 2.4. Figure 2.5 shows the terminal ready to be filled in by the user after typing "CRE".

(5) Some menus in the MAIN selection and in "TTL" menu set help the user at the functional level (Figure 2.6). When the user does not know the specific device name or number, the program lists all the device names performing that function. Note that the user must change the source program or IMAP to modify the list of devices and

***** GENERATION OF HARDWARE DIAGRAM. *****

```
[ ] LIBRARY : CREATE & DELETE DATA IN LIBRARY.
[ ] USER : EDIT,CREATE,DELETE & DISPALY USER FILE.
[ ] DIS : DISPLAY AN IC SPECIFICATION (Default).
[ ] HEL : HELP |
[ ] BYE : EXIT THE PROGRAM (IMAP)
```

```
[0] TTL : TI TTL SERIES
[1] MIC : MICROPROCESSOR
[2] RAM : RAM
[3] ROM : ROM
[4] PRO : PROM
[5] EPR : EPROM
[6] PLA : PLA
[7]
[8]
```

[*** MAIN MENU ***]

[] Please enter an option or IC type no. : *

Figure 2.2. MAIN menu selection.

Type Number:74LS00
 Discription:QUAD.2-INPUT POS.NAND GATES
 Area(sq.in):0.24335
 Price(\$) :0.85
 Ty.Power.mW:2.55
 Mx.Power.mW:
 Ty.Delay.ns:9.5
 Mx.Delay.ns:
 No. of pins:14

74LS00			
---- 1	1A	Vcc	14----
---- 2	1B	4B	13----
---- 3	1Y	4A	12----
---- 4	2A	4Y	11----
---- 5	2B	3B	10----
---- 6	2Y	3A	9----
---- 7	GND	3Y	8----

<CR> to continue *

Figure 2.3. An example display of 74LS00.

GENERATION OF HARDWARE DIAGRAM.

```
[ ] CRE : CReate an IC data in LIBRARY.  
[ ] EDI : EDit an IC data in LIBRARY.  
[ ] DEL : DElete an IC data in LIBRARY.  
[ ] DIS : DIsplay an IC data in LIBRARY.  
[ ] HEL : HELP |  
[ ] QUI : QUIT editing LIBRARY FILE.
```

[*** LIBRARY MENU ***]

[] Please enter an option or IC type no. : *

Figure 2.4. LIBRARY menu set.

```
[ ] Enter data as followings ;  
Type Number:  
Discription:  
Area(sq.in):  
Price( $ ) :  
Ty.Power.mW:  
Mx.Power.mW:  
Ty.Delay.ns:  
Mx.Delay.ns:  
No. of pins:  
PIN 1 : I/O/C/N/:I  
  
< Prompt will show up as the number of pins >  
< Enter I for input, O for output, C for open collector  
or tri-state and N for no internal connection pin >
```

Figure 2.5. Selection of "CRE" in Library mode.

GENERATION OF HARDWARE DIAGRAM.

[0.]	NAN	:	NAND
[1.]	NOR	:	NOR
[2.]	AND	:	AND
[3.]	OR	:	OR
[4.]	INV	:	INVERTER
[5.]	XOR	:	EXCLUSIVE OR
[6.]	FFP	:	FLIP-FLOP
[7.]	ADD	:	ADDER
[8.]	MUL	:	MULTIPLIER
[9.]	SHI	:	SHIFT REGISTER
[10.]	DAT	:	MULTIPLXER/DATA SELECTOR

[*** TTL MENU ***]

[] Please enter an option or IC type no. : *

Figure 2.6. TTL menu set.

these menu sets. Figure 2.7 and Figure 2.8 illustrate the selections of "NAND" and "RAM", respectively.

(6) After checking all the devices in the Library, type "QUIT" to leave Library mode and obtain User mode by typing "USER". The VDT will show up as shown in Figure 2.9. Now, the user can create, delete, edit and display their design by selecting the corresponding menu. Figure 2.10 shows the steps to record one device (e.g.: U01) of the hardware diagram (see Figure 2.1). Repeat the same steps for each chip with the same methods. Each device implemented here may be inspected with the display function (DIS) in the User mode and the device designator (e.g.: U01) uniquely assigned.

The user may return to the MAIN menu selection by selecting "QUI" in any mode.

2.3 CMAP

Since the time-consuming and tedious job of entering data from the circuit diagram has been done in IMAP, The documentation-generation routine, CMAP, simply processes the data produced by IMAP. IMAP can be executed as shown below.

```
[ ] XPT
    PROGRAM FILE: DIR.PRO
    TASK NAME OR ID: CMAP
        INPUT: <CR>
        OUTPUT: <CR>
        MESSAGES: <CR>
    MODE (F,B,D): FOREGROUND
        MEMORY: 12,1
```

Then all the user needs to do is to enter the file names and the

- [1] QUAD 2-INPUT POSITIVE-NAND GATES
7400, 74H00, 74L00, 74LS00, 74S00
5400, 54H00, 54L00, 54LS00, 54S00
- [2] TRI. 3-INPUT POSITIVE-NAND GATES
74S10, 74H10, 74LS10, 7410, 74L10
54S10, 54H10, 54LS10, 5410, 54L10
- [3] DUAL 4-INPUT POSITIVE-NAND GATES
74S20, 74H20, 74LS20, 7420, 74L20
54S20, 54H20, 54LS20, 5420, 54L20
- [4] 8-INPUT POSITIVE-NAND GATES
74S30, 74H30, 74LS30, 7430, 74L30
54S30, 54H30, 54LS30, 5430, 54L30
- [5] QUAD, 2-INPUT POSITIVE-NAND GATES(OPEN COL.OUTPUTS)
7401, 74LS01
5401, 54LS01, 54L01

[] Please enter an option or IC type no. : *

Figure 2.7. Selection of "NAND" in TTL or MAIN menu.

- [1] 256-BIT RAM(TRI-STATE OUTPUT) (256x1)
74S201, 54S201
- [2] 256-BIT RAM(OPEN COL.OUTPUT) (256x1)
74S301, 54S301
- [3] 64-BIT WORDS RAM(OPEN COL.OUTPUT) (16x1)
74S289, 7489, 54S289

[] Please enter an option or IC type no. : *

Figure 2.8. Selection of "RAM" in MAIN menu.

GENERATION OF HARDWARE DIAGRAM.

```
[ ] CRE : CReate an IC data in USER FILE.  
[ ] EDI : EDit   an IC data in USER FILE.  
[ ] DEL : DElete an IC data in USER FILE.  
[ ] DIS : DISplay an IC data in USER FILE.  
[ ] HEL : HELP |  
[ ] QUI : QUIT editing USER FILE.
```

[*** USER MENU ***]

[] Please enter an option or IC type no. : *

Figure 2.9. Selection of "USER" in MAIN menu.

```
[ ] Enter an IC type number to put in file: 74LS00
[ ] Enter QU to quit making file.
```

Figure 2.10. Creation of a device in User mode.
Step 1 : Selection of "CRE".

PAGE*

Type Number:74LS00
Discription:QUAD.2-INPUT POS.NAND GATES
Area(sq.in):0.24335
Price(\$) :0.85
Ty.Power.mW:2.55
Mx.Power.mW:
Ty.Delay.ns:9.5
Mx.Delay.ns:
No. of pins:14

74LS00

----	1	1A	Vcc	14----
----	2	1B	4B	13----
----	3	1Y	4A	12----
----	4	2A	4Y	11----
----	5	2B	3B	10----
----	6	2Y	3A	9----
----	7	GND	3Y	8----

```
[ ] Please, enter a page number (eg: PAGE5) :
[ ] Enter QU to quit making file.
```

Figure 2.10. (Continued)
Step 2 : VDT shows 74LS00 spec.,
asking page no.

Type Number:74LS00
 Discription:QUAD.2-INPUT POS.NAND GATES
 Area(sq.in):0.24335
 Price(\$) :0.85
 Ty.Power.mW:2.55
 Mx.Power.mW:
 Ty.Delay.ns:9.5
 Mx.Delay.ns:
 No. of pins:14

74LS00			
---- 1	1A	Vcc	14----
---- 2	1B	4B	13----
---- 3	1Y	4A	12----
---- 4	2A	4Y	11----
---- 5	2B	3B	10----
---- 6	2Y	3A	9----
---- 7	GND	3Y	8----

*

[] Enter a circuit designator (eg: U12).
 [] Enter QU to quit making file.

Figure 2.10. (Continued)
 Step 3 : Enter a circuit designator.

Type Number:74LS00
 Discription:QUAD.2-INPUT POS.NAND GATES
 Area(sq.in):0.24335
 Price(\$) :0.85
 Ty.Power.mW:2.55
 Mx.Power.mW:
 Ty.Delay.ns:9.5
 Mx.Delay.ns:
 No. of pins:14

*

74LS00			
---- 1	1A	Vcc	14----
---- 2	1B	4B	13----
---- 3	1Y	4A	12----
---- 4	2A	4Y	11----
---- 5	2B	3B	10----
---- 6	2Y	3A	9----
---- 7	GND	3Y	8----

U01

[] Enter signal names on pins.
 [] Begin I/O signal form/to external with * or @

Figure 2.10. (Continued)
 Step 4(a) : Enter signal names.

PAGE0

Type Number:74LS00
 Discription:QUAD.2-INPUT POS.NAND GATES
 Area(sq.in):0.24335
 Price(\$) :0.85
 Ty.Power.mW:2.55
 Mx.Power.mW:
 Ty.Delay.ns:9.5
 Mx.Delay.ns:
 No. of pins:14

74LS00

X1	----	1	1A	Vcc	14-----	Vcc
Q0	----	2	1B	4B	13-----	D11
D11	----	3	1Y	4A	12-----	D12
X2	----	4	2A	4Y	11-----	D13
-Q0	----	5	2B	3B	10-----	X2
D12	----	6	2Y	3A	9-----	X1
GND	----	7	GND	3Y	8-----	D14

U01

[] Enter signal names on pins.
 [] Begin I/O signal form/to external with * or @

Figure 2.10. (Continued)
 Step 4(b) : Signal names entered.

title of the design following the prompts as illustrated in Figure 2.11. Each time the file name is typed in, the pathname of the file will be automatically displayed on the terminal. The length of a file name is limited to eight characters.

2.4 Results

The output files, such as the net list, pin assignment list, check list and list of materials, generated by CMAP can be obtained on the terminal or line printer.

(1) On the terminal

```
[ ] SF
    FILE PATHNAME:  OUTFILE.<filename>
```

(2) On the line printer

```
[ ] PF
    FILE PATHNAME(S):  OUTFILE.<FILENAME>
    ANSI FORMAT?:     NO
    LISTING DEVICE:    LP01
    DELETE AFTER PRINTING?:  NO
    NUMBER OF LINES/PAGE:  <CR>
```

(3) List the directory of the output files.

```
[ ] LD
    PATHNAME:  DIR.OUTFILE
    LISTING ACCESS NAME:  <CR>
```

Figures B.1-B.4 in Appendix B, respectively, show the net list, list of material, check list and pin assignment list of the circuit diagram previously shown in Figure 2.1, page 5.

```

[] Enter a source key filename to make all the output files
   (eg: TESTMAP) : *
[] Enter a title for your design (eg: Up-down Octal Counter)

[] Enter a filename to save net list (eg:NETMAP) :      NET
[] Enter a filename to save list of material(eg:LISTMAP):  LIST
[] Enter a filename to save check list (eg:CHECKMAP) :    CHECK
[] Enter a filename to save list of pins (eg:PINMAP) :     PIN

OUTPUT FILE PATHNAME : .RES.RES999.OUTFILE.NET
OUTPUT FILE PATHNAME : .RES.RES999.OUTFILE.LIST
OUTPUT FILE PATHNAME : .RES.RES999.OUTFILE.CHECK
OUTPUT FILE PATHNAME : .RES.RES999.OUTFILE.PIN

```

Figure 2.11. VDT display when executing CMAP.

3. SOFTWARE DEVELOPMENT

The basic rules of IC-MAP are (1) the output of a gate cannot be connected with the output(s) of some gates except the open-collector and tri-state outputs which are commonly used to form buses in TTL and MOS systems, and (2) except for external input/output signals, an input signal must have a driving source and an output signal must connect to an input line.

Pascal was chosen for the development of the software package IC-MAP due to its character manipulation capability and its portability. In the following sections, IMAP and CMAP are documented in detail to allow for further modifications. Figures 3.1-3.2 show the profile of IC-MAP.

3.1 Key Indexed File Structure

Texas Instrument Pascal (TIP) provides a special file structure called key indexed file (KIF) [6]. The key indexed file allows random access to its records via a key. The key is a character string of as many as 100 characters, located in a fixed position within each file record. One to fourteen individual keys can be specified. For example, the records in a student file can be accessed by keys that indicate the student's name, date of birth, and social security number.

A single procedure, KEY\$FILE, enables a TIP program to perform a variety of operations on key indexed files. Each call to KEY\$FILE

```

Program IMAP;                                (* FILENAME : SOURCE.IMAP *)

(* External Procedures and function *)
KEY$FILE          SET$ACNM          FIND$SYN
INITSCREEN        DISPLAY          ACCEPT
CLEARSCREEN       STATIONID        SVC$

(* Internal Procedures *)
PROCEDURE Charin( character:string2; VAR intnum:integer );
PROCEDURE Clearword( VAR word:PACKED ARRAY[1..?]OF char );
PROCEDURE Realnumcheck( word:string; VAR character:string2 );
PROCEDURE Inchar( intnum:integer; VAR character:string2 );
PROCEDURE VDT;
PROCEDURE Mainmenu;
PROCEDURE Functionalmenu;
PROCEDURE Librarymenu;
PROCEDURE Editmenu;
PROCEDURE Userresponse( VAR menu:selection;
                        VAR menus:menuset; VAR key:string );
PROCEDURE Underline( row,col,len:integer );
PROCEDURE Hline( row,col,len:integer; lmode:linemode );
PROCEDURE Vline( row,col,len:integer; lmode:linemode );
PROCEDURE Box( crow,ccol,width,height:integer );
PROCEDURE Specdisplay;
PROCEDURE Icdisplay;
PROCEDURE Sigdisplay;
PROCEDURE Checksig;
PROCEDURE Getsignal;
PROCEDURE Checksig( VAR signal:string );
PROCEDURE Specaccept( VAR jsize:integer );
PROCEDURE Specline( k:integer; VAR key:string );
PROCEDURE Readlib;
PROCEDURE T74display;
PROCEDURE Writelib;
PROCEDURE Edit;
PROCEDURE Openfiles;
PROCEDURE Makesci( f:PACKED ARRAY[1..?]OF char; n:integer;
                    VAR s:sci_string );
PROCEDURE Closefiles;
PROCEDURE Maincases;

BEGIN (* Main program *)
    VDT;
    Openfiles;
    Maincases;
    Closefiles;
END; (* Main program *)

```

Figure 3.1. Main program and procedures of IMAP.

```

Program CAMP;                                (* FILENAME : SOURCE.CMAP *)

(* External Procedures and function *)
KEY$FILE      SET$ACNM      FIND$SYN
INITSCREEN    DISPLAY      ACCEPT
CLEARSCREEN    STATIONID    SVC$

(* Internal Procedures *)
PROCEDURE Charin( character:string2; VAR intnum:integer );
PROCEDURE Writeline( line:half_line );
PROCEDURE Clearword( VAR word:PACKED ARRAY[1..?]OF char );
PROCEDURE Inchar( intnum:integer; VAR character:string2 );
PROCEDURE Charreal( word:string; VAR realnum:real );
PROCEDURE VDT;
PROCEDURE Openfiles;
PROCEDURE Concatenate( first:string; second:char;
                      VAR ksort:string9 );
PROCEDURE Readnet( VAR sigcount:integer );
PROCEDURE Kswap( VAR a,b:string9 );
PROCEDURE Swap( VAR a,b:t_signal );
PROCEDURE Quicksort( lowerbound,upperbound );
PROCEDURE Bubblesort( lowerbound,upperbound );
PROCEDURE Writefiles;
PROCEDURE Materiallist;
PROCEDURE Closefiles;

BEGIN (* Main program *)
  VDT;
  Openfiles;
  Readnet( sigcount );
  Quicksort( 1,sigcount );
  Writefiles;
  Bubblesort( 1,sigcount );
  Materiallist;
  Closefiles;
END; (* Main program *)

```

Figure 3.2. Main program and procedures of CMAP.

includes a command code specifying the operation to be performed. The following paragraphs describe the use of KEY\$FILE.

(1) Procedure KEY\$FILE has a single entry point for all calls. All calls to KEY\$FILE must have specify the following four parameters: (a) K_COMMAND--specifies which operation is to be done, (b) K_KEYNAME--indicates the value of the key to be processed, (c) K_BUFNAME--provides the name of the buffer that will provide or receive data, and (d) K_STS_BK--indicates the user control block.

(2) The type required and procedure format used to call KEY\$FILE are described in the following example:

```
TYPE
  T_K_STS_BK = PACKED RECORD
    K_STS_CODE : 0..255; (* return code *)
    K_KEY_NO   : 0..255; (* key   to be accessed *)
    K_ACCESS   : 0..255; (* file access flag *)
    K_RESERVED : 0..255; (* reserved *)
    K_RECORD_L : integer; (* record size *)
    K_PATHNAME : PACKED ARRAY[1..30] OF char
                  (* file name *)
  END;

PROCEDURE KEY$FILE( K_COMMAND:integer;
  UNIV K_KEYNAME:PACKED ARRAY[1..?] OF char;
  UNIV K_BUFNAME:PACKED ARRAY[1..?] OF char;
  VAR K_STS_BK :T_K_STS_BK ); EXTERNAL;
```

The key indexed files have been used to keep the IC specification in the Library file and user input data file in the IC-MAP program. Some parameters of KIF's used in IC-MAP are described in the next section.

3.2 Data Structure

Some variables of key importance in the data structure are organized as records which can contain a variety of information

about those variables. The three major records INPART, OUTPART, and T_SIGNAL shown in Figure 3.3 are described as follows:

(1) INPART : This record is used to store an IC specification;

(a) NAME has the type number of the device (e.g.: 74LS00). This field is used as the key word to access the Library file which contains the updated IC specification using the key indexed file structure (b) REFER may have 40 characters detailing the functional description of NAME (e.g.: QUAD 2-INPUT NAND GATES). (c) AREA and COST contain data corresponding to unit cost and area, respectively. (d) TYPCUR, MAXCUR, TYPDEL, MAXDEL are the typical maximum current dissipation and typical, maximum time delay in each gate or device, respectively. (e) SIZE indicates the number of pins of NAME (e.g.: 14). (f) INOUT contains the input/output characteristic of a pin. The value of INOUT must be one of followings: I for input, O for output, C for open collector or tri-state, and N for no internal connection pin.

(2) OUTPART : The record of OUTPART stores some information which the user types in during the design process: (a) CIRDES is the circuit designator (e.g.: U05) which is used as the key word to access the output key file. (b) PAGE is the page number of the design. (c) NAME has the type number or name of the device. (d) SIGNAL is ARRAY[1..maxpin] containing all the signal names corresponding to the pins.

(3) T_SIGNAL : Once NAME containing the signal name is known,

```

CONST  K_INIT =1;
       K_OPEN = 3;
       K_READ BY KEY_LOCKED = 10;
       K_INSERT_REC = 19;
       K_REWRITE_REC_UN = 21;
       nmax = 80;
       minpin = 14;
       blank = ' ';
       speccol = 13;
       str2len = 2;
       crow = 12;
       fnamelen= 8;

       K_TERM = 2;
       K_CLOSE = 4;
       K_READ BY KEY = 9;
       K_REWRITE_REC = 20;
       K_DELETE_BY_KEY = 24;
       half = 40;
       maxpin =24;
       wordlen = 8;
       str4len = 4;
       str1len = 1;
       ccol = 40;
       constflen =20;

TYPE   byte = 0..#FF;
       btype = ARRAY[1..16] OF integer;
       string =PACKED ARRAY[1..wordlen] OF char;
       string4=PACKED ARRAY[1..str4len] OF char;
       string2=PACKED ARRAY[1..str2len] OF char;
       string1=PACKED ARRAY[1..str1len] OF char;
       halfline=PACKED ARRAY[1..half] OF char;
       fname = PACKED ARRAY[1..fnamelen] OF char;
       inpart = RECORD
           name : string;           refer: halfline;
           area : string;          cost : string;
           typcur : string;        maxcur : string;
           typdel : string;        maxdel : string;
           size : string2;
           pindes : ARRAY[1..maxpin] OF string4;
           inout : ARRAY[1..maxpin] OF string1;
       END;
       outpart = RECORD
           circes : string4;        page : string1;
           name : string;
           signal : ARRAY[1..maxpin] OF string
       END;

       t_signal = RECORD
           name :string;            inout :string1;
           circes:string4;          ictype:string;
           size :string2;           pinnum:string2;
           page :string1;
       END;

```

Figure 3.3. Some data structures extracted from IC-MAP.

the other information may be accessed by this record. ICTYPE has the type number or name of the device.

3.3 Sorting Schemes

In order to generate a net list, pin assignment list, check list and list of materials, the documentation-generation routine, CMAP, must process information extracted from the data files which were having all information of each signal must be stored by appropriate fields of the record to find out the relation between signals or to list pin assignments in order.

Sorting is one of the most important operations in data processing. Generally speaking, the sorting problem is to arrange a sequence of records so the values of their key fields form a nondecreasing sequence. That is, given record $r_1, r_2, \dots, r_n$, with $k_1, k_2, \dots, k_n$, respectively, we must produce the same records in an order $r_{i_1}, r_{i_2}, \dots, r_{i_n}$, such that $k_{i_1} \leq k_{i_2}, \dots, k_{i_n}$. Many sorting algorithms have been developed with various execution times. However, some of the principal internal sorting algorithms, called bubble sort, quicksort and linked list sorting, are implemented and will be discussed. Then more efficient sorting algorithms which can be implemented with VLSI circuit are introduced. These fast sorting algorithms could be used to improve the performance of CMAP.

The simplest sorting methods is bubble sort, which has an $O(n^2)$ running time, in both the worst case and average case [7-8]. The basic idea behind bubble sort is that the records to be sorted are

kept in an array held vertically. Making repeated passes over the array from bottom to top, two adjacent elements are reversed if they are out of order. The bubble sort algorithm is described below, and the example used in CMAP in Figure 3.4.

```

PROCEDURE Swap( VAR x,y: recordtype );
  (* Swap exchanges the records of x and y *)
  VAR temp : recordtype;
  BEGIN
    temp:=x;      x:=y;      y:=temp;
  END (* Swap *)

PROCEDURE Bubblesort( VAR A:ARRAY[m..n] of recordtype );
  (* Bubblesort sorts an array into nondecreasing order
    on key *)
  VAR i,j : integer;
  BEGIN
    FOR i:=m TO n-1 DO
      FOR j:=n DOWNTO i+1 DO
        A[j-1].k>A[j].k THEN Swap(A[j],A[j-1]);
      END; (* Bubblesort *)
    END;
  END;

```

Another sorting scheme with a very good average behavior developed by C. A. R. Hoare is the quicksort scheme [7]. The quicksort with an average running time of $O(n \log n)$ sorts an array $A[1], A[2], \dots, A[n]$ by picking some key value k in an array as a pivot element around which to rearrange the elements in the array. Hopefully, the pivot is near the median key value in the array, so it is preceded by about half the keys and followed by half. We permute the elements in the array so that for some j , all the records with keys less than k appear in $A[1], \dots, A[j]$, and all those with keys k or greater appear in $A[j+1], \dots, A[n]$. Since all keys in the first group precede all keys in the second group, the entire array will thus be sorted. The quicksort algorithm can

```

PROCEDURE Swap( VAR x,y:t_signal );
(* Exchange the two records of x, y *)

VAR temp:t_signal;
BEGIN
    temp:=x;          x:=y;          y:=temp;
END

PROCEDURE Bubblesort( lowerbound,upperbound:integer );
(* Bubble sort sorts array netsignal by name and inout
   in order of N,C,O,I *)

VAR i,j : integer;
BEGIN (* Bubblesort *)
    FOR i:lowerbound TO upperbound-1 DO
        BEGIN
            FOR j:=upperbound DOWNTO i+1 DO
                BEGIN
                    WITH netsignal[j] DO
                        BEGIN
                            (* First key comparison *)
                            IF name<netsignal[j-1].name THEN
                                Swap( netsignal[j],netsignal[j-1] );
                            (* Second key comparison *)
                            IF name=netsignal[j-1].name THEN
                                BEGIN
                                    IF inout[1]='O' AND netsignal[j-1].inout[1]='I'
                                        THEN Swap(netsignal[j],netsignal[j-1]);
                                    IF inout[1]='C' AND (netsignal[j-1].inout[1]='I'
                                        OR netsignal[j-1].inout[1]='O')
                                        THEN Swap(netsignal[j],netsignal[j-1]);
                                    IF inout[1]='N' AND netsignal[j-1].inout[1]<>'N'
                                        THEN Swap(netsignal[j],netsignal[j-1]);
                                END;
                            END; (* WITH *)
                        END; (* j *)
                    END; (* upperbound i *)
                END;
            END; (* Procedure Bubblesort *)
        END;
    END;

```

Figure 3.4. Bubble sort with two keys in CMAP.

be described as follows, and the example used in CMAP is shown in Figure 3.5.

```

PROCEDURE Quicksort( A:ARRAY[m..n] of recordtype );
  (* Quicksort sorts array A into nondecreasing order
    on key k *)
  VAR i,j : integer;      pivotkey : keytype;
  BEGIN
    IF m<n THEN
      BEGIN
        i:=m;      j:=n+1;      pivotkey:=A[i].k;
        WHILE i<j DO
          BEGIN
            REPEAT i:=i+1 UNTIL A[i].k>=pivotkey;
            REPEAT j:=j+1 UNTIL A[j].k>=pivotkey;
            if i<j THEN Swap( A[i],A[j] );
          END;
          Swap( A[m],A[j] );
          Quicksort( m,j-1 );
          Quicksort( j+1,n );
        END;
      END;
    END;
  END;

```

The quicksort algorithm and the examples described previously have assumed that the pivot key is the very first element of each subfile while sorting. The quicksort in Figure 3.5 sorts by two keys, NETSIGNAL[n].NAME and INOUT. These keys are concatenated to form the sort key. The order of INOUT is N/C/O/I, because the order is used to find out the relation between signals.

There is another "speed up" to quicksort that is actually a trade of space for time. If space is available we can create array pointers to the records in array A, then make the comparisons between the keys of records pointed to. But instead of moving the records we move the pointers to records in the same way quicksort moves the records themselves. At the end, the pointers, read in left-to-right order, point to the records in the desired order [9].

```

PROCEDURE Concatenate( first:string; second:char;
                      VAR ksort:string9 );
VAR   i : integer;
BEGIN
  FOR i:=1 TO 8 DO ksort[i]:=first;
  CASE second OF
    'N' : ksort[9]:='A';
    'C' : ksort[9]:='B';
    'O' : ksort[9]:='C';
    'I' : ksort[9]:='D';
  END;
END;

PROCEDURE Kswap( VAR a,b:PACKED ARRAY[1..9] OF char;
VAR   c:PACKED ARRAY[1..9] OF char;
BEGIN   c:=a;    a:=b;    b:=c;          END;
PROCEDURE Swap( VAR a,b:t_signal );
VAR   c:t_signal;
BEGIN   c:=a;    a:=b;    b:=c;          END;

PROCEDURE Quicksort( lowerbound,upperbound:integer );
VAR   i,j:integer; key:PACKED ARRAY[1..9] OF char;
BEGIN (* Quicksort *)
  IF lowerbound < upperbound THEN
    BEGIN
      i:=lowerbound                j:=upperbound+1
      key:=sortkey[lowerbound];
      WHILE i<j DO
        BEGIN
          REPEAT i:=i+1 UNTIL sortkey[i]>=key;
          REPEAT j:=j-1 UNTIL sortkey[j]<=key;
          IF i<j THEN
            THEN
              BEGIN
                Swap( netsignal[i],netsignal[j] );
                Kswap( sortkey[i],sortkey[j] );
              END;
            END;
          Swap( netsignal[lowerbound],netsignal[j] );
          Kswap( sortkey[lowerbound],sortkey[j] );
          Quicksort( lowerbound,j-1 );
          Quicksort( j+1,upperbound );
        END
      END; (* Quicksort *)

```

Figure 3.5. Quicksort implementation in CMAP.

The sorting scheme using the linked list structure is implemented as an example (Figure 3.6).

These sorting schemes mentioned above could be implemented in VLSI technique for the fast sorting. With the same idea of the linked list sorting scheme a new sorting scheme called Parallel Enumeration Sort for VLSI was proposed recently [10]. In this scheme, we make only n exchanges of records, instead of $O(n \log n)$ exchanges, which makes a substantial difference if the records are large. Another sorting algorithm, called Pipe-lined Sinking Sort algorithm, proposed by Perez [11] has almost the same performance as the Parallel Enumeration Sorting scheme. The Sinking Sort algorithm, however, will require less communication between cells in VLSI implementation.

3.4 Procedures in IC-MAP

In this section, all procedures and functions in IC-MAP are briefly described including some built-in external procedures and functions in TIP.

3.4.1 External Procedures [6]

(1) SET\$ACNM specifies the actual access name for a file.

(2) FIND\$SYN obtains the value of an system command interpreter (SCI) synonym. In order to call the procedure, place the maximum length of the value string (as limited by the size of the array into which it is placed) into element 0 of the value of string.

```

(* ----- Ponter type sort ----- *)

(* TYPE declaration *)
t_signal = RECORD
    name : string;          inout : string1;
    circes: string4;        ictype: string;
    size : string2;         pinnum: string2;
    page : string1;
END;
t_sigpointer = @t_signode;
t_signode = RECORD
    siginfo:t_signal;      link:t_sigpointer;
END;

(* VAR declaration *)
copy,head : t_sigpointer;

PROCEDURE Insertnode( netsignal:t_signal;
                     VAR q:t_sigpointer; r:t_sigpointer);
BEGIN
    NEW(q@.link);
    WITH q@.link@ DO
    BEGIN
        siginfo:=netsignal;      link:=r
    END; (* WITH *)
END;

PROCEDURE Listinsert( VAR listhead:t_sigpointer;
                     netsignal:t_signal );
VAR last,next:t_sigpointer;
BEGIN
    next:=listhead; (* update last and next pointer *)
    WHILE netsignal.name>=next@.siginfo.name AND
        next@.link<>nil DO
    BEGIN
        last:=next;      next:=next@.link;
    END; (* WHILE *)
    (* Insert new netsignal data in the list *)
    IF netsignal.name>=next@.siginfo.name
    THEN (* Attach to the end of the list after next node *)
        Insertnode( netsignal,next,nil )
    ELSE
        IF netsignal.name<next@.siginfo.name
        THEN Insertnode( netsignal,last,next );
    END;
END;

```

Figure 3.6. Sorting scheme using the linked list.

```

PROCEDURE Read_sort( VAR listhead:t_sigpointer );
VAR i,pincount,index : integer;
    netsignal:t_signal; outicpart:outpart; inicpart:inpart;
    first:boolean;      istring:string2;
BEGIN
    OUTICPART.CIRDES:='0000';      FIRST:=TRUE;
    KEY$FILE(k_read_by_key,outicpart.cirdes,outicpart,ok_sbk );
    WHILE ok_sbk.k_scode=0 THEN
    BEGIN
        KEY$FILE(k_read_by_key,outicpart.name,inicpart,k_sbk);
        Charin( inicpart.size,pincount );
        IF first THEN
        BEGIN
            first:=false;          NEW(listhead);
            WITH listhead@.siginfo,listhead@ DO
            BEGIN
                FOR index:=1 TO wordlen DO name[index]:=' ';
                inout[1]:=' ';      cirdes:=outicpart.cirdes;
                pinnum:=istring;    ictype:=outicpart.name;
                size:=inicpart.size; page:=outicpart.page;
                link:=nil;
            END;
        END; (* IF THEN *)
        FOR i:=1 TO pincount DO
        BEGIN
            Inchar(i,istring);
            WITH netsignal DO
            BEGIN
                name:=outicpart.signal[i]; inout:=inicpart.inout[i];
                cirdes:=outicpart.cirdes;  ictype:=outicpart.name;
                size:=inicpart.size;      pinnum:=istring;
                page:=outicpart.page;
            END;
            Listinsert( listhead,netsignal );
        END; (* FOR pincount *)
    END; (* IF *)
    END; (* WHILE *)
END; (* Read_sort *)

(* ----- Main program section ----- *)
Read_sort(head);      copy:=head@.link;
WHILE copy<>nil DO
    BEGIN
        WITH copy@.siginfo DO
        BEGIN
            Writeln(nfile,name);      copy:=copy@.link
        END;
    END; (* End of list *)

```

Figure 3.6. (Continued)

(3) INITSCREEN initializes the data structure needed to perform I/O to the video display terminal (VDT Model 911). This call must precede any other VDT I/O procedures.

(4) CLEARSCREEN clears the screen of the VDT.

(5) DISPLAY displays a given string on the terminal at the specified location. The resolution is 24(rows)x80(columns).

(6) ACCEPT reads in a string value from the VDT at the specified location.

(7) Function STATIONID obtains the station-identifier of the station associated with the task.

3.4.2 Internal Procedures

(1) Function CHARIN changes a string type variable into the value of integer.

(2) CLEARWORD clears a specified length of word.

(3) REALNUMCHECK checks if an accepted word is a real number.

(4) INCHAR changes an integer number into the string type number.

(5) MAINMENU displays the set of MAIN menu selections on the terminal.

(6) FUNCTIONALMENU displays the set of the TTL menu.

(7) LIBRARYMENU displays the set of selections used in Library mode.

(8) EDITMENU displays the set of menu used in User mode.

(9) USERRESPONSE prompts the user to select a menu and accepts

the response.

(10) UNDERLINE draws a line on the terminal.

(11) HLINE draws a horizontal line by using a graphic character on the VDT.

(12) VLINE draws a vertical line by using a graphic character on the VDT.

(13) BOX outlines a rectangle used for drawing the template of an IC chip.

(14) SPECDISPLAY displays the specification of a given IC chip.

(15) ICDISPLAY displays an IC chip with the pin numbers and names.

(16) SIGDISPLAY displays the signal names of a specified IC designator.

(17) GETSIGNAL accepts the net names from the terminal.

(18) SPECACCEPT accepts the specification of an IC chip from the terminal.

(19) T74DISPLAY displays the specification, pin names and number of a given chip.

(20) READLIB reads the Library file and shows if the given device exists.

(21) WRITELIB writes the Library file with the data entered by user.

(22) EDIT handles all functions in the User mode.

(23) OPENFILES opens all files used in the program.

(24) CLOSEFILES closes all files used in the program.

(25) MAKESCI makes a given string to fit the structure of sci_string.

(26) MAINCASES orients the program according to the user's request.

(27) READNET reads the two key files, the Library file and the user's input key indexed file, then makes the records of NETSIGNAL.

(28) SWAP and KSWAP exchanges the content of two records and keys, respectively.

(29) QUICKSORT sorts the records of NETSIGNAL[n] by two keys, NETSIGNAL[n].NAME and NETSIGNAL[n].INOUT, with the well-known quicksort algorithm.

(30) BUBBLESORT sorts the records of NETSIGNAL[n] by two keys, ICTYPE and CIRDES, with the simple bubble sort scheme.

(31) WRITEFILES writes some output files.

(32) MATERIALLIST writes the list of materials.

(33) CHECKSIG surpresses blank characters in a signal name.

4. RECOGNITION OF A HAND-DRAWN SCHEMATIC DIAGRAM

4.1 General

IC design methodology ranges from totally manual to fully automated procedures. Computer aids for design have been developed on the basis of these two extremes. The overall scope of the design process is from the problem specification to the final realization, as shown in Figure 4.1. A totally automated design may be achieved by developing a set of tools to transform a problem specification into a circuit structure. Automation of the transformation from the behavioral description to the circuit structure is still a research topic [13-14].

Computerized methods to recognize hand-drawn logic circuit diagrams have been developed in conjunction with the improvement of computer technology. CAD systems may be classified as Type 1 or Type 2, depending on their input devices (Figure 4.1). A Type 1 system features a graphic interface and, by means of pattern recognition techniques, feeds the schematics to the system, thus minimizing man-machine interaction. Although such an effective recognition system has not been developed yet, there is active research in this area [15-17]. A Type 2 system is the most common system in use today. It uses a digitizer or a light-pen to feed data into CAD system. Therefore designers must digitize the points, one by one, on a hand-drawn diagram according to a menu composed of a list of strings and/or graphical symbols. Figure 4.2 outlines the

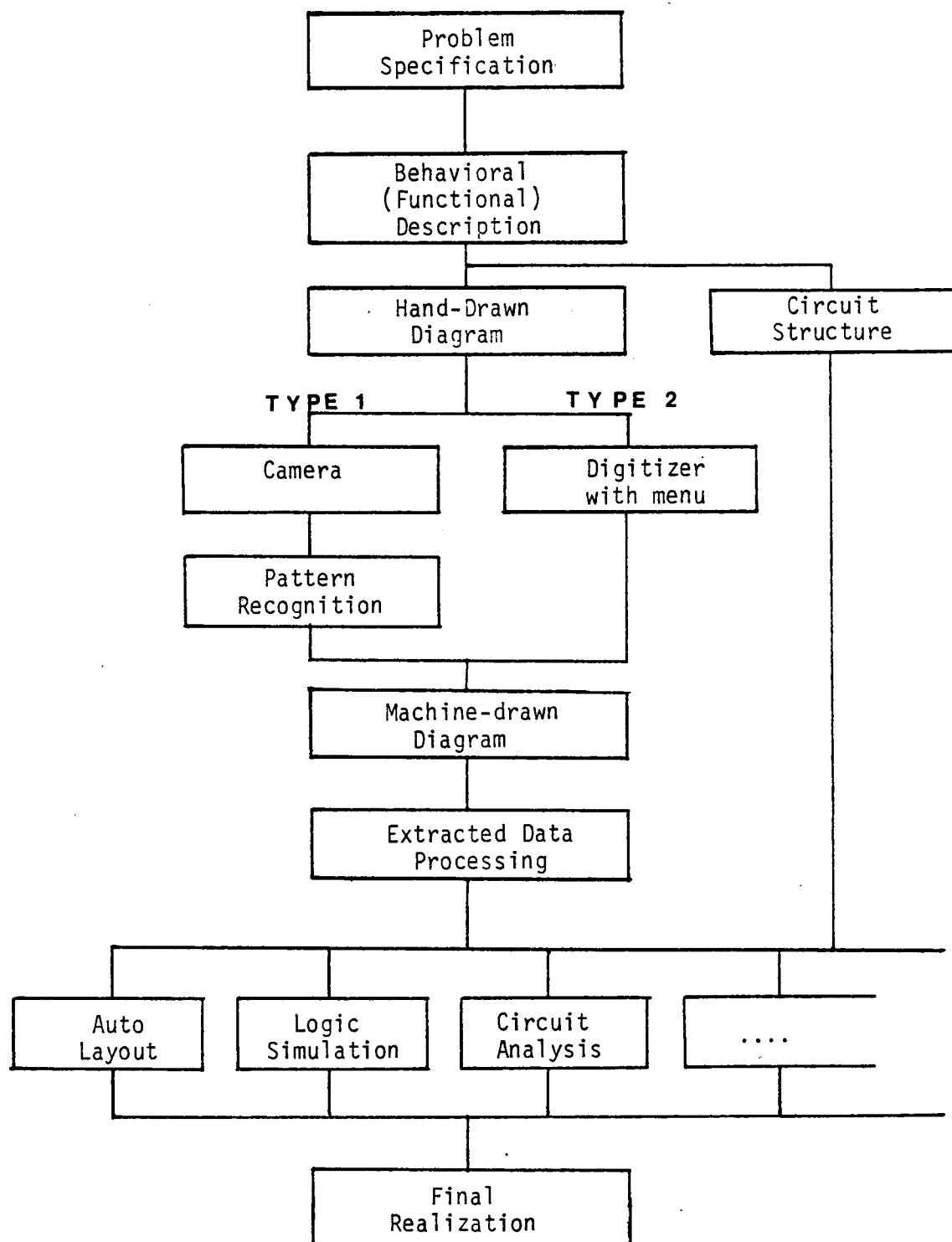


Figure 4.1. Design process of a circuit.

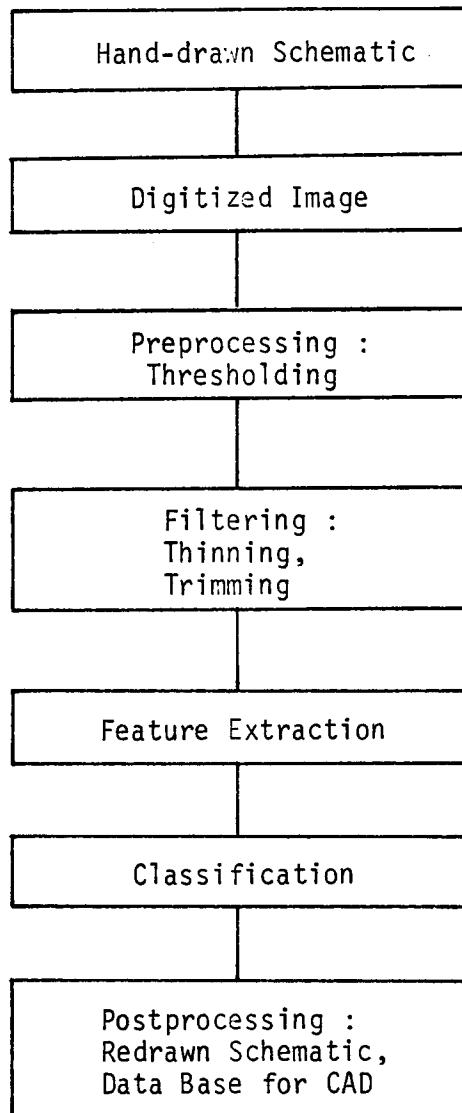


Figure 4.2. Recognition Procedures

recognition system presented here. A more detailed description is given in following sections.

In order to simplify the computer recognition steps, the following restrictions were made:

- (1) size of an input diagram is A1 paper size ($8\frac{1}{2} \times 11$ ");
- (2) logic symbols to be recognized are limited as shown in Figure 4.3;
- (3) lines and symbols should be clearly drawn in a black pen or pencil so that all the pixels on logic symbols can be connected;
- (4) the illumination on the input diagram is uniform enough to segment line figures from the background by one threshold intensity value.

4.2 Preprocessing

The preprocessing of a hand-drawn schematic diagram consists of two stages, quantization and thresholding. Quantization is the digitizing of an input image into a gray level pixel matrix or digital image. The input diagram shown in Figure 4.4 is digitized into eight bit per pixel so the range of the gray level is from 0 to 255 with a resolution of 512×512 pixels. Various algorithms have been proposed to extract line figures from multi-gray level pictures [15], [18-19]. If the lines in the input image are drawn with a reasonably uniform brightness against a background of varying brightness, a threshold gray level value T may be chosen easily and then used to segment the lines from the background. The decision rule

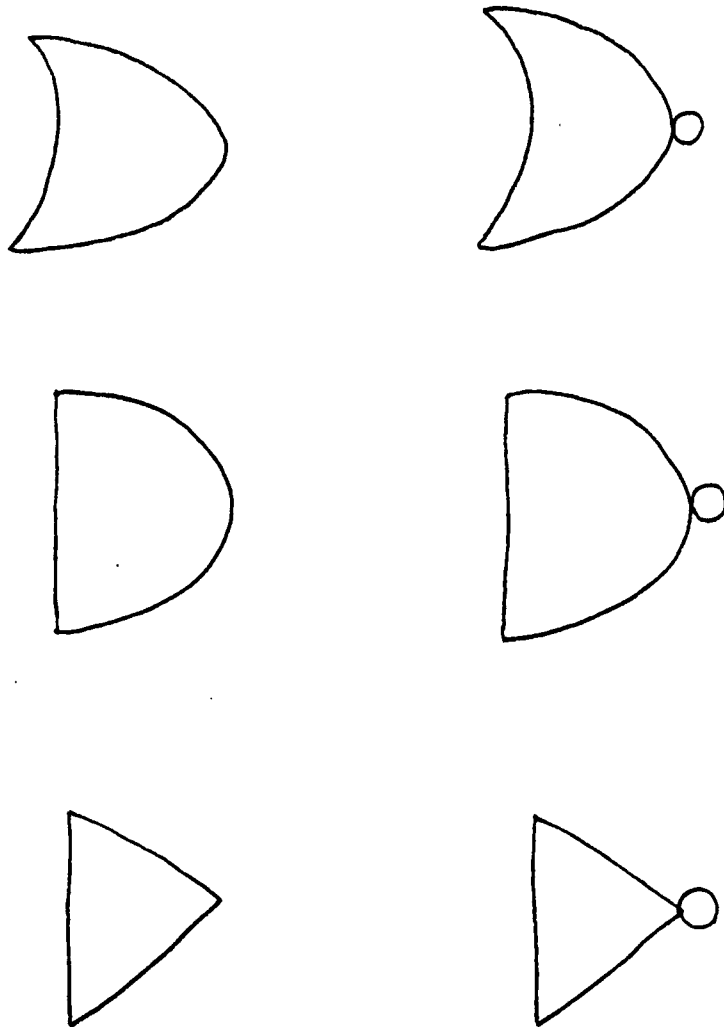


Figure 4.3. Logic symbols to be recognized.

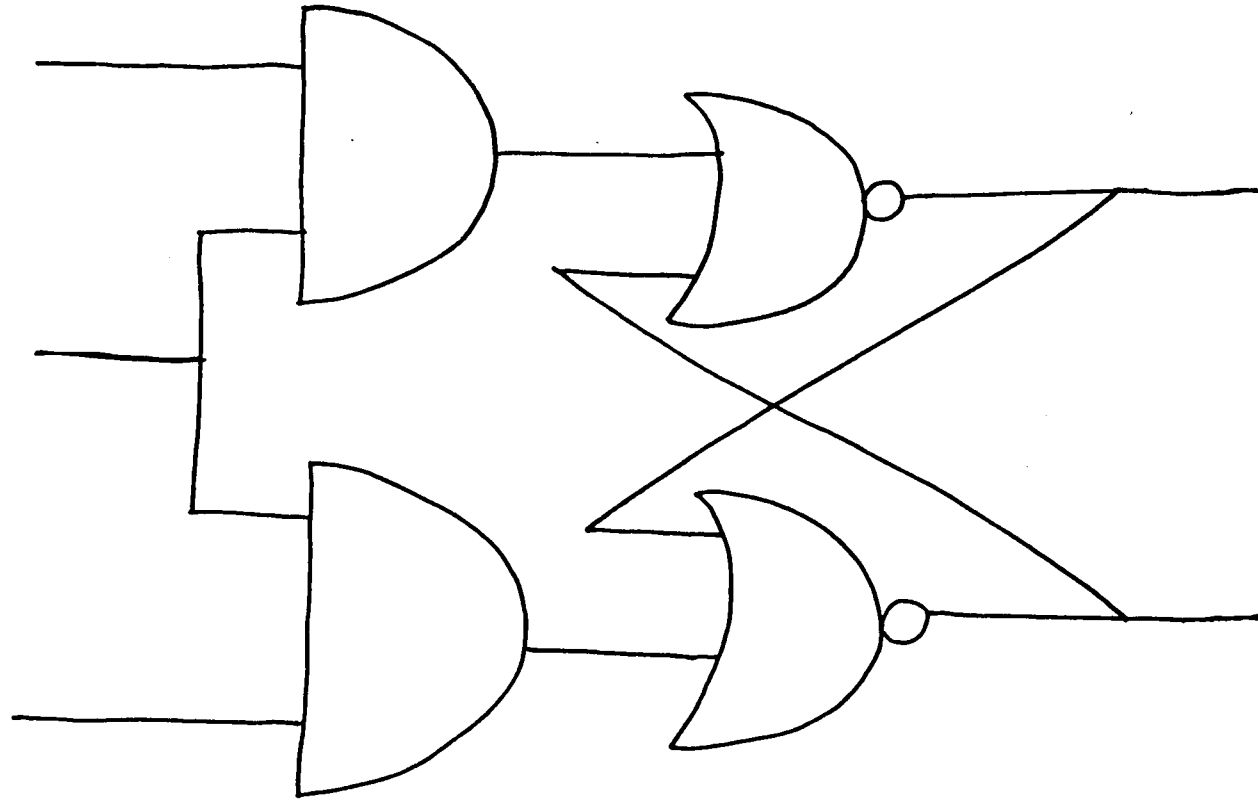


Figure 4.4 A hand-drawn schematic diagram.

$$g(x,y) = \begin{cases} 1 & , \text{ if } f(x,y) > T \\ 0 & , \text{ otherwise} \end{cases}$$

applied to each point of $f(x,y)$ would produce a simpler binary picture. Generally speaking, however, determining an optimum gray level threshold T to segment objects (lines in this case) from a background which has noise as a result of flaws in drawing or in the nonuniformity of illumination is one of the most difficult tasks in the field of image processing.

Many problems associated with thresholding have been studied [20], and some of them have solved by using a priori information and local varying thresholds [21-22]. However, heuristic techniques are often used in practice [23]. A gray level threshold T in Figure 4.4 can be chosen trial and error, assuming that the image $f(x,y)$ has reasonably uniform brightness in the background so that the pixels in the gate symbols are 8-connected after thresholding. Some connectivities, including 8-connectivity, are described in the following subsections. The output binary image $g(x,y)$ after thresholding is shown in Figure 4.5.

4.3 Filtering

The procedure of filtering the image consists of two steps: thinning and trimming. It is expected that the binary image $g(x,y)$, extracted after thresholding, has all the line figures connected, with the exception of minimal noise or short branches. Noise and short branches are removed by the thinning and trimming operations.

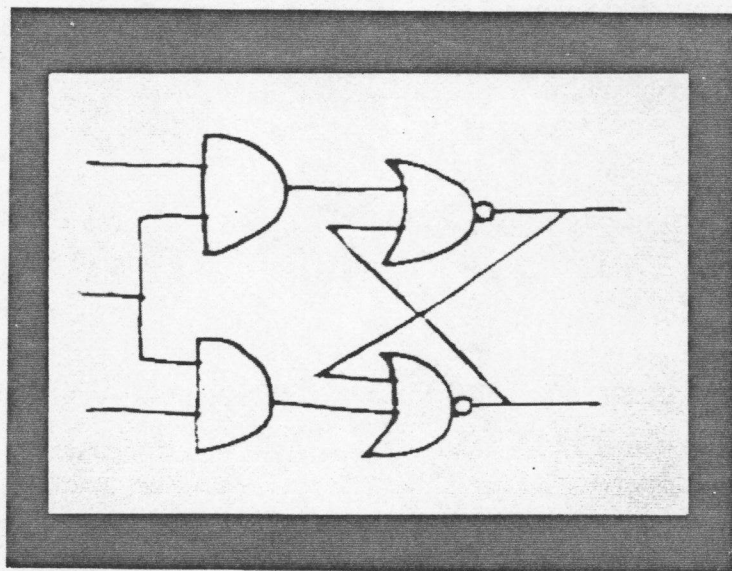


Figure 4.5. The output binary image after thresholding.

In order to introduce these operations, some local properties of the digital picture are discussed first.

4.3.1 Local Properties of Digital Picture

In the digitized picture, the plane is divided into squared grids (called pixels or rasters) which represent a level of intensity. Let us consider an arbitrary pixel P at coordinates (x,y) surrounded by eight neighbors (pixels) as shown in Figure 4.6. The four horizontal and vertical neighbors of the pixel $P(x,y)$ are called the 4-neighbors of P , denoted by $N_4(P)$. The distance between neighboring pixels will be taken as a unit length. The four diagonal neighbors of P , denoted by $N_d(P)$, together with the $N_4(P)$ defined above, are called the 8-neighbors of P and denoted as $N_8(P)$.

Let V be the set of values of the pixels which are allowed to be connected, and $\bar{V}$ be the complement set of V . For example, if connectivity of only those pixels with an intensity value of 20 is desired, then V is $\{20\}$. For a binary picture, if V is $\{1\}$, then $\bar{V}$ must be $\{0\}$. They may be denoted as $V=\{1\}$ and $\bar{V}=\{0\}$, or simply $V\{1\}$ and $\bar{V}\{0\}$, respectively. Two pixels P and Q with intensity values from V are connected if there exists a sequence of neighbors between P and Q such that the values of them are from V . We consider four types of connectivity:

(1) 4-connectivity. Two pixels P and Q with values from V are 4-connected if Q is in the set $N_4(P)$.

(2) 8-connectivity. Two pixels P and Q with values from V are

$P(x-1,y-1)$	$P(x,y-1)$	$P(x+1,y-1)$
$P(x-1,y)$	$P(x,y)$	$P(x+1,y)$
$P(x-1,y+1)$	$P(x,y+1)$	$P(x+1,y+1)$

$$N_4 = \{ P(x-1,y), P(x+1,y), P(x,y-1), P(x,y+1) \}$$

$$N_d = \{ P(x-1,y-1), P(x+1,y-1), P(x-1,y+1), P(x+1,y+1) \}$$

$$N_8 = N_4 \cup N_d$$

Figure 4.6. Neighborhood of a pixel $P(x,y)$.

8-connected if Q is in the set $N_8(P)$.

(3) m -connectivity (mixed connectivity). Two pixels P and Q with values from V are m -connected if (i) Q is in $N_8(P)$, and (ii) the value of Q is not equal to the value of any of the pixels in the intersection of the sets $N_4(P)$ and $N_8(P)$. (This is the set of pixels that are 4-neighbors of both P and Q and whose values are from V). If this set is empty, Condition (ii) is automatically satisfied [24].

(4) u -connectivity (unique connectivity). Two pixels P and Q with values from V are u -connected if (i) Q is in $N_8(P)$, and (ii) the values of the two pixels in the intersection of $N_8(P)$ and $N_4(Q)$ are from $\bar{V}$.

An example shown in Figure 4.7 illustrates all types of connectivity.

In the case of 8-connectivity, a pixel P with value from V is called an interior pixel if the value of any pixels in $N_4(P)$ are from V , and it is called an isolated pixel if all values of pixels in $N_8(P)$ are from $\bar{V}$.

The set of pixels from V is called a connected component when they are connected. A connected component from V surrounded by a connected component from $\bar{V}$ is called a hole. A connected component with no holes is said to be simply connected; otherwise it is said to be multiple connected.

The connectivity number (N_c) of a pixel P in a binary picture is defined as follows [25]:

	P_1	P_2
	P_0	
		P_3

(a) An example pixel arrangement (P_0 - P_3 are from V).

Connectivity Between P_0 &	4-	8-	m-	u-
P_1	yes	yes	yes	no
P_2	no	yes	no	no
P_3	no	yes	yes	yes

(b) Connectivities of the pixel arrangement above.

Figure 4.7. Several types of connectivity.

For the 4-connectivity case,

$$Nc_4(P_0) = \sum_{P_k \in N_4(P_0)} (P_k - P_k P_{k+1} P_{k+2}),$$

for the 8-connectivity case,

$$Nc_8(P_0) = \sum_{P_k \in N_8(P_0)} (P_k - P_k P_{k+1} P_{k+2})$$

Here $\bar{P}_k$ means $1 - P_k$ and the value of subscript k greater than eight is assumed to be $k - 8$, and the eight neighboring pixels of P_0 , $N_8(P_0)$, are labeled as follows

$$\begin{array}{ccc} P_4 & P_3 & P_2 \\ P_5 & P_0 & P_1 \\ P_6 & P_7 & P_8 \end{array} \quad \begin{array}{l} \text{with } P_i = 1 \text{ if black,} \\ 0 \text{ if white} \end{array}$$

$i = [0, \dots, 8].$

A pixel from a connected component are denoted by $P\{1\}$ or simply P , and a pixel from $\bar{V}$ denoted by $P_i\{0\}$ is the pixel in the background (white). If the values of the set V are not specified, we consider it to be $\{1\}$, denoted by $V\{1\}$. Therefore an arbitrary black pixel P from V may be denoted by P , $P\{1\}$, or $P_0\{1\}$. This notation is kept through the subsequent sections.

The connectivity number, denoted by $Nc_8(P)$ or simply $Nc(P)$, shows the relationship between a pixel P and its 8-neighbors. A pixel P is reasonably classified by the value of Nc of P as follows:

$N_c(P) = 0$, interior or isolated pixel

- 1, edge pixel
- 2, connecting pixel
- 3, branching pixel
- 4, crossing pixel.

A pixel P of edge pixels ($N_c(P)=1$) is defined to be an end pixel if $\sum N_8(P)=1$. The pixel P is a junction pixel if N_c is equal to 3 or 4; that is, if there are three or four pixels from V in $N_8(P)$. A pixel is deleteable if a change of value from $P\{1\}$ into $P\{0\}$ preserves the connectivity of the picture [27]. It is noted that $\sum N_8(P)$ does not correspond to $N_c(P)$. The connectivity number of a pixel P is coincident with $\sum N_8(P)$, if and only if P is u -connected with its $N_8(P)$.

In the next subsection the connectivity number is used to classified a pixel.

4.3.2 The Shrinking and Thinning Algorithms

In this section the shrinking algorithm and its properties are presented [25]. Then the shrinking algorithm is modified to the thinning algorithm for extracting u -connected line figures in a binary picture.

4.3.2.1 Shrinking Algorithm. A picture is raster-scanned and then pixel P is changed into $P\{0\}$ if $N_c(P)=1$. Then repeat this procedure until no deleteable pixel is found in the picture.

The resulting picture is called the shrunk skeleton. The

shrunk skeleton of any simply connected component is an isolated pixel, and the one of a multiple connected component is a shrunk multiple connected component with the same number of hole. The shrunk skeleton of the connected component having one hole contains only connecting pixels ($\sum N_8(P) = N_c(P) = 2$). These properties are guaranteed by the following theorems.

Theorem 1. Even if any P at which $N_c(P) = 1$ is removed (changed to $P\{0\}$), the connectivity of a picture is kept unchanged [25].

Theorem 2. Any simply connected component which consists of two or more P 's contains at least two deletable pixels [27].

Theorem 3. If a connected component has $N+1$ holes, then its shrunk skeleton contains at most $2N$ P 's having $N_c > 2$ [27].

These theorems were proved in the literature [25-28].

4.3.2.2 Thinning Algorithm. While scanning a picture delete the pixel P if $N_c(P) = 1$ and $\sum N_8(P) \neq 1$. Repeat this procedure until no deletable pixel is found in the picture.

The condition $\sum N_8(P) \neq 1$ in the thinning algorithm guarantees no deletion of line figures in the picture. Theorem 3 in the shrinking algorithm shows that a shrunk picture is one pixel wide and u-connected almost everywhere.

4.3.3 Filtering

Most thinning algorithms discussed in the literature [29-30] rely on what is called the steady erosion of objects. This means that thick connected components are thinned in a recursive fashion,

maintaining the connectivity of the shape. Since these thinning algorithms put special emphasis on connectivity, they produce topologically incorrect skeletons with extra tails (Figure 4.8a-b), and lead to a "necking" effect at cross-overs (Figure 4.8c-d) [29]. The thinning operation presented above is one of these operations.

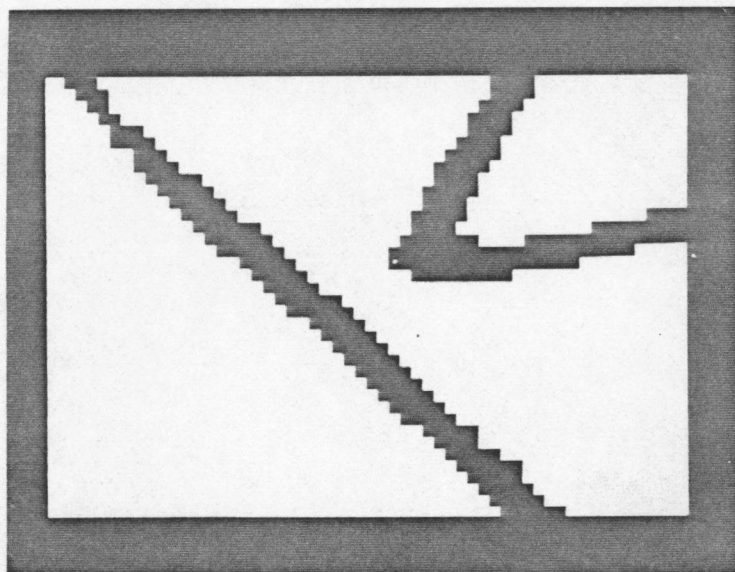
The second stage of filtering is the trimming operation. The trimming operation deletes branches whose length is less than a certain length, and it also deletes isolated pixels to make the picture u-connected. During the thinning and trimming operations, the coordinates of the end pixels and junction pixels are saved. Furthermore, if there exists more than one junction pixel within a given length of radius (20 units used), those pixels are considered as a possible logical negate symbol or a line cross-over (necking effect), and saved in separate data files.

The next step, described in the following section, is feature extraction--the most important step to recognize lines and gate symbols.

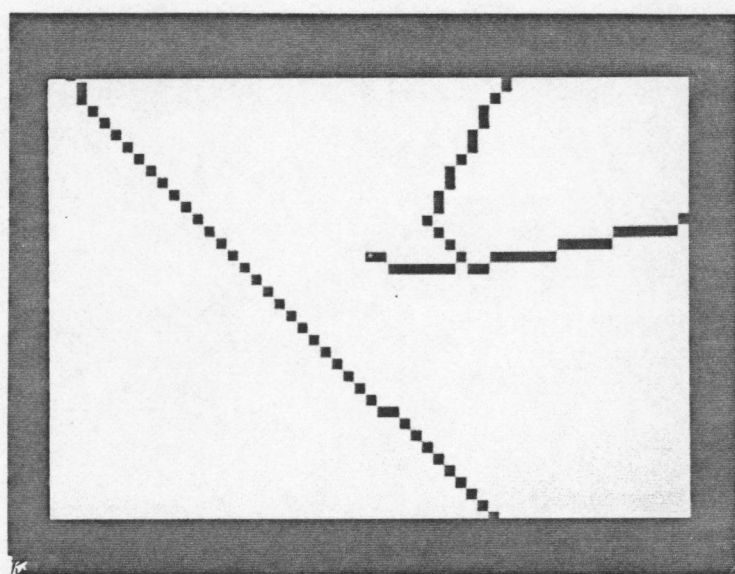
4.4 Feature Extraction and Classification

In order to recognize and classify an object or pattern, it is reasonable to consider the problems of what discriminatory features to select and how to extract these features. The lines, slope or shape of the lines, and the relation between them are selected as the discriminatory features for schematic diagrams.

In the next section the membership function is introduced

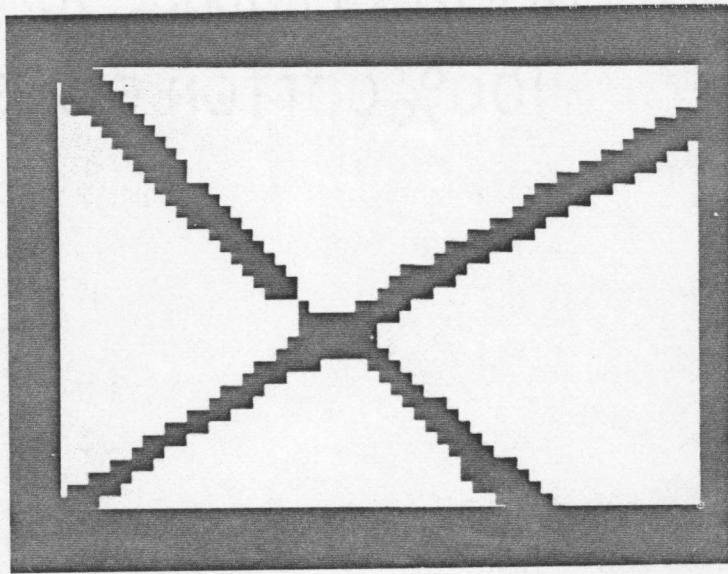


(a) Before thinning operation.

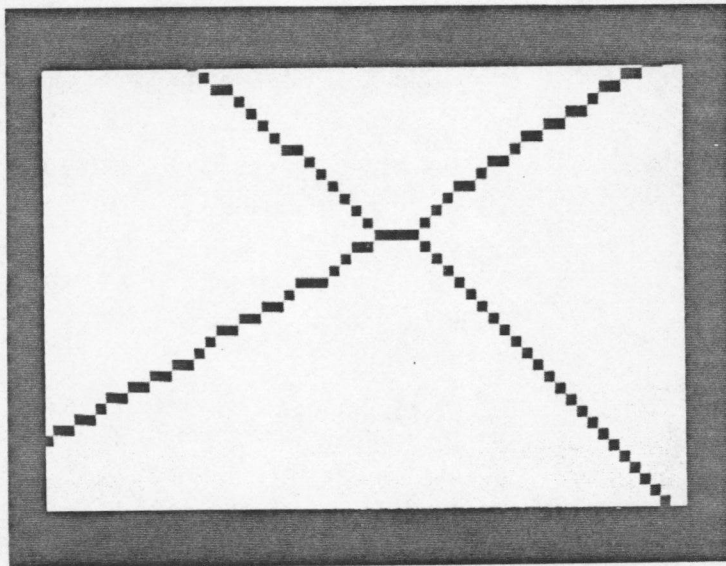


(b) After thinning operation (extra tails).

Figure 4.8. Some details of the binary image.



(c) Before thinning operation.



(d) After thinning operation (necking effect).

Figure 4.8. (Continued)

first, then applied to feature classification.

4.4.1 Membership Functions

Membership functions [33] are used here for interpretation and description of a schematic diagram. A line is defined as a simply and u-connected component. The line x connected from P_1 to P_n has each membership grade of "vertical," "horizontal," and "oblique," denoted by $M_v(x)$, $M_h(x)$, and $M_{ob}(x)$, respectively. The grades of these membership can be computed as follows;

$$M_v(x) = \frac{1}{N-k} \sum_{i=1}^{N-k} \mu_v(P_i)$$

$$M_h(x) = \frac{1}{N-k} \sum_{i=1}^{N-k} \mu_h(P_i)$$

$$M_{ob}(x) = \frac{1}{N-k} \sum_{i=1}^{N-k} \mu_{ob}(P_i)$$

where

$$\mu_v(P_i) = \begin{cases} 1 - \left| \frac{1}{m_i} \right|^{Fe} & , \quad |m_i| > 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_h(P_i) = \begin{cases} 1 - |m_i|^{Fe} & , \quad |m_i| < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_{ob}(P_i) = \begin{cases} 1 - \left| \frac{-45}{45} \right|^{Fe} & , \quad 0 < |m_i| < \infty \\ 0 & \text{otherwise.} \end{cases}$$

Here $m_i (= \tan \theta)$ is the gradient of a line x connecting between P_i and

P_{i+k} , and P_{i+k} indicates k th pixel apart from P_i as shown in Figure 4.9a. $\mu_v(P_i)$, $\mu_h(P_i)$, and $\mu_{ob}(P_i)$ represent the membership function for vertical, horizontal, and oblique, respectively. The membership function having positive value in the interval $[0,1]$ denotes the degree to which a line x may be a member of or belong to a group such that

$$\mu_v(P_i) \rightarrow 1 \quad \text{as } |\theta| \rightarrow 90^\circ$$

$$\mu_h(P_i) \rightarrow 1 \quad \text{as } |\theta| \rightarrow 0^\circ$$

$$\mu_{ob}(P_i) \rightarrow 1 \quad \text{as } |\theta| \rightarrow 45^\circ$$

and

$$\mu_v(P_i) \geq \mu_h(P_i) \quad \text{as } |\theta| \geq 45^\circ$$

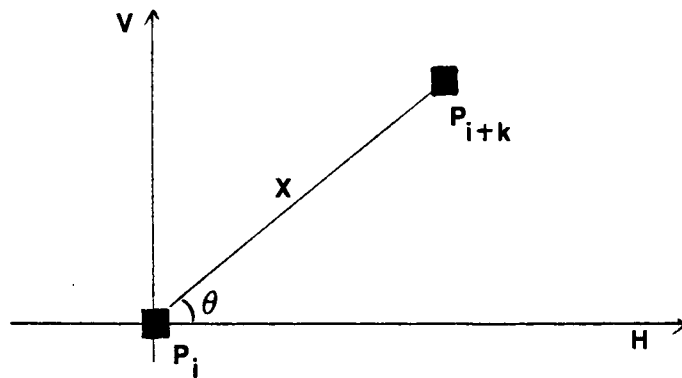
The positive constant F_e (0.5 used) controls the shape of the membership function (Figure 4.9b).

Similarly the grade of membership of a line segment x to be an "arc" may be expressed by the function

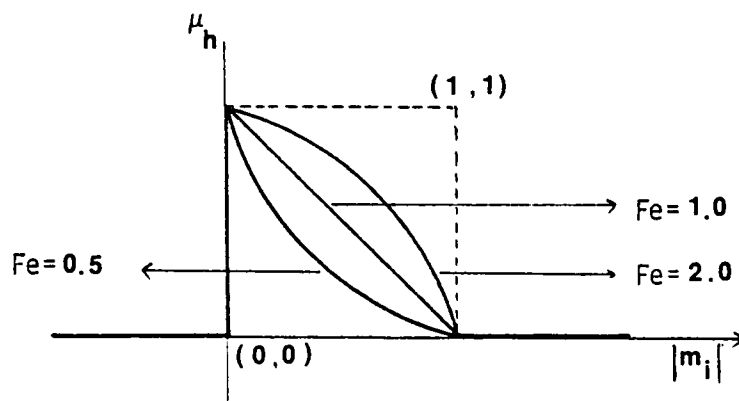
$$\mu_{arc}(x) = \left(1 - \frac{a}{l} \right)^{F_e}$$

where a is the length of the line joining the two extreme points of an arc x as shown in Figure 4.9c and l is the arc length such that the lower the ratio (a/l) , the higher the degree of "arcness." In essence it provides a quantitative measure of curvature of an arc. As an example, the membership grades of a line segment are calculated as illustrated in Figure 4.10.

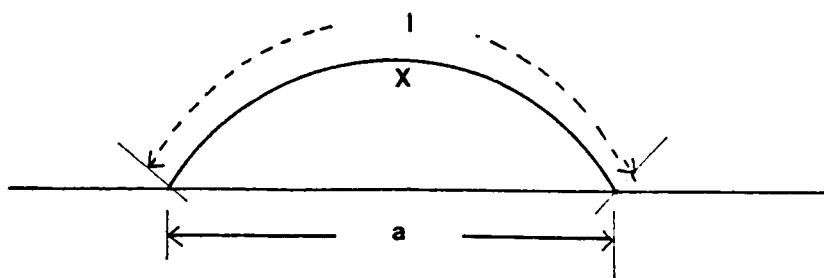
In the next subsection membership functions previously



(a) Line.

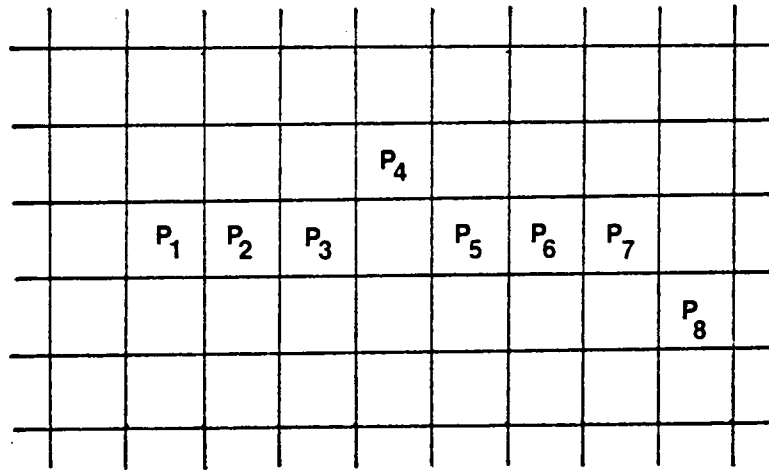


(b) Graph of membership function $\mu_h(P)$



(c) Arc

Figure 4.9. Membership functions.



(a) A line ($P_1 - P_8$ are from V).

	M_v	M_h	M_{ob}	μ_{arc}
k=1	0.0	0.5	0.429	0.142
k=2	0.0	0.646	0.211	0.142

(b) Membership grades of the line above with two different k's. ($N=8$, $Fe=0.5$)

Figure 4.10. An example of membership grades.

described have been implemented and applied to the extraction of primitives in a schematic diagram in order to interpret that image.

4.4.2 Separation of Lines and Symbols

To extract information from the binary image and identify the gate type, gate symbols must first be isolated from the connected lines.

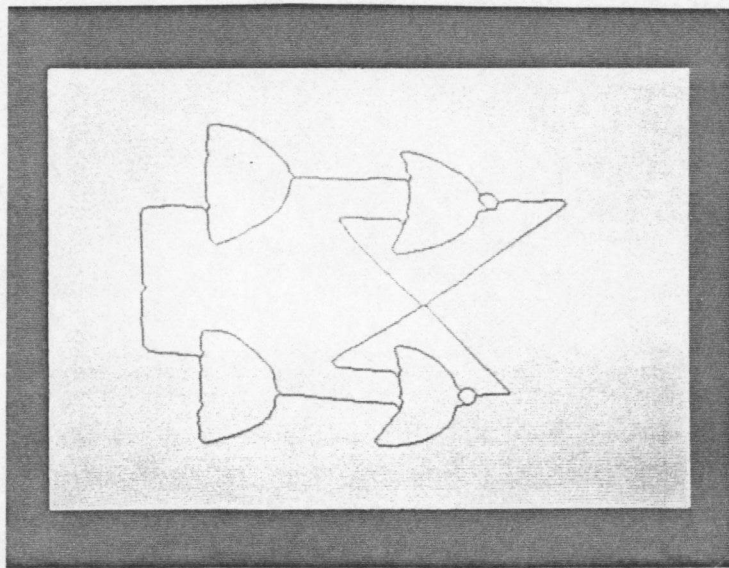
After obtaining the u-connected binary image, coordinates of the end pixels ($N_c = 1$) and the junction pixels ($N_c > 2$), subsequent steps for separating the straight lines from the logic symbols are as follows;

- (1) Follow the border at an end pixel $P(\sum N_s(P)=1 \text{ or } N_c(P)=1)$.

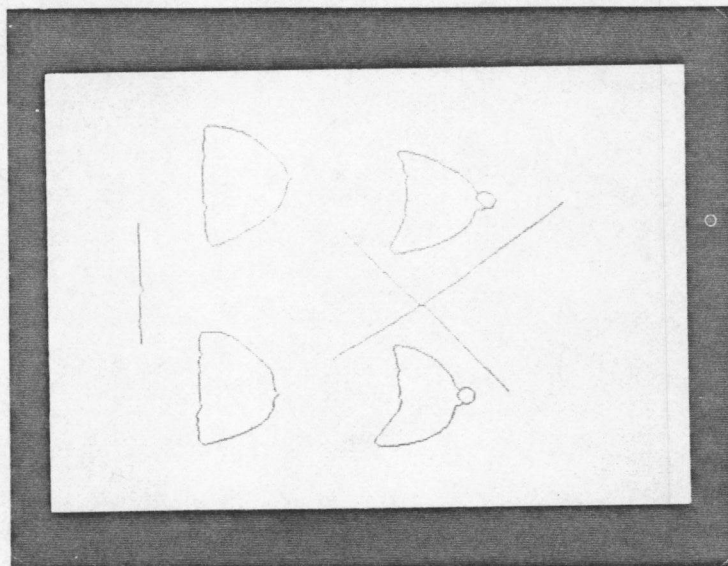
- (2) Then delete that border if the border-following terminates at a junction pixel $Q (N_c(Q) > 2)$. Store the coordinates of the two pixel P, Q into proper data files after finding the shape of the line. This shape can be one of "L", "_", "I", "I". These lines could be classified by the membership functions. However, they are done by inspecting the relation between two pixels P and Q due to their simplicity.

Note that this step which removes input/output lines connected to a gate can be done during the trimming operation simultaneously (Figure 4.11a).

- (3) Follow the border from a junction pixel, then delete the line if it is horizontally connected to another junction pixel. Additionally, when following a segment of a border considered as



(a)



(b)

Figure 4.11. Separation of gates from lines.

horizontal, if the acceptable range of the membership grade (0.8 used) is exceeded, stop deleting pixels and save that point the data base of the junction pixel. This step remove horizontal line segments of a line connecting between two gates.

(4) Repeat the same steps above for every end and junction pixel.

From the steps previously described, data such as all input/output lines of gates have been saved, and deleted since they were connected to gates. Therefore gate symbols are separated from the lines (Figure 4.11b).

4.4.3 Classification of Symbols

Since all symbols of gates are separated from the attached lines, the coordinates of a symbol can be obtained in the following manner;

(1) Follow border from a junction pixel P at which $N_c(P)=2$ since the attached line was deleted.

(2.a) Terminate the border-following at $N_c(P)=1$ or if it returned to P .

(2.b) Terminate the border-following if a junction pixel ($N_c(P)>2$) is encountered and then examine each direction of a certain number of pixels on the previous path and the next path. Then choose the closest one to the previous direction.

(3) If the border-following returns to P , save the coordinates of the pixels with a gate symbol found. If the border-following

terminates at $N_c(P)=1$, save the two coordinates of P and the last pixel as a line segment.

(4) Repeat these steps until the entire data set of the junction pixels is used.

Once all the coordinates of each gate symbol in the picture are obtained, the following parameters for each symbol are to be estimated (Figure 4.12):

(1) Three corner points: $P(x,y)$, $Q(x,y)$, $R(x,y)$

(2) Center of gravity: $C(\text{center } x, \text{center } y)$

$$\text{center } x = (\text{max } x + \text{min } x) / 2$$

$$\text{center } y = (\text{max } y + \text{min } y) / 2$$

(3) Size of symbol: with, height

$$\text{width} = \text{max } x - \text{min } x$$

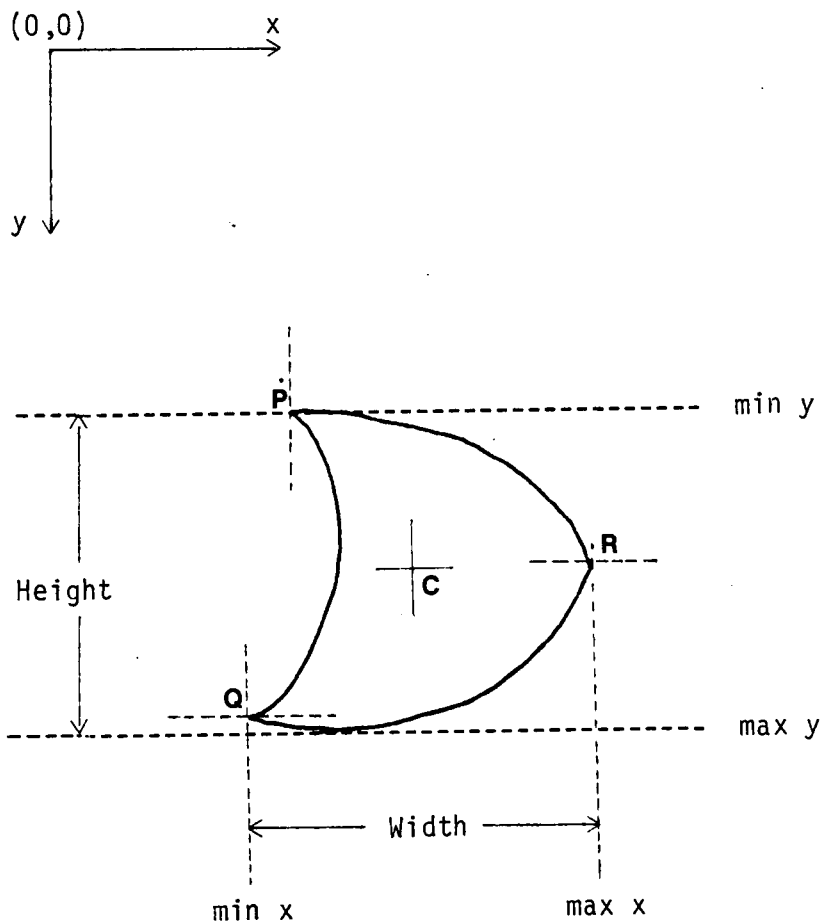
$$\text{height} = \text{max } y - \text{min } y$$

(4) The arcness of PQ, and the average arcness of PR, QR,

where $\text{max } x$, $\text{max } y$ are the maximum value of x , y and $\text{min } x$, $\text{min } y$ are the minimum value of x , y . If some clustered pixels ($N_c(P)>2$) exist near the pixel R, those pixels must be on the negative symbol of a gate. The membership of grade can be calculated with the membership functions defined in Section 4.3.1. The method, called the Euclidean distance, is used to measure distance [24]. The distance between two pixels $P(x_1, y_1)$, $Q(x_2, y_2)$, denoted by $De(P, Q)$, is defined as

$$De(P, Q) = ((x_1 - x_2)^2 + (y_1 - y_2)^2)^{0.5}.$$

The symbols are classified with the arcness of PQ, PR and QR.

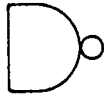


- P : Left-top-corner point.
- Q : Left-down-corner point.
- R : Right-most-corner point.
- C : Gravity center point.

Figure 4.12. Parameters in a symbol.

If the arcness of the border connecting two points is greater than the predetermined threshold which can be obtained with a set of training samples, the border is classified as an arc. Depending on the three sets of arcness of the corner points on a logic symbol, the type of the symbol is determined as illustrated in Figure 4.13.

Once the parameters described above are obtained, the drawing of the schematic diagram is no longer a problem. The final output of the schematic diagram is shown in Figure 4.14.

Logic Negate Symbol	$\frac{\mu_{\text{arc}}(\overline{PR}) + \mu_{\text{arc}}(\overline{QR})}{2}$	$\mu_{\text{arc}}(\overline{PQ})$	Classified Symbol
x	x	x	
x	x		
x		x	Ambiguous
x			
	x	x	
	x		
		x	Ambiguous
			

- (1) $Fe = 0.5$
- (2) Threshold of arcness = 0.4
- (3) "X" indicates existence of an arc or a logic negate symbol.

Figure 4.13. Classification of symbols.

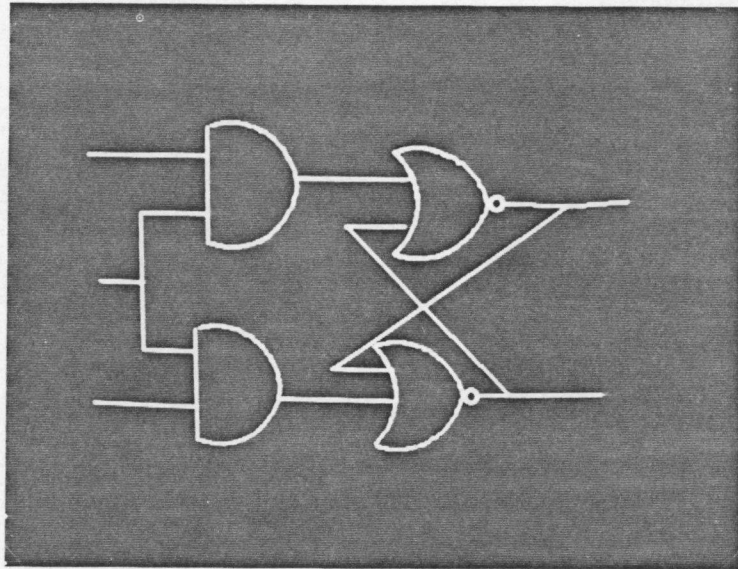


Figure 4.14. The final output of the schematic diagram.

5. CONCLUDING REMARKS

The program IC-MAP was implemented on a TI-990 computer system. IMAP accepts the input data from the hardware (wiring) diagram or the completely labeled schematic diagram. Since the VDT (Model 911) used as a display unit is not a graphic display terminal, the visual area on the screen (24x80) is at most one chip at a time. This restriction causes the most significant limitation of the visual aid in developing and using IMAP. Furthermore the awkward shape of the IC template generated by IMAP is unavoidable. It is very tedious to prepare the wiring diagram and type in all the information in the diagram. But once input data is entered, the editing is fairly easy and convenient. The documents generated by CMAP may compensate for the laborous tasks in IMAP. Although there are many limitations in IC-MAP, the groundwork of a CAD system has been established. If the following features are added to IC-MAP on the TI-990 computer system, a powerful aid will be available to designers:

- (1) Availability of graphic display terminal and plotter.
- (2) More convenient input device such as a cursor or joystick.
- (3) Direct input from the schematic and automatic chip (device) assignment.
- (4) Acceptability of general components (e.g.: resistors, transistors, capacitors).
- (5) Timing analysis of the signals.

Since the IC-MAP is programmed in Pascal, the program is more

portable and readable. Therefore, the program can be easily be run on an other computer with minor modifications.

In addition to IC-MAP, for more efficient VLSI circuit design using CAD system, an automatic computer recognition system has been studied for hand-drawn schematic diagram. Based on u-connectivity, thinning and trimming operations have been developed. The automatic classifier of some symbols in a schematic is implemented with membership functions.

In order to put this system into practical use, many problems still need to be solved:

- (1) Recognition of hand-drawn characters.
- (2) Minimization of quantization error and nonlinearity in illumination.
- (3) Selection of optimum threshold value.
- (4) Development of an efficient parallel thinning algorithm and its hardware implementation.
- (5) Development of algorithm extracting and linking primitives in noisy picture.
- (6) More flexible writing and drawing rules, size of input schematic diagrams.

As far as the character recognition problem is concerned, restrictions or writing rules, such as (i) use of block letters only and/or (ii) specified size of letters, will be given for the present time. Also, if textual information is to be processed, enough resolution for the classification of letters is needed. The

quantization error and irregularity of illumination which have significant effects can be minimized with a better input device and/or software development. Since the quantization and illumination are not expected to be perfect, a method to extract the binary coded line figures from the multi-gray level picture must be studied. Extracting the binary picture can be done by threshoding or by using a wave propagation method [18,19]. To make the fast recognizer, a parallel thinning algorithm and its hardware implementation are a must [31,33]. Since the binary coded line figures can have noise and can be disconnected, an algorithm should be developed to eliminate noise and to link the disconnected lines [34]. The recognition system must have as few limitations on writing and drawing as possible. The size of the input diagram recognizable by the system must be large enough to use in circuit design. However, since the size of the input picture is usually limited to the memory capacity used in the system, this problem can be solved by the "divide and synthesize method" [16].

BIBLIOGRAPHY

BIBLIOGRAPHY

- [1] C. Mead, L. Conway, Introduction to VLSI System, Addison-Wesley, 1980.
- [2] L. A. Leventhal, Introduction to Microprocessors: Software, Hardware, Programming, McGraw-Hill, 1978.
- [3] J. B. Peatman, Digital Hardware Design, McGraw-Hill, 1980.
- [4] The TTL Data Book for Design Engineers, Second Ed., Texas Instruments Inc., 1982.
- [5] Model 990 Computer DX10 Operating System Systems Programming Guide, vol. 5, Texas Instruments Inc., 1981.
- [6] Model 990 Computer DX10 TI Pascal Programmer's Guide, Texas Instruments Inc., 1981.
- [7] E. Horowitz, S. Sahni, Fundamentals of Data Structures, Computer Science Press, Inc. pp.335-378, 1976.
- [8] D. Cooper, M. Clancy, Oh! Pascal An Introduction to Programming, Norton, 1982.
- [9] A. V. Aho, J. E. Hopcroft, J. D. Ullman, Data Structures and Algorithms, Addison-Wesley, pp.253-290, 1983.
- [10] H. Yajima, H. Asuura, N. Takagi, "The Parallel Enumeration Sorting Scheme for VLSI," IEEE Trans. on Computers, vol. C-31, no.12, Dec. 1982.
- [11] A. Perez, "Pipe-lined Sinking Sort Algorithm for VLSI", The University of Tennessee, To be published, Oct. 1983.
- [12] R. C. Gonzalez, J. T. Tou, Pattern Recognition Principles, Coden APMCC, 1974.
- [13] C. Nissen, "Hierarchical Design Methodologies and Tools for VLSI Chips," Proceedings of IEEE., vol 71, no.1, pp.66-75, Jan. 1983.
- [14] J. A Darring, W. H. Joyner, L. Berman, and L. Trevillyn, "Experiments in Logic Synthesis," IEEE Int. Conf. on Circuits and Computers, ICCS, 1980.

- [15] J. F. Jarvis, "The Line Drawing Editor: Schematic Diagram Editing Using Pattern Recognition Technique," Computer Graphics and Image Processing, vol.6, no.5, pp.452-482, 1977.
- [16] S. Kakumoto, Y. Fujimoto, and J. Kawasaki, "Logic Diagram Recognition by Divide and Synthesis Method," Artificial Intelligence and Pattern Recognition in Computer Aided Design, J. C. Latombe, Ed., North-Holland Publishing Co., pp.457-476, 1978.
- [17] W. C. Lin, "Machine Recognition and Plotting of Hand-Sketched Line Figures," IEEE Trans. on Systems, man, and Cybernetics, vol.SMC-8, no.1, 1978.
- [18] K. Shikano, J. Toriwaki, T. Fukumora, "Wave propagation method - A method for conversion of gray pictures into line figures," Systems.Computers.Controls, vol.3, no.5, pp.668-675, 1972.
- [19] K. Shikano, J. Toriwaki, T. Fukumora, "Modified Algorithms of Wave Propagation Method for Processing of Multileveled Pictures," System.Computer.Controls, vol.4, no.5, pp.24-50, 1973.
- [20] J. S. Weszka, "A Survey of Threshold Selection Technique," Computer Graphics and Image Processing, vol.7, pp.259-265, 1978.
- [21] S. Peleg, "A new probabilistic relaxation scheme," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. PAMI-2, pp.362-369, July 1980.
- [22] B. Bir, D. F. Oliver, "Segmentation of Images Having Unimodal Distributions," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.PAMI-4, no.4, July 1982.
- [23] K. Robert, W. C. Lin, "An On-Line Minicomputer-Based System for Reading Printed Text Aloud," IEEE Trans. on Systems, Man, and Cybernetics, vol.SMC-8, no.1, Jan. 1978.
- [24] R. C. Gonzalez, Class Notes on Digital Image Processing, The University of Tennessee, April 1983.
- [25] S. Yokoi, J. Toriwaki, T. Fukumura, "Topological properties in digitized binary pictures," System.Computers.Controls, vol.4, pp.32-39, 1973.

- [26] S. M. Selkow, "One-pass complexity of digital picture properties," Journal of the ACM, vol.19, no.2, pp.283, Jan..1970.
- [27] A. Rosenfeld, "Connectivity in the digital pictures," Journal of the ACM, vol.17, no.1, pp.146-160, Jan. 1970.
- [28] J. C. Alexander, A. I. Thaler, "The boundary count of digital pictures," Journal of the ACM, vol.18, no.1, p.105, 1971.
- [29] F. W. M. Stentiford and R. G. Monitor, "Some New Heuristics for Thinning Binary Hand-printed Characters for OCR." IEEE Trans. on Systems, Man, and Cybernetics, vol.SMC-13, no.1, pp.81-84, Jan/Feb. 1983.
- [30] E. R. Davies, A. P. N. Plummer, "Thinning algorithms and their role in image processing," The British Pattern Recognition Association Conference on Pattern Recognition, pp. 9-11, Jan. 1980.
- [31] R. Stefanelli, A. Rosenfeld, "Some parallel thinning algorithms for digital pictures," Journal of the ACM, vol.18, no.2, pp.255-264, April 1971.
- [32] K. P. Sankar, A. K. Robert, A. H. Abdullah, "Image Description and primitive Extraction using Fussy set," IEEE Trans. on Systems, Man, and Cybernetics, vol.SMC-13, no.1, pp.94-100 Jan/Feb 1983.
- [33] E. E. Triendl, "Skeletonization of noisy hand-drawn symbols using parallel operation," Pattern Recognition, vol.2, no. pp.215-226, 1970. .
- [34] T. S. Huang, R. Y. Tsai, J. K. Cheng, "Toward Automatic Undersea Search Using Pattern Recognition Techniques," Pattern Recognition, vol.16, no.3, pp.307-318, 1983.

APPENDICES

APPENDIX A

USER'S GUIDE FOR KPATH

USER'S GUIDE FOR KPATH

1. GENERAL

A program called KPATH is written to draw logic circuit diagrams on a terminal or a plotter. This interactive program can be run on any Tektronix graphic display terminal connected to a host computer which has an Interactive Graphics Library (IGL version 2.0) package. In order to operate KPATH on the other systems, the program has to be slightly modified. Experimental outputs printed by a Tektronix 4662 Interactive Digital Plotter are not satisfactory, but they will be drawings of draftsman-like quality if plotted on a better plotter such as CalComp Electromechanical Plotter with various color pens.

Fortran program KPATH is aimed at drawing a schematic circuit diagram and considering a possibility of the development of the low-cost and user-oriented CAD system. Figure A.1 shows the interconnections between devices needed to operate KPATH program. First user selects one of the graphics terminals available. The user can use a thumbwheel cursor, or cursor on keyboard, or digitizer (mouse) as an input device to draw the symbols displayed on the terminal. Once drawn on the terminal, the schematic diagrams stored in files can be redrawn by an attached plotter, or the data files can be transferred to the other better quality plotter to get the finished schematics. The next sections will show how to use KPATH, and also the global structure of it.

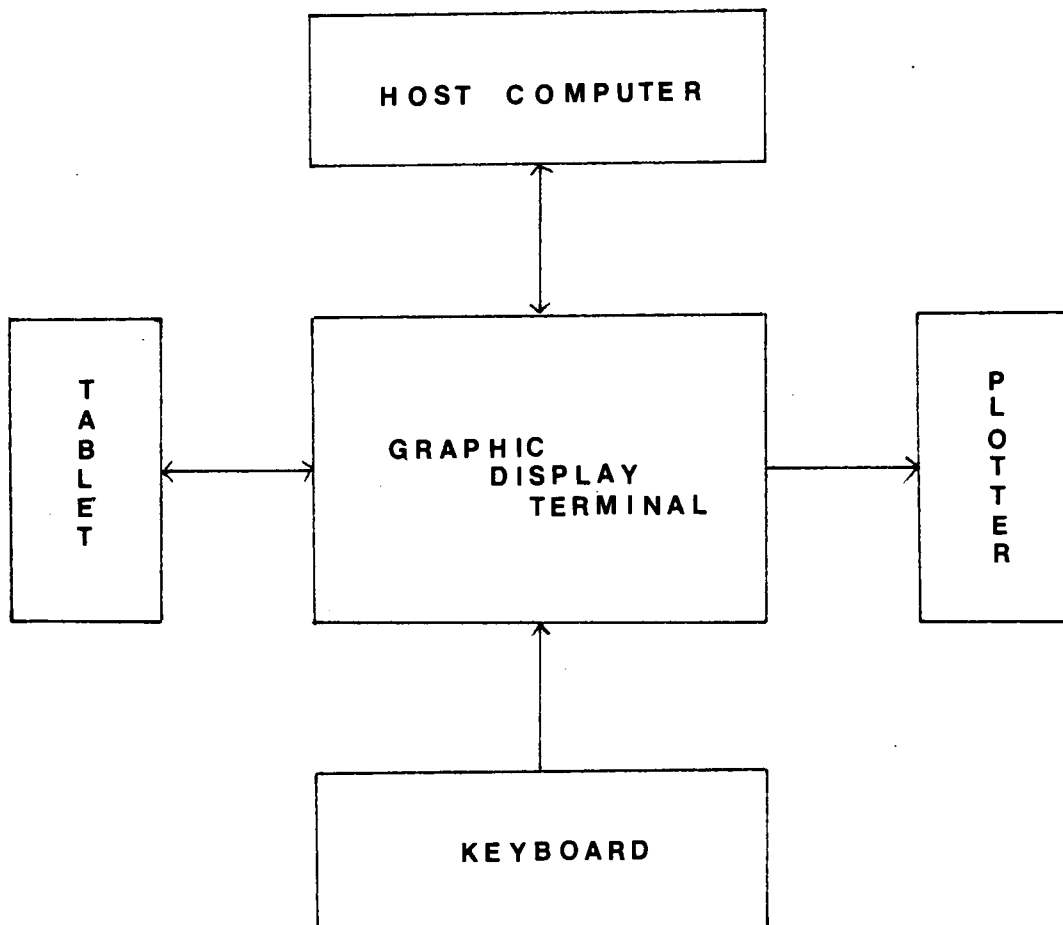


Figure A.1. Interconnection between devices.

2. STRUCTURE OF KPATH

2.1 Display and Input Devices

To run the program device-independent, first the user selects one of terminals (Tektronix 4010, 4014, 4025, 4027) available. Some parameters are set in different ways depending on the terminal used. Since Tektronix 4010's and 4014's are usually attached with a digitizing tablet, a chance to choose either mouse or thumbwheel cursor as input device will be given to the user.

2.2 Primary Selection

Once the input and the display unit are fixed, every drawing step is advanced interactively. The functions of the primary selection shown in Figure A.2 are as follows:

(1) CREATE : Get file name to create a schematic diagram.

(2) COPY : Get a file name to print out. After outputting the ordered diagram, KPATH returns to the primary selection.

(3) EDIT : Get a file name to edit. After displaying the specified diagram, KPATH shows all the commands and symbols in the secondary selection so that the user can add or delete symbols.

(4) EXIT : Go back to the operating system.

(5) LAYER 1 : Set the drawing pen to red or the parameter Layer = 1.

(6) LAYER 2 : Set the drawing pen to green or the parameter Layer = 2.

(7) LAYER 3 : Set the drawing pen to blue or the parameter Layer = 3.

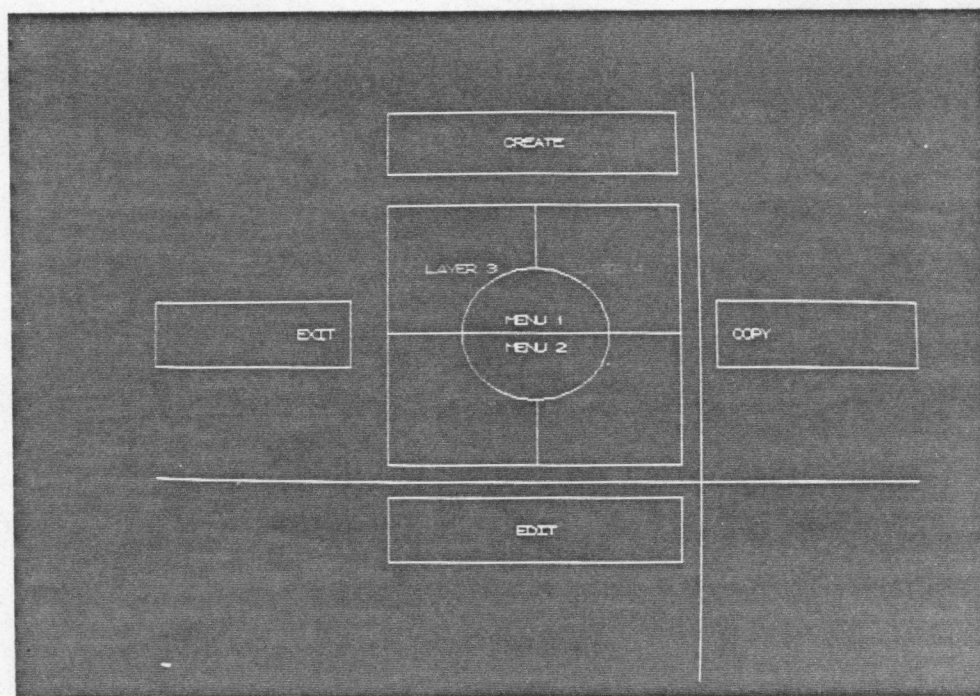


Figure A.2. Primary selection.

(8) LAYER 4 : Set the drawing pen to yellow or the parameter Layer = 4.

(9) MENU 1 : Choose Menu set 1.

(10) MENU 2 : Choose Menu set 2.

2.3 Secondary Selection

In the secondary selection shown in Figure A.3 and Figure A.4 any templates, curves, lines, and texts needed in the schematic diagram can be drawn and deleted in an interactive way. The commands available are as follows:

(1) QUIT : Quit creating a file or editing the file , and return to the primary selection.

(2) SAVE : Save all the drawings in disk files, and return to the primary selection.

(3) PATH : Connect all the points pointed by input device,

(4) LINE : Connect pairs of points.

(5) CIRCLE : Draw a circle with the first point as center and the second point as radius.

(6) TEXT : Accept a string from the keyboard and print it with upper-case letter, centered or left-adjusted mode, on the terminal.

(7) ERASE : Erase any texts or drawings which the cursor has located.

(8) L.TEXT : Same as TEXT ,but print input characters with lower-case letter except the first one.

(9) ARC3PT : Connect 3 points as an arc.

(10) DASH LINE : Connect pairs of points in dotted pattern.

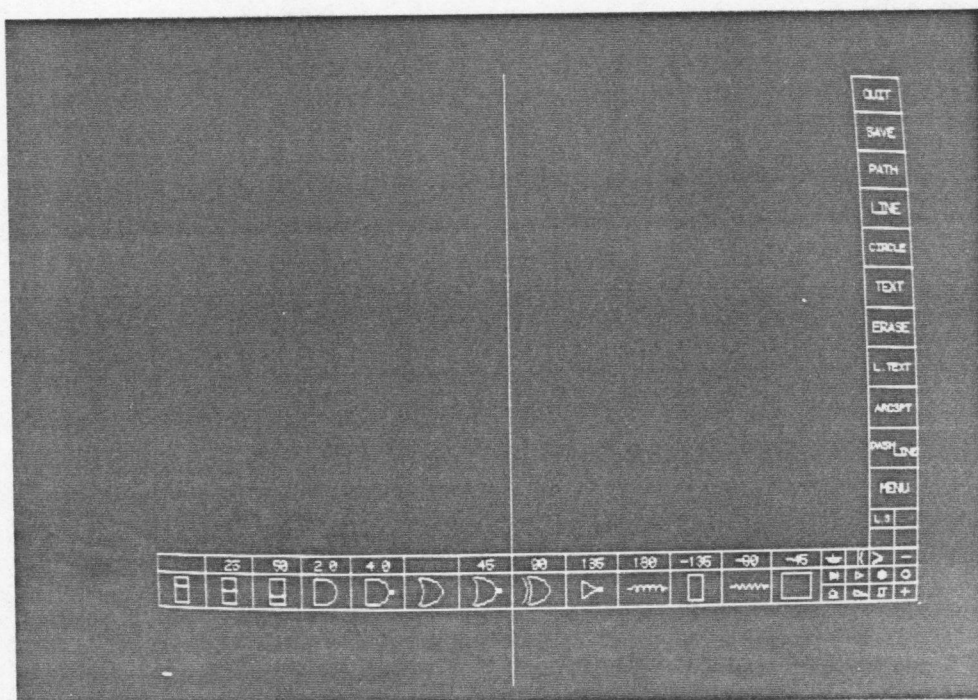


Figure A.3. Secondary selection with MENU 1.

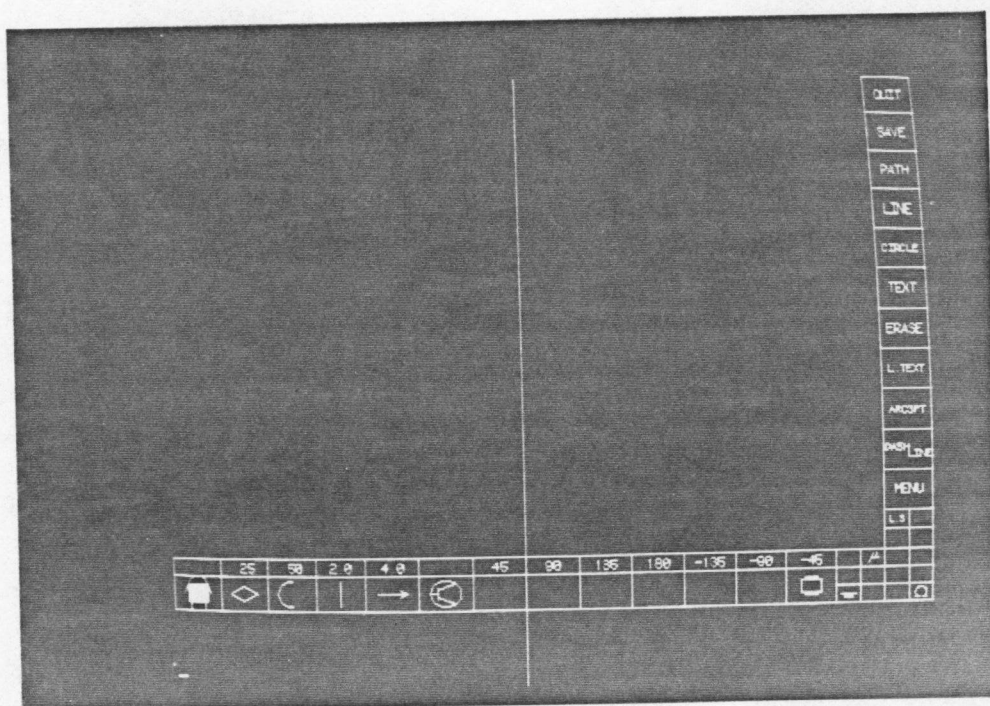


Figure A.4. Secondary selection with MENU 2.

(11) MENU : Erase the current menu set and display the other one.

(12) L.1, L.2, L.3, L.4 : Set the parameter LAYER to 1 to 4 or change the current color pen into the corresponding color pen pointed by cursor.

(13) SCALE 1 (0.25, 0.50, 2.0, 4.0) : Set the scale factor as pointed, and write the scale factor with the corresponding layer color to show the effective scale factor.

(14) ROTATE 0 (45, 90, 135, 180, -135, -90, -45) : Same as SCALE but it shows rotation factor.

(15) Symbols (Templates) : Change the color of symbol with the corresponding effective color to notify the user the symbol being drawn as pointed. Blank spaces in MENU 2 are left for the later use.

Besides the symbols and the commands mentioned above, there are some hidden commands in SCALE 1, TEXT , and L.TEXT. If they are selected two times in sequence; The actual scale factor is 3 times the current displayed factor or return to the original scaling. Therefore the scale factors available actually are 1.0, 0.25, 0.50, 2.0, 4.0, 3.0, 0.75, 1.5, 6.0, 12.0. In case of TEXT and L.TEXT, the output string can be located centered or left-adjusted.

Some experimental outputs shown in Figure 5 were drawn on a Tektronix 4027 terminal.

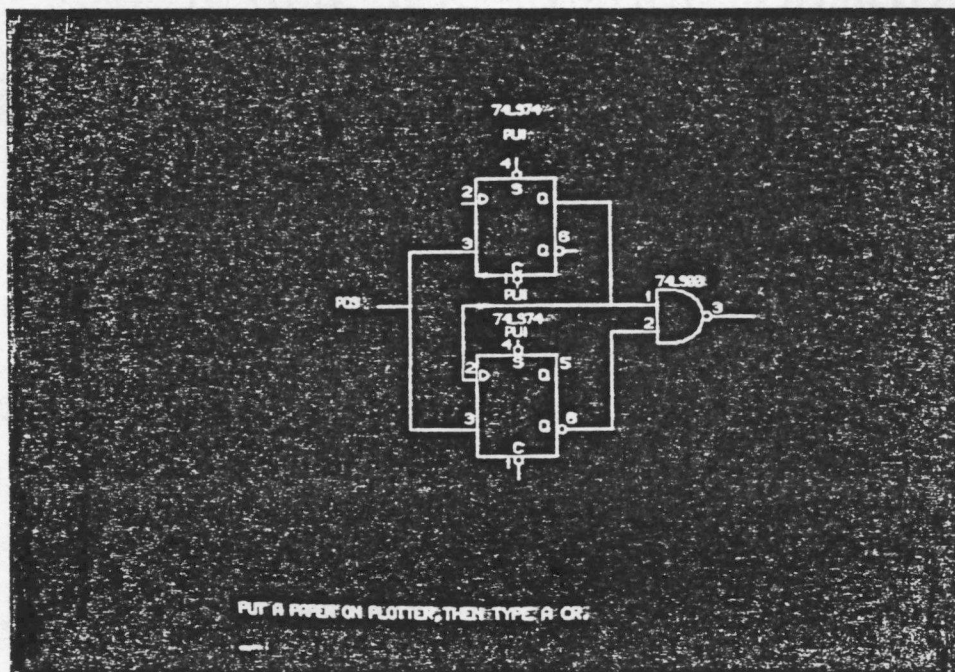
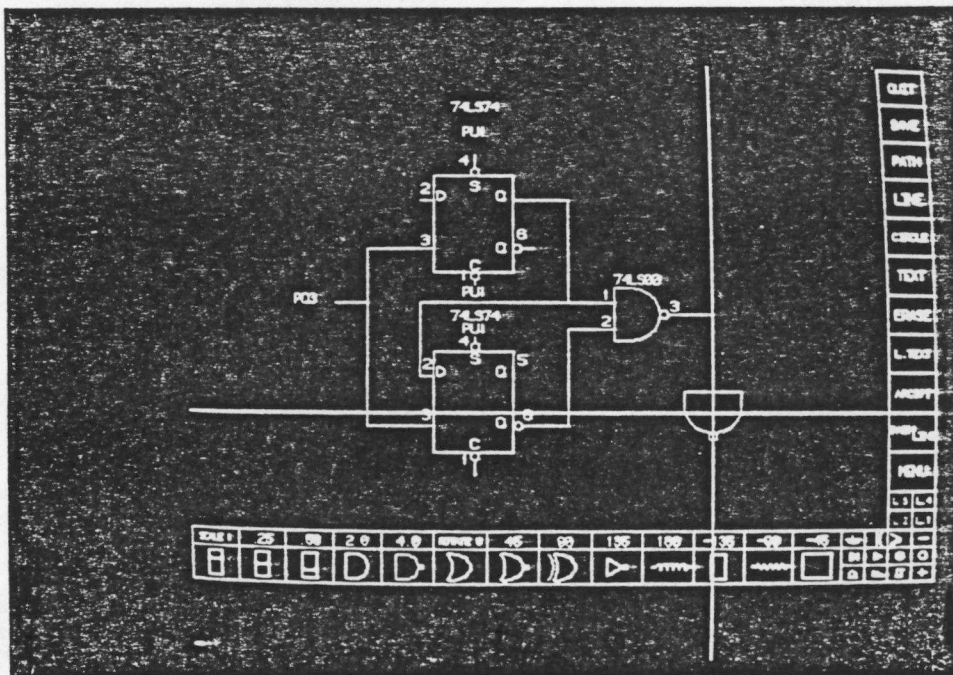


Figure A.5. Some experimental output figures.

APPENDIX B

OUTPUT DOCUMENTS

--- Up-down Octal Counter ---

CREATED, UPDATED: 10/21/83

***** NET LIST *****

NET NAME	SOURCE PIN	DESTINATION PINS	PAGE NO.	REFERENCE
*LED1	U02 - 8		0	[!] Output to the external device
*LED2	U02 -10		0	[!] Output to the external device
*LED3	U02 -12		0	[!] Output to the external device
*X1		U00 -13	0	[!] Input from the external device
*X1		U01 - 9	0	[!] Input from the external device
*X1		U01 - 1	0	[!] Input from the external device
*X2		U00 - 3	0	[!] Input from the external device
*X2		U01 - 4	0	[!] Input from the external device
*X2		U01 -10	0	[!] Input from the external device
-Q0	U04 - 6	U04 - 2	0	[!] Output to the external device
		U00 - 5	0	[!] Output to the external device
		U01 - 5	0	[!] Output to the external device
-Q1	U03 - 8	U00 - 4	0	[!] Output to the external device
		U03 -12	0	[!] Output to the external device
-Q2	U03 - 6	U03 - 2	0	[!] Output to the external device
CLK1	U02 - 4	U03 -11	0	[!] Output to the external device
CLK2	U02 - 2	U03 - 3	0	[!] Output to the external device
D01	U00 - 6	U00 -10	0	[!] Output to the external device
D02	U00 -12	U00 -11	0	[!] Output to the external device
D03	U00 - 8	U02 - 1	0	[!] Output to the external device
D11	U01 - 3	U01 -13	0	[!] Output to the external device
D12	U01 - 6	U01 -12	0	[!] Output to the external device
D13	U01 -11	U02 - 3	0	[!] Output to the external device
D14	U01 - 8	U04 - 3	0	[!] Output to the external device
Q0	U04 - 5	U00 - 2	0	[!] Output to the external device
		U01 - 2	0	[!] Output to the external device
		U02 - 9	0	[!] Output to the external device
Q1	U03 - 9	U00 - 1	0	[!] Output to the external device
		U02 -11	0	[!] Output to the external device
Q2	U03 - 5	U02 -13	0	[!] Output to the external device

----- NOTE -----

[!] : OK , NOT AN ERROR
 [?] : ARE YOU SURE? , A POSSIBLE ERROR
 [x] : FATAL ERROR

Figure B.1. Output documents : net list.

```

      --- Up-down Octal Counter ---                      CREATED, UPDATED: 10/21/83

      *****      LIST OF MATERIAL      *****

ITEM PART      UNIT      UNIT      UNIT      QTY      TOTAL      TOTAL      TOTAL
NO.  NO.      AREA      COST      POWER      AREA      COST      POWER
-----
  1  74LS00      0.243  0.850  2.550   1    0.243  0.850  2.550
--- QUAD. 2-INPUT POS. NAND GATES : U01

  2  74LS04      0.243  0.800  2.000   1    0.243  0.800  2.000
--- HEX INVERTERS : U02

  3  74LS10      0.000  0.850  0.000   1    0.000  0.850  0.000
--- Triple 3-input Positive-NAND Gates : U00

  4  74LS74      0.000  0.850  0.000   2    0.000  1.700  0.000
--- Dual D-type Pos. Edge-Triggered Flip-Flop : U03 U04

-----
Total number of chips      :      5
Total area occupied       :      0.487 Sq. inch
Total cost estimated      :      4.200 Dollars
Total power dissipation   :      4.550 mW

```

Figure B.2. Output documents : list of material.

```

      ---- Up-Down Octal Counter ----                      CREATED, UPDATED: 01/30/83

      *****      CHECK LIST      *****

NET      Chip_PIN      IC      PAGE      CHECK
NAME      No.  NO.      TYPE      NO.      DESCRIPTION
-----
                                U02 - 6      74LS04      0 [?]Unused or No signal name specified
                                U04 - 8      74LS74      0 [?]Unused or No signal name specified
                                - 9                      [?]

GND      U00 - 7      74LS10      0 [!]
          U01 - 7      74LS00      0 [!]
          U02 - 7      74LS04      0 [!]
          U03 - 7      74LS74      0 [!]
          U04 - 7      74LS74      0 [!]
          11                      [!]

Vcc      U00 - 9      74LS10      0 [!]
          14                      [!]
          U01 -14      74LS00      0 [!]
          U02 - 5      74LS04      0 [!]
          14                      [!]
          U03 - 1      74LS74      0 [!]
          4                      [!]
          10                     [!]
          13                     [!]
          14                     [!]
          U04 - 1      74LS74      0 [!]
          4                      [!]
          10                     [!]
          12                     [!]
          13                     [!]
          14                     [!]

-----
NOTE
[!] : OK , NOT AN ERROR
[?] : ARE YOU SURE? , A POSSIBLE ERROR
[x] : FATAL ERROR

```

Figure B.3. Output documents : check list.

***** PIN ASSIGNMENT LIST *****

TYPE NO.	Chip NO.	PIN NO.	I/O TYPE	NET NAME	PAGE NO.	DESCRIPTION
74LS00	U01	- 1	I	*X1	0	QUAD. 2-INPUT POS. NAND GATES
		- 2	I	Q0	0	
		- 3	O	D11	0	
		- 4	I	*X2	0	
		- 5	I	-Q0	0	
		- 6	O	D12	0	
		- 7	I	GND	0	
		- 8	O	D14	0	
		- 9	I	*X1	0	
		-10	I	*X2	0	
		-11	O	D13	0	
		-12	I	D12	0	
		-13	I	D11	0	
		-14	I	Vcc	0	
74LS04	U02	- 1	I	D03	0	HEX INVERTERS
		- 2	O	CLK2	0	
		- 3	I	D13	0	
		- 4	O	CLK1	0	
		- 5	I	Vcc	0	
		- 6	O		0	
		- 7	I	GND	0	
		- 8	O	*LED1	0	
		- 9	I	Q0	0	
		-10	O	*LED2	0	
		-11	I	Q1	0	
		-12	O	*LED3	0	
		-13	I	Q2	0	
		-14	I	Vcc	0	
74LS10	U00	- 1	I	Q1	0	Triple 3-input Positive-NAND Gates
		- 2	I	Q0	0	
		- 3	I	*X2	0	
		- 4	I	-Q1	0	
		- 5	I	-Q0	0	
		- 6	O	D01	0	
		- 7	I	GND	0	
		- 8	O	D03	0	
		- 9	I	Vcc	0	
		-10	I	D01	0	
		-11	I	D02	0	
		-12	O	D02	0	
		-13	I	*X1	0	
		-14	I	Vcc	0	
74LS74	U03	- 1	I	Vcc	0	Dual D-type Pos. Edge-Triggered Flip-F1
		- 2	I	-Q2	0	
		- 3	I	CLK2	0	
		- 4	I	Vcc	0	
		- 5	O	Q2	0	
		- 6	O	-Q2	0	
		- 7	I	GND	0	
		- 8	O	-Q1	0	
		- 9	O	Q1	0	
		-10	I	Vcc	0	
		-11	I	CLK1	0	
		-12	I	-Q1	0	
		-13	I	Vcc	0	
		-14	I	Vcc	0	
74LS74	U04	- 1	I	Vcc	0	Dual D-type Pos. Edge-Triggered Flip-F1
		- 2	I	-Q0	0	
		- 3	I	D14	0	
		- 4	I	Vcc	0	
		- 5	O	Q0	0	
		- 6	O	-Q0	0	
		- 7	I	GND	0	
		- 8	O		0	
		- 9	O		0	
		-10	I	Vcc	0	
		-11	I	GND	0	
		-12	I	Vcc	0	
		-13	I	Vcc	0	
		-14	I	Vcc	0	

Figure B.4. Output documents : pin assignment list.

APPENDIX C

PROGRAM LIST OF IC-MAP

```

Program IMAP;                                (* FILE NAME : SOURCE.IMAP *)

CONST  K_INIT = 1;                            K_TERM = 2;
       K_OPEN = 3;                            K_CLOSE = 4;
       K_READ_BY_KEY_LOCKED = 10;             K_READ_BY_KEY = 9;
       K_INSERT_REC = 19;                     K_REWRITE_REC = 20;
       K_REWRITE_REC_UN = 21;                 K_DELETE_BY_KEY = 24;
       NMAX = 80;                             HALF = 40;
       MINPIN = 14;                           MAXPIN = 24;
       BLANK = ' ';                           WORDLEN = 8;
       blankline = ' ';
       SPECCOL = 13;                           STR4LEN = 4;
       STR2LEN = 2;                           STR1LEN = 1;
       CROW = 12;                             CCOL = 40;
       FNAMELEN = 8;                           CONSTFLEN = 20;
TYPE   BYTE = 0..#FF;                          BTYPE = ARRAY[1..16] OF INTEGER;
       STRING = PACKED ARRAY[1..WORDLEN] OF CHAR;
       STRING4 = PACKED ARRAY[1..STR4LEN] OF CHAR;
       STRING2 = PACKED ARRAY[1..STR2LEN] OF CHAR;
       STRING1 = PACKED ARRAY[1..STR1LEN] OF CHAR;
       HALFLINE = PACKED ARRAY[1..HALF] OF CHAR;
       FNAME = PACKED ARRAY[1..FNAMELEN] OF CHAR;
       INPART = RECORD
           NAME : STRING;                      REFER : HALFLINE;
           AREA : STRING;                     COST : STRING;
           TYPCUR : STRING;                   MAXCUR : STRING;
           TYPDEL : STRING;                   MAXDEL : STRING;
           SIZE : STRING2;
           PINDES : ARRAY[1..MAXPIN] OF STRING4;
           INOUT : ARRAY[1..MAXPIN] OF STRING1;
       END;
       OUTPART = RECORD
           CIRDES : STRING4;                  PAGE : STRING1;
           NAME : STRING;
           SIGNAL : ARRAY[1..MAXPIN] OF STRING;
       END;
       SCI_STRING = PACKED ARRAY[0..NMAX] OF CHAR;
       LINEMODE = (THIN, THICK);
       SELECTION = (LIB, CRE, DEL, EDI, DIS, HEL, BYE, GUI, TTL, MIC, RAM, ROM, PRO, EPR,
                    PLA, NAN, NOR, ANDG, ORG, XOR, INV, FFP, ADD, MUL, SHI, T74, DAT, NUL);
       MENUSET = SET OF SELECTION;
       T_K_SBK = PACKED RECORD
           K_SCODE : 0..255;                  K_KEY_NUMBER : 0..255;
           K_ACCESS : 0..255;                 KIF_RESERVED : 0..255;
           K_REC_LEN : INTEGER;
           K_PATHNAME : PACKED ARRAY[1..30] OF CHAR;
       END;
VAR   EDITING, FOUND, DONE : BOOLEAN;
       STID : BYTE;                           BK : BTYPE;
       FILENAME : FNAME;                     EM : CHAR;
       SCISTRING : SCI_STRING;                LMODE : LINEMODE;
       INICPART : INPART;                    OUTICPART : OUTPART;
       MENUS : MENUSET;                      MENU : SELECTION;
       K_SBK : T_K_SBK;                      OK_SBK : T_K_SBK;
       KEY : STRING;
       PATHNAME : PACKED ARRAY[1..CONSTFLEN+FNAMELEN] OF CHAR;

PROCEDURE KEY$FILE(K_COMMAND: INTEGER;
                  UNIV K_KEYNAME : PACKED ARRAY[1..?] OF CHAR;
                  UNIV K_BUFFER_NAME : PACKED ARRAY[1..?] OF CHAR;
                  VAR K_SBK : T_K_SBK); EXTERNAL;
PROCEDURE SET$ACNM(VAR F: TEXT; VAR A: SCI_STRING); EXTERNAL;
PROCEDURE FIND$SYN(VAR S, V: SCI_STRING); EXTERNAL;
PROCEDURE INITS$CREEN(VAR BK: BTYPE; UNIT: INTEGER); EXTERNAL;
PROCEDURE DISPLAY(VAR BK: BTYPE; LINE, COLUMN: INTEGER;
                  BUFFER: PACKED ARRAY[1..?] OF CHAR;
                  LEN: INTEGER); EXTERNAL;
PROCEDURE ACCEPT(VAR BK: BTYPE; LINE, COLUMN: INTEGER;
                  VAR BUFFER: PACKED ARRAY[1..?] OF CHAR;
                  LEN: INTEGER;
                  VAR TERMINATION_CHARACTER: CHAR); EXTERNAL;
PROCEDURE CLEAR$CREEN(VAR BK: BTYPE); EXTERNAL;
FUNCTION STATIONID(): BYTE; EXTERNAL;

```

```

FUNCTION CHARIN( CHARACTER: STRING2 ): INTEGER;
VAR ZERO: INTEGER;
BEGIN
    ZERO := ORD('0');
    CHARIN := ( ORD(CHARACTER[1]) - ZERO ) * 10 + ( ORD(CHARACTER[2]) - ZERO )
END;

PROCEDURE CLEARWORD( VAR WORD: PACKED ARRAY[1..256] OF CHAR;
                    LEN: INTEGER );
VAR I : INTEGER;
BEGIN FOR I := 1 TO LEN DO WORD[I] := ' ' END;

PROCEDURE REALNUMCHECK( WORD: STRING; VAR FOUND: BOOLEAN );
VAR I : INTEGER;
BEGIN
    FOUND := FALSE;
    FOR I := 1 TO WORDLEN DO
        BEGIN
            IF WORD[I] = '.' THEN FOUND := TRUE;
            IF WORD[I] IN (['0'..'9'] - ['0'..'9', '.']) THEN FOUND := FALSE
        END;
        IF WORD = ' ' THEN FOUND := TRUE;
        IF NOT FOUND THEN CLEARWORD(WORD, WORDLEN)
    END;

PROCEDURE INCHAR( INTNUM: INTEGER; VAR ELEMENT: STRING2);
BEGIN
    ELEMENT[1] := CHR( ( INTNUM DIV 10 ) + ORD('0') );
    IF ELEMENT[1] = '0' THEN ELEMENT[1] := ' ';
    ELEMENT[2] := CHR( INTNUM - (INTNUM DIV 10) * 10 + ORD('0') )
END;

PROCEDURE VDT;
TYPE VDTNAME = @VDT_REC;
VDT_REC = PACKED ARRAY[0..41] OF CHAR;
SVCB = RECORD
    CODE, LUND, DUMYS, DUMY1, DUMY2, DUMY3 : INTEGER;
    ZERO, DUMY4, FLAGS, ZERO1, ZERO2 : INTEGER;
    VDTNA : VDTNAME;
    ZERO3, ZERO4 : INTEGER
END;

VAR STID : BYTE; BK : BTYPE;
    SVCBK: SVCB; SSTID : PACKED ARRAY[1..41] OF CHAR;

PROCEDURE SVC*(VAR B: SVCB); EXTERNAL;

BEGIN (* PROC VDT *)
    STID := STATIONID;
    WITH SVCBK DO
        BEGIN
            NEW(VDTNA);
            CODE := #0000; LUND := #9100;
            ZERO := #0000; FLAGS := #0400; ZERO1 := #0000;
            ZERO2 := #0000; ZERO3 := #0000; ZERO4 := #0000;
            VDTNA[0] := CHR(4); VDTNA[1] := 'S'; VDTNA[2] := 'T';
            VDTNA[3] := CHR( STID DIV 10 + ORD('0') );
            VDTNA[4] := CHR( STID MOD 10 + ORD('0') );
        END;
        SVC*(SVCBK); STID := SVCBK.LUND - #9100;
        INITSCREEN(BK, STID);
    END (* PROC VDT *);

PROCEDURE MAINMENU;
BEGIN (* MAIN MENU *)
    CLEARSCREEN(BK);
    DISPLAY(BK, 1, 16,
        '***** GENERATION OF HARDWARE DIAGRAM. *****', 80);
    DISPLAY(BK, 3, 16,
        '[ ] LIBRARY : CREATE & DELETE DATA IN LIBRARY. ', 80);
    DISPLAY(BK, 5, 16,
        '[ ] USER : EDIT, CREATE, DELETE & DISPLAY USER FILES. ', 80);
    DISPLAY(BK, 6, 16,
        '[ ] DIS : DISPLAY AN IC SPECIFICATION. (Default) ', 80);
    DISPLAY(BK, 7, 16, '[ ] HEL : HELP ', 80);
    DISPLAY(BK, 8, 16, '[ ] BYE : EXIT THE PROGRAM. ', 80);
    DISPLAY(BK, 10, 16, '[0] TTL : TI TTL SERIES ', 80);
    DISPLAY(BK, 11, 16, '[1] MIC : MICROPROCESSOR ', 80);
    DISPLAY(BK, 12, 16, '[2] RAM : RAM ', 80);

```

```

    DISPLAY(BK, 13, 16, '[3] ROM : ROM', 80);
    DISPLAY(BK, 14, 16, '[4] PRO : PROM', 80);
    DISPLAY(BK, 15, 16, '[5] EPR : EPROM', 80);
    DISPLAY(BK, 16, 16, '[6] PLA : PLA', 80);
    DISPLAY(BK, 17, 16, '[7]', 80);
    DISPLAY(BK, 18, 16, '[8]', 80);
    DISPLAY(BK, 20, 16, '[ *** MAIN MENU *** ]', 80);
END; (* MAIN MENU *)

PROCEDURE FUNCTIONMENU;
BEGIN (* FUNCTIONAL MENU *)
    CLEARSCREEN(BK);
    DISPLAY(BK, 1, 20, 'GENERATION OF HARDWARE DIAGRAM', 55);
    DISPLAY(BK, 3, 16, '[0] NAND : NAND', 55);
    DISPLAY(BK, 4, 16, '[1] NOR : NOR', 55);
    DISPLAY(BK, 5, 16, '[2] AND : AND', 55);
    DISPLAY(BK, 6, 16, '[3] OR : OR', 55);
    DISPLAY(BK, 7, 16, '[4] INV : INVERTER', 55);
    DISPLAY(BK, 8, 16, '[5] XOR : EXCLUSIVE OR', 55);
    DISPLAY(BK, 9, 16, '[6] FFP : FLIP-FLOP', 55);
    DISPLAY(BK, 10, 16, '[7] ADD : ADDER', 55);
    DISPLAY(BK, 11, 16, '[8] MUL : MULTIPLIER', 55);
    DISPLAY(BK, 12, 16, '[9] SHI : SHIFT REGISTER', 55);
    DISPLAY(BK, 13, 16, '[10] DAT : MULTIPLEXER/DATA SELECTOR', 55);
    DISPLAY(BK, 20, 16, '[ *** TTL MENU *** ]', 55);
END; (* FUNCTIONAL MENU *)

PROCEDURE LIBRARYMENU;
BEGIN (* LIBRARY MENU *)
    CLEARSCREEN(BK);
    DISPLAY(BK, 1, 20, 'GENERATION OF HARDWARE DIAGRAM', 32);
    DISPLAY(BK, 3, 16, '[ ] CRE : Create an IC data in LIBRARY', 80);
    DISPLAY(BK, 4, 16, '[ ] EDI : Edit an IC data in LIBRARY', 80);
    DISPLAY(BK, 5, 16, '[ ] DEL : Delete an IC data in LIBRARY', 80);
    DISPLAY(BK, 6, 16, '[ ] DIS : Display an IC data in LIBRARY', 80);
    DISPLAY(BK, 7, 16, '[ ] HEL : HELP !', 80);
    DISPLAY(BK, 8, 16, '[ ] GUI : QUIT editing LIBRARY FILE', 80);
    DISPLAY(BK, 20, 16, '[ *** LIBRARY MENU *** ]', 80);
END;

PROCEDURE EDITMENU;
BEGIN (* EDIT MENU *)
    CLEARSCREEN(BK);
    DISPLAY(BK, 1, 20, 'GENERATION OF HARDWARE DIAGRAM', 32);
    DISPLAY(BK, 3, 16, '[ ] CRE : Create an IC data in USER FILE', 80);
    DISPLAY(BK, 4, 16, '[ ] EDI : Edit an IC data stored in USER FILE', 80);
    DISPLAY(BK, 5, 16, '[ ] DEL : Delete an IC data in USER FILE', 80);
    DISPLAY(BK, 6, 16, '[ ] DIS : Display an IC data in USER FILE', 80);
    DISPLAY(BK, 7, 16, '[ ] HEL : HELP !', 80);
    DISPLAY(BK, 8, 16, '[ ] GUI : QUIT editing USER FILE', 80);
    DISPLAY(BK, 20, 16, '[ *** USER MENU *** ]', 80);
END; (* EDIT MENU *)

PROCEDURE USERRESPONSE(VAR MENU: SELECTION; VAR MENUS: MENUSET;
                        VAR KEY: STRING);
VAR i: integer;
BEGIN (* Userresponse *)
    MENU := NUL;
    DISPLAY(BK, 23, 10, '[ ] Please enter an option or IC type no. :', 80);
    DISPLAY(BK, 23, 56, key, 80);
    REPEAT
        ACCEPT(BK, 23, 56, KEY, WORDLEN, EM);
        FOR i:=1 TO 2 DO IF key[i] IN ['a'..'z']
            THEN key[i]:=CHR(ORD(key[i])-32);
        IF (KEY[1]='7') AND (KEY[2]='4') THEN MENU:=T74;
        IF (KEY[1]='L') AND (KEY[2]='I') THEN MENU:=LIB;
        IF (KEY[1]='C') AND (KEY[2]='R') THEN MENU:=CRE;
        IF (KEY[1]='D') AND (KEY[2]='E') THEN MENU:=DEL;
        IF (KEY[1]='E') AND (KEY[2]='D') THEN MENU:=EDI;
        IF (KEY[1]='U') AND (KEY[2]='S') THEN MENU:=EDI;
        IF (KEY[1]='D') AND (KEY[2]='I') THEN MENU:=DIS;
        IF (KEY[1]='H') AND (KEY[2]='E') THEN MENU:=HEL;
        IF (KEY[1]='G') AND (KEY[2]='U') THEN MENU:=GUI;
        IF (KEY[1]='B') AND (KEY[2]='Y') THEN MENU:=BYE;
        IF (KEY[1]='T') AND (KEY[2]='T') THEN MENU:=TTL;
        IF (KEY[1]='M') AND (KEY[2]='I') THEN MENU:=MIC;
        IF (KEY[1]='R') AND (KEY[2]='A') THEN MENU:=RAM;
        IF (KEY[1]='R') AND (KEY[2]='O') THEN MENU:=ROM;
    UNTIL MENU <> NUL;
END;

```

```

IF (KEY[1]='P') AND (KEY[2]='R') THEN MENU:=PRO;
IF (KEY[1]='E') AND (KEY[2]='P') THEN MENU:=EPR;
IF (KEY[1]='P') AND (KEY[2]='L') THEN MENU:=PLA;
IF (KEY[1]='N' AND KEY[2]='A') THEN MENU:=NAN;
IF (KEY[1]='N' AND KEY[2]='O') THEN MENU:=NOR;
IF (KEY[1]='A' AND KEY[2]='N') THEN MENU:=ANDG;
IF (KEY[1]='O' AND KEY[2]='R') THEN MENU:=ORG;
IF (KEY[1]='X' AND KEY[2]='O') THEN MENU:=XOR;
IF (KEY[1]='I' AND KEY[2]='N') THEN MENU:=INV;
IF (KEY[1]='N' AND KEY[2]='F') THEN MENU:=FFP;
IF (KEY[1]='A' AND KEY[2]='D') THEN MENU:=ADD;
IF (KEY[1]='M' AND KEY[2]='U') THEN MENU:=MUL;
IF (KEY[1]='S' AND KEY[2]='H') THEN MENU:=SHI;
IF (KEY[1]='D' AND KEY[2]='A') THEN MENU:=DAT;
UNTIL ( MENU IN MENUS )
END; (* PROC USERRESPONSE *)

PROCEDURE UNDERLINE(ROW,COL,LEN: INTEGER);
VAR I: INTEGER;      ELEMENT : PACKED ARRAY [1..1] OF CHAR;
BEGIN
  ELEMENT[1] := CHR(#5F);
  FOR I:=1 TO LEN DO
    BEGIN
      DISPLAY(BK,ROW,COL,ELEMENT,1);      COL:= COL+1
    END;
  END;
END; (* PROC UNDERLINE *)

PROCEDURE HLINE(ROW,COL,LEN: INTEGER; LMODE: LINEMODE);
VAR I: INTEGER;      ELEMENT : PACKED ARRAY [1..1] OF CHAR;
BEGIN
  ELEMENT[1] := CHR(#16);
  IF LMODE=THIN THEN ELEMENT[1] := CHR(#2D);
  FOR I:=1 TO LEN DO
    BEGIN
      DISPLAY(BK,ROW,COL,ELEMENT,3);      COL:= COL+1
    END;
  END;
END; (* PROC HLINE *)

PROCEDURE VLINE(ROW,COL,LEN: INTEGER; LMODE: LINEMODE);
VAR I: INTEGER;      ELEMENT : PACKED ARRAY [1..1] OF CHAR;
BEGIN
  ELEMENT[1] :=CHR(#09);
  IF LMODE=THIN THEN ELEMENT[1] := CHR(#7C);
  FOR I:=1 TO LEN DO
    BEGIN
      DISPLAY(BK,ROW,COL,ELEMENT,3);      ROW:= ROW+1
    END;
  END;
END; (* PROC VLINE *)

PROCEDURE BOX(CROW,CCOL,WIDTH,HEIGHT: INTEGER);
VAR
  WIDTH2,HEIGHT2,RIGHTCOL,BOTROW,TOPROW,LEFTCOL : INTEGER;
  ELEMENT : PACKED ARRAY [1..1] OF CHAR;
BEGIN
  WIDTH2 := WIDTH DIV 2;      HEIGHT2:= HEIGHT DIV 2;
  TOPROW := CROW - HEIGHT2;   BOTROW := TOPROW + HEIGHT;
  LEFTCOL:= CCOL - WIDTH2;    RIGHTCOL := LEFTCOL + WIDTH;
  ELEMENT[1] := CHR(#0A);      DISPLAY(BK,TOPROW,LEFTCOL,ELEMENT,1);
  ELEMENT[1] := CHR(#1A);      DISPLAY(BK,TOPROW,RIGHTCOL,ELEMENT,1);
  ELEMENT[1] := CHR(#0D);      DISPLAY(BK,BOTROW,LEFTCOL,ELEMENT,1);
  ELEMENT[1] := CHR(#1D);      DISPLAY(BK,BOTROW,RIGHTCOL,ELEMENT,1);
  HLINE(TOPROW,LEFTCOL+1,WIDTH-1,THICK);
  HLINE(BOTROW,LEFTCOL+1,WIDTH-1,THICK);
  VLINE(TOPROW+1,LEFTCOL,HEIGHT-1,THICK);
  VLINE(TOPROW+1,RIGHTCOL,HEIGHT-1,THICK)
END; (* PROC BOX *)

PROCEDURE SPECDISPLAY;
BEGIN
  DISPLAY(BK,2,1,'Type Number: ', 80);
  DISPLAY(BK,3,1,'Discription: ', 80);
  DISPLAY(BK,4,1,'Area(sq.in): ', 80);
  DISPLAY(BK,5,1,'Price( $ ) : ', 80);
  DISPLAY(BK,6,1,'Ty. Power.mW: ', 80);
  DISPLAY(BK,7,1,'Mx. Power.mW: ', 80);
  DISPLAY(BK,8,1,'Ty. Delay.ns: ', 80);
  DISPLAY(BK,9,1,'Mx. Delay.ns: ', 80);
  DISPLAY(BK,10,1,'No. of pins: ', 80);
  WITH INICPART DO
    BEGIN

```

```

        DISPLAY(BK, 2, SPECCOL, NAME,      80);
        DISPLAY(BK, 3, SPECCOL, REFER,    80);
        DISPLAY(BK, 4, SPECCOL, AREA,      80);
        DISPLAY(BK, 5, SPECCOL, COST,      80);
        DISPLAY(BK, 6, SPECCOL, TYPCUR,    80);
        DISPLAY(BK, 7, SPECCOL, MAXCUR,    80);
        DISPLAY(BK, 8, SPECCOL, TYPDEL,    80);
        DISPLAY(BK, 9, SPECCOL, MAXDEL,    80);
        DISPLAY(BK, 10, SPECCOL, SIZE,     80);
    END; (* WITH *)
END; (* PROC SPECDISPLAY *)

PROCEDURE ICDISPLAY;
VAR
    TOPROWLABEL, TOPCOLLABEL, RHTROWLABEL,
    LFTROWLABEL, LFTCOLLABEL, RHTCOLLABEL,
    I, INSIZE, SIZE2, WIDTH, HEIGHT, WIDTH2, HEIGHT2 : INTEGER;
    ELEMENT : STRING2;
BEGIN
    CLEARSCREEN(BK);
    INSIZE:=CHARIN( INICPART.SIZE );
    SIZE2 := INSIZE DIV 2;
    HEIGHT:=SIZE2 + 1;
    HEIGHT2:=HEIGHT DIV 2;
    WIDTH := HEIGHT + 2;
    WIDTH2:=WIDTH DIV 2;
    TOPROWLABEL:= CROW-HEIGHT2-1; TOPCOLLABEL:= CCOL-3;
    LFTROWLABEL:= CROW-HEIGHT2+1; RHTROWLABEL:= LFTROWLABEL;
    LFTCOLLABEL:= CCOL-WIDTH2-2; RHTCOLLABEL:= CCOL-WIDTH2+WIDTH+1;
    BOX( CROW, CCOL, WIDTH, HEIGHT );
    DISPLAY(BK, TOPROWLABEL, TOPCOLLABEL, INICPART. NAME, 10);
    FOR I:=1 TO SIZE2 DO
        BEGIN
            INCHAR( I, ELEMENT );
            UNDERLINE(LFTROWLABEL, LFTCOLLABEL-5, STR4LEN);
            DISPLAY(BK, LFTROWLABEL, LFTCOLLABEL, ELEMENT, 2);
            DISPLAY(BK, LFTROWLABEL, LFTCOLLABEL+3, INICPART. PINDES[I], 4);
            LFTROWLABEL := LFTROWLABEL+1
        END;
    FOR I:= INSIZE DOWNT0 SIZE2+1 DO
        BEGIN
            INCHAR( I, ELEMENT );
            UNDERLINE(RHTROWLABEL, RHTCOLLABEL+2, STR4LEN);
            DISPLAY(BK, RHTROWLABEL, RHTCOLLABEL, ELEMENT, 2);
            DISPLAY(BK, RHTROWLABEL, CCOL+1, INICPART. PINDES[I], STR4LEN);
            RHTROWLABEL := RHTROWLABEL+1
        END;
    END; (* DISPLAY IC *)

PROCEDURE SIGDISPLAY;
VAR
    BOTROWLABEL, BOTCOLLABEL, RHTROWLABEL,
    LFTROWLABEL, LFTCOLLABEL, RHTCOLLABEL,
    I, INSIZE, SIZE2, WIDTH, HEIGHT, WIDTH2, HEIGHT2 : INTEGER;
    ELEMENT : STRING2;
BEGIN
    INSIZE:=CHARIN( INICPART.SIZE );
    SIZE2 := INSIZE DIV 2;
    HEIGHT := SIZE2 + 1;
    HEIGHT2:=HEIGHT DIV 2;
    WIDTH := HEIGHT + 2;
    WIDTH2:=WIDTH DIV 2;
    BOTROWLABEL:= CROW+HEIGHT2+1;
    BOTCOLLABEL:= CCOL-2;
    LFTROWLABEL:= CROW-HEIGHT2+1;
    RHTROWLABEL:= LFTROWLABEL;
    LFTCOLLABEL:= CCOL-WIDTH2-2;
    RHTCOLLABEL:= CCOL-WIDTH2+WIDTH+1;
    DISPLAY(BK, 23, 20, 'Edit user file as shown : ', 80);
    DISPLAY(BK, BOTROWLABEL, BOTCOLLABEL, OUTICPART. CIRDSE, 80);
    LFTROWLABEL := CROW-HEIGHT2+1;
    RHTROWLABEL := LFTROWLABEL;
    FOR I:= 1 TO SIZE2 DO
        BEGIN
            DISPLAY(BK, LFTROWLABEL, LFTCOLLABEL-13, OUTICPART. SIGNAL[I], 80);
            LFTROWLABEL := LFTROWLABEL+1
        END;
    FOR I:= INSIZE DOWNT0 SIZE2+1 DO
        BEGIN
            DISPLAY(BK, RHTROWLABEL, RHTCOLLABEL+6, OUTICPART. SIGNAL[I], 80);
            RHTROWLABEL := RHTROWLABEL+1
        END;
    END; (* PROC SIGDISPLAY *)

```

```

PROCEDURE Checksig( VAR signal:string );
VAR i : integer;
BEGIN
  IF signal<>blank then
    BEGIN
      write( signal );
      WHILE signal[i]=' ' DO FOR I:=1 TO 7 DO signal[i]:=signal[i+1];
      write( '----', signal );
      IF signal[i]='G' OR signal[i]='V' THEN
        BEGIN
          CASE signal[i] OF
            'G' : BEGIN
              IF signal[2]='N' and signal[3]='D' THEN
                signal:=' GND ';
              END;
            'V' : BEGIN
              IF signal[2]='c' AND signal[3]='c' OR
                signal[2]='C' AND signal[3]='C' THEN
                signal:=' Vcc ';
              END;
            END;
          writeln( '----', signal );
        END;
      END;
    END; (* Checksig *)

PROCEDURE GETSIGNAL;
VAR
  BOTROWLABEL, BOTCOLLABEL, RHTROWLABEL,
  LFTROWLABEL, LFTCOLLABEL, RHTCOLLABEL,
  I, INSIZE, SIZE2, WIDTH, HEIGHT, WIDTH2, HEIGHT2 : INTEGER;
  ELEMENT : STRING2;
BEGIN
  INSIZE:=CHARIN( INICPART.SIZE );
  SIZE2 := INSIZE DIV 2;
  HEIGHT := SIZE2 + 1;
  WIDTH := HEIGHT + 2;
  BOTROWLABEL:= CROW+HEIGHT2+1;
  LFTROWLABEL:= CROW-HEIGHT2+1;
  LFTCOLLABEL:= CCOL-WIDTH2-2;
  WITH OUTICPART DO
    BEGIN
      IF EDITING = FALSE THEN
        BEGIN
          DISPLAY(BK, 23, 10,
            '[ ] Enter a circuit designator ( eg: U12 ).', 80);
          DISPLAY(BK, BOTROWLABEL, BOTCOLLABEL, CIRDEN, STR4LEN);
          ACCEPT(BK, BOTROWLABEL, BOTCOLLABEL, CIRDEN, STR4LEN, EM)
        END;
      IF CIRDEN[1]<>'G' OR CIRDEN[2]<>'U' THEN
        BEGIN
          DISPLAY(BK, 24, 10,
            'Begin I/O signal from/to external with * or @.', 80);
          DISPLAY(BK, 23, 10,
            '[ ] Enter signal names on pins.', 80);
          LFTROWLABEL := CROW-(HEIGHT DIV 2)+1;
          FOR I:= 1 TO SIZE2 DO
            BEGIN
              IF INICPART.INOUT[I,1]<>'N' THEN
                BEGIN
                  IF I=SIZE2 AND SIGNAL[I]=BLANK THEN SIGNAL[I]:=' GND ';
                  DISPLAY(BK, LFTROWLABEL, LFTCOLLABEL-13, SIGNAL[I], 80);
                  ACCEPT(BK, LFTROWLABEL, LFTCOLLABEL-13, SIGNAL[I], WORDLEN, EM);
                  Checksig(signal[i]); WRITELN('@', SIGNAL[I], '@');
                END
              ELSE
                BEGIN
                  SIGNAL[I]:=BLANK;
                  DISPLAY(BK, LFTROWLABEL, LFTCOLLABEL-13, SIGNAL[I], 80)
                END;
              LFTROWLABEL := LFTROWLABEL+1
            END;
          RHTROWLABEL := LFTROWLABEL-1;
          FOR I:= SIZE2+1 TO INSIZE DO
            BEGIN
              IF INICPART.INOUT[I,1]<>'N' THEN
                BEGIN
                  IF I=INSIZE AND SIGNAL[I]=BLANK THEN SIGNAL[I]:=' Vcc ';
                  DISPLAY(BK, RHTROWLABEL, RHTCOLLABEL+6, SIGNAL[I], 80);
                  ACCEPT(BK, RHTROWLABEL, RHTCOLLABEL+6, SIGNAL[I], WORDLEN, EM);
                END;
            END;
          END;
        END;
      END;
    END;
  END;

```

```

        Checksig(signal[i]);      WRITELN('e',SIGNAL[i], 'e');
    END
ELSE
    BEGIN
        SIGNAL[i] := BLANK;
        DISPLAY(BK, LFTROWLABEL, LFTCOLLABEL+6, SIGNAL[i], 80);
    END;
    RHTROWLABEL := RHTROWLABEL-1
END;
END; (* NOT QUIT *)
END; (* WITH *)
END; (* PROC GETSIGNAL *)

PROCEDURE SPECACCEPT( VAR JSIZE: INTEGER );
LABEL 20;
VAR I : INTEGER;                ISTRING : STRING2;

PROCEDURE SPECLINE( K: INTEGER; VAR KEY: STRING );
LABEL 10;
BEGIN
    10: ACCEPT(BK, K, SPECCOL, KEY, WORDLEN, EM);
    REALNUMCHECK( KEY, FOUND );
    IF NOT FOUND THEN
        BEGIN
            DISPLAY(BK, K, 30, '** Enter real number OR blank **', 80);
            GOTO 10
        END
    ELSE DISPLAY(BK, K, 30, blankline, 80)
END;
BEGIN
WITH INICPART DO
    BEGIN
        SPECLINE( 4, AREA );                SPECLINE( 5, COST );
        SPECLINE( 6, TYPCUR );                SPECLINE( 7, MAXCUR );
        SPECLINE( 8, TYPDEL );                SPECLINE( 9, MAXDEL );
    20: ACCEPT(BK, 10, SPECCOL, SIZE, STR2LEN, EM);
        JSIZE := CHARIN( SIZE );
        IF JSIZE < MINPIN OR JSIZE > MAXPIN OR ( JSIZE MOD 2 ) <> 0 THEN
            BEGIN
                DISPLAY(BK, 10, 20,
                    '** Enter an even no. between          **', 80);
                INCHAR(MINPIN, ISTRING);      DISPLAY(BK, 10, 49, ISTRING, 80);
                DISPLAY(BK, 10, 52, 'AND', 80); INCHAR(MAXPIN, ISTRING);
                DISPLAY(BK, 10, 57, ISTRING, 80);
                GOTO 20
            END;
            DISPLAY(BK, 10, 20, blankline, 80)
        END; (* WITH INICPART *)
    END; (* SPECACCEPT *)

PROCEDURE Askcr;
BEGIN
    Display(bk, 23, 40, '<CR> to continue', 80); Accept(bk, 23, 60, key, 1, em)
END;

PROCEDURE Errorlib;
BEGIN
    Display(bk, 22, 10, inicpart.name, 80);
    Display(bk, 22, 20, ': Not implemented in the library', 80);
    Askcr; Display(bk, 22, 10, blankline, 80)
END;

PROCEDURE READLIB ;
BEGIN
    FOUND := FALSE;
    (* READ RECORD *)
    KEY$FILE(K_READ_BY_KEY, INICPART.NAME, INICPART, K_SBK);
    IF K_SBK, K_SCODE=0 THEN FOUND := TRUE;
    IF NOT(FOUND) THEN Errorlib;
END; (* PROC READLIB *)

PROCEDURE T74DISPLAY;
BEGIN
    CLEARSCREEN(BK);                READLIB;
    IF FOUND THEN
        BEGIN
            ICDISPLAY;                SPECDISPLAY;                Askcr
        END;
    END; (* T74DISPLAY *)

```

```

PROCEDURE WRITELIB;
LABEL 5, 10, 20;
VAR I, J, STARTCOL, STARTROW : INTEGER;
BEGIN
  5: LIBRARYMENU;          MENUS := [ LIB., GUI, T74 ];
  USERRESPONSE(MENU, MENUS, INICPART.NAME);
  CLEARSCREEN(BK);
  CASE MENU OF
    GUI, bye, HEL : ;
    T74 : BEGIN      T74DISPLAY;      WRITELIB      END;
    CRE : BEGIN
      WITH INICPART DO
        BEGIN
          CLEARWORD( REFER, HALF );
          CLEARWORD( NAME, WORDLEN);      CLEARWORD( AREA, WORDLEN);
          CLEARWORD( COST, WORDLEN);      CLEARWORD( TYPCUR, WORDLEN);
          CLEARWORD( MAXCUR, WORDLEN);      CLEARWORD( TYPDEL, WORDLEN);
          CLEARWORD( MAXDEL, WORDLEN);      CLEARWORD( SIZE, STR2LEN );
          DISPLAY(BK, 1, 10, '[ Enter data as followings : ', 80);
          SPECDISPLAY;
          ACCEPT(BK, 2, SPECCOL, NAME, WORDLEN, EM);
          ACCEPT(BK, 3, SPECCOL, REFER, HALF, EM);
          SPECACCEPT(J);
          FOR I:=1 TO J DO
            BEGIN
              IF I<15 THEN
                BEGIN
                  STARTROW := I+10;      STARTCOL := 1
                END
              ELSE
                BEGIN
                  STARTROW := I-14;      STARTCOL := 50
                END;
              CLEARWORD( PINDES[I], STR4LEN );
              DISPLAY(BK, STARTROW, STARTCOL, 'PIN ', 4);
              INCHAR(I, SIZE);
              DISPLAY(BK, STARTROW, STARTCOL+4, SIZE, 3);
              DISPLAY(BK, STARTROW, STARTCOL+8, ': ', 2);
              IF I=(J DIV 2) THEN
                DISPLAY(BK, STARTROW, STARTCOL+10, 'GND ', 4);
              IF I=J THEN
                DISPLAY(BK, STARTROW, STARTCOL+10, 'Vcc ', 80);
                ACCEPT(BK, STARTROW, STARTCOL+10, PINDES[I], STR4LEN, EM);
                10: DISPLAY(BK, STARTROW, STARTCOL+20, 'I/O/C/N: I', 80);
                ACCEPT(BK, STARTROW, STARTCOL+28, INOUT[I], STR1LEN, EM);
                IF INOUT[I, 1]<>'I' AND INOUT[I, 1]<>'O' AND
                  INOUT[I, 1]<>'C' AND INOUT[I, 1]<>'N' OR I=1 THEN
                  BEGIN
                    DISPLAY(BK, STARTROW, STARTCOL+31,
                      '** I: input , O: output , N: no internal connection', 80);
                    DISPLAY(BK, STARTROW+1, STARTCOL+31,
                      ' C: open collector or 3-state output      **', 80);
                    GOTO 10
                  END;
                    DISPLAY(BK, STARTROW, STARTCOL+31, blankline, 80);
                    DISPLAY(BK, STARTROW+1, STARTCOL+31, blankline, 80);
                END; (* FOR *)
                KEY$FILE(K_INSERT_REC, NAME, INICPART.K_SBK);
                IF K_SBK.K_SCODE <> 0 THEN Errorlib;
                GOTO 5;
            END; (* WITH *)
          END; (* CRE *)
        ENDI :
        BEGIN
          WITH INICPART DO
            BEGIN
              DISPLAY(BK, 23, 20, '[ Enter an IC type number to edit : ', 80);
              ACCEPT(BK, 23, 60, NAME, WORDLEN, EM);
              KEY$FILE(K_READ_BY_KEY_LOCKED, NAME, INICPART.K_SBK);
              IF K_SBK.K_SCODE=0 THEN
                BEGIN
                  DISPLAY(BK, 1, 10, '[ Edit spec. as followings. ', 80);
                  SPECDISPLAY;      ACCEPT(BK, 3, SPECCOL, REFER, HALF, EM);
                  SPECACCEPT(J);
                  FOR I:=1 TO J DO
                    BEGIN
                      IF I<15 THEN

```

```

        BEGIN
            STARTROW := I+10;                STARTCOL := 1
        END
    ELSE
        BEGIN
            STARTROW := I-14;                STARTCOL := 50
        END;
        DISPLAY(BK, STARTROW, STARTCOL, 'PIN ', 4);
        INCHAR(I, SIZE);
        DISPLAY(BK, STARTROW, STARTCOL+4, SIZE, 3);
        DISPLAY(BK, STARTROW, STARTCOL+8, ': ', 2);
        DISPLAY(BK, STARTROW, STARTCOL+10, PINDES[I], 80);
        IF I=(J DIV 2) THEN
            DISPLAY(BK, STARTROW, STARTCOL+10, 'QND ', 4);
        IF I=J THEN
            DISPLAY(BK, STARTROW, STARTCOL+10, 'Vcc ', 80);
        ACCEPT(BK, STARTROW, STARTCOL+10, PINDES[I], STR4LEN, EM);
        DISPLAY(BK, STARTROW, STARTCOL+20, 'I/O/C/N: ', 80);
20:    ACCEPT(BK, STARTROW, STARTCOL+28, INOUT[I], 1);
        ACCEPT(BK, STARTROW, STARTCOL+28, INOUT[I], 1, EM);
        IF INOUT[I, 1]<>'I' AND INOUT[I, 1]<>'O' AND
            INOUT[I, 1]<>'C' AND INOUT[I, 1]<>'N' OR i=1 THEN
            BEGIN
                DISPLAY(BK, STARTROW, STARTCOL+31,
                    '** I: input , O: output , N: no internal connection', 80);
                DISPLAY(BK, STARTROW+1, STARTCOL+31,
                    'C: open collector or 3-state output      **', 80);
                GOTO 20
            END;
            DISPLAY(BK, STARTROW, STARTCOL+31, blankline, 80);
            DISPLAY(BK, STARTROW+1, STARTCOL+31, blankline, 80);
        END; (* FOR *)
        KEY$FILE(K_REWRITE_REC_UN, NAME, INICPART, K_SBK);
    END (* IF FOUND *)
    ELSE Errorlib;
    END; (* WITH *)
    GOTO 5
    END; (* EDI *)
DEL :
    BEGIN
        DISPLAY(BK, 23, 10, '[ ] Enter an IC type number to delete : ', 80);
        ACCEPT(BK, 23, 50, INICPART, NAME, WORDLEN, EM);
        KEY$FILE(K_READ_BY_KEY, INICPART, NAME, INICPART, K_SBK);
        IF K_SBK.K_SCODE=0 OR K_SBK.K_SCODE=180 THEN
            KEY$FILE(K_DELETE_BY_KEY, INICPART, NAME, INICPART, K_SBK)
        ELSE Errorlib;
        GOTO 5;
    END; (* DEL *)
DIS :
    BEGIN
        DISPLAY(BK, 23, 10, '[ ] Enter an IC type number to display : ', 80);
        ACCEPT(BK, 23, 50, INICPART, NAME, WORDLEN, EM);
        KEY$FILE(K_READ_BY_KEY, INICPART, NAME, INICPART, K_SBK);
        IF K_SBK.K_SCODE=0 OR K_SBK.K_SCODE=180 THEN
            BEGIN
                ICDISPLAY;          SPECDISPLAY;          Askcr
            END
        ELSE Errorlib;
        GOTO 5
    END; (* DIS *)
LIB : GOTO 5
END; (* CASE *)
END; (* Writelib *)

PROCEDURE EDIT;
LABEL 5, 10, 20;

PROCEDURE Erroruser;
BEGIN
    Display(bk, 22, 10, outicpart, circes, 80);
    Display(bk, 22, 20, 'No circuit/device designated in userfile', 80);
    Display(bk, 22, 10, blankline, 80);          Askcr;
END;

BEGIN
5:  EDITMENU;                MENUS := [ CRE., GUI, T74 ];
    USERRESPONSE( MENU, MENUS, INICPART, NAME );
    CLEARSCREEN( BK );
    CASE MENU OF
        GUI, bye, HEL : ;

```

```

T74 : BEGIN      T74DISPLAY;      GOTO 5      END;
CRE : BEGIN
      WITH OUTICPART DO
      BEGIN
        CLEARWORD( CIRDES,STR4LEN ); CLEARWORD( PAGE,STR1LEN );
        CLEARWORD( NAME,WORDLEN );
        FOR I:=1 TO MAXPIN DO
          CLEARWORD(OUTICPART.SIGNAL[I],WORDLEN);
10 :   CLEARSCREEN(BK);
        DISPLAY(BK,23,10,
          '[ ] Enter an IC type number to put in file : ',80);
        DISPLAY(BK,23,60,NAME,80);
        DISPLAY(BK,24,10,
          '[ ] Enter QU to quit making file. ',80);
        ACCEPT( BK,23,60,NAME,WORDLEN,EM);
        IF NAME[1]<>'Q' OR NAME[2]<>'U' THEN
          BEGIN
            INICPART.NAME:=NAME;      READLIB;
            IF FOUND THEN
              BEGIN
                ICDISPLAY;      SPECDISPLAY;
                DISPLAY(BK,23,10,
                  '[ ] Please, enter a page number (eg: PAGE5) : ',80);
                DISPLAY(BK,1,64,'PAGE',80); DISPLAY(BK,1,68,PAGE,80);
                DISPLAY(BK,24,10,
                  '[ ] Enter QU to quit making file. ',80);
                ACCEPT(BK,1,68,PAGE,STR1LEN,EM);
                IF PAGE[1]<>'Q' THEN
                  BEGIN
                    GETSIGNAL;
                    KEY$FILE(K_INSERT_REC,CIRDES,OUTICPART,OK_SBK);
                    IF OK_SBK.K_SCODE <>0 THEN
                      BEGIN
                        Clearscreen(bk);
                        DISPLAY(BK,23,20,
                          'The circuit designator duplicated. ',80);
                        Askcr;
                        GOTO 10
                      END;
                    END; (* NOT QUIT *)
                    END; (* IF FOUND *)
                    END; (* NOT QUIT FIRST CHECK *)
                    END; (* WITH OUTICPART *)
                    GOTO 5
                  END; (* CRE *)
EDI : BEGIN
      WITH OUTICPART DO
      BEGIN
        CLEARWORD( PAGE,STR1LEN ); CLEARWORD( CIRDES,STR4LEN );
20 :   CLEARSCREEN(BK);
        DISPLAY(BK,23,10,
          '[ ] Enter a circuit designator to edit ( eg: U15 ) : ',80);
        DISPLAY(BK,23,65,CIRDES,80);
        DISPLAY(BK,24,10,'[ ] Enter QU to quit editing file. ',80);
        ACCEPT(BK,23,65,CIRDES,STR4LEN,EM);
        IF CIRDES[1]<>'Q' OR CIRDES[2]<>'U' THEN
          BEGIN
            KEY$FILE(K_READ_BY_KEY_LOCKED,CIRDES,OUTICPART,OK_SBK);
            IF OK_SBK.K_SCODE<>0 THEN Erroruser
            ELSE
              BEGIN
                INICPART.NAME:=NAME;
                READLIB;      ICDISPLAY;
                SPECDISPLAY;      SIGDISPLAY;
                DISPLAY(BK,1,64,'PAGE',80); DISPLAY(BK,1,68,PAGE,80);
                ACCEPT(BK,1,68,PAGE,STR1LEN,EM); EDITING := TRUE;
                GETSIGNAL;      EDITING := FALSE;
                KEY$FILE(K_REWRITE_REC_UN,CIRDES,OUTICPART,OK_SBK);
                GOTO 20
              END; (* NOT QUIT *)
            END; (* NO QUIT FIRST *)
            END; (* WITH OUTICPART *)
            GOTO 5
          END; (* EDI *)
DEL : BEGIN
      WITH OUTICPART DO
      BEGIN
        CLEARWORD( CIRDES,STR4LEN );
        DISPLAY(BK,23,10,
          '[ ] Enter a circuit designator to delete (eg. U11) : ',80);

```

```

        ACCEPT(BK, 23, 65, CIRDES, STR4LEN, EM);
        KEY$FILE(K_READ_BY_KEY, CIRDES, OUTICPART, OK_SBK);
        IF OK_SBK.K_SCODE=0 THEN
            KEY$FILE(K_DELETE_BY_KEY, CIRDES, OUTICPART, OK_SBK)
        ELSE Erroruser;
    END; (* WITH OUTICPART *)
    GOTO 5
END; (* DEL *)
DIS : BEGIN
    WITH OUTICPART DO
    BEGIN
        DISPLAY(BK, 23, 10,
        '[ ] Enter a circuit designator to display (eg. U13) : ', 80);
        ACCEPT(BK, 23, 65, CIRDES, STR4LEN, EM);
        CLEARSCREEN(BK);
        KEY$FILE(K_READ_BY_KEY, CIRDES, OUTICPART, OK_SBK);
        KEY$FILE(K_READ_BY_KEY, NAME, INICPART, K_SBK);
        IF OK_SBK.K_SCODE = 0 THEN
            BEGIN
                ICDISPLAY;          SPECDISPLAY;
                SIGDISPLAY;         Askcr;
            END
            ELSE Erroruser;
        END; (* WITH OUTICPART *)
        GOTO 5
    END; (* DIS *)
END; (* CASE *)
END; (* PROC EDIT *)

PROCEDURE OPENFILES;
CONST STARTCOL1 = 10;          STARTCOL2 = 35;
PROCEDURE MAKESCI(F: PACKED ARRAY[1..?] OF CHAR; N: INTEGER;
    VAR S: SCI_STRING);
VAR I: INTEGER;
BEGIN
    S[0] := CHR(N); FOR I:=1 TO N DO S[I] := F[I];
END;

PROCEDURE GETFNAME( F: FNAME; VAR S: SCI_STRING);
CONST CONSTFNAME = ' . OUTFILE. ';
VAR I: INTEGER;
    UID, USERID : SCI_STRING;
BEGIN
    PATHNAME := CONSTFNAME;    USERID[0] := CHR(CONSTFLEN+FNAMLEN);
    MAKESCI('%UI', 4, UID);    FIND$SYN(UID, USERID);
    FOR I:=2 TO 4 DO PATHNAME[I] := USERID[I-1];
    FOR I:=6 TO 11 DO PATHNAME[I] := USERID[I-5];
    FOR I:=CONSTFLEN+1 TO CONSTFLEN+FNAMLEN DO
        PATHNAME[I] := F[I-CONSTFLEN];
    WRITELN(PATHNAME);    MAKESCI(PATHNAME, CONSTFLEN+FNAMLEN, S)
END;
BEGIN (* Openfiles *)
(* INITIALIZE KEY FILE ROUTINE *)
KEY$FILE(K_INIT, KEY, INICPART, K_SBK);
K_SBK.K_PATHNAME := 'VOL1 RES RES349 LIBFILE LIB74 ';
KEY$FILE(K_OPEN, KEY, INICPART, K_SBK);
DISPLAY(BK, 7, 5,
'[ ] Enter a output key filename to create/edit your IC design. ', 80);
DISPLAY(BK, 8, 5,
' This file will be input key file for CMAP (eg: KEYTEST): ', 80);
CLEARWORD( FILENAME, FNAMLEN );
ACCEPT(BK, 9, 8, FILENAME, FNAMLEN, EM);
GETFNAME(FILENAME, SCISTRING);
FOR I:=1 TO CONSTFLEN+FNAMLEN DO
    OK_SBK.K_PATHNAME[I] := SCISTRING[I];
KEY$FILE(K_OPEN, OUTICPART, CIRDES, OUTICPART, OK_SBK);
DISPLAY(BK, 11, 8, 'OUTPUT KEY FILE PATHNAME : ', 80 );
DISPLAY(BK, 11, 35, PATHNAME, 80);    Askcr;
K_SBK.K_KEY_NUMBER := 1;    INICPART.NAME := '0000LIBR';
KEY$FILE(K_INSERT_REC, INICPART, NAME, INICPART, K_SBK);
OK_SBK.K_KEY_NUMBER := 1;    OUTICPART.CIRDES := '0000';
KEY$FILE(K_INSERT_REC, OUTICPART, CIRDES, OUTICPART, OK_SBK);
inicpart.name := 'LIBRARY';    outicpart.cirdes := 'U00';
END;

PROCEDURE Closefiles;
BEGIN
    (* CLOSE FILES *)
    KEY$FILE(K_CLOSE, KEY, INICPART, K_SBK);
    KEY$FILE(K_CLOSE, KEY, OUTICPART, OK_SBK);

```

```

(* TERMINATE KEYFILE FUNCTION *)
KEY$FILE(K_TERM,KEY,INICPART,K_SBK);
CLEARSCREEN(BK);          DISPLAY(BK,11,35,PATHNAME,80);
DISPLAY(BK,11,15,'KEY FILE PATHNAME:',80);
DISPLAY(BK,13,15,'This file can be a input key file to CMAP',80);
END;

PROCEDURE MAINCASES;
LABEL 100;
BEGIN
  DONE := FALSE;
  REPEAT
    MAINMENU;
  100 : MENUS := [ LIB, DAT ];
    USERRESPONSE( MENU,MENUS,INICPART.NAME );
    CASE MENU OF

      LIB : Writelib;
      CRE,EDI,DEL : BEGIN          EDIT          END;
      DIS :
        BEGIN
          DISPLAY(BK,23,10,'Enter an IC type number to display:',80);
          ACCEPT(BK,23,30,INICPART.NAME,WORDLEN,EM);
          READLIB;
          IF FOUND THEN
            BEGIN
              ICDISPLAY;          SPECDISPLAY;          Askcr
            END;
          END ; (* DIS *)
        T74 : T74DISPLAY;
        TTL : BEGIN          FUNCTIONALMENU;          GOTO 100;          END;
        RAM :
          BEGIN
            CLEARSCREEN(BK);
            DISPLAY(BK,1,5,
              '[1] 256-BIT RAM(3-STATE OUTPUT)(256*1)',80);
            DISPLAY(BK,2,9,'74S201, 54S201',80);
            DISPLAY(BK,3,5,
              '[2] 256-BIT RAM(OPEN COL. OUTPUT)(256*1)',80);
            DISPLAY(BK,4,9,'74S301, 54S301',80);
            DISPLAY(BK,5,5,
              '[3] 64-BIT WORDS RAM(OPEN COL. OUTPUT)(16*1)',80);
            DISPLAY(BK,6,9,'74S289, 74B9, 54S289',80);
            GOTO 100
          END;

        ROM :
          BEGIN
            CLEARSCREEN(BK);
            DISPLAY(BK,1,5,
              '[1] 2048-BIT ROM(OPEN COL. OUTPUTS)',80);
            DISPLAY(BK,2,9,'74S270(512*4), 74S271(256*8)',80);
            DISPLAY(BK,3,9,'54S270(512*4), 54S271(256*8)',80);
            DISPLAY(BK,4,5,
              '[2] 2048-BIT ROM(3-STATE OUTPUT)',80);
            DISPLAY(BK,5,9,'74S370(512*4), 74S371(256*8)',80);
            DISPLAY(BK,6,9,'54S370(512*4), 54S371(256*8)',80);
            GOTO 100
          END;
        PRO,EPR,PLA,MIC: BEGIN          MAINMENU;          END;
        NAN :
          BEGIN
            CLEARSCREEN(BK);
            DISPLAY(BK,1,5,'[1] QUAD 2-INPUT POSITIVE-NAND GATES',80);
            DISPLAY(BK,2,9,'7400, 74H00, 74L00, 74LS00, 74S00',80);
            DISPLAY(BK,3,9,'5400, 54H00, 54L00, 54LS00, 54S00',80);
            DISPLAY(BK,4,5,'[2] TRI. 3-INPUT POSITIVE-NAND GATES',80);
            DISPLAY(BK,5,9,'74S10, 74H10, 74LS10, 7410, 74L10',80);
            DISPLAY(BK,6,9,'54S10, 54H10, 54LS10, 5410, 54L10',80);
            DISPLAY(BK,7,5,'[3] DUAL 4-INPUT POSITIVE-NAND GATES',80);
            DISPLAY(BK,8,9,'74S20, 74H20, 74LS20, 7420, 74L20',80);
            DISPLAY(BK,9,9,'54S20, 54H20, 54LS20, 5420, 54L20',80);
            DISPLAY(BK,10,5,'[4] 8-INPUT POSITIVE-NAND GATES',80);
            DISPLAY(BK,11,9,'74S30, 74H30, 74LS30, 7430, 74L30',80);
            DISPLAY(BK,12,9,'54S30, 54H30, 54LS30, 5430, 54L30',80);
            DISPLAY(BK,13,5,
              '[5] QUAD. 2-INPUT POSITIVE-NAND GATES(OPEN COL. OUTPUTS)',80);
            DISPLAY(BK,14,9,'7401, 74LS01',80);
            DISPLAY(BK,15,9,'5401, 54LS01, 54L01',80);
            GOTO 100
          END;
    END;
  UNTIL DONE;
END;

```

```

END;

NOR :
BEGIN
  CLEARSCREEN(BK);
  DISPLAY(BK,1,5,'[1] QUAD 2-INPUT POSITIVE-NOR GATES',80);
  DISPLAY(BK,2,9,'74S02, 74LS02, 7402, 74L02',80);
  DISPLAY(BK,3,9,'54S02, 54LS02, 5402, 54L02',80);
  DISPLAY(BK,4,5,'[2] TRI. 3-INPUT POSITIVE-NOR GATES',80);
  DISPLAY(BK,5,9,'7427, 74LS27, 5427, 54LS27',80);
  DISPLAY(BK,6,5,'[3] DUAL 4-INPUT POSITIVE-NOR GATES',80);
  DISPLAY(BK,7,9,'7425, 5425',80);
  DISPLAY(BK,8,5,'[4] DUAL 5-INPUT POSITIVE-NOR GATES',80);
  DISPLAY(BK,9,9,'74S260',80);
  GOTO 100
END;

ANDG :
BEGIN
  CLEARSCREEN(BK);
  DISPLAY(BK,1,5,'[1] QUAD 2-INPUT POSITIVE-AND GATES',80);
  DISPLAY(BK,2,9,'7408, 74LS08, 74S08',80);
  DISPLAY(BK,3,9,'5408, 54LS08, 54S08',80);
  DISPLAY(BK,4,5,'[2] TRI. 3-INPUT POSITIVE-AND GATES',80);
  DISPLAY(BK,5,9,'74S11, 74H11, 74LS11',80);
  DISPLAY(BK,6,9,'54S11, 54H11, 54LS11',80);
  DISPLAY(BK,7,5,'[3] DUAL 4-INPUT POSITIVE-AND GATES',80);
  DISPLAY(BK,8,9,'74H21, 74LS21',80);
  DISPLAY(BK,9,9,'54H21, 54LS21',80);
  DISPLAY(BK,10,5,'[4] QUAD 2-INPUT POSITIVE-AND GATES(OPEN COL. OUTPUTS)',80);
  DISPLAY(BK,11,9,'74S09, 7409, 74LS09',80);
  DISPLAY(BK,12,9,'54S09, 5409, 54LS09',80);
  DISPLAY(BK,13,5,'[5] TRI. 3-INPUT POSITIVE-AND GATES(OPEN COL. OUTPUTS)',80);
  DISPLAY(BK,14,9,'74S15, 74H15, 74LS15',80);
  DISPLAY(BK,15,9,'54S15, 54H15, 54LS15',80);
  GOTO 100
END;

ORG :
BEGIN
  CLEARSCREEN(BK);
  DISPLAY(BK,1,5,'[1] QUAD 2-INPUT POSITIVE-OR GATES',80);
  DISPLAY(BK,2,9,'7432, 74LS32, 74S32',80);
  DISPLAY(BK,3,9,'5432, 54LS32, 54S32',80);
  GOTO 100
END;

INV :
BEGIN
  CLEARSCREEN(BK);
  DISPLAY(BK,1,5,'[1] HEX INVERTERS',80);
  DISPLAY(BK,2,9,'7404, 74LS04, 74S04, 74H04, 74L04',80);
  DISPLAY(BK,3,9,'5404, 54LS04, 54S04, 54H04, 54L04',80);
  DISPLAY(BK,4,5,'[2] HEX INVERTERS(OPEN COL. OUTPUTS)',80);
  DISPLAY(BK,5,9,'7405, 74LS05, 74S05, 74H05',80);
  DISPLAY(BK,6,9,'5405, 54LS05, 54S05, 54H05',80);
  GOTO 100
END;

XOR, FFP, ADD, MUL, SHI, DAT, HEL, GUI : ;
BYE : DONE:=TRUE
END (* CASE *)
UNTIL DONE;
END; (* MAINCASE *)

BEGIN (* MAIN PROGRAM *)
VDT;
OPENFILES;
MAINCASES;
CLOSEFILES
END.

```

```

Program CMAP;                                (* FILE NAME : SOURCE.CMAP *)
(* MAKE OUTPUT FILES *)

CONST K_INIT = 1;                            K_TERM = 2;
      K_OPEN = 3;                            K_CLOSE = 4;
      K_READ_GREATER = 7;                   K_READ_BY_KEY = 9;
      MINPIN = 14;                          MAXPIN = 24;
      NMAX = 80;                            MAXNOCHIP = 10;
      HALF = 40;                            WORDLEN = 8;
      BLANK = ' ';                          STR4LEN = 4;
      STR2LEN = 2;                          STR1LEN = 1;
      FNAMELEN = 8;                         str9len = 9;
      dottedline=

-----
TYPE BYTE = 0..#FF;                          BTYPE = ARRAY[1..16] OF INTEGER;
      STRING = PACKED ARRAY[1..WORDLEN] OF CHAR;
      string9= PACKED ARRAY[1..str9len] OF CHAR;
      STRING4= PACKED ARRAY[1..STR4LEN] OF CHAR;
      STRING2= PACKED ARRAY[1..STR2LEN] OF CHAR;
      STRING1= PACKED ARRAY[1..STR1LEN] OF CHAR;
      HALF_LINE=PACKED ARRAY[1..HALF] OF CHAR;
      FNAME = PACKED ARRAY[1..FNAMELEN] OF CHAR;
      INPART = RECORD
        NAME : STRING;                      REFER: HALF_LINE;
        AREA : STRING;                      COST : STRING;
        TYPCUR : STRING;                    MAXCUR : STRING;
        TYPDEL : STRING;                    MAXDEL : STRING;
        SIZE : STRING2;
        PINDES : ARRAY[1..MAXPIN] OF STRING4;
        INOUT : ARRAY[1..MAXPIN] OF STRING1;
      END;
      T_SIGNAL = RECORD
        NAME : STRING;                      INOUT : STRING1;
        CIRDES : STRING4;                  ICTYPE: STRING;
        SIZE : STRING2;                    PINNUM: STRING2;
        PAGE : STRING1;
      END;
      OUTPART = RECORD
        CIRDES : STRING4;                    PAGE : STRING1;
        NAME : STRING;
        SIGNAL : ARRAY[1..MAXPIN] OF STRING;
      END;
      SCI_STRING = PACKED ARRAY[0..NMAX] OF CHAR;
      T_K_SBK = PACKED RECORD
        K_CODE : 0..255;                    K_KEY_NUMBER : 0..255;
        K_ACCESS : 0..255;                  KIF_RESERVED : 0..255;
        K_REC_LEN : INTEGER;
        K_PATHNAME : PACKED ARRAY[1..30] OF CHAR;
      END;

VAR sigcount, blanksignal, gndvcc : INTEGER;
    EM : CHAR;                             ISTRING : STRING2;
    BK : BTYPE;                             SCISTRING: SCI_STRING;
    OUTICPART : OUTPART;                    INICPART : INPART;
    NETSIGNAL : ARRAY[1..maxnochip*maxpin] OF T_SIGNAL;
    sortkey : ARRAY[1..maxnochip*maxpin] OF string9;
    NFILE, LFILE, CFILE, SFILE : TEXT;
    FILENAME : FNAME;                       HALFLINE : HALF_LINE;
    OK_SBK : T_K_SBK;                       K_SBK : T_K_SBK;
    KEY : STRING;

PROCEDURE KEY*FILE(K_COMMAND: INTEGER;
  UNIV K_KEYNAME : PACKED ARRAY[1..?] OF CHAR;
  UNIV K_BUFFER_NAME : PACKED ARRAY[1..?] OF CHAR;
  VAR K_SBK : T_K_SBK); EXTERNAL;
PROCEDURE SET*ACNM(VAR F: TEXT; VAR A: SCI_STRING); EXTERNAL;
PROCEDURE FIND*SYN(VAR S, V : SCI_STRING); EXTERNAL;
PROCEDURE DISPLAY(VAR BK: BTYPE; LINE, COLUMN: INTEGER;
  BUF: PACKED ARRAY[1..?] OF CHAR; LEN: INTEGER); EXTERNAL;
PROCEDURE ACCEPT(VAR BK: BTYPE; LINE, COLUMN: INTEGER;
  VAR BUFFER: PACKED ARRAY[1..?] OF CHAR;
  LEN: INTEGER; VAR TERM: CHAR); EXTERNAL;
PROCEDURE DELAY(MILLISECONDS: LONGINT); EXTERNAL;
PROCEDURE INITSCREEN(VAR BK: BTYPE; UNIT: INTEGER); EXTERNAL;

```

```

PROCEDURE CLEARSCREEN(VAR BK: BTYPE); EXTERNAL;
FUNCTION STATIONID(): BYTE; EXTERNAL;

PROCEDURE CHARIN( CHARACTER: STRING2; VAR INTNUM: INTEGER );
BEGIN
    INTNUM := (ORD(CHARACTER[1]) - ORD('0')) * 10 + (ORD(CHARACTER[2]) - ORD('0'))
END;

PROCEDURE CLEARWORD(VAR WORD: PACKED ARRAY[1..?] OF CHAR; LEN: INTEGER );
VAR I: INTEGER;
BEGIN (* CLEARWORD *)
    FOR I := 1 TO LEN DO WORD[I] := ' ';
END; (* CLEAR WORD *)

PROCEDURE INCHAR( INTNUM: INTEGER; VAR CHARACTER: STRING2 );
BEGIN
    CHARACTER[1] := CHR(( INTNUM DIV 10 ) + ORD('0'));
    IF CHARACTER[1] = '0' THEN CHARACTER[1] := ' ';
    CHARACTER[2] := CHR( INTNUM - (INTNUM DIV 10) * 10 + ORD('0') )
END;

PROCEDURE CHARREAL(WORD: STRING; VAR REALNUM: REAL);
VAR DECI, FRAC: REAL; I, DECCOUNTER, FRCCOUNTER, COUNTER, ZERO: INTEGER;
BEGIN
    IF WORD <> BLANK THEN
        BEGIN
            COUNTER := 0; DECCOUNTER := 0; FRCCOUNTER := 0;
            REPEAT
                COUNTER := COUNTER + 1; DECCOUNTER := DECCOUNTER + 1;
            UNTIL COUNTER > WORDLEN OR WORD[DECCOUNTER + 1] = ' ';
            COUNTER := COUNTER + 1;
            REPEAT
                COUNTER := COUNTER + 1; FRCCOUNTER := FRCCOUNTER + 1;
            UNTIL COUNTER > WORDLEN OR WORD[COUNTER + 1] = ' ';
            ZERO := ORD('0'); DECI := 0;
            FOR I := 1 TO DECCOUNTER DO
                DECI := DECI + (ORD(WORD[I]) - ZERO) * (EXP((DECCOUNTER - I) * LN(10)));
            COUNTER := 0; FRAC := 0;
            FOR I := DECCOUNTER + 2 TO DECCOUNTER + FRCCOUNTER + 1 DO
                BEGIN
                    COUNTER := COUNTER + 1;
                    FRAC := FRAC + (ORD(WORD[I]) - ZERO) * (EXP((-COUNTER) * LN(10.0)));
                END;
            REALNUM := DECI + FRAC
        END
    ELSE REALNUM := 0.0
END; (* PROC CHARREAL *)

PROCEDURE VDT;
TYPE VDTNAME = @VDT_REC;
VDT_REC = PACKED ARRAY[0..4] OF CHAR;
SVCB = RECORD
    CODE, LUND, DUMYS, DUMY1, DUMY2, DUMY3: INTEGER;
    ZERO, DUMY4, FLAGS, ZERO1, ZERO2: INTEGER;
    VDTNA: VDTNAME; ZERO3, ZERO4: INTEGER;
END;
VAR STID: BYTE; BK: BTYPE;
SVCBK: SVCB; SSTID: PACKED ARRAY[1..4] OF CHAR;

PROCEDURE SVC*(VAR B: SVCB); EXTERNAL;

BEGIN (* PROC VDT *)
    STID := STATIONID;
    WITH SVCBK DO
        BEGIN
            NEW(VDTNA);
            CODE := #0000; LUND := #9100;
            ZERO := #0000; FLAGS := #0400; ZERO1 := #0000;
            ZERO2 := #0000; ZERO3 := #0000; ZERO4 := #0000;
            VDTNA[0] := CHR(4); VDTNA[1] := 'S'; VDTNA[2] := 'T';
            VDTNA[3] := CHR( STID DIV 10 + ORD('0') );
            VDTNA[4] := CHR( STID MOD 10 + ORD('0') );
        END;
    SVC*(SVCBK); STID := SVCBK.LUND - #9100;
    INITSCREEN(BK, STID);
END (* PROC VDT *);

```

```

PROCEDURE Openfiles;
CONST startcol1 = 10;          startcol2 = 35;
      constflen = 20;
VAR   pathname : PACKED ARRAY[1..constflen+fnamelen] OF char;

PROCEDURE Makesci(f:PACKED ARRAY[1..?] OF char; n:integer;
                   VAR s:sci_string);
VAR i:integer;
BEGIN
  s[0]:=CHR(n); FOR i:=1 TO n DO s[i]:=f[i];
END;

PROCEDURE GETFNAME( F:FNAME; VAR S:SCI_STRING);
CONST CONSTFNAME = '          ' OUTFILE;
VAR   I: INTEGER;
      UID,USERID : SCI_STRING;
BEGIN
  PATHNAME := CONSTFNAME;   USERID[0]:=CHR(CONSTFLEN+FNAMLEN);
  MAKESCI('$$UI',4,UID);    FIND$SYN(UID,USERID);
  FOR I:=2 TO 4 DO PATHNAME[I]:=USERID[I-1];
  FOR I:=6 TO 11 DO PATHNAME[I]:=USERID[I-5];
  FOR I:=CONSTFLEN+1 TO CONSTFLEN+FNAMLEN DO
    PATHNAME[I]:=F[I-CONSTFLEN];
  WRITELN(PATHNAME);
  MAKESCI(PATHNAME,CONSTFLEN+FNAMLEN,S)
END;
BEGIN (* OPENFILES *)
  KEY$FILE(K_INIT,KEY,INICPART,K_SBK);
  K_SBK.K_PATHNAME:= 'VOL1.RES.RES549.LIBFILE.LIB74 ';
  KEY$FILE(K_OPEN,KEY,INICPART,K_SBK);
  K_SBK.K_KEY_NUMBER:=1;

  DISPLAY(BK,7,5,
  '[ ] Enter a source key filename to make all the output files',80);
  DISPLAY(BK,8,8, '(eg: TESTMAP) : ',80);
  DISPLAY(BK,9,5,
  '[ ] Enter a title for your design(eg:Up-down Octal Counter)',80);
  DISPLAY(BK,11,5,
  '[ ] Enter a filename to save net list (eg:NETMAP) : ',80);
  DISPLAY(BK,11,68, 'NET',80);
  DISPLAY(BK,12,5,
  '[ ] Enter a filename to save list of material(eg:LISTMAP): ',80);
  DISPLAY(BK,12,68, 'LIST',80);
  DISPLAY(BK,13,5,
  '[ ] Enter a filename to save check-list (eg:CHECKMAP) : ',80);
  DISPLAY(BK,13,68, 'CHECK',80);
  DISPLAY(BK,14,5,
  '[ ] Enter a filename to save pin assignment list(eg:PINMAP) : ',80);
  DISPLAY(BK,14,68, 'PIN',80);

  CLEARWORD( FILENAME,FNAMLEN );
  ACCEPT(BK,8,24,FILENAME,FNAMLEN,EM);
  GETFNAME(FILENAME,SCISTRING);
  FOR I:=1 TO CONSTFLEN+FNAMLEN DO
    OK_SBK.K_PATHNAME[I]:=SCISTRING[I];
  KEY$FILE(K_OPEN,OUTICPART.CIRDES,OUTICPART,OK_SBK);
  OK_SBK.K_KEY_NUMBER := 1;

  CLEARWORD( HALFLINE,HALF );
  ACCEPT(BK,10,8,HALFLINE,HALF,EM);

  CLEARWORD( FILENAME,FNAMLEN );
  ACCEPT(BK,11,68,FILENAME,FNAMLEN,EM);
  GETFNAME(FILENAME,SCISTRING);
  SET$ACNM(NFILE,SCISTRING);
  DISPLAY(BK,16,STARTCOL1, 'OUTPUT FILE PATHNAME : ',80);
  DISPLAY(BK,16,STARTCOL2,PATHNAME,80);
  REWRITE(NFILE);

  CLEARWORD( FILENAME,FNAMLEN );
  ACCEPT(BK,12,68,FILENAME,FNAMLEN,EM);
  GETFNAME(FILENAME,SCISTRING);
  SET$ACNM(LFILE,SCISTRING);
  DISPLAY(BK,17,STARTCOL1, 'OUTPUT FILE PATHNAME : ',80);
  DISPLAY(BK,17,STARTCOL2,PATHNAME,80);
  REWRITE(LFILE);

  CLEARWORD( FILENAME,FNAMLEN );
  ACCEPT(BK,13,68,FILENAME,FNAMLEN,EM);
  GETFNAME(FILENAME,SCISTRING);

```

```

SET$ACNM(CFILE, SCISTRING);
DISPLAY(BK, 18, STARTCOL1, 'OUTPUT FILE PATHNAME : ', 80);
DISPLAY(BK, 18, STARTCOL2, PATHNAME, 80);
REWRITE(CFILE);

CLEARWORD( FILENAME, FNAMELEN );
ACCEPT(BK, 14, 68, FILENAME, FNAMELEN, EM);
GETFNAME(FILENAME, SCISTRING);
SET$ACNM(SFILE, SCISTRING);
DISPLAY(BK, 19, STARTCOL1, 'OUTPUT FILE PATHNAME : ', 80);
DISPLAY(BK, 19, STARTCOL2, PATHNAME, 80);
REWRITE(SFILE);

DATE(KEY);
WRITELN(NFILE, ' ');
WRITELN(NFILE, ' ', HALFLINE, 'CREATED, UPDATED: ', KEY);
WRITELN(NFILE, ' ');
WRITELN(NFILE,
'
***** NET LIST ***** ');
WRITELN(NFILE, ' ');
WRITELN(NFILE,
'NET SOURCE DESTINATION PAGE REFERENCE ');
WRITELN(NFILE,
'NAME PIN PINS NO. ');
WRITELN(NFILE, dottedline);

WRITELN(CFILE, ' ');
WRITELN(CFILE, ' ', HALFLINE, 'CREATED, UPDATED: ', KEY);
WRITELN(CFILE, ' ');
WRITELN(CFILE,
'
***** CHECK LIST ***** ');
WRITELN(CFILE, ' ');
WRITELN(CFILE,
'NET Chip_PIN IC PAGE CHECK ');
WRITELN(CFILE,
'NAME No. NO. TYPE NO. DESCRIPTION ');
WRITELN(CFILE, dottedline);

WRITELN(LFILE, ' ');
WRITELN(LFILE, ' ', HALFLINE, 'CREATED, UPDATED: ', KEY);
WRITELN(LFILE, ' ');
WRITELN(LFILE,
'
***** LIST OF MATERIAL ***** ');
WRITELN(LFILE, ' ');
WRITELN(LFILE,
'ITEM PART UNIT UNIT UNIT QTY TOTAL TOTAL TOTAL ');
WRITELN(LFILE,
'NO. NO. AREA COST POWER AREA COST POWER ');
WRITELN(LFILE, dottedline);

WRITELN(SFILE, ' ');
WRITELN(SFILE, ' ', HALFLINE, 'CREATED, UPDATED: ', KEY);
WRITELN(SFILE, ' ');
WRITELN(SFILE,
'
***** PIN ASSIGNMENT LIST ***** ');
WRITELN(SFILE, ' ');
WRITELN(SFILE,
'TYPE Chip_PIN I/O NET PAGE DESCRIPTION ');
WRITELN(SFILE, 'NO. NO. NO. TYPE NAME NO. ');
WRITELN(SFILE, dottedline);
END; (* Openfiles *)

PROCEDURE Concatenate(first:string; second:char; VAR ksort:string9);
VAR i : integer; (* concatenate two keys in order *)
BEGIN
FOR i:=1 TO 8 DO ksort[i]:=first[i];
CASE second OF
'N' : ksort[9]:='A' ;
'C' : ksort[9]:='B' ;
'O' : ksort[9]:='C' ;
'I' : ksort[9]:='D' ;
END;
END; (* Concatenate *)

PROCEDURE Readnet(VAR sigcount:integer);
(* read library, user input file, make netsignal records*)
VAR i, pincount : INTEGER;
BEGIN
OUTICPART.CIRDES := '0000'; sigcount := 0;
KEY$FILE(K_READ_BY_KEY, OUTICPART.CIRDES, OUTICPART, OK_SBK);

```

```

WHILE OK_SBK.K_SCODE=0 DO
  BEGIN
    KEY$FILE(K_READ_GREATER,OUTICPART.CIRDES,OUTICPART,OK_SBK);
    IF OK_SBK.K_SCODE=0 THEN
      BEGIN
        KEY$FILE(K_READ_BY_KEY,OUTICPART.NAME,INICPART,K_SBK);
        CHARIN( INICPART.SIZE,PINCOUNT );
        FOR I:=1 TO PINCOUNT DO
          BEGIN
            SIGCOUNT:=SIGCOUNT+1; INCHAR(I,ISTRING);
            WITH NETSIGNAL[SIGCOUNT] DO
              BEGIN
                NAME :=OUTICPART.SIGNAL[I]; INOUT :=INICPART.INOUT[I];
                CIRDES :=OUTICPART.CIRDES; ICTYPE:=OUTICPART.NAME;
                SIZE := INICPART.SIZE; PINNUM:=ISTRING;
                PAGE := OUTICPART.PAGE;
                Concatenate(name,inout[I],sortkey[sigcount]);
              END; (* WITH *)
            END; (* FOR pincount *)
          END; (* IF *)
        END; (* WHILE *)
        sortkey[sigcount+1]:='zzzzzzzz';
      END; (* Readnet *)

      PROCEDURE Kswap(VAR a,b:string9); (* exchange two keys *)
      VAR c:string9;
      BEGIN
        c:=a; a:=b; b:=c;
      END;

      PROCEDURE Swap(VAR a,b:t_signal);
      VAR c:t_signal;
      BEGIN
        c:=a; a:=b; b:=c;
      END;

      PROCEDURE Quicksort(lowerbound,upperbound:integer);
      VAR i,j:integer; key:string9;
      BEGIN (* Quicksort *)
        IF lowerbound < upperbound THEN
          BEGIN
            i:=lowerbound;
            j:=upperbound+1;
            Key:=sortkey[lowerbound];
            WHILE i < j DO
              BEGIN
                REPEAT i:=i+1 UNTIL sortkey[i]>=Key;
                REPEAT j:=j-1 UNTIL sortkey[j]<=Key;
                IF i < j THEN
                  BEGIN
                    Swap(netsignal[i],netsignal[j]);
                    Kswap(sortkey[i],sortkey[j]);
                  END;
                END;
                Swap(netsignal[lowerbound],netsignal[j]);
                Kswap(sortkey[lowerbound],sortkey[j]);
                Quicksort(lowerbound,j-1);
                Quicksort(j+1,upperbound);
              END
            END; (* Quicksort *)

      PROCEDURE Bubblesort(lowerbound,upperbound:integer);
      VAR i,j,cpin,spin : integer;
      BEGIN
        FOR i:=lowerbound TO upperbound-1 DO
          BEGIN
            FOR j:=upperbound DOWNT0 i+1 DO
              BEGIN
                WITH netsignal[j] DO
                  BEGIN
                    IF ictype<netsignal[j-1].ictype THEN
                      Swap(netsignal[j],netsignal[j-1]);
                    IF ictype=netsignal[j-1].ictype THEN
                      BEGIN
                        IF cirdes<netsignal[j-1].cirdes
                        THEN
                          Swap(netsignal[j],netsignal[j-1])
                        ELSE
                          IF cirdes=netsignal[j-1].cirdes THEN
                            BEGIN

```

```

Charin(pinum, cpin);
Charin(netsignal[j-1], pinum, spin);
IF cpin<spin THEN
    Swap(netsignal[j], netsignal[j-1]);
END;
END;
END; (* with *)
END; (* downto *)
END; (* upperbound *)
END; (* Procedure Bubblesort *)

PROCEDURE Writefiles;
(* Write net list and some of check-list. *)
CONST ERRO = ' [?]Unused or No signal name specified';
ERR1 = ' [x]Signal associated with only one pin';
ERR2 = ' [x]No driving source';
ERR3 = ' [x]No destination specified';
ERR4 = ' [x]Signal duplicated, multiple driving sources';
MES0 = ' [?]'; MESN = ' [!]';
MES0 = ' [!]No internal connection';
MES1 = ' [!]Output to the external device';
MES2 = ' [!]Input from the external device';
MES3 = ' (Open collector output(s), 3-state output(s))';
blank2='';
VAR i, pincount, small, current, ncsignal, countc: integer;
BEGIN
    (* Count no. of no internal connection pins and
    pins which does not have a net name. *)
    ncsignal:=0; gndvcc:=0; blanksignal:=0;
    FOR I:=1 TO SIGCOUNT DO
        BEGIN
            WITH NETSIGNAL[I] DO
                BEGIN
                    IF INOUT[I]='N' THEN NCSIGNAL:=NCSIGNAL+1;
                    IF name=' GND ' OR name=' Vcc ' THEN GNDVCC:=GNDVCC+1;
                    IF NAME = BLANK THEN BLANKSIGNAL:=BLANKSIGNAL+1;
                END;
            END;
        END;
    (* Sort 'blank' net name by circuit designator and pin no. *)
    FOR I:=NCSIGNAL+1 TO GNDVCC-1 DO
        BEGIN
            SMALL:=I;
            FOR CURRENT:=I TO GNDVCC DO
                BEGIN
                    IF ( NETSIGNAL[CURRENT].CIRDES<NETSIGNAL[SMALL].CIRDES ) OR
                    ( NETSIGNAL[CURRENT].CIRDES=NETSIGNAL[SMALL].CIRDES AND
                    NETSIGNAL[CURRENT].PINNUM<NETSIGNAL[SMALL].PINNUM ) THEN
                        SMALL := CURRENT
                END;
            Swap(netsignal[small], netsignal[i]);
            END; (* NCSIGNAL+1 *)
        END;
    (* Start writing check-list , list pins with no internal connection *)
    FOR I:=1 TO NCSIGNAL DO
        BEGIN
            WITH NETSIGNAL[I] DO
                BEGIN
                    WRITE(CFILE, NAME, ' ', CIRDES, '-', PINNUM, ' ', ICTYPE, ' ', page);
                    IF i=1 THEN WRITELN(CFILE, MES0) ELSE WRITELN(CFILE, MESN);
                END; (* WITH *)
            END; (* FOR NCSIGNAL *)
            WRITELN(CFILE, ' ');
            (* List Vcc, GND pins *)
            FOR I:=NCSIGNAL+1 TO BLANKSIGNAL DO
                BEGIN
                    WITH NETSIGNAL[I] DO
                        BEGIN
                            IF I=NCSIGNAL+1 OR NETSIGNAL[I-1].CIRDES<>CIRDES THEN
                                WRITELN(CFILE,
                                NAME, ' ', CIRDES, '-', PINNUM, ' ', ICTYPE, ' ', PAGE, ERRO);
                            IF NETSIGNAL[I-1].CIRDES=CIRDES THEN WRITELN(CFILE,
                                NAME, ' ', ' ', '-', PINNUM, ' ', MES0)
                            END;
                        END;
                    END;
                    WRITELN(CFILE, ' ');
                    FOR I:= BLANKSIGNAL+1 TO BLANKSIGNAL+GNDVCC DO
                        BEGIN
                            WITH NETSIGNAL[I] DO
                                BEGIN
                                    FOR I:=1 TO WORDLEN-2 DO NAME[I]:=NAME[I+2];

```

```

        IF NAME<>NETSIGNAL[I-1].NAME THEN WRITE(CFILE,NAME)
        ELSE WRITE(CFILE, ' ');
        WRITELN(CFILE, ' ', CIRDES, '-', PINNUM, ' ', ICTYPE, ' ', PAGE, MESN)
    END;
END;
WRITELN(CFILE, ' ');

countc:=0;
FOR I:=BLANKSIGNAL+GNDVCC+1 TO SIGCOUNT DO
BEGIN
    WITH NETSIGNAL[I] DO
    BEGIN
        IF NAME[I]='*' OR NAME[I]='e' THEN
        BEGIN
            CASE INOUT[I] OF
                'O': WRITELN(NFILE,
                    NAME, ' ', CIRDES, '-', PINNUM, ' ', PAGE, MES1);
                'C': BEGIN
                    WRITELN(NFILE, NAME, ' ', CIRDES, '-', PINNUM,
                        ' ', PAGE, MES1);
                    WRITELN(NFILE, blank2, MES3);
                END;
                'I': WRITELN(NFILE,
                    NAME, ' ', CIRDES, '-', PINNUM, ' ', PAGE, MES2);
            END; (* CASE *)
        END
    ELSE
        IF NETSIGNAL[I-1].NAME=NAME OR NETSIGNAL[I+1].NAME=NAME THEN
        BEGIN
            IF INOUT[I]='C' THEN
            BEGIN
                IF NETSIGNAL[I+1].NAME=NAME THEN
                BEGIN
                    IF NETSIGNAL[I+1].INOUT[I]='C' THEN
                        countc:=countc+1;
                    IF NETSIGNAL[I+1].INOUT[I]='I' THEN
                    BEGIN
                        FOR J:=countc DOWNT0 0 DO
                        BEGIN
                            IF J=countc THEN
                                WRITELN(NFILE, NAME, ' ', NETSIGNAL[I-J].CIRDES, '-',
                                    NETSIGNAL[I-J].PINNUM, ' ',
                                    NETSIGNAL[I-J].PAGE, MES3)
                            ELSE
                                WRITELN(NFILE, ' ', NETSIGNAL[I-J].CIRDES,
                                    '-', NETSIGNAL[I-J].PINNUM,
                                    ' ', NETSIGNAL[I-J].PAGE, MESN)
                        END;
                        countc:=0
                    END;
                END; (* INOUT=C, [I+1]NAME=NAME *)
                IF (NETSIGNAL[I+1].NAME=NAME AND NETSIGNAL[I+1].INOUT[I]='O')
                OR (NETSIGNAL[I+1].NAME<>NAME) THEN
                BEGIN
                    FOR J:=countc DOWNT0 0 DO
                    BEGIN
                        WRITELN(CFILE, NAME, ' ', NETSIGNAL[I-J].CIRDES, '-',
                            NETSIGNAL[I-J].PINNUM, ' ', ICTYPE, ' ',
                            NETSIGNAL[I-J].PAGE, ERR4);
                        WRITELN(CFILE, blank2, ERR3); WRITELN(CFILE, blank2, mes3)
                    END;
                    countc:=0
                END;
            END (* INOUT=C *)
        ELSE (* INOUT<>C *)
        BEGIN
            IF INOUT[I]='O' THEN
            BEGIN
                IF NETSIGNAL[I+1].INOUT[I]<>'I' OR
                    NETSIGNAL[I+1].NAME<>NAME OR
                    NETSIGNAL[I-1].INOUT[I]='C' THEN
                BEGIN
                    WRITELN(CFILE,
                        NAME, ' ', CIRDES, '-', PINNUM, ' ', ICTYPE, ' ', PAGE, ERR4);
                    WRITELN(CFILE, blank2, err3)
                END
            ELSE
                WRITE(NFILE, NAME, ' ', CIRDES, '-', PINNUM)
            END;
        END;
    END;
END;

```

```

IF INOUT[I]= 'I' THEN
  BEGIN
    IF NETSIGNAL[I-1].NAME=NAME THEN
      BEGIN
        IF NETSIGNAL[I-1].INOUT[I]= 'O' THEN
          WRITELN(NFILE,
            ' ', CIRDDES, '-', PINNUM, ' ', ' ', PAGE, MESN);
          IF (NETSIGNAL[I-1].INOUT[I]= 'I' OR
            NETSIGNAL[I-1].INOUT[I]= 'C') THEN
            WRITELN(NFILE,
              CIRDDES, '-', PINNUM, ' ', ' ', PAGE, MESN)
          END
        ELSE
          WRITELN(CFILE,
            NAME, ' ', CIRDDES, '-', PINNUM, ' ', ICTYPE, ' ', PAGE, ERR2)
        END; (* INOUT=I *)
      END; (* SMALL ELSE , INOUT<> C *)
    END
  ELSE
    BEGIN
      WRITELN(CFILE, name, ' ', cirdes, '-', pinnum, ' ', ictype,
        ' ', page, err1); WRITE(CFILE, blank2);
      CASE INOUT[I] OF
        'I': WRITELN(CFILE, err2); 'O', 'C': WRITELN(CFILE, err3);
      END; (* CASE *)
    END; (* LARGEST ELSE *)
  END; (* WITH *)
END; (* SIGCOUNTER *)
END; (* PROC WRITEFILES *)

PROCEDURE Materiallist;
(* Write the list of material *)
VAR I, J, CHIPCOUNT, UNIT : INTEGER;
    TCOST, TAREA, TPWER, UCOST, UAREA, UPWER, SCOST, SAREA, SPWER: REAL;
    CHIPLIST : ARRAY[1..MAXNOCHIP+1] OF STRING4;

PROCEDURE Writeline( LINE: HALF_LINE );
VAR count : integer;
BEGIN (* Writeline *)
  count := 0;
  REPEAT
    count := count+1; WRITE(LFILE, LINE[count]);
  UNTIL (LINE[count]= ' ' AND LINE[count+1]= ' ') OR (count=HALF);
  WRITE( LFILE, ' ');
END; (* Writeline *)

BEGIN
  FOR I:=1 TO SIGCOUNT+1 DO
    BEGIN
      WITH NETSIGNAL[I] DO
        BEGIN
          IF I=1 THEN
            BEGIN
              J:=1; UNIT:=1; CHIPCOUNT:=0; CHIPLIST[UNIT]:=CIRDDES;
              KEY$FILE(K_READ_BY_KEY, ICTYPE, INICPART, K_SBK);
              WRITELN(SFILE, ICTYPE, ' ', CIRDDES, '-', PINNUM, ' ', INOUT, ' ',
                NAME, ' ', PAGE, ' ', INICPART, REFER);
              CHARREAL(INICPART, AREA, UAREA); SAREA:=UAREA;
              CHARREAL(INICPART, COST, UCOST); SCOST:=UCOST;
              CHARREAL(INICPART, TYPCUR, UPWER); SPWER:=UPWER;
              TAREA:=UAREA; TPWER:=UPWER;
              TCOST:=UCOST
            END
          ELSE
            BEGIN
              IF CIRDDES=NETSIGNAL[I-1].CIRDDES THEN
                WRITELN(SFILE, ' ', PINNUM, ' ', INOUT, ' ',
                  NAME, ' ', PAGE)
              ELSE
                BEGIN
                  CHIPCOUNT:=CHIPCOUNT+1;
                  IF ICTYPE=NETSIGNAL[I-1].ICTYPE THEN
                    BEGIN
                      WRITELN(SFILE, ICTYPE, ' ', CIRDDES, '-', PINNUM, ' ',
                        INOUT, ' ', NAME, ' ', PAGE, ' ', INICPART, REFER);
                      UNIT:=UNIT+1; CHIPLIST[UNIT]:=CIRDDES;
                      SAREA:=SAREA+UAREA; TAREA:=TAREA+UAREA;
                      SCOST:=SCOST+UCOST; TCOST:=TCOST+UCOST;
                      SPWER:=SPWER+UPWER; TPWER:=TPWER+UPWER
                    END
                  END
                END
            END
          END
        END
      END
    END
  END

```

```

END
ELSE
BEGIN
  WRITELN(LFILE, J:2, ' ', NETSIGNAL[I-1], ICTYPE, UAREA:7:3,
    UCOST:7:3, UPWER:7:3, UNIT:3, ' ', SAREA:7:3,
    SCOST:7:3, SPWER:7:3);
  WRITE( LFILE, '---- '); WRITELINE( INICPART.REFER );
  FOR K:=1 TO UNIT DO
    WRITE( LFILE, CHIPLIST[K], ' ');
  WRITELN( LFILE, ' ');
  UNIT:=1; J:=J+1; CHIPLIST[UNIT]:=CIRDES;
  IF I<>SIGCOUNT+1 THEN
    BEGIN
      KEY$FILE(K_READ_BY_KEY, ICTYPE, INICPART, K_SBK);
      WRITELN(SFILE, ICTYPE, ' ', CIRDES, ' ', PINNUM, ' ',
        INOUT, ' ', NAME, ' ', PAGE, ' ', INICPART.REFER);
      CHARREAL(INICPART, AREA, UAREA); SAREA:=UAREA;
      CHARREAL(INICPART, COST, UCOST); SCOST:=UCOST;
      CHARREAL(INICPART, TYPCUR, UPWER); SPWER:=UPWER;
      TAREA := TAREA+UAREA;
      TCOST := TCOST+UCOST;
      TPWER := TPWER+UPWER;
    END;
    WRITELN(LFILE, ' ');
  END;
END;
END; (* I>1 *)
END; (* WITH *)
END;
WRITELN(LFILE, dottedline);
WRITELN(LFILE, 'Total number of chips : ', CHIPCOUNT:3);
WRITELN(LFILE, 'Total area occupied : ', TAREA:7:3, ' Sq. inch');
WRITELN(LFILE, 'Total cost estimated : ', TCOST:7:3, ' Dollars');
WRITELN(LFILE, 'Total power dissipation : ', TPWER:7:3, ' mW');
END; (* Materiallist *)

PROCEDURE Closefiles;
CONST note1 = '----- NOTE -----';
note2 = '[!] : OK , NOT AN ERROR';
note3 = '[?] : ARE YOU SURE? , A POSSIBLE ERROR';
note4 = '[x] : FATAL ERROR';
BEGIN
  WRITELN(NFILE, ' '); WRITELN(NFILE, dottedline);
  WRITELN(NFILE, note1); WRITELN(NFILE, note2);
  WRITELN(NFILE, note3); WRITELN(NFILE, note4);
  CLOSE(NFILE); WRITELN(LFILE, dottedline);
  WRITELN(CFILE, ' '); WRITELN(CFILE, dottedline);
  WRITELN(CFILE, note1); WRITELN(CFILE, note2);
  WRITELN(CFILE, note3); WRITELN(CFILE, note4);
  CLOSE(CFILE); WRITELN(SFILE, dottedline);
  CLOSE(LFILE); CLOSE(SFILE);
  KEY$FILE(K_CLOSE, KEY, INICPART, K_SBK);
  KEY$FILE(K_CLOSE, KEY, OUTICPART, OK_SBK);
  KEY$FILE(K_TERM, KEY, INICPART, K_SBK);
END; (* PROC CLOSEFILES *)

BEGIN (* MAIN PROGRAM *)
  VDT;
  Openfiles;
  Readnet(sigcount);
  Quicksort(1, sigcount);
  Writefiles;
  Bubblesort(1, sigcount);
  Materiallist;
  Closefiles;
END.

```

VITA

Youngsup Kim was born on June 15, 1954, in Seoul, Korea. Following years of elementary and high school in that city, he entered the Yon Sei University in 1974, where he received the 1977 Outstanding Scholarship Award, and graduated with a Bachelor of Science degree in Electronics Engineering in February 1980. During that period Mr. Kim served in the military services for two years and was a summer employee of AGIE (AG Industrial Electronics Ltd.) located at Locarno, Switzerland. He also completed a year's study at the Yon Sei Graduate School of Engineering and worked as a research assistant for Korea Telecommunication Research Institute.

In June 1981 the author entered the Graduate School of Electrical Engineering at the University of Tennessee, Knoxville, where he was a teaching assistant on Plasma Science, Logic Circuit Design, and Interactive Computer Graphics. Mr. Kim received the Master of Science in Electrical Engineering in December 1983. He will continue his graduate study toward a Ph.D. degree.

Mr. Kim has been married to Higyung Joa since 1981 and has a daughter, Goeun.