



University  
of Glasgow

<https://theses.gla.ac.uk/>

Theses Digitisation:

<https://www.gla.ac.uk/myglasgow/research/enlighten/theses/digitisation/>

This is a digitised version of the original print thesis.

Copyright and moral rights for this work are retained by the author

A copy can be downloaded for personal non-commercial research or study, without prior permission or charge

This work cannot be reproduced or quoted extensively from without first obtaining permission in writing from the author

The content must not be changed in any way or sold commercially in any format or medium without the formal permission of the author

When referring to this work, full bibliographic details including the author, title, awarding institution and date of the thesis must be given

Enlighten: Theses

<https://theses.gla.ac.uk/>  
[research-enlighten@glasgow.ac.uk](mailto:research-enlighten@glasgow.ac.uk)

**AN INTEGRATED ENVIRONMENT FOR  
ELECTRO-MECHANICAL SYSTEMS  
DESIGN AND ANALYSIS**

**by**

**Azeddin Mohamed Shaeb Margani (B.Eng., MSc)**

**Thesis presented for the  
Degree of Doctor of Philosophy  
under general regulations**

**Department of Mechanical Engineering  
Faculty of Engineering  
University of Glasgow**

**© A. M. S. Margani, 1997**

**September, 1997**



ProQuest Number: 10391393

All rights reserved

INFORMATION TO ALL USERS

The quality of this reproduction is dependent upon the quality of the copy submitted.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if material had to be removed, a note will indicate the deletion.



ProQuest 10391393

Published by ProQuest LLC (2017). Copyright of the Dissertation is held by the Author.

All rights reserved.

This work is protected against unauthorized copying under Title 17, United States Code  
Microform Edition © ProQuest LLC.

ProQuest LLC.  
789 East Eisenhower Parkway  
P.O. Box 1346  
Ann Arbor, MI 48106 – 1346

*Thesis 10957  
copy 2*



# Synopsis



The purpose of this research was to examine current methodology and design tools for concurrent engineering of complex systems. A prime objective of the work was to develop an integrated CAE system for concurrent engineering and to investigate performance issues, including: Design management, productivity, modelling and simulation. The main aspects that are important in achieving total design integration have been considered and extensively studied, including: System functionality requirements, design flow, system integration, applications design, and data management. An integrated design environment has been developed utilising open architecture, commercial CAD systems and databases. An extensive range of design tools have been developed which facilitate the process of design and design management.

Issues relating to transfer of data across domains (or disciplines) are addressed specifically. A "netlist" generating systems has been designed to facilitate data transfer between domains and from design to simulation. Several techniques have been assessed and fully functional system developed utilising database extraction.

A database was used to assist in many areas, in particular, the strong need for efficient design data management. Various tools have been designed and developed to ensure that integration between the different environments is achieved and also to accommodate the needs of profile interchange between tools. The development of these tools comprises three major parts. In part 1, methods of design reports generation are described. Part 2 concerned with the design and development of data update tools. Part 3 is concerned with data manipulation. Design examples are used to demonstrate the performance of these tools.

## *Synopsis*

The work on implementing the graphical user interface (GUI) has produced a large variety of design tools and utilities. This work describes the design and development of these tools and their applications to perform design, modelling and simulation.

Evaluation and performance of the software have been demonstrated based on a coupled example of electro-mechanical design. A set of design issues involving graphics constructions of both 2D schematics and 3D modelling, design tool capabilities, and design simulation was considered and performed.

The results of this work have produced many specific design tools. In a broader sense, the research has examined key issues of design management in concurrent engineering and drawn important conclusions.

# Acknowledgment

---

I would like to express my gratitude to Dr. D. McDonald for his encouragement and guidance during this research, and for his useful discussions and ideas.

Particular thanks are due to the technicians: Gordon Hicks and Bernard Hoey for their assistance and useful suggestions. I would also like to thank Ian Peden for the invaluable assistance with computer hardware.

Finally, I would like to thank my family for their encouragement and continuous support during my study.

# Contents

---

<b>SYNOPSIS</b>	ii
<b>ACKNOWLEDGEMENT</b>	iv
<b>LIST OF FIGURES</b>	xi
<b>LIST OF TABLES</b>	xx
<b>NOMENCLATURE</b>	xxii
 <b>CHAPTER 1. INTRODUCTION.</b>	
<b>1.1 BACKGROUND.</b>	1
<b>1.2 DESIGN PRACTICE.</b>	1
<b>1.3 DESIGN SYSTEMS.</b>	5
<b>1.4 MARKET FOR MECHANICAL DESIGN SOFTWARE.</b>	6
<b>1.5 DESIGN SYSTEMS AND METHODOLOGY.</b>	8
1.5.1 Product Design Approach	8
<b>1.6 DETAILED DESIGN PROCESSES.</b>	10
1.6.1 Electronics Schematics and PCB Layout	10
1.6.2 Circuit Simulation and Analysis	11
1.6.3 Thermal Analysis	13
1.6.4 Mechanical Drafting and Solid Modelling	14
1.6.5 Mechanical Dynamics	14
1.6.6 Finite Element Analysis	15
<b>1.7 CONCLUSIONS.</b>	15
<b>1.8 SCOPE OF PRESENT WORK.</b>	16
1.8.1 Outline of Thesis	16
<b>REFERENCES.</b>	18
 <b>CHAPTER 2. SYSTEM DESIGN AND DEVELOPMENT.</b>	
<b>2.1 INTRODUCTION.</b>	21
<b>2.2 DESIGN CONSIDERATIONS.</b>	24
<b>2.3 SYSTEM DESIGN ARCHITECTURE.</b>	25
2.3.1 Graphical User Interface (GUI)	25
2.3.2 Cells Libraries	27

2.3.3 Design Activities	27
<b>2.4 SYSTEM IMPLEMENTATION.</b>	28
2.4.1 Hardware Considerations	29
2.4.2 System Design Software	29
2.4.3 System Functionality	32
2.4.4 System Integration	33
2.4.5 Design Flow	34
2.4.6 Design Data Management	34
<b>2.5 SPECIFIC SOFTWARE REQUIREMENTS.</b>	19
2.5.1 Physical Design Aids	40
2.5.2 Design Connectivity and Testability	40
2.5.3 Integration With Standard Database	41
2.5.4 Circuit Documentation and Data Extraction	42
<b>2.6 CONCLUSION.</b>	56
<b>REFERENCES.</b>	56
 <b>CHAPTER 3. COMPONENT DESIGN AND MODELLING.</b>	 59
<b>3.1 INTRODUCTION.</b>	59
<b>3.2 COMPONENT MODELLING.</b>	60
3.2.1 Modelling Objective	60
3.2.2 Fundamental Modelling Concepts	
3.2.3 PSPICE and Parts Models	60
3.2.4 Circuit Modelling Languages	61
<b>3.3 ELECTRO-MECHANICAL DESIGN.</b>	63
3.3.1 Electro-mechanical relational Design	64
3.3.2 Parametric Design	65
3.3.3 Parts and Attributes	66
<b>3.4 DESIGN AND MODELLING PROCEDURE.</b>	69
3.4.1 The Procedure Requirements	70
<b>3.5 COMPONENTS DESIGN ENVIRONMENT.</b>	71
3.5.1 Integration of Tool sets	71
3.5.2 Libraries Cells Design	72
3.5.3 Interaction with Database	72
<b>3.6 COMPONENT DESIGN EXAMPLES.</b>	73





3.6.1	Charging Inductor Design	73
3.6.2	Problem Definition	75
3.6.3	Theoretical Calculations	76
3.6.4	Design Optimisation	78
3.6.5	Software Program Development	80
3.6.6	Design of a High Frequency Transformer	84
3.6.7	Transformer Theory	85
3.6.8	Problem Definition	87
3.6.9	Transformer Theoretical Calculations	88
<b>3.7</b>	<b>DESIGN SIMULATION</b>	<b>94</b>
3.7.1	Simulation Results using PSPICE	94
3.7.2	Simulation Results using TK solver	96
<b>3.8</b>	<b>CONCLUSION.</b>	<b>99</b>
	<b>REFERENCES.</b>	<b>99</b>
<b>CHAPTER 4.</b>	<b>AUTOMATIC NETLIST GENERATION.</b>	<b>102</b>
<b>4.1</b>	<b>INTRODUCTION.</b>	<b>102</b>
<b>4.2</b>	<b>THE IMPORTANCE OF CIRCUIT NETLISTS.</b>	<b>103</b>
<b>4.3</b>	<b>RELATED WORK.</b>	<b>103</b>
4.3.1	The need for Computer Aided Circuit Analysis	107
4.3.2	PSPICE Background Information	108
4.3.3	Circuit Definition in PSPICE	108
4.3.4	Process for Using PSPICE in Electronic CAD	109
<b>4.4</b>	<b>NETLIST DESIGN DEVELOPMENT.</b>	<b>111</b>
4.4.1	Problem Definition	112
4.4.2	General Development Rules	113
4.4.3	Connectivity Establishment	114
<b>4.5</b>	<b>DEFERENT TECHNIQUES USED FOR NETLIST GENERATION.</b>	<b>115</b>
4.5.1	Matrix Comparison Technique	115
4.5.2	Circuit Data Extraction	118
4.5.3	Software Program Description	119
4.5.4	The use of Database	127
4.5.5	User Defined Netlist	128
<b>4.6</b>	<b>CONCLUSIONS.</b>	<b>134</b>
	<b>REFERENCES.</b>	<b>135</b>

<b>CHAPTER 5. DESIGN FOR DATABASE AUTO LINK.</b>	<b>137</b>
<b>5.1 INTRODUCTION.</b>	<b>137</b>
<b>5.2 A HISTORY AND EVALUATION OF DATABASE SYSTEMS.</b>	<b>139</b>
<b>5.3 THE ROLE OF DATABASE IN CAD.</b>	<b>139</b>
<b>5.4 INTERFACING WITH SOFTWARE ENGINEERING.</b>	<b>141</b>
<b>5.5 COMMERCIAL DATABASE SYSTEMS.</b>	<b>141</b>
<b>5.6 RELATIONAL DATABASE CONCEPTS.</b>	<b>142</b>
5.6.1 Data Relational Model	143
5.6.2 Relationships Determination	145
5.6.3 The use of Structural Query Language	145
<b>5.7 DESIGN DEVELOPMENT.</b>	<b>146</b>
5.7.1 Applications Design Methodology	149
5.7.2 Database Editing Interface	152
5.7.3 Setting up and Accessing the Database	152
5.7.4 Linking Circuit Elements to Database Records	153
5.7.5 Data Tables Design	153
5.7.6 Automatic Element Recognition	154
<b>5.8 DATA QUERIES AND REPORTS GENERATION.</b>	<b>154</b>
5.8.1 Design for Data Query	155
5.8.2 Design for Data retrieval	157
5.8.3 Design for Reports Generation	161
5.8.4 Design for Fence Reporting	162
<b>5.9 DESIGN FOR DATA UPDATE.</b>	<b>163</b>
5.9.1 Update by Fence Design	166
<b>5.10 DESIGN FOR DATA MANIPULATION.</b>	<b>167</b>
5.10.1 Data Deletion	167
5.10.2 Automatic Data Linkage Dropping	168
<b>5.11 CONCLUSION.</b>	<b>170</b>
<b>REFERENCES.</b>	<b>170</b>
 <b>CHAPTER 6. DESIGN OF GRAPHICS USER INTERFACE (GUI).</b>	 <b>173</b>
<b>6.1 INTRODUCTION.</b>	<b>173</b>
<b>6.2 REVIEW OF GRAPHICS SYSTEMS.</b>	<b>174</b>

<b>6.3 GRAPHICS DESIGN AND ENGINEERING CONSIDERATION.</b>	174
<b>6.4 GRAPHICS AND TOOL CUSTOMISATION.</b>	175
<b>6.5 GRAPHICS MODELLING METHODS.</b>	176
6.5.1 Graphic Model Representation	176
6.5.2 Symbol hierarchies	177
<b>6.6 GRAPHICS DESIGN AND DEVELOPMENT.</b>	177
6.6.1 Design Methodology	177
<b>6.7 DESIGN OF USER INTERFACE.</b>	178
6.7.1 Review of User Interface Generators	179
6.7.2 User Interface Design Levels	181
<b>6.8 ELECTRICAL SCHEMATIC DESIGN TOOLS</b>	184
6.8.1 Schematics Workspace Design	184
6.8.2 Editors Tools Design	185
6.8.2.1 Design of Drafting Tools	186
6.8.2.2 Design of Schematics Palettes	188
6.8.2.3 Design of Configuration Tools	202
6.8.3 Design By Levels	207
<b>6.9 DESIGN OF ASCII REPORTS TOOLS.</b>	209
<b>6.10 DESIGN OF TRANSFER TOOLS.</b>	210
6.10.1 Pull-Down Menus Environment Design	210
6.10.2 3D Environment Design	214
<b>6.11 DESIGN OF LIBRARIES TOOLS.</b>	215
6.11.1 Symbols Creation	218
<b>6.12 DESIGN OF DATABASE TOOLS SET.</b>	221
<b>6.13 DESIGN OF DATA TOOLS.</b>	229
<b>6.14 DESIGN OF EXIT TOOLS.</b>	229
<b>6.15 CONCLUSION.</b>	229
<b>REFERENCES.</b>	229
 <b>CHAPTER 7. SOFTWARE PERFORMANCE AND EVALUATION.</b>	 231
<b>7.1 INTRODUCTION.</b>	231
<b>7.2 PERFORMANCE EVALUATION.</b>	231

<b>7.3 A DESIGN EXAMPLE.</b>	232
7.3.1 Circuit General Description	232
7.3.2 2D Circuit Representation	233
<b>7.4 CIRCUIT DESIGN AND ANALYSIS.</b>	234
7.4.1 Design of First stage	234
7.4.2 Simulation of First Stage	238
7.4.3 Design Optimisation and Performance	241
7.4.4 Design of the Second Stage	241
7.4.5 Simulation of Second Stage	244
<b>7.5 MECHANICAL DESIGN.</b>	246
7.5.1 Components Design Description	246
7.5.2 3D Layout Representation	249
7.5.3 The Assembly Wiring Connections	251
<b>7.6 ENVIRONMENT INTEGRATION.</b>	254
7.6.1 The Interface between 2D and 3D layouts	255
7.6.2 Components Modelling Procedure	255
7.6.3 Integration with the Common Database	256
<b>7.7 DESIGN COFIGURATION.</b>	259
<b>7.8 DESIGN DOCUMENTATION.</b>	260
<b>7.9 CONCLUSIONS.</b>	261
<b>REFERENCES.</b>	262
 <b>CHAPTER 8. GENERAL CONCLUSIONS AND FUTURE WORK.</b>	 263
<b>8.1 GENERAL CONCLUSION.</b>	263
<b>8.2 FUTURE WORK.</b>	268

# List of Figures

---

## CHAPTER 1.

### Section 1.5

- 1.1 Product-design process supported by CAD applications. 9
- 1.2 Customer requirements and product characteristics matrix. 10

### Section 1.6

- 1.3 Flowchart of SPICE general analysis process. 11

## CHAPTER 2.

### Section 2.1

- 2.1 Overview of system design life cycle. 22
- 2.2 Breakdown for electronics products systems. 23

### Section 2.2

- 2.3 The system components block diagram. 25

### Section 2.3

- 2.4 Illustration of the design detailed structure. 26

### Section 2.4

- 2.5 Applications development procedure. 30
- 2.6 Programming of MDL and UCMs as separate applications. 31
- 2.7 MDL and UCMs combined as single application. 32
- 2.8 Concept of system functionality levels. 33
- 2.9 The system integrated environments. 34
- 2.10 Design flow proceeds towards complete integration. 35
- 2.11 Design flow and supporting tools. 36
- 2.12 Extracted data and files extensions. 38
- 2.13 Overall architecture of data management design. 39

### Section 2.5

- 2.14 Design of circuit design tools. 40
- 2.15 Flowchart of circuit connectivity program. 46
- 2.16 Circuit information flow and database linkage. 42
- 2.17 Flow of information in an automated document generator. 43
- 2.18 Circuit automation tools and data extraction. 44
- 2.19 Flowchart of circuit Bill Of Material extraction program. 47
- 2.20 Flowchart of wiring list extraction program. 49

## *List of Figures*

2.21a Flowchart of drawing information extraction program part (1).	50
2.21b Flowchart of drawing information extraction program part (2).	51
2.22 Flowchart of automatic wires identity placement program.	53
2.23 Flowchart of automatic junction placement program.	54
2.24 Flowchart of automatic junction numbering program.	55

### **CHAPTER 3.**

#### **Section 3.2**

3.1 The scope of VHDL compared to some other modelling techniques.	62
--	----

#### **Section 3.3**

3.2 Parametric design supports different applications.	65
3.3 Part-attributes relationship translation.	67
3.4 The concept of mechanical part-attributes definition.	67
3.5 Inductor part-attributes description.	68
3.6 Transformer part-attributes description.	69

#### **Section 3.4**

3.7 Design and modelling general concept.	70
---	----

#### **Section 3.5**

3.8 Illustration of the integrated environments and their tools.	72
3.9 Block diagram illustrates the use of database.	73

#### **Section 3.6**

3.10 Inductor detailed description.	74
3.11 Charging inductor design circuit.	75
3.12 The equivalent design model of $L_c$	75
3.13 Inductor with detailed dimensions.	76
3.14 Wave forms of charging current and voltage.	79
3.15 Flowchart of inductor design program.	81
3.16 Inductor parameters calculations part.	81
3.17 Inductor design template.	83
3.18 Database retrieved information using SQL.	83
3.19 The inductor extracted model.	83
3.20 The usage of transformer.	84
3.21 Transformer detailed description.	85
3.22 Voltage square wave excitation.	86
3.23 Half bridge forward converter design circuit.	87
3.24 Transformer with detailed dimensions.	87

### **Section 3.7**

3.25	Flowchart of transformer design program.	93
3.26	Transformer parameters calculations part.	94
3.27	Display of PFL circuit and PSPICE.	95
3.28a	Transient analysis response of $V(0)$ when $V_{DC}=3.5V$ .	95
3.28b	Transient analysis response of $V(0)$ when $V_{DC}=1.75kV$ .	95
3.29a	Peak current response when $L_c=9H$ .	96
3.29b	Peak current response when $L_c=0.9H$ .	96
3.30	A display of TK solver interface.	96
3.31	Voltage versus maximum charging time.	97
3.32	Peak current versus maximum charging time.	97
3.33	Air gap required versus number of turns.	97
3.34	Flux density versus peak current.	98
3.35	Flux density versus air gap required.	98

## **CHAPTER 4.**

### **Section 4.3**

4.1	Design tree of a full adder.	104
4.2	3-steps process of netlist creation using OrCAD system.	105
4.3	Files generated by both compiler and linker.	106
4.4	Final stage of netlist format files (flat, hierarchical netlist).	106
4.5	Creating a netlist in sabre system.	107
4.6	Generic PSPICE input file format.	109
4.7	Process flowchart of using PSPICE.	110
4.8	The link between system environments and PSPICE.	111

### **Section 4.4**

4.9	Illustration of circuit connection possibilities for netlist generation.	114
4.10	A circuit as connected components.	114

### **Section 4.5**

4.11	Components matrix form of four co-ordinates.	116
4.12	Components matrix form of six co-ordinates.	116
4.13	Components matrix form of eight co-ordinates.	117
4.14	Wires matrix form.	117
4.15	Matrices comparison (components versus wires).	118
4.16	Resultant node connection matrix.	118
4.17	General procedure for data extraction.	119
4.18	Flowchart of circuit input data extraction program.	120
4.19	A circuit and its extracted data.	121

## *List of Figures*

4.20	Wires, components, and components co-ordinates/nodes matrices forms.	122
4.21	Illustration of the basic structure of one way linked list.	123
4.22	Flowchart of read data file program.	124
4.23	Flowchart of node determination program.	125
4.24	Procedure of data processing.	126
4.25	MDL debugger during data separation.	126
4.26	The use of database dbase(iv).	127
4.27	User defined netlist using dialog editor.	128
4.28	Flowchart of dialog data editor program.	129
4.29	Flowchart of netlist final format extraction.	130
4.30	Example of generating netlist using dialog editor.	131
4.31	Generated netlist file.	132
4.32	Circuit Bill Of Material extraction.	132
4.33	Circuit wiring list extraction.	133
4.34	Circuit drawing information extraction.	133
4.35	Automatic nodes-wires identity placement.	134

## **CHAPTER 5.**

### **Section 5.1**

5.1	Engineering database (towards integration-engineering database).	138
-----	--	-----

### **Section 5.5**

5.2	Database-centred and executive systems.	142
5.3	Database-centred and executive systems.	142

### **Section 5.6**

5.4	Illustration of entities-relationship conceptual model.	144
5.5	Organisation of database.	146

### **Section 5.7**

5.6	Database dbase(iv) used for different design applications.	147
5.7	Illustration of the link between MicroStation and dbase(iv) environment.	148
5.8	Illustration of database tools design plan.	149
5.9	Applications design organisation.	151
5.10	Tables are shared between applications.	151
5.11	General architecture of database communication link.	152
5.12	Element to record linkage.	153
5.13	Flowchart of element recognition for data editing program.	154

### **Section 5.8**

5.14	Illustration of data query design.	155
------	------------------------------------	-----



## *List of Figures*

5.15	Flowchart of query extraction program.	156
5.16	Different user dialog boxes used for design data query.	157
5.17	Demonstration of components-netlist queries.	157
5.18	Illustration of data retrieval design.	158
5.19	Flowchart of element recognition for data retrieval program.	159
5.20	Demonstration of design data retrieval.	160
5.21	Illustration of reports generation design.	161
5.22	Illustration of fence reporting design.	162
5.23	Dialog boxes used for fence report generation.	163

### **Section 5.9**

5.24	Illustration of data update design.	164
5.25	Flowchart of data update program.	165
5.26	Demonstration of design data update.	166
5.27	Illustration of fence updating design.	166
5.28	Demonstration of fence update design.	167

### **Section 5.10**

5.29	Illustration of delete data design.	168
5.30	Dialog for data deletion.	168
5.31	Flowchart of linkage dropping program.	169

## **CHAPTER 6.**

### **Section 6.6**

6.1	Illustration of graphics design general concept.	178
-----	--	-----

### **Section 6.7**

6.2	Common industrial development process flowchart.	179
6.3	Different types of user interface generators.	180
6.4	Illustration of GUI design levels.	182
6.5	User interface main dialog.	183
6.6	2D schematics interface main dialog.	183

### **Section 6.8**

6.7	Display of schematics workspace environment.	185
6.8a	Schematics design tools part (1).	187
6.8b	Schematic design tools part (2).	188
6.9	Place main palette.	189
6.10	Schematic junctions sub-palette.	190
6.11	Place wires sub-palette.	190

## *List of Figures*

6.12	Bus symbols sub-palette.	190
6.13	Ground symbols sub-palette.	190
6.14	Power supply symbols sub-palette.	190
6.15	Schematic place text sub-palette.	191
6.16	Module ports symbols sub-palette.	191
6.17	Amplifiers symbols sub-palette.	191
6.18	Batteries symbols sub-palette.	191
6.19	Transformers symbols sub-palette.	191
6.20	Capacitors symbols sub-palette.	192
6.21	Diodes symbols sub-palette.	192
6.22	Inductors symbols sub-palette.	192
6.23	Resistors symbols sub-palette.	192
6.24	Switches symbols sub-palette.	193
6.25	Voltage supply symbols sub-palette.	193
6.26	Transistors symbols sub-palette.	193
6.27	IC symbols sub-palette.	193
6.28	Rectifiers symbols sub-palette.	194
6.29	Thermal elements symbols sub-palette.	194
6.30	Thermocouple symbols sub-palette.	194
6.31	Lamps and visual devices symbols sub-palette.	194
6.32	Discontinuity symbols sub-palette.	195
6.33	Coils symbols sub-palette.	195
6.34	Place text dialog.	195
6.35	Modify elements main palette.	196
6.36	Angled line sub-palette.	197
6.37	Copy line parallel sub-palette.	197
6.38	Group-ungroup elements sub-palette.	197
6.39	Place fillet sub-palette.	197
6.40	Change colour dialog.	198
6.41	Change style dialog.	198
6.42	Change weight dialog.	198
6.43	Workspace top main palette.	199
6.44	Netlist-wire sub-palette.	199
6.45	Open view main dialog.	200
6.46	3D modelling main palette.	200
6.47	Components UCMs program attachment.	201
6.48	Flowchart of component placement program.	201
6.49a	Configuration tools hierarchy part (1).	203
6.49b	Configuration tools hierarchy part (2).	203
6.49c	Configuration tools hierarchy part (3).	204

## *List of Figures*

6.50	Settings tools dialog of sheet (1).	203
6.51	Automatic colours configuration dialog part (1).	203
6.52	Automatic colours configuration dialog part (2).	204
6.53	Automatic style configuration dialog.	204
6.54	Automatic font-weight configuration dialog.	205
6.55	Schematics configuration tools sheet (1).	206
6.56	Sheet advance group.	205
6.57	Schematics configuration tools sheet (2).	206
6.58	Schematics configuration tools sheet (3).	207
6.59	Toggle settings dialog.	207
6.60	The concept of design by levels.	208
6.61	Schematics levels control dialog.	208
 <b>Section 6.9</b>		
6.62	ASCII reports dialog.	209
 <b>Section 6.10</b>		
6.63	Pull-down menus tools design hierarchy.	211
6.64	Pull-down menus environment.	211
6.65	File tools submenus.	212
6.66	Settings tools submenus.	212
6.67	Palettes submenus.	212
6.68	Database submenus.	213
6.69	ASCII reports submenus.	213
6.70	3D environment design hierarchy.	214
 <b>Section 6.11</b>		
6.71	The 3D modelling environment.	215
6.72	Libraries tools edit environment.	216
6.73	Library design tools (Edit submenus).	217
6.74	Library design tools (graphics submenus).	217
6.75	Library design tools (part definition submenus).	217
6.76	Components libraries classification.	218
6.77	An example of transformer symbol design.	218
6.78	Palette of components different pins.	219
6.79	Batteries-capacitors symbols.	219
6.80	Thermal elements-transistors symbols.	219
6.81	Inductors-transformers symbols.	220
6.82	Schematic objects-switches symbols.	220
6.83	Module ports-pins symbols.	220
6.84	Diodes-resistors symbols.	221

**Section 6.12**

6.85	Database tools dialog.	221
6.86	Database query main dialog.	222
6.87	Components query main dialog.	222
6.88	Components mslink query dialog.	222
6.89	Components table query dialog.	223
6.90	Components IDs query dialog.	223
6.91	Wires query main dialog.	223
6.92	Wires mslink query dialog.	223
6.93	Wires table query dialog.	224
6.94	Wires IDs query dialog.	224
6.95	Netlist query main dialog.	224
6.96	Netlist mslink query dialog.	224
6.97	Netlist tables query dialog.	224
6.98	Netlist reference query dialog.	225
6.99	Reports creation main dialog.	225
6.100	Data retrieval main dialog.	225
6.101	Fence report generation dialog.	226
6.102	Wires data editing dialog.	226
6.103	Netlist editor dialog.	226
6.104	Components data editing dialog.	227
6.105	Components data update dialog.	227
6.106	Clear database records main dialog.	227
6.107	Clear database main dialog.	228
6.108	Inductor modelling template.	228
6.109	Transformer modelling template.	228

**CHAPTER 7.**

**Section 7.3**

7.1	Resonant charging circuit drawing on 2D schematic environment.	223
7.2	2D schematic different views.	234
7.3	The 2D schematic of charging circuit.	234

**Section 7.4**

7.4	Schematic design of the first stage.	235
7.5	The equivalent design model of L1.	235
7.6	Netlist file generated for the first stage.	239
7.7	A display of the 2D schematic environment and PSPICE.	239
7.8a	Peak current $\hat{I}$ through swt1.	240
7.8b	Peak current $\hat{I}$ through L1.	240

## *List of Figures*

7.9	Output voltage $V(0)$ through inductor L1.	240
7.10	Energy stored in L1.	241
7.11	Schematic design of the second stage.	241
7.12	The equivalent design model of L3.	242
7.13	Graph of inductance versus former diameter and number of turns.	244
7.14	Netlist generated for the second stage.	244
7.15	Peak current $\hat{I}$ when $L3=10\mu H$ .	245
7.16	Output voltage $V(0)$ when $L3=10\mu H$ .	245
7.17	Peak current $\hat{I}$ when $L3=1\mu H$ .	245
7.18	Output voltage $V(0)$ when $L3=1\mu H$ .	245
7.19	Energy stored in L3 when $L3=10\mu H$ and $1\mu H$ .	246

### **Section 7.5**

7.20	Inductor L1 design.	247
7.21	Spiral inductors (L1, L3) design.	247
7.22	Capacitor (C1) design.	248
7.23	Thyratron CX1836 design.	248
7.24	Thyratron CX1747 design.	249
7.25	3D model components stored as libraries cells.	249
7.26	3D different views representation.	250
7.27	Isometric representation of the 3D model.	250

### **Section 7.6**

7.28	Illustration of environments integration.	254
7.29	The interface between 2D and 3D using common database.	255
7.30	Dialog editor and L1 extracted model.	256
7.31	Components data editing and data retrieval.	257
7.32	Components data updating.	258
7.33	Display of inductor database storage.	258
7.34	Display of first stage netlist stored in database.	259

### **Section 7.7**

7.35	Display of graphics configuration tools for (2D and 3D).	259
7.36	Display of settings and levels control tools.	260

### **Section 7.8**

3.37	Display of Documents generation tools.	260
7.38	Circuit B.O.M. and wiring list extraction.	260

# List of Tables

---

## CHAPTER 1.

### Section 1.2

1.1	Product design and its techniques.	3
-----	------------------------------------	---

### Section 1.3

1.2	List of ECAD systems and tools integration approaches.	6
-----	--	---

### Section 1.6

1.3	List of existing programs for circuit simulation.	13
-----	---	----

## CHAPTER 2.

### Section 2.5

2.1	Connectivity testing utilities and its development functions.	48
2.2	Bill Of Material utilities and its development functions.	48
2.3	Wiring list utilities and its development functions.	52
2.4	Drawing information utilities and its development functions.	52

## CHAPTER 3.

### Section 3.6

3.1	Circuit design specification.	75
3.2	Component output parameters.	78
3.3	The database structure for inductor design.	82
3.4	Transformer input data.	87
3.5	Transformer output data.	92

## CHAPTER 4.

### Section 4.5

4.1	Summary of utilities, functions and their operations.	121
4.2	Summary of database extraction utilities, functions and their operations.	131

## CHAPTER 5.

### Section 5.3

5.1	Database comparison.	140
-----	----------------------	-----

**Section 5.6**

5.2 A summary of SQL statements.	146
----------------------------------	-----

**CHAPTER 6.**

**Section 6.9**

6.1 Workspace initial settings commands and their operations.	202
6.2 Schematics level assignment.	209

**Section 6.10**

6.3 Commands and their functions keys.	213
--	-----

**CHAPTER 7.**

**Section 7.3**

7.1 Circuit general parameters.	232
---------------------------------	-----

**Section 7.4**

7.2 First stage input parameters.	238
7.3 Second stage input parameters.	242
7.4 Second stage design parameters.	243

**List of photographs**

**CHAPTER 7.**

**Section 7.5**

7.1 Laser charging pulser.	251
7.2 Laser charging pulser (side view).	252
7.3 Laser charging pulser (isometric view).	253

# Nomenclature

Symbol	Quantity	Units
$A$	Area	$m^2$
$A_c$	Effective iron area	$cm^2$
$A_p$	Area product	$cm^4$
$B_m$	Magnetic flux density	$T$
$B$	Flux density	$T$
$C$	Capacitor	$F$
$C_1$	Intermediate capacitor	$nF$
$C_s$	Smoothing capacitance	$F$
$C_n$	Network capacitor	$nF$
$D$	Diode	
$c$	Thread length	$m$
$d$	Mean diameter	$m$
$d_o$	Total length	$m$
$d_i$	Winding length	$m$
$E$	Magnetic energy	$J$
$F$	Frequency	$Hz$
$G$	The length of the core limb	$mm$
$\hat{I}$	Peak charging current	$A$
$\hat{I}_C$	Normal peak current	$A$
$\hat{I}_F$	Peak fault current	$A$
$\bar{I}_t$	Mean current	$A$
$I_s$	Secondary current	$A$
$I_p$	Primary current	$A$
$K_f$	Wave form coefficient	<i>Square</i>
$K_j$	Current density coefficient	
$K_u$	Window utilisation factor	<i>Ratio</i>
$L$	Inductor	$H$
$L_1$	Charging inductor	$\mu H$
$L_2$	Inductor	$\mu H$
$L_3$	Inductor	$\mu H$
$L_c$	Charging inductor	$H$
$l_a$	Air gap length	$mm$
$l_p$	Primary linear dimension	$m$
$l_m$	Core magnetic length	$mm$
$l_s$	Secondary linear dimension	$m$
$MTL$	Mean length turn	$cm$



## Nomenclature

$N$	Number of turns	
$N_s$	Secondary turns	
$N_p$	Primary turns	
$P_o$	Transformer output power	$W$
$P_t$	Apparent power	$W$
$P_{fe}$	Core loss	$W$
$P_g$	Air gap loss	$W$
$P_p$	Primary loss	$W$
$P_s$	Secondary loss	$W$
$P_{cu}$	Total copper loss	$W$
$R_p$	Primary resistance	$\Omega$
$R_s$	Secondary resistance	$\Omega$
$V_{NL}$	Voltage with no load	$V$
$V_{FL}$	Voltage with full load	$V$
$V_r$	Ripple voltage	$V$
$V_{DC}$	DC voltage	$V$
$V_p$	Primary voltage	$V$
$V_s$	Supply voltage	$V$
$\hat{V}_p$	Peak primary voltage	$V$
$Wf_e$	Core weight	$g$
$W_a$	Window area	$m^2$

## Greek letters

$\tau_{max}$	Maximum charging time	$\mu s$
$\tau_c$	Charging duration cycle	$\mu s$
$\mu_0$	Absolute permeability	
$\mu_\Delta$	Incremental permeability	
$\alpha$	Regulation	% Reg.
$\eta$	Efficiency	
$\phi_{max}$	Maximum magnetic flux	
$\phi$	Magnetic flux	
$\xi$	Electro-magnetic field	

# 1 Introduction



## 1.1 BACKGROUND

Design of all but the simplest products now generally involves multi-disciplinary teams of engineers working in close tandem. For example, the design of a medical diagnostic product may involve: biochemists, optics designers, analogue designers, mechanical engineers, digital designers, software engineers and production engineers. Obviously, it is critical on such projects to manage design information during both the concept phase and detailed design phase. The objectives of each project is primarily the creation of a competitive product or system and communication is arguably the most critical factor governing success of any project.

In the context of design in general, the purpose of this work is to examine current methodology for "concurrent engineering" and to develop a user-friendly environment for engineering multi-disciplinary products and systems. Before going on to detail the full scope of this work, it is worthwhile considering current design practice.

## 1.2 DESIGN PRACTICE

During the last decade major advances have taken place in computer-aided design (CAD) systems for assisting the rapid and efficient design of electronic circuits. This has been made possible by the advent of low-cost, powerful and easily accessible microcomputers. The market today for printed circuit board (PCB) design is very competitive with many suppliers offering fully integrated systems at comparatively low cost. Common tools provided include: Digital and Mixed Signal analysis, logic synthesis, Programmable

Array Logic, thermal analysis and latterly electro-magnetic compatibility (EMC) analysis. Tools for the implementation of application specific integrated circuits (ASICs) are now readily available and offer a custom alternative to discrete designs. The market for products specifically for PCB design and layout is now quite mature, with new tools and features evolving at a much slower pace than during the eighties.

There is a wide spectrum of systems's design which involves more detailed engineering analysis than simple schematic design and PCB layout, which may span a range of disciplines. For example, the design of switched-mode power supplies involves not only the design of drive and conditioning circuits, but also a very iterative design cycle for magnetic components, which is closely coupled to mechanical layout of cores and windings. It is therefore necessary for the designer to iterate between mechanical and electronics designs in order to optimise circuit performance. There are many examples of this type of design particularly in power electronics. Although computer-aided engineering (CAE) systems are available for each distinct aspect of the design (e.g. mechanical CAD, electronics, control simulation) a fully integrated environment for system engineering is not available commercially; at best each organisation must develop their own integrated, bespoke, system from available software products.

In the mechanical domain, CAD systems have developed along two paths: 2D draughting and 3D solid modelling. In 2D draughting, tools have evolved which automate the draughting process with standard libraries of components (e.g. fasteners). However, it is still reliant on the user to generate the engineering views of the components. Many vendors have added on 3D modelling with some solid modelling features to enhance the product but their main application is still primarily in 2D draughting. On the other hand, advanced modelling packages have been developed using solid modelling techniques and surface modelling to provide true 3D mathematical representations of components. Coupled with parameteric dimension driven design capability, such tools provide on unparalleled approach for mechanical design. 2D drawings can be generated automatically from the 3D models and models transferred to rapid prototyping systems such as stereolithography. Such systems are generally UNIX based and systems are currently evolving rapidly. Advanced PC-based systems (Windows NT ) are now appearing, which is predicted to dominate the market.

It is apparent that between the extremes of electronics circuit tools and mechanical modelling that there is an opportunity to provide an integrated environment for design, aimed at the problems which fall into both electronic and mechanical domains, and wider disciplinary projects. Power electronics is one area but there are many more, including: mechatronics, building services and electro-magnetics. What is required is a design

environment that combines the best of both worlds so that the designers can transfer between electronics and mechanical layouts (or domains) rather like a PCB designer transfers between schematics and layout. Even within PCB design this may of significant advantage given the requirements of packaging; the display of full 3D circuit models would give the designer a greater appreciation of layout, particularly when mechanical component such as heat sinks and connectors need to be added, and PCB's interfaced in complex assemblies.

The growing market for design software tools indicates an increase in the use of computers throughout the product design process [1.1]. In addition, there is a demand for full product models (digital mock-ups) and links to simulators. These advances contribute to a more efficient design process and facilitate innovative design. Product models play an important role for the designer: by using product models, the designer is able to store, retrieve, and modify data, as well as the data on design methodology itself. Table [1.1] shows the core procedure employed in the design process and the methods used to achieve a final product.

Table [1.1]. Product design and its techniques.

Design Core	Techniques/Applications	To yield	To benefit
Marketing	Competitive; information and parametric analysis.	Understanding of competition, their technology and markets	Customer; company and their employees
Product specification	Customer's order; Quality Function Deployment (QFD).	Real customer requirements and constraints	—
Design Concept	Concept generation; ideas ;decision making and CAD applications	Concepts in shorter time frames	—
Detail design	Experiments design/parameters ;technical drawing and analysis	Much better components	Design Database
Manufacturing operations	Just-In-Time, NC, CNC, MRP, OPT and CIM.	Reduced inventory	—
Product Sales	Customer's order	More profit	Customer satisfaction

Requirements of modern markets in electronics are distinguished by dramatically decreasing prices, higher demands for flexibility, reduced "time-to-market" and growing product complexity. Only well designed products with high performance and quality are able to ensure a company's competitiveness. The application of CAD systems is now vital for competitiveness. Design departments are responsible for product quality, product cost, and for planning and forecasting the innovation potential of each product

[1.1]. Therefore, there has been a great deal of effort to improve the design process itself, and the systems employed. The application of engineering design methodology and its integration with CAD have contributed greatly to the speed with which competitive products can be developed.

Reducing the "time-to-market" is one of the greatest challenges facing industry today, Sadiq et al. [1.2]. The total development time includes the time to design, prototype, test and certify new products. In comparison to mechanical design, studies have shown that in a typical electronic product development cycle, 85 per cent of development costs are committed in the first 5 per cent of the design cycle. In this early (conceptual) design stage, important decisions are made regarding fundamental product technology, materials and manufacturing processes [1.3].

The management of the conceptual phase is critical to the success of the project. Organisations must look closely at the tools available to designers and the sharing of information amongst project teams, particularly when a project encompasses a wide range of technologies (e.g. mechanics, optics, sensors, digital, analogue and power electronics). The optimisation of designs and the requirements for reporting vary considerably depending on the project and demarcations within the projects teams. Obviously, a common engineering environment where data is available to all project engineers, irrespective of discipline would be of major advantage.

Today, there are examples of large organisations that have developed integrated engineering environments (e.g. Boeing, Ford), however, the systems employed have been of high cost and generally out of each the normal capital expenditure of most companies. The benefits to any organisation implementing concurrent engineering and managing design on an integrated basis are manifold. What is required, therefore, is the availability of such a system at a realistic cost.

The market for CAE systems is becoming more resistant to change, particularly as products become established. In mechanical design, for example, AutoCad is now widely used with a large, loyal, user base. There are other examples of this and it is vital for new systems to capitalise on the existing user base. As a result, many mainstream companies now offer an open architecture which permits third party vendors to supply add-on utilities, which can greatly enhance productivity in particular applications. For AutoCad, there are probably more than 1000 third party enhancements available in practically every application area.

### 1.3 DESIGN SYSTEMS

It is now well recognised that CAD tools can be used to speed up electronic circuit design. Jain et al. [1.4] describes a designed package called AJITA, used for rapidly producing electronic circuit design, in printed circuit form, from direct specification in terms of functional building blocks. It provides a library of prototype building blocks that are dimensioned by user interaction for each application (i.e. inputs, outputs and power supply availability). The appropriate circuit, component values and circuit board are then automatically generated. However, AJITA provides a full printed circuit board (PCB) design, given a tool to speed up their design process. With the advent to high-speed digital computers, some of the pioneering work of Kron [1.5], [1.6] was applied to the simultaneous solution of network equations. This formulation was used in part by computer codes such as network analysis program (NET-1) by Malmberg [1.7] and electronic circuit analysis program (ECAP) [1.8].

With the availability of workstations with high resolution graphics, a further step in the evolution of integrated design system becomes possible. Simulated profiles from the layout design interface for x-windows (SIMPL-DIX) [1.9], is an integrated system that utilise graphical interfaces. It provide process design, rather than device design, environments, but it employs an open architecture combined with workstation-based graphics to aid the user interface.

CAD systems have also evolved from electrical-electronic (ECAD) or computer integrated manufacture (CIM) systems [1.10]. The distinction between advanced ECAD, extended CIM, and dictated technology computer-aided design (TCAD) systems is sometimes not trivial [1.11]. Nevertheless, a summary of a designed ECAD systems shown in table [1.2], where each of the systems listed exhibits a different architecture and emphasis certain aspects of TCAD.

New capabilities are provided within the methodology of developing these systems by, for example, extending existing applications, or by integrating additional existing applications which implement the required functionality. From the table shown above, these system are compared using technology CAD (TCAD) levels (i.e. a term which correspond to different views of integrated multi-tool systems): The data level is the process representation which provides the database for tool coupling. The tool level is where the simulation functions are stored. The task level a control environment where operations and flows are defined and executed. Finally, the representation level is the interface through which the user interacts. Additional, comparison schemes can be employed such as the methods of integration used and overall performances. In this table also:

- *internal* means that the system is not used outside the institution.
- *Production* means that the system is known to be in use somewhere.
- *Commercial* means that the system is commercially available.
- *Blank fields* indicate lack of reliable information.

Table [1.2]. List of ECAD systems and tools integration approaches [1.11]

Name	Institution	Status	Data level	Task level	Presentation level
MECCA	AT&T	int. prod.	awk/sed, c++	UNIX, Shell	Tk
PREDITOR	CMU	experim.& commercial	CDB/HCDB	Tcl	Tk, Motif
EASE	Intel	int. prod.	PIF derivate.	UNIX Shell, FASST/TEL	Motif
CAFE	MIT	internal	BPIF/Gestalt	MIT PFR	
P&D Workbench	NEC	int. prod.		MEDLEY+ ESCORT	DAIJOBDA
UNISAS	Oki	int. prod.	GCOS	UNICOL	
IDDE	Philips UK	int. prod.	ASCII PIF derivative.	none	Apollo DIALOGUE
PROSE	UC Berkeley	internal	BPIF, SWR	Tcl	Tk + VEM
SIMPL-IPX	UC Berkeley	internal	Converter (BTU)	none	SIMPL-DIX
OrCAD/VST	OrCAD sys. corporation	commercial	PASCAL or C	IBM PC AT / XT	OrCAD/VST
eCAD plus	Microdata systems Srl	commercial	EFORM	IBM PC AT-286, 486	eCAD OrCAD/STD

#### 1.4 MARKET FOR MECHANICAL DESIGN SOFTWARE

Generally, revenues in the 90's for CAD computer-aided manufacturing (CAM) and CAE rose, according to market researcher Dataquest [1.12]. The leading applications area are mechanical, electronic CAE, integrated circuits (ICs) layout, PCB layout. Dataquest's top vendors in total factory revenues were IBM, Sun Microsystems, Hewlett-Packard, Intergraph, Digital, Computer vision, and AutoDesk. In mechanical software, IBM, AutoDesk and parametric technology (PTC) lead. In the top 10 companies, total growth was almost double that of the market as a whole. AutoDesk and Intergraph are setting the pace in ECAD software applications, with respective market shares of 21.7 and 13.8 per cent.

The growth of the use of CAD software has climbed rapidly each time computer hardware crossed a threshold; first minicomputer, then UNIX workstation to PC's [1.13]. Mechanical CAD software is one tool that is widely used on these platforms. As they vary in their structure and use, many of these software incorporates advanced features like unified parametric geometry which facilitates 3D wireframe, surfaces and feature-

based solid modelling, into a single associative data structure with combined parametric modelling. Market research carried for mechanical CAD systems show a high level of competitions between world leading companies. According to Dataquest's market statistic, CAD continues to dominate the world-wide CAD software market comparing to its competitor IBM. For example, in the mechanical vendor community, 1995 software market leader AutoDesk was replaced by IBM, which moved to the top position world-wide, followed by parametric technology (PT).

Mechanical system developed by AutoDesk has carved a solid position as the price-performance leader in mechanical design. Its comprehensive application programming interface available that can deliver graphic performance under Windows or DOS. In addition, AutoDesk have developed the new software in combination with programs developed by other industrial leading companies from the mechanical sector to provide complete design. An enhancement version of mechanical CAD was made including additional features: New interactive 3D graphics visualisation capabilities and other customer requested enhancements. Market analysts believe that AutoDesk mechanical DeskTop provides a complete design through manufacturing solution for the mechanical CAD market.

IBM keep improving their CAD products such CATIA. The latest updated versions available in the market contains different products grouped into different engineering areas: Mechanical design; manufacturing; analysis and simulation; architecture. Additional features include the capability to provide custom modelling facilities to suit various different tasks and to update the modelling technology to include parametrics and variational geometry with modern sketching facilities. CATIA is now multi-mode modeller with integrated solids, wireframe and surfaces. For consistency with other systems CATIA has bi-directional associativity between the various modelling technologies. This is very much under the control of the user where data can be updated with the assistance of data management system. The system provides a programming language called interactive user access (IUA), which can be used for developing further software routines and applications of different tasks.

Available on PCs and UNIX workstations, Intergraph's MicroStation is one of the most complete draughting packages currently on the market. The package is now on its fifth version. Design tools enhancement make it flexible system to use. The development language (MDL) enables users to developed third party software of various engineering areas to perform specific tasks. Similarly, AutoCAD is a full featured CAD system, capable of handling most designs duties including mechanical design.



Pro/Engineering is another CAD system used for mechanical design, it provides rapid prototyping to speed up the whole process of design and manufacture in a wide range of industries [1.14]. It contains facilities such as stereolithography which works by taking a 3D model of a component and slicing it into thin layers. These layers are further processed, for example, to ensure an even thickness for each layer of resin.

Despite all the facilities that are available within the system, many improvements have been carried out. At release 16, the definition of geometric tolerances has been much improved. Instead of selecting a series of menu picks, tolerance can be defined in dialog editors, which are windows-based and contain a panel of tolerances symbols so that the selection can be performed graphically. New set of commands have been added concerning general feature creation, to allow creation advanced general shapes (e.g. fillets, chamfers). These facilities have been improved by the addition of a dialog editors, allowing different features to be changed, extended, and redefined. Further improvements have been carried towards the placement conditions in assemblies. As the facility was available but less interactive, a component can be placed dynamically using either, translation or rotation techniques. Further improvement also taken towards the design of GUI including some enhancement to design tools. With these tools assembly files are automatically generated for each component in the assembly, as this saves time and facilitates the individual specification of attributes (i.e. creates assemblies and readily control colour and texture).

## **1.5 DESIGN SYSTEMS AND METHODOLOGY**

Mechanical design and electronics design are typically performed as separate functions in engineering [1.3]. However, the increasing use of electronics in what were once predominately mechanical products and the increasing emphasis on packaging and other mechanical considerations in electronics products, requires the implementation of a concurrent engineering approach.

### **1.5.1 Product Design Approach**

When designing a product, decisions have to be made on what to optimise, with several consequences on manufacturing costs. These decisions can completely defeat the designer's intentions. Top management, engineering, purchasing, personal, and manufacturing each contribute to the success or failure of a product. The design requires a great deal of analysis, investigation of basic physical processes, experimental verification, difficult decisions and choices. In many cases, the choices become more and more difficult as the design gradually works its way toward acceptability. Figure 1.1

shows the product design process supported by the concept of computer-aided design (CAD).

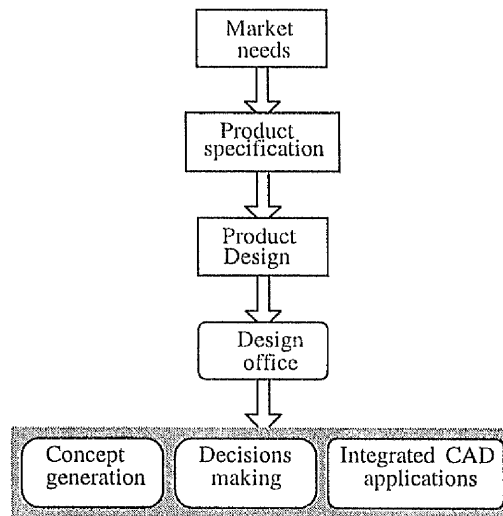


Figure 1.1 Product-design process supported by CAD applications.

The essence of the approach is to increase the quality of the influential, early decisions made. Such consequences include product function predictions, estimates of fabrication and assembly cost, concept designs of fabrication and assembly systems and equipment, methods of testing, etc.

Quality function deployment (QFD), which is based on the assessment of market needs, is a procedure used to assist in the design core operation (as shown in table [1.1]). To enable market requirements to be met, company skills and experience must be employed at this stage. Critical characteristics that represent the market requirements will be developed. The role of the QFD procedure at this stage, is to convert the market requirements plan for the finished product into its components and the characteristics demanded [1.15]. These characteristics are critical to market requirements, and will be developed as the basis for product design and process development. Added to this, is the market evaluation of competitor products against customer criteria. This enables direct assessment of the product specification and determines the potential positioning of the product design against the competition. Figure 1.2, illustrates the use of QFD procedure, where a matrix of product control characteristics is developed.

The resulting targets for product design process are cascaded into subsystem characteristics, and are used to measure the performance of the product and process development activities at various stages in which engineering designs, such as (electrical, mechanical, optics, etc.) could be involved. Further, these characteristics are deployed to ensure that customer needs flows through the entire production and supply process, from design through development and production to market and sales.

			Product requirements									
Customer requirements	Primary	Secondary	Tertiary	Steering feedback	Ride	Handling	Chassis isolation	Engine isolation	Performance	Economy	Durability	Packaging
	Quiet chassis	No thumps	No road noise									
	Low engine noise	Smooth acceleration	No shunt									
	No rattles	Shake free										

Figure 1.2 Customer requirements and product characteristics matrix [1.15].

## 1.6 DETAILED DESIGN PROCESSES

Within the framework of the overall design methodology, there are specific detailed engineering design activities which are employed. These will now be discussed with a view to identifying the potential for integration.

### 1.6.1 Electronics Schematics and PCB Layout

Circuit layouts detail the position, type, reference numbers, and component connectivity. Layouts are constructed by selecting components from libraries that contains a selection of standard electronic components. Associated with each component is a complete description of geometry, mechanical properties, thermal characteristics and pin configuration. Systems also provide for the creation of user defined libraries of components (i.e. symbols design environment). When constructing in this manner from library components with associated data, the resulting board model contains the necessary information required later for thermal, analysis and reliability assessment.

Software systems involving PCB's, have a single database for mechanical and electronic design, eliminating problems in transferring data between two separate systems, to permit mechanical considerations to be factored into the design process earlier. Such integrated packages are also useful for passing data to manufacturing. Most of these integrated

systems, a 2D layout of the circuit is used as a basis for generating the PCB layout. Standard (gopher) files can be passed to PCB manufactures for rapid manufacture of PCB's.

### 1.6.2 Circuit Simulation and Analysis

Circuit analysis is a necessary part of circuit design [1.16]. Once a design for a circuit has been completed, the performance evaluation of the design must be tested to ensure that the circuit does perform to specification. Often this involves testing for DC operating point and performance under applied stimuli.

The circuit to be analysed is described to the simulator, using a special purpose circuit description language or format which defines the component, its value and its connectivity. The circuit simulator simulation program with integrated circuit emphasis (SPICE) developed by Nagel [1.17], has evolved to provide an extremely accurate and widely used simulation tool for electronics. Figure 1.3 illustrates the whole process of using SPICE for circuit analysis.

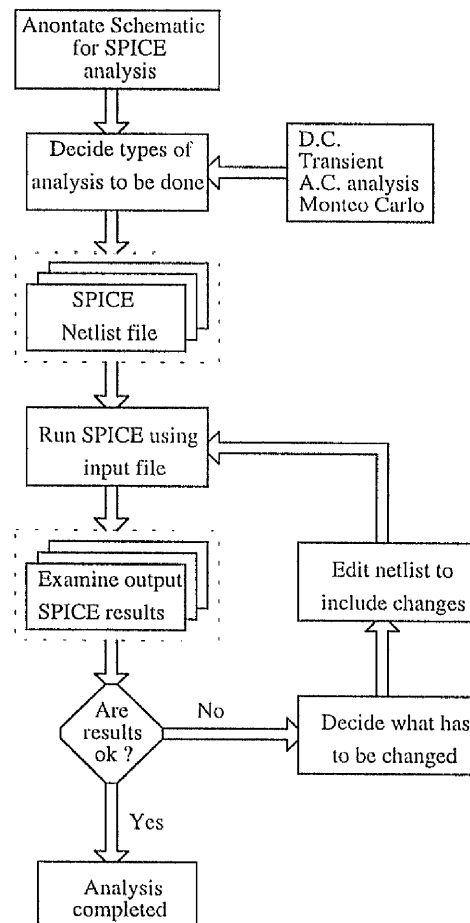


Figure 1.3 Flowchart of SPICE general analysis process.

Another option available as part of SPICE package is monte carlo (.MC) analysis which allows the designer to do statistical, sensitivity, and worst case analyses on circuits [1.18]. It has three elements : tolerances, multiple runs and output. These elements are common to both analyses: monte carlo and Sensitivity, worst case (.WCASE). With monte carlo analysis, model parameters can be given tolerances, and multiple analysis (i.e. DC, AC, or transient) can run using these tolerances. A typical application of using monte carlo analysis would be predicting yields on production runs of a circuit. The worst case analysis is used to find the maximum or minimum value of a parameter given device tolerances. With worst case and sensitivity analyses, model parameters are varied one at a time, and other analyses (DC, AC, or transient) are run for each variation. A typical application of worst case and sensitivity analysis would be investigating the effect of critical components values.

CAE techniques are becoming virtually indispensable in design and analysis of electrical-electronic circuits as manufactures condense increasingly complex devices into smaller spaces [1.3]. Much of complex circuitry would be impractical to develop without the main function of the computer. Automated drafting was the main function of the computers in developing circuits. Now sophisticated software tools are used to cover a range of tasks, one of the most important of which is analysis that helps reduce errors in the circuitry and highlights areas for improvement.

In order to compare the salient features of the different levels of simulation. Table [1.3] adapted from Russell *et al* [1.19], lists the essential characteristics of a circuit level simulator.

Modelling devices can be divided into at three distinct areas, namely process, device, and circuit simulation. Many simulation codes have been developed which fall into each of these categories, e.g. complete modelling program of silicon technology (COMPOSITE) [1.20], for process modelling; CURRY [1.21] for fundamental modelling of devices; SPICE [1.17] for circuit modelling. Each program has its own form of input in which the user must describe the device, process or circuit. These programs vary considerably in user friendliness but generally tend to be text based, requiring a description in the form of keywords and parameters.

Previous attempts to produce an integrated device modelling environment show an evolutionary trend. The system described by Boning [1.22] introduced uniform user interface to all the programs contained within the system but still lacks the concept of standard interface files. The user requires input language to drive all the modelling programs integrated to the system. This is replaced by a common user interface via

which input is translated into program specific instructions. However, it was concluded that the user friendliness is improved, but the lack of standard interface files makes the system unsuitable for further extension.

Table [1.3]. List of existing programs for circuit simulation [1.19].

Existing programs:	ASPEC, ASTAP, DIANA, IMAG III, MSINC, SLIC-M, SPICE, SPLICE.
Function:	To perform accurate a.c., d.c. and transient analysis, and also noise temperature and sensitivity analysis.
Simulation primitives	Resistors, capacitors and inductors, diodes, bipolar transistors, field-effect transistors, sub-circuits.
Signal Algorithms	Currents and voltages as functions of time. Non-linear analysis techniques. Numerical integration techniques. Sparse matrix techniques.
Comments	(1) Very close link to technology. (2) Ideally, the input to this level of simulator should come directly from the layout. (3) Simulation accuracy is limited only by the device model accuracy (4) Major applications in digital circuitry are. (a) simulation of circuits with tight coupling between input and output: e.g. Schmitt triggers, sense, amplifiers, flip-flops; (b) examine of electric faults which do not manifest themselves as logical faults: e.g clock feed through, charge sharing. (5) Used extensively for the characterization of basic cells.

### 1.6.3 Thermal Analysis

Thermal analysis of circuitry is important to ensure that heat build up does not exceed the rated capacity of components, and that thermal expansion does not deform the board itself [1.3]. In this case building the layout model is important to perform such analysis, allowing thermal properties to be passed on as loads and boundary conditions to the finite element model for thermal analysis.

The resultant output of thermal analysis is a coloured coded display of the entire board, with the highest temperatures shown in a unique colour. This allows the designer to easily detect general overheating of the board, and other trouble-some heat distributions, and make necessary alterations to layout or cooling system design.

#### **1.6.4 Mechanical Drafting and Solid Modelling**

Many CAD systems are customised for various disciplines, with different standard libraries, dimensioning features, and links to other analysis systems [1.23]. Usually, these systems are classified according to the engineering field or which the system has special features and certain techniques for creating drawings, and links to analysis and manufacturing. Mechanical CAD, for example, supports dimensioning, has symbols for constructing 2D and 3D layouts with the assistance of database for part numbering and material take-off. Extensive libraries of components are now available for many standard parts. (particularly AutoCad and MicroStation ).

Solid modelling deals with the design and representation of real-world objects such as structures, machines, components, and assemblies of parts. Besides producing realistic images, solid modelling can be used for analysis. Finite element models, for examples, can be constructed to analyse stress, deflection, vibration, or temperature distribution. Kinematics models of mechanisms and robots can be checked with animated solid models. Mechanical parts assemblies can be checked for fit interference. Sophisticated techniques can be developed to support these applications. Using a unified database, in manufacturing, solid models can be used to generate programs to control lathes, milling machines, and other machine tools.

There are three basic modelling techniques available in design 3D models: wireframe, surface modelling, and solid modelling [1.24]. Wireframe modelling is a relatively uncomplicated technique, whereby designers build up the raw skeleton of a 3D model. Wireframes are classed as a two and half dimension technique. In order to facilitate such capabilities, there are specific surface modellers provide features such as clashing detection and surface connectivity. Most of the advanced solid modellers offer some kind of method by which 3D models are built. One commonly used method is the parametric design method, where parts or component features designed using different parameters. However, new and different modelling techniques are employed in some packages. Feature-based design based on constructive solid geometry (CSG) methodology is one example, allows users to utilise 'real' components (e.g. a screw will have an actual thread). Other techniques such as boundary representation (B-Rep) is also used to developed different automated techniques (e.g. automatic feature recognition).

#### **1.6.5 Mechanical Dynamics**

Solid models are ideal for studying the dynamics of mechanics, machine tools, and robots. They can be used not only for kinematics analysis of motion, but also for the dynamic analysis of stress and deflection [1.23]. Manufacturing and numerical control

(NC) is an example of using solid modelling to investigate kinematics simulation. In this case, solid modelling used to produce geometric and topological information in database, and further processes take place such as determining the sequence of production operations, part programming starting with the geometric definition of the part that is stored in the database, verification permits observation of the entire machining sequence and material used, followed by the production stage, where setting up machining tools and management movement are finalised.

System such as SolidDesigner, is designed for dynamic modelling [1.25]. With the support of database technology the system allows to modify and regenerate geometry at any stage in its creation sequence. The system includes various design facilities and functions, allowing different models to be built and manipulated at any stage, new freeform modelling capabilities allow to construct lofted surfaces. These parts can include imported surfaces models, and models can be used as a non-planar profile, which can be lofted to build a solid model. The models can be linked to other applications such as NC or FEA. Another significant feature included in this system is that it provides a common Lisp based programming environment, facilitates third party developers to build add-on, user interfaces and different applications.

### **1.6.6 Finite Element Analysis**

Analysis system identifies stresses and strain for mechanical, thermal and other effects. Some of these systems seem to have been designed without considering integration with core design system [1.26]. Most analysis systems on the market are based on the finite element method. Analysis software such as boundary element method (BEM) designed for computational mechanics, contains different modules: mechanical design which includes stress and thermal analysis, corrosion simulation product, and 3D surface elements allows to define rough description of the problem and the software will automatically take an adaptive approach.

## **1.7 CONCLUSIONS**

ECAD systems for electronic and mechanical design have developed separately and have evolved tools specific to each discipline. Tools for PCB design (schematics and layout) are now mature with many vendors offering similar high level functionality. However, in today's markets the drive to reduce product development time-scales necessitates a truly multi-disciplinary, concurrent engineering approach. Electronic and mechanical systems can no longer be designed in isolation. For example, a typical project for a mobile phone will involve digital, radio frequency (RF), power and analogue electronics designers as well as industrial designers, and mechanical engineers. Examination of the project plan



for such a project will reveal that key to reducing time-to-market for such products is the concurrent design of electronics and packaging. It is therefore necessary at an early stage to define mechanical constraints (e.g. PCB control drawings). The design is therefore closely coupled and the designers must utilise tool-sets which permit design data to be transferred easily in 3D form. What is required, therefore, is a means of designing products on a common platform, where designers can access a shared database, which evolves as the product design evolves. Although, the individual pieces of this jigsaw are available as individual products, a unified approach to product design is not yet available. It is therefore the prime objective of this work to examine the issues which are important in the development of such a system, and to develop a demonstrator to experimentally establish limitations and solutions.

## **1.8 SCOPE OF PRESENT WORK**

The objectives of this work and scope are as follows:

- (1) To review existing design systems and define limitations precluding use in a fully integrated concurrent engineering system.
- (2) To specify design requirements for a concurrent engineering software system for a PC platform, building on an existing open architecture system.
- (3) To develop a concurrent engineering design and analysis system.
- (4) Experimentally examine functionality issues and performance on multi-disciplinary product designs.

### **1.8.1 Outline of Thesis**

This thesis is organised in eight Chapters: Chapter 2 looks in detail at the design and development of the design system as a unified environment. It starts by illustrating the system design life cycle in which design specifications and functionality requirements are considered. The chapter then gives a detailed review of the implementation methodology including the system core parts, hardware considerations. Particular attention is given to the software used in the development, by introducing Intergraph MicroStation and its development languages and the reasons why this system was chosen. Attention is also directed towards the design architecture where the basic design items are identified and described together with their roles. A large portion of the work has focused on the specific software requirements including design tools and automation.

Chapter 3 deals with the investigation of development tool-sets that can be used for component design and modelling. It first reviews a detailed section on component

modelling where modelling objectives and fundamental concepts are highlighted and a survey on different circuit modelling languages is reported. The chapter then looks at the issue of electro-mechanical design. In particular, it concentrates on the use of parametric design techniques and the relationships between parts and their attributes. In addition, a design and modelling procedure is described by illustrating its intention and its performance measured by studying a design examples of inductor and transformer each of which applies different design specification.

Chapter 4 investigates the automatic generation of netlists from circuit schematics. It describes the importance of netlist in electronics design, followed by a survey of previous used techniques. The chapter also reveals the role of using PSPICE package as circuit simulation tool whereas data flow diagram illustrating the direct link between PSPICE and other design environment is discussed. Based on the main problem definition and general development rules, the chapter then looks at the different techniques investigated for generating schematics nelists. Finally, the chapter ends by illustrating a case study of circuit documentation and data extraction showing the performance of automated tools.

Chapter 5 reports on the database facilities, where a short history and evaluation of database systems is revealed by describing the role of database in CAD, followed by a brief literature survey of commercial database systems giving a detailed description of rational database concepts and development models. Based on the developed conceptual model, design and development of different database tools and applications is described, where three main development aspects are considered (i.e. report, update, and manipulate).

Chapter 6 Concerns the design and development of a graphics user interface (GUI). It gives a review of graphics systems and their methodologies for developing user interfaces followed by engineering consideration towards graphics design and user interface generators. A large part of the work has focused on the design methodology which based on using MDL and UCMs languages. It describes the system design levels, where each level contain a set of tools and functions performing various design tasks.

Chapter 7 reviews the performance and evaluation of the software. It demonstrates the principles of integrating different design environments by considering a coupled example of electro-mechanical design.

Finally, Chapter 8 covers the general conclusions of this work together with recommendations for future work.

**REFERENCES**

- [1.1] Frank B., "Modelling Methods for a flexible Computer-Aided Embodiment Design Systems," Research in Engineering Design, Verlag new York Inc., pp. 15-34, Spring 1990.
- [1.2] Sadiq M. et al., "Prototype system for standard cell based ICs will ultimately reduce 'time to market,'" IEEE Circuits & Devices, Vol. 11, No. 2, pp. 15-24, March 1995.
- [1.3] John K., "Manager's guide to Computer-Aided Engineering," On Word Press, 1580 Centre Drive, Santa Fe, NM 87505 USA, ISBN 1-56690-038-7, 1993.
- [1.4] Jain L. C. et al., "PC software for electronic circuit design in printed circuit form ," South Australian Institute of Technology, Computer-Aided Engineering Journal, Vol. 5, No. 4, pp. 148-152, August 1988.
- [1.5] Kron G., "Tensor Analysis of Network," New York, 1993.
- [1.6] Korn G., "A set of principles to interconnect the solutions of physical systems," Journal of Applied Physic, Vol. 24, pp. 965-980, 1993.
- [1.7] Malmberg A. F. et al., "NET-1 Network Analysis Program," Los Alamos Scientific Laboratory, Los Alamos, N.M., Rept. LA3119, 7090/94 version, August 1964.
- [1.8] "1620 Electronic Circuit Analysis Program [ECAP] [1620-EE-02X] User's Manual," IBM Application Program File H20-0170-1, 1965.
- [1.9] Wu H. C., "Simulated profiles from Layout-Design interface in X (SIMPLE-DIX)," in IDEM Tech. Dig., pp. 328-331, 1988.
- [1.10] Kleinfeldt S. et al., "Design methodology management," Proc. IEEE Trans., Vol. 82, pp. 231-250, February 1984.
- [1.11] Stefan H. et al., "VISTA: User Interface, Task Level, and Tool Integration," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 14, No. 10, pp. 1209-1221, October 1995.

## *Chapter 1: Introduction*

- [1.12] CAD/CAM., "Dataquest report finds modest growth in 1993," CAD/CAM Journal, The guide to integrated design and production, Vol. 13, No. 8, pp. 5, August 1994.
- [1.13] Mark N., "The design technology revolution," Engineering Designer Journal, Vol. 18, No. 2, pp. 4-5, March/April 1992
- [1.14] Charles C., "Pro/Engineer," CAD/CAM Journal, The guide to integrated design and production, Vol. 15, No. 3, pp. 43-36, March 1996.
- [1.15] Pugh S., "Total design," Integrated Methods for Successful Product Engineering, ISBN 0-201-41639-5, 1990.
- [1.16] Walter B., "Computer-Aided Circuit Analysis using SPICE," Prentice-Hall International, ISBN 0-13-168394-2, 1989.
- [1.17] Nage L. W., "SPICE2: A Computer Program to Simulate Semiconductor Circuits," ERL Memo Memo No. ERL-M520, Uni. Calif., Berkeley, Electronic Research Lab., May 1975.
- [1.18] MicroSim corporation, "PSpice Circuit analysis user's guide," 20 Fairbanks, Irvine, California 92718, Version 5.0, July 1991.
- [1.19] Russell K. C. M. et al., "C.A.D for V.L.S.I," 1-Integrated circuits-Very Large Scale Integration-Data processing, ISBN 0-442-30618-0, 1985.
- [1.20] Lorenz J. et al., "COMOSITE- A complete modelling program of silicon technology," IEEE Trans. Computer-Aided Design, Vol. 4, No. 4, pp. 421-429, October 1985.
- [1.21] Polak S. et al., "The curry algorithm in Simulation of Semiconductor Devices and Processes," Pineridge, Vol. 2, pp. 131-135, 1986.
- [1.22] Boning D. S. et al., "A workstation approach to IC process and device design," IEEE Design & Test of Computers, Vol. 5, No. 6, pp. 36-47, April 1988.
- [1.23] Bruce R. D., "Computer Graphics for Engineers," by Harper & Row, Publisher, Inc., ISBN 0-06-041670-X, 1988.

## *Chapter 1: Introduction*

- [1.24] Charles C., "Dynamic design," CAD/CAM Journal, The guide to integrated design and production, Vol. 15, No. 2, pp. 29-32, February 1996.
- [1.25] Joe O., "3D modelling," CAD/CAM Journal, The guide to integrated design and production, Vol. 15, No. 1, pp. 51-52, January 1996.
- [1.26] "CAD/CAM," The guide to integrated design and production, Vol. 13, No. 8, pp. 5, August 1994.

# 2

## System Design and Development

### 2.1 INTRODUCTION

This chapter presents an integrated design environment for electro-mechanical design and analysis, based on the use of the CAD system MicroStation and its development language (MDL). The environment is divided into four main areas, viz: the design of 2D schematic capture; 3D modelling for electro-mechanical layout; database for data storage and manipulation; and simulation for design analysis and performances.

The system design concept is concerned with the structure of the functions that perform at each level and by each component of the system. The features which are essential in system design are as follows:

- The system core parts and their objectives, which includes preliminary description and design specifications.
- The establishment of the operational capability of the system and physical equipment needed. Physical facilities depend upon the form of computer and storage requirements and space needed, while operational requirements involve the manner in which the physical facilities are used and the way in which the information is organised and accessed.
- The specification of the functional structure of the system and the development of a description of each system component. The hierarchical structure of the components that comprise the system is identified and each component is described in detail. Figure 2.1 gives an overall diagram of system design stage. It describes the three

main steps needed to develop a functional system, where the inputs, outputs, and internal logic of each component are defined.

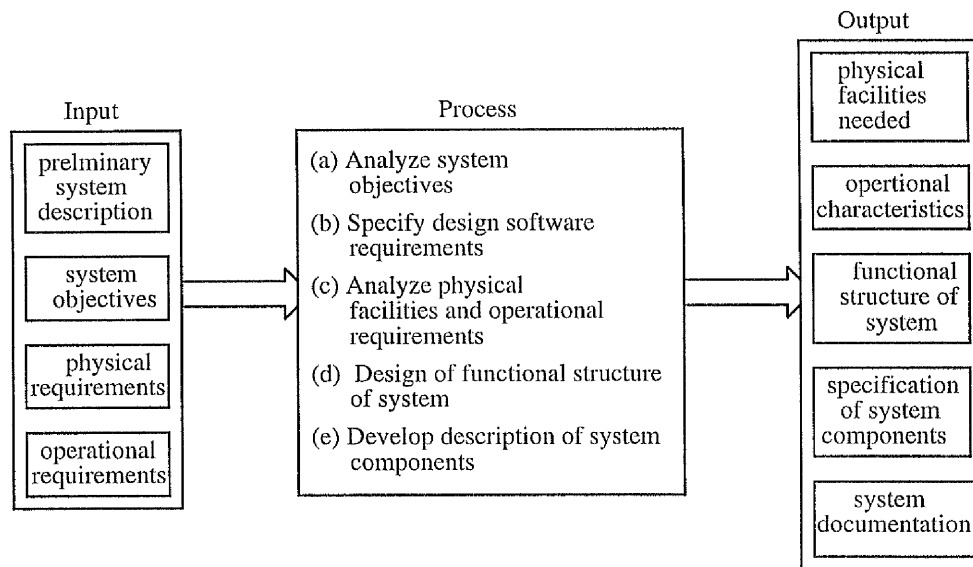


Figure 2.1 Overall system design life cycle.

Additionally, the starting point in system development is the functional requirement [2.1]. The designer begins by establishing the system's function, determining the system purpose and expected outputs. The essential content of the system specification is a detailed statement of system function. This is to give an overview of the design in general, so that the individual functions and performance specifications that have to be achieved can be identified in the finished product [2.2].

CAD/CAE is one way of coping with the problem of designing and developing an integrated systems. This problem is acute in electrical/electronics design. The following requirements are essential in developing an ideal CAD system:

- automatic generation of design data compatible between design tools. This indicates that some kind of translation facility will be required to produce design data in a useful and valid form.
- high-level design languages and tools.
- a supportive and adaptive human-computer interface. This is to provide graphical input/output wherever possible.
- access to information in the system, combined with security against accidental damage or loss of files.

- a dedicated, customised system which provides good relevant information quickly.
- There should be an increase over the current speed of information retrieval, verification and simulation.

These points can only be resolved by referring to the existing markets at which these CAD systems are aimed. Having specified some of the features which would be expected on an integrated CAD system, it is possible to outline its high level structures and operations. Production specification remains a unique function because of the informal relationship between customer and designer. It could be thought of as kind of translation of how the system is likely to be designed. Figure 2.2 illustrates a typical work breakdown for electronics products systems.

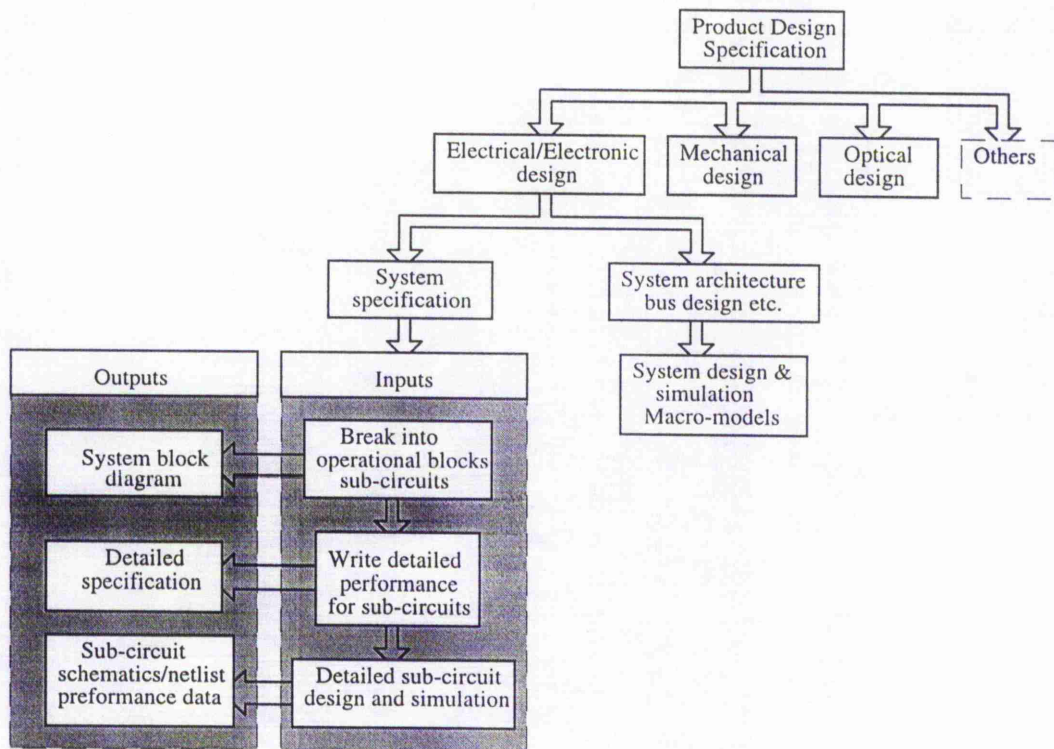


Figure 2.2 Breakdown for electronics products systems.

Obviously, there is a considerable effort involved in developing a full CAE system from scratch. As the objectives of this work are to demonstrate a working system which would be easily integrated into a design office, these objectives are best served by utilising an existing open architecture CAD system of the two dominant market products.

MicroStation is the chosen CAD system for developing this software. This is because it has several properties that can assist in developing new applications. These properties are as follow:



- It is widely used PC-level system in the component manufacturing industry, and it provides enough flexibility in drawing generation.
- It provides the ability in designing third party applications using its development language (MDL) and user commands macros (UCMs). The syntax of MDL is compatible with the C-language in order to keep the portability. The macros language allows to merge and execute commands in simplified way. Detailed description to these languages is explained in Section 2.4.
- MicroStation is the core graphics software program for different applications development (e.g. mapping, architecture, electrical/electronic, electro-mechanical designs etc.). It has many design features and benefits, including an easy-to-use graphical user interface (GUI).
- MicroStation provides structural query language (SQL) extension to enable the system to access and interface with different types of database (e.g. relational interface system (RIS), informix, oracle, xbase) through built-in server.
- It has library facilitated environment, allows to construct various library cells to represent components in both 2D and 3D.
- It has the ability to render, this makes it possible to design with 3D modelling taking into account the ability to convert 2D design files to 3D formats.

In the following sections, the system framework architecture is described with the highlight of key components. The potential issues that must be resolved to obtain an integrated environment are detailed, with particular attention paid to design flow levels, integration of the entire design process, and data management design. In addition, the outline of the software specific requirements is detailed.

## **2.2 DESIGN CONSIDERATIONS**

The system's design process begins with an analysis of the structures that compose the problem domain, where understanding the problem domain equates with the ability to identify the core parts and their relationships. This provides the ability to map from one part to another. At an early stage in the system design process it is only possible to define overall relationships between parts of a system, and a block diagram may be most appropriate. The block diagram shown in Figure 2.3, outlines the system core parts. In

addition, detailed descriptions of the development software and hardware requirements are outlined in the following section.

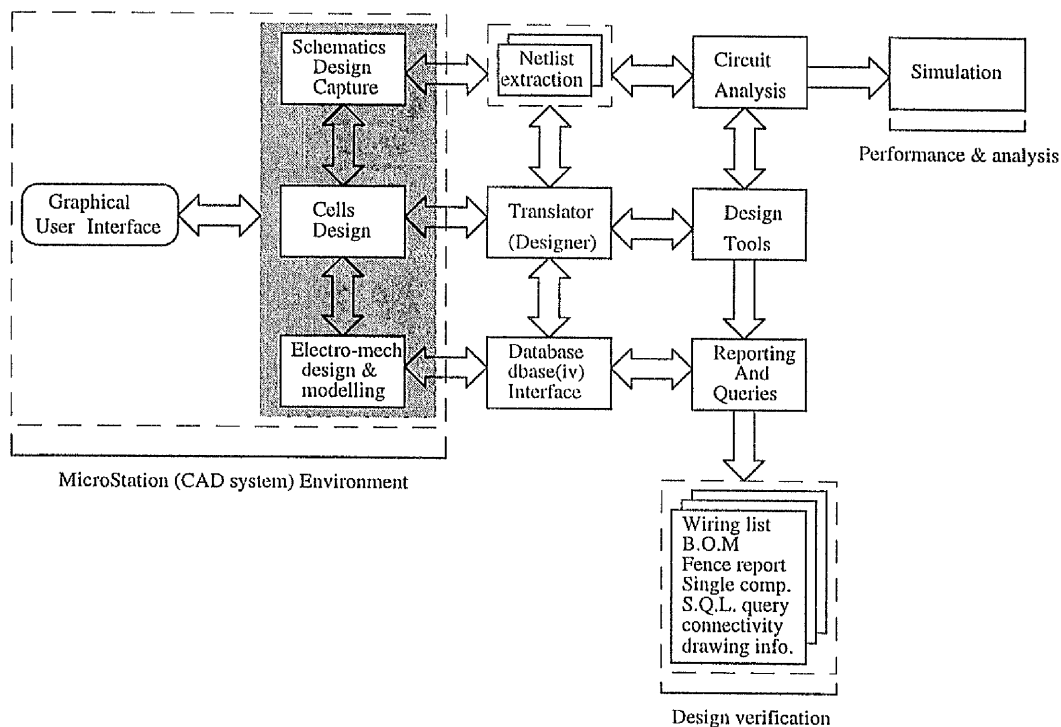


Figure 2.3 The system components block diagram.

## 2.3 SYSTEM DESIGN ARCHITECTURE

The system universal architecture is shown in Figure 2.4, which consists essentially of basic design items interacting with each other and of interfaces that handle the flow of information between the other involved environments. The key items comprise graphical user interface (GUI), Cells libraries for components symbols design, Database dbase(iv) interface, and Design activities include schematic design and modification, electro-mechanical design, automation includes netlist extraction for simulation, reports generation, data attachment, retrieving and reviewing. Description of these items is outlined below:

### 2.3.1 Graphical User Interface (GUI)

The GUI consists of a number of interactive levels through which the designer and the system communicate. It includes the keyboard, display, buttons, pull-down menus, palettes, icons, dialogue boxes, labels, and many other functions which can be used to assist design tasks and performances. It provides the editing of operations corresponding

to user requests, and it provides design templates containing text editors for components design and modelling parameters, as described in Chapter 3.

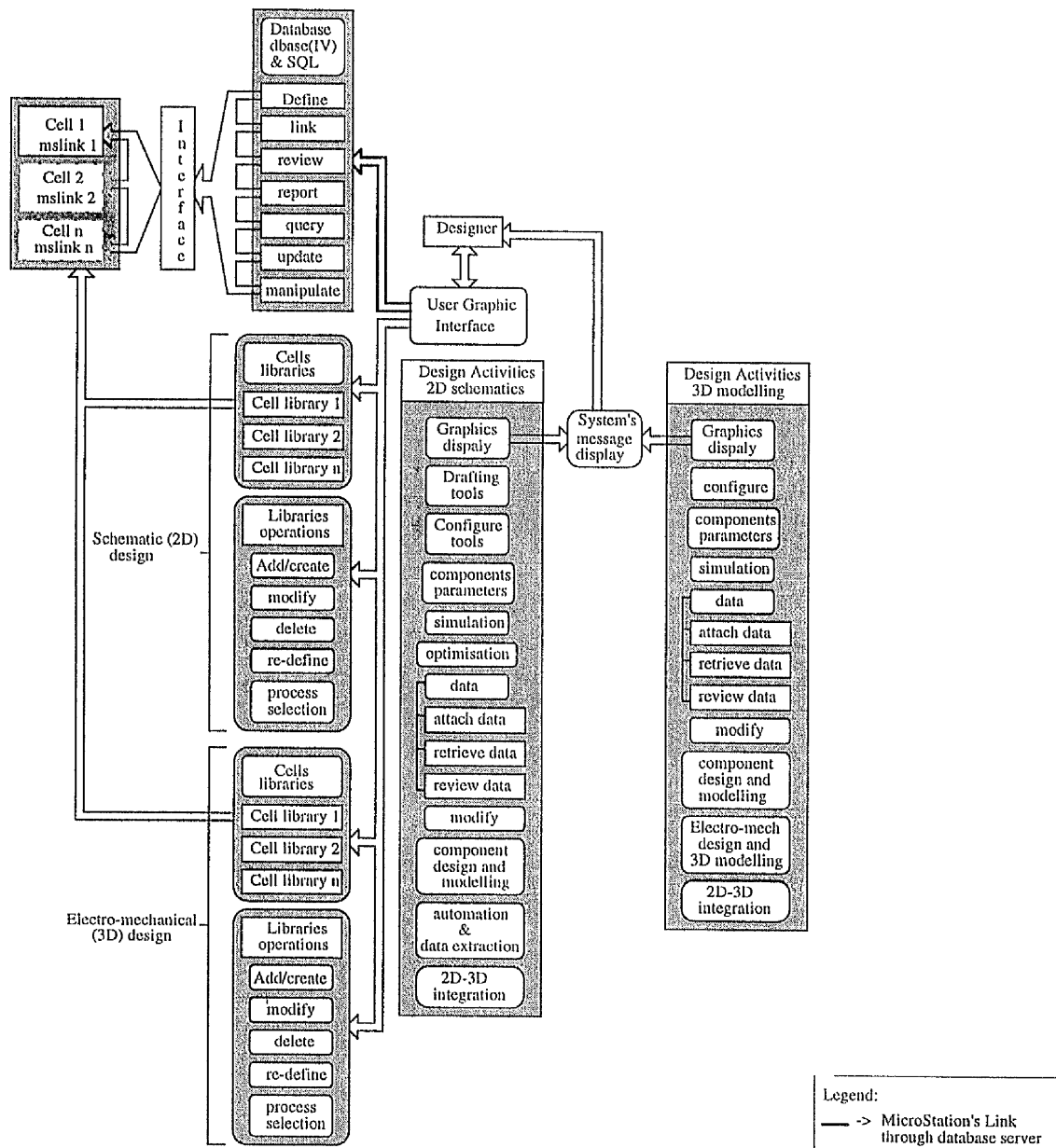


Figure 2.4 Illustration of the design detailed architecture.

The GUI consists of different control buttons that can be used to invoke and allow different applications and tools to be activated. There are several control buttons, each displays different views of utilities. For example, The drafting control button allows tools utilities window to be displayed as an iconic menu that displays several utility icons. These icons are used to open and invoke different applications and commands. In addition, other facilities and tools are included, and detailed description of their design and development can be found in Chapter 6.

### **2.3.2 Cells Libraries**

The components used in designing schematic drawing are called cells, and they are stored in specified libraries. They consist of different elements that MicroStation uses (i.e. lines, arcs, circles, text, etc.), sometimes referred to as complex elements. The system components libraries support international standards including IEEE, BS. Commonly used devices are stored in symbols libraries and are easily accessible with on screen icon menus (i.e. place palette). As each component is placed into the active schematic drawing, name, location and other descriptive information stored in the database (i.e. referred to as header file). This information are extracted and used for other design purposes. Libraries operations contain functions allow to carry other design tasks such as add/create customised libraries to establish design standards. Additional tools are also designed to modify, delete, and re-define existing components and devices (refer to Chapter 6).

### **2.3.3 Design Activities**

Most of these contents are software programs written using either MDL or UCMs. They perform, advise, and assist design tasks and operations. Their role is briefly described below:

- Configuration, consists of different written software programs which can be controlled and manipulated to perform specific design changes and modification to design workspace, 2D schematics, electro-mechanical components.
- Schematic design, interactive tools for circuit creation and editing are designed. A complete circuit can be placed efficiently. Schematics design basically is a function of a circuit with little emphasis to its physical characteristics. It is important for use in equipment production testing and analysis. It also used to describe all circuit functions and values of components using standard symbols in most cases. The primary goal of using schematics diagrams are used to show the functional relationships between parts, allowing the circuit to be analysed by means of circuit extracted data (i.e. netlists), and simulation program with integrated circuit emphasis (PSPICE).
- Optimisation, is the process which involves the probing of component design using different techniques based on the variation performances with respect to design parameters and specifications. The objective of this process is to maximise the parametric issue for the most possible design solution.
- Modify, is part of configuration process described above where different tools are designed for the purpose of schematic designs modification (e.g. tools palette). Their

role including: replace component with another, wires are extended, stretched and copied using fence or cursor selection. Components and wires graphics attributes can be changed and modified (i.e. style, weight, and colours) either by group or as individual selection

- Automation, is referred to as various software programs written to perform specific tasks automatically. Components, wires, and drawings each have design information attached which can be extracted and used for further design tasks. These data can be edited to include components IDs, wires numbering, and part numbers. For example, part numbers may be assigned to generate bills of materials and wire identities assigned to generate wiring lists as an ASCII formats. Extracted data can also be merged into schematic workspace for convenient review. Other automated tools are also designed and developed to assist the construction of 2D schematic. These tools include: automatic junction placement, automatic element recognition using a file scanning technique.

## **2.4 SYSTEM IMPLEMENTATION**

The precise nature of the implementation stage is necessarily dependent upon the type of system been developed [2.4]. For example, in hardware system development, detailed logic diagrams for the components of the system are constructed before they are built. It is subjected to a functional test to ensure that it operates according to specifications. The various components are then assembled according to a pre-established plan to insure that the interfaces between the components are properly designed and that the system meets its operational objectives.

An integrated design environment meeting the previous outlined requirements has been designed for electro-mechanical design and analysis. The development and operation of the software is divided into two separate stages. The first stage involves the writing of software to carry out the required tasks. The second stage involves the use of existing software such as PSPICE for circuit simulation and database dbase(iv) for data storage and manipulation. The following are software development areas:

- Identifying critical aids in the design process based on the user needs (i.e. tools and applications needed for design tasks).
- Software tools and functions to assist the drafting process.
- Software tools and functions to assist the 3D modelling creation.
- Software tools and functions to assist creating components libraries for both 2D and 3D construction.

- Software to allow the integration of appropriate applications into the system design framework (e.g. database applications of different tasks).
- Software to allow input of component data and other data attributes (i.e. design templates), for example, to allow editing component attributes for netlist generation, parameters for electro-magnetic design and modelling, and components general definition.
- Software to provide output data as part of design verification. Some of this data is to be used for further investigation.
- Software to read the output data and provide additional information for further design investigation and performances (e.g. generated schematic data used as input to generate netlist file format).

#### **2.4.1 Hardware Considerations**

MicroStation is a CAD software package from the Intergraph Corporation. It has a rich set of design tools, with many advanced productivity features, and it is available on different hardware platforms such as; PC, Mac; Sun SPARCStations; HP700, open architecture third party software. The system was developed using the Intergraph CAD package (i.e. MicroStation). To be able to run the system and its applications satisfactorily, it is necessary for the computer system to meet specific hardware and software requirements. These requirements are listed as follows:

- At least 8 MB RAM.
- A graphic input device (mouse or digitising tablet).
- A video monitor and display adapter capable of displaying graphics.
- A hard disk drive with at least 36 MB of free disk space.
- DOS (Disk Operating System) version 3.1 or above.
- The system can be run as windows based application (version 3.1 or above).

#### **2.4.2 System Design Software**

Most large industrial companies designed their CAD products from scratch [2.5]. As the request of new software applications has increased, the sophistication of the various CAD packages grew. In concert with this, a number of industry standards began to merge.

These came about as various engineering societies and committees began to wrestle with standardising the use of the growing number of functions provided.

Many CAD systems provide the user with one or more programming languages with which to define routines and procedures for processing information. These user-created procedures or programs, are intended to be used in all facets of the system's operation from helping facilities to create graphics from existing parametric data to engineering and manufacturing analysis of complete designs. MicroStation development language (MDL) is an example of such a programming language. It is a structured programming language which has a collection of standard commands, functions, and objects that allow to create an application programs. Figure 2.5, illustrates the general flowchart of the overall development procedure using MDL.

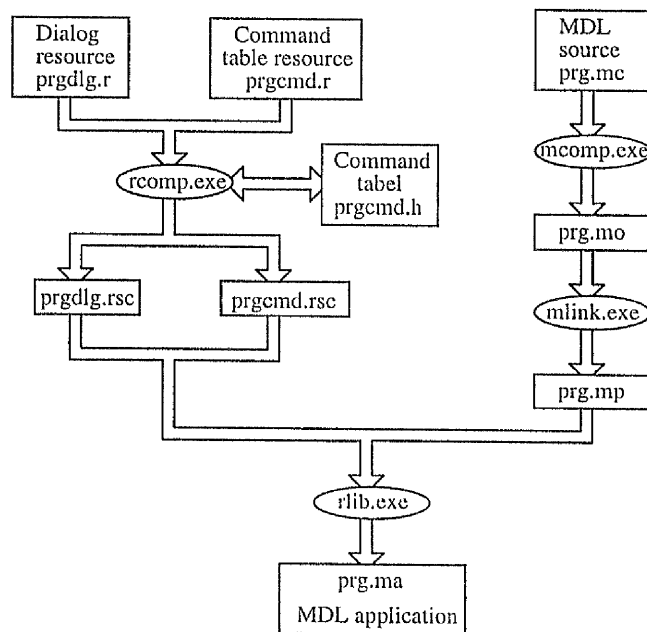


Figure 2.5 Applications development procedure [2.5].

The development of this system is based on the use of MDL and the macro procedures language user commands (UCMs). MDL is a complete development environment [2.5], which lets applications take full advantages of the MicroStation CAD engine. It is also used for a complete spectrum of customisation of MicroStation, ranging from simple utilities or customised commands to sophisticated commercial applications for markets. It is designed primarily for interactive applications and provides all tools needed to create applications that is consistent with MicroStation's graphical user interface.

At its core, MDL is first and foremost a C-like programming language. UCMs are a macro procedures allow to define composite operations in terms of commands that already exist in the system's platform. However, these are only of limited value because

they do not support many of the functions required in engineering computation and decisions making. Figure 2.6 illustrates different applications can be developed in both MDL and UCMs separately or are combined by means of built-in-function to form a single application as shown in Figure 2.7.

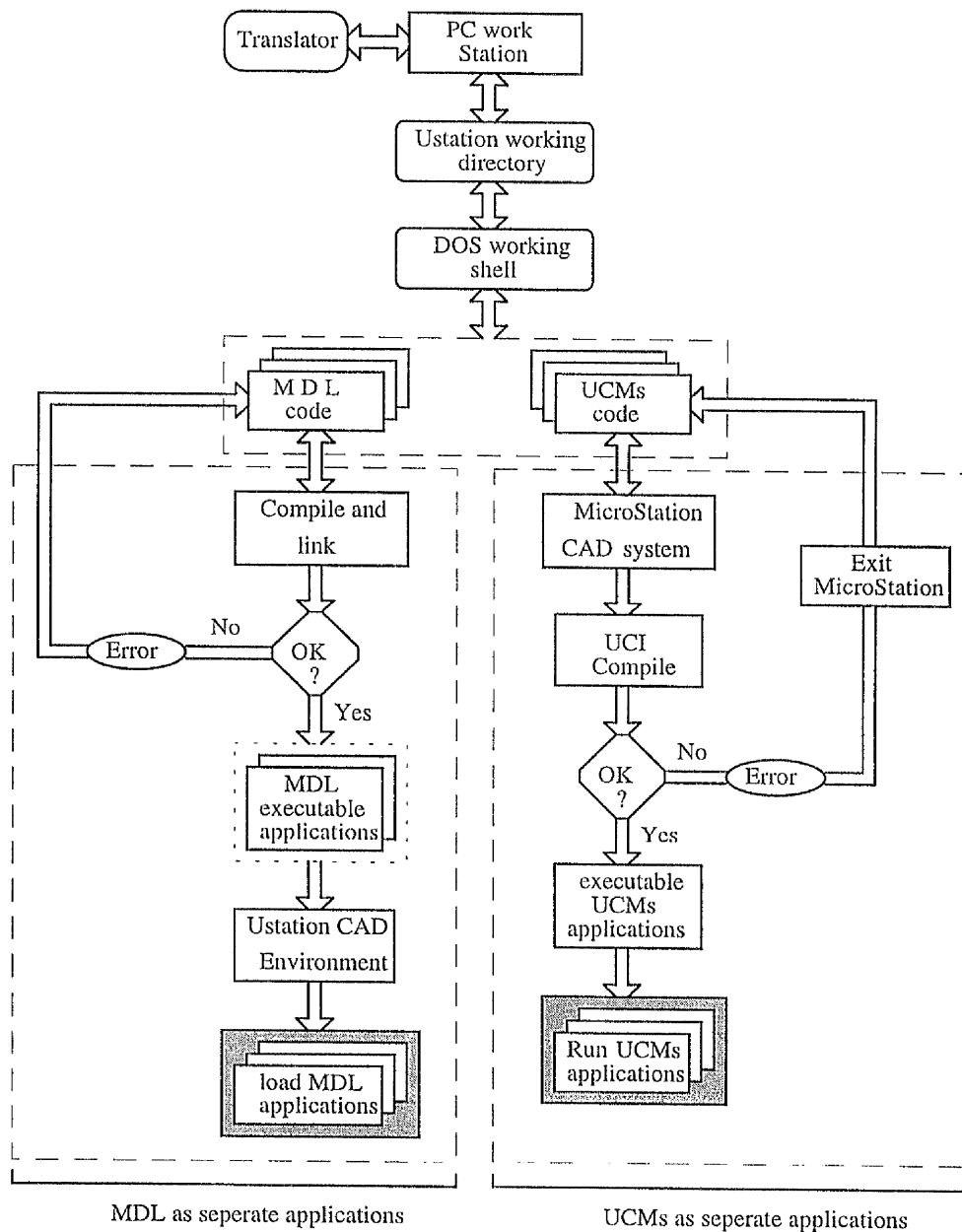


Figure 2.6 Programming of MDL and UCMs as separate applications

MicroStation contains a set of libraries routines [2.6], which allow very complex programs to be coded in MDL form and executed within MicroStation, allowing user-interface related data to be organised, so that applications developer can efficiently set up an application to translate for non-English speaking markets.



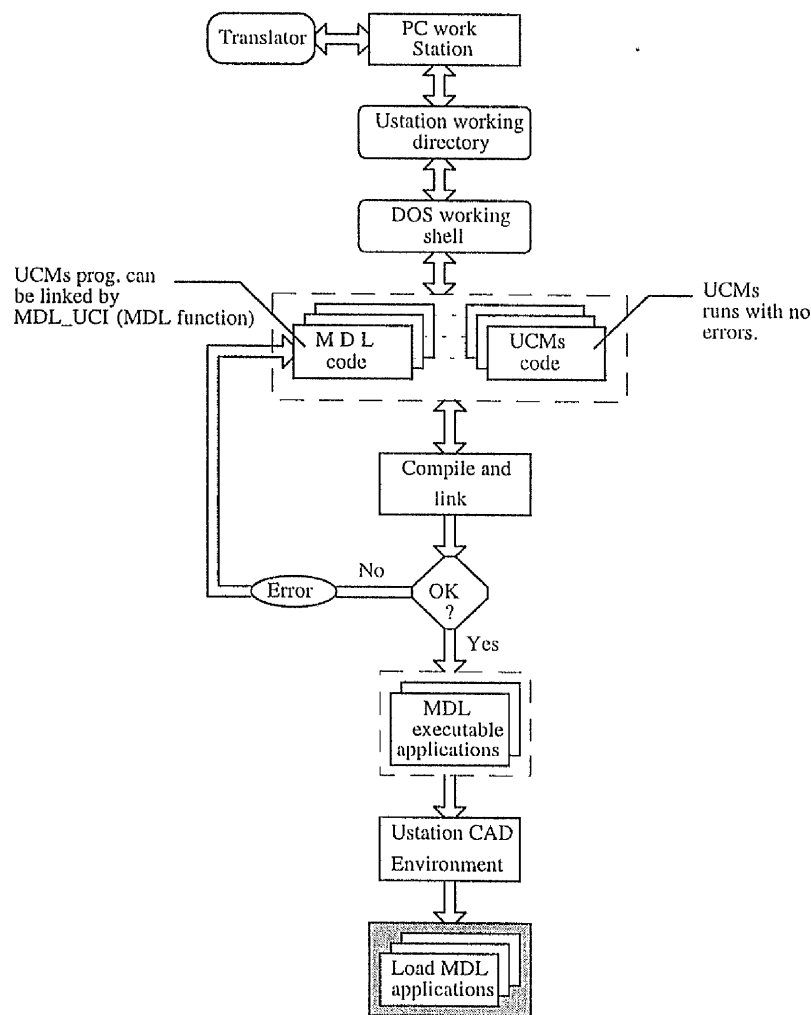


Figure 2.7 MDL and UCMs combined as single application.

### 2.4.3 System Functionality

The application of functionality approach towards information processing requires all possible functions to be defined at a level of sufficient generality to be universally applicable (i.e. to achieve an integrated system) [2.7]. Schematic place and routing system presented by Stephen et al. [2.8], for example, describes the functional term as relationship of a given set of modules. This relationship addressed at two levels: A logical level where related modules are grouped together, and physically level where the modules are mapped to their placement. The functional synthesis begins at the architecture level with a description of the overall behaviour of a large system and interaction with the environment [2.9], whereas synthesis proceeds, structural details are added in form of connected structures.

The definition of functions is one of the steps needed in a methodological approach to system design. Therefore, the next step in the development of tools and techniques which

the system designer can use to apply the functional approach to specific problems. The development of these tools could include (models, simulation, graphics, databases, etc.). The design method required consists of defining systems functions at a very generic level and implementing those functions so that they can be used in a variety of ways. Figure 2.8 illustrates the system design functional levels, which form the full set required for information-processing.

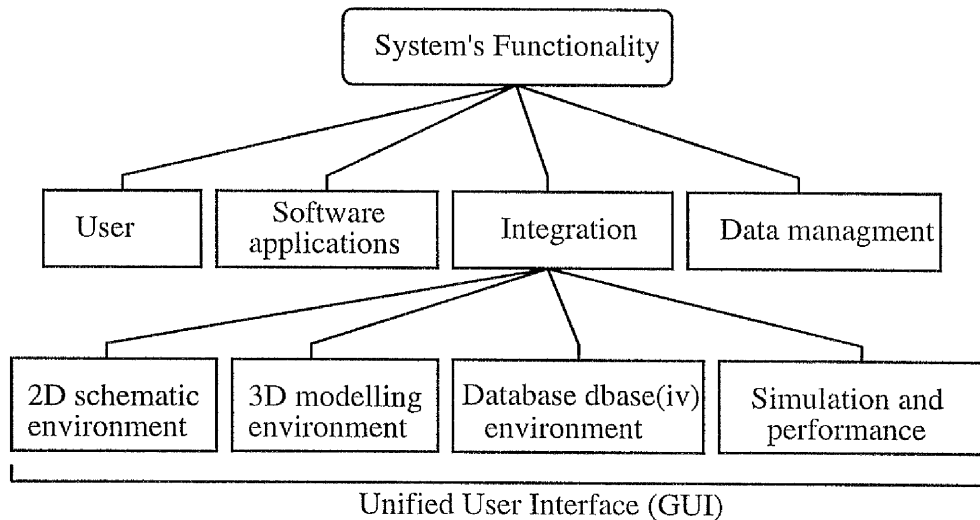


Figure 2.8 Concept of system functionality levels.

#### 2.4.4 System Integration

Integrated applications are required that can be accessed for a single workstation and used by an individual development engineer for drafting, modelling, simulation, and various other activities [2.10]. The common language between these applications is the design data. It provides a means of associating different environments that need to be integrated. This is especially useful when integrating multi-tools CAE systems, where the information stored within a common database provides the input to other applications, thus reducing the need for user input.

The design system being developed, is an integrated design environment composed of different design levels supported by various tools. The data requirements of analysis applications are fulfilled by specialised tools that extract and transform data among other environments. Figure 2.9, illustrates the universal concept of the system integration. As this figure shows, the different environments interact with each other through a common database to form a unified design environment. The graphical editors and supporting software applications form a design support system that facilitates these levels and provides the evaluation of electro-mechanical designs by means of data sharing and data exchange.

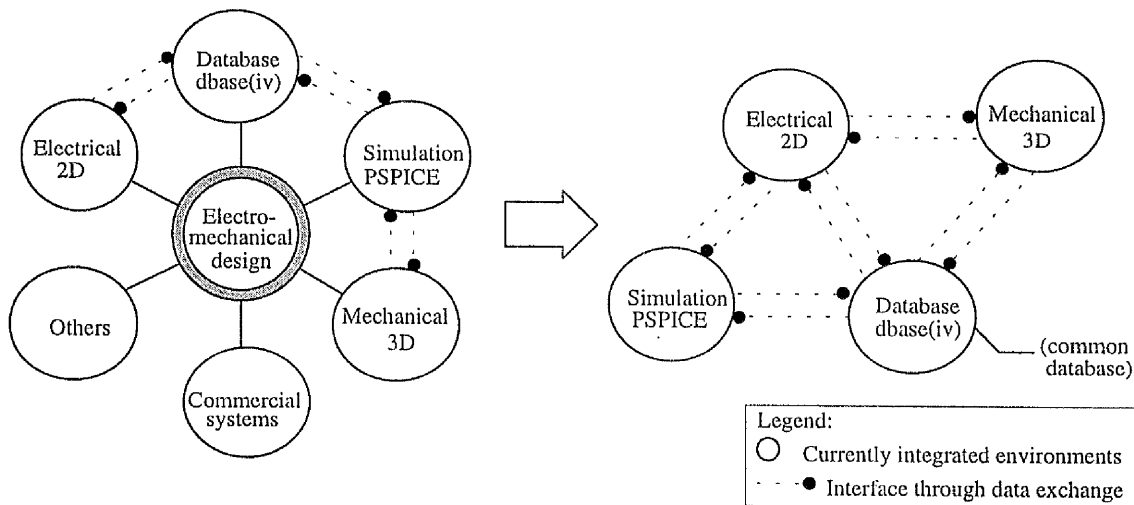


Figure 2.9 The system integrated environments.

The scope of the integration approach is to capture the breadth of the electro-mechanical design domain as an integrated environment. This allows the determination of feasibility of integrating 2D schematics and 3D modelling layouts, offering data exchange in design rather like PCB's design and layout. In addition, it allows to determine the feasibility of components modelling and design procedures to establish a framework by which an integrated design environment can be performed. Figure 2.10, illustrates design flow levels proceed to form total integration.

### 2.4.5 Design Flow

In general, the design flow is considered as a collection of activities for the accomplishment of tasks. Each sequence of activities has to be performed in accordance with the design process. The design flow is composed of several levels, as shown in Figure 2.11. At each level the designer tends to select tools and functions to assist various design tasks. The following are detailed descriptions to each level.

#### 2D electrical schematic level

The first segment of the schematic capture process is schematics preparation using the provided drafting tools. In order to trace the schematics topology, elements, inputs and outputs are labelled using text editor and placement tools. These elements are stored in the database and can be annotated as part of schematic construction. Additional text associated with each component describing output type, name, simulation model, manufacturer is also included.

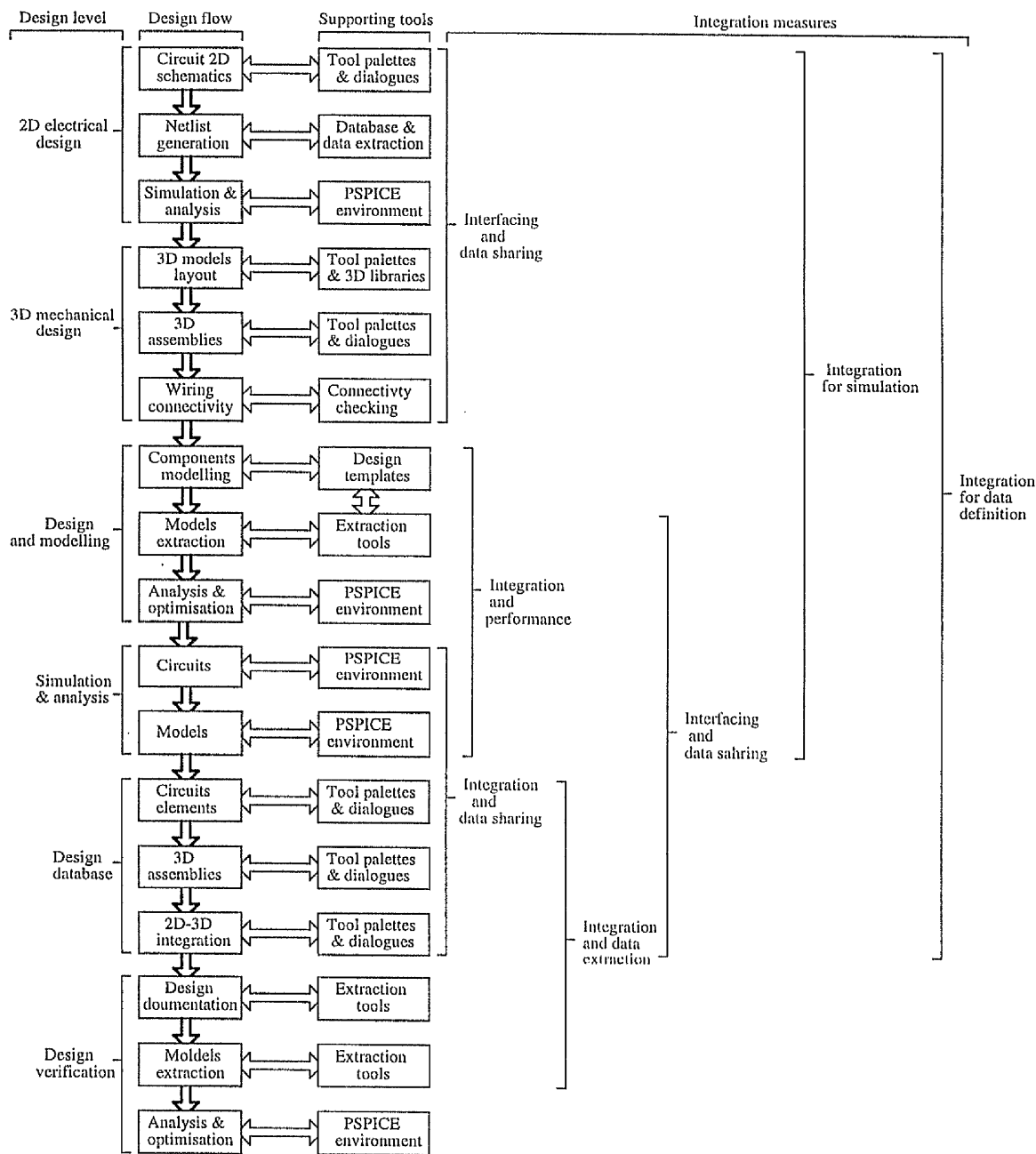


Figure 2.10 Design flow proceeds towards complete integration.

### 3D Modelling level

This level used to construct the 3D modelling layout for electro-mechanical design, using the provided tools. From design flow shown in Figure 2.11, it is shown that the need to integrate 2D and 3D data is essential. This allows additional tasks to be performed (e.g. components can be modelled in 3D form and results are annotated into 2D form). Additional tasks can be performed such as 3D rendering operations, components placement, and wiring construction.

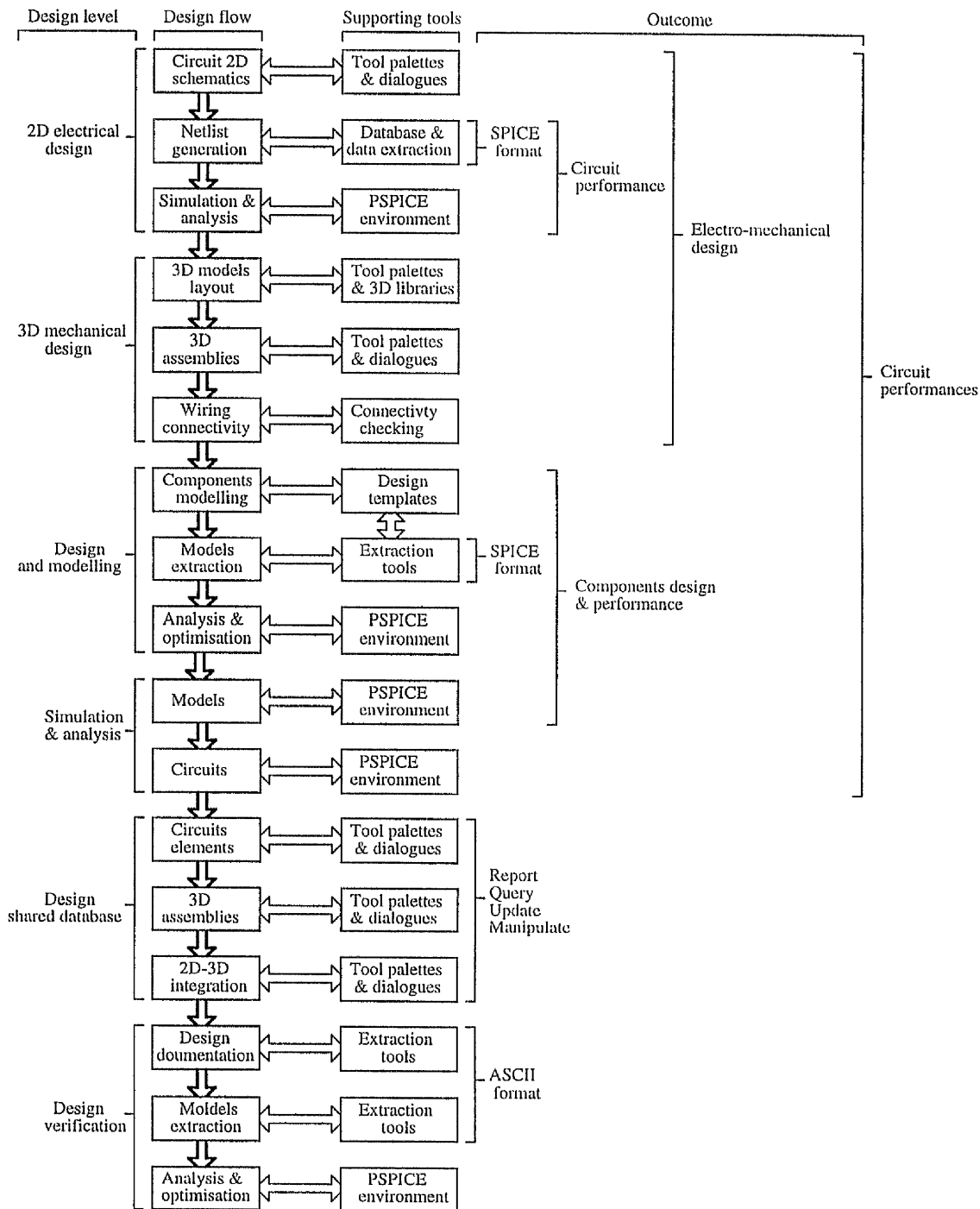


Figure 2.11 Design flow and supporting tools.

### Design and Modelling level

At this level, the principle feature is the ease with which users can create models. Models for electro-magnetic components are created using parametric design technique by means of software programs linked to design templates. These programs are activated by selecting a component within 2D or 3D environments. Based on circuit input

parameters, these models are built giving an output format as a result. This format is used with PSPICE for components simulation and parameter performances. In general, the modelling procedure is an iterative process that requires a number of repeated entries to reach an optimised solutions (refer to Chapter 3).

### **Simulation and analysis level**

Simulation, is the operation required for testing electrical/electronic circuit designs and modelling performances using different analysis techniques. It is one of the most critical and time consuming tasks performed on a design before the manufacturing stage. This level involves the simulation of the extracted models and circuits netlists. From the design flow shown in Figure 2.11, it can be seen that the modelling level is directly linked to the simulation level. This makes it possible to simulate the modelling results iteratively. They are simulated and analysed using especial formats that can be recognised by PSPICE package.

### **Design database level**

A Database is used to manage schematics drawings by including different tables containing the important attributes of components, wires, and models. Software applications are developed to assist this task. It create records for each item in the schematic design without interrupting the drafting process. In addition, the database used to hold the product design information which can be subsequently used by other design environments. Design data produced by the other design levels are stored in organised tables and used for drawing annotation, reports generation including wiring list, bill of materials, circuit drawing information, and user-defined reports (e.g. fence report). Structural query language (SQL) is also used to locate and retrieve design information and different design queries.

### **Design verification level**

Design verification involves three steps, viz: document generation, models extraction, and simulation and analysis. The first is performed by capturing the design information using database applications and other extraction tools (refer to Section 2.5). The second is concerning the extraction of components models performed using design templates. The third is performed using PSPICE environment, where simulation of the extracted data will be performed.

With reference to design flow previously described, data management tools have been designed that cover co-operative work between environments, their associated tools and document manipulation. Every element or component can be defined using graphical

dialog editors. By the inclusion of elements and components identities in drawings, different design information are automatically generated. This information are organised using file extensions to help recognise the type of extracted formats. Figure 2.12 shows some of extracted data and their file extensions.

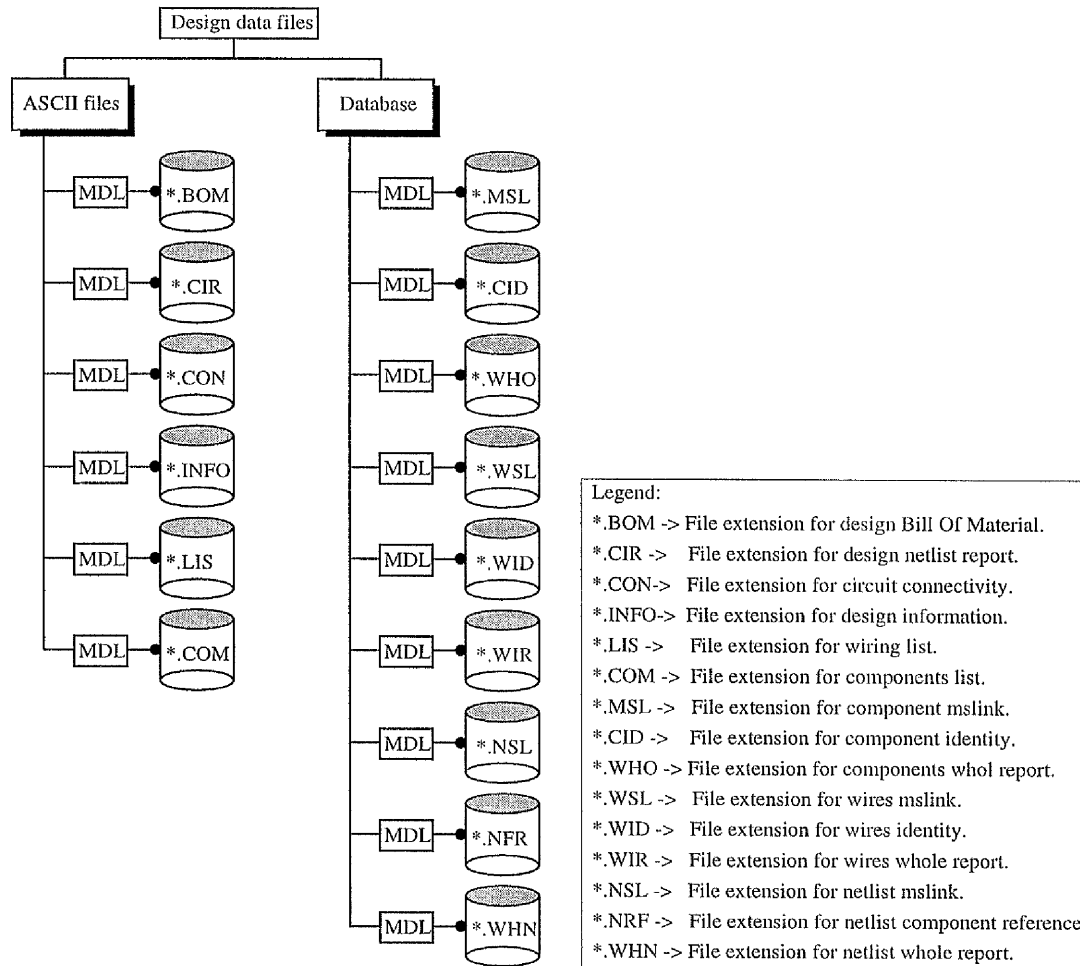


Figure 2.12 Extracted data and files extensions.

## 2.4.6 Design Data Management

As previously mentioned, MicroStation provides the database interface module and SQL support to enable to access and manipulate database information. Databases applications can be developed to perform these tasks. However, there are many formats of data accessed during the design process, therefore, design information should be linked to achieve simultaneous data manipulation. A key issue is the communication between tools and all the different database that are necessary for design and performances. Data management organisation is therefore an important issue so that design data are properly managed. Figure 2.13 shows the overall architecture of data management design.

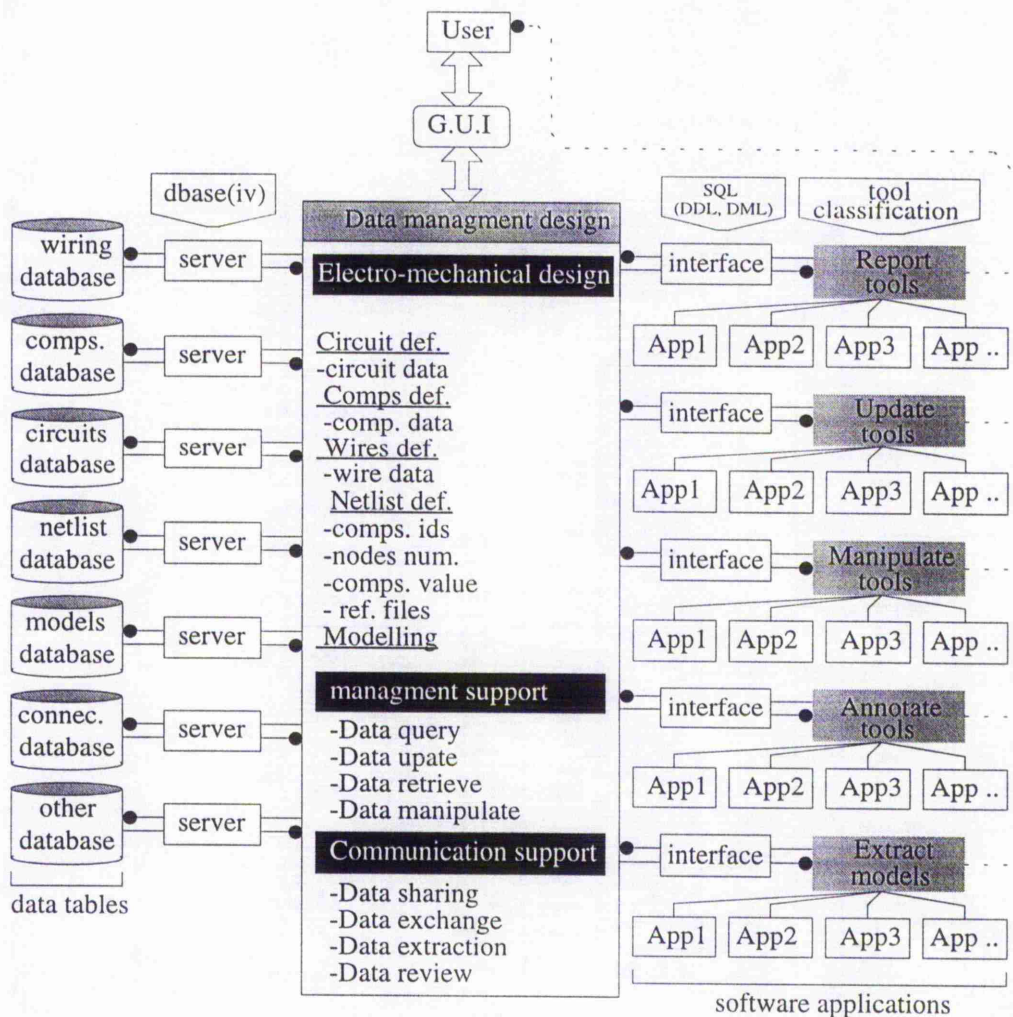


Figure 2.13 Overall architecture of data management design.

## 2.5 SPECIFIC SOFTWARE REQUIREMENTS

Software must support both functional and physical design. Functional design includes synthesis, simulation, and testing. Physical design aids support layout, and topological analysis at all design levels. Functionality, testability, and physical design must be considered in parallel throughout the design process. Therefore, it is desirable for an integrated system to support various levels of design process using a unified GUI and different tools.

To achieve these goals, the interaction between individual programs, databases, and designers must be carried through a single, consistent GUI. This is to support monitoring the progress of a design, supply options at any stage in the design process, and assist in design documentation and modification. In addition, specific requirements for software and interfaces are outlined later in this section. The tasks that have been automated in CAD systems up to the present day have been those which computers have



been considered to do well (e.g. large, complex calculations), as reported by Vivienne [2.11].

- Problems that combine the manipulation of large amounts of data with the calculation of complex functions (e.g. simulation and circuit analysis).
- Problems of conversion from one form of representation to another (e.g. logic diagrams to layout).
- Problems that involve checking routines (e.g. design rule checking , layout verification, wiring list connection, connectivity between parts)

### 2.5.1 Physical Design Aids

Physical design tools are necessary to aid construction of specific designs. Their role is to place, modify, ensure connectivity between circuit components, and also to generate design verification to ensure a design has been performed properly and as desired. In order to provide an optimum system, the physical issue (i.e. layout, schematics capture, database interaction) must be considered in parallel with functional design (i.e. synthesis, simulation, analysis, and optimisation). Figure 2.14, illustrates the different tools designed and developed for this purpose.

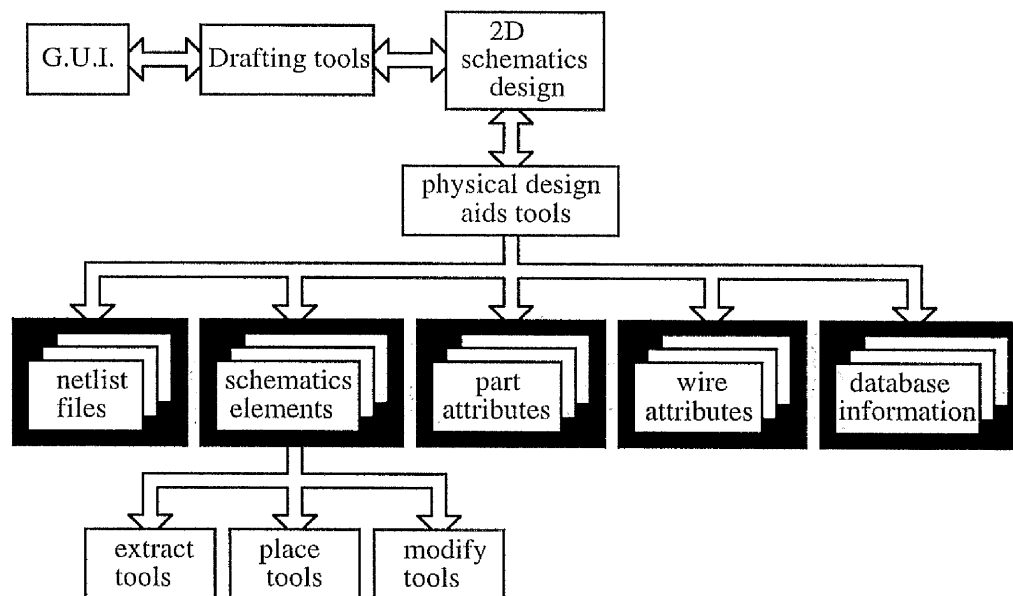


Figure 2.14 Design of circuit design tools.

### 2.5.2 Design Connectivity and Testability

Testing is the process of verifying that the system is performing as desired. To ensure that a circuit can be tested, design for testability techniques must be used throughout the

design process. According to Marvin et al. [2.12], design for testability fall into two major categories: testable design style and testability by measuring analysis. This information is used in the design process to locate potential circuit testing problems and provide feedback about the effect of circuit modifications on testability.

Testing a circuit design basically depends upon the components connectivity. It can be used to reconstruct a detailed circuit that can be used in performing an accurate functional and timing analysis. For this purpose, software application has been developed for connectivity checking between components. It is based on the use of circuit scanning technique, involving both wires and connected components. Once the connection between components and wires is established, the procedure starts by comparing both the components and wires co-ordinates. Detailed description of the program is shown in Figure 2.15 (see page 46), while Table [2.1] (see page 48) illustrates the software utilities, functions and their operations used in developing this application.

### **2.5.3 Integration With Standard Database**

The necessity of using Database is to support the design [2.13]. Therefore, an efficient database is vital for co-ordinating the various functions. As systems become more complex, it is desirable that an overall database be established and maintained to ensure consistency in design and eliminate duplication of information.

A CAD database may contain several types of descriptive information that can be used by a design system [2.14]. These include component geometry, functional properties and technical specification. Several software packages are available that can access CAD data and translate it into a frame representation. Therefore, each component is represented as a single frame at a level of data appropriate to the problem area. Data would then be automatically available for processing for routine and variable design problems.

The system uses database dbase(iv) to store and retrieve different design information. For this purpose, software applications are developed for entering components and wires data within electrical schematics. Components and wires data can be entered, updated and reviewed at any stage during the drawing process and reports generated for complete, partial circuits, single component or wire as defined by fence or user cursor selection. Figure 2.16, shows the design of circuit information data flow and the use of database dbase(iv).

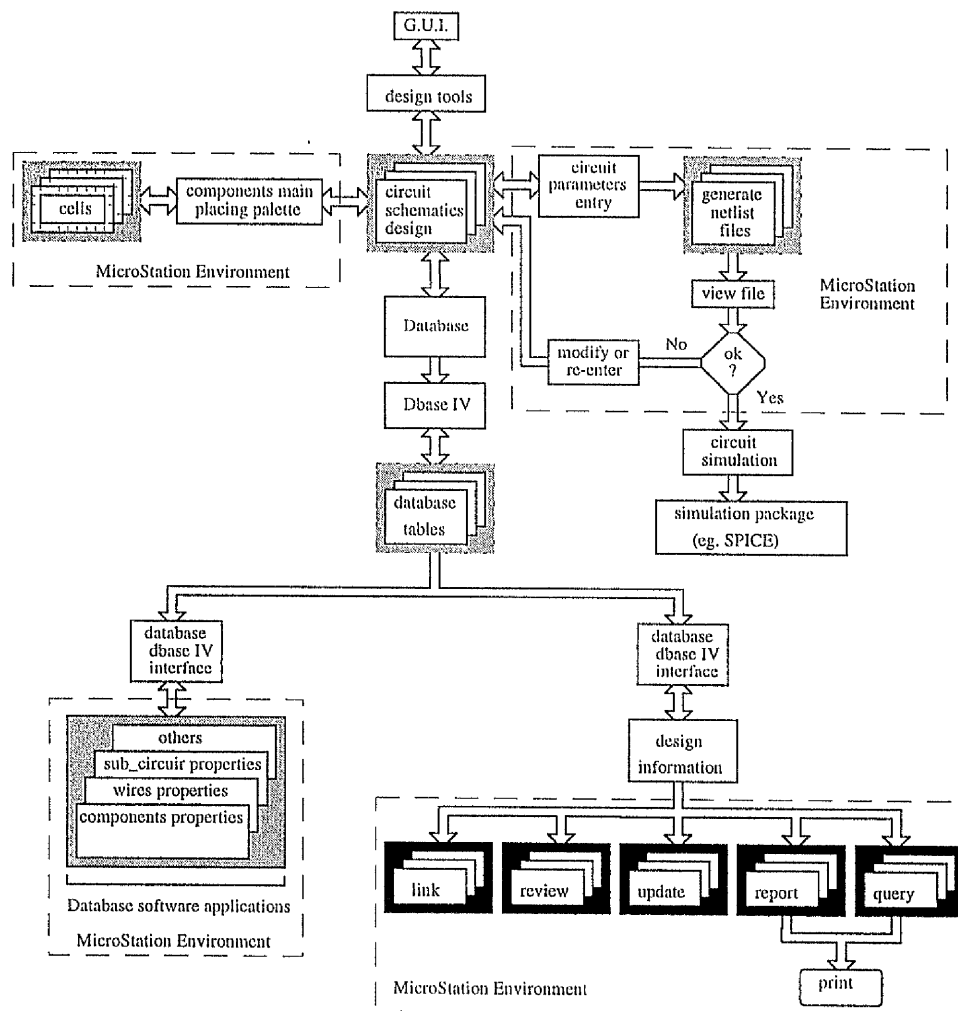


Figure 2.16 Circuit information flow and database linkage.

The CAD system MicroStation provides support for several major database packages that run on several different computing platforms [2.13]. The ability to link design elements in the MicroStation environment to a standard database application provides the designer with a means to produce detailed reports and material take-offs, and also the ability to display components on the basis of defined filter criteria [2.15].

This section focuses on the database side of MicroStation and the link between the two packages. Later in Chapter 5 detailed explanation of database interface and other applications development is discussed.

## 2.5.4 Circuit Documentation and Data Extraction

Data extraction is the process of extracting circuit information from a circuit schematic capture; using dedicated CAD designed tools. In general most CAD/CAE applications operate independently, each having their own optimised data representation structure

[2.16]. Data is captured relevant to the specific applications needs. It is quite common for the output files from one applications to be required as part of the input to another (e.g. extracted netlist file for PSPICE).

Larry [2.16] has suggested, design documents are the preferred and most effective way to communicate information in every engineering discipline. Documents whether they be hard copy or electronic are the evidence that engineering tasks have been performed. Automating document generation is frequently seen as a matter of facilitating information exchange between software development tools and document publishing systems. The automated document generators must be flexible enough to extract information from design environment. Thus, it must have knowledge of many methods and development tools. The structure of Figure 2.17, shows the document generator developed in [2.16]. Using the knowledge sets, the automated document generators maps the data supplied by various development tools. The automated generator then extracts the appropriate data, formats it, and inserts in into the document, thus instantiating the place holders with the information to a particular method combination.

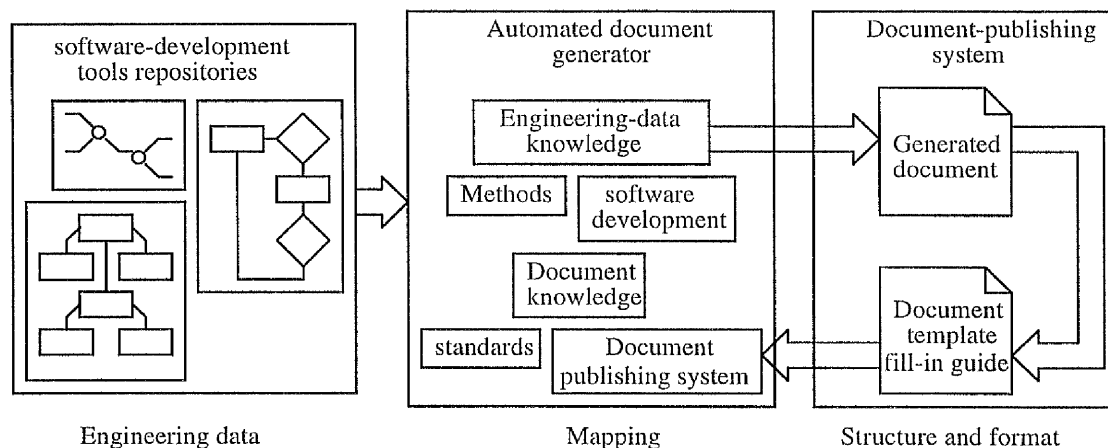


Figure 2.17 Flow of information in an automated document generator [2.16].

Generating design documentation is useful for reviewing and modification. Once the circuit design has been completed, the design can be documented by collecting information as an ASCII format files either using database dbase(iv) applications or software written programs. Figure 2.18 illustrates the design general concept of automation functions and data extraction tools.

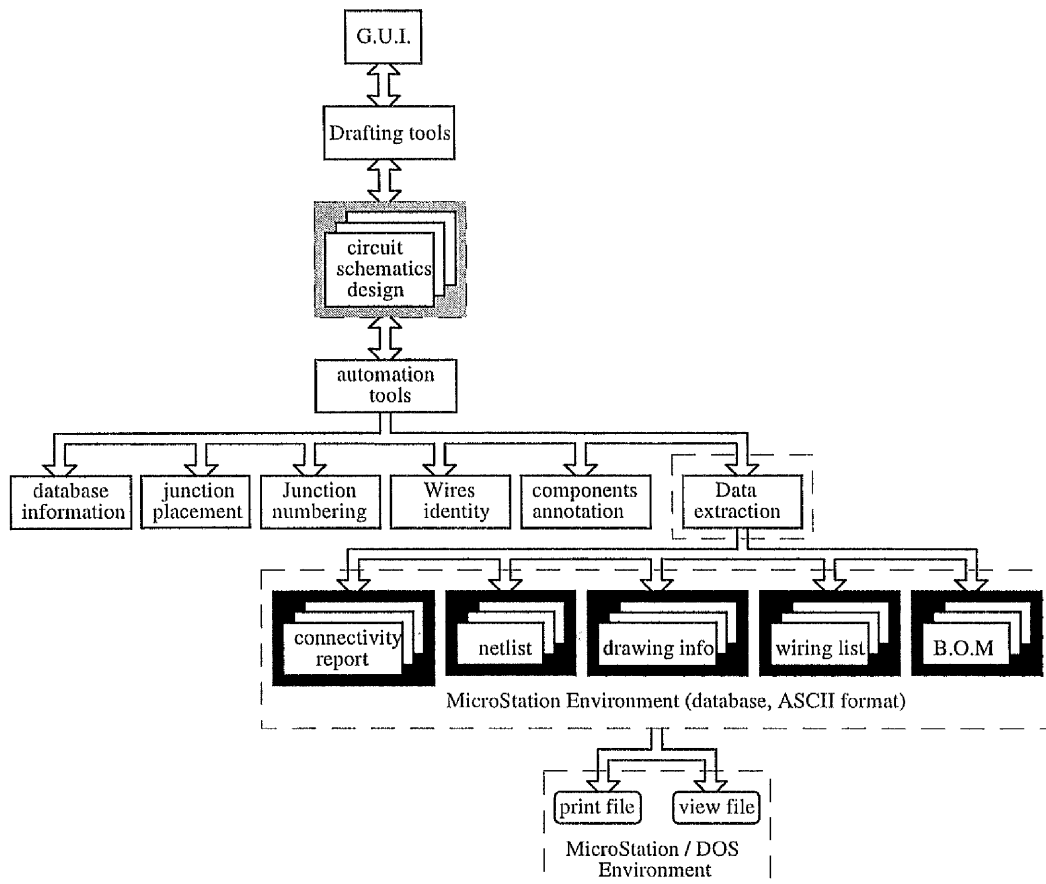


Figure 2.18 Circuit automation tools and data extraction.

Data retrieval is one of the most important nongraphical capabilities for electrical/electronics design. As a component is placed into schematics drawings, its physical properties are defined, along with the attributes of other various components. This data can be used for nongraphical needs, for example, reference designation, component value helps to generate bill of material (B.O.M) to support later modifications. Types of data that can be extracted using the developed tools are described below accompanied with their programming flowcharts, and utilities tables.

### Automatic Bill Of Material Extraction

An automated design tool has been developed to extract the design part-lists. It consists of item number, quantity, reference identity and part value. Figure 2.19 (see page 47) shows the application's program flowchart, while Table [2.2] (see page 48) illustrates software utilities, functions and their operations used in developing this application.

### Automatic Wiring list Extraction

Data can also be generated along with automatic wiring identity. The extraction procedure is based on the use of MDL scanning functions. Figure 2.20 (see page 49) shows the

program flowchart for wiring list extraction, while Table [2.3] (see page 52) illustrates software utilities, functions and their operations used in developing this application.

### **Automatic Drawing Information Extraction**

It has been stressed that the main representation of a design is to generate further data representations for assessment and manufacturing information. Drawings are easily misread either because of ambiguity or error in the drawing or simply of misinterpretation. For example, schematic drawings can have a unique information attached that can be captured. It helps the designer refer to circuit components and wires by means of reference, value, level, weight, style and colour so that it can easily be modified.

Figures 2.21a, 2.21b (see page 50, 51), show the program flowchart for extracting drawing information, while Table [2.4] illustrates, software utilities, functions and their operations used in developing this application.

### **Automatic Placement**

Different tools have been implemented to facilitate elements placements. Such tools involve automatic placement of junction by simply selecting a wire, automatic wires identity placement, automatic numbering of junctions. Figures (2.22-2.24) show programs flowcharts of these tools. The performance of these tools is demonstrated as part of a case study and can be found in Chapter 4.

## Chapter 2: System Design and Development

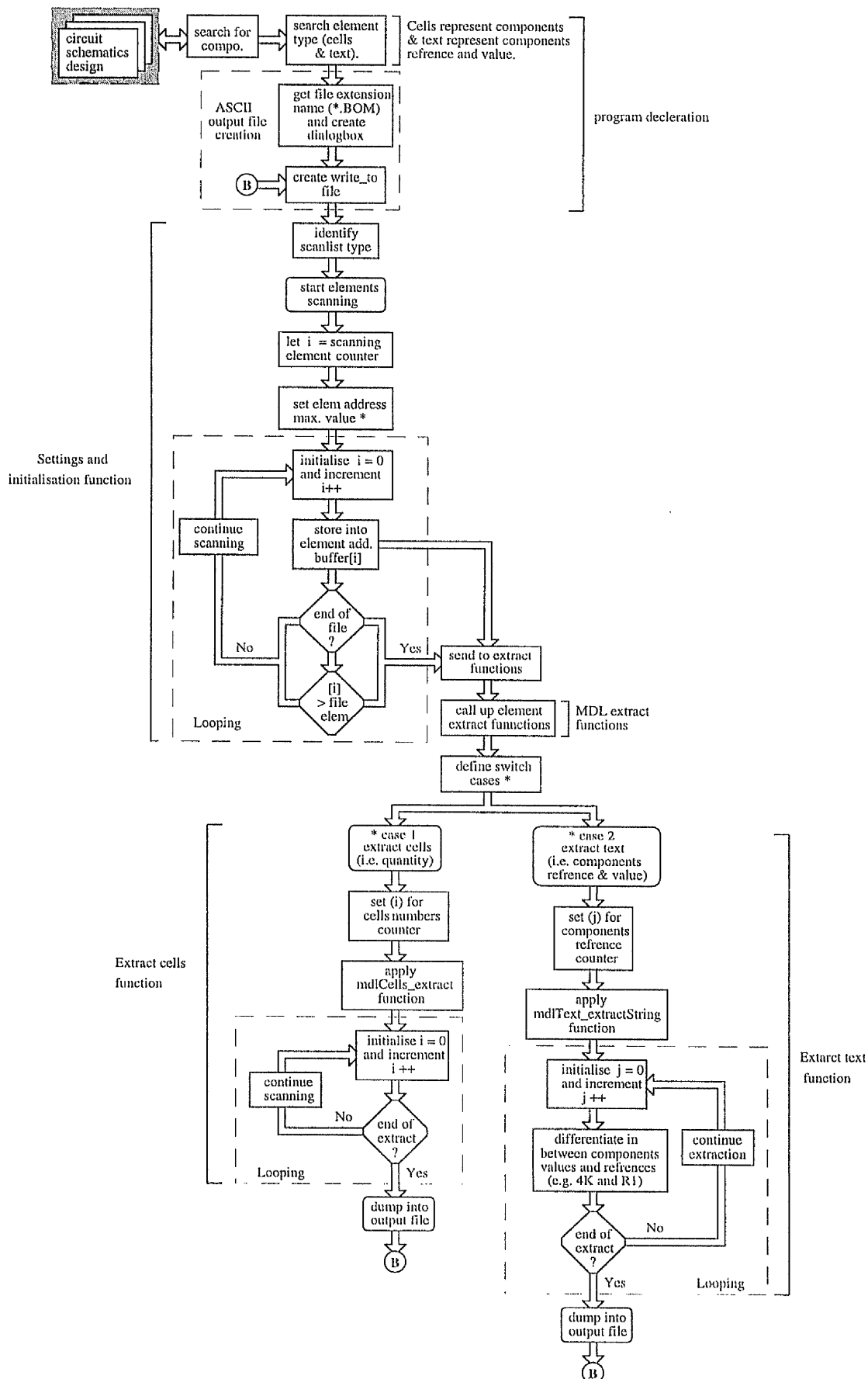


Figure 2.15 Flowchart of circuit connectivity program.

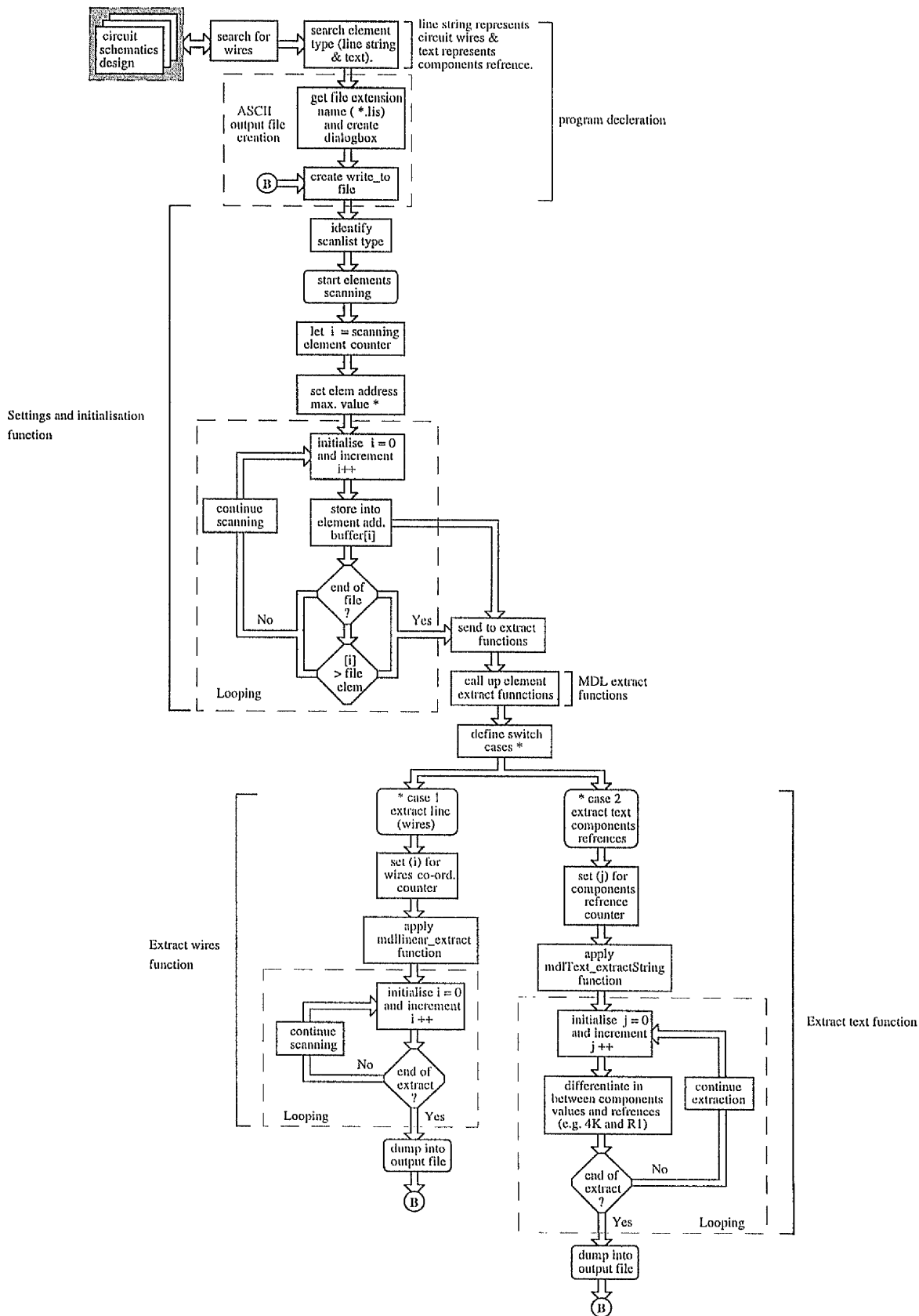


Figure 2.19 Flowchart of circuit B.O.M. extraction program.



## Chapter 2: System Design and Development

Table [2.1]. Connectivity testing utilities and its development functions.

Utility	MDL_Function	Operation
Identify element type	searchType[] (CELL_HEADER_ELM) (LINE_STRING_ELM) (ELLIPSE_ELM)	search for wires and components
Create dump data file	mdlFile_getcwd	retrieve name of current working directory.
State the derivation of file name to be parsed.	mdlFile_parsename	drive the components of file name
Create built-in dialog box for files list	mdlDialog_fileCreate	create file name extension "*.con" and dialog box title
get file name based on the user given name	mdlFile_find	find a file based on an incomplete file name specification
Open the file to write_to	fopen (filename, "w")	direct write to file
Indicate an error if a file can not be opened	mdlDialog_openAlert	open an alert dialog box
Add sheet top information	fprintf ("dump file", "")	write into output file
Display of complete extraction	mdlOutput_message	display an output message once extraction is completed
Initialize a scanning list	mdlScan_initScanlist	set scanning procedure from the beginning of the file
Load up scanList into the microStation design file scanner	mdlScan_initialize	set the scanning list so it will return displayable elements
Locate element type on the design file	scanList.typemask[1] (TMSK0_LINE_STRING) (TMSK1_CELL_HEADER)	establish a type mask for element location
Extract array of coordinates from line, linestring	mdlLinear_extract	establish wires coordinates and number off
Extract arc or ellipse element	mdlArc_extract	establish component's pins coordinates

Table [2.2]. Bill Of Material utilities and its development functions.

Utility	MDL_Function	Operation
Identify element type	searchType[] (TEXT_ELM)	search for components reference and value
Create dump data file	mdlFile_getcwd	retrieve name of current working directory.
State the derivation of file name to be parsed.	mdlFile_parsename	drive the components of file name
Create built-in dialog box for files list	mdlDialog_fileCreate	create file name filter (*.bom) and dialog box title
get file name based on the user given name	mdlFile_find	find a file based on an incomplete file name specification
Open the file to write_to	fopen (filename, "w")	direct write to file
Indicate an error if a file can not be opened	mdlDialog_openAlert	open an alert dialog box
Add sheet top information	fprintf ("dump file", "")	write into output file
Display of complete extraction	mdlOutput_message	display an output message once extraction is completed
Initialize a scanning list	mdlScan_initScanlist	set scanning procedure from the beginning of the file
Load up scanList into the microStation design file scanner	mdlScan_initialize	set the scanning list so it will return displayable elements
Locate element type on the design file	scanList.typemask[1] (TMSK1_TEXT)	establish a type mask for element location
Extract array of coordinates from line, linestring	mdlLinear_extract	establish text extraction and string comparison (e.g. reference and value)

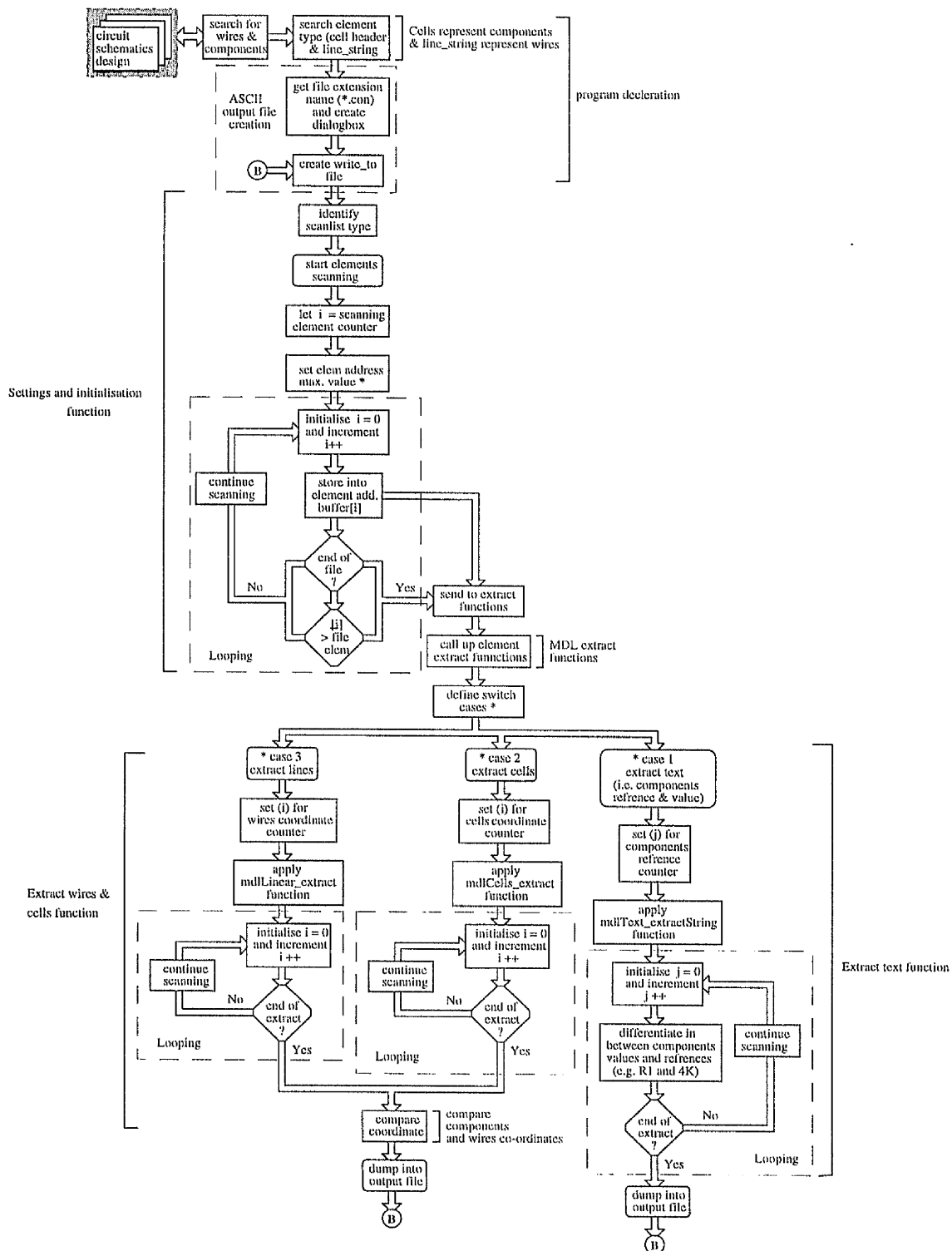


Figure 2.20 Flowchart of wiring list extraction program.

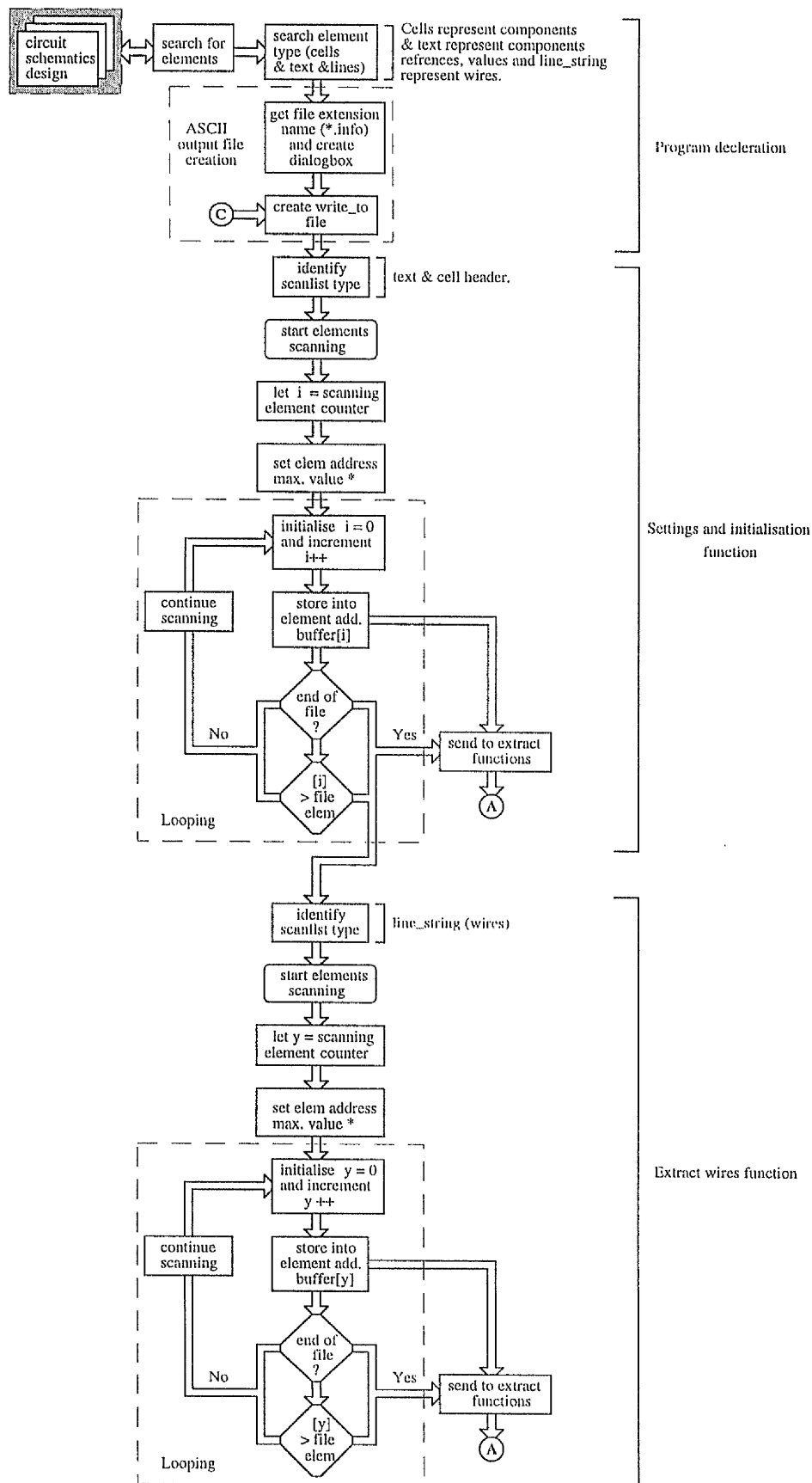


Figure 2.21a Flowchart of drawing information extraction program part (1).

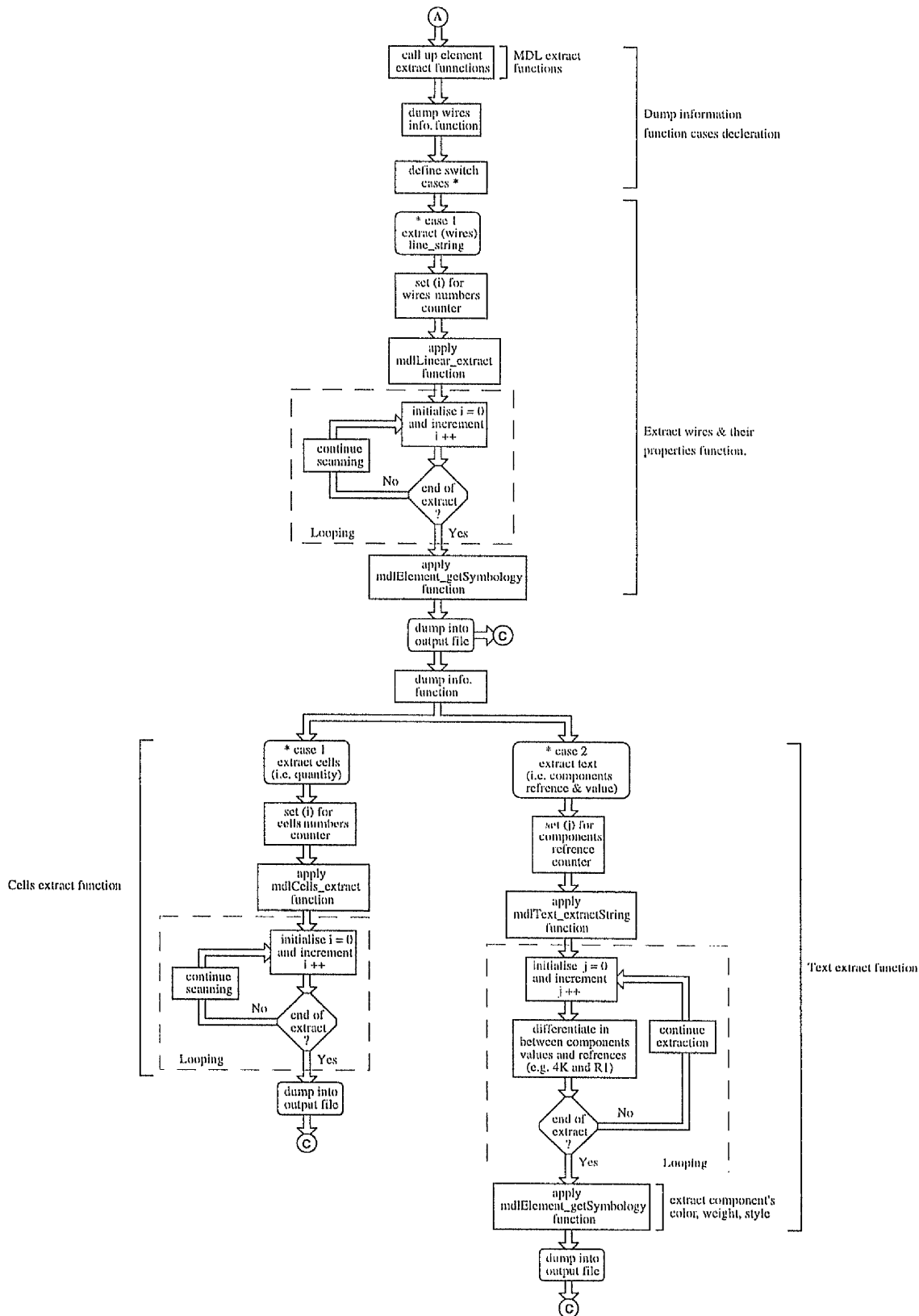


Figure 2.21b Flowchart of drawing information extraction program part (2).

## Chapter 2: System Design and Development

Table [2.3]. Wiring list utilities and its development functions.

Utility	MDL_Function	Operation
Identify element type	searchType[] (CELL_HEADER_ELM) (LINE_STRING_ELM) (ELLIPSE_ELM)	search for components reference, value and coordinates
Create dump data file	mdlFile_getcwd	retrieve name of current working directory.
State the derivation of file name to be parsed.	mdlFile_parsename	drive the components of file name
Create built-in dialog box for files list	mdlDialog_fileCreate	create file name filter (*.list) and dialog box title
get file name based on the user given name	mdlFile_find	find a file based on an incomplete file name specification
Open the file to write_to	fopen (filename, "w")	direct write to file
Indicate an error if a file can not be opened	mdlDialog_openAlert	open an alert dialog box
Add sheet top information	fprintf ("dump file", "")	write into output file
Display of complete extraction	mdlOutput_message	display an output message once extraction is completed
Initialize a scanning list	mdlScan_initScanlist	set scanning procedure from the beginning of the file
Load up scanList into the microStation design file scanner	mdlScan_initialize	set the scanning list so it will return displayable elements
Locate element type on the design file	scanList.typemask[1] (TMSK0_LINE_STRING) (TMSK1_CELL_HEADER) (TMSK1_TEXT)	establish a type mask for element location
Extract array of coordinates from line, linestring	mdlLinear_extract	establish wires coordinates and number off
Extract arc or ellipse element	mdlArc_extract	establish component's pins coordinates
Extract character string from text element	mdlText_extractString	establish text extraction and string comparison (e.g. reference and value)

Table [2.4]. Drawing information utilities and its development functions.

Utility	MDL_Function	Operation
Identify element type	searchType[] (CELL_HEADER_ELM) (LINE_STRING_ELM) (TEXT_ELM)	search for wires and components reference, value, properties
Create dump data file	mdlFile_getcwd	retrieve name of current working directory.
State the derivation of file name to be parsed.	mdlFile_parsename	drive the components of file name
Create built-in dialog box for files list	mdlDialog_fileCreate	create file name filter (*.info) and dialog box title
get file name based on the user given name	mdlFile_find	find a file based on an incomplete file name specification
Open the file to write_to	fopen (filename, "w")	direct write to file
Indicate an error if a file can not be opened	mdlDialog_openAlert	open an alert dialog box
Add sheet top information	fprintf ("dump file", "")	write into output file
Display of complete extraction	mdlOutput_message	display an output message once extraction is completed
Initialize a scanning list	mdlScan_initScanlist	set scanning procedure from the beginning of the file
Load up scanList into the microStation design file scanner	mdlScan_initialize	set the scanning list so it will return displayable elements
Locate element type on the design file	scanList.typemask[1] (TMSK0_LINE_STRING) (TMSK1_CELL_HEADER) (TMSK1_TEXT)	establish a type mask for element location
Extract array of coordinates from line, linestring	mdlLinear_extract	establish wires coordinates and number off extraction
Return element's display symbology	mdlElement_getSymbology	establish wires, components drawing information (e.g. style weight, colour, level, etc.)

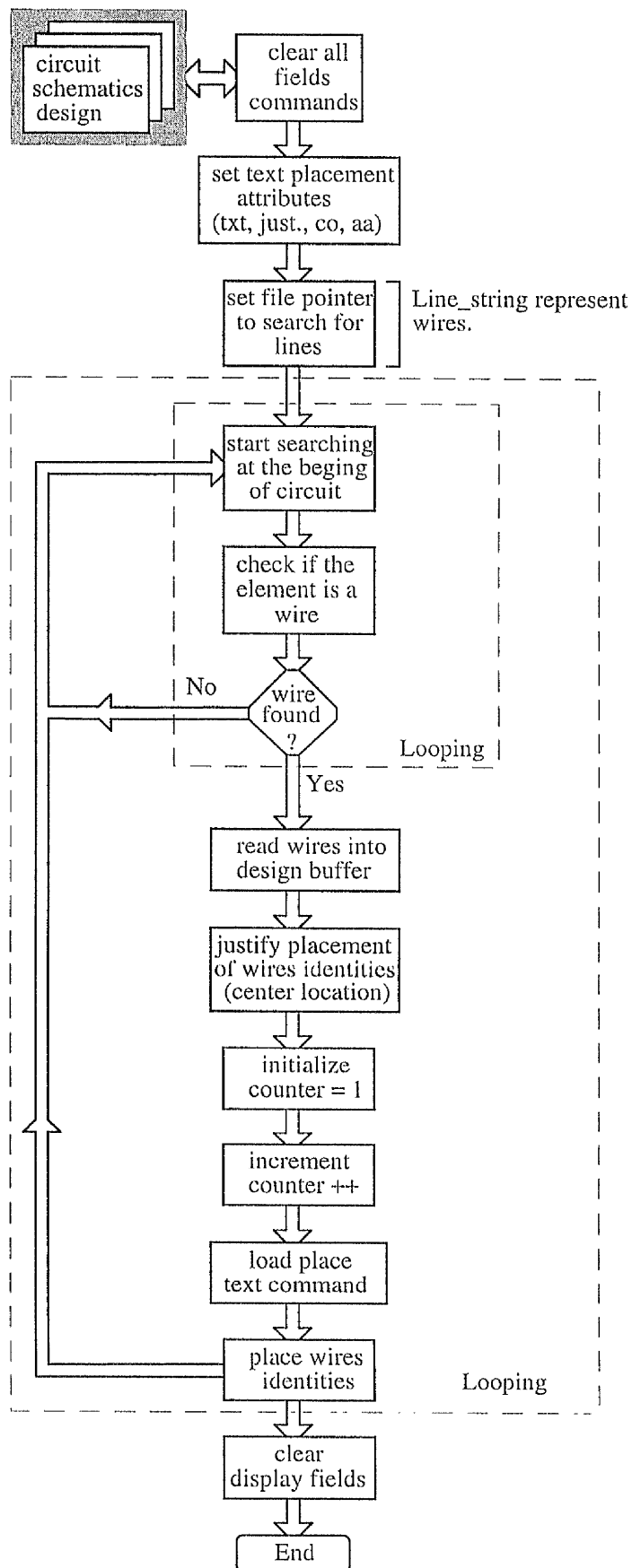


Figure 2.22 Flowchart of automatic wires identity placement program.

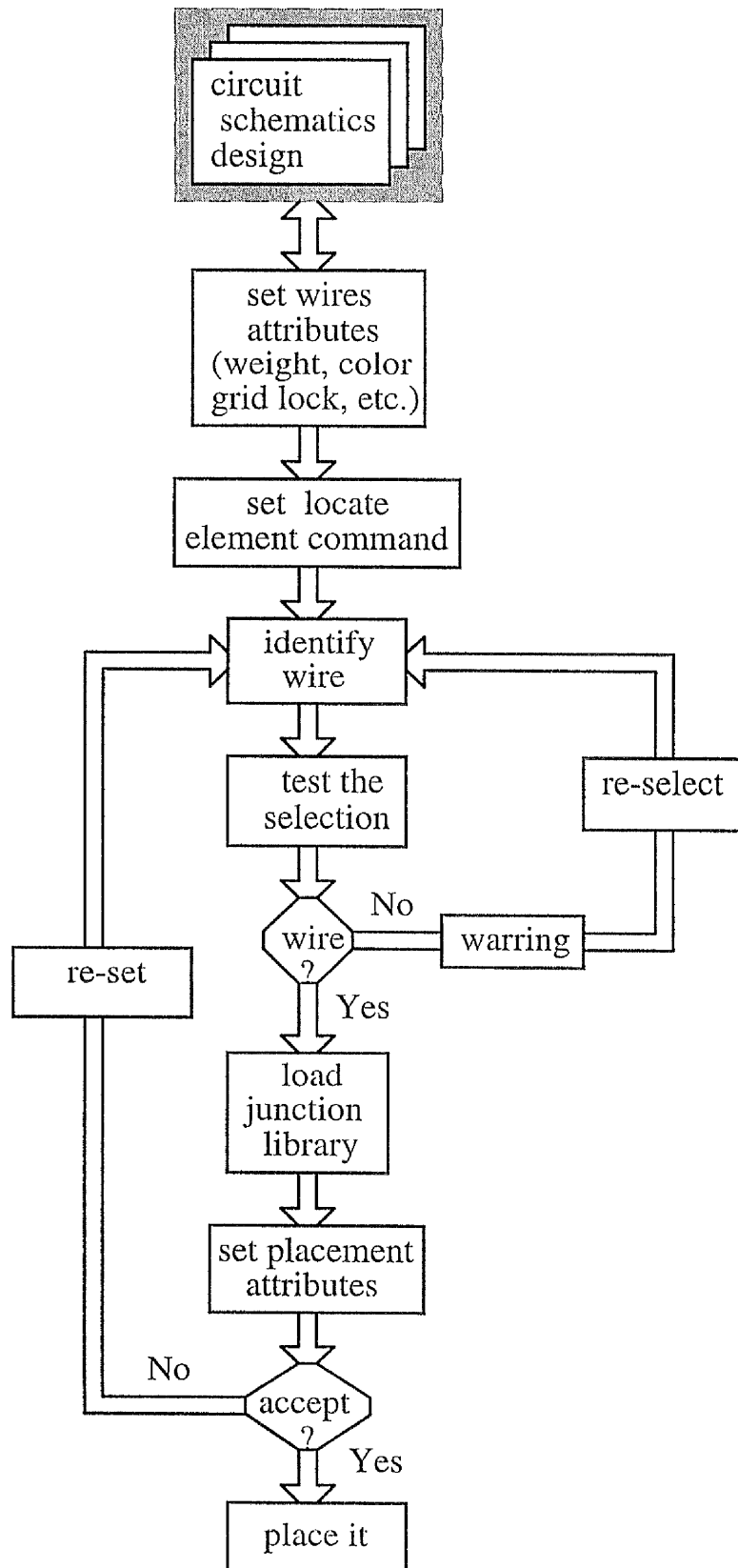


Figure 2.23 Flowchart of automatic junction placement program.

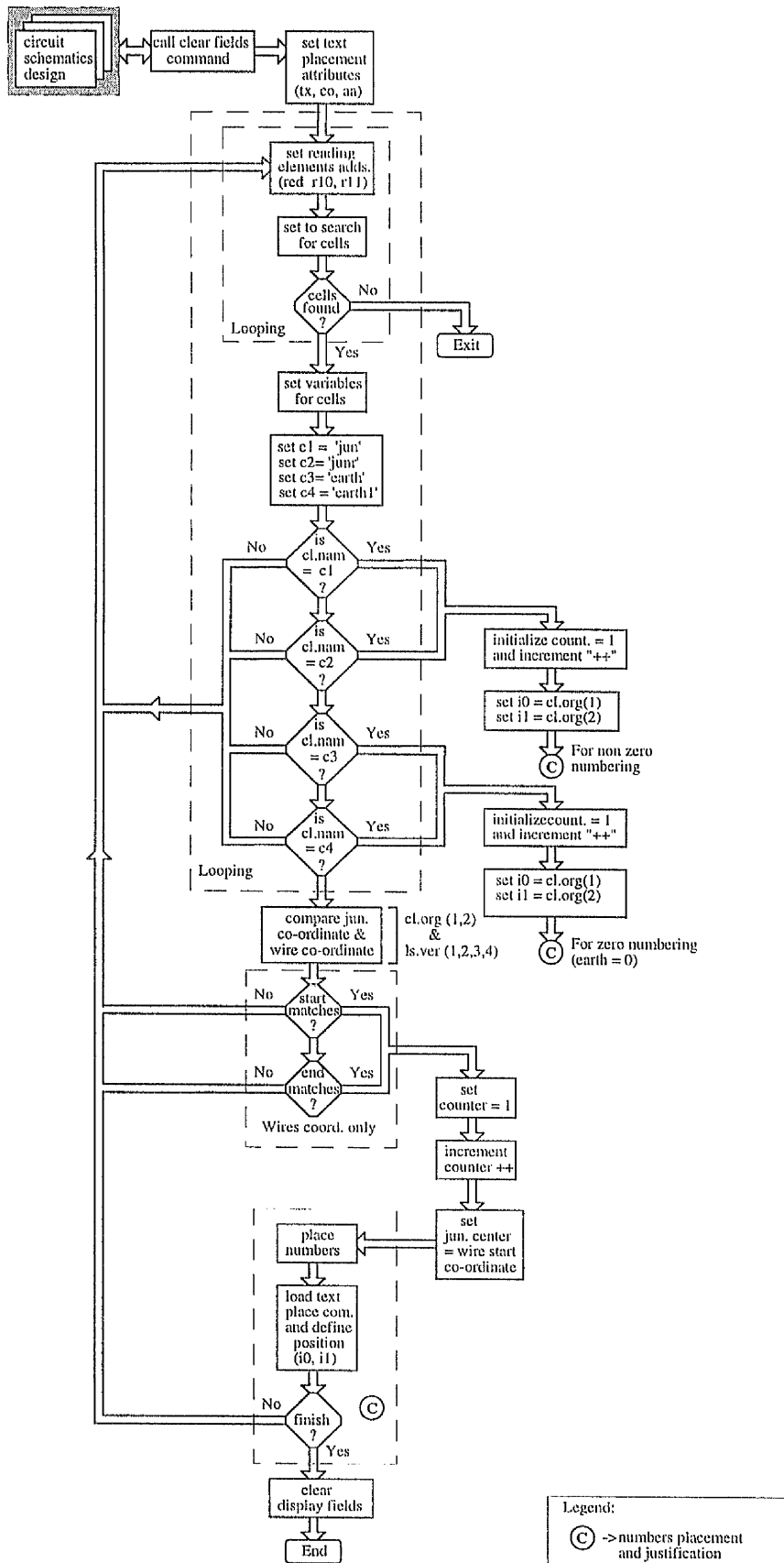


Figure 2.24 Flowchart of automatic junction numbering program.



## **2.6 CONCLUSION**

In this chapter an integrated environment for electro-mechanical design and analysis has been described and designed. The system framework architecture has described the components that comprise the design main features, which include: Unified GUI, 2D circuit schematic construction, 3D modelling layout design, simulation and optimisation performances, automated tools for design data generation, and database interface.

Attention was directed towards the entire system integration. In this case, data communication was the key issue among the involved environments. Database dbase(iv) is the centre element through which these environments communicate. This leads to the importance of data management, enabling the flow of design and other document to be controlled throughout the design process.

Generally, relevant problems in designing an integrated CAD system were identified and several needs were highlighted. These needs are listed as follows:

- A unified user interface through which different environments integrate.
- The importance of using common database in environments integration.
- Tools and automation to facilitate different design tasks.
- The ability to integrate commercial applications into design environment.
- Design and documentation plays a major role in most stages of design systems and serves as the initial process of passing information between various levels. For example, there is a need for data to be passed into simulation packages such as PSPICE to simulation and analysis for best possible solutions.

## **REFERENCES**

- [2.1] Jackson M. A., and Cameron J. R., "System development," Prentice-Hall International Series in Computer Science C.A.R. Hoar, Series Editor, ISBN 0-13-880328-5, 1983.
- [2.2] Charnely M., "A model design approach for electronic CAE," Computer-Aided Engineering Journal., Vol. 2, No. 6, pp. 192-199, December 1986.
- [2.3] Laden H. N., and Gildersleeve T. R., "System design for computer applications," Fifth printing, ISBN 63-17363, January 1967.

- [2.4] Harry K., "Systems design and documentation," An introduction to the HIPO method, Litton educational publishing, Inc., ISBN 0-442-24267-0, 1987.
- [2.5] "MicroStation PC MicroStation Development Language," Manual, Document Number DGA022410, Version Number 4.0, pp. M1-1, January 1991.
- [2.6] Bill S., "Pocket MDL programmers guide," Supports all 4.X platforms including DOS, UNIX, Mac, and VMS, first edition, ISBN 10987654321, 1993.
- [2.7] Gracy M. B., "Functional analysis of information processing," A structured approach for simplifying systems design , A wiley-interscience publication, ISBN 0-471-08846-3, 1973.
- [2.8] Stephen T. F. et al., "SPAR. A schematic place and route system," IEEE Trans. on Computer-Aided Design of Integrated Circuits and System, Vol. 12, No. 7, pp. 956-973, July 1993.
- [2.9] Marvin E. D. et al., "CAD system for IC design," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. CAD-1, No. 1, pp. 2-12, January 1982.
- [2.10] Jonathan S. et al., "An integrated, Intelligent Design Environment," Engineering with Computer Journal., Vol. 7, pp. 11-22, 1991.
- [2.11] Vivienne B., "Developing expert CAD systems," First edition, ISBN 0-85038-81-X, 1984.
- [2.12] Marvin E. et al., "CAD system for IC design," IEEE Trans. on Computer-Aided Design of Integrated Circuits and systems, Vol. CAD-1, No. 1, pp. 2-12, January 1982
- [2.13] Jones B., W. et al., "MicroStation database book," Tools for linking ORACLE, Informix, and dBase to MicroStation, published by onward Press, ISBN 10987654321, 1992.
- [2.14] Hamid R., and William G., "Concurrent Engineering," First edition, ISBN 0-412 46510 8, 1993.

- [2.15] Azeddin M. S., and Donald W. M., "Auto-Link Information to database," InterUser journal, Journal of the Intergraph UK & Ireland Graphics User Group, pp. 14-16, April 1995.
- [2.16] Larry S., "Trends in automating document generation," Advanced technology applications, Inc., IEEE Software Journal, Vol. 13, pp. 114-118, February 1996.

# 3

## Component Design and Modelling

### 3.1 INTRODUCTION

Many electrical components require detailed design (e.g. transformer, inductors) to perform to given circuit specifications. In addition to performing the overall circuit design tasks, it is a key functional requirement to be able to model each component where necessary within the circuit at a detailed level. In many instances an iterative approach must be taken, since component design will depend on other circuit performances, which will be driven by the interaction of components.

Generally, designers work with structural CAE tools to detail components. What is required, therefore, is a basic tool set for types of components being used which utilises common applications formats (e.g. Mathcad, Excel, Matlab, TK Solver) providing detail design parameters of the components and on completion, generating a behavioural model or sub circuit for inclusion in SPICE modelling. Following performance evaluation using SPICE, the design parameters of the components can be adjusted to provide optimum performance. Utilising an iterative approach, this has the potential to automate design optimisation.

Key to this philosophy is the ability of designers to develop specific design modules for components. For example a company designing switch mode power supplies could develop design templates for transformers which could be entered simply by selecting the component symbols on the schematic. The design templates would allow designers to investigate the effects of detailed changes in components (e.g. changing material type in a transformer).

In this chapter, general modelling procedure is fully described and compared to other different techniques. To demonstrate the application of this approach, design examples for an inductor and transformer are carried along with simulation and optimisation of different strategies to clarify the tools necessary.

## **3.2 COMPONENT MODELLING**

### **3.2.1 Modelling Objective**

Components and interconnections contain circuit elements such inductance, capacitance, or resistance and often a combinations. An understanding of the magnitude of these elements and the characteristic of components will ensure the correct choice and applications of a component. Where components cannot be modelled as a sub circuit, a behavioural modelling approach can be taken where the component is not easily classified as for example in laser gas charging applications [3.1].

### **3.2.2 Fundamental Modelling concepts**

The process of replacing a system with a simplified model, then investigating its properties by controlled experiment is referred to as simulation [3.2]. Therefore, in order to obtain meaningful simulation results, the model must be accurately represented with the constraint of those experiments. Furthermore, the model may exhibit completely different behaviour. This means that a system can be represented by a variety of different models, each having a specific purpose.

Typically, simulation of a model consists of a mathematical description of the component properties. As for example, a transient analysis is typically used to investigate the response of the system to time-dependent excitation. In order to obtain more meaningful and accurate simulation results, the system model must include accurate description to product performance.

### **3.2.3 PSPICE and Parts Models**

Most of the CAE tools currently used in discrete component design are based on SPICE circuit simulator. The importance of a simulator is that it correctly predicts the circuit output parameters; this requires the model and model parameter values to be accurate.

SPICE is an aid used to determine the model parameters for standard devices, transistors, and sub circuit definitions for more complex models such as amplifiers [3.3]. SPICE parts environment allows the designer to convert information from a component

data sheet into parameters which can be used by SPICE. Macromodels in SPICE are in common use for simulation analysis. They are sub circuits comprising transistors, diode models, resistors, and capacitors which are needed by the CAD program to simulate a system performance. Models to simulate a variety of semiconductor devices have been proposed, and they can be adapted through parameters adjustments to power applications as they are available from a library parts.

According to Graeme et al. [3.4], the idea of using macromodelling in electronic circuits has increased and it is very common at the system level usage. For example, to determine the actual system performance, a prototype circuit is constructed and tested. The size and complexity of integrated circuits (ICs) are large; therefore, the cost of using present simulators for design and evaluation can be very large. The cost for large ICs design can be justified if large volume manufacture is anticipated. The aim of using macromodelling design is therefore; to obtain a circuit model of ICs or a portion of an IC which enables a reduction in complexity in order to provide smaller and less costly simulation time. The resultant macromodels can then easily be incorporated by designers into system SPICE models. Many manufactures are now therefore supplying macromodels files for incorporation.

#### **3.2.4 Circuit Modelling Languages**

The design of systems that include analog and digital sections requires the development of advanced modelling and simulation methodologies, tools, and techniques [3.5]. Key essential elements to this development process are modelling methodologies and standards for describing designs. In the case of digital systems hardware description language (HDL) plays a vital role in the design of discrete systems.

Most CAD system uses a HDL such as verilog description hardware language (VDHL) proposed by Engel et al. [3.6]. This is accomplished through a provided tools. Other authors such as Hands [3.7], described the VHDL as a powerful and comprehensive language that allows hardware description in a form which can be readable to user and machines that uses it. VHDL also allows to design systems and circuits at different levels of abstraction, including the ability to design models. The use of this language supports many descriptions of hardware at many levels. Figure 3.1, illustrates the scope of very high speed integrated circuits hardware design language in comparison with some other modelling languages such as behavioural language modelling (BLM).

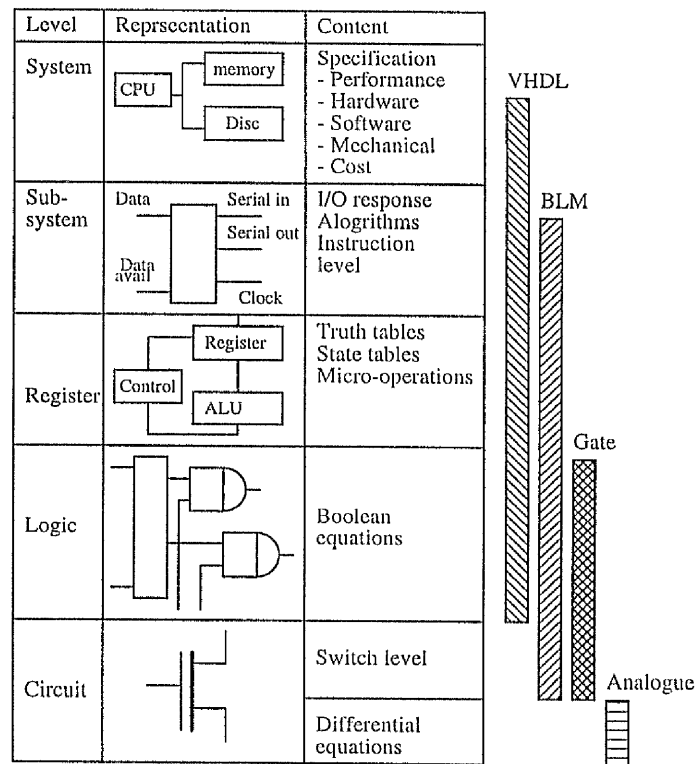


Figure 3.1 The scope of VHDL compared to some other modelling techniques [3.6].

With reference to these languages, Foulk [3.8] described them as registers because they were used to specify systems at the level of registers. Their scope, however, is now much increased. Simulators are used with a descriptive language to allow electronics computer-aided design (ECAD) system to be simulated at functional level. The parts of the system must then be designed at the individual component level, and the system is defined as a interconnection of electronics modules.

Hardware description languages (HDLs) are also defined as descriptive language that allows users to describe both structure and behaviour of analogue and digital systems. They used to support specification of both design behaviour and design structure as they can be written in different styles. For example, to specify design behaviour, HDLs provide a set of variables and set of operations for computing current values. Most languages, however, provide rules and conditions (e.g. if and case statements).

Another type of modelling language such as MAST used by analogy saber package. It used to represent a model in terms of non-linear equations [3.9]. The saber simulation can then use these models to perform further simulation analysis. The interaction of this language and saber simulator provides a powerful method of investigating systems that contain a wide variety of models. However, because the MAST has capabilities of HDL, it can address general modelling requirements that are independent of system technology.

The use of MAST language can also be applied to assist in building models of non electrical system such as mechanical, hydraulic, or thermal systems modelling.

Additionally, the OrCAD system uses a programmable logic design (PLD) tools [3.10]. It is fully integrated electronic design automation tools, that allows to define logic in different ways. The design environment of these tools are structured to allow the user to focus on what is important. Designs are organised in project by project basis, with all relevant design files such as schematics, design database, part lists, source code, and simulation results stored together. The logic definition for PLD, consists of a series of text objects placed in schematics diagram that represents a circuit layout.

Various modelling techniques mentioned by Farid [3.5], provides a rich set of constructs to model the behaviour of analog systems. Using analog hardware description language (AHDL), these constructs can be modelled using various set of equations and mathematical operators. These constructs supports a variety of modelling styles for analog systems in various analysis and design domains. The use of AHDL intend to focus on the simulation needs, and have been developed with the intent of modelling and representing circuit systems in order to be simulated efficiently. However, since that the other aspects of analog CAD/CAE still under development, its yet unclear as to what role of AHDL would play in terms of other design tasks such as synthesis, layout generation, design optimisation, design specification and documentation, and other relevant design aspects. The most common features provided by AHDL towards the design representation across different CAD applications are defining simulation models in terms of model interconnections defined across different levels and domains, and finally the ability of multiple viewing for a model, so that analysis can be carried in different analysis areas.

It is clear that digital design and simulation is being well catered for, however, both analog and power electronics design systems have not evolved as far. This is mostly due to the complexity of the design and non-linear behaviour of circuits, which are much more difficult to simulate.

### **3.3 ELECTRO-MECHANICAL DESIGN**

The increase use of electronics and mechanical components emphasis on designing and modelling electro-mechanical CAD/CAE packages. The design of electro-magnetic devices such as electrical machines actuators, transformers, etc.; has developed on both experience and analysis techniques [3.11]. Each development in analysis techniques has allowed the design to be simulated more accurately in testing a design. CAD systems



have already evolved to guide the search for the proper design using different techniques but the knowledge to evaluate the results in comparison with the provided specifications.

Designing a CAD tool to be able to perform different level of design of analysis for specific component a feedback of information from the analysis to compare with the initial specifications must be provided. The results then need to be critically examined, if sufficient result is not reached, the design is modified and re-analysed using simulation packages. Lowter et al. [3.11], has suggested the important aspect of developing a design of this kind is to have some sort of feedback in both the optimisation process and the refinement of the weak design constraints in order to reduce an acceptable design. In order to improve the entire software performance accurate set of rules must be provided, therefore, achieving a functional design with minimal computing effort.

### **3.3.1 Electro-mechanical relational Design**

Competition and short design cycle forces designers to consider the complete system and to strive for the most optimal design possible [3.12]. Combining electrical and mechanical consideration simultaneously in the design process is known as Mechatronics. Such a combination in design practice requires an improvement in design tools to support a design environment focused on achieving optimal product within a short design cycle.

Historically, the term Mechatronics was first proposed by Yaskawa in 1969 [3.13]. As it became known, development in component related electronics, such as microprocessors, and in power electronics such as high-capacity, high-speed switching characteristics has led to the adoption of electronics in machinery. Products based on this classification offers significantly enhanced function and performance. With the provement of function and performance a variety of electro-mechanical products began to appear on the market. The products are a combination of mechanical and electronics disciplines.

The integration of mechanical and electronics systems leads to many new possibilities for process design and automatic functions [3.14]. As for example, mechanical systems provide motion or transfer forces or torque, while electronics systems allow the performance on many more functions. Mechanical and microelectronics are easily integrated forming a unified unit. This, however, leads to the integration in the following sense:

- Mechanics (mechanical engineering, mechanical devices).
- Electronics (Microelectronics, power electronics, measurement and actuators technologies).

- Information technology (systems theory, automation, software design, embedded control).

### 3.3.2 Parametric Design

The ability to represent a family of objects similar in their geometric shape or in a functional sense is called "parametric design". The terms means different things in different contexts. What is needed is a product descriptions that is complete, and contains semantic information that reflect the needs of different engineering domains, that reflects the different levels of detail required for different tasks, and that clarifies the whole product creation process.

Parametric design function can be applied to different engineering applications. Figure 3.2, shows the parametric design as an application to product design concept, whereas several engineering applications are involved. For example, The parametric approach involves complete or partial automation of the production of drawings, geometric models and other aspects of design data for particular classes of electro-mechanical designs. This can also involve variation in the dimensions and perhaps the arrangement of standard design elements.

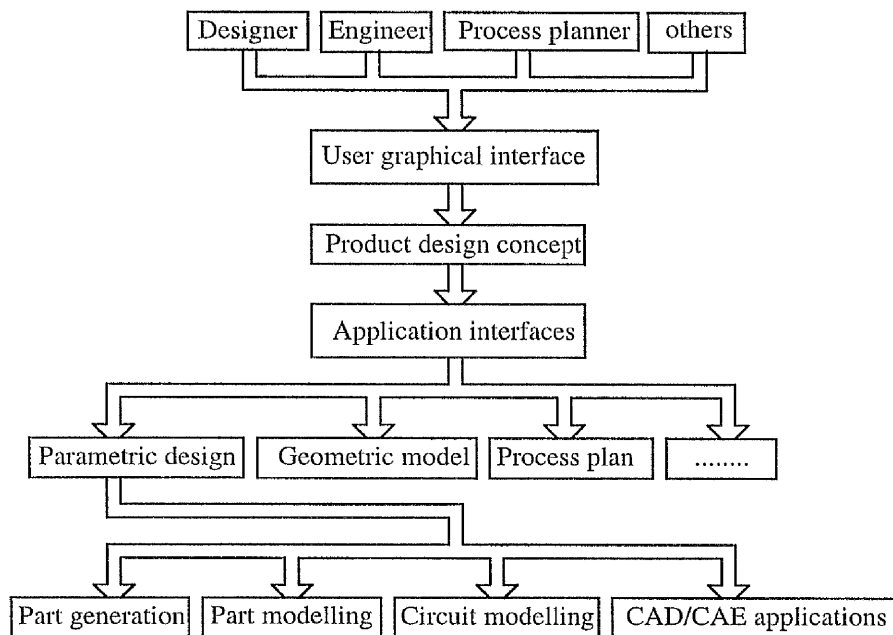


Figure 3.2 Parametric design supports different applications.

As general awareness of CAD and users expectations increase, software developers seek to supply more comprehensive facilities that can be used to assist in CAD different

design activities, including design by features. Mechanical design of engineering parts, for instance, can be classified into a class of design that may be regarded as parametric. This includes examples, such as fasteners, bearing, etc.

Applying parametric design techniques to electrical components design also have their own impact, an example of this, is the modelling and designing of electrical circuits and components. There is a need for an automated design tool to perform these tasks. CAE/CAD software tools are considered and devoted specifically to the design and modelling of circuit components, where the objective is to control the parametric issues with respect to the circuit specification parameters. These parameters have a great influence in developing a detailed design. For example, with respect to transformer design, parameters expressions and equations are used to relate the performance of both circuit and the component itself. For instance, parameters such as input/output voltages and powers have very significant influence on the transformer efficiency (i.e. used to measure the effectiveness of the design). By assuming, the core part has two involved windings  $P_i$ ,  $P_o$ , where is  $P_o$  is the transformer output power and  $P_i$  is the transformer input power. Also by assuming the window area  $W_a$  divided to handle the power capability of the two windings using equal current density. Since that, the transformer has to be designed to accommodate both  $P_i$ ,  $P_o$ . Therefore, concern must be taken towards the power handling capability of the transformer core and windings. Detailed description of calculating the transformer efficiency and other relevant parameters can be found in the transformer design theoretical calculation section.

There is a clear distinction in applying the application of parametric design into both mechanical and electrical design problems. The principle idea lies in identifying the functional requirements that able to model and design a component in a circuit with possible detailed descriptions. Furthermore, merging these two aspects, electro-mechanical design is obtained and the application of parametric design function is considered to be a combined effort between electrical and mechanical components.

### 3.3.3 Parts and Attributes

Typically, attributes are defined as the quality of a thing and are declared within the part module which defines it. Figure 3.3 illustrates hierarchy structure of part-attribute translation whereas components and circuits are treated as two separate design issues. For example, in designing part such as cylinder might be defined as (cylinder (att(len), att(dia))). While in designing electrical circuits, parameters and assumptions are considered to achieve a parametric function.

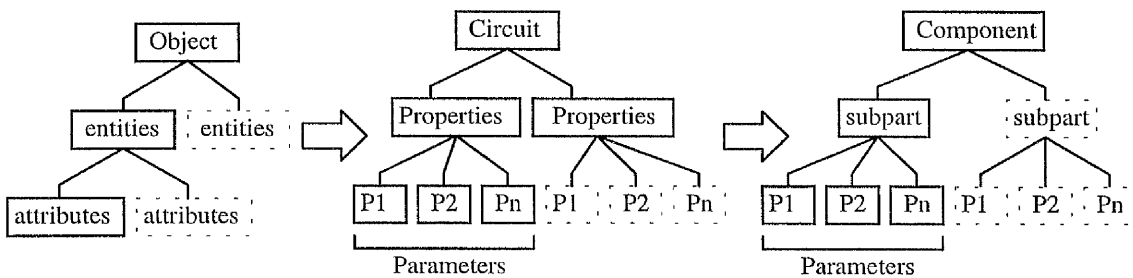


Figure 3.3 Part-attributes relationship translation.

Furthermore, attributes can also be applied to any collection and level of features, complete part, or to any process that is needed to produce a product (i.e. referred to it as manufacturing feature). Figure 3.4 illustrates part-attributes that may be included to design mechanical standard part, whereas parameters such as type, size, length, and view type can be considered.

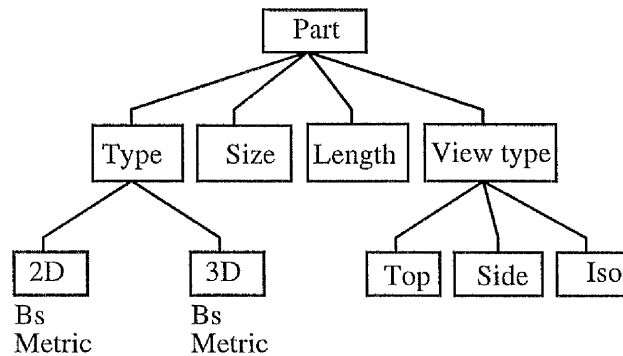


Figure 3.4 The concept of mechanical part-attributes definition.

Despite mechanical components have their collection of attributes and parameters. Electrical parts are also have their own attributes defined with respect to their functionality. There is a need for methods in CAD systems to support attributes at any design level, allowing different approaches to support a wide variety of products and applications. For example, the electro-magnetic design system (EDS), proposed by saldanha [3.15] is a software program which uses artificial intelligence (AI) techniques for electro-magnetic devices design. It is a framework which simulates the design process for different electro-magnetic devices. The basic methodology in this design is that each device model involves a set of rules used to interface the design by a given specification. The design also involves the use of knowledge representation of different kinds such as the structural and functional of a device, the model of analysis, and the expert design rules.

With respect to inductor and transformer design examples, the parameters are organised as part-attributes relationship. By structural variables parts are referred to as entities (i.e.

core, primary winding etc.), while attributes related to each one of these entities are either calculated or user-defined. Figures (3.5, 3.6) respectively illustrate detailed description of inductor, transformer and their design attributes.

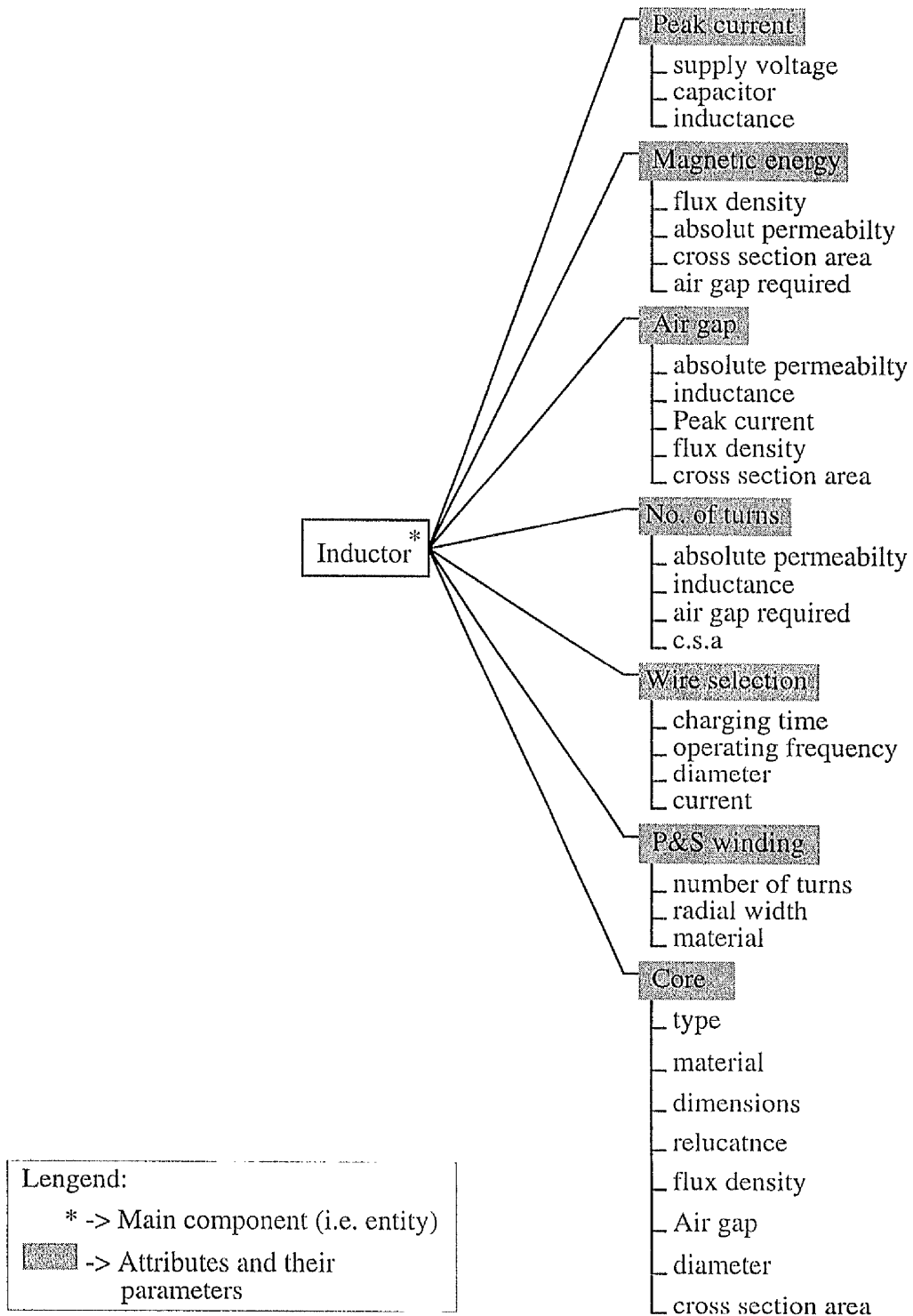


Figure 3.5 Inductor part-attributes description.

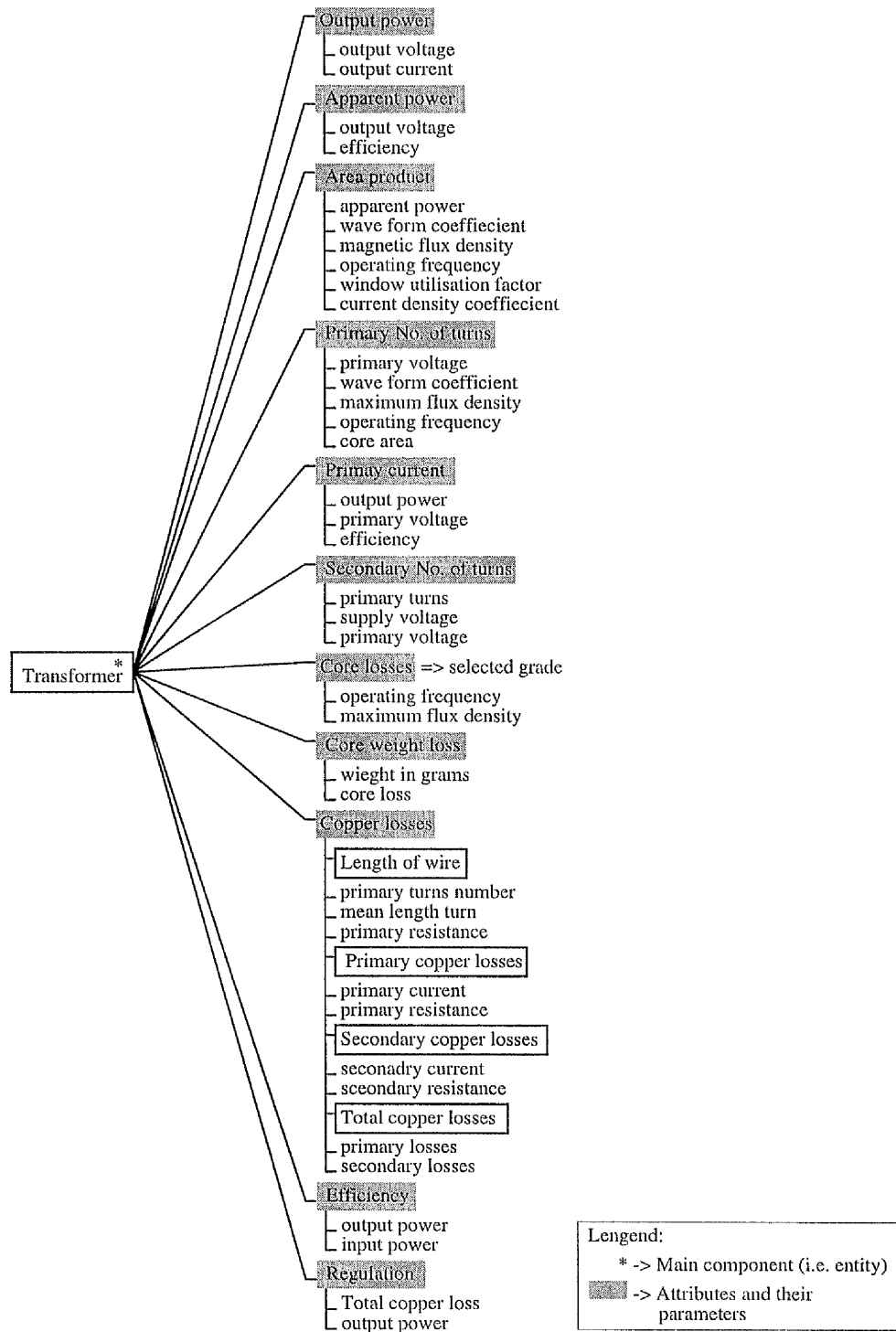


Figure 3.6 Transformer part-attributes description.

### 3.4 DESIGN AND MODELLING PROCEDURE

In this section, the modelling procedure is described, giving the ability to translate the design from functional description into a form suitable for testing. Using this procedure as to design components into details, offers the following advantages:

- Detailed design makes the circuit to be developed in shorter time using existing models through out the process.
- The design approach tends to break the circuit into a number of steps, therefore, circuits are more efficiently designed.

### 3.4.1 The Procedure Requirements

The objective of this section is to develop an understanding of the process taken towards the design and modelling for simulation. It begins with the descriptive scope of components, which underlines the following requirements:

- Structural knowledge about the circuit and its components, giving a general idea of the key element which will be involved in modelling that component.
- Knowledge about the component operation functions and their parameters.

Based on these requirements, the design process is performed as an iterative process to evaluate and modify a selected component. Figure 3.7 shows the general procedure for component design and modelling process.

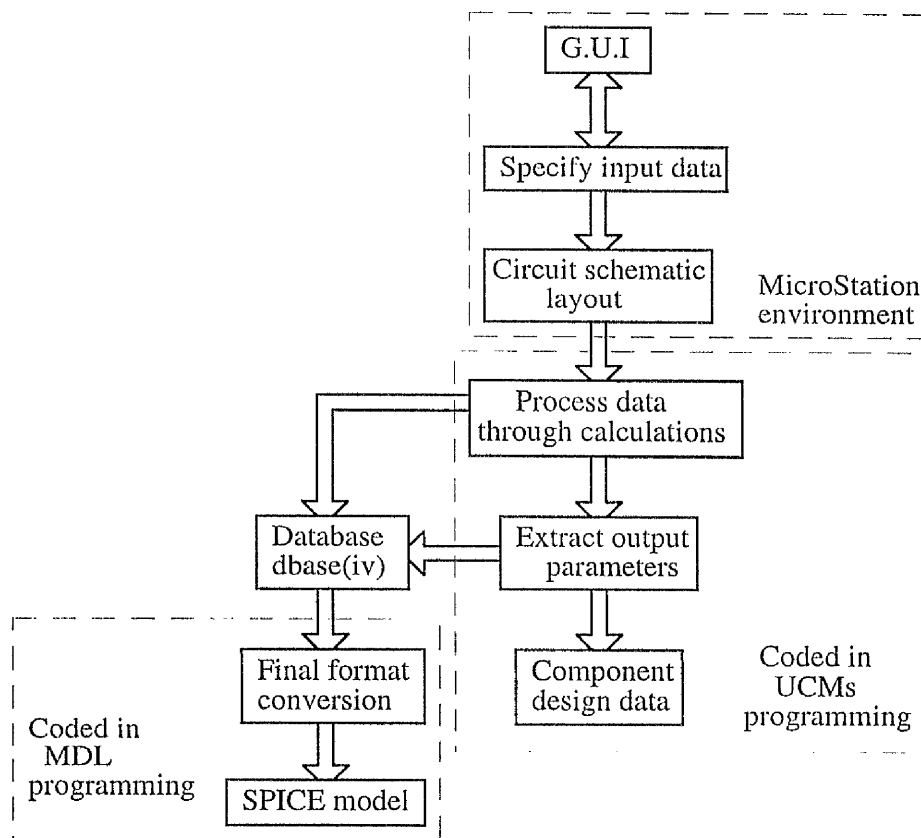


Figure 3.7 Design and modelling general concept.

The concept for the procedure is to operate depending on a selected component which basically involves the following steps:

- Components are selected within 3D modelling layout.
- Components input data are the circuit specification.
- Software programs used to model the selected component with respect to this specification.
- The component modelling parameters are supplied or calculated by means of mathematical equations.

Based on these fundamentals, components are designed in more details and the results are therefore considered as the circuit output performance. In addition, the proposed approach is a generalisation which accommodates component design, and is therefore designed to exploit factors and parameters which are common to most components. In the examples given in Section 3.6, the most obvious determinant is the physical laws and principles of electro-magnetuim, which plays an important role in forming the basis for the operation of all components.

### **3.5 COMPONENTS DESIGN ENVIRONMENT**

The design environment provides a set of common services, which can be used by the provided tools. Such services include facilities for structured data storing and manipulation. Steps towards the integration have been taken by providing the design process. In this environment, tools communicate via a database which used to recall all relevant information about specific process taken.

#### **3.5.1 Integration of Tool sets**

In order to provide an automated support to aid the whole process, the integration of tool sets is necessary to ensure that the complete process is supported in flexible manner. Integrating the evolved environments leads to a number of advantages:

- They share utilities including those for database access (i.e. interfacing).
- They employ common internal representations of programs and have common means with varying sets of facts.

The integration concept shown in Figure 3.8 holds design database as a common store. This provides central location at which to perform a specific design task in supporting the design process in general. The advantage of this is that all access to these data is done



through various designed tools, whereas four main areas are involved: Schematics workspace for 2D drafting, Database for information storage, 3D modelling layout, and PSPICE environment for simulation and analysis.

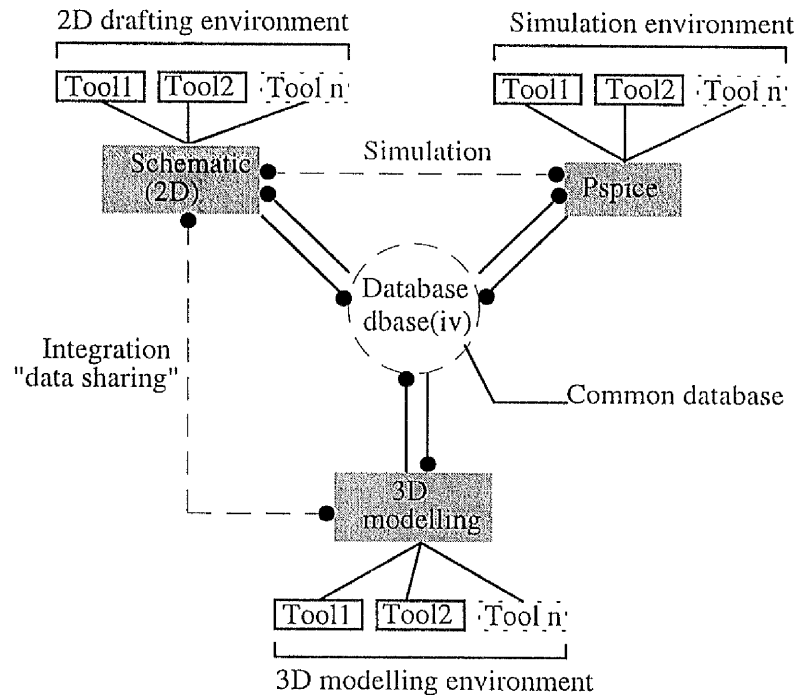


Figure 3.8 Illustration of the integrated environments and their tools.

### 3.5.2 Library Cells Design

Using the MicroStation different graphical elements, cells are constructed as graphical symbols to represent 2D and 3D components. They are provided to be used as for data definition entry and modelling of components. Chapter 6 reveals an example of how these cells are constructed using library environment with the assistance of the customised tools.

### 3.5.3 Interaction with Database

In addition to the component graphical data, the product descriptive data should be linked with the information of product geometry representation, such as types of material used in manufacturing, the vendor information, and some other functional specifications [3.16]. Because of the extensive use of the relational databases especially in the industry, both product description and geometry parameters values are stored in relational database system. For this purpose database tables of different formats are created to define components and their associated parameters in order to be applied in the geometric specification. Detailed explanation about the use of database can be found in Chapter 5.

In this case, the user interacts with the design database at several stages throughout the design process. The most important stage of all is the problem definition, where a request is taken firstly by selecting the desired component and secondly by entering the component specifications. These specifications are constrain component parameters including circuit performance parameters. They permit the software program to determine the other unknown parameters and storing them into the used database tables by means of interfacing and UCMs programming. This structure is illustrated in Figure 3.9. As this figure shows, the use of database consists of two major issues, namely: store and extract data. The two issues are basically software written programs interact with the used database through the MDL software built-in functions.

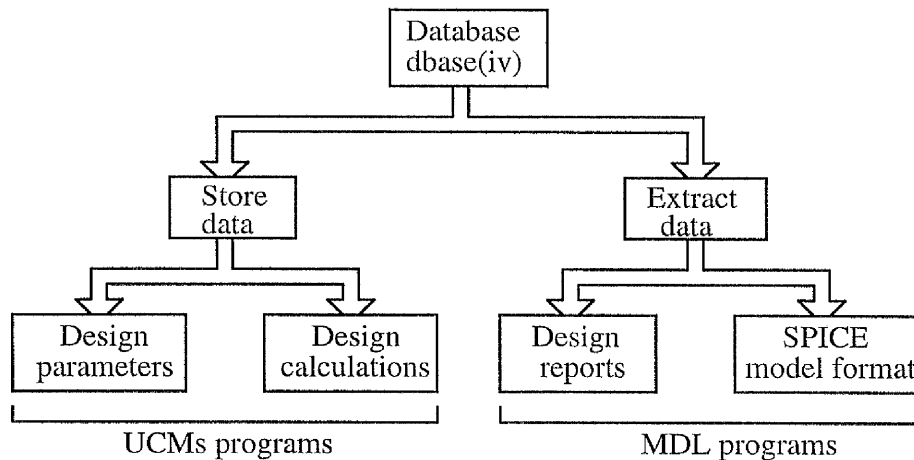


Figure 3.9 Block diagram illustrates the use of database.

### 3.6 COMPONENT DESIGN EXAMPLES

In this section, design examples of electro-magnetic components such as inductor and transformer are used to test the accuracy and performance of the modelling procedure described above, where each example is provided with its input data. The application of the modelling procedure to simulation of circuit component is presented in Section 3.7 and the output contains program performances and other relevant parameters are reported.

#### 3.6.1 Charging Inductor Design

A set of constraints are identified when designing inductors which they must observe [3.17]. One of these is the material loss (e.g. copper)  $P_{cu}$  due to the resistance of the windings. The windings used must be designed to deliver current to the load with specific regulation limits. Some other losses as in gapped inductors have three kinds of losses in which they must be taken into account, viz: Core losses  $P_{fe}$ , which for a given voltage

and frequency are practically independent on the applied load, and air gap losses  $P_g$  or the stray losses largely due to eddy currents induced by the leakage fluxes. The maximum efficiency can be reached in designing an inductor or transformer when the air gap  $l_a$  is zero and  $P_{cu}$  and iron loss  $P_{fe}$  are equal. Designing with cores made of ferrites material the gap loss will be reduced this due to ferrite material have such high resistivity.

For higher inductance values a magnetic core is used inside the former. These comes in different shapes and sizes, and different types of material. For example, for low frequency applications iron is generally used to reduce eddy current. At higher frequencies (i.e. ranging from KHz to MHz), ferrites are used because of low losses. The disadvantage of ferrites as compared to iron is low  $B_{sat}$  (i.e. magnetic flux), 0.3T compared to 1.5T for iron. Figure 3.10 shows detailed description of inductor with magnetic core.

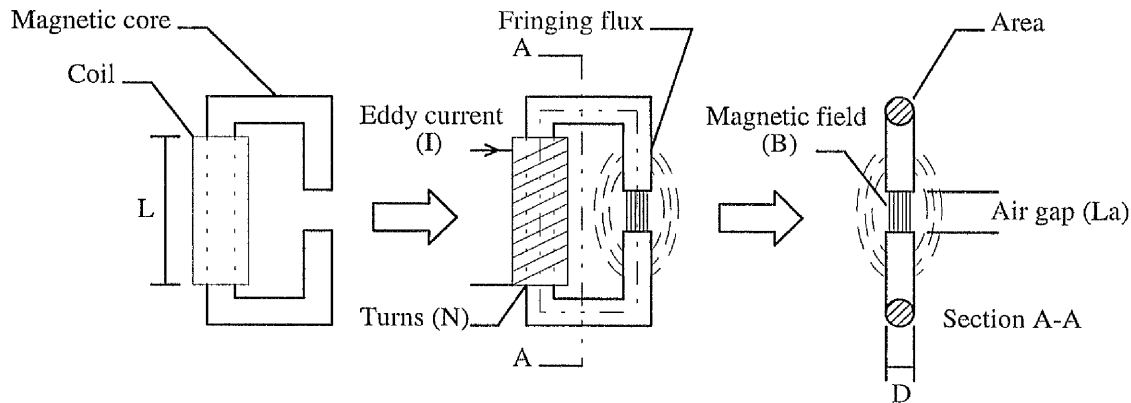


Figure 3.10 Inductor detailed description.

The problem which follows, is a selected design example for designing and modelling a component. The circuit model shown in Figure 3.11 adopted from McDonald [3.18], illustrates the theory of transferring the energy from DC power supply to Pulse Firing Network (PFN) inductance.

In this design, a numerical example is performed to illustrate the proposed approach of the inductor design. The response characteristics are used as determined by PSPICE, and the circuit of the above figure together with specification parameters of Table [3.1] have been analysed to obtain the final output parameters which are summarised in Table [3.2]

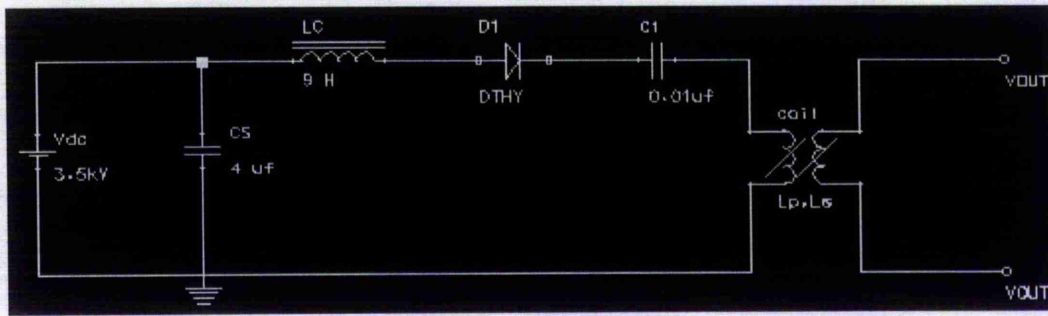


Figure 3.11 Charging inductor design circuit [3.18].

Table [3.1]. Circuit design specification.

Parameter	Name	Value	Units
Charging capacitor	Lc	1	$\mu F$
DC voltage	Vdc	3.5	kV
Time scale	T	100 (max. 95)	$\mu s$
Frequency	F	10	kHz

### 3.6.2 Problem Definition

Capacitor  $C_1$  needs to be charged to 3.5 kV after each pulse within a time scale of  $100 \mu s$  for an operating pulse repetition frequency (PRF) of 10 kHz. This can be achieved by resonant charging via a charging choke  $L_c$  and charging diode  $D_1$ . The detailed performance parameters for the whole circuit is obtained by defining the problem, and are shown in Table [3.1]. The input parameters used in this example are employed in order to obtain an accurate estimation of the inductor with respect to the circuit involved components (i.e. current, voltages, maximum charging time, etc.). Figure 3.12 illustrates the equivalent design model for  $L_c$ , while Figure 3.13 shows the inductor detailed dimensions.

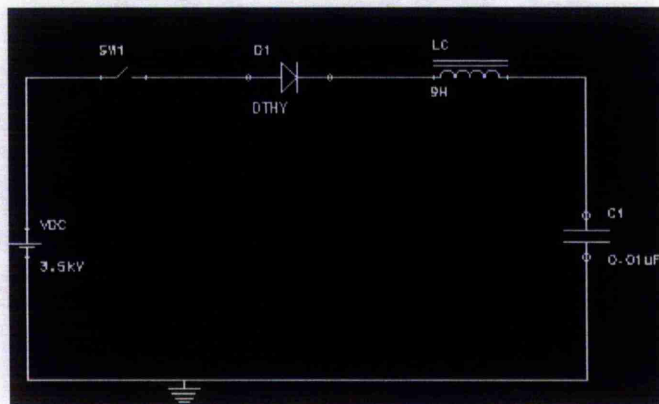


Figure 3.12 The equivalent design model of  $L_c$

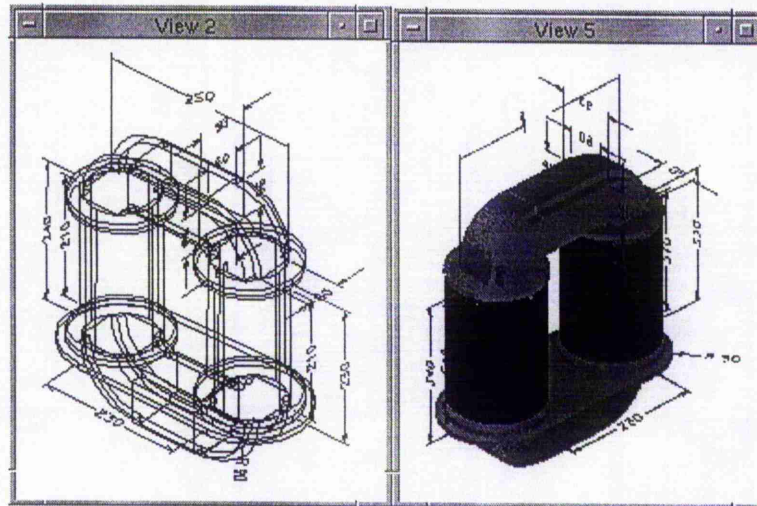


Figure 3.13 Inductor with detailed dimensions.

### 3.6.3 Theoretical Calculations

#### Step 1.

Calculate the inductance value for maximum charging time of  $\tau_{\max} = 95\mu s$

$$\tau = \pi\sqrt{LC} \quad (1)$$

$$L = \frac{\tau_{\max}^2}{\pi^2 c} = \frac{(95 \times 10^{-6})^2}{\pi^2 \times 1 \times 10^{-9}} = 9 \text{ H} \quad (2)$$

#### Step 2.

Calculate charging peak current:

from peak charging current:

$$\hat{I} = V_{DC} \cdot \left[ \frac{C_1}{L_c} \right]^{\frac{1}{2}} \quad (3)$$

For resonant charging

$$V_{DC} = \frac{1}{2} \cdot V_0 = 1.75 \text{ kV}$$

$$\hat{I} = 1.75 \times 10^{-3} \left[ \frac{1 \times 10^{-9}}{9} \right]^{\frac{1}{2}} = 60 \text{ mA}$$

#### Step 3.

Calculate cross section area (CSA), using two U cores with equal gaps in each leg. The peak magnetic energy stored in the gap to suit the peak current is given by:

$$E_{mag} = \frac{1}{2} \cdot L \hat{I}^2 = \frac{1}{2} \cdot \frac{B^2}{\mu_0} \cdot A l_a \quad (4)$$

Limit peak flux to  $0.1\tau$

$$A \cdot l_a = \frac{L \hat{I}^2}{B^2} \cdot \mu_0 = \frac{4\pi \times 10^{-7} \times 9 \times (60 \times 10^{-3})^2}{0.1^2} \quad (5)$$

$$A \cdot l_a = 4.1 \times 10^{-7} = \frac{\pi}{4} D^2 l_a$$

To avoid the flux firing at the gap assume  $l_a = 0.01D$

$$\therefore D = \left[ \frac{4}{\pi} \cdot \frac{4.1 \times 10^{-7}}{0.01} \right]^{\frac{1}{3}} = 37.4 \text{ mm}$$

This is a bit too large : try design with 20 mm core.

Therefore,  $A = 2.41 \times 10^{-4} m^2$

#### Step 5.

In order to avoid saturation, a small air gap is made in the core between the two sections of the core.

$$l_a = \frac{\mu_0 L \hat{I}^2}{B^2 A} = \frac{4\pi \times 10^{-7} \times 9 \times (60 \times 10^{-3})^2}{0.1^2 \times 2.41 \times 10^{-4}} \quad (6)$$

$$\therefore l_a = 1.7 \text{ mm} \quad (\text{i.e. } 0.85 \text{ each leg}).$$

#### Step 6.

Number of turns can be calculated using:

$$N = \left[ \frac{l_a L_c}{\mu_0 A} \right]^{\frac{1}{2}} \quad (7)$$

$$N = \left[ \frac{1.7 \times 10^{-3} \times 9}{4\pi \times 10^{-7} \times 2.41 \times 10^{-4}} \right]^{\frac{1}{2}}$$

$$\therefore N = 2248 \text{ turns}$$

#### Step 7.

Calculate the value of smoothing capacitor, which is designed to minimised ripple voltage:

Ripple voltage  $V_r = 5\%$  of  $1.75 \text{ kV} = 87.6 \text{ V}$

$$C_s = \frac{\bar{I}_t}{V_r} = \frac{35 \times 10^{-3}}{87.5} \times 0.01 \quad (8)$$

$$\therefore C_s = 4 \mu F$$

Tabel [3.2]. Component output parameters.

Parameter	Name	Value	Units
Charging choke	Lc	9	H
Peak charging current	Ipeak	60	mA
Core diameter	D	37.4	mm
Air gap	La	1.7	mm
Air gap (each leg)	La/2	0.85	mm
Number of turns	N	2248	
Smoothing capacitor	Cs	4	$\mu F$

### 3.6.4 Design Optimisation

Design optimisation basically involves the probing of design space by analytical techniques based on the variation performance with the design parameters [3.10]. The theory of optimising a design is specifically directed toward component design as well as circuit design. Thus, the design parameters are elements of the circuit model. The optimisation of these parameters identifies common features in circuit and system design. As they are limited number of parameters to be optimised, a certain computation load is incurred due to excessively complicated objective functional evaluation needed for specifically the optimisation of transient characteristic. Because of the increasing complexity of circuits, this is considered to be a complicated process concerning modern circuits designs, thus, time consuming. Therefore, evaluation processes are necessary so that high level of efficiency can be obtained.

Assumptions have been made for an ideal switch to be connected to provide the supply of constant voltage to the charging inductor  $L_c$ . Since that the PFN inductance can usually be ignored in comparison to the charging inductance, the PFN inductance can be modelled as capacitor equal in capacitance equal to the total capacitance in the network  $C_n$ . If the  $C_n$  is initially charged to a voltage of  $V_n(0)$ , then upon closing the charging switch the passing voltage can therefore be obtained from:

$$V_n(t) = V_s (V_n(0) - V_s) \cos(\omega t) \quad (9)$$



$$I(t) = \frac{(V_s - V_n(0)) \sin \omega t}{\sqrt{L_c/C_n}} \quad (10)$$

Where  $\omega = \frac{1}{\sqrt{L_c/C_n}}$  is the circuit resonant frequency. This can be presented graphically as shown in Figure 3.14.

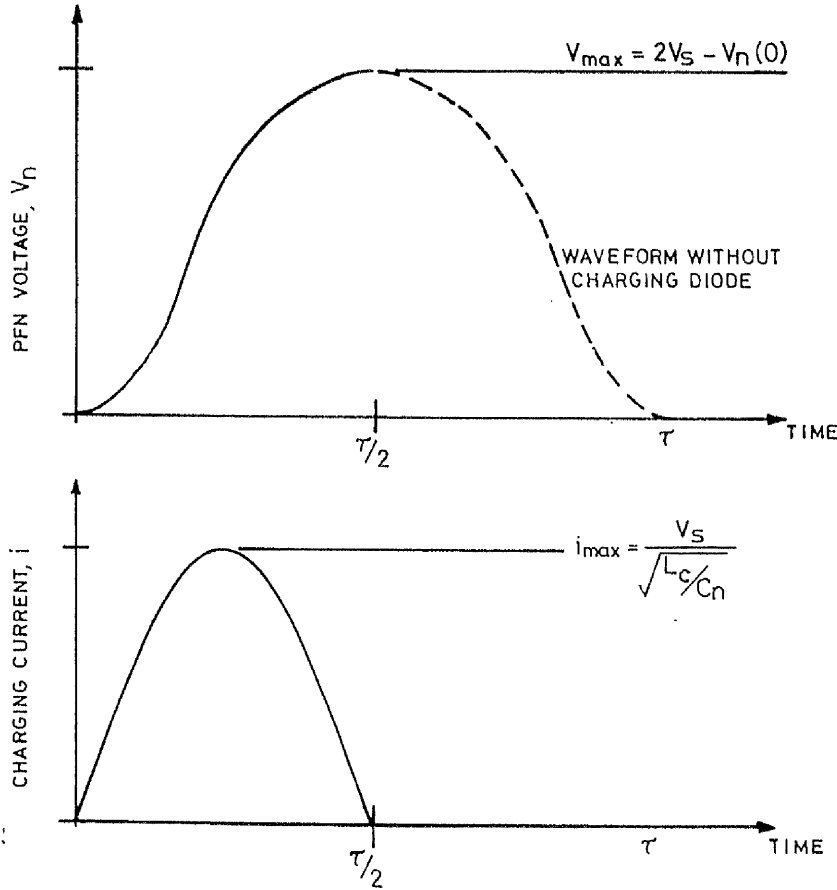


Figure 3.14 Wave forms of charging current and voltage [3.18].

It can be seen that  $C_n$  can be charged to the maximum voltage in time half the period of resonance of the circuit, at which the current begins to discharge. A charging Diode is usually included to avoid the current reversal at this point and maintain at the peak voltage. Therefore, with the inclusion of the Diode the  $C_n$  final voltage can be expressed as:

$$\tau = \pi \sqrt{L_c C_n} \quad (11)$$

from the value of the charging inductance  $L_c$  which requires a charging time of  $\tau_c$  can be calculated. The circuit schematic contains assumptions and in practice the circuit losses will therefore, result in a lower network voltage. As the mean powers required for the present applications, it is quite difficult to provide a constant DC supply voltage,



which usually requires smoothing capacitance across the output terminal to minimise droop.

Flux density  $B$  is linearly dependent on the peak current as graphically presented in Section 3.3.6. In designing the inductor, it is desirable to choose the core area  $A$  and gap length  $l_a$  to suit value of peak current  $\hat{I}$ . To avoid flux fringing at the gap  $l_a \ll D$  where  $D$  is the core diameter (typically  $l_a / D = 0.01$ ). It is possible to introduce the optimisation procedure into a simulation to determine the whole circuit performances and therefore, the final inductor design parameters are obtained as fully described in Section 3.6.

### 3.6.5 Software Program Development

The software program basically consists of two main parts each of which performs different task. The first part is coded in UCMs macro language to provide commands for entering design data. While the second part which involves the extraction of PSPICE model format is coded in MDL. These parts are fully explained as follows:

#### Data entering program

The procedure of prompting the design for input values of parameters and the instant of interacting with the database takes place once a component is selected. This is done by the assistance of the MicroStation and database server (refer to Chapter 5). The program works forward from those parameters specified by the user to database storing. Its intention is to manipulate numerical values, whereas equations can be solved so that solutions can be obtained for any design variable (or parameter). There are four main steps involved in constructing this program, namely for: loading of design data table, routine checking for the selected component, data supply commands, and finally calculations processing and data storage. The program flowchart of Figure 3.15, generally illustrates these steps in sequence.

In the database structure shown in Table [3.5], the first field provided is to store the number of mslink which is considered to be the most important field in establishing the link between schematic components and database table. The number of fields depends on the given design specification, where parameters of the input data are identified. In the inductor design example for instance, there are two types of data fields provided: Input parameters fields to store the design data inputs specified by the user (i.e. descriptive data), and the other is to store the design data outputs and are used for further design tasks.

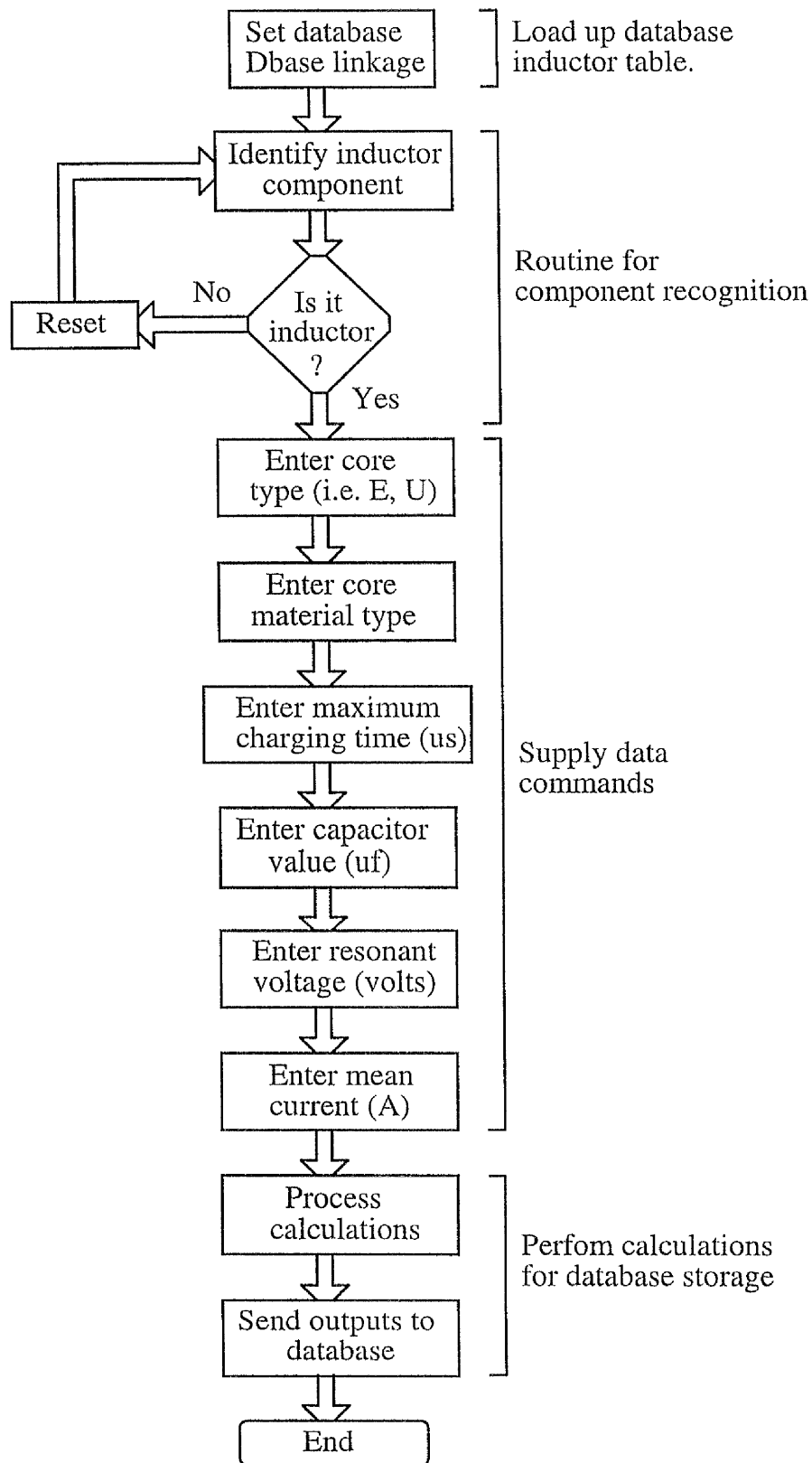


Figure 3.15 Flowchart of inductor design program.

Table [3.3]. The database structure for inductor design.

Num	FieldName	Fieldtype	Width	Dec	Index
1	mslink	Numeric	10	0	y
2	core type	character	10		n
3	material	character	10		n
4	charging time	character	10		n
5	capacitor	character	10		n
6	indotor	character	10		n
7	voltage	character	10		n
8	peak current	character	10		n
9	air gap	character	10		n
10	air gap/leg	character	10		n
11	no. of turns	character	10		n
12	mean current	character	10		n
13	r/voltage	character	10		n
14	C/smoothing	character	10		n

Legend:

-> Supplied data

-> Calculated parameters

Figure 3.16 illustrates the parameters calculation part extracted from the inductor design program.

```

;-----Calculations part only-----
set a3=c2*c2*0.0001
set a4=9.8696*c3
set a5=a3/a4
set r6=a5/100
set r6=c3/r6          ;the inductor value (uH)
SQR a0, a6
set a8=c4*a0
set r7=a8*100          ;Peak current value (mA)

;-----Calculate the air gap required-----
set a9=a8*a8          ; for peak current sqr.
set a10=12.567*r6*a9  ; u0*inductor value
set a11=0.0241        ; density*c.s.a.
set a12=a10/a11       ; a12 for (air gap required)
set r5=a12/2          ; air gap required in each leg.

;-----Calculate the number of turns-----
set a13=a12*a5
set a14=0.030285
set a15=a13/a14       ; a15 (number of turns)
SQR a1, a15           ; a1 is where no. of turns are stored.

;-----Calculations for smoothing capacitor-----
set a2=c4*0.05        ; a2 is the ripple voltage (i.e .5% of Vdc)
set r3=c5/a2
set r4=r3/100         ; r4 is the final smoothing capacitance uf.

```

Figure 3.16 Inductor parameters calculation part.

After the input stage of the program has successfully completed, the procedure general concept of Section 3.4 constructs additional tasks that are required to achieve the second

part. It basically involves the extraction of PSPICE model format and design additional reports using the MDL extraction techniques (described in Chapter 2). These tasks are explained as follows:

### Extraction of PSPICE model

It is the major part of the simulation stage as its purpose to translate data stored in the database into a format that can be read by PSPICE. It basically consists of the circuit involved components and their calculated values. Figures (3.17, 3.18, 3.19), demonstrate the final stage of the program, whereas the final component model is extracted and ready to be simulated.

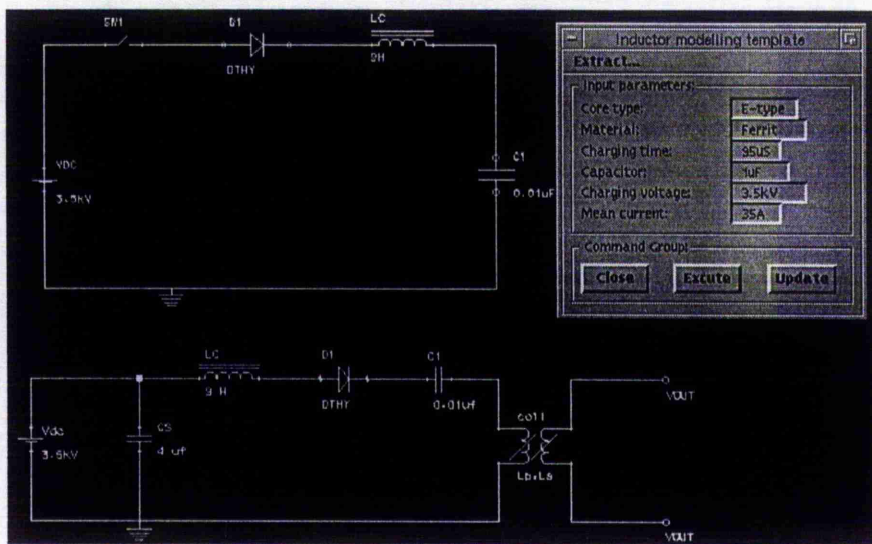


Figure 3.17 Inductor design template.

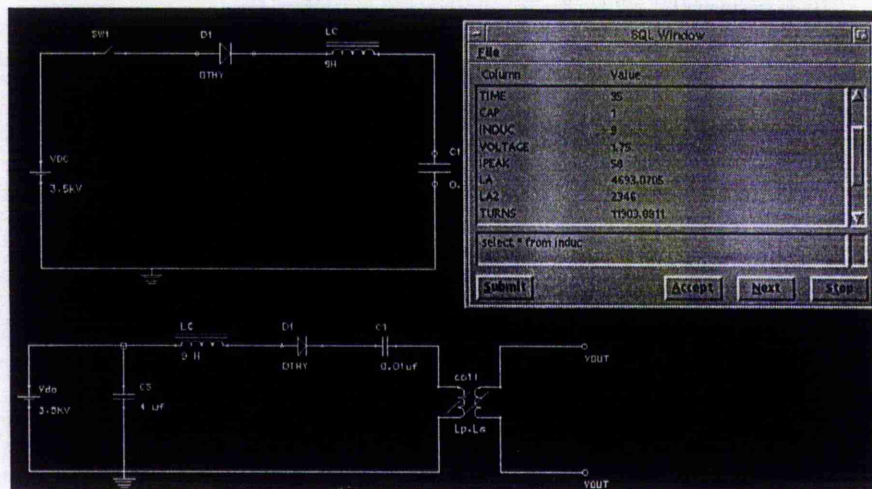


Figure 3.18 Database retrieved information using SQL.



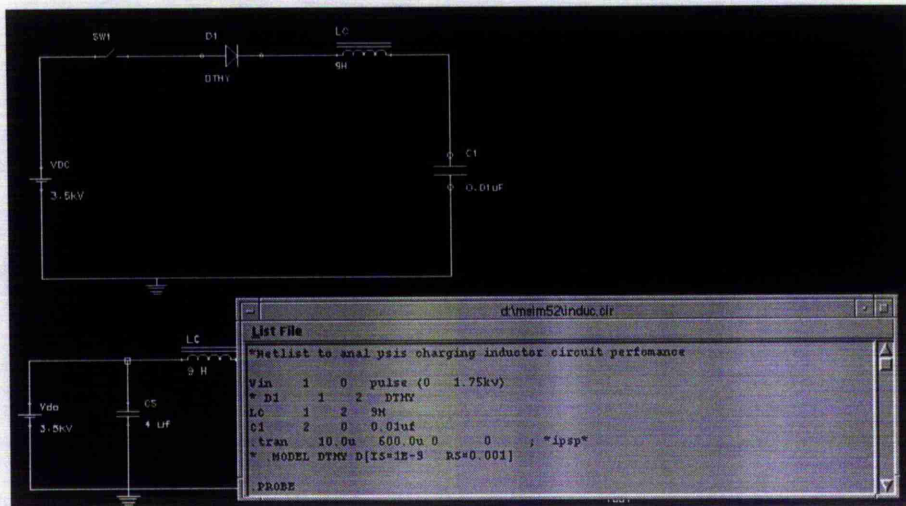


Figure 3.19 The inductor extracted model.

### 3.6.6 Design of a High Frequency Transformer

A transformer basically consists of a magnetic coils built up of insulated material upon which are wound two distinct sets of coils suitably located with respect to each other and the primary and secondary windings. Such a combination may be used to derive a voltage higher or lower than that available (i.e. used to convert a time varying voltage from one level to another) as illustrated in Figure 3.20. It consists of other objects such as the magnetic core, the primary and the secondary winding.

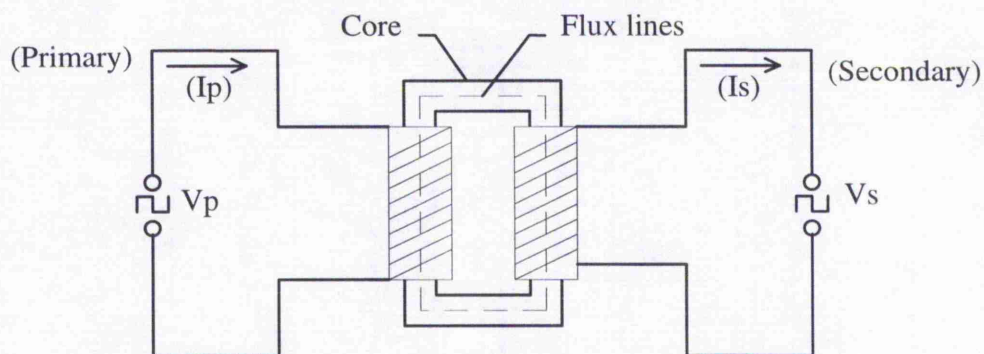


Figure 3.20 The usage of transformer.

The transformer also described by certain functional parameters, such as the input, output power, and efficiency and reactance as shown in Figure 3.21.

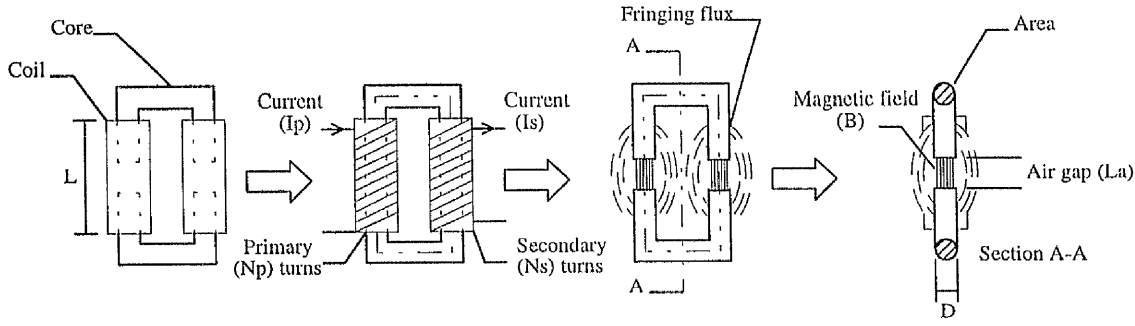


Figure 3.21 Transformer detailed description.

### 3.6.7 Transformer Theory

Assuming the magnetic flux links both coils (i.e. primary and secondary) with no leakage. This is expressed as:

$$\phi_m = \text{constant} \quad (12)$$

Where  $\phi_{\max}$  is the maximum value of flux in the core. Assuming the  $\phi_{\max}$  passing through a circuit. If this flux changes then an electro-magnetic field is generated, which is given by the expression.

$$\xi = -\frac{d\phi_{\max}}{dt} \quad (13)$$

$$\xi_1 = N_1 \frac{d\phi_m}{dt} \quad (14)$$

$$\xi_2 = N_2 \frac{d\phi_m}{dt} \quad (15)$$

$$\frac{\xi_1}{N_1} = \frac{\xi_2}{N_2} \quad (16)$$

$$\frac{V_p}{V_s} = \frac{N_p}{N_s} \quad (17)$$

Sinusoidal excitation:

$$B(t) = B_{\max} \cdot \sin(\omega t) \quad (18)$$

$$\frac{d\phi_m}{dt} = A_c \cdot B_{\max} \cdot \frac{d}{dt}(\sin \omega t) \quad (19)$$

Where:

$$A_c = \text{core area} \quad (20)$$

$$\frac{d\phi_{\max}}{dt} = A_c \cdot B_{\max} \cdot \omega \cos(\omega t) \quad (21)$$

$$V_p = N_p \cdot A_c \cdot B_{\max} \cdot 2\pi F \cdot \cos(\omega t) \quad (22)$$

Thus, the peak primary voltage is:

$$\hat{V}_p = N_p \cdot A_c \cdot B_{\max} \cdot 2\pi F \quad (23)$$

With transformers design it is more common to work in terms of (rms) voltage.

$$V_p(rms) = \frac{\hat{V}_p}{\sqrt{2}} \quad (24)$$

$$V_p = 4.44 \cdot N_p \cdot A_c \cdot B_{\max} \cdot F \quad (25)$$

Using this equation the number of turns and core area required for a given  $B_{\max}$  and frequency  $F$  can be determined.

With square wave excitation the factor 4.44 becomes 4.0 as the rms value equals the pulse value, as illustrated in Figure 3.22.

$$V_p = 4.0 \cdot N_p \cdot A_c \cdot B_{\max} \cdot F \quad (26)$$

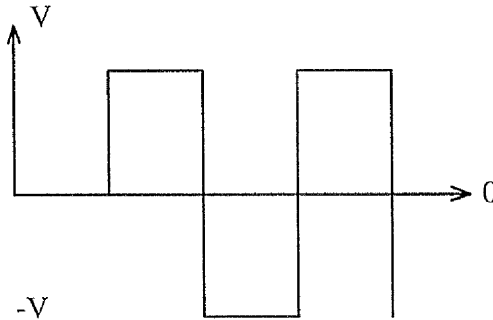


Figure 3.22 Voltage square wave excitation.

equation (26), determines the number of turns and core required for a given  $B_{\max}$  and frequency  $F$ . Clearly, the higher the frequency, the smaller the required core area for a given voltage. Selecting switching frequencies in the kilohertz range (20-100 kHz) offers an advantage in reducing core size. At these frequencies hysteresis losses are very high laminated iron wires. Fortunately magnetic materials have been developed for these frequencies which have reduced losses. These materials called ferrites. Today, practice shows dramatic change from 50Hz linear power supply technology to high frequency switch mode techniques which have improved efficiency, regulation, and reduced size.

### 3.6.8 Problem Definition

Figure 3.23 shows the schematic design of a half bridge forward converter circuit. The transformer has to be designed with respect to the circuit given specification shown in Table [3.3]. Figure 3.24 illustrates transformer and its detailed dimensions.

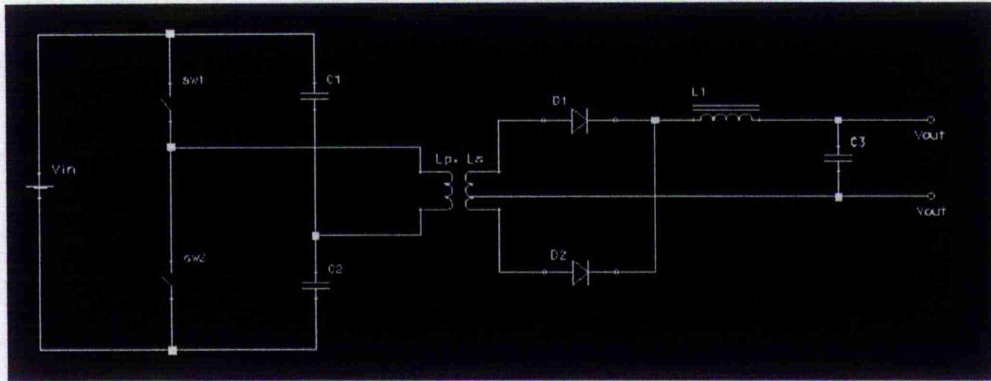


Figure 3.23 Half bridge forward converter design circuit.

Table [3.4]. Transformer input data

Parameter	Name	Value	Units
Input voltage	Vin	28	V
Output voltage	Vo	28	V
Output current	Io	3	A
Frequency	F	20	kHz
Efficiency	$\eta$	0.98	%
Flux density	Bm	0.3	Tesla
Core material	ferrite		
Core shape	C		

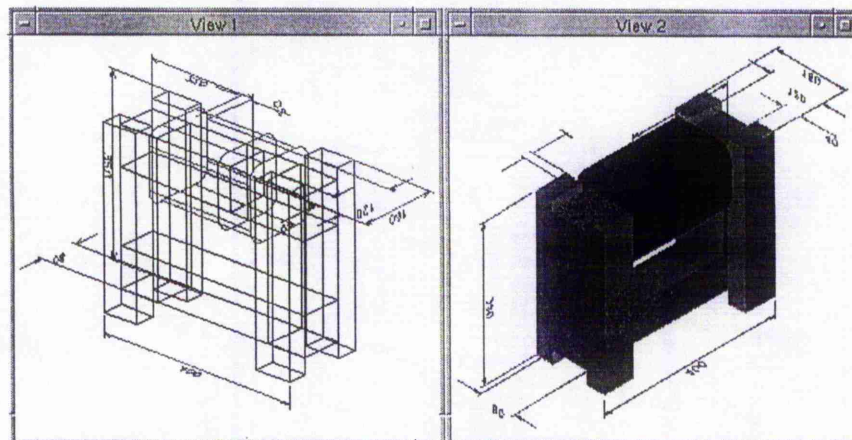


Figure 3.24 Transformer with detailed dimensions.



### 3.6.9 Transformer Theoretical Calculations

Design of transformer for a half bridge converter with a given specification shown in Table [3.3].

#### Step 1.

Calculate transformer output power  $P_o$ , allowing 1.0V drop from diodes.

$$P_o = V_o \cdot I_o \quad (27)$$

$$(V_o + V_{diode}) \cdot I_o = (28 + 1) \times 3 = 87 \text{ W}$$

#### Step 2.

The apparent power  $P_t$  which is associated with the geometry of transformer is of great important. Figure 3.29 shows the transformer primary winding handle  $P_i$ , while secondary winding handles  $P_o$  to the applied load. Since that the transformer has to be designed to accommodate the primary  $P_i$  and  $P_o$ , thus:

$$P_t = P_i + P_o \quad (28)$$

Because of the interrupted current passing through the primary and secondary windings,  $P_i$  increases to 2.828 times (i.e. depending on the circuit which the transformer uses). Therefore, the apparent power can expressed as:

$$P_t = P_o \cdot \left( \frac{\sqrt{2}}{\eta} + \sqrt{2} \right) \quad (29)$$

$$= 87 \times \left( \frac{1.41}{0.98} + 1.41 \right)$$

$$P_t = 248 \text{ Watts}$$

#### Step 3.

The power of handling the capability of a core is related to its area product  $A_p$ . The  $A_p$  is the product area of the available window area  $W_a$  of the area  $A_c$ ; thus the area product can be calculated from:

$$A_p = \text{core area} \times \text{window area} \quad (30)$$

$$A_p = \left( \frac{P_t \times 10^4}{K_f \cdot B_m \cdot F \cdot K_u \cdot K_j} \right)^x \quad (31)$$

The involved factors flux density  $B_m$ , frequency of operation  $F$ , window utilisation factor  $K_u$  which defines space occupied by copper,  $K_f$  which gives the control of copper loss, and the constant  $K_j$  relates to temperature rise, all have an influence on the product area. Thus, by substituting the given values into the equation where  $x$  (depends on core type)  $\Rightarrow$  obtained from table.

$$A_p = \left( \frac{249 \times 10^4}{4 \times 0.3 \times 20000 \times 0.4 \times 323} \right)^{1.16}$$

$$A_p = 0.775 \text{ cm}^4$$

Step 4.

Using standard table to select a core from manufactures data sheet with  $A_p > 0.775 \text{ cm}^4$ .

$$A_c = 0.4 \text{ cm}^2$$

$$A_p = 0.97 \text{ cm}^4$$

Step 5.

Calculate number of turns for each side of transformer (i.e.  $N_p, N_s$ ).

$$N_p = \frac{V_p \times 10^4}{K_f \cdot B_m \cdot F \cdot A_c} \quad (32)$$

$$= \frac{28 \times 10^4}{4 \times 0.3 \times 20000 \times 0.4}$$

$$N_p = 29.2 \Rightarrow 29 \text{ turns}$$

Step 6.

Calculate primary current for the same side.

$$I_p = \frac{P_o}{V_p \times \eta} \quad (33)$$

$$= \frac{87}{28 \times 0.98}$$

$$\therefore I_p = 3.17 \text{ A}$$

Step 7.

Calculate secondary turns.

$$N_s = N_p \times \frac{V_s}{V_p} \quad (34)$$

$$= 29 \times \frac{29}{28} = 30.035$$

$$\therefore N_s = 30 \text{ turns.}$$

Step 8.

Calculate core losses.

for ferrite grade 3c8

$$\text{watts / kg} = 1.01 \times 10^{-3} \cdot F^{1.35} \cdot B_m^{2.12} (w / kg)$$

$$= 1.01 \times 10^{-3} \times (20 \times 10^3)^{1.35} \times (0.3)^{2.12}$$

$$= 50.37 \text{ w / kg}$$

Weight of core.

$$Wf_e = 16 \text{ g}$$

$$Pf_e = \text{watts / kg} \times Wf_e \quad (35)$$

$$Pf_e = 50.37 \times 0.016$$

$$Pf_e = 0.81 \text{ Watts}$$

Step 9.

Calculate copper losses. This is due to winding resistance, therefore, it is needed to select to suit primary and secondary current.

Primary:

$$I_p = 3.17 \text{ A} \Rightarrow \text{choose } 1.25 \text{ mm (around } 3 \text{ A / mm}^2)$$

Length of wire:

$$\text{Wire length} = N_p \times MLT \quad (36)$$

$$l_p = 4.9 \times 29 = 1.42 \text{ m}$$

$$R_p = l_p \times fw (\Omega / m) \quad (37)$$

$$R_p = 1.42 \times 1.39 \times 10^{-2} = 0.02 \Omega$$

Calculate primary copper loss:

$$\begin{aligned} P_p &= I_p^2 \cdot R_p \\ (3.17)^2 \times 0.02 &= 0.2 \text{ Watts} \end{aligned} \quad (38)$$

Calculate secondary current:

$$\begin{aligned} I_s &= \frac{I_p}{\sqrt{2}} \\ &= \frac{3.17}{\sqrt{2}} \\ &= 2.24 \text{ A} \end{aligned} \quad (39)$$

Select wire with diameter of 0.63mm

$$\begin{aligned} l_s &= N_s \times MLT \\ &= 30 \times 4.9 = 1.47 \text{ m} \end{aligned} \quad (40)$$

$$\begin{aligned} R_s &= l_s \cdot f_w (\Omega / \text{m}) \\ &= 5.48 \times 10^{-2} \times 1.47 \\ &= 0.081 \Omega \end{aligned} \quad (41)$$

Calculate secondary copper loss:

$$\begin{aligned} P_s &= I_s^2 \times R_s \\ &= (2.24)^2 \times 0.081 \\ &= 0.406 \text{ Watts} \end{aligned} \quad (42)$$

Therefore, the total copper losses can be calculated as:

$$\begin{aligned} P_{cu} &= P_p + P_s \\ &= 0.2 + 0.406 = 0.61 \text{ Watts} \end{aligned} \quad (43)$$

#### Step 10.

Transformer efficiency is used to measure the effectiveness of the design, which defined as the ratio of the output power  $P_o$  to the input power  $P_i$ . The difference between  $P_o$  and  $P_i$  is due to different losses previously described in Section 3.3; thus the efficiency can be calculated as:

$$\begin{aligned}
 \text{Efficiency}(\eta) &= \frac{P_o}{P_i} \\
 &= \frac{P_o}{P_o + R_s + P_{cu}} \\
 &= \frac{87}{87 + 0.81 + 0.61} \\
 &= 0.983 = 98.3\%
 \end{aligned}
 \tag{44}$$

Step 11.

The minimum size of a transformer is usually determined either by a temperature rise limit, or by an allowable voltage regulation assuming the size and weight are to be minimised; thus regulation is obtained by:

$$\begin{aligned}
 \alpha &= \frac{V_{nl} - V_{fl}}{V_{nl}} \\
 &= \frac{P_{cu}}{P_o + P_{cu}} \\
 &= \frac{0.61}{87 + 0.61} \times 100 = 0.7\%
 \end{aligned}
 \tag{45}$$

Table [3.5]. Transformer output data

Parameter	Name	Value	Units
Trasnformer output power	Po	87	W
Apparent power	Pt	249	W
Area product	Ap	0.775	cm <sup>4</sup>
Primary turns	Np	29	
Secondary turns	Ns	30	
Primary current	Ip	3.17	A
Core losses:			
Weight of core	Pfe	0.81	W
Primary linear dimension	Lp	1.42	m
Primary resistance	Rp	0.02	Ω
Primary loss	Pp	0.2	W
Secondary current	Is	2.24	A
Secondary linear dimension	Ls	1.47	m
Secondary resistance	Rs	0.081	Ω
Secondary loss	Ps	0.406	W
Total copper loss	Pcu	0.61	W
Regulation		0.7	%

## Program Development

The principles of developing this program are similar to those used in developing of the inductor program. Using the circuit specification shown in Table [3.4] as data input, the program proceeds in sequence towards obtaining of the final design stage. The flowchart of Figure 3.25 illustrates the overall process.

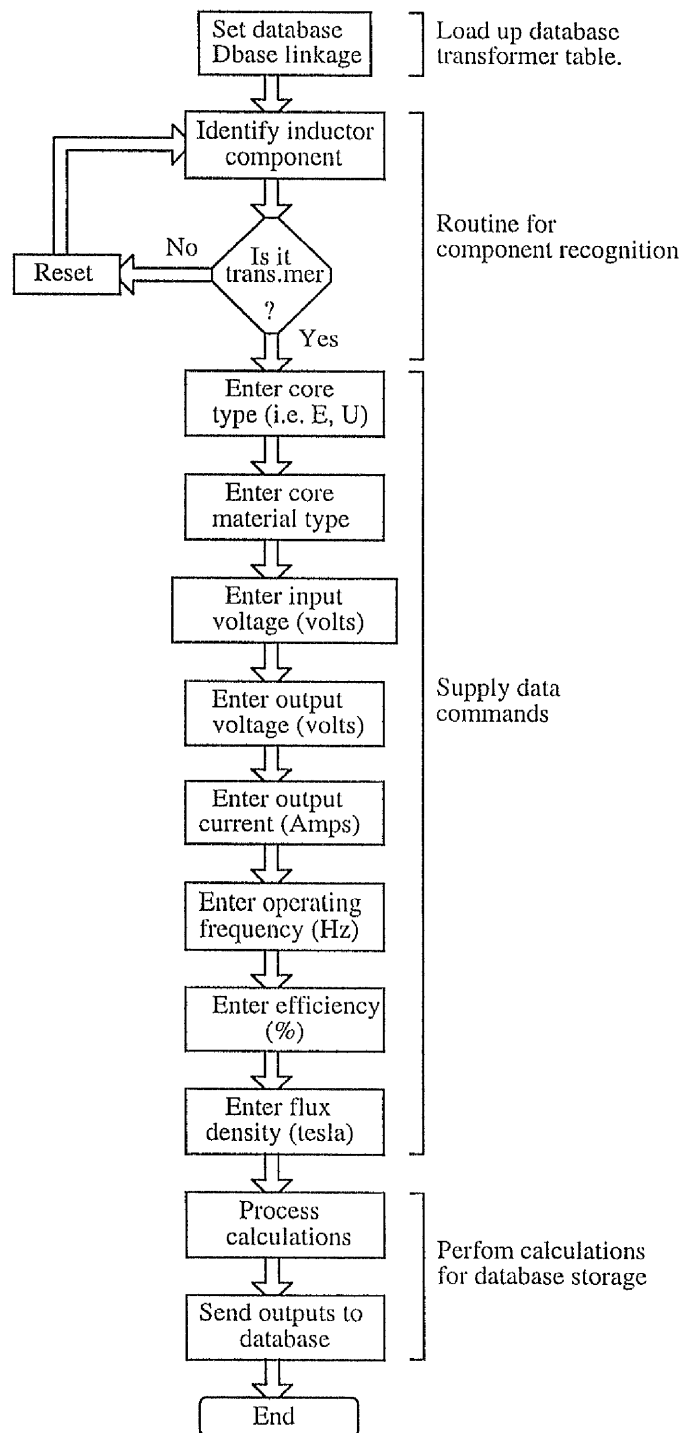


Figure 3.25 Flowchart of transformer design program.

Database table has been designed for this particular design to store the input and the calculated data using the macros UCMs. Figure 3.26 shows these macros defined to calculate the transformer output parameters.

```
;-----Transformer calculations part only-----  
set a0=1  
set a1= c3+a0  
set a2=a1*c4           ; the output power Po.  
set a3=a2*1.41         ; the apparent power.  
set a4=c6+1.41  
set a5=4*c7*c5*323     ; area product  
set a7=a4/a5  
set a8=4*0.3*c5*0.4  
set a9=c2/a8           ; number of turns (primary side)  
set a10=29  
set a11=29/28*a10      ; a9*a9/c2 (number of turns secondary side).
```

Figure 3.26 Transformer parameters calculations part.

### 3.7 DESIGN SIMULATION

The design simulation of the shown circuit was carried to clarify the accurate estimation of parameters using PSPICE package. PSPICE, is commercially available software package designed to facilitate modelling of electrical circuits. The extracted circuit model was used as the input file, which contains the circuit components and their node connections to test the circuit performances with respect to design specification. The output from simulation which includes circuit and voltage waveforms is presented throughout the circuit analysis.

#### 3.7.1 Simulation Results using PSPICE

The design process of the inductor is considered to be an iterative process that requires number of repeated simulations in order to meet the specification requirements. However, the simulation itself is governed by one of the three analysis regimes: (AC, DC, Transient). In this case by considering the peak current  $\hat{I}$  and voltages  $V_{DC}$ ,  $V_o$  as variables of interest, the transient analysis has been performed. Figure 3.27 shows a display of the 2D schematic environment and the PSPICE package.

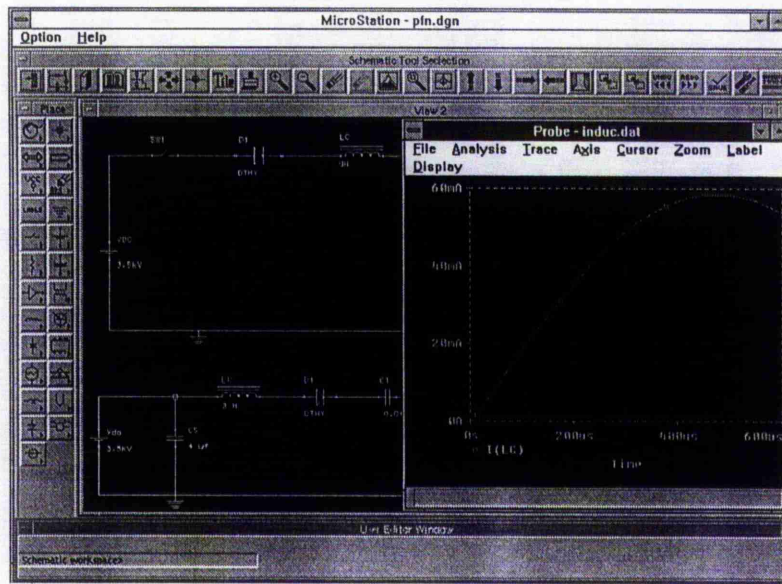
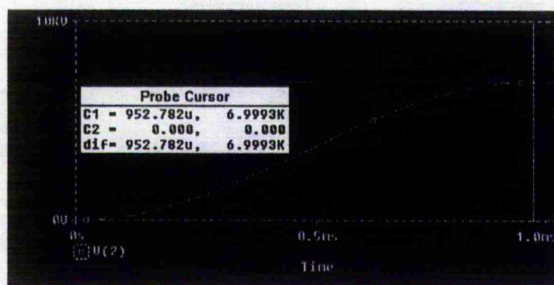
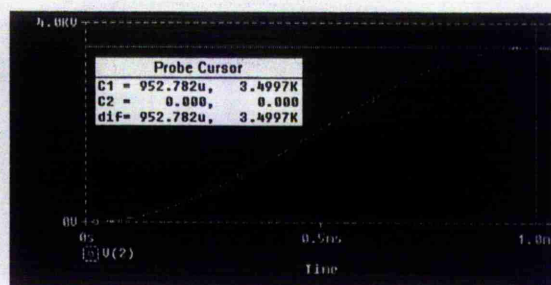


Figure 3.27 Display of PFL circuit and PSPICE.

The charging inductor circuit is connected to pulse voltage source  $V_{DC} = 3.5V$  with charging time of  $\tau_{max} = 95\mu s$ . From equation (3), the supply voltage for resonant charging is expressed as ( $V_o = 2V_{DC}$ ), therefore, a comparison can be made by applying the two different values of the supply voltages. Figure 3.28a shows voltage-time wave form representation across the inductor when  $V_{DC} = 3.5V$  which found to be almost double (i.e. 6.999kV), while Figure 3.28b reveals the same result when  $V_{DC} = 1.75kV$ , which found to be 3.4997kV.



(a)



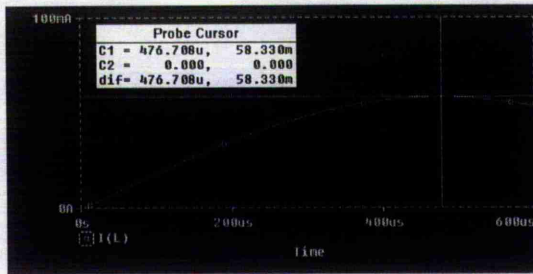
(b)

Figure 3.28 Transient response of V(0) when  $V_{DC} = 3.5V$  and 1.75kV

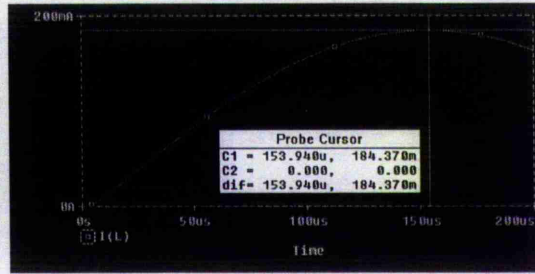
Using the calculated values of  $L_c = 9H$  and  $C = 0.01\mu f$ , the circuit is simulated to obtain the peak charging current  $\hat{I}$  which found to be 58.330mA approximately similar to the calculated value of equation (3), as shown in the obtained waveform for  $\hat{I}$  versus  $\tau_{max}$  of Figure 3.29a. During the simulation procedure it was noticed that reducing the calculated value of  $L_c$  to smaller size of  $L_c = 0.9H$ , will increase the value of the peak current  $\hat{I} = 148mA$  and a charging cycle of shorter duration as shown in Figure 3.29b. This will effect inductor losses as they will be increased due to the higher rms changing



current and increased core losses at higher frequency. Nevertheless, provided ripple currents and core losses are acceptable, using an inductor smaller than the maximum allowable can give a substantial cost saving.



(a)



(b)

Figure 3.29 Peak current response for  $L_c = 9H$  and  $0.9H$

### 3.7.2 Simulation Results using TK solver

The Tool Kit Solver (TK Solver) is a software package that has been introduced in 1982, it has evolved engineering tasks ranging from equation solver software product to an integrated problem-solving and knowledge management environment. It is windows-based software uses standard windows navigation and allows the users to interact in a flexible way. Like most windows applications, its working environment consists of two open windows referred to as variable sheet and the rule sheet as shown in Figure 3.30.

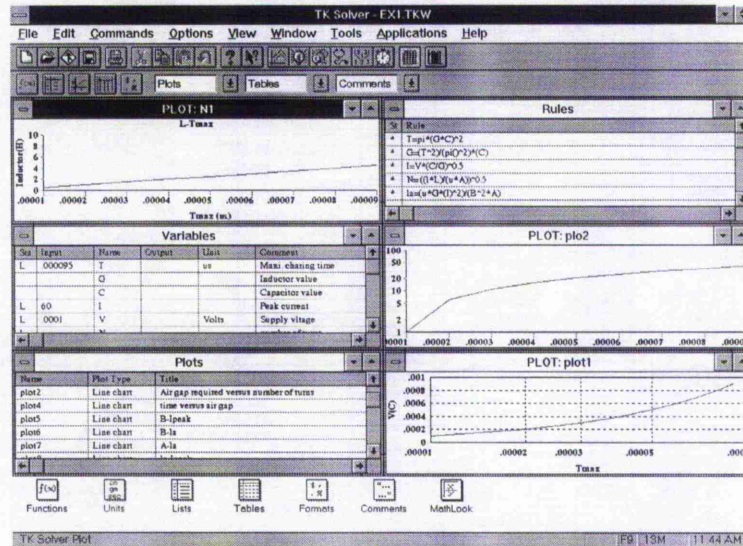


Figure 3.30 A display of TK solver interface.

The rule sheet contains the relationships between variable and other related values, while the variable sheet restricts itself to the declaration of input and output values by assigning to the status of each variable. Other sheets contain facilitates such as unit conversion,

plots definition, and user defined function etc. It used here to investigate the design problem defined previously and the results are graphically presented and compared to those obtained by PSPICE.

Using the mathematical equations described in the theoretical section, the results obtained by PSPICE are compared to those obtained by TK solver, using different design considerations. The focal points of interest were the accuracy of obtaining the output parameters. The results show that values found using PSPICE are almost equal to those achieved by TK solver. Iterative procedures and a varied set of graphical presentation are produced from a single behaviour. The results also confirm that the application of an iterative methods demonstrates the greatest efforts in allowing more optimisation opportunities. The figures which follow graphically illustrate the results obtained by using the TK solver software. Figures 3.31 and 3.32 show voltage, current versus charging time respectively.

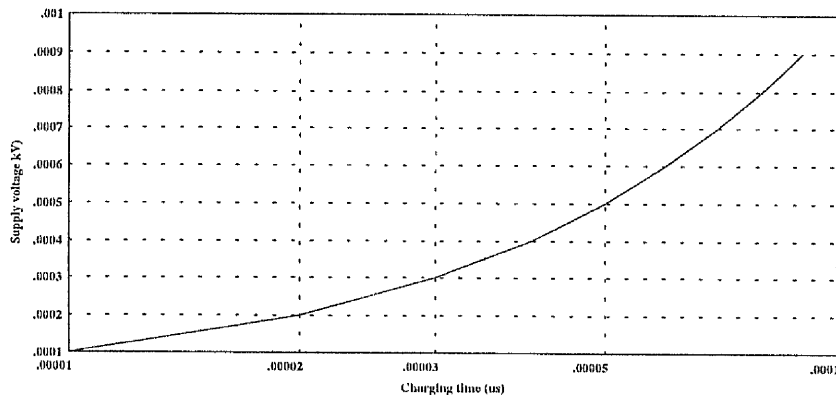


Figure 3.31 Voltage versus maximum charging time.

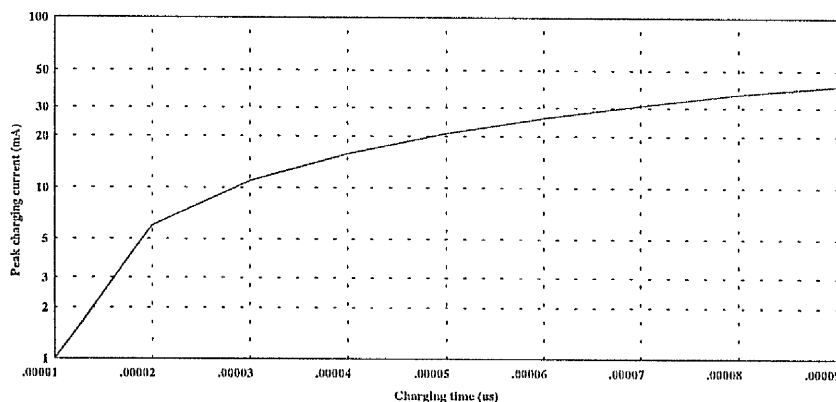


Figure 3.32 Peak current versus maximum charging time.

The narrow space between the two elements, known as the air gap, it is the critical region of the component, and the theory is mainly concerns the conditions in or near the air gap. The magnetic flux is distributed throughout the core, but because of the high

permeability  $\mu_o$  of the material involved, the air gap required to prevent saturation can be calculated as illustrated in step 5 of the theoretical calculation section. Figure 3.33 shows air gap required against number of coil turns.

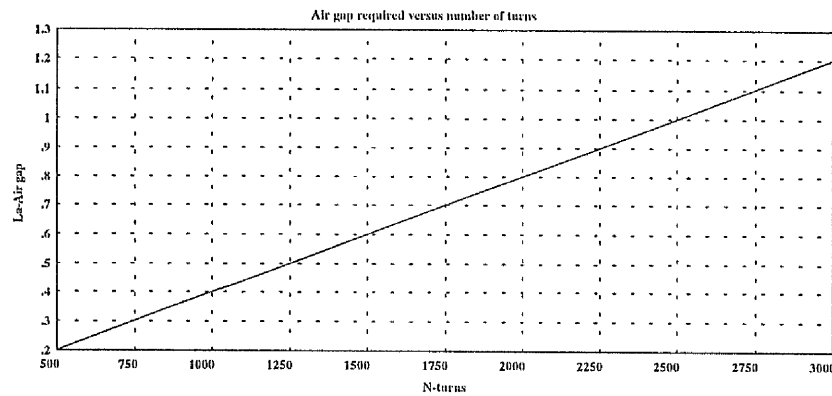


Figure 3.33 Air gap required versus number of turns.

Figures 3.34, 3.35 shows the flux density  $B$  is linearly dependent on the peak current  $\hat{I}$ . The chosen value of  $B$  was 0.1 tesla (for ferrite material). This suggests that a care for determining the maximum value of  $B_{\max}$  is needed, so that, the magnetic saturation can not be produced.

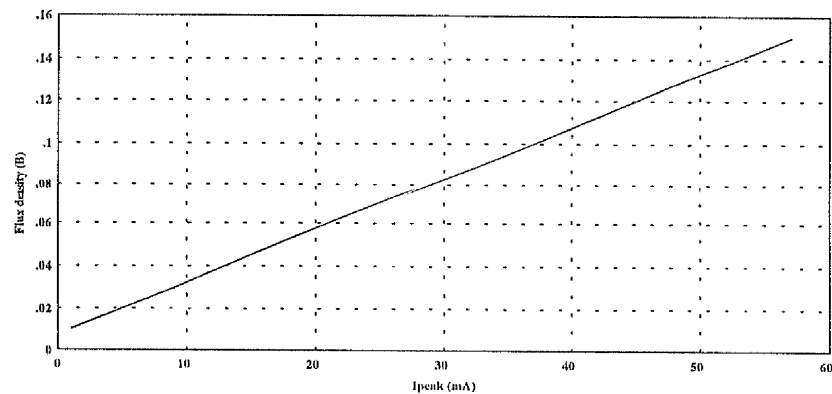


Figure 3.34 Flux density versus peak current.

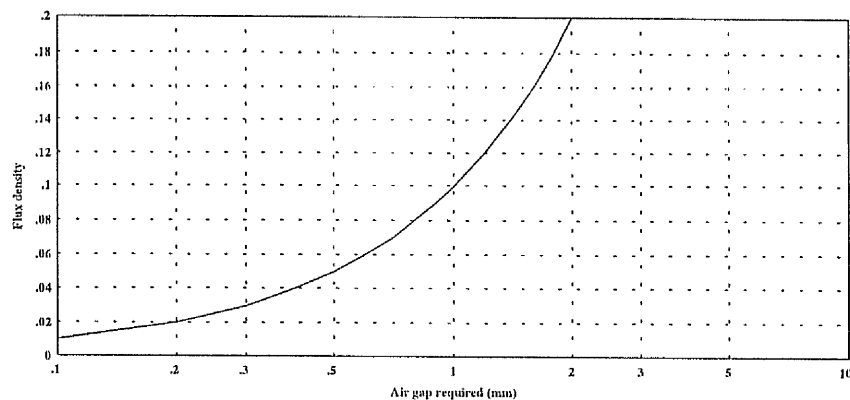


Figure 3.35 Flux density versus air gap required.

### **3.8 CONCLUSION**

This chapter has examined the tools necessary which can be used to support the integration of different design environments. Design examples have been provided as illustration of the relationship which exists between design activities, and hence environment designed to support these activities. In this case, the integration of the various environments was necessary to ensure that the complete approach is supported in a convenient, flexible manner. This chapter has examined the motivation and objective of this issue by considering the three different environments schematics, database, and PSPICE as the areas of interest, with particular emphasis on the key aims of integration upon which this approach is founded.

As part of the investigation, this chapter has also shown the use of circuit modelling languages and their role in circuit performance modelling. By using the fact that simulation packages such as SPICE to predict design parameters, a two part software program has been employed. The key element to this process is the user interaction with the standard database dbase (iv). This is required, as the process to build the model is completely based on two a way procedure (i.e. insert to store and extract to simulate).

This chapter has shown the design process of component within each electrical/electronic circuit is an iterative process which requires a repeated simulations, depending upon design specifications. Building component models for simulation was the other issue in investigation this approach. Once these models were built by means of an automated process, further applications such as circuit optimisation and simulation are carried to justify component design.

### **REFERENCES**

- [3.1] Von H. M., Swart P. H., "Thyristor-driven pulsers for multikilowatt average power lasers," IEEE Proceedings-B, Vol. 139, No. 2, pp. 123-130, March 1992.
- [3.2] Resve S., and Andrew y., "Simulation and Modelling-SPICE2 Voltage-Programmable Transfer Function Modelling," IEEE Circuit and Device Journal, Vol. 9, No. 1, pp. 8-14, January 1993.
- [3.3] MicroSim corporation, "PSpice Circuit analysis user's guide," 20 Fairbanks, Irvine, California 92718, Version 5.0, July 1991.

- [3.4] Graeme R. B., et al., "Macromodelling of Integrated Circuit operational amplifiers," *IEEE Journal of solid-state Circuits*, Vol. SC-9, No. 6, pp. 353-363, December 1974.
- [3.5] Farid N., "Simulation and Modelling," *Circuit and Devices Journal*, Vol. 143, No. 3, pp. 12-15, July 1996.
- [3.6] Engel J., et al., "Design Methodology for IBM ASIC Products," *IBM Journal Res. development*, Vol. 40, No. 4, pp. 387-405, July 1996.
- [3.7] Hands J., "What is VHDL," *Computer-Aided Design Journal.*, Vol. 22, No. 4, pp. 246-249, May 1990.
- [3.8] Foulk P. W., "CAD in electronics," *Computer-Aided Engineering Journal*, Vol. 16, No. 3, pp. 166-171, May 1984.
- [3.9] Analogy system reference guide, release 3.2 , Analogy Inc. , August 1993.
- [3.10] OrCAD, Electronic Design Automation Tools, "Programmable Logic Design Tools," user guide, Document No. OR9066B6-13-91, 1991.
- [3.11] Lowther D. A., et. al. "The application of expert systems to CAD in Electromagnetic," *IEEE Trans. on Magnetic*, Vol. MAG-21, No. 6, pp. 2559-2566, November 1985.
- [3.12] The Cutting Edge., "The newsletter for saber user," *Analogy*, Vol. 7, No. 1, pp.1-9, April 1994.
- [3.13] Nobuhiro K., and Hirosuke O., "Mechatronics-An Introduction Perceptive," *IEEE/ASME Trans. on Mechatronics*, Vol. 1, No. 1, pp. 10-15, March 1996.
- [3.14] Rolf I., "Modelling and design methodology for Mechatronics Systems," *IEEE/ASME Trans. on Mechantronics*, Vol. 1, No. 1, pp. 16-28, March 1996.
- [3.15] Saldanha C., and Lowther D., "Automating the design process for electromagnetic devices," *Computer-Aided Engineering Journal*, Vol. 3, No. 5, pp. 173-179, October 1986.

- [3.16] Ting k., et al. "CAD-Integrated engineering-data-management system for spring design," *Robotics & Computer-Integrated Manufacturing*, Vol. 22, No. 3, pp. 271-281, 1996.
- [3.17] Colonel W., M., "Magnetic core selection for transformers and inductors," A user guide to practice and specification, ISBN 0-8247-1873-9, 1982.
- [3.18] McDonald D., W., "10 kHz pulse repetition frequency  $CO_2$  laser for processing high damage threshold materials," PhD. Thesis, Department of Mechanical Engineering, Glasgow University, October 1989.

# 4

## Automatic Netlist Generation

### 4.1 INTRODUCTION

Electrical circuits require accurate methods of assessing circuit performance, prior to production release. For the design of discrete circuits, it is normal practice to breadboard and test circuits [4.1]. However, some problems may not be found during breadboard testing but will show up on production circuits. The circuit can be probed to isolate the causes of a failure, and designs can be modified. The disadvantage of full breadboard is the time taken to produce prototype circuits. It is therefore very advantageous to analyse systems performance by computer simulation.

During the design of a digital or "mixed" systems, the simulation process is carried out at several levels, each level yielding information on different aspects of the design. Circuit level simulation is considered as the lowest level used in the design of digital systems. Circuits simulator are capable of performing both A.C. and D.C. linear and non-linear analysis, transient analysis, and Monte Carlo Analysis. The circuit to be analysed is described to the simulator using a special circuit description format which defines the type of component, its value and connectivity. This file is referred to as netlist. Most printed circuit board (PCB) packages use netlisting of different formats that are used for both simulation and circuit layout.

The first step in running a simulation is to generate a netlist file of circuit schematic design. Netlists are defined as a collection of modules interconnected by electrical nodes. Modules represent the primitives of the target used in implementing a particular design. Each module has an associated set of pins representing the interface of the module to the rest of the netlist. Each electrical node has an associated set of module pins that are

connected to it and carry the same signal. Each pin is connected to exactly one electrical node.

This chapter deals with the investigation of automatic netlist generation for electrical-electronics circuit simulation. It has been considered as the key element in carrying out circuit design performance. Therefore, previous attempts are mentioned as part of literature and different generating techniques along with general rules and conditions are discussed.

## **4.2 THE IMPORTANCE OF CIRCUIT NETLISTS**

Netlisting is an important issue in many steps of electrical/electronic design, analysis, and manufacturing. It can be used to trace electrical path of a given signal. This path will then identify the original of the signal and those components that use the signal as an input. Therefore, the ability to visualise a circuit by netlist is important in understanding the paths between components. Netlist is used in simulation because, viewed abstractly, a circuit is a collection of gating components and their connections. It ignores the passive components and the actual geometry of the circuit layout.

## **4.3 RELATED WORK**

Currently automatic tools produce netlist that are inferior to those done manually. According to Evangelos [4.2], the effective network simulation, analysis, problem diagnosis, and redesign require three types of circuit knowledge:

- Knowledge about parts hierarchies, is used for specifying that a component is a part of another component or of a circuit network. It also contains the attributes of each component, relations between the attributes, and the connectivity information between the components. It is used by the simulation in order to derive the behaviour of a component and the entire circuit network.
- Knowledge about component categories, used to establish information inquire about component hierarchies, features, and connectivity relationships.
- Constraint knowledge, used for placing timing and signal strength restrictions on the network.

Derivation of the circuit's behaviour for the purpose of simulation is then done by circuit simulators. PSPICE is one example of a commonly used simulator that uses the knowledge of parts hierarchies that identifies the low level of components of a device (e.g. transistors, resistors, etc.) and their connectivity.



Design verification tools such as simulators, and timing verifiers usually require the circuit be described as a flat netlist consisting only of invisible modules whose pins are identified with specific electrical nodes. Netlist compilation of hierarchical schematics is defined as the process of extracting the flat netlist underlying a hierarchical circuit representation, and is therefore an important component in any integrated design system.

Larry [4.3], presents techniques for creating the netlist for hierarchical schematics design. The techniques are used for creating the netlist as a data structure for further on-line processing or for creating a file use with other off-line design tools that are downstream from the compilation process. Figure 4.1 gives an example of hierarchical representations design tree for a full adder implemented using circuit logical gates (i.e. and, or, xor) as the target primitives.

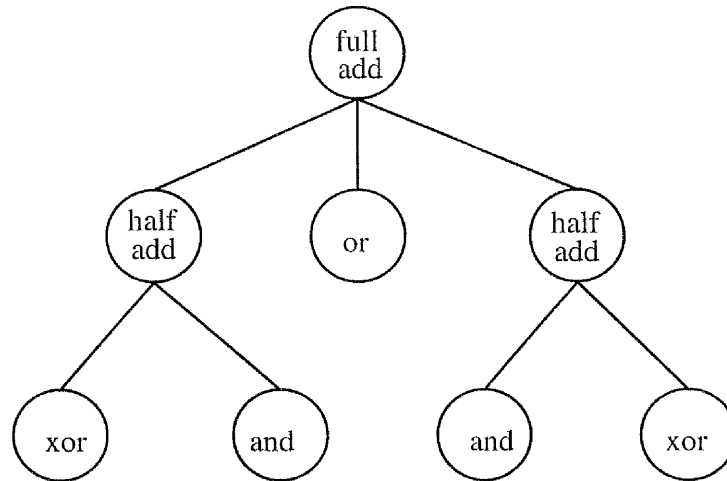


Figure 4.1 Design tree of a full adder [4.3].

Each component has an associated schematic, that describes the construction of the component from other lower-level components in the tree. The term schematic here is used for a component that consists of a set of sub components, along with a set of nets. Sub components are reference to lower level components, while nets represent electrical nodes, each carrying a signal that is relevant to the design of the component.

Netlists and translators are limited, because they contain a small part of the information in complete design database [4.4]. The electronic design system OrCAD, uses an incremental database. The advantage of using incremental database is that speeds up the design process, especially during revision, verification, and maintenance cycles for hierarchical design. Figure 4.2 shows the process of creating a netlist. It is composed of three main steps namely: (compile, link, and format). As in the process of creating a computer program, the schematic (source) files are processed by the compiler into an

intermediate form, and are linked to produce a file that contains all of the connectivity information (i.e. connectivity database). Finally, the connectivity database is then translated to produce two different formats.

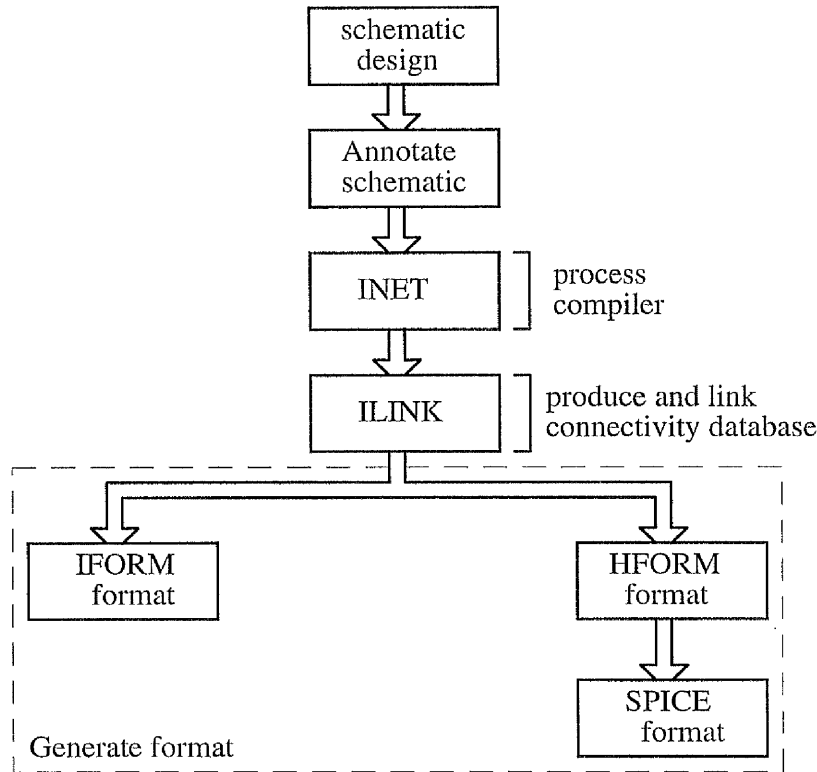


Figure 4.2 3-steps process of netlist creation using OrCAD system [4.4].

IFORM for making flat netlist, and the other is HFORM for making hierarchical netlist (e.g. SPICE). The file compiler (.INET) creates the incremental database and produces two different files : .INF, for each sheet in the design, .INX for the entire design, its purpose is to speed processing and organise the database. The linking command (ILINK) is used to create the incremental netlist structure. This consists of instance (.INS), resolved (.RES), pipe (.PIP), and linked connectivity file (.LNF). This process is illustrated in Figure 4.3, where is the final form in producing the netlist is reached. Each type of formatter (i.e. IFORM, HFORM) reads its associated files directly and produces the final form of netlist, as shown in Figure 4.4.

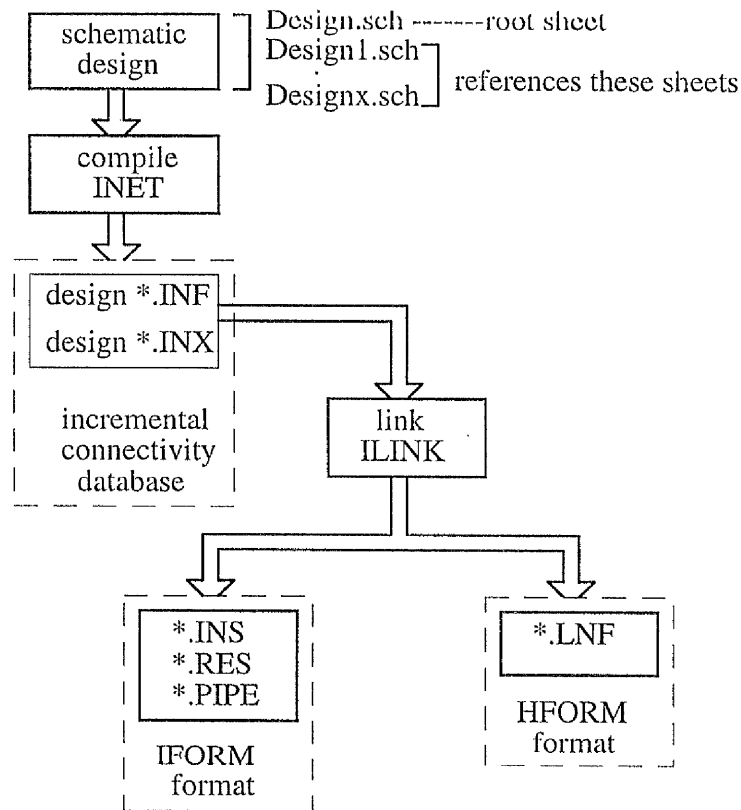


Figure 4.3 Files generated by both compiler and linker [4.4].

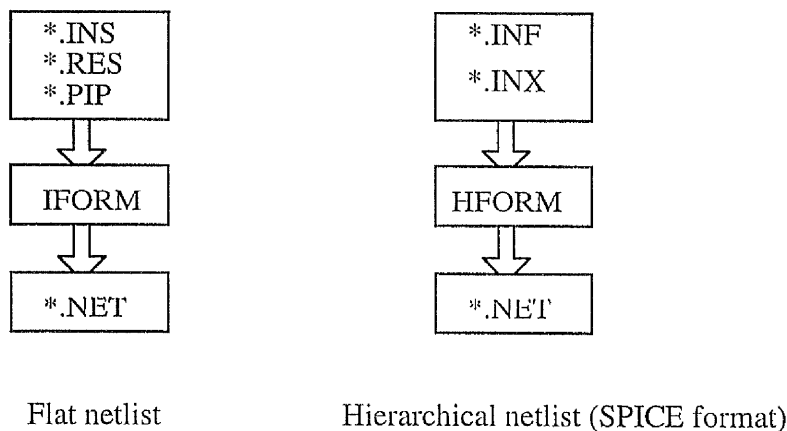


Figure 4.4 Final stage of netlist format files (flat, hierarchical netlist) [4.4].

A schematic representation of a physical system is very effective in illustrating the system's function and structure. The analogy system Saber [4.5] uses different techniques in generating a netlist for simulation. The schematic design in this case must be constructed into a medium that the computer can interpret. This medium is an input file that contains a description of the components that constitute the system and they are interconnected. Figure 4.5 shows generic procedure for generating a netlist using saber system. A netlist is assembled independently from saber simulator by using word

processor that creates ASCII file, but it also may be extracted automatically from a schematic database file.

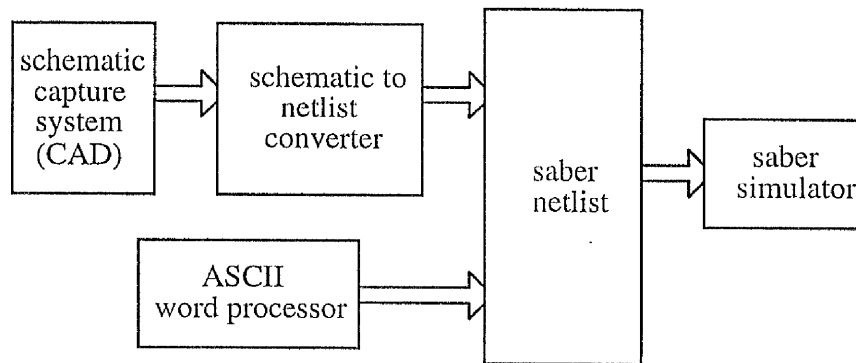


Figure 4.5 Creating a netlist in saber system [4.5].

The problem of entering design into CAD systems at the behavioural level has received much attention in recent years. To make the design process interactive, a tool that translates a structural level description into a schematic diagram is necessary. A schematic gives better insight into designs and also facilitates design documentation. For these reasons, highly interested motivations towards the development of automatic schematic generation systems is in progress. The point to be made here is that netlists are used for the automatic generation of schematics as well as circuit simulation.

Many design automation firms are offering automatic schematic generators as part of synthesis tools. The Autodraft system developed by Majewski [4.6], is a program intended for a silicon compiler environment where the schematics produced for modular portions of a design. A generic netlist text file produced by the AutoCircuit generators provides a circuit model for the schematic synthesis. Naveen [4.7], also describes a CAD system used to improve the speed of a schematic generator. The system's key attributes are an integrated design environment and a consistent user interface. It consists of three main components: The first component reads the netlist and graphical information about the modules. The second component places all the design modules and routes the nets in a readable way. The third component writes the generated schematic information into the system global database.

### 4.3.1 The need for Computer Aided Circuit Analysis

Computer-aided circuit analysis (CACA) first became popular in the mid of 1960's when the electrical computer-aided program (ECAP) was invented and made available. It made it possible for an engineer to measure analyse performances of an electronic circuit without having to write equations defining the circuit.

The necessity of circuit analysis is that once a design for a circuit has been determined, the design must be tested to ensure that the circuit gives the desired performances. This often involves testing for DC operating point and performances with signal applied, over a range of DC supply voltages, and temperature.

### **4.3.2 PSPICE Background Information**

A computer program that simulates the electrical performance of an electronic circuit circumvents many of the practical problems that are encountered in circuit characterisation [4.8]. The circuit is preformed in mathematical terms, and numerical analysis procedure that correspond to typical laboratory measurement are performed.

Programmable simulation program for integrated circuits and emphasis (PSPICE) performs a task that allows the user to simulate and test a circuit designs for both analogue, digital components and mixed systems without having to build the hardware equivalent. Response over time to different inputs, different frequencies, the noise, and other circuit details are conveniently calculated. Another aspect of using PSPICE, is that it allows the designer to create a "computer breadboard" of a circuit for testing and refinement before actually having to fabricate it.

Accurately verifying the performance and functionality of circuit designs prior to fabrication is an essential task in producing integrated circuits (ICs) in a timely and cost effective manner. The SPICE has taken a key role in making this verification a reality since the 1970's [4.9].

Circuit simulation is one of the most critical and time consuming computational tasks to be performed in integrated circuit design [4.10]. Electrical/electronic circuits design requires extensive, accurate simulation for performance evaluation. Not only must the circuit be simulated under nominal conditions, but it must also be simulated under a variety of operating conditions including extremes of both temperatures and supply voltage. Additional simulations must performed in order to assure manufacturability and to guarantee that the circuit will be functional and meet specifications over a wide range of process variations.

### **4.3.3 Circuit Definition in PSPICE**

The program input defines the circuit to be simulated on an element by element basis [4.8]. Elements included in PSPICE package such as resistors, capacitors, inductors, coupled inductors, independent voltage and current source, and common semiconductors

devices: the diode, the bipolar-junction transistor (BJT), the junction field-effect transistor (JFET), and insulated-gate field-effect transistors (IGFET or MOSFET).

SPICE input file "i.e. netlist" consists of set of commands that defines a circuit. The first command is the title command; the contents of this command are printed as the heading for the various sections of the results output. The last command usually defines the .END (i.e. control line) which serves only to signify the end of input file. Typical format of PSPICE input file is shown in Figure 4.6. Each element in the circuit is defined by an element command that contains the element name and reference, nodes that connect elements, and the values of the parameters that define the electrical characteristics of the element.

In addition, PSPICE specifies the analyses to be performed and the output to be generated. This information is entered on control commands. Typically, the simulation of an electrical requires a combination of three basic analyses: D.C. analysis, time domain transient analysis, and small signal A.C. analysis. In addition, PSPICE contains several sub analysis capabilities such as Time-Domain response, Fourier analysis, Noise analysis.

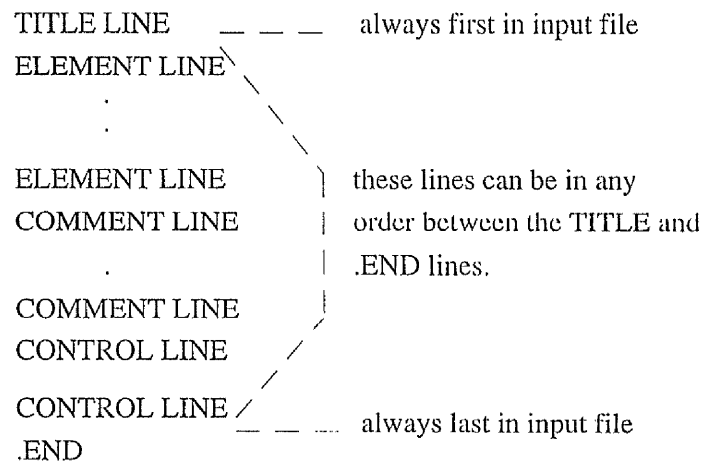


Figure 4.6 Generic PSPICE input file format [4.11].

PSPICE reads the input file and figures out all the circuit elements connected to each node. It then uses Kirchoff's current laws to create a system of equations for the circuit, where the voltages at each node are the unknowns, and the admittance of each branch connecting two nodes are the known quantities.

#### 4.3.4 Process for using PSPICE in Electronic CAD

In most engineering application environments there are a few production programs for which turnaround times map almost directly into the productivity of the engineer [4.12].

It is often the case that an engineer must wait for the completion of that program before doing what ever is next. The PSPICE package, is one such production program for electronic CAD environments.

The circuit simulators are the most heavily used tools in an electronic CAD environments. A descriptions of the circuits, its underlying device characteristics and its initial state are used to simulate circuit operation by evaluating device conductances and node voltages over time. The value of such simulator is obvious, since it allows the designer to test the circuit many times before it is actually realised. The generic process of using PSPICE to analyse an electronic circuit is shown in Figure 4.7.

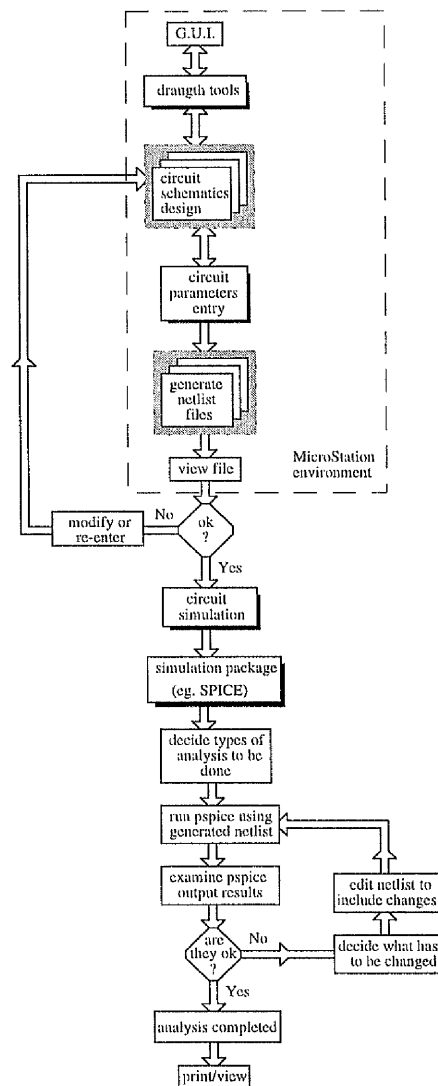


Figure 4.7 Process flowchart of using PSPICE.

Circuit schematic diagram is drawn using (MicroStation-system environment), and components parameters are defined. This is done by giving each component, or circuit

element a name and value. For the PSPICE analysis process, components must have a different name. The tool to generate a complete netlist format is selected format is described in Section 4.2.4. In addition, detailed investigation of automatic netlist generation including elements node assignment is discussed in Section 4.3. The netlist file is then reviewed from within MicroStation or by using DOS operating command as an ASCII format. This is where the extracted netlist file is checked and decision is made whether to proceed with simulation or modifying circuit components parameters (i.e. reference and value). Figure 4.8 shows the link between the system environment and simulation package (PSPICE).

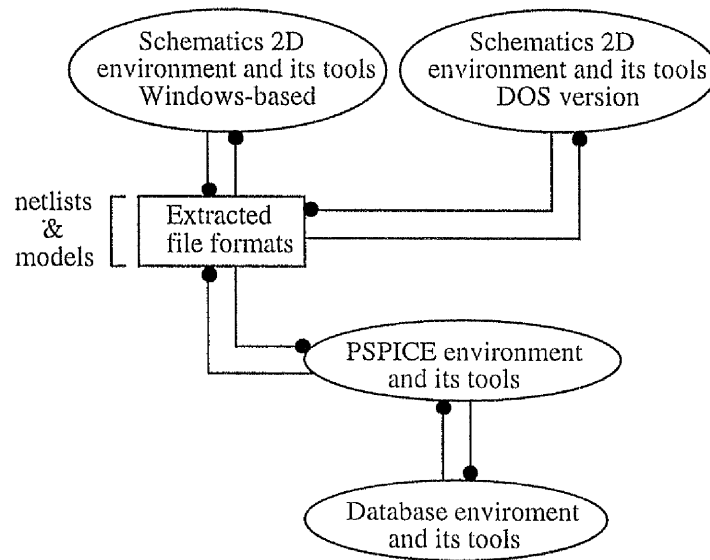


Figure 4.8 The link between schematics and PSPICE environment.

Once the netlist compiled, it is time for simulation to take place. As it has been mentioned in Section 4.2.4, the simulation of an electronic circuit requires a combination of three basic analysis namely: (DC, AC and transient). Based on these types, one or more control lines will have to be added to the input netlist file. A control line could specify the values of DC voltage or current for a source, the range of frequencies on AC source is to have, or the time interval over which a transient analysis is to take place and time step size to be used. The function to run simulation is selected, which will perform a circuit analysis and creates an output file. This is used to examine the analysis results, and decision is made if these results are not satisfactory. If this is so, the circuit is revised either by modifying the circuit schematic or by editing the netlist file. To reflect the changes made PSPICE is run again until the desired result is achieved.

#### 4.4 NETLIST DESIGN DEVELOPMENT

The problem of circuit extraction is to take a schematic layout and determine the circuit connectivity, and obtain estimates for various electrical parameters. The circuit extraction



problems has two main aspects: netlist extraction and parameter extraction. Netlist extraction involves the determination of the electrically connected regions (i.e. nets). Parameter extraction involves the extraction of reference designation and components values.

This section examines the impacts of different methods of circuit netlist automatic extraction. It basically deals with extracting the necessary information that can be used and assist in developing an automated techniques. Netlist is a description of a circuit that lists each component and its values in the circuit and how it is interconnected with other components. Therefore, netlist development involves identifying cells (components), wiring connection, finding interconnections, propagating the signal and labels through the circuit, and the use of software scanning functions.

Figure 4.9 of this section, gives an overview of the wiring relationships between different type of components that provides the circuit data structure. The central data structure in this development is the cells that represent components and the wires that connect these components. A cell describes the pins and other physical and functional attributes. Lines which represents wires connected with other components via nodes (i.e. where two or more components are met).

In addition, as cells are connected by means of wires, the data structure represent this connection are stored in the MicroStation (CAD engine). By using the provided software (MDL), these data are extracted and used as an input for further programming development. Both cell data and wires lists contain enough information to specify completely the connectivity. However, the combination of both lists simplify the connectivity recognition. For instance, to achieve a common electrical node connection between R1, R2 as shown in Figure 4.9a, the start and end for wires co-ordinates must match one side of both components pins. Description of software programs developed to recognise connectivity and netlist generation are discussed in this section.

#### **4.4.1 Problem Definition**

The netlist problem is a problem of assigning a given number of circuit components to positions on the circuit by means of electrical nodes. Given a circuit schematic diagram, a netlist file shows components connections, their relevant references and values have to be automatically extracted for PSPICE simulation. However, from the study of various drawn schematics, the following set of considerable characteristics are derived:

- Most of the signal flow is from left to right and top to bottom.

- Symbols representing circuit components are placed first.
- Wiring between components is assigned from left to right or inverse.
- A node is a point in a circuit which is common to two or more circuit elements.
- A junction or principle node is a point in the circuit where three or more elements are connected together.
- A branch is a path containing one circuit element, and which connects one node to another.
- Space should be left around each graphic symbols for reference designations and other identification material, and also to avoid overlapping.
- Netlisting takes place within the span of the net's pins, for easy tractability.

#### **4.4.2 General Development Rules**

The rules and assumptions considered in developing the netlist are as follows:

- The system drawing environment is considered as a blank pad onto which components and wires are placed in random order.
- There must be wire connections between every component. (i.e. no floating lines).
- The data generated from the circuit shows the sequence of input components and wires.
- There can be no electrical connections between of the items in the sequence. They are placed in order for the convenience of spatial arrangement.
- The IC chips that would be used will have a limited number of pins.
- There are possibilities of more than two connections in parallel to form an electrical path.
- There is only a single layer of circuit.
- There must be more than two components sharing the same node (i.e. junction connection) as in parallel connection.

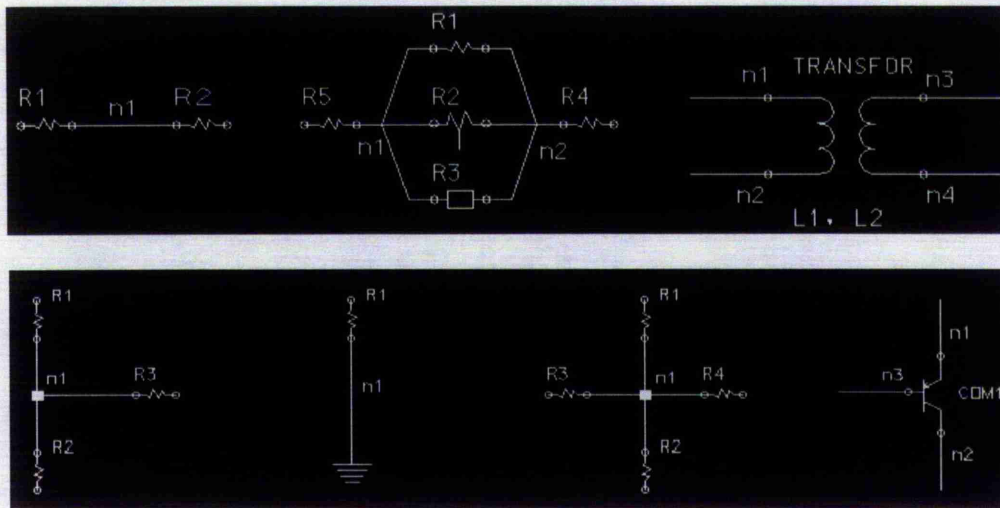


Figure 4.9 Illustration of circuit connection possibilities for netlist generation.

#### 4.4.3 Connectivity Establishment

More specific to very large scale integration (VLSI) is the notion of connectivity. Electronic components have wires coming out of them to connect to other components. Every component is therefore interconnected with all other components through some path. The collection of paths through a circuit is its topology.

Connectivity is considered as distinguishing characteristic in electrical/electronics design. Since that electronics involves the movement of charge, components are viewed in terms of how they move signals back and forth across wires [4.13]. Network is merely collections of nodes that are connected with arcs. In most schematic and IC design systems, connectivity is essential to layout. Components must be wired together to form a completely connected circuit. Figure 4.10 shows a circuit as connected components.

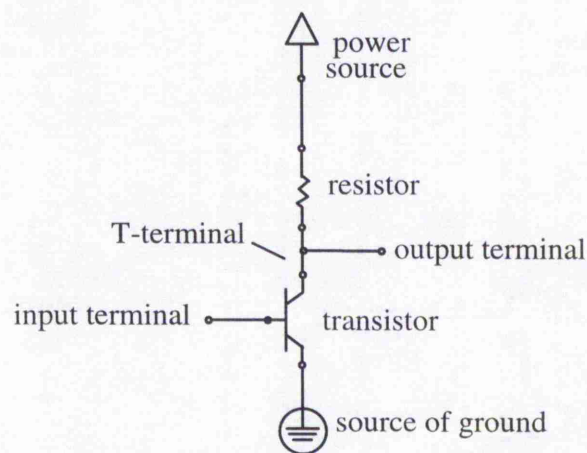


Figure 4.10 A circuit as connected components.

As it has been described in Chapter 2, a software program capable of extracting and comparing data structure from a schematic design has been developed to investigate the connectivity, and its effect on netlist generation. It establishes which components are connected to each other. An extracted report indicates the assignment of connections is then produced.

#### 4.5 DEFERENT TECHNIQUES USED FOR NETLIST GENERATION

The design data has been the focal point of the most design environments and considered to be a major area of interest in the construction of more general design work frames and tools development. However, the need for full design data tools is necessary and does play an important role in the operation of obtaining the final data formats. In particular, the following presented methods for netlist generation, requires sufficient translation capabilities to create the necessary input files for processing.

This section deals with the representation of different methods used in the investigation of netlist generation. Two levels of data representation are considered:

- The first is the data extracted using database or other extraction programs. It is at this level that all necessary data are realised.
- The second level is the relations between design data (i.e. reference design file) and tools that assist in performing further programming actions. It is at this level that the extracted data to tool interactions are maintained. These levels are considered and applied to each of the methods described below.

##### 4.5.1 Matrix Comparison Technique

This section concerns itself with some of the most common matrix operations and topological applications. A matrix is defined as a rectangular of array of elements, consisting of  $M$  rows and  $N$  columns. A matrix by itself has no numerical value nor can it be evaluated since its main purpose is to store an ordered collection of data. The use of matrix comparison outlines the following performance characteristics [4.14]:

Does it required memory (RAM) allocation ?, and does it required a mount of time to execute certain operations ?. Can it handle data elements of varying lengths?. Are the programs complex or simple ?.

In addition, three considerations are critical to success of this method. These are as follows:

- The extraction of all cell components co-ordinates.

- The extraction of all wires co-ordinates.
- The extraction of all included objects.
- Matrix solving procedure.

Once the extraction of the necessary data is completed, matrix solving procedure is applied and data are represented in terms of matrix forms. The following are descriptions to these matrices:

- Components matrix #1: is expressed as  $[4][Cn]$ , where  $[4]$  is the number of co-ordinates involving components with two-pins (e.g. resistors, capacitance, battery, etc.) and  $[Cn]$ , is the number of components in a circuit. Figure 4.11 shows four co-ordinates matrix of two-pins component.

		Components number						
		c1	c2	c3	c4	c5	c6	cn..
number of co-ordinates	1st	X1c1	X1c2	X1c3	X1c4	X1c5	X1c6	X1cn
		Y1c1	Y1c2	Y1c3	Y1c4	Y1c5	Y1c6	Y1cn
	2nd	X2c1	X2c2	X2c3	X2c4	X2c5	X2c6	X2cn
		Y2c1	Y2c2	Y2c3	Y2c4	Y2c5	Y2c6	Y2cn

Figure 4.11 Components matrix form of four co-ordinates.

- Components matrix #2: is expressed as  $[6][Cn]$ , where  $[6]$  is the number of co-ordinates involving components with three-pins (e.g. transistors, amplifiers, etc.) and  $[Cn]$ , is the number of components in a circuit. Figure 4.12 shows 6 co-ordinates matrix of three-pins component.

X1c1	X1c2	X1c3	X1c4	X1c5	X1c6	X1cn
Y1c1	Y1c2	Y1c3	Y1c4	Y1c5	Y1c6	Y1cn
X2c1	X2c2	X2c3	X2c4	X2c5	X2c6	X2cn
Y2c1	Y2c2	Y2c3	Y2c4	Y2c5	Y2c6	Y2cn
X3c1	X3c2	X4c3	X5c4	X6c5	X7c6	X3cn
Y3c1	Y3c2	Y4c3	Y4c4	Y5c5	Y7c6	Y3cn

Figure 4.12 Components matrix form of six co-ordinates.

- Components matrix 3: is expressed as  $[8][Cn]$ , where  $[8]$  is the number of co-ordinates involving components with four-pins (e.g. transformers, etc.) and  $[Cn]$ , is

the number of components in a circuit. Figure 4.13 shows eight co-ordinates matrix of four-pins component.

X1c1	X1c2	X1c3	X1c4	X1c5	X1c6	X1cn
Y1c1	Y1c2	Y1c3	Y1c4	Y1c5	Y1c6	Y1cn
X2c1	X2c2	X2c3	X2c4	X2c5	X2c6	X2cn
Y2c1	Y2c2	Y2c3	Y2c4	Y2c5	Y2c6	Y2cn
X3c1	X3c2	X4c3	X5c4	X6c5	X7c6	X3cn
Y3c1	Y3c2	Y4c3	Y4c4	Y5c5	Y7c6	Y3cn
X4c1	X4c2	X4c3	X4c4	X4c5	X4c6	X4cn
Y4c1	Y4c2	Y4c3	Y4c4	Y4c5	Y4c6	Y4cn

Figure 4.13 Components matrix form of eight co-ordinates.

- Wires matrix: is expressed as  $[4][Ln]$ . Where  $[4]$  is the number of wires co-ordinates and  $[Ln]$ , the number of wires in a circuit. Figure 4.14 shows wires matrix form.

		Wire number						
		L1	L2	L3	L4	L5	L6	Ln..
number of wires coordinates	start	X1L1	X1L2	X1L3	X1L4	X1L5	X1L6	X1Ln
		Y1L1	Y1L2	Y1L3	Y1L4	Y1L5	Y1L6	Y1Ln
	end	X2L1	X2L2	X2L3	X2L4	X2L5	X2L6	X2Ln
		Y2L1	Y2L2	Y2L3	Y2L4	Y2L5	Y2L6	Y2Ln

Figure 4.14 Wires matrix form.

There are some manipulations and operations that occur quite often when executing data structure operations. To achieve the connection between components, matrix comparison begins by finding the first co-ordinates of components and comparing it against each wire co-ordinate. This is done by the help of "if-then" rules (e.g.  $\text{if}((x1c1==x1L1) \&\& (y1c1==y1L1)) ==>$  means a connection) and so forth. The important point to be made here is that, the time taken will be proportional to the number of comparisons that have to be made. Therefore, from using this method, it can be seen that the number of comparisons will be one of the major quantities that affects the speed of execution. Figure 4.15 illustrates such a comparison between the different data structure.

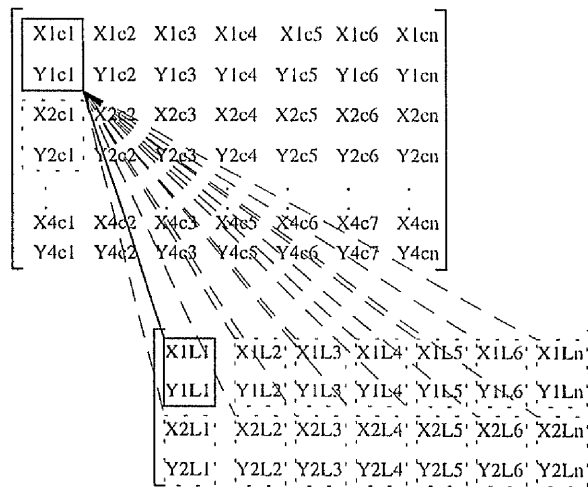


Figure 4.15 Matrices comparison (components versus wires).

When the entire data structure is completed, an array of node [Nn][Cn] numbers is obtained to show which components are connected and related to same electrical path. Figure 4.16 illustrates the node connection matrix form as a result. The information is then converted to a netlist form shows the circuit components and their connections.

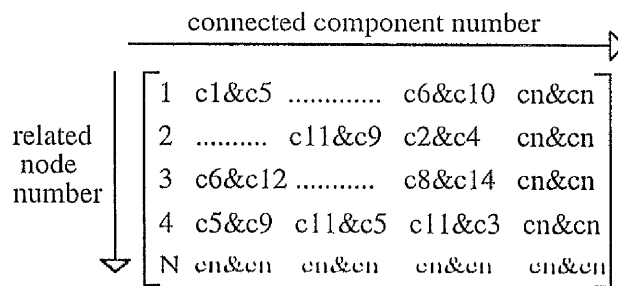


Figure 4.16 Resultant node connection matrix.

## 4.5.2 Circuit Data Extraction

In general most CAD/CAE applications operate independently, each having their own optimised data representation structure [4.15]. Data is captured relevant to the specific applications needs with little reference to other tools. It is quite common for the output files from one application to be required as part of the input to another. Design data can be schematic diagrams, integrated circuits layouts, netlists, and other text documentation. These data arise from various design steps.

Schematic diagram provides a signal flow graph of the circuit. Standard conventions in schematic design and other information imposed for readability incorporated into netlist generation are as follows:

- Library components represented by cells designed as standard or customised symbols.

- A route between two symbols in a series of horizontal and vertical lines realises the most direct connection paths for easy tractability.
- Components identification, includes information such as name, values and cells name for earth symbol recognition.

### 4.5.3 Software Program Description

Having identified the main issues that can be used as a guide in generating schematic netlist, the task of using this method is carried out in stages, as described below.

#### The first stage

It involves the extraction of the necessary data from circuit schematic and decomposes it into an output file (i.e. ASCII format). Figure 4.17 illustrates the generic interaction between data extraction program and circuit schematic. The extracted data show the sequence of input components values, references and wires start-end co-ordinates. Figure 4.18 shows the program flowchart used for circuit data extraction. For the convenient of output format, the program is based on using two main MDL scanning functions, namely:

- mdiCell\_extract function for components comparison and co-ordinates extraction.
- mdlLinear\_extract function for wires co-ordinates extraction.

Figure 4.19 shows circuit data file before proceeding the second stage, while Table [4.2] illustrates a summary of the program utilities, functions and their operations.

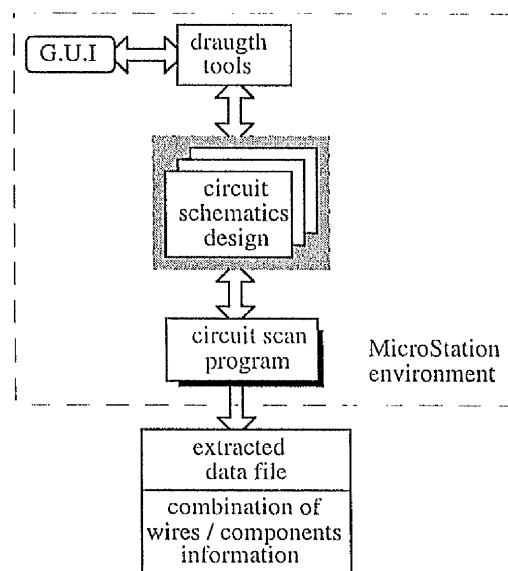


Figure 4.17 General procedure for data extraction.



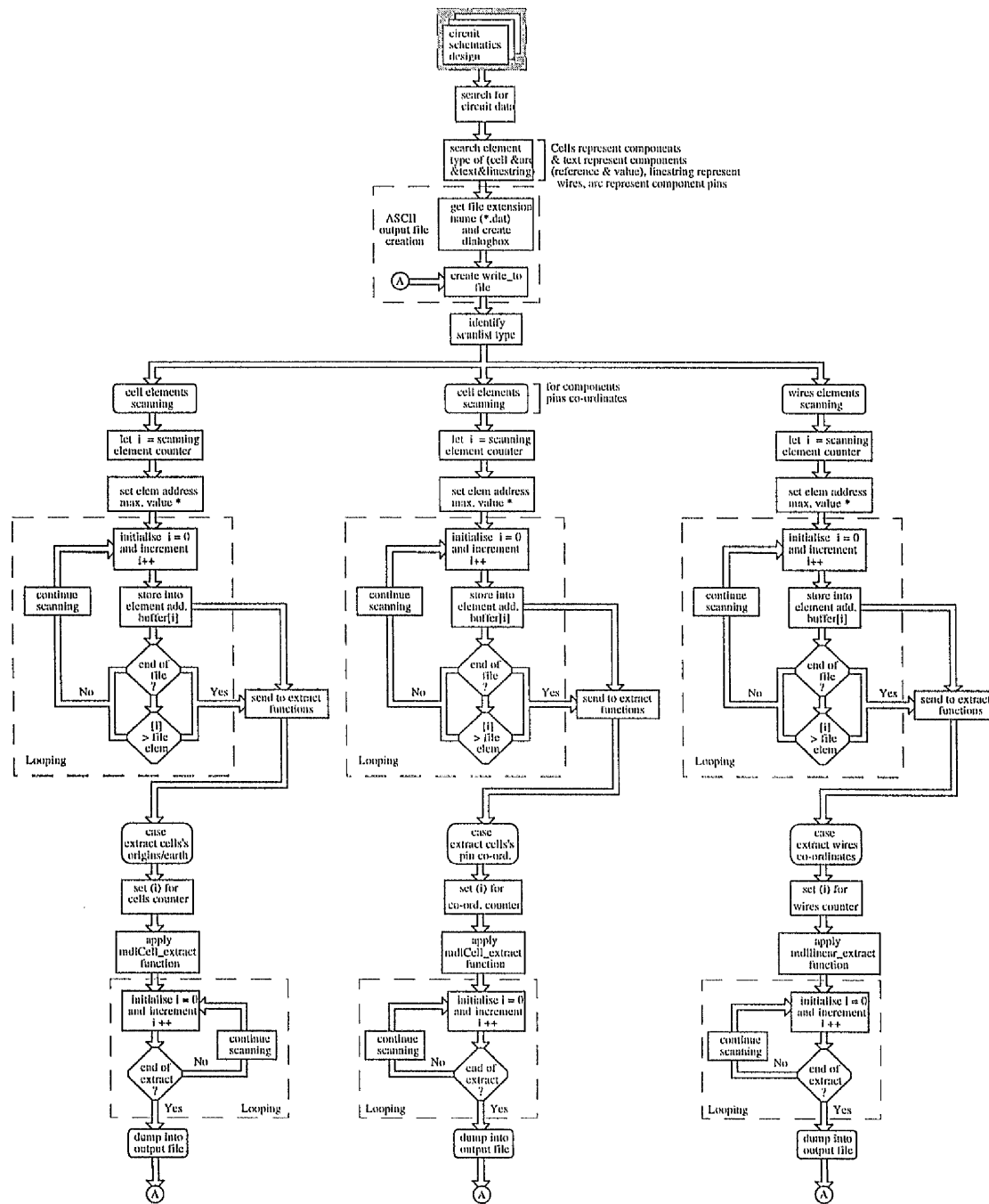


Figure 4.18 Flowchart of data extraction program.

## Chapter 4: Automatic Netlist Generation

Table [4.2]. Summary of utilities, functions and their operations.

Utility	MDL_Function	Operation
Identify element type	searchType[] (TEXT_ELM) (CELL_HEADER_ELM) (LINE_STRING_ELM) (ELLIPSE_ELM)	search for components reference, value and pins co-ordinates
Create dump data file	mdlFile_getcwd	retrieve name of current working directory.
State the derivation of file name to be parsed.	mdlFile_parsename	drive the components of file name
Create built-in dialog box for files list	mdlDialog_fileCreate	create file filter (*.dat) and dialog box title
get file name based on the user given name	mdlFile_find	find a file based on an incomplete file name specification
Open the file to write_to	fopen (filename, "w")	direct write to file
Indicate an error if a file can not be opened	mdlDialog_openAlert	open an alert dialog box
Add sheet top information	fprintf ("dump file", "")	write into output file
Display of complete extraction	mdlOutput_message	display an output message once extraction is completed
Initialize a scanning list	mdlScan_initScanlist	set scanning procedure from the beginning of the file
Load up scanList into the microStation design file scanner	mdlScan_initialize	set the scanning list so it will return displayable elements
Locate element type on the design file	scanList.typemask[1] (TMSK1_TEXT) (TMSK0_CELL_HEADER) (TMSK0_ELLIPSE) (TMSK0_LINE_STRING)	establish a type mask for element location
Extract array of coordinates from line, linestring	mdlLinear_extract	establish wires coordinates and number off
Extract arc or ellipse element	mdlArc_extract	establish component's pins coordinates
Extract character string from text element	mdlText_extractString	establish text extraction and string comparison (e.g. reference and value)

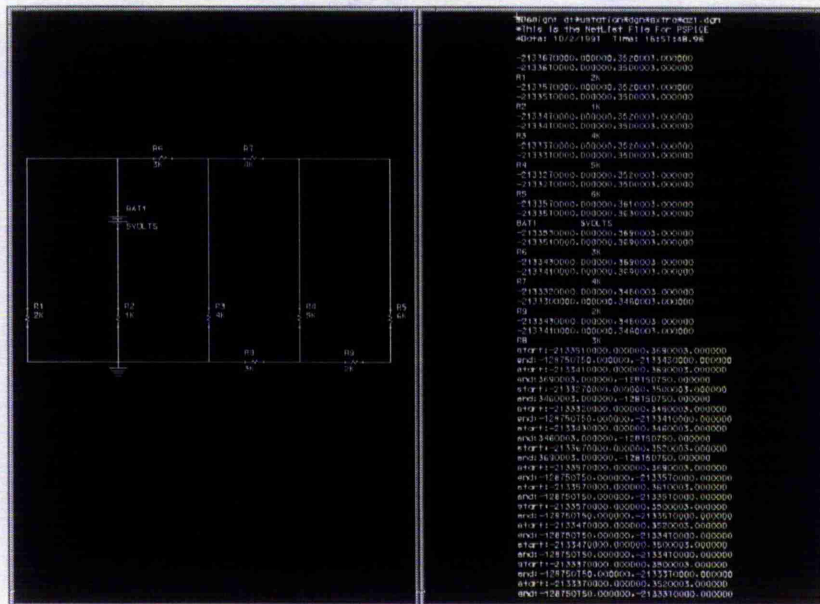


Figure 4.19 A circuit and its extracted data.

### The second stage

It involves processing the extracted data and comprises the creation of temporary files and with a matrix data structure. Since the extracted data file shows a combination of both

circuits elements (i.e. wires, components), the creation of temporary files and data matrix data structure is performed to ease data handling during nodes determination.

Using the extracted data file as an input, data lists are sorted in a way that can easily be handled by first: creating a matrix data structure under the name *coord[ ]* and *lines[ ]* and second copying these data into temporary files under the name *w.lis* and *c.lis* for wires and components respectively. Figure 4.20 shows the two types of metrics before processing into nodes determination stage.

SL1	X1L1,Y1L1	C1ref	C1val	X1C1, Y1C1	N1
EL1	X1L1,Y1L1	C2ref	C2val	X2C1, Y2C1	N2
SL2	X2L2,Y2L2	C3ref	C3val	XnC1, YnC1	Nn
EL2	X2L2,Y2L2	C4ref	C4val		
SL3	X3L3,Y3L3	C5ref	C5val		
EL3	X3L3,Y3L3	Earth		X1Cn, Y1Cn	Nn
SLn	XnLn,YnLn	C6ref	C6val	X2Cn, Y2Cn	Nn
ELn	XnLn,YnLn	Cnref	Cnval	XnCn, YnCn	Nn

Figure 4.20 Wires, components, and components co-ordinates/nodes matrices form.

The matrix *linesc[ ]* is a linked structure matrix. It consists of three other matrices, namely: *comps[ ]* matrix is the component data main matrix, *compname[ ]* matrix is where components name and values are stored and *coordc[ ]* matrix is used to store components co-ordinates data. The linked list can be manipulated to meet the nodes determination target. Therefore, to be able to access or copy data into each of these matrices, data pointers are required.

A linked list is defined as a set of data elements which are linked together. In this case, each list element consists of two parts: a data element which holds the data contained in that element, and the pointer which gives the address of the next element in the list. Figure 4.21 shows, the one way linked structure.

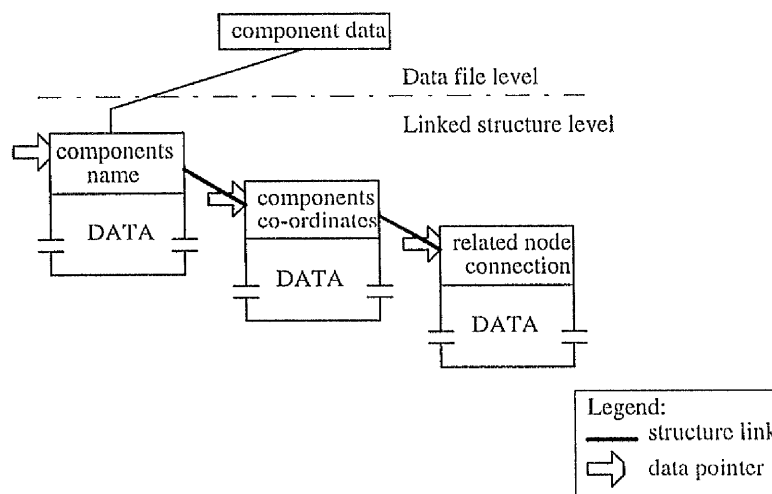


Figure 4.21 Illustration of the basic structure of one way linked list.

In such a list the program only needs the location of either the first or the last member in the list depending on whether the pointer point forwards or backwards. From this it can look at the pointers in each element to find the next element in the list. Figure 4.22 shows the program flowchart where data are stored into different structure of matrices.

Using the created matrices and their pointers, data are copied to perform the third stage, which is the nod determination.

### The third stage

Node determination is considered to the most complex stage in deriving the final netlist format. The procedure can be summarised as follows:

- Read the ending co-ordinate of the first component from `coordc[ ]` matrix.
- From the wires co-ordinates matrix `wireco[ ]` determine the wire element by comparing the starting co-ordinate with ending co-ordinate of the component. If the connection is established, then the wire is identified, else keep scanning until the matching is achieved.
- Scan through the wires matrix `wireco[ ]` to next element and compare to determine the next connected component to that wire. If this is true, then a node is determined and stored in the node buffer for later scanning. The determined node in this case is shared between the two components.
- A problems can occur in the case. Spatial or T-terminal connections are considered as a circuit complex connections, this means that same electrical node is shared between the components involved, as shown previously in Figure 4.9 in Section 4.3.2. To solve this problem, another matrix `arypnt[ ]` to hold the connected wire is needed.

The electrical node shared between the components is then updated by the same number.

- Scan through the compname [ ]. If the earth name is found, then the corresponding node co-ordinate will be zero. Figure 4.23 shows the program flowchart for node determination stage.

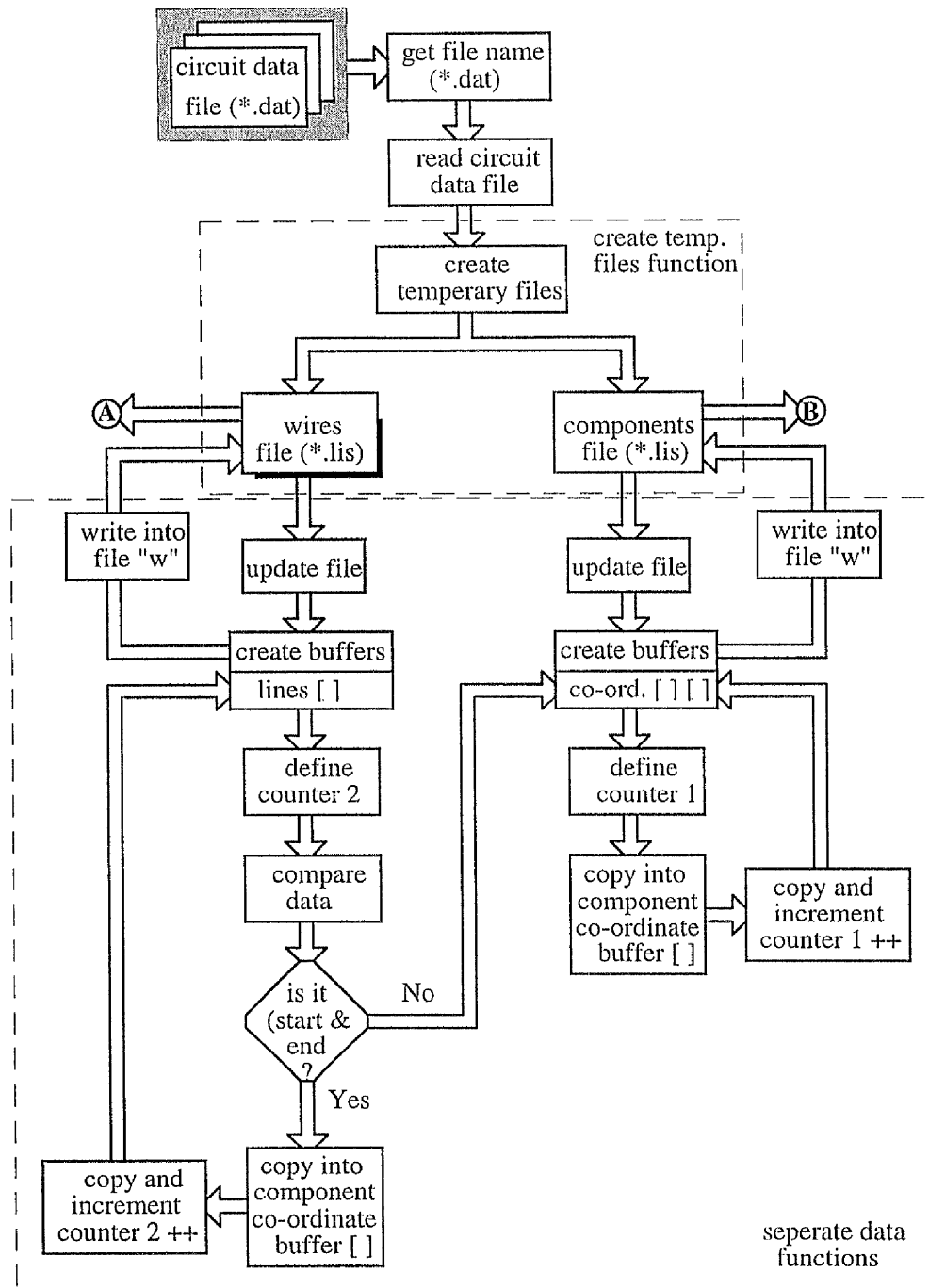


Figure 4.22 Flowchart of read data file program.

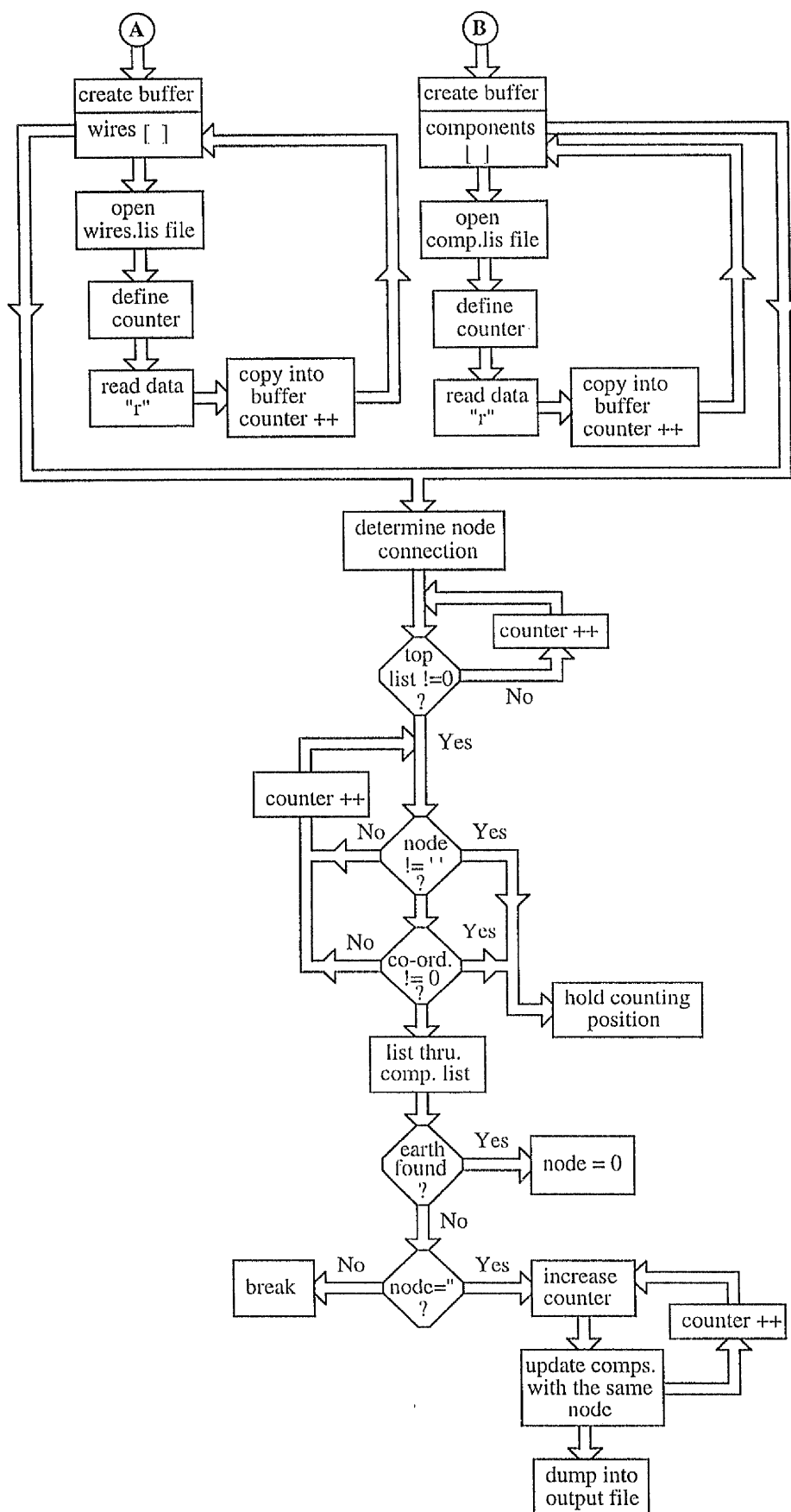


Figure 4.23 Flowchart of node determination program.



The above procedure is repeated until the whole data structure is completed and the final netlist format is obtained. Figure 4.24 shows the data processing general procedure for netlist development and Figure 4.25 demonstrates the corresponding screen display using MDL debugger.

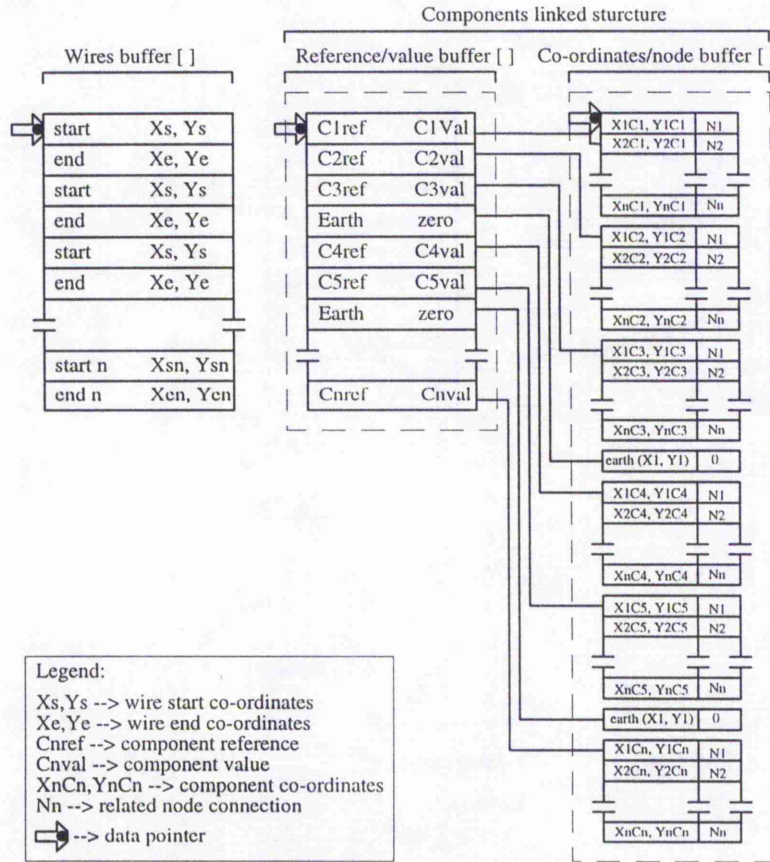


Figure 4.24 Procedure of data processing.

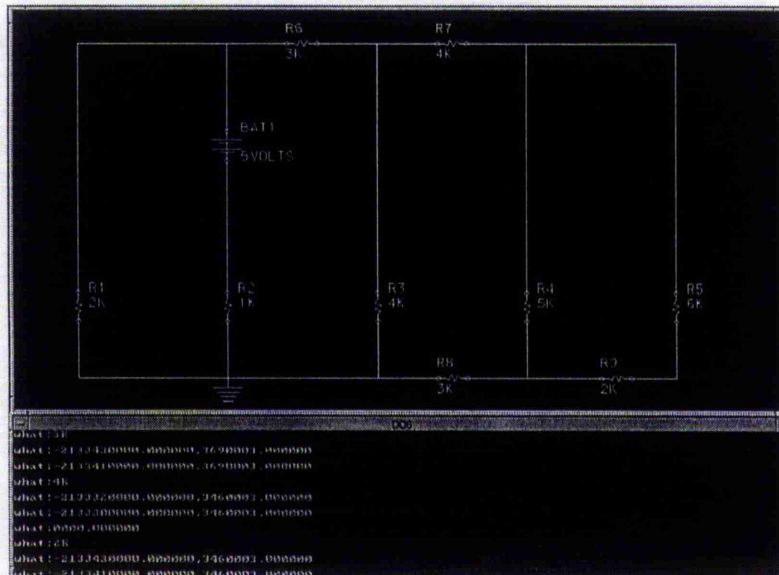


Figure 4.25 MDL debugger shows data during separation.

#### 4.5.4 The Use of Database

CAD database refers to not only the central database facility but also to all files and information produced through the invocation of CAD tools. The data involved may be generated in various ways including recording on interactively made designer decisions or through computation procedures. The objective of using this method is to investigate the use of database dbase(iv) in generating an automated netlist format file suitable for simulation with PSPICE.

Chapter 2 has described that MDL is a programming language supporting different database functions, allowing elements data to be scanned and automatically edited into database tables, as illustrated in Figure 4.26. Using the database in this case requires the following:

- Create database tables consisting of data fields considering circuit different connection possibilities shown in Figure 4.9 Section 4.3.2.
- Interface establishment between 2D schematic environment and dbase(iv) (refer to Chapter 5 )
- Programs to store and extract data .

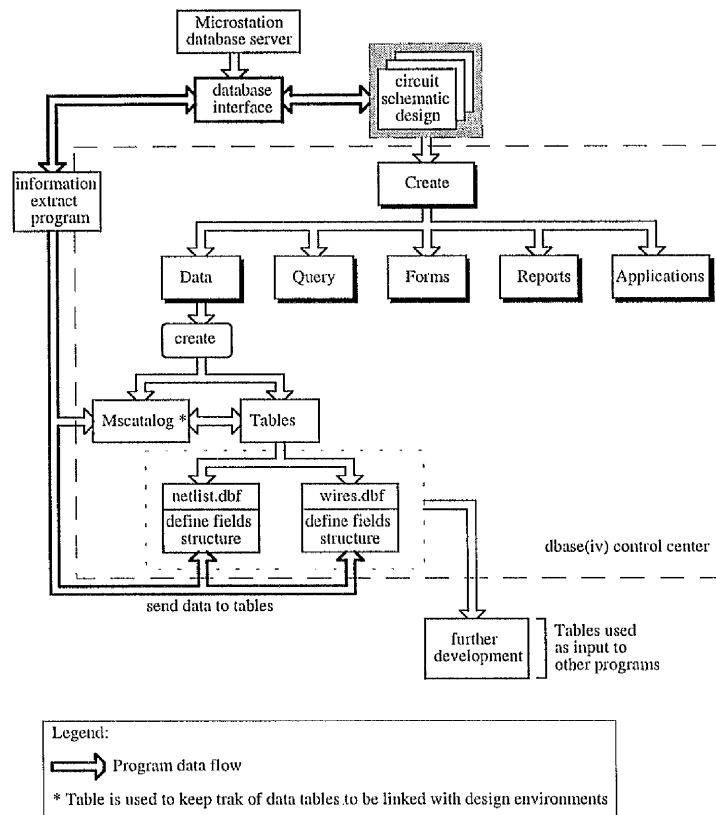


Figure 4.26 The use of database dbase(iv).



### 4.5.5 User Defined Netlist

Figure 4.27 illustrates the interface between the two environments and the functionality needed in order to process the specified data.

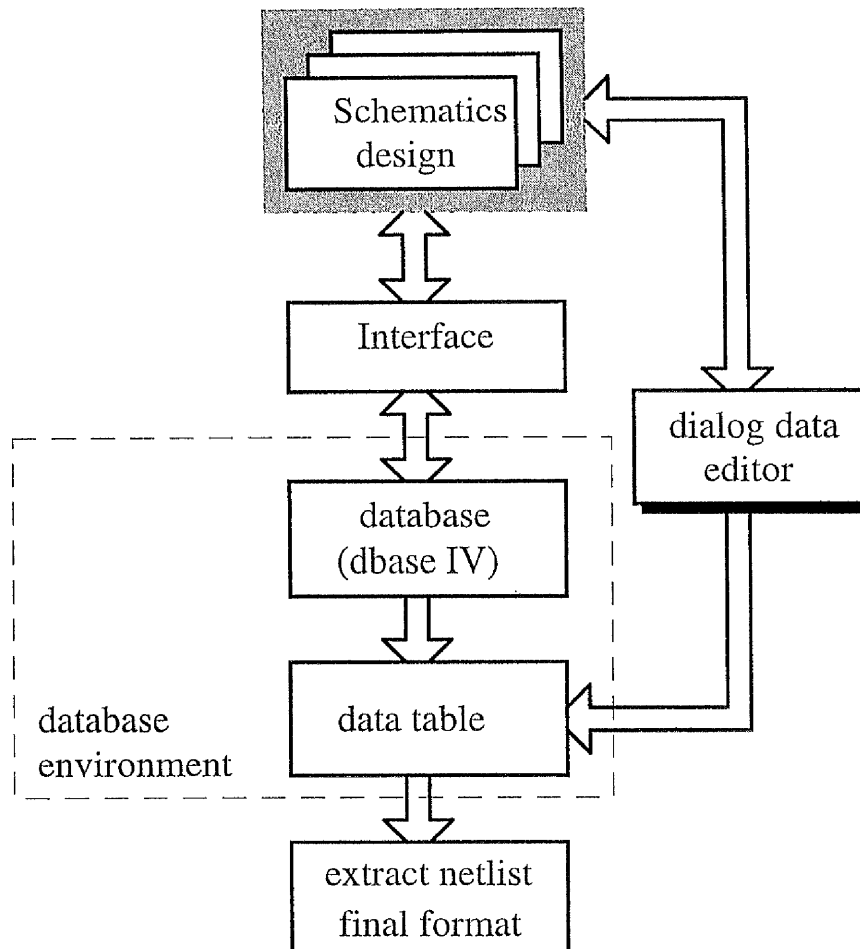


Figure 4.27 User defined netlist using dialog data editor.

The objective of using this method is to allow the netlist to be defined using dialog editor. The whole method consists of two main programs, namely: Specify data and extract final format. A description to these programs as follows:

#### The Software program

The first task performed is to select the component and information including (reference, nodes and values) are then specified and stored in database table. The program also allows these data to be linked to the selected component via (mslink). Mslink field must be added to database file, so that the link can be established between the database and graphics elements. Figure 4.28 illustrates the program flowchart for dialog editor.

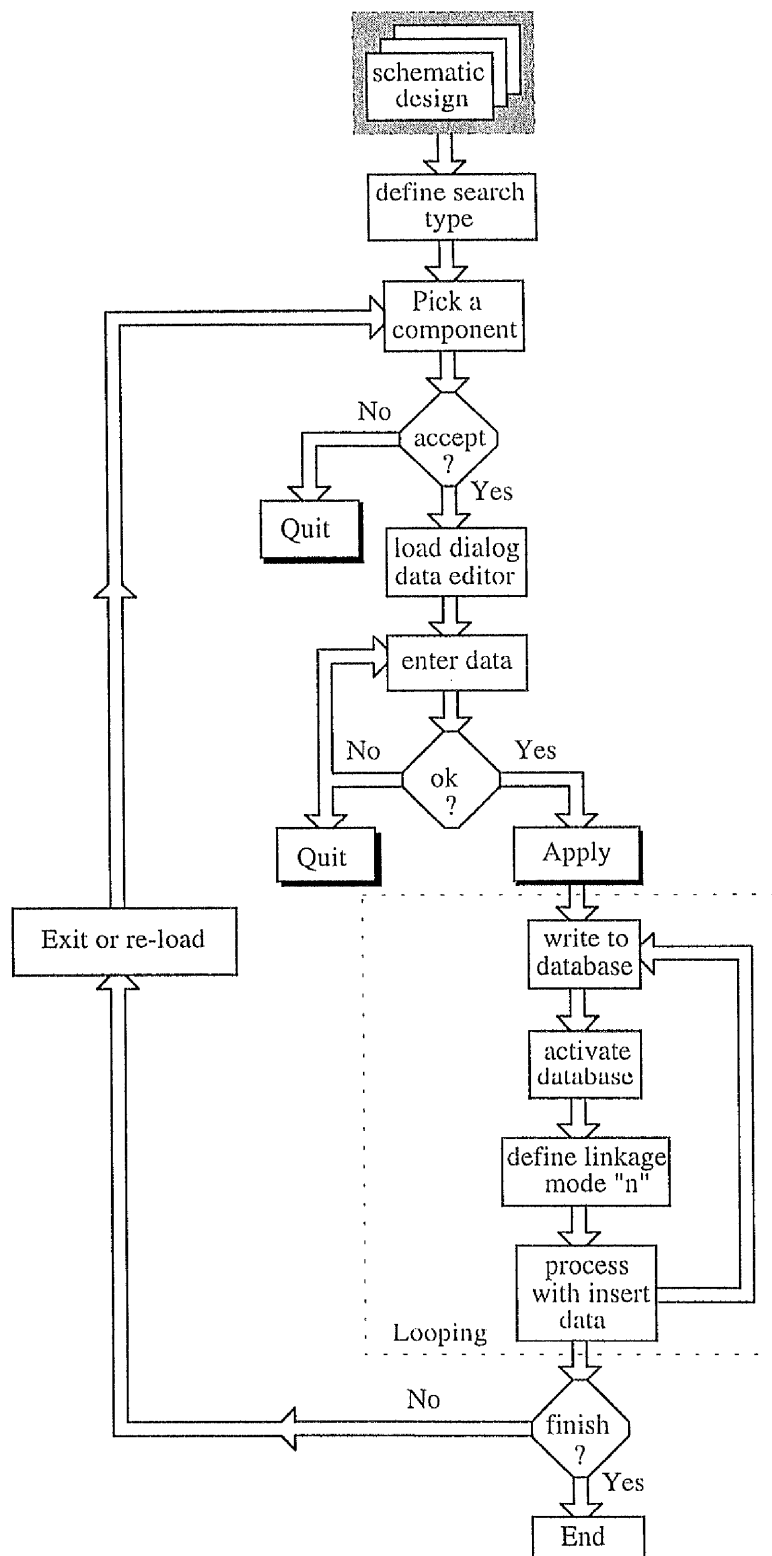


Figure 4.28 Flowchart of dialog editor program

The final netlist format is produced using the second program which deals with extracting the information from database. Figure 4.29 illustrates the program flowchart, while Table [4.2] shows the MDL functions used for extracting netlist final format.

Figure 4.31 shows netlist file in its final format edited using the graphical dialog editor. Figure 4.31 shows the corresponding circuit netlist obtained as result of this program.

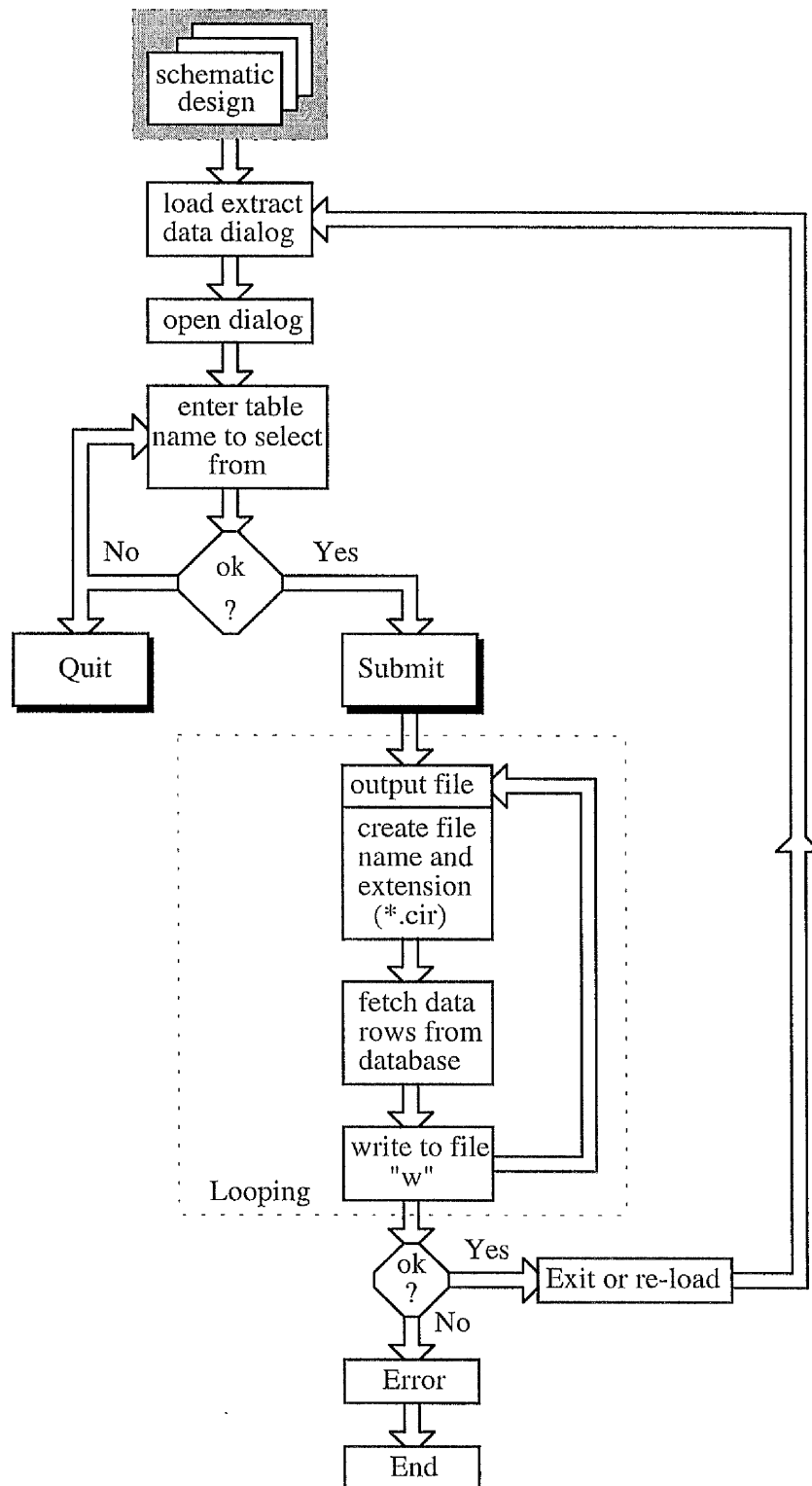


Figure 4.29 Flowchart of netlist final format extraction.

## Chapter 4: Automatic Netlist Generation

Table [4.2]. Summary of database extraction utilities, functions and their operations.

Utility	MDL_Function	Operation
Load database table	mdlDB_activeDatabase	activate database table
Identify element type	searchType[] (TEXT_ELM) (CELL_HEADER_ELM) (LINE_STRING_ELM)	search for components reference , value and pins co-ordinates, wires co-ordinates
Extract cell header information	mdlCell_extract	establish components co-ordinates extraction
Define data to insert to components table	sprintf(sqlStatement, "insert into netlist (fields)values (%?" , ..., arguments)	direct data for adding
Link a graphic element with the active entity using SQL statement	mdlDB_defineAEBBySQLInsert (sqlStatement)	add new rows of data
Define mode choice	mdlDB_activeLinkageMode ("new")	(mode=new) for adding new rows
Extract array of coordinates from line, linestring	mdlLinear_extract	establish wires co-ordinates extraction
Define data to insert to wires table	sprintf(sqlSt, "insert into wires (fields)values (%?" , ...), arguments)	direct data for adding
Link a graphic element with the active entity using SQL statement	mdlDB_defineAEBBySQLInsert(sqlSt)	add new rows of data
Extract character string from text element	mdlText_extractString	establish text extraction and string comparision (e.g. reference and value)
define data to insert to components data table	sprintf(sqlSt, "update netlist set value =%?" where mslink =%u" , arguments)	direct data for adding
define data to update fromcomponents table	sprintf(sql, select mslink from netlist where name=%s" , arguments)	direct data for updating
update choosen data row	sprintf(sqlStatement, update netlist set values where mslink =%u and name=%?" arguments)	
process previous defined statement	mdlDB_processSQL(sqlStatement)	process previous defined data

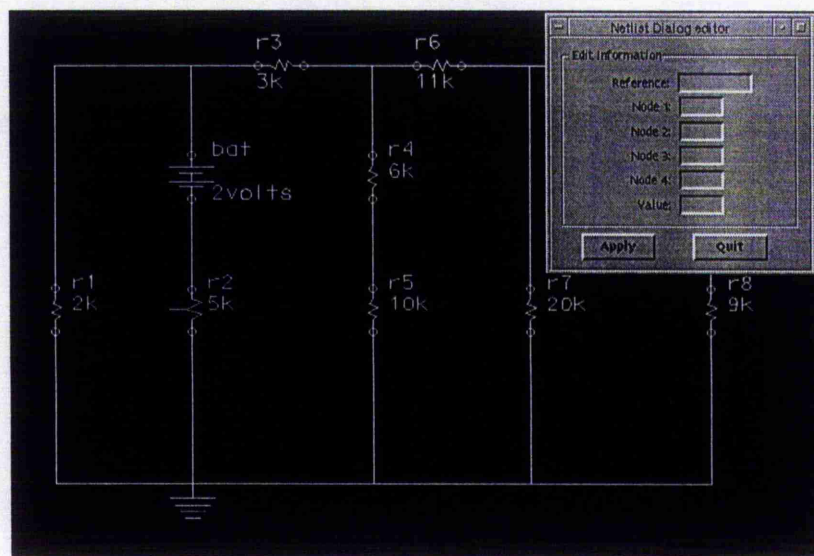


Figure 4.30 Example of generating netlist using dialog editor.

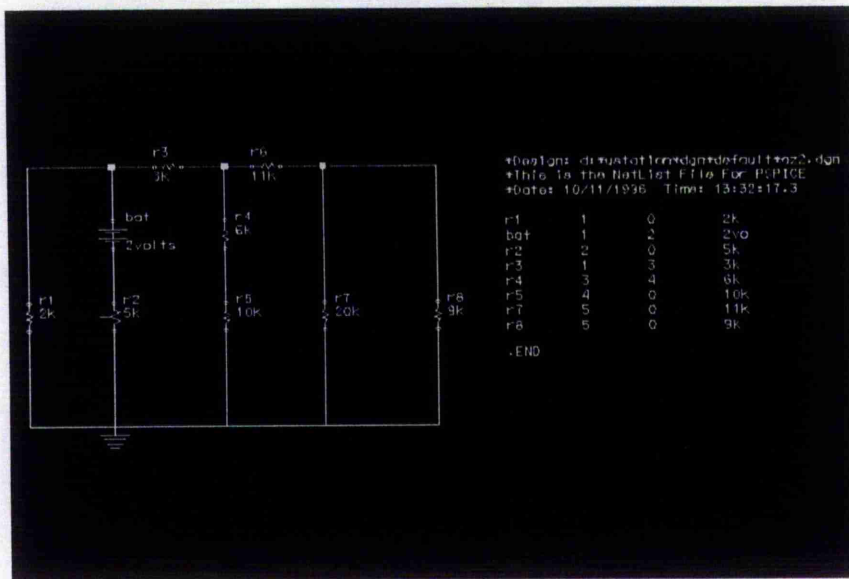


Figure 4.31 Generated netlist file.

## Data Extraction Case Study

An example of electrical circuit design is selected to illustrate the performance of data extraction tools. Based on the programs flowcharts previously described in Chapter 2, design data such as B.O.M, wiring list, drawing information are extracted and reviewed. Figures (4.32-4.34) show the performance of these tools which are related to the generation of netlist files, where connectivity is an important element. Figure 4.36 shows the automatic placement of junction numbering and wires identities.

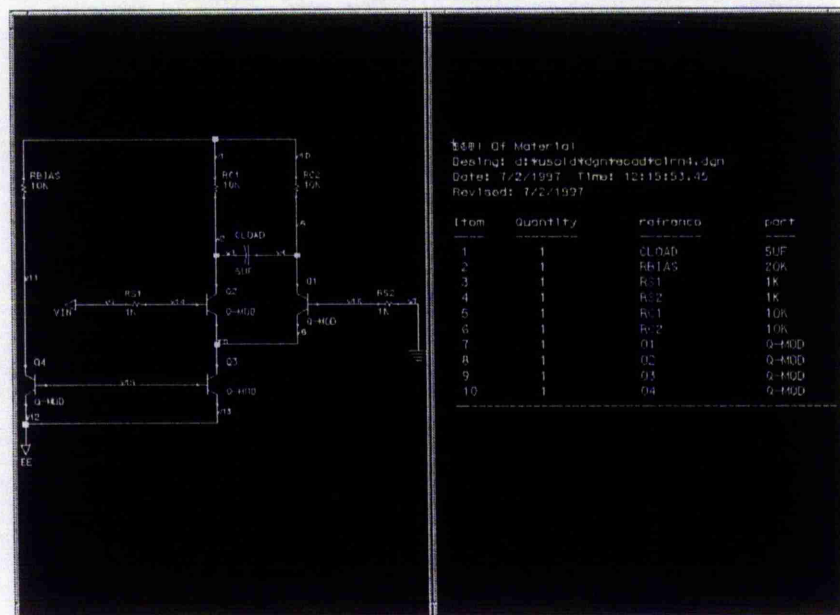


Figure 4.32 Circuit Bill Of Material extraction.



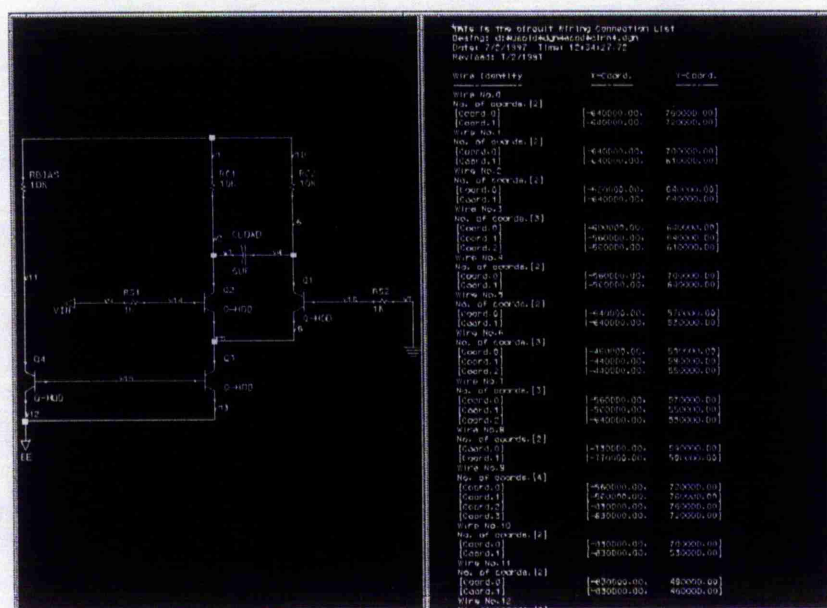


Figure 4.33 Circuit wiring list extraction.



Figure 4.34 Circuit drawing information extraction.

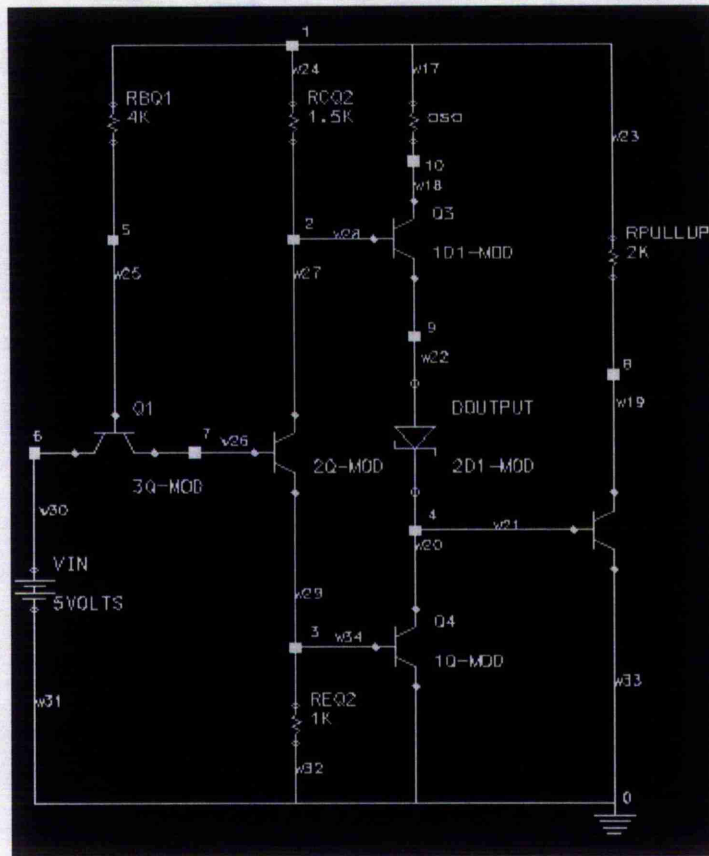


Figure 4.35 Automatic nodes-wires identity placement.

#### 4.6 CONCLUSIONS

This chapter has described the importance of netlist generation in circuit design process and analysis. An important characteristic, specific to generating a circuit netlist is connectivity establishment which is related to the notion of components-wires connection.

The objective of this chapter was to investigate this problem, where different techniques have been investigated. Based on preliminary study of rules and conditions, a key feature to this investigation was the data extraction that represents circuit schematic construction.

The software programs intended for investigating netlist generation have been divided into different stages. The first program which referred to as extract-process involved the use of matrix solving method. A generic data file produced and further processed through the second stage of the same program. During the process of this data, the time taken to process the whole program was one of the main concerns. Although the process taking to

extract schematic data was fast, the process of analysing this data was found to be slow due to the following reasons:

- The program uses a number of matrices that need to be processed.
- Time taken to separate data from the main data file into different files for easy tractability.
- Data processing by data comparison and the creation of node matrix.
- The scanning of node matrix for netlist final format determination.

## **REFERENCES**

- [4.1] Larry G. J., "Fast batch and incremental netlist compilation of hierarchical schematics," IEEE Tran. on Computer-Aided Design, Vol. 10, No. 7, pp. 922-931, July 1991.
- [4.2] Evangelos S., "A knowledge-based system for the evaluation and redesign of digital circuit networks," IEEE Tran. on Computer-Aided Design, SIMOUDIS: Digital Circuit Networks, Vol. 8, No. 3, pp 32-315, March 1989.
- [4.3] Larry G. J., "Fast online/offline compilation of hierarchical schematics," ACM/IEEE Design Automation Conference, ACM, 1989.
- [4.4] OrCAD electronic design automation tools. "Schematic tools reference guide," OrCAD L.P., document number OR9063C-8-23-91, 1991.
- [4.5] Analogy, "Introduction to the saber simulator," Analogy user manual, Inc., registered in England no. 2372734, 1991-1993.
- [4.6] Majewski M. A., at el. "Autodraft: Automatic synthesis of circuit schematics," Proc. Int'l Conf. Computer-Aided Design, IEEE Computer Society Press, Mamitos, Calif., pp. 435-438, 1986.
- [4.7] Naveen B., "An automatic netlist-to-schematic generator," IEEE Design & Test of Computers, Vol. 10, No. 1, pp. 36-40, March 1993.



- [4.8] Nagel L. W., "SPICE2: A computer program to simulate semiconductor circuits," PhD. Dissertation thesis, Uni. Calif., Berkeley, Electronic Lab., May 1975.
- [4.9] MicorSim corporation, "Circuit analysis user's guide," 20 Fairbanks, Irvine, Calif. 92718, Version 5.0, July 1991.
- [4.10] Paul F. C., et al. "Direct circuit simulation algorithms for parallel processing," IEEE Trans. on Computer-Aided Design, Vol. 10, No. 6, pp. 714-725, June 1991.
- [4.11] Walter B., "Computer aided circuit analysis using SPICE," Prentice-Hall International Editions, ISBN 0-13-168394-2, 1989.
- [4.12] Robert O., and Steve M., "Parallelizing SPICE in a timesharing environment," Applications of superscomputers, Elsevier science publishing Co., Inc., pp.185-195, 1986.
- [4.13] Steven M. R., "Computer aids for VLSI design," Schlumberger Palo Alto Research, ISBN 0-201-05824-3, 1987.
- [4.14] Daniel F., and Neil W., "Data structure with abstract data types and Pascal," Calif. Polytechnic State Uni., San Luis Obispo, ISBN 0-534-03819-0, 1984.
- [4.15] Hamid R. P., and William G. S., "Concurrent engineering," Contemporary issues and modern design tools, First edition, ISBN 0-412-46510-8, 1993.

# 5

## Design for Database Auto-link

### 5.1 INTRODUCTION

Database applications have had a major impact on the growing use of computers playing a critical role in almost all areas where computers are used, including engineering. Database engineering is fundamental to CAD systems in terms of functionality of the engine and system outputs. Because much of this fundamental engineering data is created by the design functions, the heart of the CAD system has to be the engineering database.

Integration consolidation of data using a relational database allows controlled distribution and rapid access in information. For example, design geometry, text, analysis, test data and production costing data for a part can be consolidated on to an engineering database. This provides the ability to manage and provide queries into all information on released products. The key components of engineering database taken from Don [5.1] are summarised as follows:

- The storage of any kind of data this includes (e.g. geometry from multiple CAD systems is held in special format, parts list in a common format, software programs, design specification, etc. All this is be held in a highly secure manner, thus providing the highest layer of data control for released information.
- The ability to store extensive descriptive data in a form which is suitable for engineering and to be able to relate this data to previous stored data.
- A controlled communication mechanism for distributing and gathering data across a network or between different CAD systems by means of data exchange format (DXF). For example, the transmission of drawing between design office and other location such as manufacturing plant.

- A user interface which smoothes the communications, the finding of data and the invocation of programs.
- The system must be capable of considerable expansion, in order to avoid constraints on the business. For examples small databases origination are not capable of handling a large volume of drawings and information generated by a growing product line.

Engineering systems such as drafting, bill of materials, analysis and test etc. have grown very quickly as independent entities. They generate data very rapidly, and shared in a limited fashion. The effective use of data have more influence upon an origination's profit levels than the system which simply create data, for example CAD/CAM integration. Without an overall approach to engineering, locally optimised computing facilities can only become less efficient.

An integrated design environment can be used to give the impression that all relevant design data is contained within one large database. Therefore it can be used to help to understand how different aspects of the design fit together to create composite design, thus providing a basis for the expression of design constraints and the development of more complex applications. Current developments towards database technology provides the key to building intelligent environments for managing design data. Some of these developments are reported as part of literature. Figure 5.1 illustrates database engineering is the scope of integration.

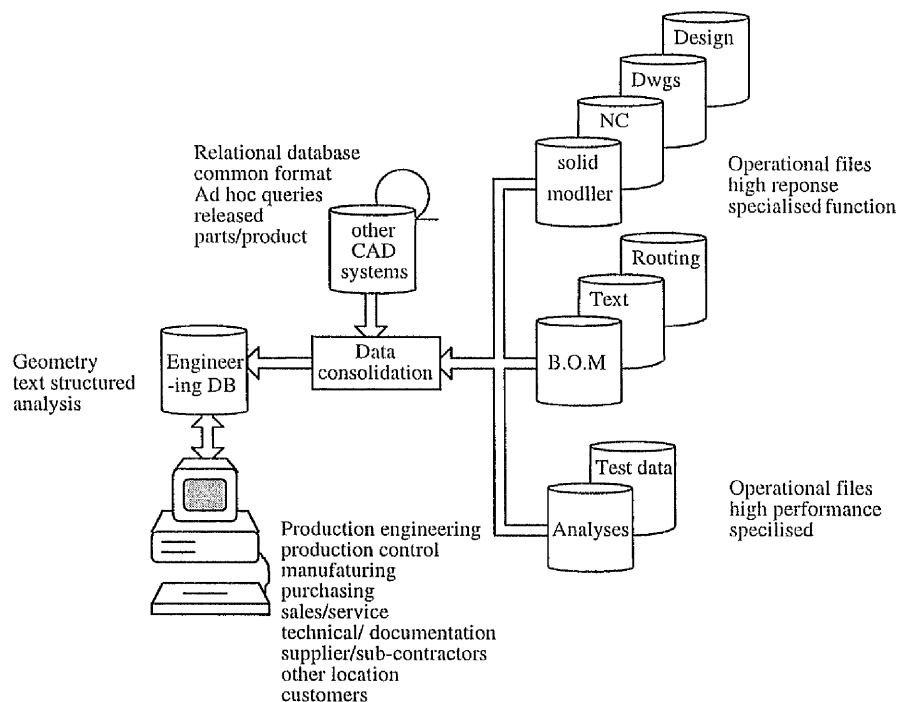


Figure 5.1 Engineering database (towards integration -engineering database) [5.1].

As the complexity of electrical circuits increases, information management is becoming crucial to the success of CAD systems [5.2]. In most cases, the CAD systems that exists today each use their own data format. For example, the format represents netlist file extracted by the OrCAD system is different to SPICE netlist format. This requires translation tools for compatibility reasons. Organising design information under the control of data management system facilitates software maintenance.

This chapter concerns the application of database theories formulated mostly in the integration area of CAD/CAE design and specifically for electro-mechanical design. The major approaches to database design are analysed and software applications are designed and developed to assist in the integration between the different environments (i.e. 2D schematic, 3D modelling, and PSPICE) and dbase(iv). In addition, illustrated examples are also discussed.

## **5.2 A HISTORY AND EVALUATION OF DATABASE SYSTEMS**

The origins of database systems started from storage structures and access mechanisms developed to solve the problems which occurs during storage of a piece information [5.3]. Through the history of information storage in computers, one of the most observable trends has been the focus on data independence [5.4]. The relational models was firstly proposed by Codd [5.5], it observed that conventional database systems store information in two different ways: First, by the contents of records stored in the database and second, by the way in which these records are connected together. Different database systems uses various methods for the connection among records.

## **5.3 THE ROLE OF DATABASE IN CAD**

Databases are an ideal medium to integrate a variety of CAD packages around a common data set, thereby speeding up communication among different design packages and avoiding tedious design errors that might occur such as data conversion. With the recent interests in VLSI. CAD is emerging as one of the more interesting areas of application for database technology.

As previously mentioned, the requirements of a database management capability for engineering design are fundamentally different from the requirements that business data processing places on database management. The application of business database has generally required to support record keeping, and therefore requires modelling of simple relationships between data types. Table [5.1] provides a summary of the differences between business and engineering databases planning.

Table [5.1]. Database comparison [5.6].

	Business database	Engineering database
Use	Record keeping; data sharing; business planning.	Engineering design and analysis.
Characteristics	Few record types; simple relationships; large number of instances of each type; static representational requirements.	Large number of data types; large number of instances; complex relationships between data elements.
Transaction	Query processing; database maintenance-add, modify, delete; short time duration; answer involves few records.	Everything from simple queries to complex analysis codes run against database; queries can span long time duration and involve large number of records.

The use of database in CAD, involves two main areas of investigation:

- First, the design of vary large scale integration (VLSI) electronics circuits. They are of a particular interesting design because of the rich set of representations used in their description. These representation may include system architecture, logic diagrams and process specifications.
- Second, the design of mechanical structures and systems. A basic challenge in mechanical design is geometric modelling, which refers to the physical shape of mechanical parts. Different CAD systems employ different geometric techniques, including wire frame, surface and solid modelling.

Chapter 2 has described the necessity of database in supporting design CAD systems. It plays an important role in designs for electronics circuits as they are expressed in different design techniques. Thus its expected to find appreciated facilities, that provides a design with flexible applications and tools. In order for database to become more effective tools in electronics design, the following requirements are essential:

- Database systems must support a variety of design methodologies.
- Databases must be able to provide a convenient interface to a wide and changing variety of programs. This interface should not be changed as the database is developed new applications and additional tools are added.
- Performances of database systems has to be such that it must be able to retrieve, store, update and manipulate design data or computed data using storing and exacting applications programs.

## **5.4 INTERFACING WITH SOFTWARE ENGINEERING**

There is general agreement on the need for better methods of handling engineering data [5.6]. This suggests that a broader view of CAD systems integration problem may be required to develop the database support required for engineering computing.

The initial aim of software engineering is to develop ways that makes software development process easier, more effective, and more efficient [5.7]. With database becoming so a commonly used environment, the development of software applications is considered in the context of data base management (DBM). Therefore, the following areas are considered in interfacing between database and software engineering:

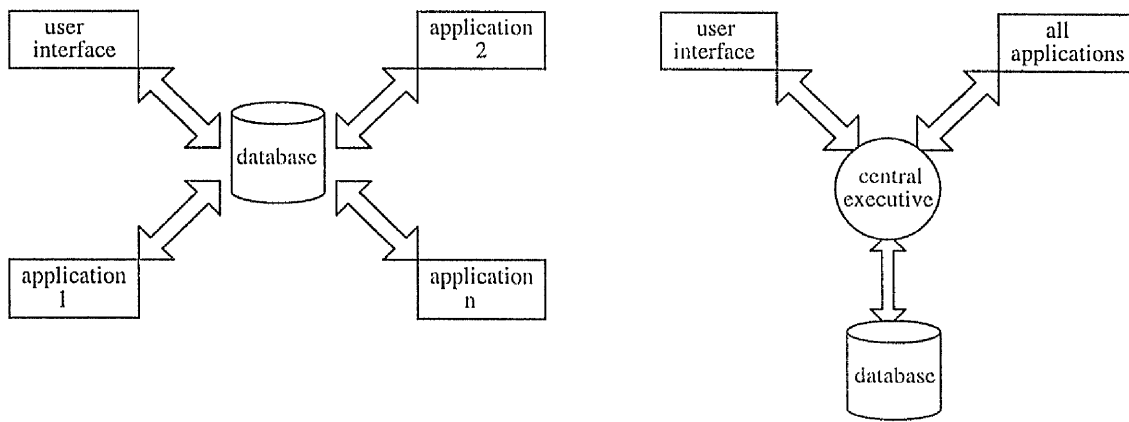
- Application design: This area is of great interest in commercial applications development. Many of commercial applications generators are on the market.
- Software tools: Developing new tools have two purposes: First, is to reduce the complex of tasks. Tools of this kind include requirements specification and analysis, integration, physical design and optimisation tools. Second, is to improve the performance of certain machines. Tools of this kind include performance provide a suitable interfaces that can be interfaced with tools such as monitoring, reorganising, and restructuring.

## **5.5 COMMERCIAL DATABASE SYSTEMS**

Database technology has dealt with both the management of large volumes of data and the problems caused by concurrency of data access. Many existing database systems supports internal area communication functions. They are dealing with inventories, where production and sales issues interface. Given this comment, database can provide tools and applications to serve the demand of integrated design. Some of these systems are described below.

The database system architectures that are most commonly used in CAD systems can be classified into two types according to Barley [5.8]:

- The type which considers database to be the central abstraction within the system. The applications packages that interact with the database have a common fixed data structure, this leads to the creation of an integrated systems as illustrated in Figure 5.2.
- The second type recognises that, in most cases, the database is not the single common factor, but the applications themselves may have much in common, as illustrated in Figure 5.3.



Figures (5.2, 5.3) Database-centred and Executive-centred systems [5.8].

Using commercial database systems outlines some advantages. The most obvious is avoiding the important effort required to custom design a complete, secure, and reliable database system [5.9]. They have built in-facilities for maintaining backups and generating data. They also incorporate means for enforcing access protection as well as handling the complex storage problems of growth and update that can be generated during design process.

Susan et al. [5.10], described the specific data technology needed to achieve product integration as a combination of database systems. Through the use of database systems, an integrated design can be developed to give the impression that the components of the design are stored as one large database. The advantages of using this approach is that a relationships between different components in a design can be established, therefore, supporting relationships between design and manufacturing and quires that present data from several different sources in an integrated way.

## 5.6 RELATIONAL DATABASE CONCEPTS

Relational database has been an effective tool for productivity with facilities such as query language and fourth generations language. There is a new requirements in using relational database system towards domain such as CAD/CAM integration, software engineering and design automation. In these areas, an information system must be designed to provide facilities for data modelling and manipulation. These data are referred to as entities or objects; such entities are usually have a hierarchical structure and can contains large number of attributes.

A major advantages of the relational database is the simplicity of its basic data structure and 2-D tables. This enhances dialog between design database and the use of data. However, given this background; the basis for a successful relational database is that it

can adopt to changing design needs and grow is to accurately modelling the data. Therefore, this intends to focus on the ways of data been structured rather than the procedures and different methods of accessing the data. Because of the fact that the same data can be viewed and used in many different ways, the most important part of database applications development is to model the data in a way that can be handled efficiently.

This section provides the most fundamental issues for implementing a relational database. It also lists a number of sources described in detail in a way that they can be useful in tackling the problem of applying database experience to serve in the VLSI design environment.

### **5.6.1 Data Relational Model**

The relational model has been gaining popularity in most areas of database applications. This is because of its flexible interfaces [5.11]. This model is well suited for representing CAD databases and its ease of use that can help to simplify the implementation. The most important tools in database design are underlying data model and its corresponding data definition and manipulation languages [5.12]. Generally, data modelling is defined as a collection of concepts that provides a guidance to specify the structure of a database and a set of associated operations for specifying retrievals and updates on the databases.

Database models have also held the attention of applications developers. There are several methods that have been developed under the name of logical database design. Generally, based upon interactive models designed to aid manual methods, as they often call for new CAD techniques. The interactions between designers and system are question-answering oriented. These interactions, however, are always guided by the system.

In this section, data conceptual model is planned. In its basic form, the model represents information in terms of entities, entities attributes, and their relationships. The entity-relation model has proven to be effective tools for communications between database and design environment because of its ease of understanding and its convenience in representation. The first step in developing the data model was by the consideration of the design entities and the relationships that shows the data representation. The relationship is an association between two entities, about which the data system stores, maintenance, and display. The main task was to identify the entities within design description. Figure 5.4 illustrates the universal conceptual model of entities relationship for circuit, wire, and component. It consists of two main types of entities (i.e. schematic and graphical entities) and are described below.



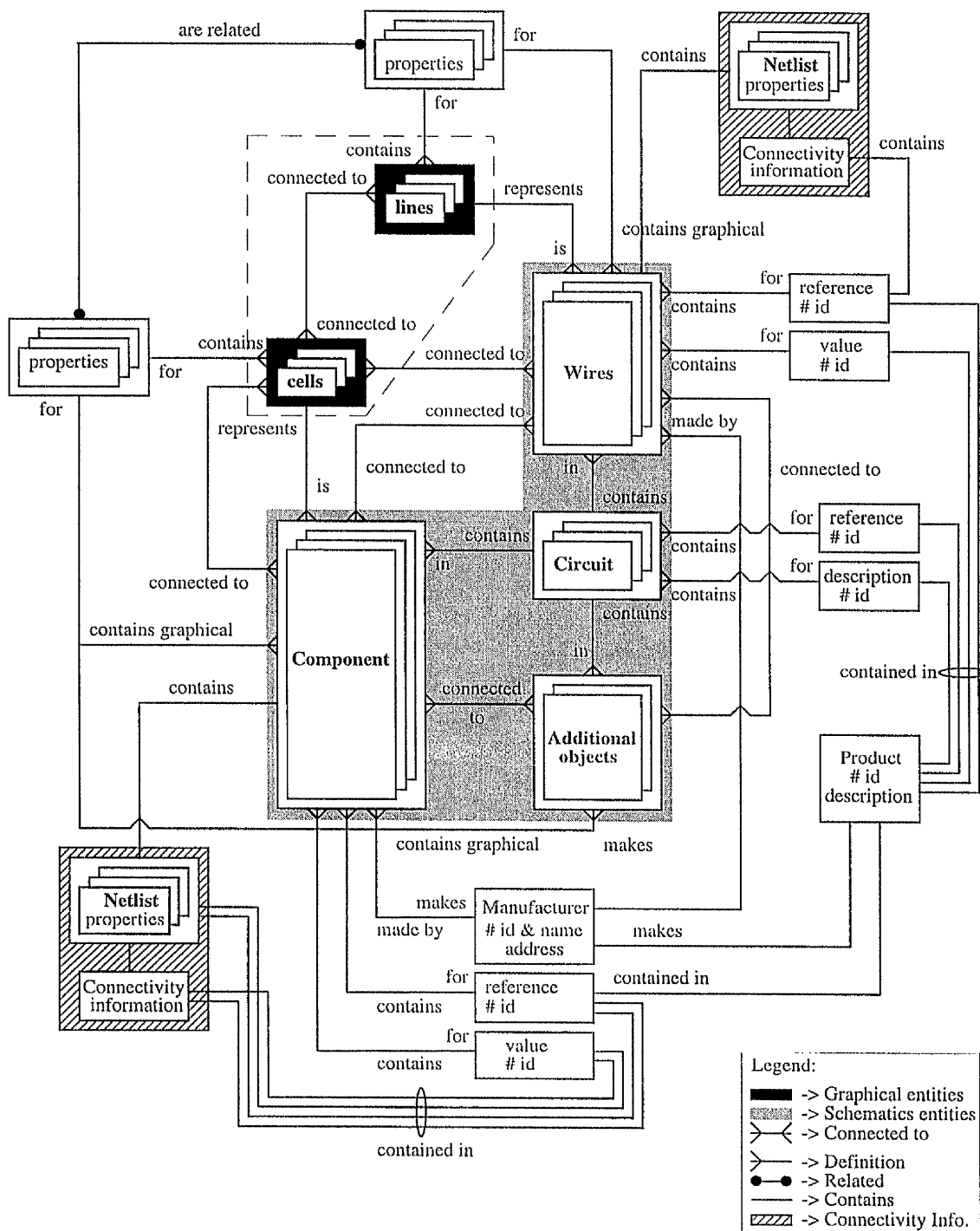


Figure 5.4 Illustration of entities-relationship conceptual model.

The conceptual model described above contains detailed description of all information that are needed to be extracted or edited to describe schematics elements. It views the whole environment description as entities and relationships among them. Given a schematics diagram, the information upon which the model works are extracted through software written programs or user defined through an interface dialog. With reference to Figure 5.4, Schematics entities shown in the shaded area, owns a set of information.

Relationships between these different elements are expressed as (connected to, defined, and contains). They are regarded as electrical/electronic circuit descriptions that can have their own attributes. For example, specific components can have attributes such as (manufacturer name and identity, reference of designation, and values, etc.). This can also be applied to circuit wires or a circuit in general. The figure also shows the graphical entities (i.e. cells and lines), provide an outline of the schematic entities. For example, cells resources which are used to represent components symbols and other relevant objects could have their own graphical properties. These properties can be extracted by means of MDL scanning programs (as mentioned in Chapter 2) to assist in developing connectivity reports and netlist generation. Lines which are used to represent wires that connect components have their own properties that can be extracted to assist in developing design reports and netlist generation.

### **5.6.2 Relationships Determination**

Determining relationship is a matter of determining the different views of the conceptual model. Relationship can be determined using related tables that contains common attributes between them. Identifying entities is considered to be the most important step in the whole process. It is the physical feature represented in the design file that needs non-graphical attributes (i.e. text) attached to it. The non-graphical attributes are described in a table, or set of related tables. Designing of data table is based on the properties of the entities represented in the design file. Once identifying design entities, attributes will become the columns that represent properties in a table.

### **5.6.3 The use of Structural Query Language**

The structural query language (SQL) is a powerful language that is the industry standard for database access and data manipulation. It is used to interactively query a database directly from within MicroStation. The MicroStation strategy is to continue improving the database tools through allowing more database choices, better development tools, and further simplifying user interfaces [5.13]. The areas in which MicroStation taken into consideration including dbase SQL support.

SQL was developed to support the relational database model. Most database implementations, including Oracle support a direct SQL language interface. Two basic classes of SQL statements are used in developing the system database applications, namely: Data definition language (DDL) structures, which are used for setting up database tables and fields. It also allows data queries to be created by joining data fields in one table with fields in other tables. The second class is data manipulation language

(DML) statements, which are used to work with the build data supplied by a design, including performing selects, updates, and other functions. Figure 5.5 below, shows the organisation of the used database.

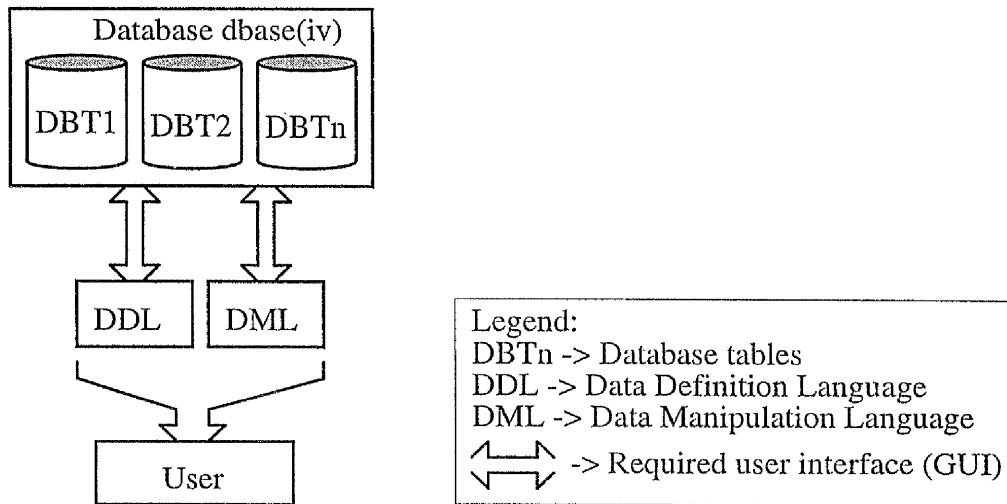


Figure 5.5 Organisation of database.

The SQL toolkit supported by MicroStation includes the ability to take SQL statements generated by graphics functions and interactive requests such as queries and MDL database functions calls. The primary role of the database server is to process SQL statements. All SQL statements recognised by the database software are supported, as shown in table [5.1]. They are used in developing database applications for circuit components and wiring.

Table [5.2]. A summary of SQL statements [5.14].

SELECT	Query database tables
UPDATE	Edit rows in a table.
DELETE	Delete rows.
INSERT	Add new rows.
CREATE	Define a new table and add it to the database.
DROP	Delete an existing table.

## 5.7 DESIGN DEVELOPMENT

Describing the application in a comprehensive manner is one of the most important problems in database design [5.8]. It could be an interactive tool allowing the database

administrator to define objects attributes. The theory of developing an application is the definition of attributes to form entities. Also the associations between entities could be specified and other constraints are expressed.

The primary objectives of database in developing new applications is to organise the data in a way that makes it convenient to store and retrieve. The need for fast retrieval and effective storage utilisation is a fundamental issue, that is, the requirements of speed usually incur a penalty of space. This is because data retrieval aids such as attributes and links takes up extra space; however without them the entire database may have to be searched item by item in order to retrieve a single piece of information. The necessity of data interfacing and elements linkage is explained in the following sections.

The intention is for the database to be the authoritative version of the design. Therefore, it can be integrated and control the various activities. 2D Schematics and 3D modelling layouts are derived from database. This means that the database must be flexible in allowing information to be added. Figure 5.6 illustrates the use of database dbase(iv) for supporting different design activities.

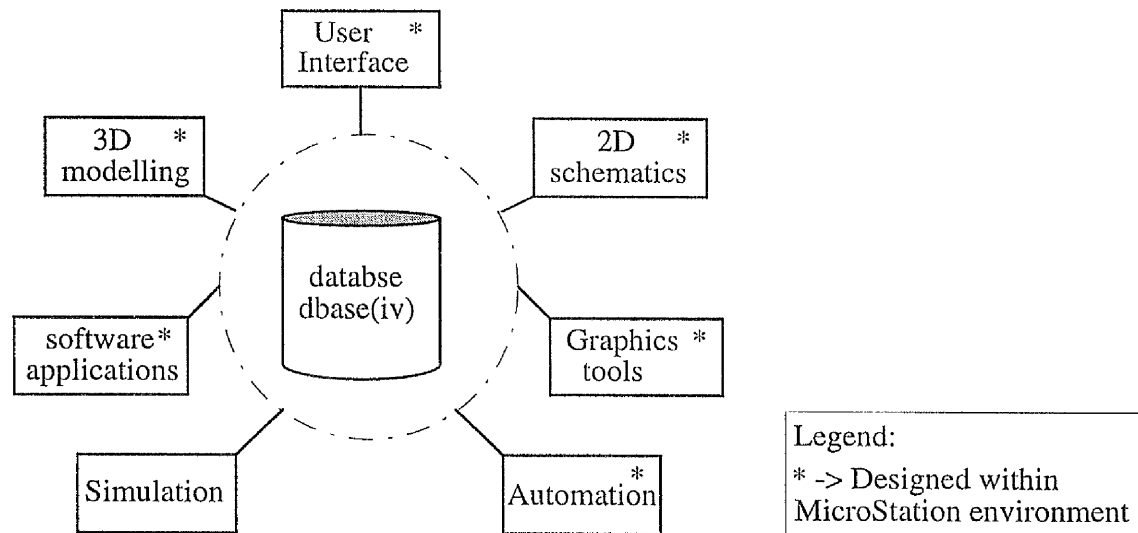


Figure 5.6 Database dbase(iv) used for different design applications.

The integration of CAD database applications considerably improves design productivity and can also provide many additional benefits, such as design error checking and cost analysis [5.15]. MDL and user commands (UCMs) language can be used to develop custom links to common database software giving a user friendly database link within the MicroStation environment. MicroStation also provides support for several major database packages that run on several different computing platforms. Figure 5.7 illustrates the links between the MicroStation and other database packages.

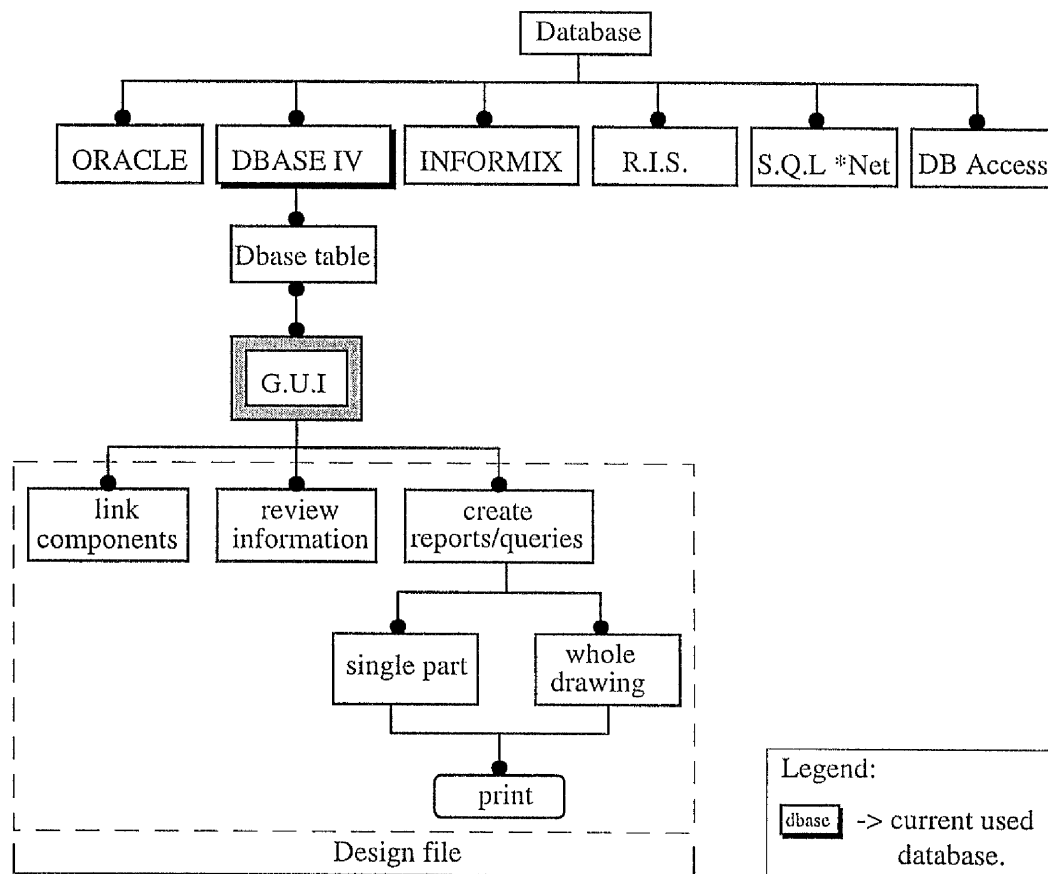


Figure 5.7 Illustrates the links between MicroStation and dbase(iv) environment.

Database has been applied to a wide variety of application fields. Areas, such as engineering designs requires database systems that manipulate data structures and special purpose database operations. The key issue is to provide the facilities to define new data structure, operations, and integrate them into a unified database systems. Figure 5.8 illustrates the general concept of database tools design. There are three main functions designed to facilitate the system's design environment, namely for: reports, update, and manipulate.

The centred database dbase(iv) views the integrated system as three key elements of equal importance: Database tables that stores designs and their information, user interface, and set of application functions needed to process these data. In addition, MicroStation resources such as design files, cell libraries, and non-graphical database, plays an important role in the connection between database and design environment.

This section restricts its self with the design of different database applications, which are fully explained in the following sections.

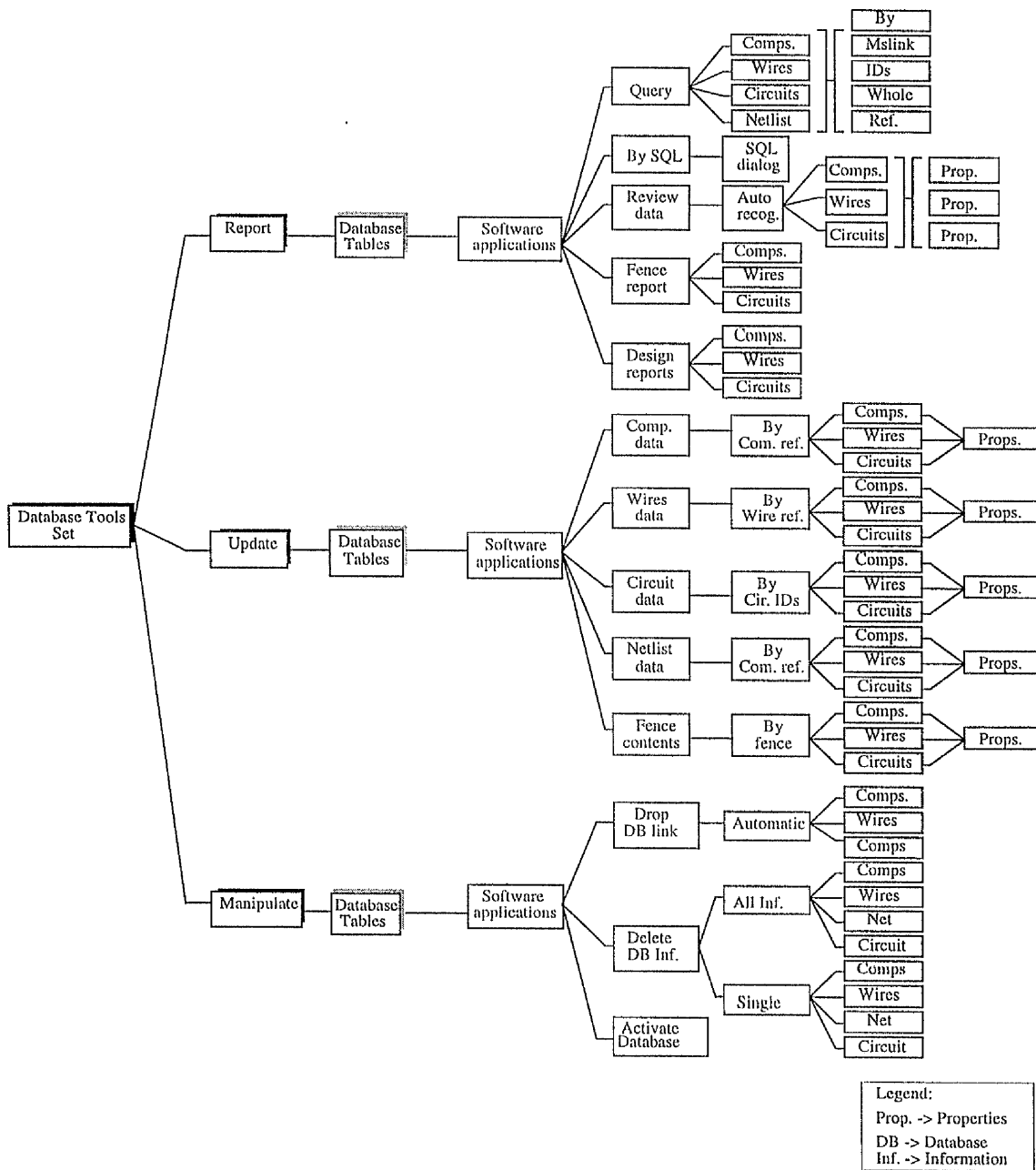


Figure 5.8 Illustration of database tools design plan.

### 5.7.1 Applications Design Methodology

Attempts have been carried towards the development of a practically useful design methodology for database to support engineering application [5.16]. Most of these attempts are based on modifying or extending tools which have been developed for business applications. In general, however, few attempts exist for a design methodology which takes into account the nature of engineering data and the complexity of its structure.

Design methodology of database applications has been the focal point in software development. Mohammed et al. [5.12] outlined some of the features that design methodology for engineering database applications should have, these are as follows:

- A comprehensive data model for handling complex objects. This involves designing equipment or parts of the overall process.
- Completeness. Design methodology should address all the relevant design questions.
- Modularity. Different sections of database systems should be capable of being modified, manipulated, and updated. Therefore, special designed application are required to perform these tasks.

The design methodology is based on the conceptual model previously described in Section 5.5. The design 'core' for developing these applications consists of five main software programs. With reference to Figure 5.9, they are described as follows:

- Extract-Store. Applications of this kind involves the storage of extracted data from 2D schematics and 3D modelling workspaces.
- Extract. Applications of this kind involves extracting the final data for reports and queries formats.
- Extract-Create. Once design information are extracted and stored, different ways of report generation and queries creation can be performed. Applications of this kind involves the creation of design queries about specific component, wire using references or generating complete design reports.
- Update. Elements information have to be updated after editing new data. Applications of this kind are performed either using dialog boxes or user fence defined for complete or partial design.
- Manipulate. Design involves databases of any kind can require changes that have to be carried at some stages. Applications of this type involves the removal of unwanted records, database tables invocation, and automatic elements linkage dropping.

Generally, the applications described above provide a means to carry out the following tasks:

- Establishment of a direct linkages to database.
- Linkages manipulation, this involves adding and removing of data records.
- Reports generation on database attributes.

- Displaying of database attributes in the design environment.
- Performances of graphical operations based on database search criteria.

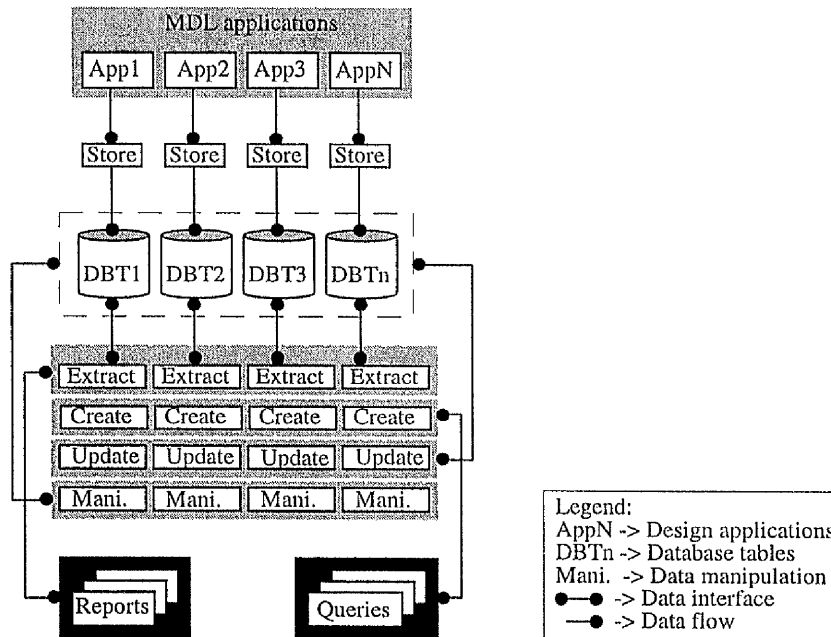


Figure 5.9 Applications design organisation.

In addition, database tables can be organised to share data storing as illustrated in Figure 5.10.

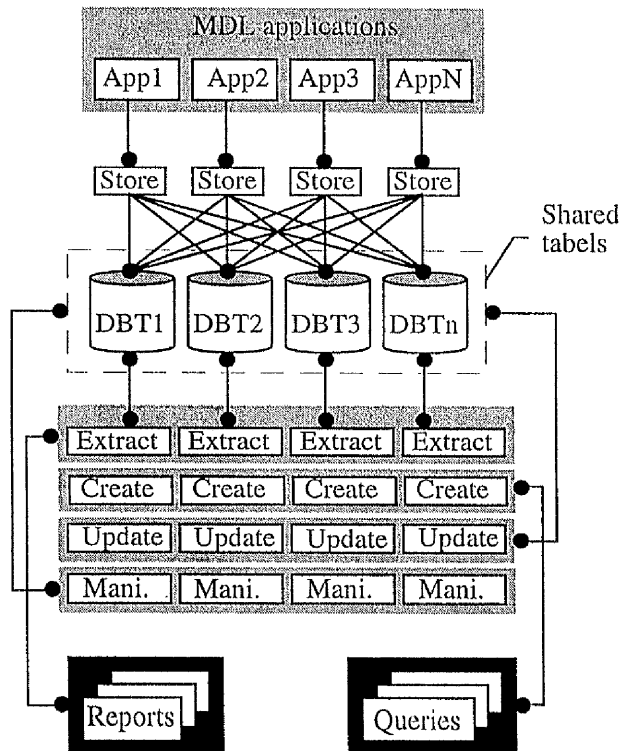


Figure 5.10 Tables are shared between applications.



### 5.7.2 Database Editing Interface

Database interface is defined as a set of tools allow graphics elements to be linked to existed row of information. Using the database interface offers many capabilities available to develop CAD applications. These range from simple applications that tracks part numbers in an assembly to a sophisticated facility management system, which tracks the operation and maintenance of a large organisation [5.14].

MicroStation has developed a better, stronger interface for dbase(iv). It is being built using the SQL database interface toolkit. It provides a two-layer approach, consisting of an MDL component and external program component. The MDL handles error checking, string resource management, and user interface. The external program component uses embedded SQL/C tools provided by the database vendor. It processes SQL statements, such as SELECT statements (UPDATE, DELETE) and SELECT cursor, and also screen forms.

### 5.7.3 Setting up and Accessing the Database

Figure 5.8 of this section, provided a general design plan of the database tools design and development. Generally, the link of these tools can be achieved through data communication between design workspace and database. The communication link itself is responsible for accepting requests from the user for data manipulation, transferring data back and forth, and reporting results queries to the design environment by means of SQL statements. Figure 5.11 provides an illustration of the general architecture of communication link. It consists of three different levels: The first level is applications that needs to be invoked. The second level is the database driver, which supports the linkage between the dbase(iv) and applications through dbase(iv) built-in server. The driver is responsible for checking and executing of all SQL statements needed for accessing or editing of data. The third level, is database files which considered to be driver-independent, they are used to store design information that are edited or extracted.

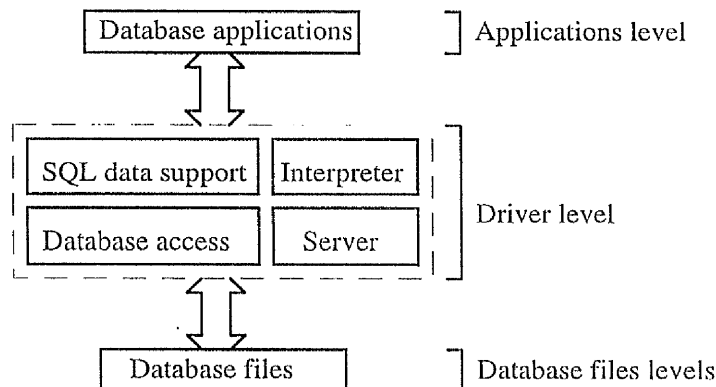


Figure 5.11 General architecture of database communication link.

In order for a software to be successfully developed, the starting point is to establish the link between data tables and design environments. This has been established by creating the following files:

- Control file (MSCTRL.dbf). This is used to keep track of all the involved data tables to be linked with design workspace.
- Database tables with mslink field (i.e. MSLINK is to allow graphics elements to be linked with their relevant information).
- A batch file used to compile these file to ensure the link is achieved.

#### 5.7.4 Linking Circuit Elements to Database records

There are several ways used to link graphics elements to database records. Figure 5.12 provides illustration of how different circuit elements and 3D assemblies are linked. For example, in one to one relationships, there is a link from one element to one record in the database. When there are many elements pointing to one record, this is a many to one relationship. In one to many relationship, the same element has a link to many different database. The entity number is to distinguish one database from another.

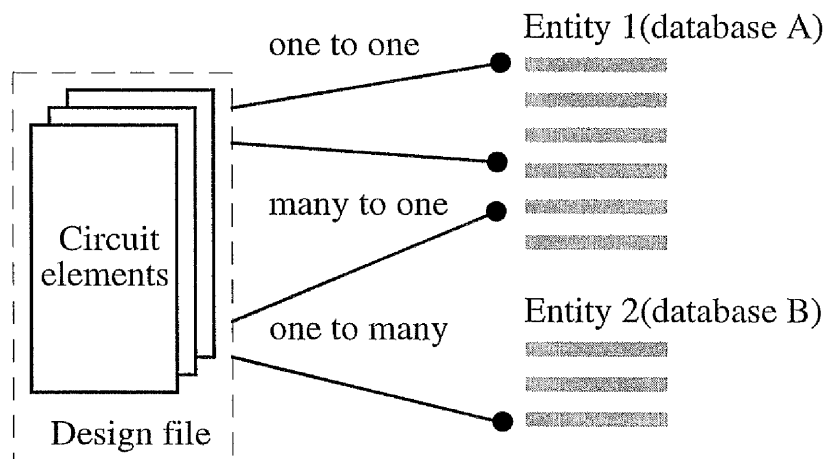


Figure 5.12 Element to record linkage.

#### 5.7.5 Data Tables Design

Designing tables for a non-graphical database that will be used with schematics design files is no different than designing tables for any database application. Thus, existing, or editing new database information can be adopted. The only specific requirements are a control table, which contains rows of data corresponding to each table that has linkages, and a key column named as mslink.

### 5.7.6 Automatic Element Recognition

Design data needs to be edited for a selected schematic element made by the user. For this purpose, an automated element recognition program is developed to differentiate between the different elements that are used in designing circuit schematic (i.e. wires, components, power source, etc.). Figure 5.13 shows the program flowchart written in user command language, activated through the selection of graphical icon. By selecting a schematic element, the program task is to recognise the element type and the corresponding data dialog is displayed and data are edited.

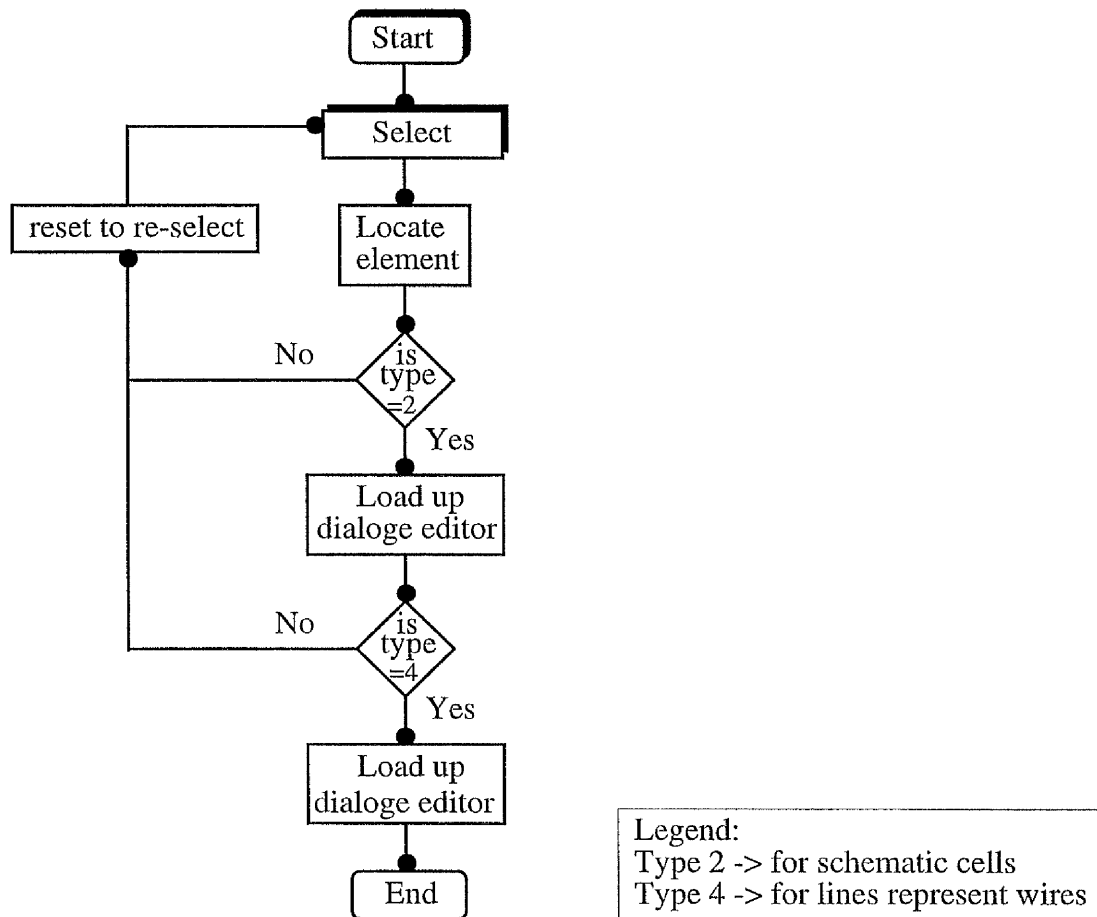


Figure 5.13 Flowchart of element recognition for data editing program.

## 5.8 DATA QUERIES AND REPORTS GENERATION

Reporting and query tools are essential to any database applications. Once design data has been edited into data tables. It needs to be made available for reviewing. Therefore, The intention of this section is to investigate the different software applications that have been designed and developed, thus, providing the necessity of integrating database systems with ECAD design environments. In addition to help understand the

functionality of these applications, detailed explanation along with diagrams is outlined below.

### 5.8.1 Design for Data Query

In order to simulate the development of database applications that are able to meet the demands of CAD/CAE, concurrent engineering. Study and normalisation to these applications is essential. This has been done by providing a framework of classification of queries. The design of these queries has been classified into four different classes, viz: Create component query, wire query, circuit query and netlist query. This classification reveals a rich structure of complex engineering structure, many of which require query searching routines. Figure 5.14 provides illustration of database query tools design.

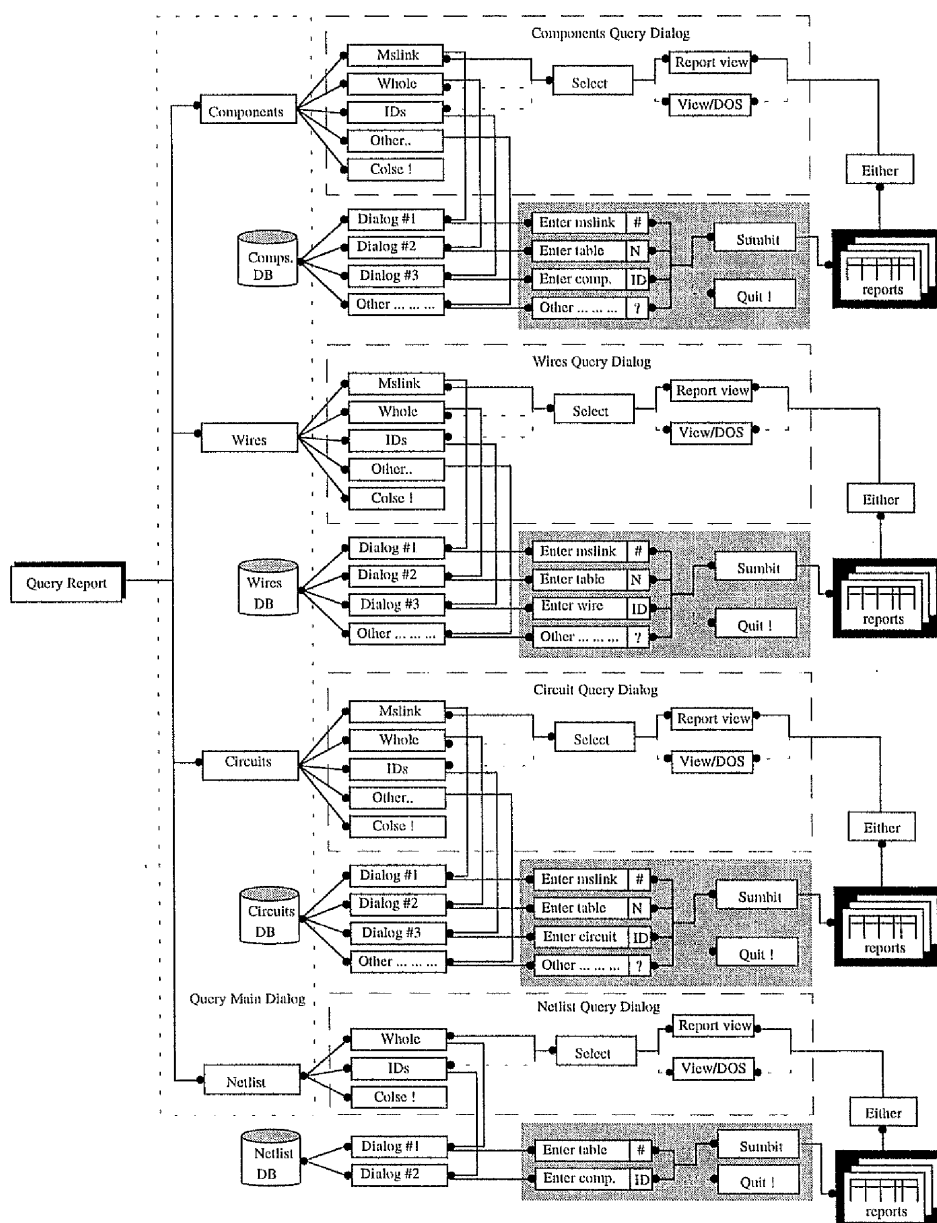


Figure 5.14 Illustration of data query design.

As engineering systems become more sophisticated, and the demands of concurrent engineering design require greater integration between design, analysis and manufacturing tools, the need for effective engineering data management become more critical [5.17]. However, the design of data query provide powerful tools for querying design data. As for example, widely used database systems provide powerful tools used for modelling and query formatted data. Applications such as CAD, CAM, CAE, and VLSI design, have been proven to be inconvenient and insufficient. This is due to the complexity of data structure provided by these applications and the simple organisation of relational database systems. Therefore, what is required is more complex database applications for the effective manipulation of application data.

Design data query is produced using three different methods, namely: Query by mslink, Query by IDs (i.e. references), and Query by table. By selecting the desired query function from query main dialog, selection of different queries is displayed. The query starts by entering a set of words, which are likely to be contained in relevant documents. Written programs for each type of query automatically determines weights for those words, and applies the resulting query to the database. As a result, documents are retrieved and displayed. Figure 5.15 illustrates program flowchart for design queries extraction and Figure 5.16 shows the different dialog boxes used for different design queries.

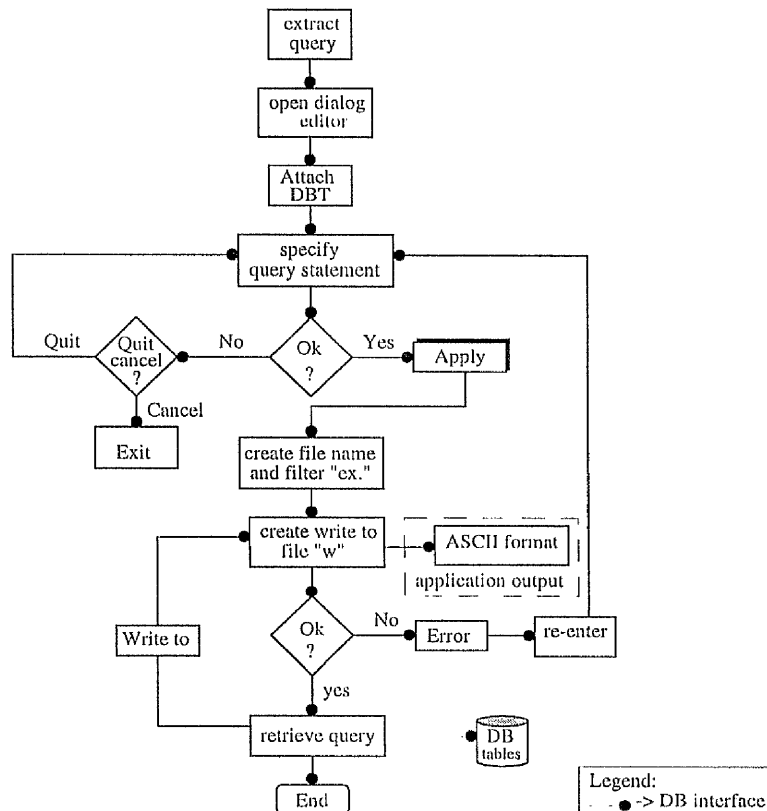


Figure 5.15 Flowchart of query extracting program.

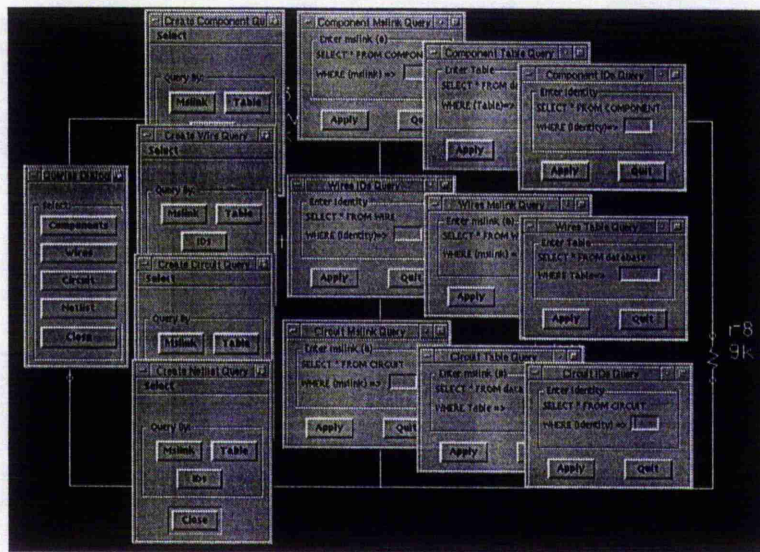


Figure 5.16 Different user dialog boxes used for data query.

The development aim was for these tools to generate data queries. To evaluate the success in achieving this aim, an example of electrical circuit was schematically constructed. These tools are applied to find all components contain the same reference designation or the same value. Netlist query can also be created, for instance finding all the components that have the same node numbering. Figure 5.17 demonstrates different requested queries for circuit components.

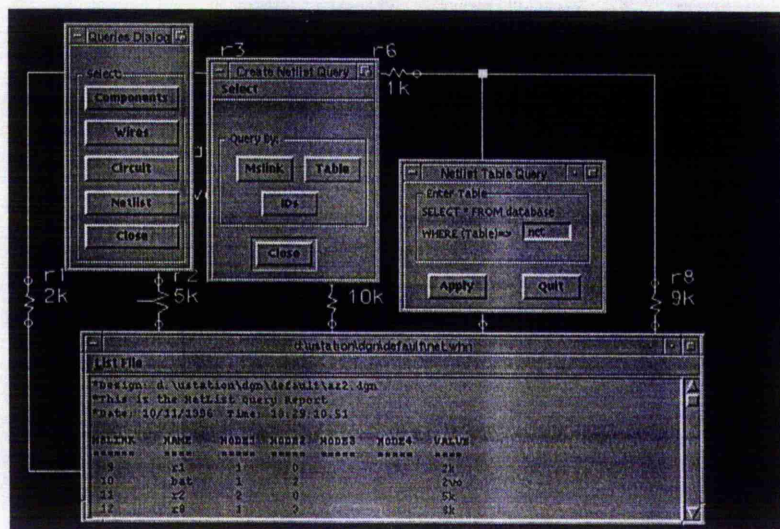


Figure 5.17 Demonstration of components-netlist queries.

### 5.8.2 Design for Data retrieval

As it has been mentioned in Chapter 2, CAD/CAE systems provide the capability to extract data from schematic drawing to automatically generate various design information. The information can then be tailored and reports can be generated. The



extracted data comes from the attributes assigned to different parts or symbols used in constructing the drawing.

There have been many attempts to extend database systems, so that they can be used to handle data in flexible manner as reported by Christos [5.18]. Data retrieval method is a scanning technique used to retrieve design information as a result of given queries statements. The main advantages of using this method is the retrieval speed. Despite the existence of some fast string search software, scanning of a large database may take a long time to retrieve.

Data retrieval applications are also require the link between design workspace and database environment. By selecting design element, the corresponding data is retrieved and reviewed. The application is driven by means of a UCM program written to recognise circuit element type. Figure 5.18 shows detailed illustration of data retrieval design and Figure 5.19 shows flowchart of element recognition program.

The functionality of data retrieval provided through the use of review command and control functions main dialog. There are three main functions and each performs different tasks:

- Components function: used to retrieve components data.
- Wires function: used to retrieve wires data.
- Circuit function: used to retrieve circuit data

The procedure of how this application works is by selecting a control button. The UCM program identifies element type and insures the correct selection. Data attributes are then displayed using SQL window. As a result, data attributes are reviewed as shown in Figure 5.20.

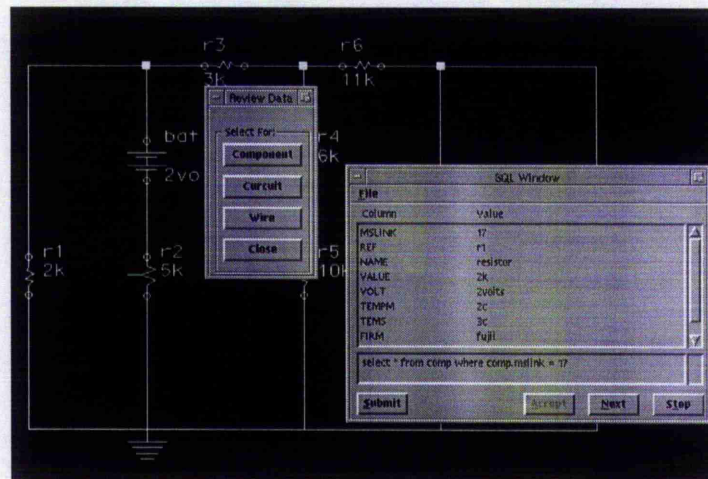


Figure 5.20 Demonstration of design data retrieval.

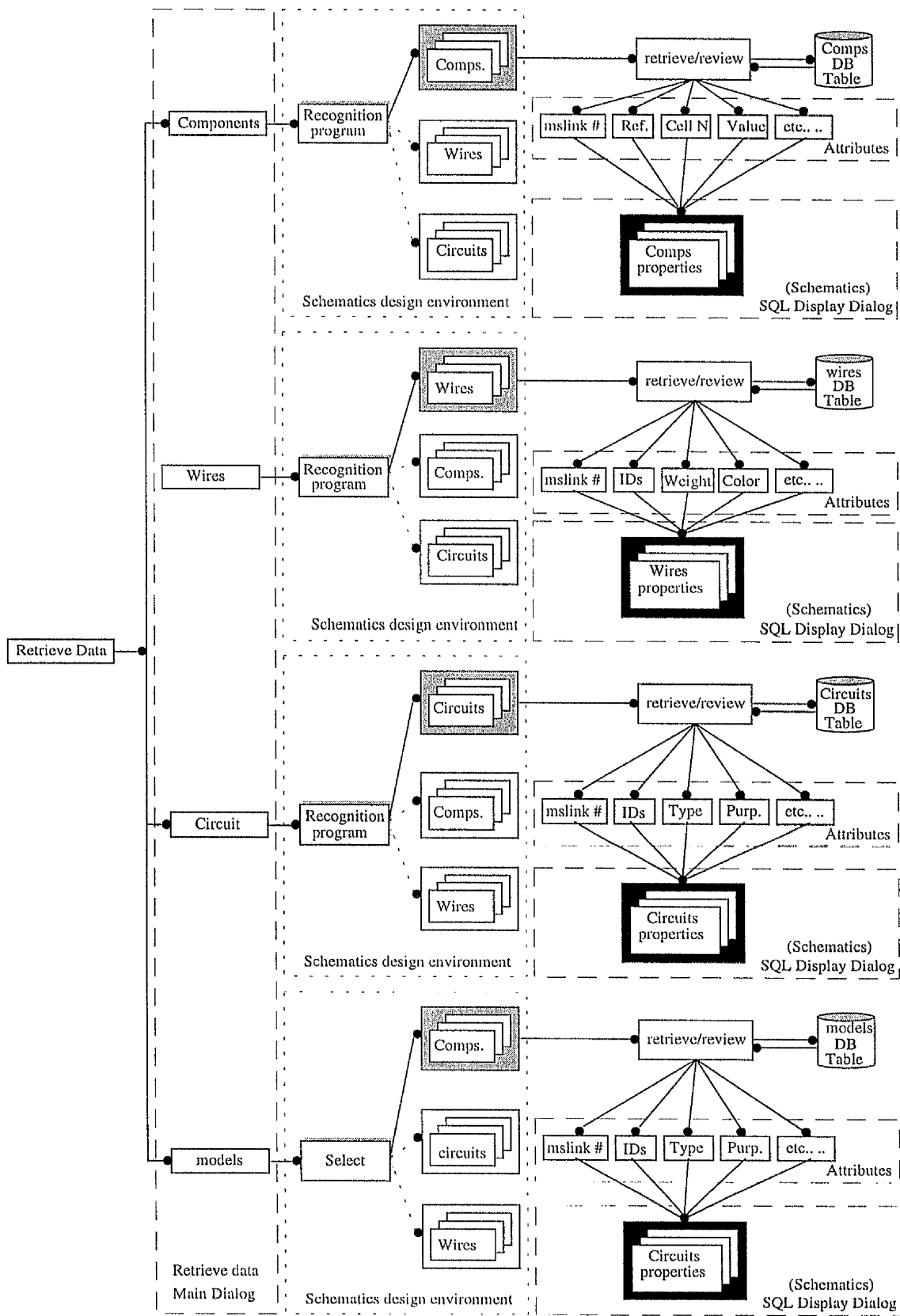


Figure 5.18 Illustration of data retrieval design.



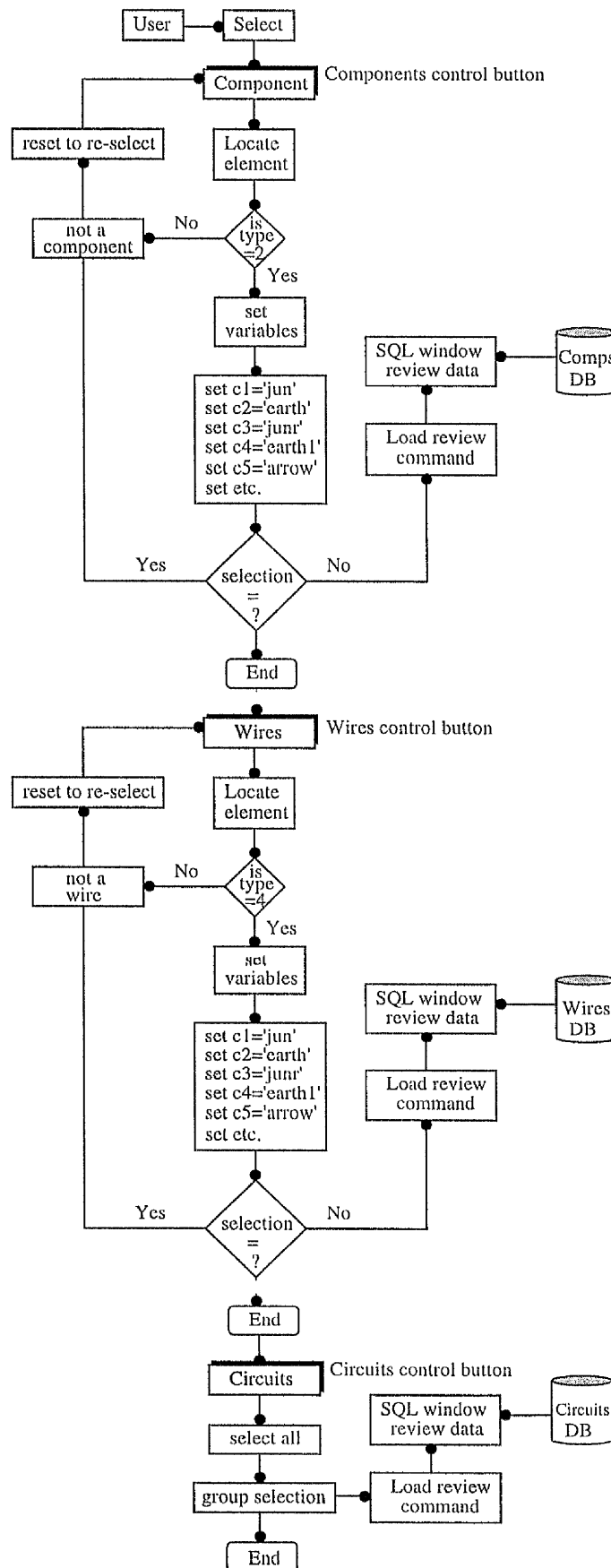


Figure 5.19 Flowchart of element recognition for data retrieval program.

### 5.8.3 Design for Reports Generation

Reports generation is another application has the ability to gather reports information from database tables. Figure 5.21 illustrates reports generation design concept. Many design reports can be generated including for components (data, list and B.O.M.), wires (data and wiring list), circuit netlist, and drawing information reports.

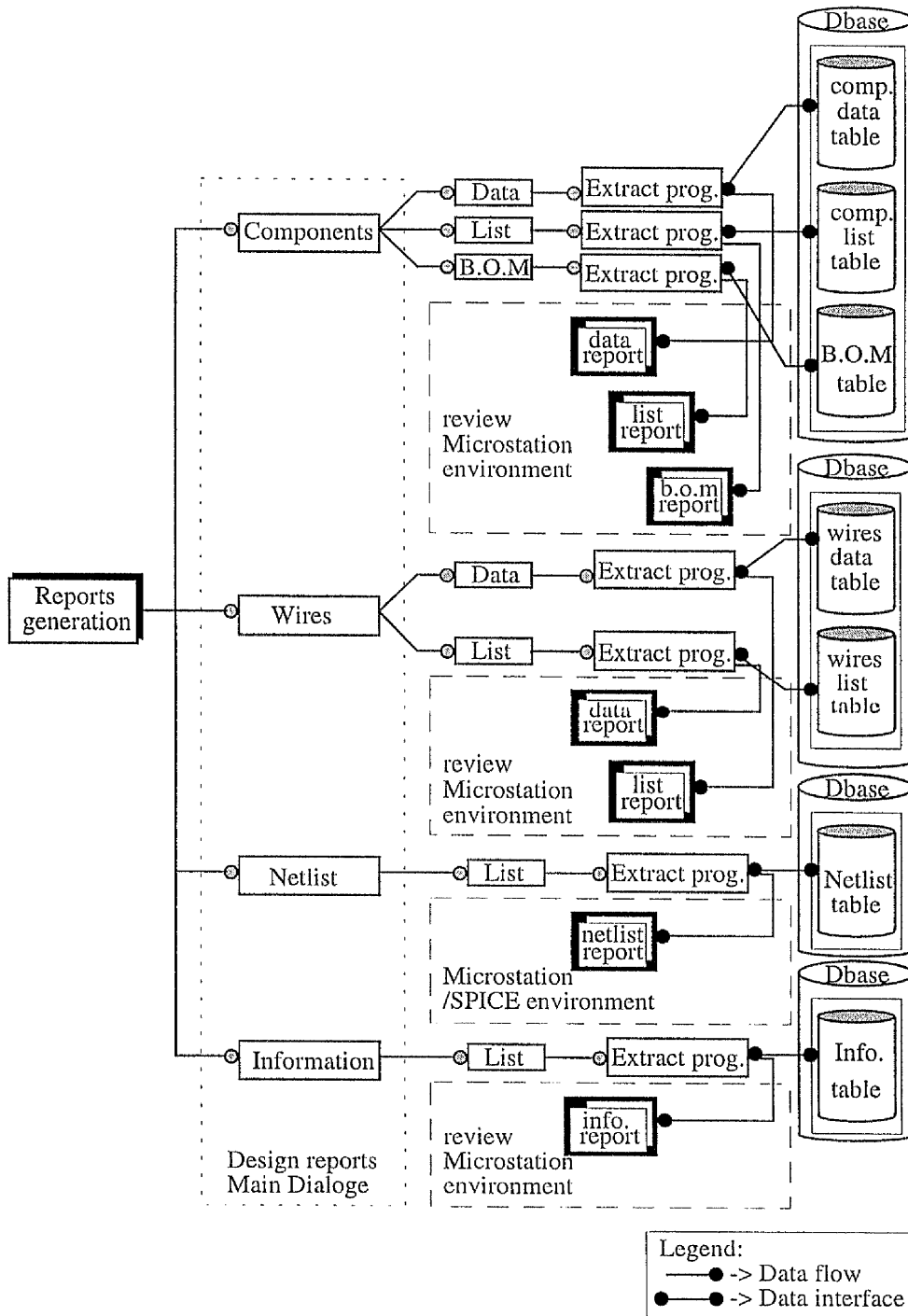


Figure 5.21 Illustration of reports generation design.

### 5.8.4 Design for Fence Reporting

Fence reporting is a method used to generate reports for circuit elements such as components, wires. It is one of database applications developed using the MDL language. It is another way of generating design data reports. Similarly to the other applications developed for this purpose, an effort is being made to make it as flexible as possible, so that they can be applied to different variety of design data.

Design for reporting by fence allows the user to create reports of all circuit elements within a user defined fence. To achieve the desired results the link of these elements should be provided. Using this applications, reports are generated for complete or partial circuits. Figure 5.22 illustrates detailed structure of fence reporting design.

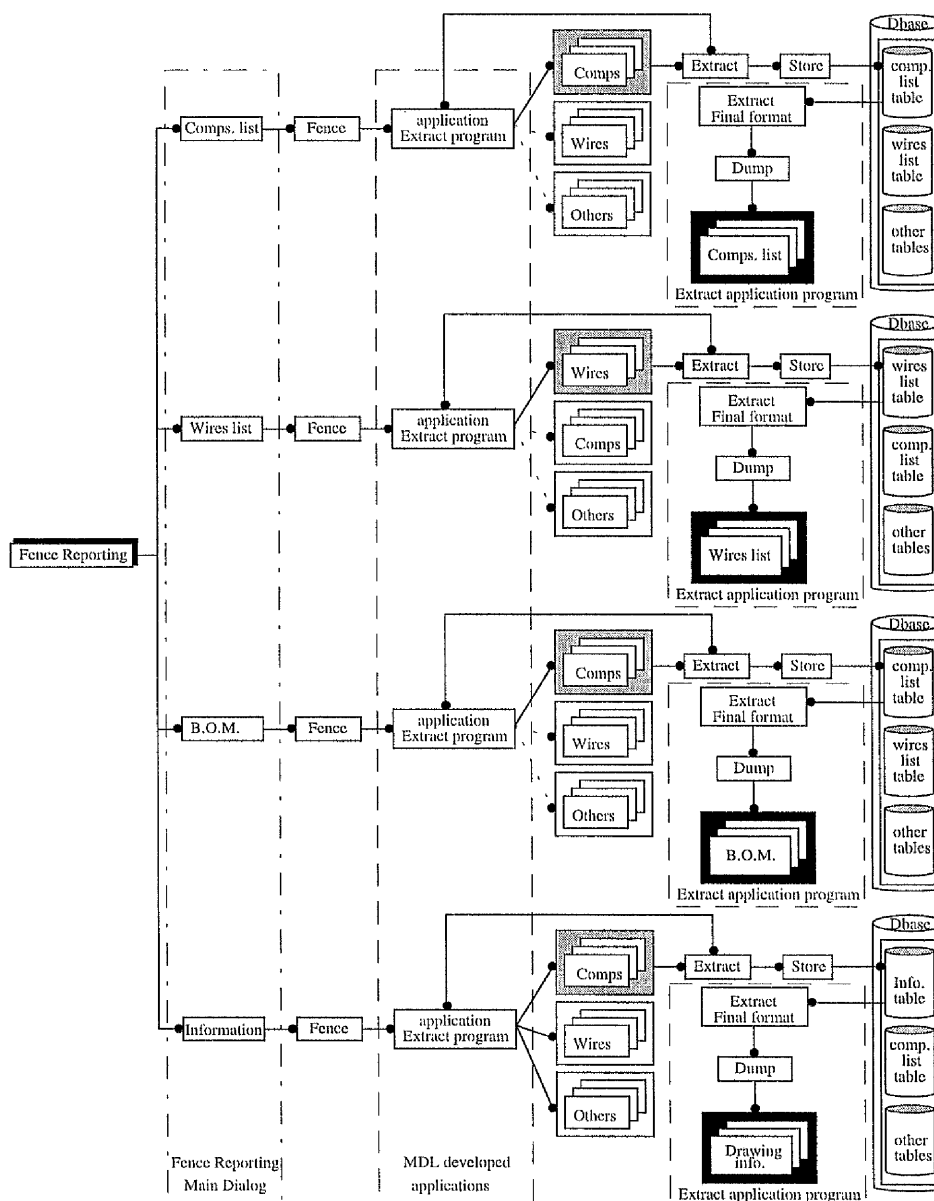


Figure 5.22 Illustration of fence reporting design.

The development of this design involves four different applications each of which handles different tasks of generating design information. Design reports such as component, wire lists, bill of material, and schematics graphics information can be generated using two separate programs, which are described as follows:

- Extract to store: Information contained within a fence are extracted and stored in database tables.
- Extract to generate: Once data for a particular report are stored in data table. This program is then applied to extract and dump reports final format.

The graphical dialog contains the functions used to execute these programs is activated through the main interface. It consists of different control buttons used for extract-store data and pull down menus used for extract-dump final format. Figure 5.23 shows the control dialog boxes used for generating design fence reports.

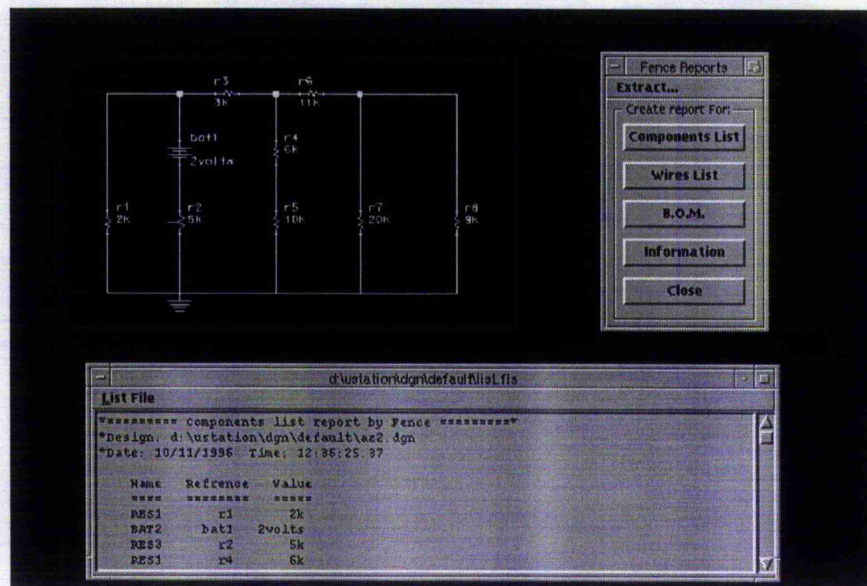


Figure 5.23 Dialog boxes used for fence report generation.

Using these applications, it was concluded that the ability to create and manipulate schematics elements with fence reporting is another powerful and useful aspect of data report generation and database linkages between different environments.

## 5.9 DESIGN FOR DATA UPDATE

Sunil K. [5.19], has described data update as an efficient approach developed to update database of design elements. Two data updating approaches are designed and developed, brief description as follows:

- Data update using graphical dialog. This approach has been written in a way that it instructs the user by first selecting a component and second by entering the element reference followed by the data needs to be updated. Figure 5.24 illustrates the general concept behind data updating process, whereas Figure 5.25 provides flowchart of the software steps taken in designing this approach and Figure 5.26 shows demonstration of data update using dialog method.
- Automated fence updating: This approach was designed so that schematic designs could be modified at any stage. It has been designed to automatically update design information within a user defined fence. A detailed description of this approach is outlined below.

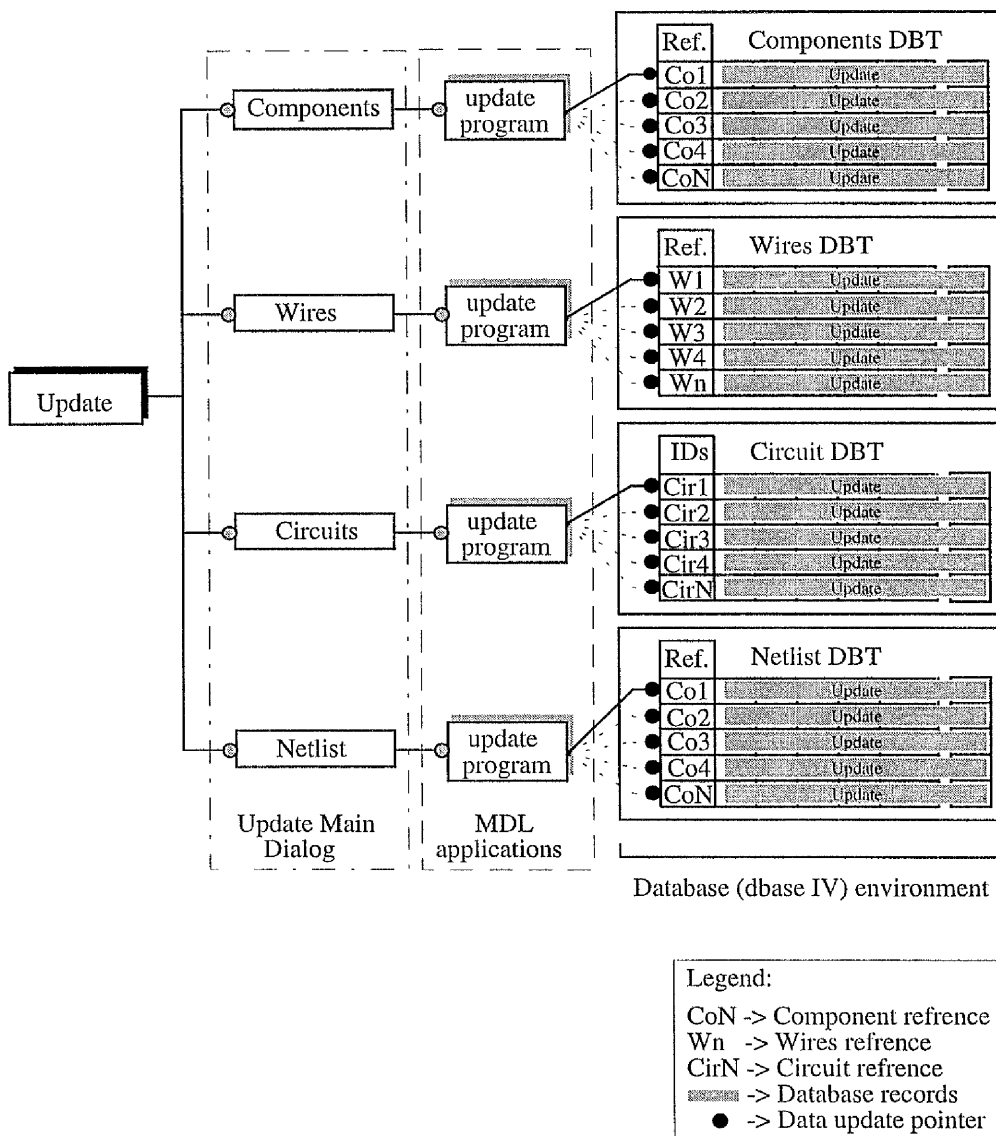


Figure 5.24 Illustration of data update design.

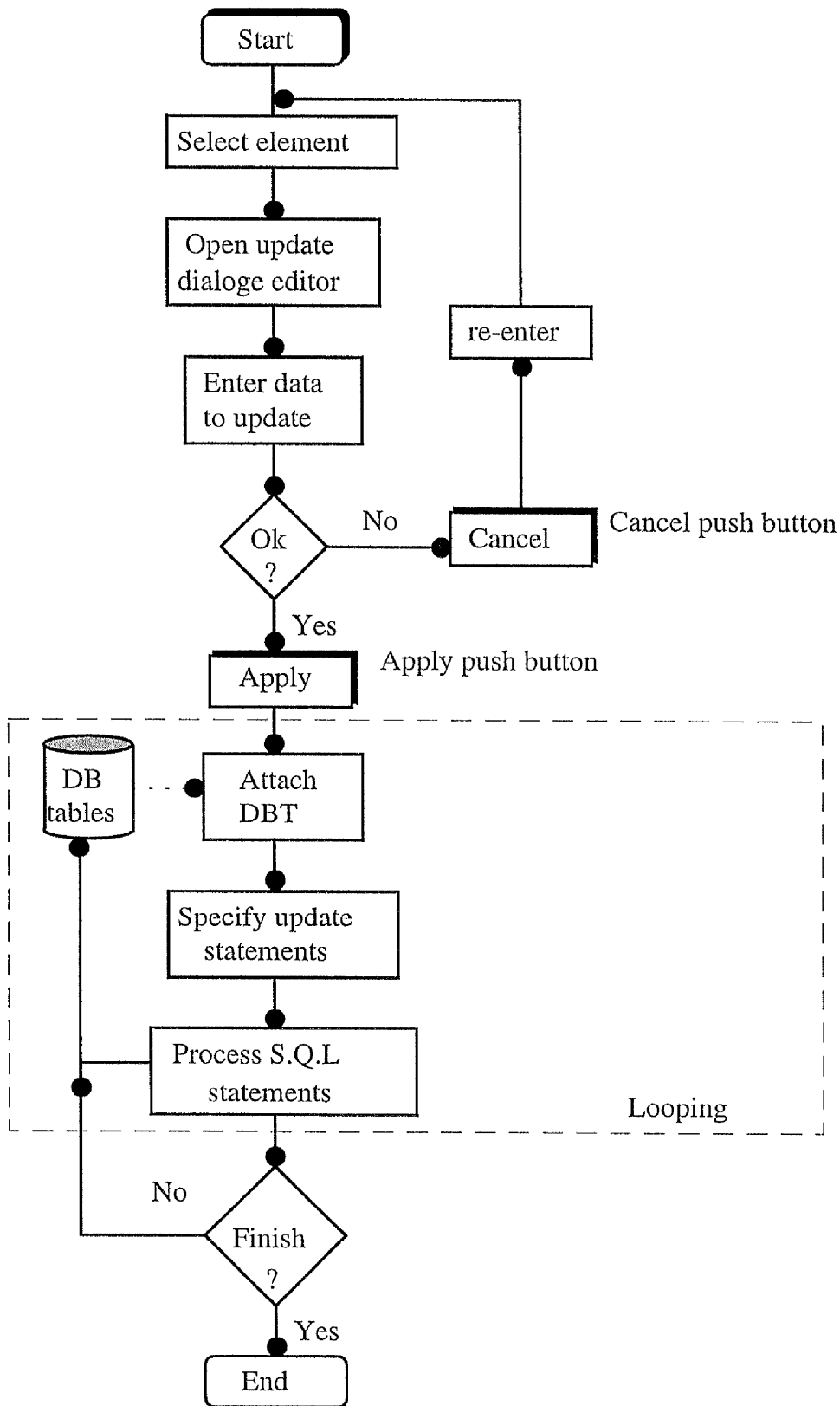


Figure 5.25 Flowchart of data update program.



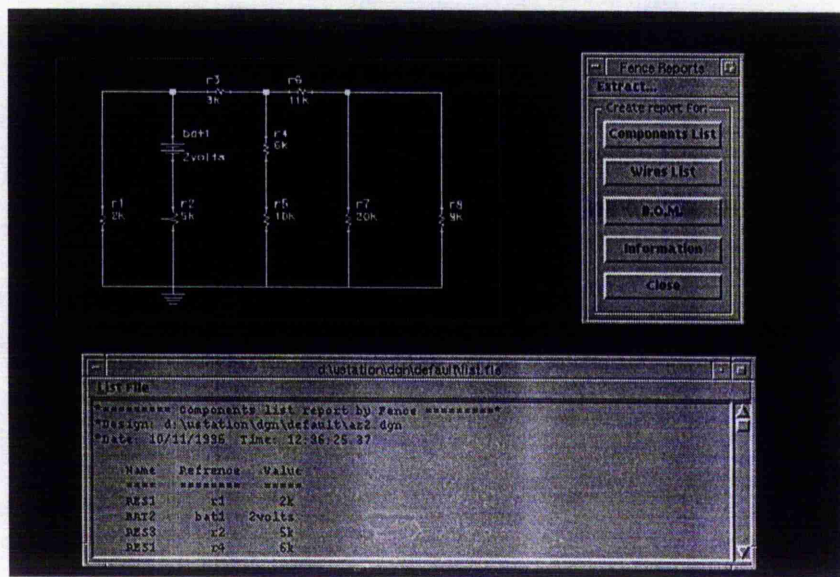


Figure 5.26 Demonstration of design data update.

### 5.9.1 Update by Fence Design

Update by fence is another application developed to update data. By defining a fence around complete or partial schematic circuit, data are automatically extracted and updated. Figure 5.27 illustrates the detailed concept of fence update design. Figure 5.28 provides demonstration of using this application.

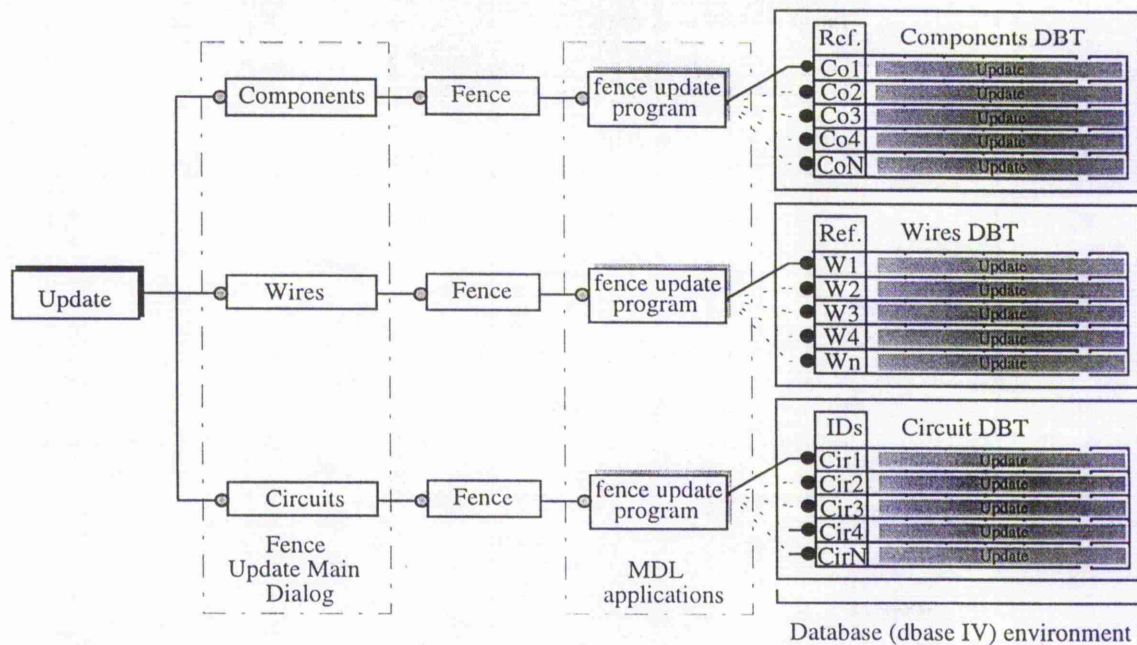


Figure 5.27 Illustration of fence updating design.

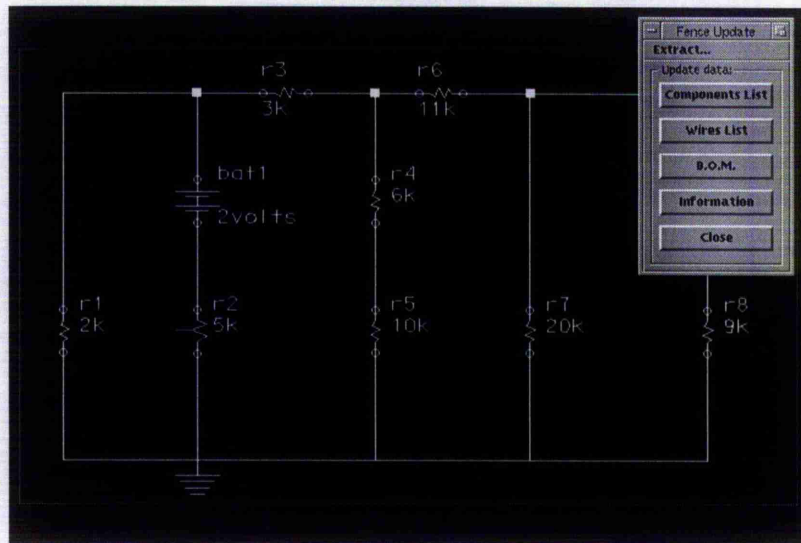


Figure 5.28 Demonstration of fence update design.

## 5.10 DESIGN FOR DATA MANIPULATION

Design of data manipulation is another technique under which data manipulation language (DML) is used. It is designed to provide access and modification to stored data. The method of achieving this aim was by using the structural query language (SQL) language to assist the development. Manipulation tools such as delete, activate database, drop linkage based on software written programs. Description of these tools is outlined below.

### 5.10.1 Data Deletion

Data deletion or removal is a tool that has the ability to delete unwanted design data from tables within MicroStation without accessing dbase(iv) environment. Drop schematics elements linkages is to allow new data connections to be defined and edited. Figure 5.29 illustrates design concept of data deletion design. Figure 5.30 illustrates dialog interface used for data deletion.



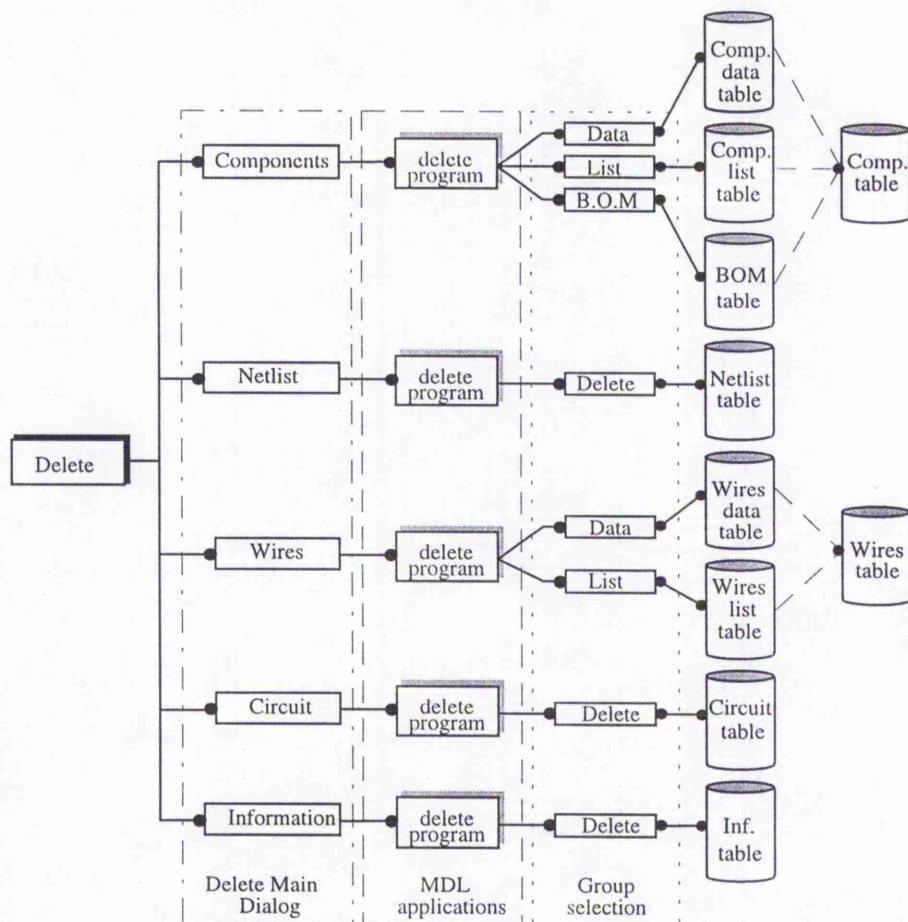


Figure 5.29 Illustration of delete data design.

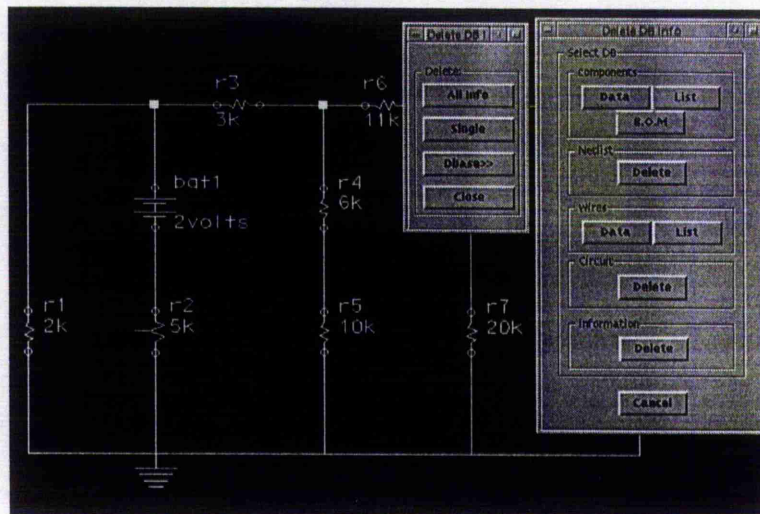


Figure 5.30 Dialog for data deletion

### 5.10.2 Automatic Data Linkage Dropping

Data dropping is used to detach linkages from circuit design elements. Once a linkage is detached, the element is no longer associated with database row. Figure 5.31 illustrates

the program flowchart designed to automatically drop database linkage for components, wires, and circuits elements or as group of elements. The program basically consists of three different control linked functions. To detach linkage from a design elements, the program intends to scan through schematics design file until the specified element is reached and detach it from database. For example, the type 2 means that the scanning procedure is associated with cells components only, while type 4 associated with type line string.

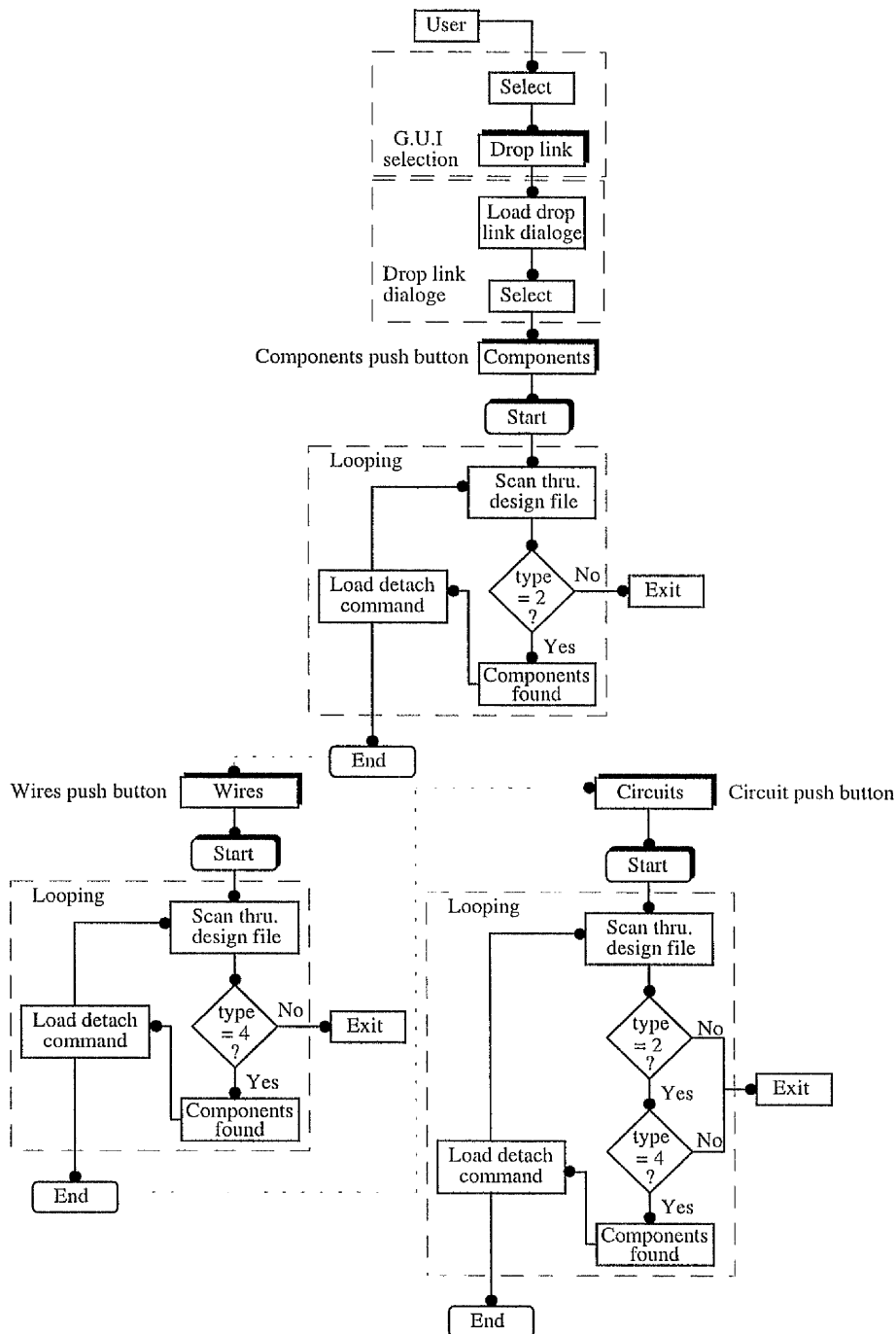


Figure 5.31 Flowchart of linkage dropping program.

## **5.11 CONCLUSION**

A key within the area of integrated environments is the information storage component. It is this kind of information that provides integration between tools by recording the data generated during design process. The objective of this work was to identify the utility of a relational database in engineering design and integration, identifying data management requirements. This indicates the fact that automated tools are required to help in data management and to perform the transformation from one representation to another. Therefore, designing database applications for the integration between different environments is necessary to bind the support tools together and ensuring transition from one environment to another.

In Section 5.5, a universal conceptual model is designed. The main task of this model was to represent schematic elements entities and their relationships. The purpose of designing a database conceptual model in this case, however, is not the approach by which the applications program processes the information. Instead, it is the combination of many ways to process the data for several database applications. Therefore, it is concluded that effort should be addressed towards the structure of design data and the relationships between the elements involved.

A database link has been developed for entering component data in electrical schematics drawing within MicroStation. Component data can be entered at any stage during the drawing process and reports generated for complete or partial circuits, as defined by a fence. Data are also updated using user dialog editor or a defined fence. Furthermore, by linking the circuit drawing environment to a database it is possible to produce an electrical netlist directly from database as explained in Chapter 4.

Finally, the main attraction of using a database for any application is that it provides centralised control of its operational data. For software engineering applications, the operational data includes all the information generated by the applications. In particular, design documentation and error reports must be recorded.

## **REFERENCES**

- [5.1] Don R., and Tony M., "Computer integrated manufacturing," Computer-Aided Engineering Journal, Vol. 4, No. 4, pp. 167-174, August 1987.
- [5.2] Yosihisa M., and Telsuo M., "An extended relational database system and it's applications to management of logic diagrams," Information systems &

Electronics development lab., Proceedings, twelfth international conference on Very Large Data Bases, Kyoto, Japan, 25-28, pp. 267-277, August 1986.

- [5.3] Mustapha K., "Database systems: their applications to CAD software design," Computer-Aided Design Journal, Vol. 15, No. 5, pp. 277-286, 1983.
- [5.4] San J., "A history and evaluation of system R," IBM research Lab., readings in database systems, ISBN 0-934613-65-6, pp. 54-68, 1988.
- [5.5] Codd E. F., "A relational model of data for large shared data banks," Comm. ACM 13, Vol. 6, pp. 33-64, June 1970.
- [5.6] Scott M., et al., "Functional specification for CAD databases," Computer-Aided Design Journal, Vol. 18, No. 3, pp. 133-138, April 1989.
- [5.7] Ramez E., and Shamkant B., "Fundamental of database systems," ISBN 0-8053-0145-3, 1989.
- [5.8] Barley S., and Cripps R., "Executive-centred system design for CAD applications," Computer-Aided Design Journal, Vol. 18, No. 4, pp. 12-18, July 1995.
- [5.9] Gio W., et al., "A database approach to communication in VSLI design," IEEE Tran. on Computer-Aided Design on integrated circuit and systems, Vol. CAD-1, No. 2, pp. 57-62, April 1982.
- [5.10] Susan D., U., et al., "Engineering data management: achieving integration through database technology," Computing & Control Engineering Journal, pp. 119-126, June 1993.
- [5.11] Gardarin G., and Gelenble E., "New applications of database," Academic press, Inc., ISBN 0-12-27550-2, 1984.
- [5.12] Mohammed B., and Pual P., E., "Methodology for design of database for engineering applications," Computer-Aided Design Journal, Vol. 18, No. 5, pp. 257-262, June 1986.
- [5.13] Barney W., et al., "MicroStation database book.", Tools for linking ORACLE, informix, and Dbase to MicroStation, Supports all platforms using Intergraph/MicroStation trough version 4.x, ISBN 0-934605-03-3, 1992.

- [5.14] Intergraph solution for the technical desktop, "MicroStation Database Guide," version 4, September 1993.
- [5.15] Azeddin M., S., and Donald W., M., "Auto-Link Information to database," InterUser, Journal of the Intergraph UK & Ireland Graphics User Group, pp. 14-16, April 1995.
- [5.16] Brown A., W., "Database support for software engineering," First edition, ISBN 1-85091-948-8, 1989.
- [5.17] George S., et al., "Query classification in object-oriented engineering design systems," Computer-Aided Design Journal, Vol. 26., No. 26, pp. 26-136, February 1994.
- [5.18] Christos F., and Stavros C., "Design of a signature file method that accounts for non-uniform occurrence and query frequencies," Computer Systems Research Institute, Proceeding of Very Large Data Bases, STOCKHOLM, pp. 165-170, August 21-23, 1985.
- [5.19] Sunil K. S., et al., "Using History information to process delayed database updates," Proceedings on Very Large Data Bases, Twelfth International Conference on Very Large Data Bases, Kyoto, Japan, pp. 71-78, August 25-28, 1986.

# 6

## Design of Graphics User Interface

### 6.1 INTRODUCTION

Because computer graphics is used in many important applications, it is a large, growing discipline according to Donald [6.1]. Computer graphics is changing rapidly because of advances in computer science and electronics technology. The purpose of graphics design is to accommodate and provide different methods of generating and manipulating output display. Software modelling routines, provides a means for defining and organising model operations in terms of components symbols hierarchies, in which they are processed by the graphics routines for displaying.

Graphical user interface (GUI) is of particular importance for computerised systems (i.e. CAD applications). This is because, its form not only completely determines the user's view of the product, but also has a major influence on the efficiency with which editing and reviewing the data functions and information may be performed. The interface used to handle all code related data input to the system.

This chapter deals with the design of GUI and customisation of different tools that combine the different environments and their associated tools and applications. It also demonstrates the use of both languages (i.e. MDL and UCMs), where Graphical tools such as component placement, sub and main palettes, schematics modification tools are designed and developed.

## **6.2 REVIEW OF GRAPHICS SYSTEMS**

The intention of this section is to provide a discussion of the different graphics systems that have been developed. The main issue in highlighting these systems is the GUI that facilities them by providing different tools and utilities.

Steven [6.2], has reported that it is for a CAD system to provide a facilitated design environment. There must be a set of tools working in these environments to manipulate the circuitry. Tools such as synthesis and analysis must be designed in a way that they can be used with common user interfaces. Therefore, It is desirable for CAD systems to have a comfortable user interface.

Graphical user interfaces are usually evaluated and judged rather by their visual appearance than by their performance and efficiency in providing new functionality for designing new products. Stefan [6.3] described graphic windows system that has two layers, the bottom layer is a subroutine library designed to simplify the development of X windows applications, where the top layer uses viennese integrated system for technology CAD applications (VISTA) user interface library. It provides higher level operations in developing sophisticated applications.

Despite claimed standards for windows systems. Software seems to evolve as workstation technology advances. This suggests that different windows systems such as (Motif, Windows NT) have been featured by digital equipment workstations in the year between (1988-1994). Multiple tools integration is highly considered in developing a unified user interface, mostly based an X-Windows design environment such as philips research integrated device environment (PRIDE) [6.4]. In addition, Simulation of Profiles from the Layout Integrated Process simulation environment with x-Windows (SIMPL-IPX) [6.5] system is directly based on Xlib, uses user editor which has facilities for running design simulators.

## **6.3 GRAPHICS DESIGN AND ENGINEERING CONSIDERATION**

The design of user interface is considered to be the primary engineering consideration. There are several areas important to the development of user interface in terms of software (i.e. Cost and schedule considerations) [6.6]. Explanation to these areas as follows:

- Cost and schedule consideration: Since that the cost of designing and developing a high quality user interface needs to be justified, several items have to be considered. Cost of designing the user interface usually consists of the cost of implementation and modifications. This, however, indicates decisions that have been taken in order to

estimate how much is required for development. Therefore, it is important to consider all the costs of the particular system under the design and implementation of user interfaces.

- Iterative refinement: This item concerns the gathering of information so that the number of iterations can be reduced. The overall implication of this operation is the software design must be made as flexible as possible.

#### **6.4 GRAPHICS AND TOOL CUSTOMISATION**

Customisation is a term used to describe the design of new graphical and applications tools. A large of activities of interaction objects can be parameterized to provide flexibility. Parameters can be specified as software code, which is read in at run time. By making it possible to adjust parameters without modifying source code, this technique however, allows the appearance of an interface to be customised in easier way.

As reported by Marina [6.7], design management is considered to be very important area for developing CAD/CAE tool environment. By definition, the idea of design management address as set of functions which are used to build, maintain, display and enforce relationships among the data among the design tools that involves in a specific design.

For the customisation and application development MicroStation, Intergraph offers microStation development environment (MDE). It includes all the development tools needed for successful programming with functional CAD product. MDE can be used to develop graphic utilities, customised commands, or sophisticated commercial application. The MDE workbench provides visual user interface environment. Tools available include:

- Dialog Box. Dynamically build and modify dialog boxes.
- Icon editor. To build and modify custom icon in both standard and user defined sizes.
- Resource source generator. To generate resource source from MDL applications and binary source.
- Access for MicroStation environment for testing. To test the application while interactively developing.

In addition, MicroStation offers a powerful way to increase the efficiency and flexibility of drawing tools by customising. Using previously described languages (i.e. MDL and



UCMs), commonly used commands can be merged to develop new customised tools. Tools customisation, contains four different graphics elements (i.e. palettes, sub-palettes, dialog control boxes, and pull down menus). For instance, designing a tool palette consists of two different levels:

- Those on the first top level and have no special name.
- Those on the second bottom level are called sub-palettes.

It is important to differentiate because the term palette is often used to refer to many sub-palettes of the main palette collectively such as lines sub-palettes. Tools customisation however, offers some advantages. These are as follows:

- It saves time finding commands, thus improving efficiency and flexibility.
- It allows the designer to take control of view attributes which determines whether parts of design drawings aids will be displayed.
- It controls the manner in which design files are displayed.

## **6.5 GRAPHICS MODELLING METHODS**

An important use of graphics is in the design and representation of different types of systems [6.7]. Engineering systems, such as electronic circuit schematics, are commonly put together using CAD/CAE methods. The representations of this system, however, are often constructed to simulate a system behaviour under different conditions. To be effective in various applications, a graphics design must process efficient methods for constructing and manipulating the graphical representations.

### **6.5.1 Graphic Model Representation**

Information describing a model is usually provided as a combination of geometric and non-geometric data. Geometric information includes co-ordinates position for locating the component parts, and putting together primitives and attributes functions to define the structure of parts, and data for constructing connections between parts, and software algorithms that describes the operating characteristics of method.

Chapter 3 has described that the information is required to construct and manipulate a specific model is usually stored in some sort of data structure (e.g. data table contains different design attributes and parameters). Database, is one way of storing model data

description. In general, a model specification would contain both data structure about component parts and procedures of how they are related. For instance, an application to perform solid modelling of objects might use information taken from some data structure to define co-ordinate positions.

### **6.5.2 Symbol Hierarchies**

Because of the complexity of schematic and wiring, and because of the need to conserve space, graphics symbols are used to denote the various electronic, electrical and mechanical components. Circuit symbols are composite and formed from a number of graphics elements that are available within the CAD engine of MicroStation (refer to Section 6.8).

Different library cells are used to organise circuit models in a hierarchy structure. These symbols are used to form composite objects (i.e. referred to as modules), which themselves are grouped to form higher level of modules for various components of model. They can be presented as two level hierarchical in which defined by symbols size, position and orientation within each work area. For the facility of schematics packages with fixed sizes of objects or symbols, position and orientation are needed to be specified by user. Positions are given as co-ordinates locations, and orientation are specified as rotations that determine which direction is the component are facing.

## **6.6 GRAPHICS DESIGN AND DEVELOPMENT**

GUI becoming the key design components for most applications [6.7]. In designing a graphics interface, consideration such as graphics operations and interaction with other tools are to be made available to the user for convenient and efficient use.

This section outlines the methodology taken towards the design of different graphics tools. It is intended to demonstrate the integration between design different environments. In addition, figures of structural hierarchies are used to illustrate the physical structure of the graphics tools and their design.

### **6.6.1 Design Methodology**

One of the problems with designing and developing user interfaces that supports many tool and data extraction formats, is that it may become overwhelmed with a large number of tools. Previous attempts have focused on creating efficient and powerful programs rather than the development, so that they can be easier to integrate into the over all design cycle. James [6.8] presented a design methodology of very large scale integration (VLSI)

framework which has been defined as an object-oriented environment that treats a tool as objects. This approach, however, simplifies CAD tool control environment making the design framework more in general, easier to use and more capable of supporting a large variety of CAD tools.

GUIs are used to access and manipulate a system's data, and are often provided in place of special purpose programming language [6.9]. Normally, the programming of user interface consists of a library of subroutines that able to access and perform graphical transformations. However, The objective of providing these subroutines is to eliminate the need for application development to be concerned with data structure and eliminate the applications that are required to manipulate and analyse graphical structure. Figure 6.1 illustrates the use of both MDL and UCMs in designing the system's different tools and applications of different areas.

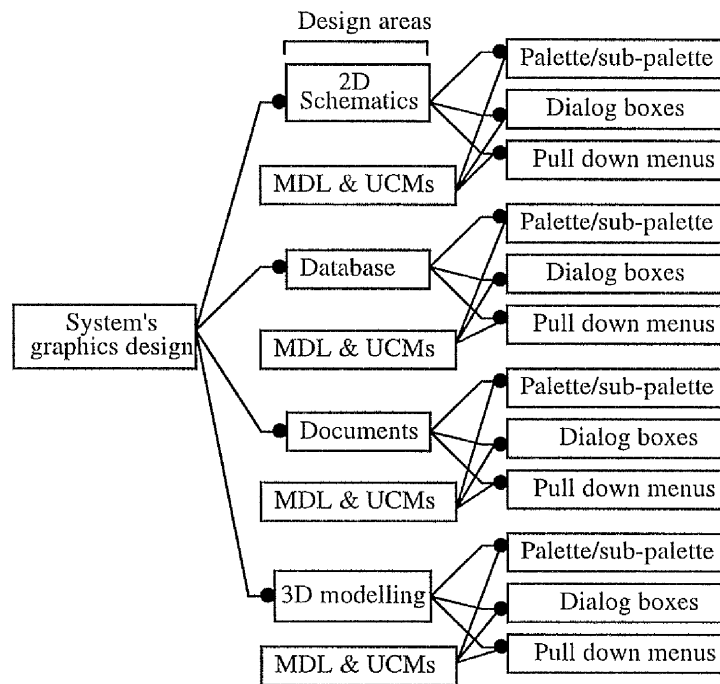


Figure 6.1 Illustration of graphics design general concept.

## 6.7 DESIGN OF USER INTERFACE

The benefits of designing system a user interface is that it provides prompts messages as a guide in processing the next design stages. To all CAD/CAM/CAE applications, these operations are considered to be the basic tasks that operate the same, whether the design is generating geometry for drafting, finite modelling, or generation of design documents. Jeng et al. [6.10], described that GUI used to support design production of circuits. This is because it can provide a way for other means of communication. The designed GUI

has the commands such as move, delete, load and save needed to trace and ensure that the circuits are connected. These steps all are in contact with a GUI. Therefore, results from GUI's feedback at each step can be monitored and decisions are taken towards new steps.

The UI also is the system layer through which the user and the system communicate [6.11]. It includes the keyboard, display, buttons, switches, labels, and any hardware, software that is part of the human/computer interaction. Developing a good UI however, exists in the traditional development strategies and the structures of many organisations. A common industrial development process of a UI adapted from [6.12] is illustrated in Figure 6.2.

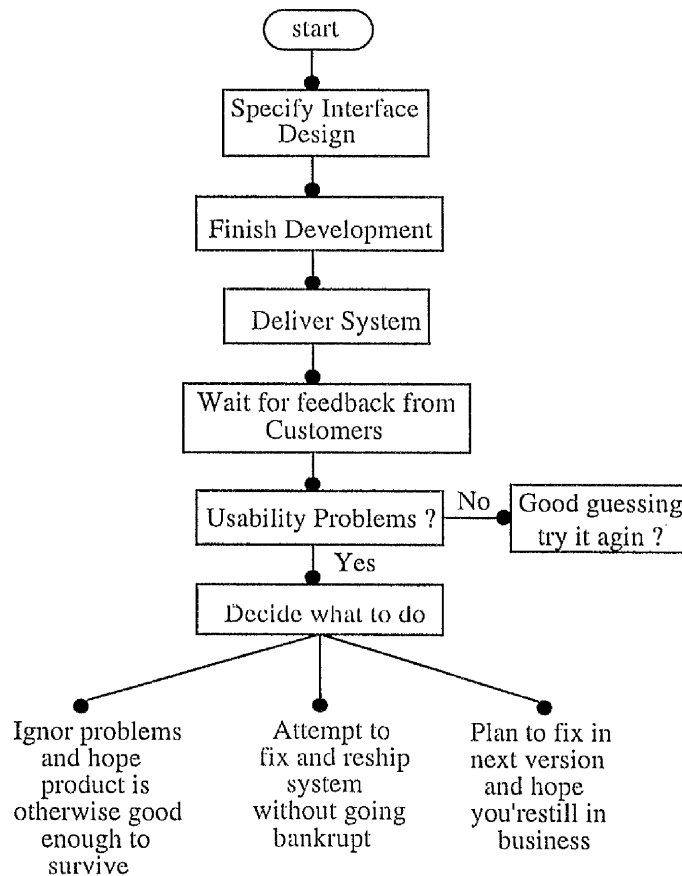


Figure 6.2 Common Industrial Development Process Flowchart [6.12].

### 6.7.1 Review of User Interface Generators

The actual construction and execution of the user interface is performed by tools that fit within the system framework [6.13]. Windows systems and user interfaces tool kits, are the most suitable tools for constructing graphical user interfaces. They are based on sequential, object oriented programming as it has been described as a suitable language in serving user interface metaphor. Most toolkits, interfaces builders, and frameworks are

all object oriented based. There are three reasons for the connection between users interface and objects:

- Users think in terms of objects as the elements of user interface.
- Objects provide a good abstraction mechanisms and operations.
- The ability of different objects to provide distinct implications for common operations. Whether for drawing, or other relating objects.

GUI for ECAD systems has followed a basic guidance as reported by Ouslis [6.14]. In order to perform a high level design functions, given simple commands and relevant information. A number of these commands typically constitute the complete package. Because it provides a simple user friendly interface and allows a great deal of knowledge to embedded into the program. Most CAD/CAE systems are structured this way. Code automatic generation tools are widely used to facilitate and assist in providing executable code. There are many automated tools that help to build design applications including user interfaces, which constitute a substantial bulk of the application that have been developed. Some of these generators are described below.

By definition a user interface generator produces a user interface from a specification. Bass et al. [6.6], described the basic components of generating a user interface as a library of interactive objects, and a specification tools. Figure 6.3 illustrates the different types of user interface generators namely for: Presentation of specification tools, It involves the retrieval of objects from defined set of classes, and attributes. Using these objects the desired image is built. Once this has been achieved, a code generated and edited to provide the correct action. The second type is dialog control specification tools: It provides the designer with special purpose language for specifying the sequencing rules for interactive objects. These rules defined as the conditions for creating, deleting and modifying of a particular graphical elements. The third is semantic specification tools: Design specification is expressed with a special purpose language; which describes the functional with regard to the user interface and it includes the definition of data structure and functions.

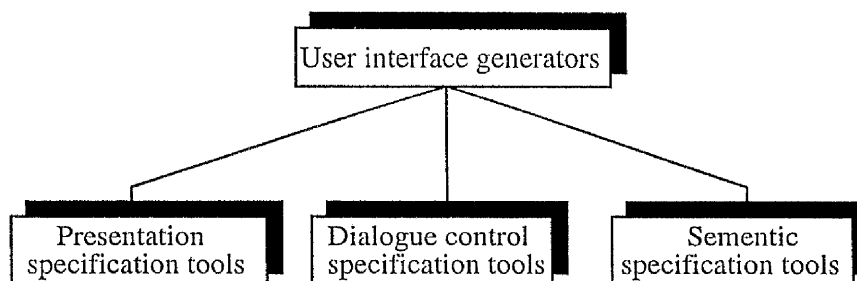


Figure 6.3 Different types of user interface generators [6.6].

User interface design tools are used as a means of interactively layout and edit user interface [nook]. Several provides simulation facilities for emulating the user machine interaction. Once the layout is completed, user interface tools can generate software code implementing the interface. User interfaces generators can be classified into two main types. These are as follows:

- The first, which is common in form layout tools, is a series of user interfaces definition files. They are linked with the application program code as part of the final executable program.
- The second, which is common among tools produces windows-based and graphical interfaces. Code generated by these tools contains a subroutines with which the application programmers interfaces.

The windowing system provides the basic tools needed to create user interfaces windows display environment, displays graphics, and respond to the user mouse selection. Generators such as Microsoft windows is a user interface environment developed for IBM PC systems. The windows itself is a subroutine library. The system contains several components such pull down menus and dialog boxes. Using this kind of environment applications can achieve the standard of all windows-based applications.

Hardwick et al. [6.15] developed a way of designing a user interface management system (UIMS). It is a tool that allows the creation and management of user interface for application programs. Using this method, however, makes it easy to construct and modify. Because many applications are designed and developed for a variety of tasks by variety of users. The paper has described a user development environment (CHIDE). It is implemented as an application of object-oriented database system (ROSE) for interactive engineering applications. This program allows to define user interfaces in a high level way. It is different from most other user interface development environments because it is has the ability to create and develop user interface without writing any application code.

### **6.7.2 User Interface Design levels**

The design of a user interface consists of different design levels as illustrated in Figure 6.4. It is mouse driven selection based on windows applications. Every design level contains graphical functions of push buttons, pull-down menus, and palettes icons designed to assist in performing different design tasks. The purpose of designing each level is to give a clear sight into how tools interact via control or data communication Figure 6.5 shows 2D schematics user interface.

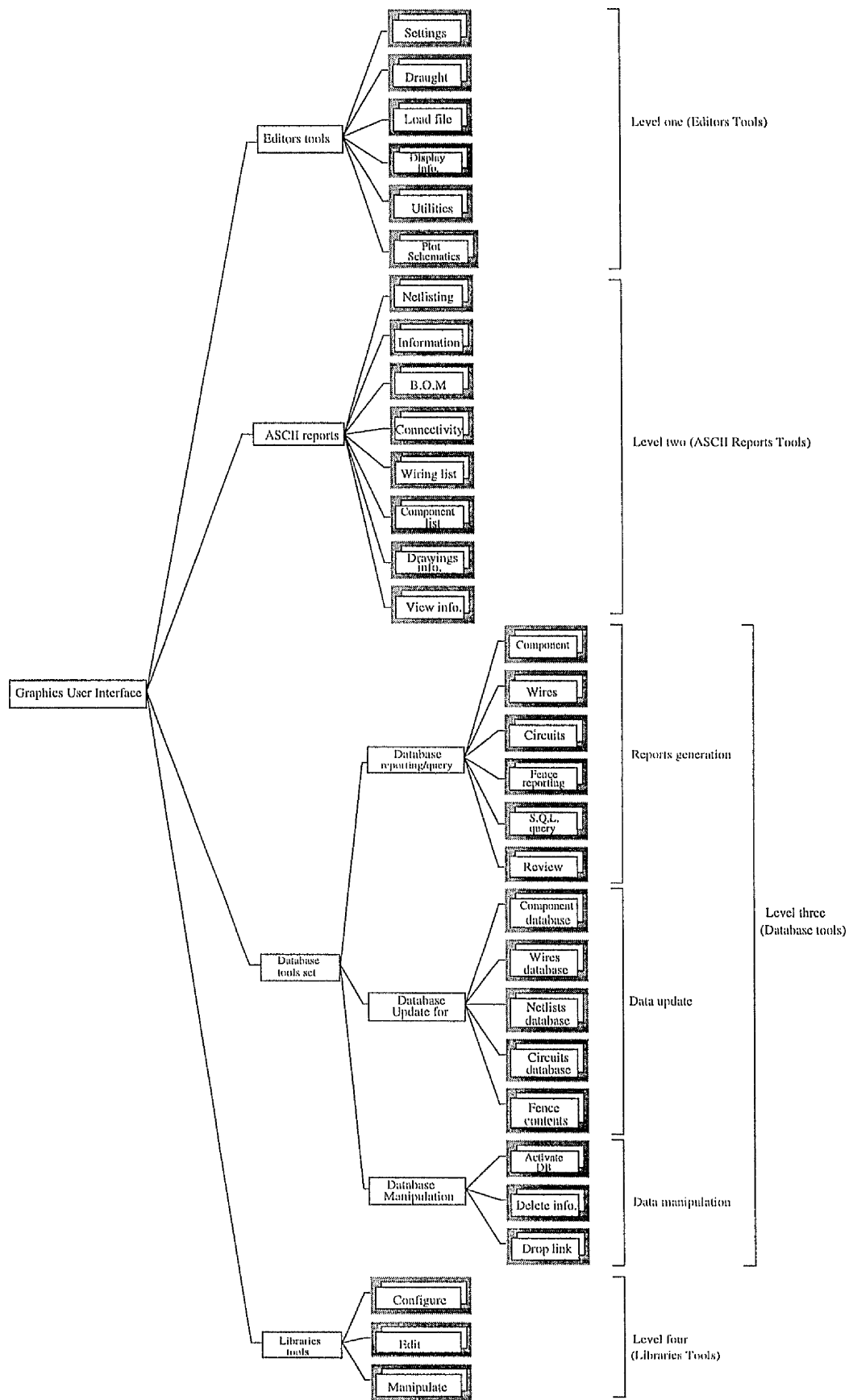


Figure 6.4 Illustration of GUI design levels.



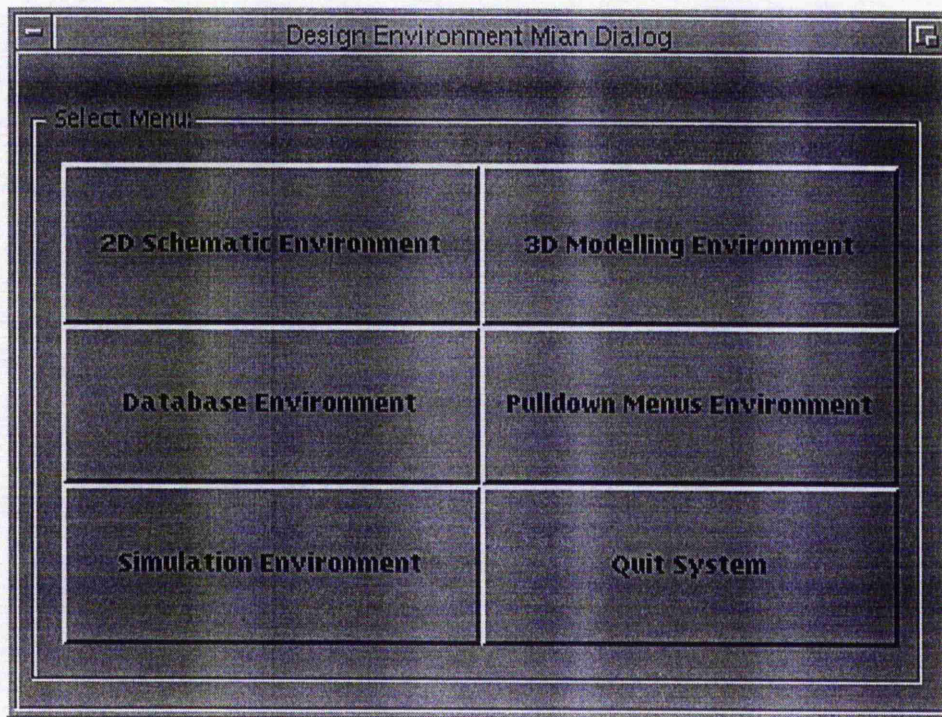


Figure 6.5 User interface main dialog.

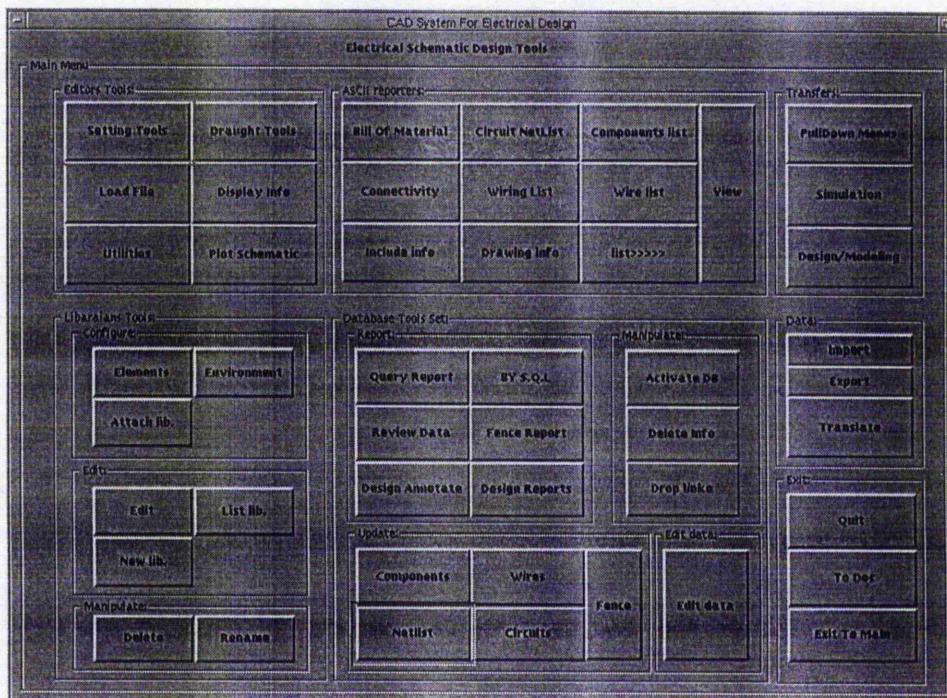


Figure 6.6 2D schematic interface main dialog.



## **6.8 ELECTRICAL SCHEMATIC DESIGN TOOLS**

The design of electrical schematic tools consists of seven different groups, namely: Editors tools, ASCII reports, Libraries tools, Database tool set, Data tools, and Exit commands. Their general role is to perform the following tasks:

- To assist in the integration between design and different environments.
- To assist in electrical/electronics circuits schematics layouts.
- Provide communications by means of design documentation.
- Provide communications with 3D models.

Detailed description of these tools is provided in the following subsections.

### **6.8.1 Schematics Workspace Design**

The operating workspace is one of the most important aspects in designing a CAD package [6.16]. They are classified into two main categories, viz:

- PC systems that runs under DOS and the other that runs under UNIX workstation (i.e. referred to as object-oriented). They are different in a way that the UNIX windows is ideal for modelling multiple sheets and tasks simultaneously,
- PC DOS packages rarely match the performance of multiwindows.

Setting up a design file requires some preparation that makes it convenient for designing a schematics diagram. Settings such as working units and precision from a seed file, sheet size can be done before starting a design session. Tools that are available while processing design drawing are determined by which workspace is active. Workspaces also contain a user interface, which, in some cases narrows the tools and menus for a particular application.

The 2D schematic environment is equipped with a multiwindows environment that makes convenient ways of displaying circuit schematics diagrams. Figure 6.7 shows the workspace designed for constructing 2D schematics drawings, whereas preliminary settings such as working units and other drawing parameters can be carried before or after the start of design session. Full details of schematics tools configuration are explained in Sub-section 6.8.2.

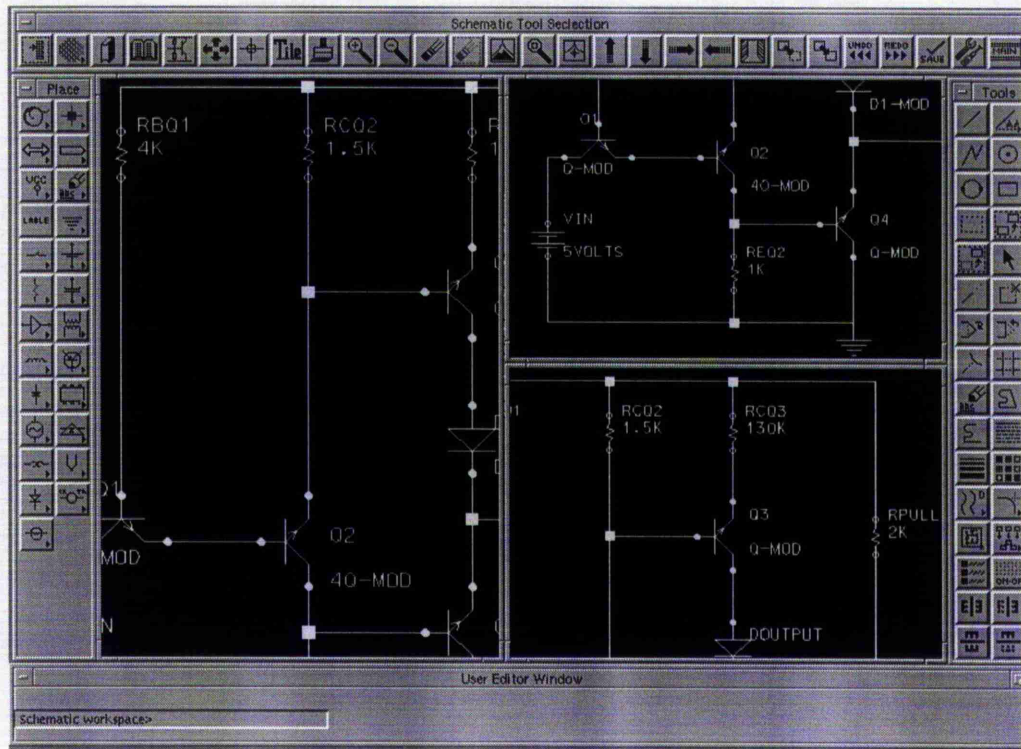


Figure 6.7 Display of schematics workspace environment.

### 6.8.2 Editors Tools Design

Editors group are used to create or modify design files. There are six different editors each of which carries its own task. They are designed to assist the settings of schematics tools consisting of (settings, draught tools, load schematics file, display information function, create new schematic file, and plotting schematic file). These editors are briefly described below:

- **Load file:** Used to load a design file by selecting load file button from the displayed dialog box.
- **Utilities:** They are functions allow design files to be created, copied, renamed, and deleted. They also have the ability to display design files which are been used.
- **Plotting of schematic drawings:** is a function designed to plot circuit schematic. It involves steps and configurations that need to be done before plotting. Steps of configuring a plotting file involve: specifying the area of the file, selecting the proper configuration file for the plotter, and generating a plot file for device transmission. MicroStation uses a default setting for sheets sizes, elements attributes, and variety of other variables. These, however, can be changed through configuration function buttons.

### **6.8.2.1 Design of Drafting Tools**

Traditionally, schematic diagrams are the master drawing and are used in preparation of wiring diagrams, mechanical layouts; part lists, and other associated drawing [6.17]. Therefore, tools should be developed to carry out this task. Creating schematic diagrams basically involves three main elements: Graphic symbols representing components; reference designations detailing components values, and interconnection between all components.

Design of schematic tools is considered to be the most intelligent in designing this system as they enable to synthesis and help to investigate simulation and analysis by generating the necessary data input. Schematic tools are designed to include different kinds of commands that can be issued to the user interface. Change commands such as move, rotate, Create and undo result in the change being set from the action taken by the designer. The process of laying out a schematic diagram is based on the engineering point view, where the following features are essential:

- Collecting data about components that are used in the schematic.
- Organising the general layout of the schematic based on circuit functions.
- Identifying components by reference designation and other data adding adequate notes on the schematic.
- Having the schematic conform in size and details to the drawings that will be produced for mechanical assemblies and details.

Considering the above mentioned features, Figure 6.8a and Figure 6.8b illustrate design hierarchy structure of schematic tools design plan. Editors tools contain push control buttons allow other design functions to be activated and displayed. Draught tools are invoked through push button allowing the schematic environment to be displayed. It consist of three different design palettes, namely: Main, top and modify. Detailed description to these groups is given in Section 6.9.

Schematics tools are used to construct circuits diagrams and to simulate them using PSPICE package. Any errors at this stage require the schematics to be altered and the simulation to be repeated. They have been designed to cover areas of ECAD integration. These include wiring and connectivity report generation and design verification.

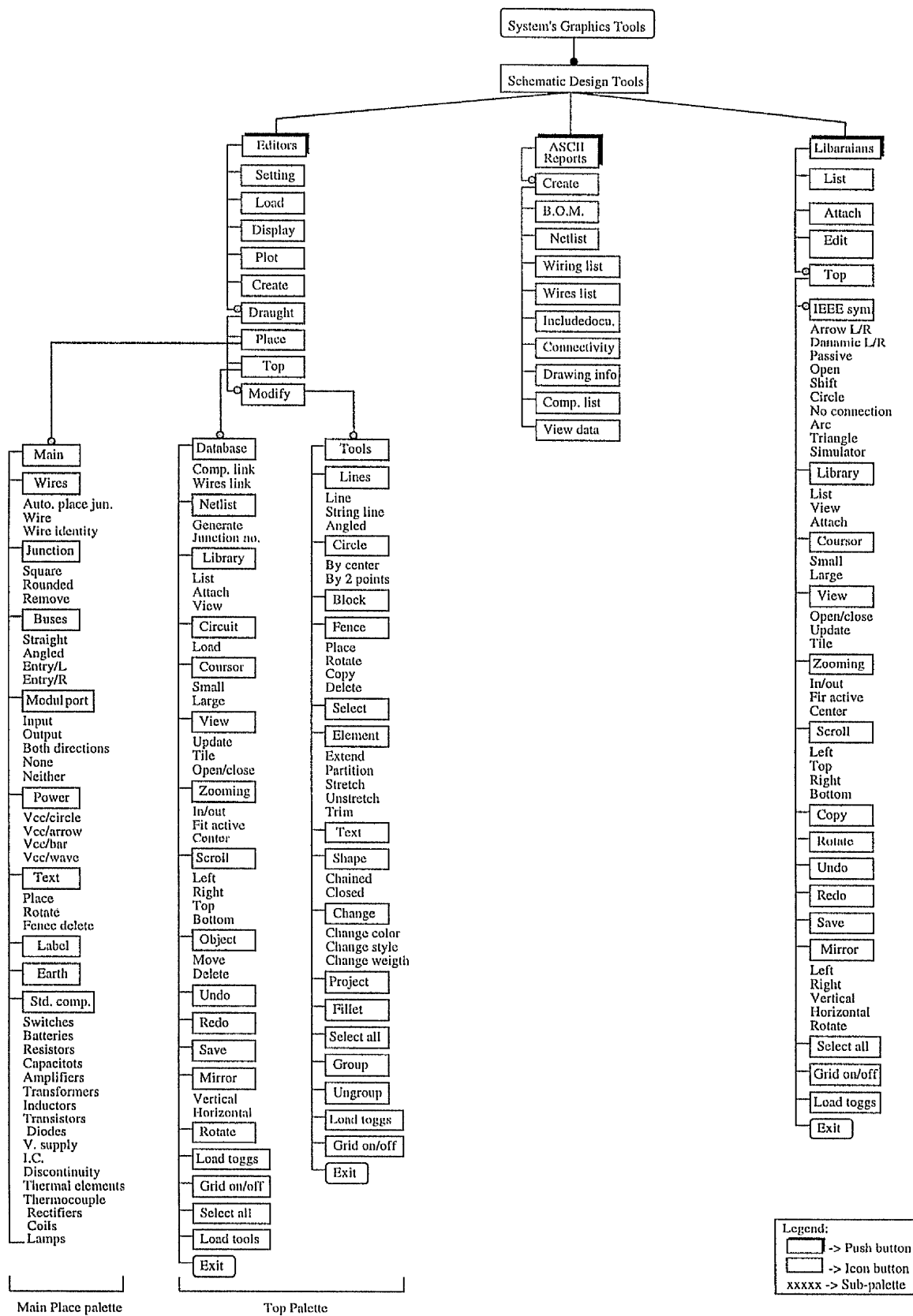


Figure 6.8a Schematic design tools hierarchy part (1).

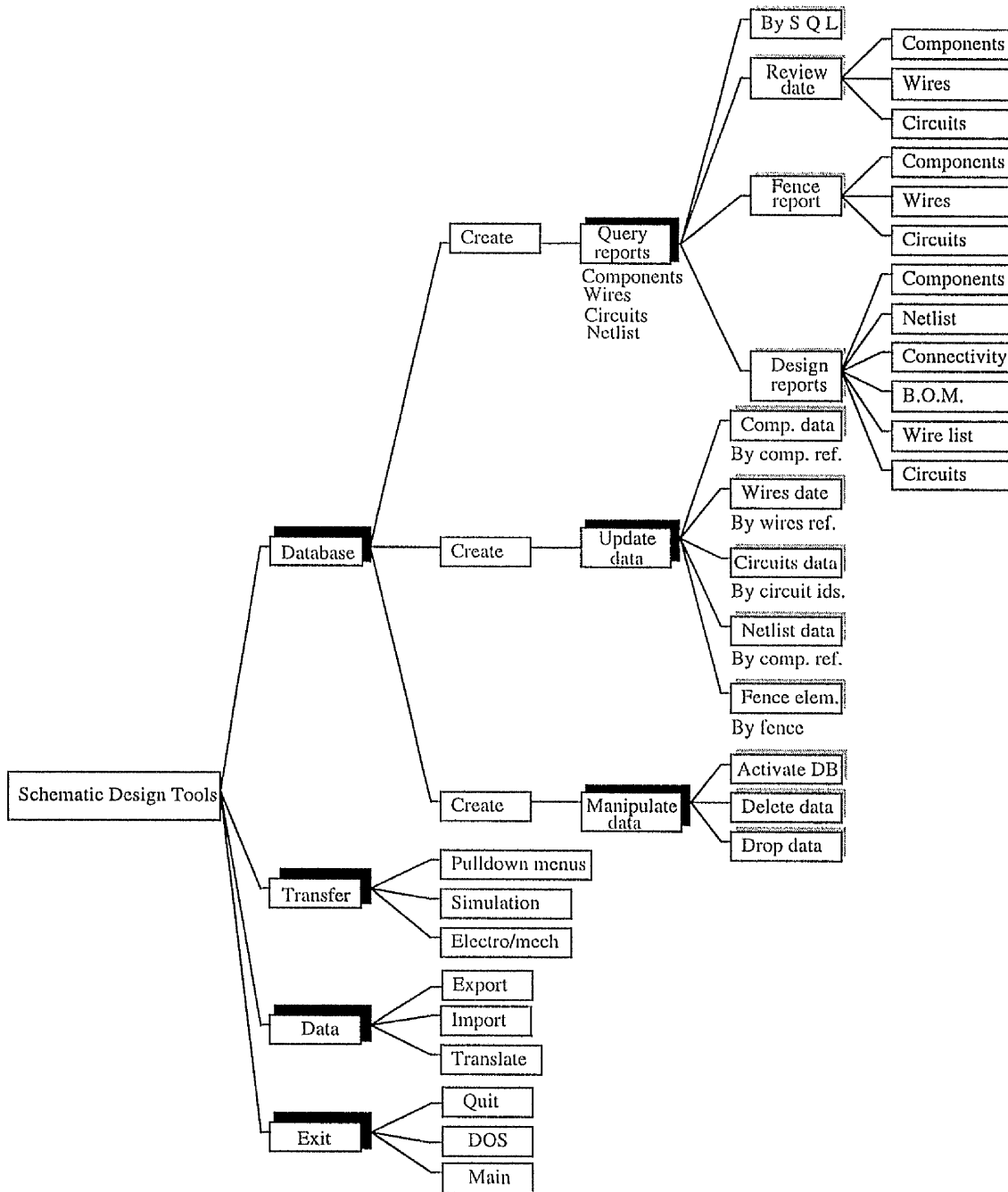


Figure 6.8b Schematic design tools hierarchy part (2).

### 6.8.2.2 Design of Schematics Palettes

Schematics palettes contain icons that represent components symbols, drawing tools which they issue various commands. There are three, different main palettes each one performs a different task. The first palette is the place palette which appears in the left side of the workspace, detailed explanation is outlined below.



### Place Main Palette

It contains various tools to perform: Wires, junction and schematics symbols placement and other objects as shown in Figure 6.9. There are many sub-palettes appear once selecting an icon as shown in Figures (6.10-6.33). The second palette, is the tools palette which contains icons such as drawing lines, copy fence contents, modify, change elements attributes, toggles settings, mirrors tools, and so forth as shown in Figure 6.35. Figures (6.36-6.42) show the sub palettes that appear once selecting an icon. The third palette, is the schematics top palette which appears at the top of the working space, it contains tools such as edit database, netlist, wires automatic numbering, reviewing of libraries symbol., and so forth as shown in Figure 6.46. Figures (6.43, 4.45), respectively show netlist sub-palette and the view control of open view dialog.

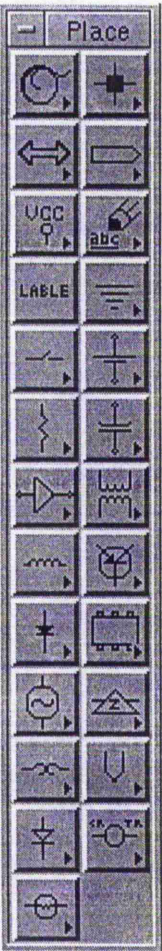
<i>Left column</i>		<i>Right column</i>
Wiring tools		Schematic junctions
Buses tools		Circuit module ports
Schematic power supply		Text adding&modifying
Place lable		Ground symbols
Switches different symbols		Batteries symbols
Resistors different symbols		Capacitors symbols
Amplifier different symbols		Transformer different symbols
Inductors symbols		Transistor different symbols
Diodes symbols		IC retrieve symbols
Voltage supply symbols		Discontinuity symbols
Thermal elements symb.		Thermocouple symbols
Rectifier symbols		Colis symbols
Lamps symbols		

Figure 6.9 Place main palette.

### Symbols sub-palettes

The following are components symbols sub-palettes used to construct 2D schematic drawings. Each individual icon is linked to UCM program for default settings and text

fields placement (refer to program flowchart shown in figure 6.47). They are activated and dragged by simply placing the cursor over a desired symbol.

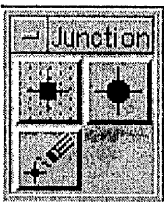
<i>Left column</i>		<i>Right column</i>
Square shaped junction		Rounded shaped junction
Automatic remove junction		

Figure 6.10 Schematic junction sub-palette.

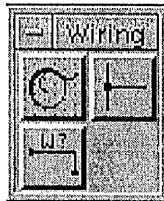
<i>Left column</i>		<i>Right column</i>
Place wires		Place junction through selected wire
Automatic place wires identity		

Figure 6.11 Place wires sub-palette.

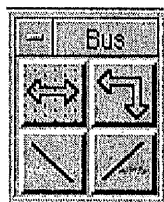
<i>Left column</i>		<i>Right column</i>
One direction bu		Two direction bus
Left hand side bus		Right hand side bus

Figure 6.12 Bus symbols sub-palette.

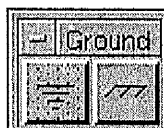
<i>Left column</i>		<i>Right column</i>
Ground symbol		Classic symbol

Figure 6.13 Ground symbols sub-palette.

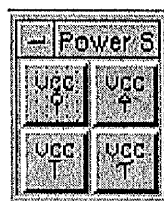
<i>Left column</i>		<i>Right column</i>
Circle power symbol		Arrow power symbol
Bar power symbol		Wave power symbol

Figure 6.14 Power supply symbols sub-palette.

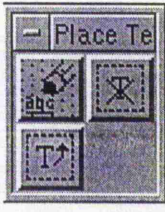
<i>Left column</i>		<i>Right column</i>
Place text		Fence delete text
Fence rotate text		

Figure 6.15 Schematic place text sub-palette.

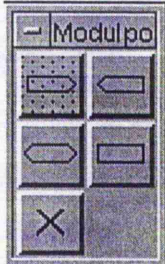
<i>Left column</i>		<i>Right column</i>
Input signal port		Output signal port
Bidirectional port		Unspecified port
No-connection		

Figure 6.16 Module ports symbols sub-palette.

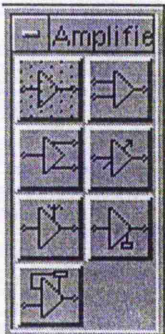
<i>Left column</i>		<i>Right column</i>
General amplifier type		Two input pins
Two output pins		Adjustable gain
Associated attenuator		Associated power supply
External feedback path		

Figure 6.17 Amplifiers symbols sub-palette.

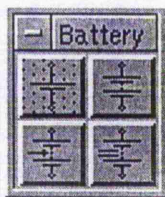
<i>Left column</i>		<i>Right column</i>
One cell		Multicell
Multicell with adjustable tap		Multicell with taps

Figure 6.18 Batteries symbols sub-palette.

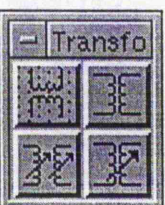
<i>Left column</i>		<i>Right column</i>
General transformer type		One winding with adjustable inductance
Each winding with adjustable inductance		Adjustable mutual inductor

Figure 6.19 Transformers symbols sub-palette.



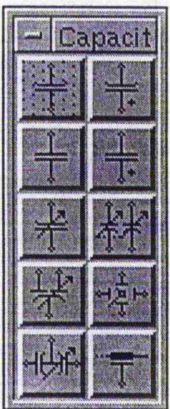
<i>Left column</i>		<i>Right column</i>
General type		Polarized
IEC general type		IEC polarized type
Adjustable or variable		Adujustable or variable with mechanical linkage
Continuously adjustable or variable differential		Phase shifter
Split stator		Feedthrough

Figure 6.20 Capacitors symbols sub-palette.

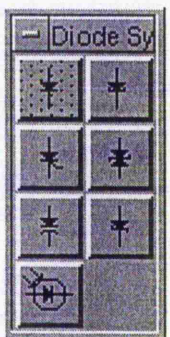
<i>Left column</i>		<i>Right column</i>
Semiconductor diode		Tunnel diode
Silicon controlled rectifier		Breakdown diode
Capacitive diode		Zener diode
Photodiode (sollar cell)		

Figure 6.21 Diodes symbols sub-palette.

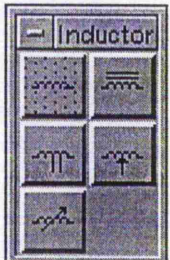
<i>Left column</i>		<i>Right column</i>
General type		Magnetic-core inductor
Tapped inductor		Adujstable inductor
Continuosly adjustable inductor		

Figure 6.22 Inductors symbols sub-palette.

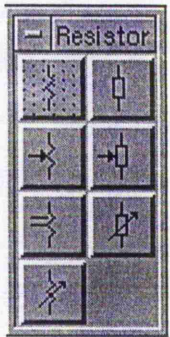
<i>Left column</i>		<i>Right column</i>
Genaral resistor		General
Tapped with adjustable contact		Tapped with adjustable contact
Tapped resistor type		Adjustable or continuosly adjustable
Adjustable or continuosly adjustable		

Figure 6.23 Resistors symbols sub-palette.

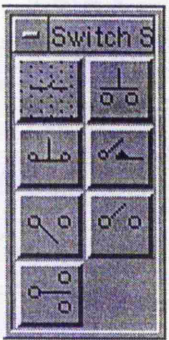
<i>Left column</i>		<i>Right column</i>
Singl-throw type		Push button, circuit closing
Push button, circuit opening		Momentary or spring return circuit closing
Locking-circuit closing		Locking-circuit opening
Locking-transfer, three-position		

Figure 6.24 Switches symbols sub-palette.

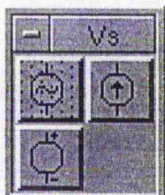
<i>Left column</i>		<i>Right column</i>
Voltage supply symbol (1)		Voltage supply symbol (3)
Voltage supply symbol (3)		

Figure 6.25 Voltage supply symbol sub-palette.

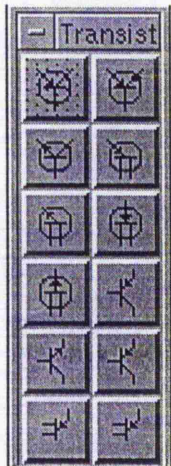
<i>Left column</i>		<i>Right column</i>
PNP transistor		PNP transistor with one electrode connected to
NPN transistor		Unijunction N-type base
Unijunction transistor (P-type base)		Field-effect transistor N-type base
Field-effect transistor p-type base		PNP transistor
NPN transistor		
Unijunction trasistor		

Figure 6.26 Transistors symbols sub-palette.

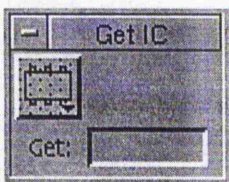
<i>Left column</i>		<i>Right column</i>
Get ICs modules		
		Text editor for active IC cells

Figure 6.27 IC symbols sub-palette.

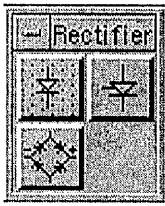
<i>Left column</i>		<i>Right column</i>
General symbol		Controlled type rectifier
Bridge type symbol		

Figure 6.28 Rectifier symbols sub-palette.

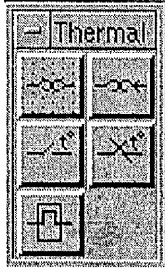
<i>Left column</i>		<i>Right column</i>
Actuating device		Thermal cutout
Thermostat (closing on rising temp.)		Thermostat with contact motion clarified
Thermal relay with normally open contact		

Figure 6.29 Thermal elements symbols sub-palette.

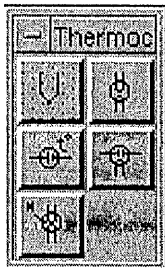
<i>Left column</i>		<i>Right column</i>
General thermocouple symbol		With integral insulated heater
Semiconductor thermo. couple, temp. measuring		Semiconductor thermo. couple, current measuring
With integral heater internally connected		

Figure 6.30 Thermocouple symbols sub-palette.

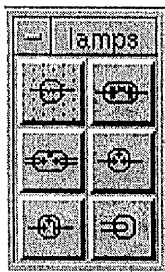
<i>Left column</i>		<i>Right column</i>
Ballast tube		Fluorescent lamp, with two-terminal
Fluorescent lamp, with four-terminal		Cold-cathode glow lamp, a-c type
Cold-cathode glow lamp, d-c type		Incandescent lamp

Figure 6.31 Lamps and visual devices symbols sub-palette.



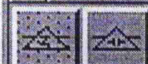



Left column	Disconti	Right column
Equivalent series element		Capacitive reactance
Inductive reactance		Inductance-capacitance circuit, infinite reactance
Equivalent shunt elem.		Capacitive susceptance
Inductive susceptance		

Figure 6.32 Discontinuity symbols sub-palette.

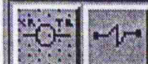
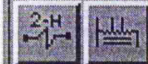
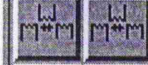
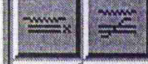

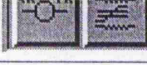
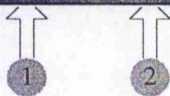
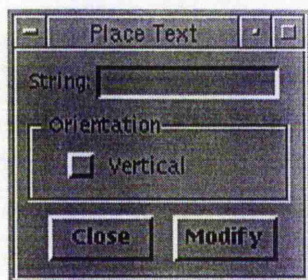
Left column	Coils	Right column
Relay timer		General solenoids
2-position hydraulic		Auto transformer
Linear variable differential transformer		
Coils (reactor-iron core)		Coils (reactor-saturable core)
Coils (reactor-air core)		Reactor (magnetic amplifier winding)
		Coils (staurable transformer)

Figure 6.33 Coils symbols sub-palette.

### Place Text Dialog

It allows the text element to be placed into schematic design file. It is activated once selecting the text icon from the place main palette shown above. The dialog is link through a UCM program allowing text fields to automatically be edited and attributes default settings. It also contains a function allows text fields to be modified.



Item	Program type	Purpose
Text editor	MDL	Enter text for schematics elements.
Toggle button	MDL	Rotate text vertical/horizontal.
Push button (1)	MDL	Modify data fields.
Push button (2)	MDL	Close edit text dialog.

Figure 6.34 Place text dialog.

### Modify Element Main Palette

Figure 6.35 shows the modify main palette that contains different tools designed specially to facilitate 2D schematic diagrams. Other icons are used to load dialog boxes through the selection of the corresponding icons.

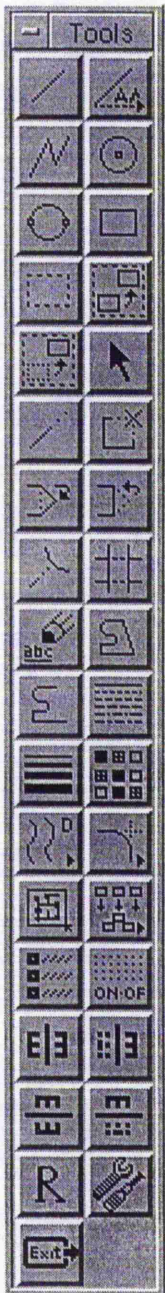
<i>Left column</i>		<i>Right column</i>
Draw striaght line type (3)		Draw angled line
Draw string line type (4)		Draw centered circle
Draw a circle with 2-Ps.		Draw rectangular shape
Place a fence		Rotate within fence
Copy within fence		Element selection
Extend a wire		Modify element
Undo expand wire		Expand a wire
Modify wire with arc		Remove additional ends
Add text		Create complex shape
Create chain		Modify drawing style
Modify drawing weight		Modify drawing color
Project elements		Create an arc
Automatic select all		Create groups
Load toggles dialogue		Grid switch On/Off
Create vertical mirror (1)		Create vertical mirror (2)
Create horizontal mirror (2)		Create horizontal mirror (1)
Rotate components		Load up set tools
Exit to main interface		

Figure 6.35 Modify elements main palette.

### Modify Element Sub-palettes

The following are sub-palettes dragged from the main modify palette shown above.

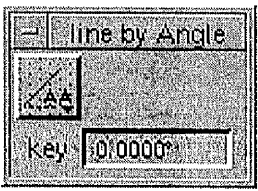
<i>Left column</i>		<i>Right column</i>
Line by angle		
		Key-in angle

Figure 6.36 Angled line sub-palette.

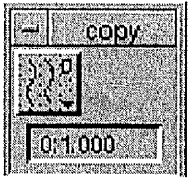
<i>Left column</i>		<i>Right column</i>
Copy line paralell		
		Key-in distance

Figure 6.37 Copy line parallel sub-palette.

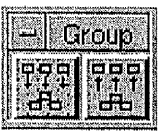
<i>Left column</i>		<i>Right column</i>
Group selected elements, wires, etc...		Ungroup selected elements, wires, etc..

Figure 6.38 Group-ungroup elements sub-palette.

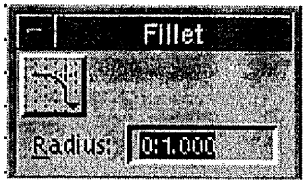
<i>Left column</i>		<i>Right column</i>
Create corner fillet		
		Key-in radius.

Figure 6.39 Place fillet sub-palette.

### Modify Element Dialog Boxes

The following are graphical dialog boxes designed and used to modify element attributes such colour, style, weight by simply selecting an element or using automatic element selection program .



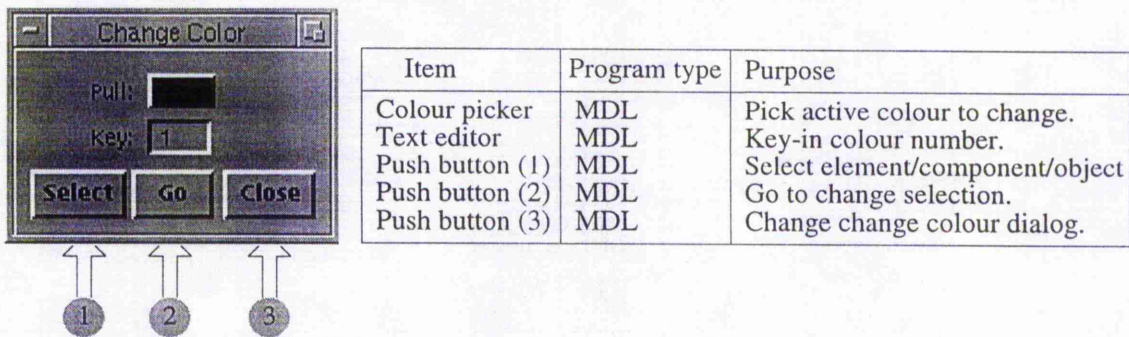


Figure 6.40 Change colour dialog.

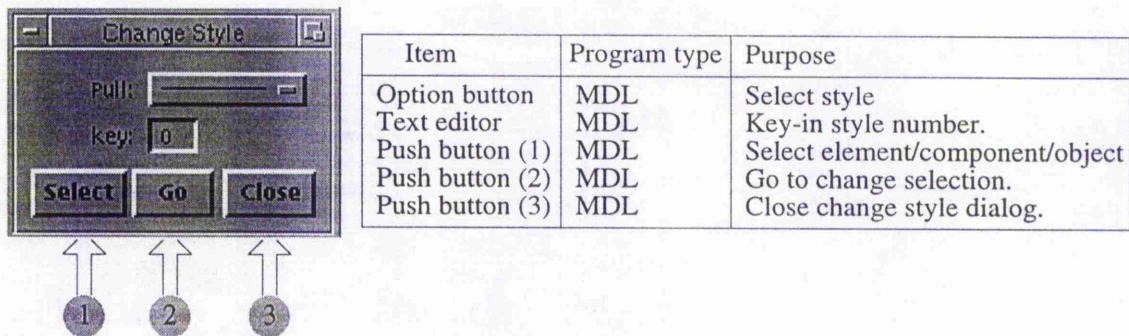


Figure 6.41 Change style dialog.

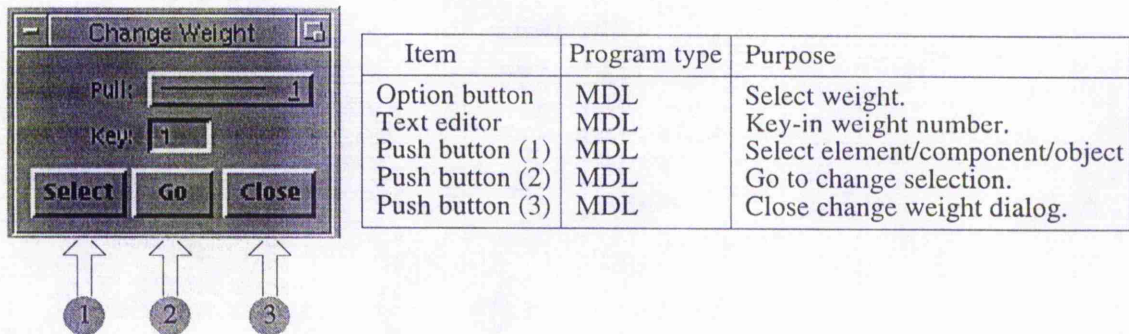


Figure 6.42 Change weight dialog.

## Top Main Palette

Top main palette, contains many other tools developed to assist in 2D, 3D constructions. Other tools such loading of component, wires, circuits dialog boxes, and netlist editing dialog which linked to database.

## The link with UCMs

Chapter 2 has described the use of UCMs and its role in developing new applications. They have the ability to develop custom design applications for specific design

requirements. UCMs also support features, which are found in any programming language. They can automate a common sequences of commands and simulate operator input, manipulate design files, access and modifying database files. Schematics design tools such as wiring and components placement need a software programs that can be used to assist in placing and wiring components in a desired locations.


<i>Left column</i>		<i>Right column</i>
Edit components data		Netlist and wires auto-numbering tools
Review libraries symbols		List design libraries
Load new design		Set large cursor
Set small cursor		Tidy windows
Auto-update views		Zoom in
Zoom out		Delete elements
Dealete fence contents		Fit active design
Zoom window		Zoom on center
Move drawing up		Move drawing down
Move drawing right		Move drawing left
Open windows views		Copy elements (1)
Copy elements (2)		Undo design changes
Redo changes		Save design session
Load tools main palette		Exit to main interface

Figure 6.43 Workspace top main palette.

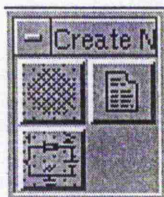
<i>Left column</i>		<i>Right column</i>
Extract netlist		Report on netlist
Junction/node automatic numbering		

Figure 6.44 Netlist-wire sub-palette.



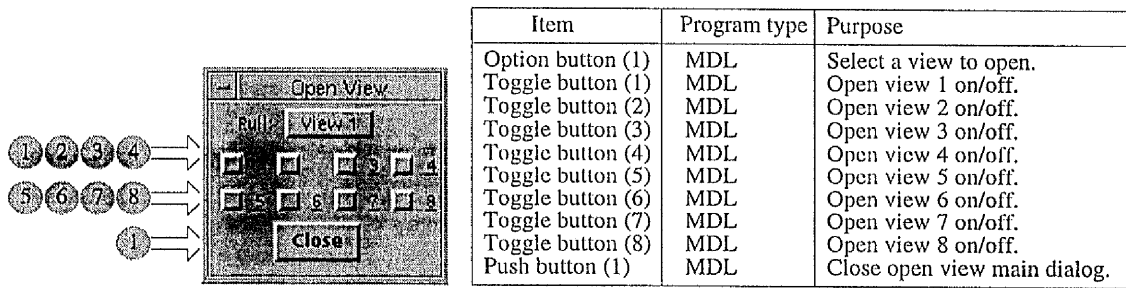


Figure 6.45 Open view main dialog.

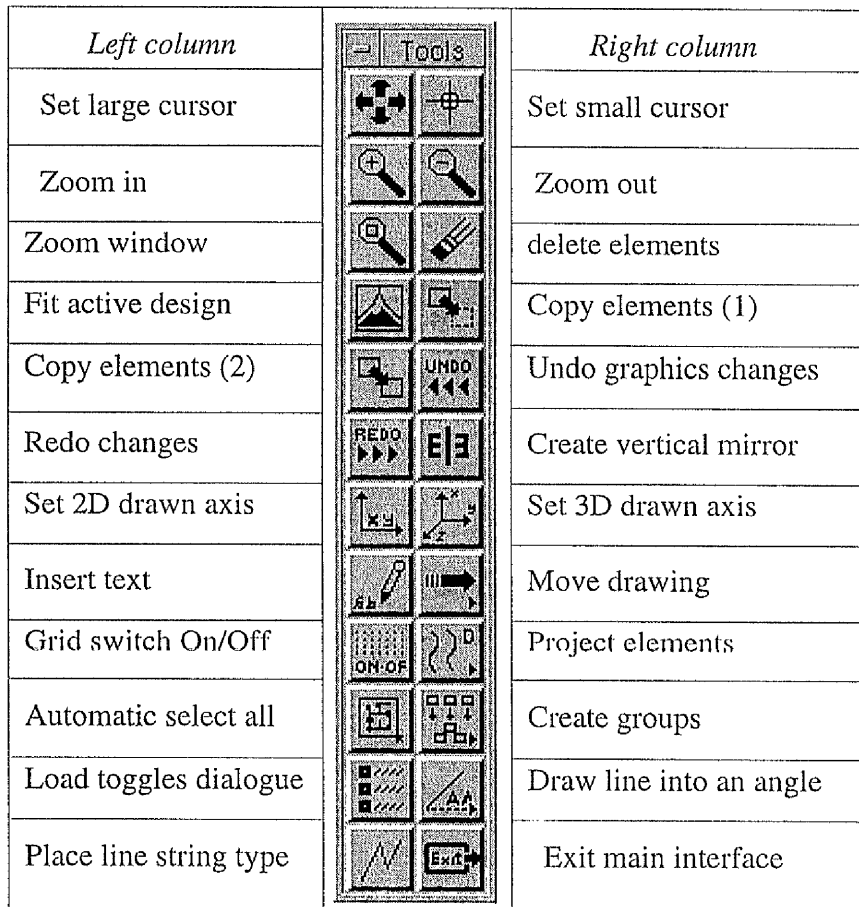


Figure 6.46 3D modelling main palette.

The idea of using UCMs in this case was by attaching these programs to the components symbols sub-palettes through the MDL command (MDL\_UCI), as illustrated in Figure 6.47. Figure 6.48 shows the UCM program flowchart developed for the purpose, where components placement and defaults are automatically set. Table [6.1] shows the software commands used for developing this program.

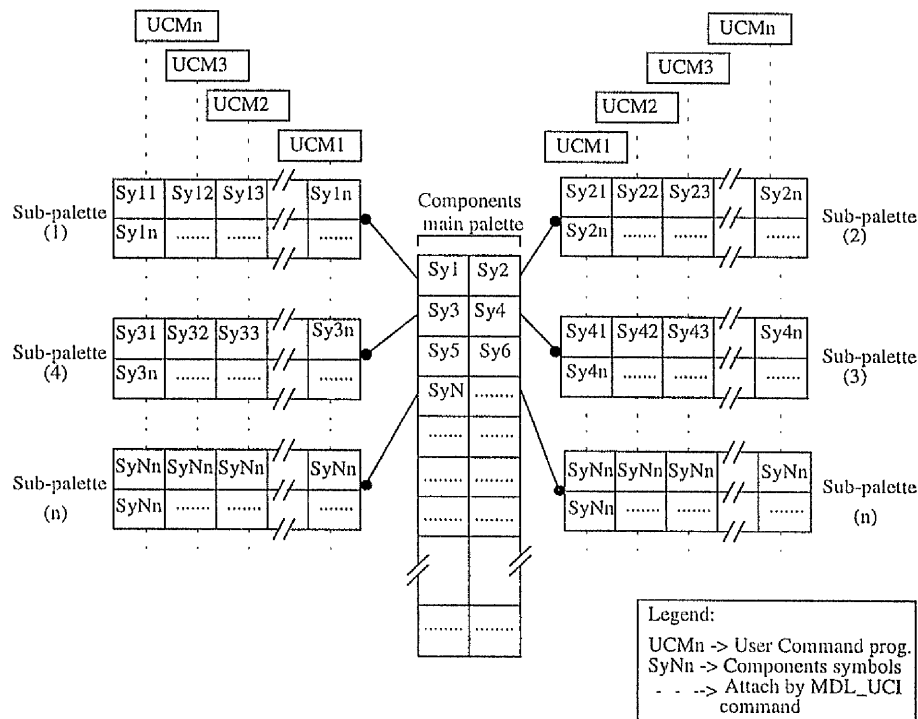


Figure 6.47 Components UCMs program attachment.

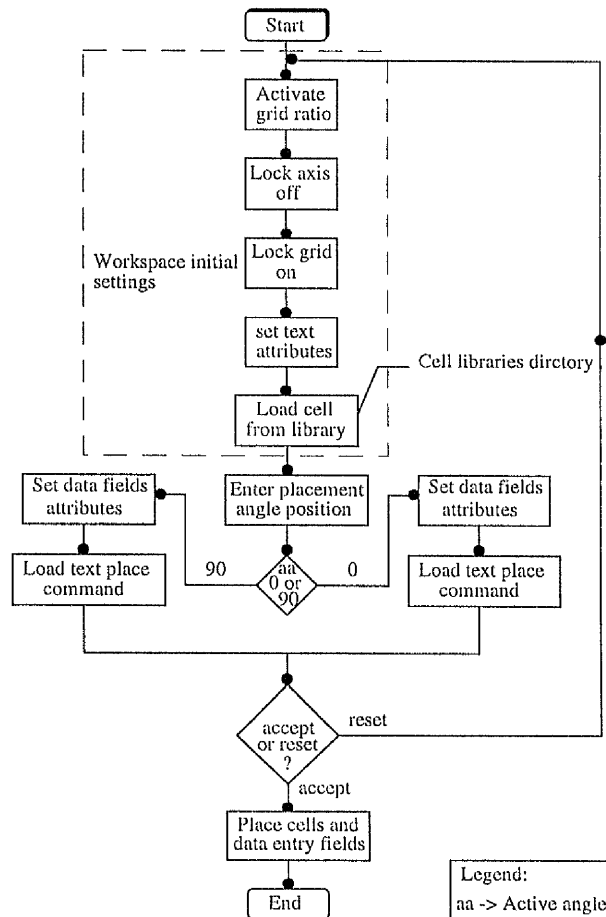


Figure 6.48 Flowchart of component placement program.

Table [6.1]. Workspace initial settings commands and their operations.

Utility	Command	Operation
Grid ratio settings	Activate grid ratio 1	Set up schematic grid ratio to 1:1 mm
Control axis locking	Lock axis off	Switch axis control off
Control grid locking	Lock grid on	Switch grid lock on
Control text editing fields	Set ed off	Switch text editing fields off
Text scale parameters	th=.6 & tw=.6	Set text fields scale for components ref. des. and values
Control views number	Selview all	Activate which view to open
Open components libraries	rc=d:\ustation\wsmod \ecad\cell\library.name	Direct opening library by sub-palette selection
Cells components scale settings	as=1	Set components scale with reference to grid ratio (i.e. 1:1mm)
Identify component rotation angle	aa=0 or aa=90	Rotate components to desired position
Place picked component	ac="cell's name"	Load to place components into schematics

### 6.8.2.3 Design of Configuration Tools

Configuration tools are programs which can be controlled to carry out specific changes to design environments and layouts. Changes to schematics elements and objects can be performed through the selection of configuration tools. Therefore, the configuration in this sense means a set of default functions which the user specifies. Some of the MicroStation settings are stored as part of the design file, and other are stored with the MicroStation and the workspace. For example, element active level and the level displays are stored with the design file; therefore, each design file has its own settings.

### Configuration Sheets

Using both described languages (i.e. MDL/UCMs), automated configuration tools were developed. Graphics modification such as levels control, weight, styles, colours, data fields etc. can be modified without selecting an element. The idea behind this was by the scanning of schematic drawing in searching for elements/objects that corresponds to the selected push buttons. Figures (6.49a, 6.49b, and 6.49c) show diagrams of different configuration tools, while Figures (6.50, 6.51, 6.52, 6.53, and 6.54) respectively, show the developed dialog boxes.

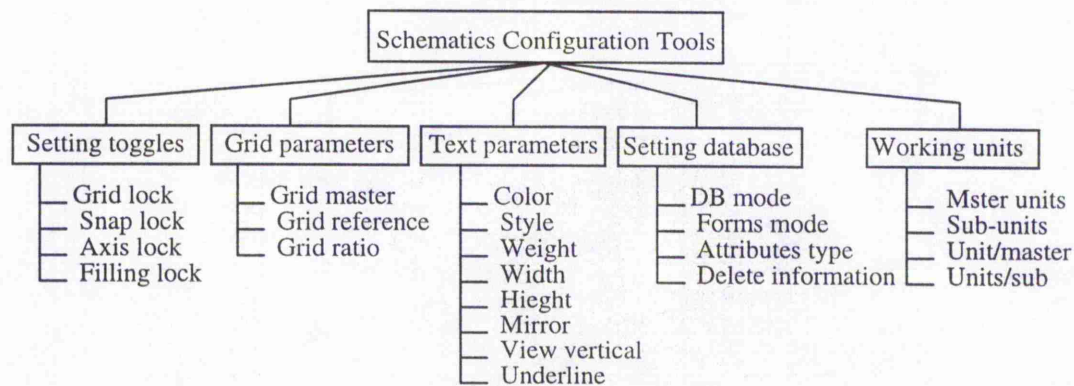


Figure 6.49a Configuration tools hierarchy part (1).

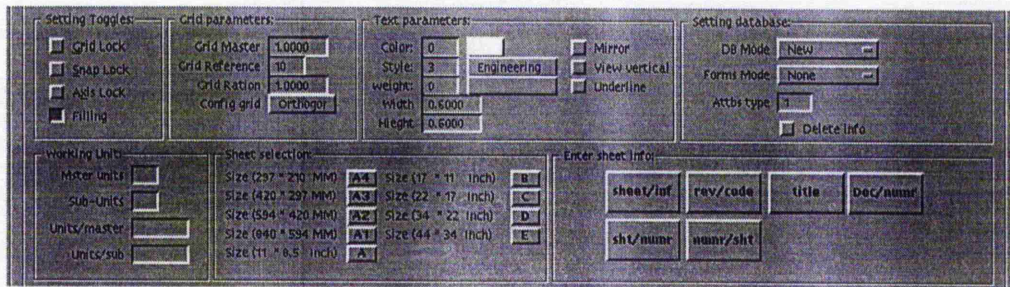


Figure 6.50 Setting tools dialog of sheet (1).

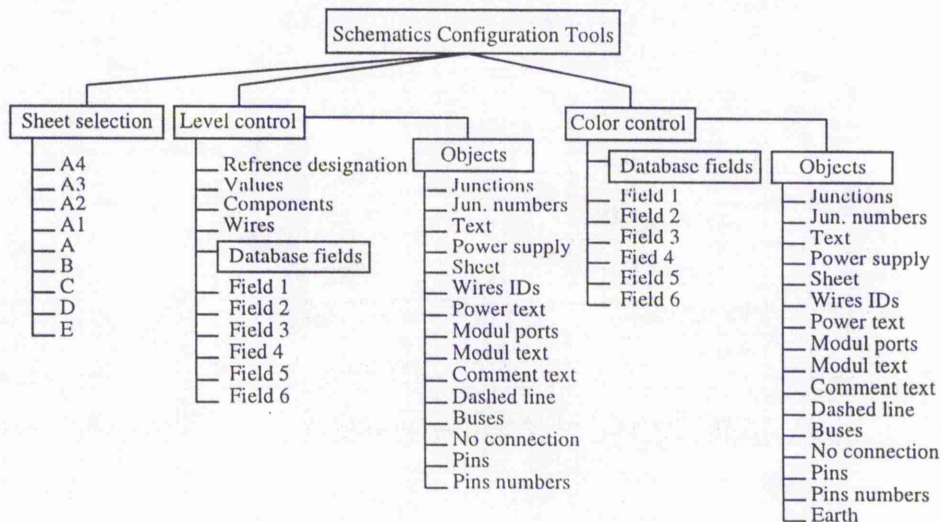


Figure 6.49b Configuration tools hierarchy part (2).

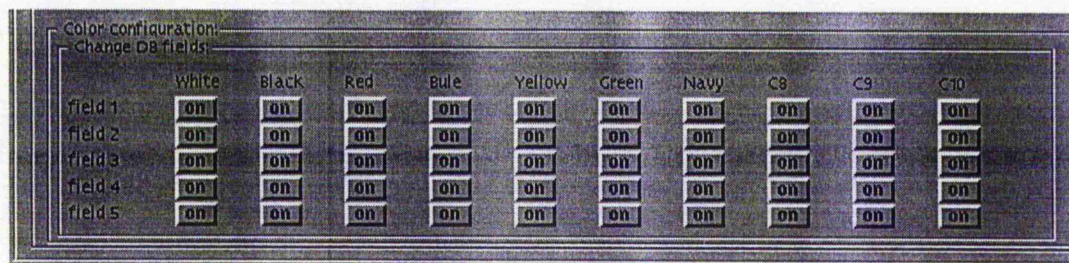


Figure 6.51 Automatic colours configuration dialog part (1).



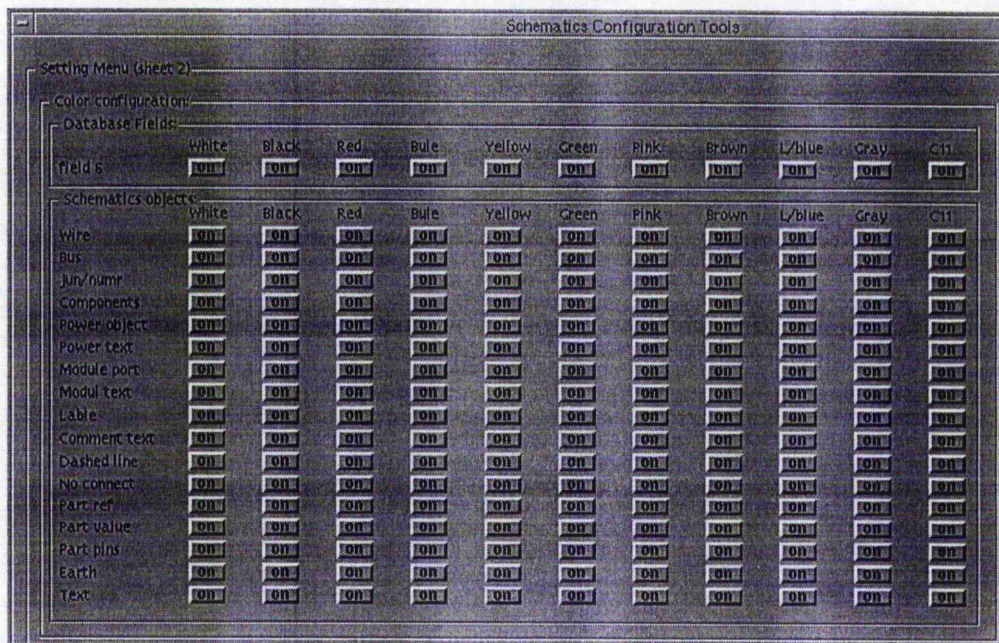


Figure 6.52 Automatic colours configuration dialog part (2).

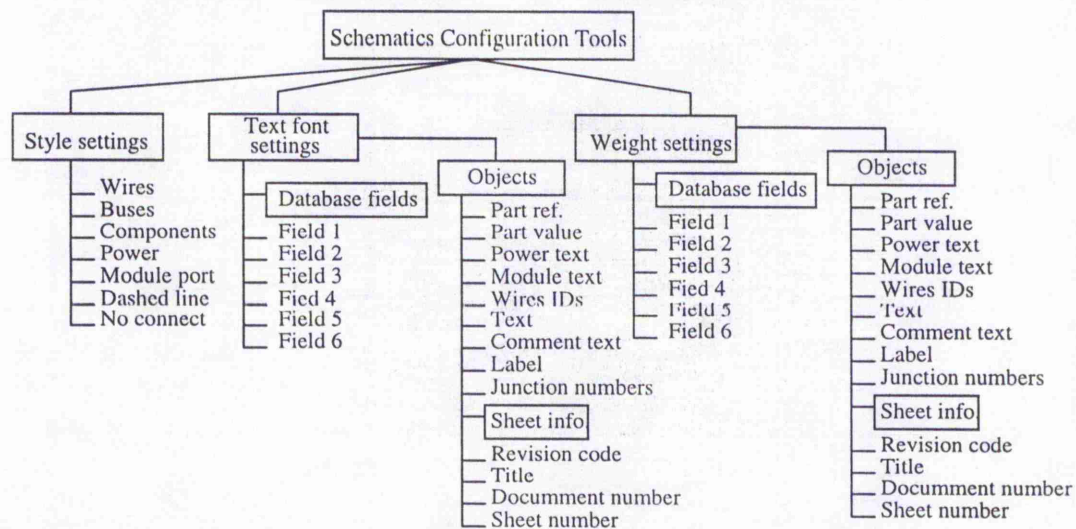


Figure 6.49c Configuration tools hierarchy part (3).

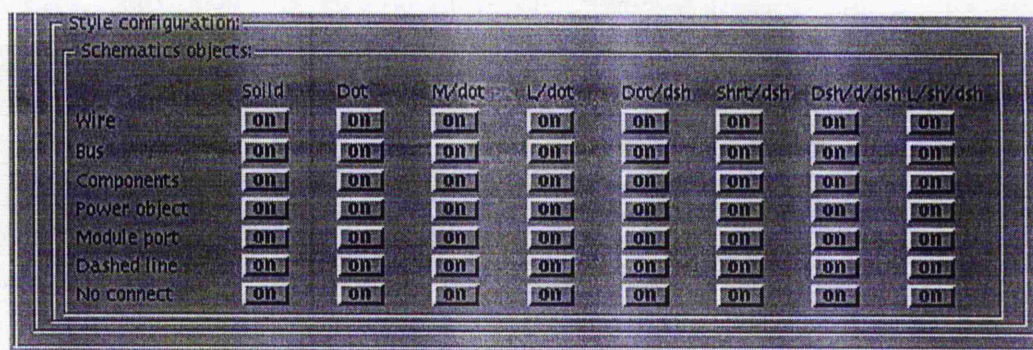


Figure 6.53 Automatic style configuration dialog.



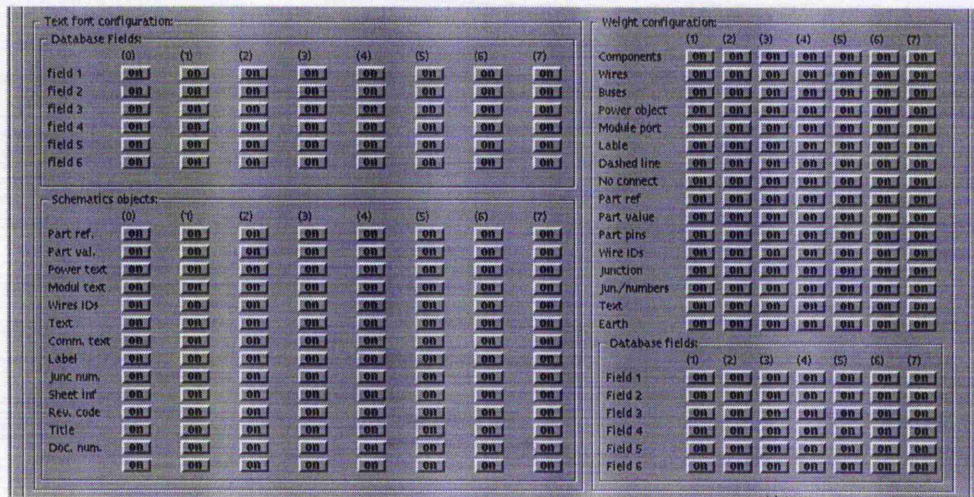


Figure 6.54 Automatic font-weight configuration dialog.

The figures which follow illustrate the kind of information contained in the configuration tools. Figure 6.55 is one of the first sheets encountered after selecting setting tools push button from editors tools. The contents of this sheet include toggles, options, push buttons, and text editors. To easily recognise the purpose of these items. They are arranged as classified groups to allow different settings and to choose between different parameters and attributes. The top portion of this sheet, for example, gives choices ranging from toggle settings to database settings buttons. The second portion gives different settings ranging from workspace unit settings to plot schematics. The third portion concerns schematics level control, this is to allow switching schematics design levels on/off as desired by pressing the corresponding push button. The last group concerns colours automatic change for both elements and objects, this can be carried without selecting an element or object. For example, the attributes of database fields changed simply by selecting a push buttons.

### Sheets Advance Dialog

Configuration sheets can be advanced using the advance group as shown in Figure 5.56. This group consists of four push buttons that appears on the right corner of this sheet. They used to allow the user to advance, return, and enter design environment or return to schematic main interface. Figures (5.57, 5.58) activated using advance push buttons.

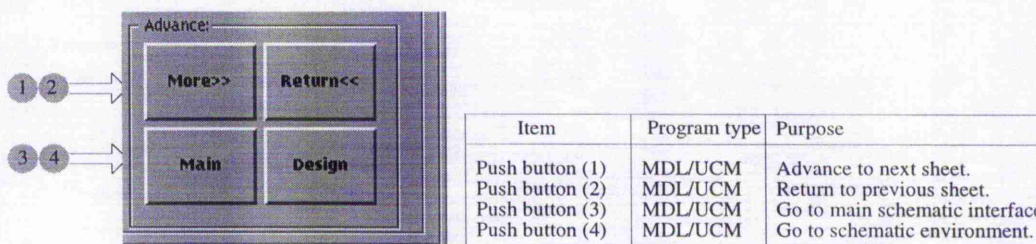


Figure 6.56 Sheet advance group.



Schematics Configuration Tools

Setting Menu (sheet 1)

Setting Toggles:

- ☐ Grid Lock
- ☐ Snap Lock
- ☐ Axis Lock
- ☐ Filling

Grid parameters:

Grid Master: 10.000  
Grid Reference: 10  
Grid Ration: 1.0000  
Config grid: Orthogonal

Text parameters:

Color: 0  
Style: 3 Engineering  
Weight: 0 Normal  
Width: 0.6.000  
Height: 0.6.000

Setting databases:

DB Mode: New  
Forms Mode: None  
Axtbs type: 1  
☐ Delete info

Working Unit:

Mixer Unit: ☐  
Sub-Units: ☐  
Units/master:   
Units/sub:

Sheet selection:

Size (297 \* 210 MM) A4 Size (17 \* 11 inch) B  
Size (420 \* 297 MM) A3 Size (22 \* 17 inch) C  
Size (594 \* 420 MM) A2 Size (34 \* 22 inch) D  
Size (840 \* 594 MM) A1 Size (44 \* 34 inch) E  
Size (11 \* 8.5 inch) A

Enter sheet info:

sheet/inf rev/code title Doc/numr  
sht/numr numr/sht

Schematics level control:

Hide reference: ☐ OFF ☐ ON  
Hide values: ☐ OFF ☐ ON

Database Fields:

Hide field 1: ☐ OFF ☐ ON  
Hide field 2: ☐ OFF ☐ ON  
Hide field 3: ☐ OFF ☐ ON  
Hide field 4: ☐ OFF ☐ ON  
Hide field 5: ☐ OFF ☐ ON  
Hide field 6: ☐ OFF ☐ ON

Hide objects:

Junctions: ☐ OFF ☐ ON  
Jun./numbers: ☐ OFF ☐ ON  
text: ☐ OFF ☐ ON  
power supply: ☐ OFF ☐ ON  
sheet: ☐ OFF ☐ ON  
wires ID: ☐ OFF ☐ ON

Power text: ☐ OFF ☐ ON  
Module port: ☐ OFF ☐ ON  
Module text: ☐ OFF ☐ ON  
Label: ☐ OFF ☐ ON  
Comment text: ☐ OFF ☐ ON  
Dashed line: ☐ OFF ☐ ON

Bus: ☐ OFF ☐ ON  
No connect: ☐ OFF ☐ ON  
components: ☐ OFF ☐ ON  
Wires: ☐ OFF ☐ ON  
Grids: ☐ OFF ☐ ON  
Pins: ☐ OFF ☐ ON

Pins numbers: ☐ OFF ☐ ON

Color configurations:

Change DB fields:

	White	Black	Red	Blue	Yellow	Green	Navy	C8	C9	C10
field 1	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
field 2	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
field 3	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
field 4	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
field 5	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON

Advance:

More>> Return<<  
Main Design

Figure 6.55 Schematics configuration sheet (1).

Schematics Configuration Tools

Setting Menu (sheet 2)

Color configurations:

Database Fields:

	White	Black	Red	Blue	Yellow	Green	Pink	Brown	L/blue	Gray	C11
field 6	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON

Schematics objects:

	White	Black	Red	Blue	Yellow	Green	Pink	Brown	L/blue	Gray	C11
Wire	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Bus	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Jun./numr	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Components	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Power object	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Power text	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Module port	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Module text	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Label	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Comment text	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Dashed line	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
No connect	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
PART ref	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Part value	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Part pins	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Earth	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Text	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON

Style configurations:

Schematics objects:

	Solid	Dot	M/dot	L/dot	Doc/dsh	shrt/dsh	Dst/d/dsh	L/sh/dsh
Wire	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Bus	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Components	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Power object	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Module port	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
Dashed line	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON
No connect	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON	<input type="checkbox"/> ON

Advance:

More>> <<Return  
Main Design

Library display:

List:

- shp.cel
- archp.cel
- archpat.cel
- areapatt.cel
- bat.cel
- capa.cel
- chap1.cel
- ddxamp.cel
- diode.cel
- disc.cel

Keyin:

Close

Figure 6.57 Schematics configuration tools sheet (2).



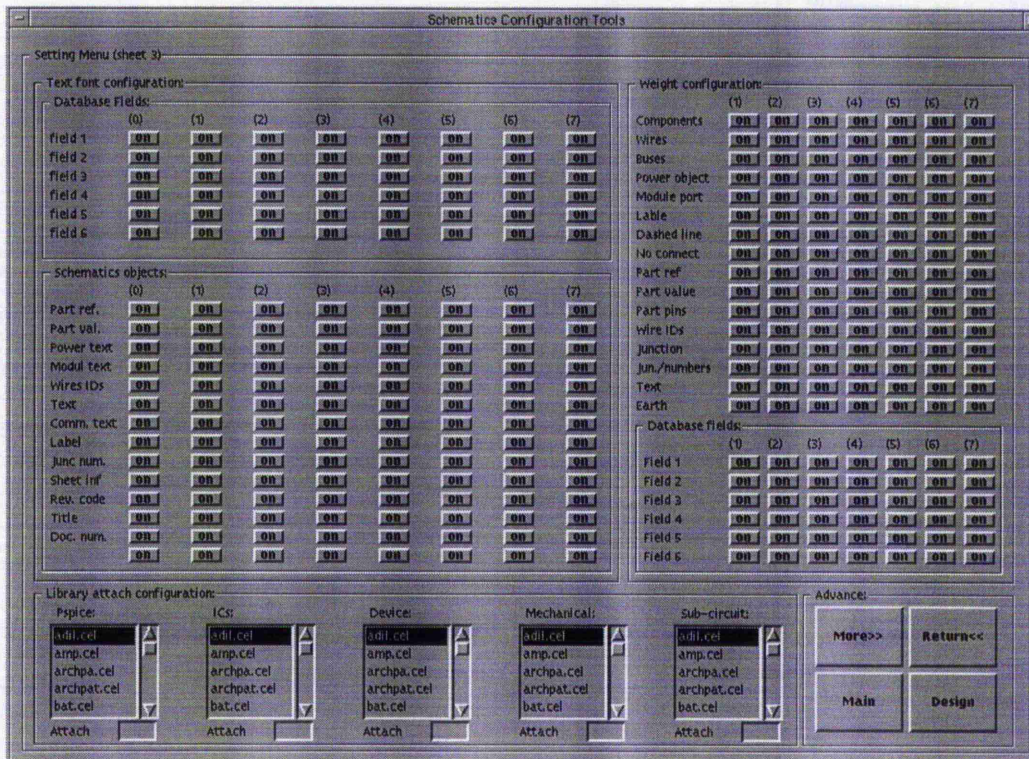


Figure 6.58 Schematics configuration tools sheet (3).

In addition to these tools, lock settings dialog is designed to control some of the MicroStation main graphics functions such as grid, snap, axis, and filling lock as shown in Figure 6.59.

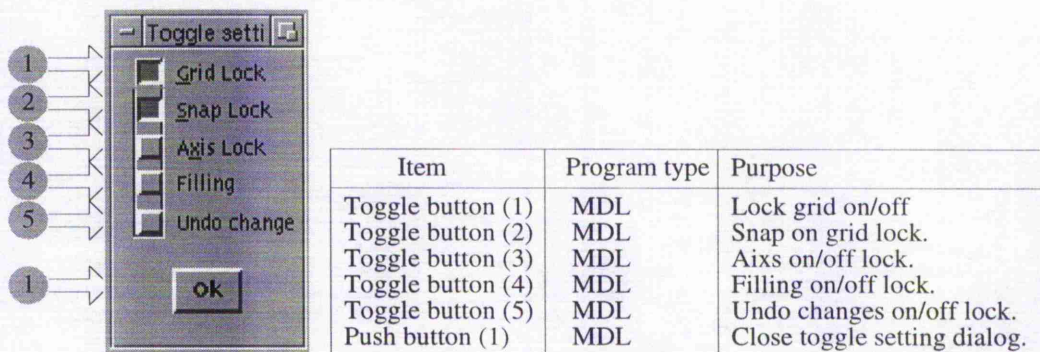


Figure 6.59 Toggle settings dialog.

### 6.8.3 Design By Levels

Many CAD/CAE systems have some type of level concept, sometimes refer to it as overlays or layers. Different levels can be used to draw schematic diagrams for electrical circuits by assigning to different elements. MicroStation has 63 different levels that are available in design environment file. They can be controlled to create a specific view of drawing. The advantage of using design by levels, is that it gives the flexibility to create



different design products, assists in design processes, and organising the design data by changing or manipulating portions of data and providing a quick check of the design for missing data by viewing one level at a time.

A dialog box was developed to control and display schematics at different levels. The concept of the design was by linking every element in the schematics diagram into different levels, so that they can be displayed as required. For example, cells which represent components placed different to wires and junctions. Figure 6.60 illustrates the idea of designing schematics diagrams using different levels.

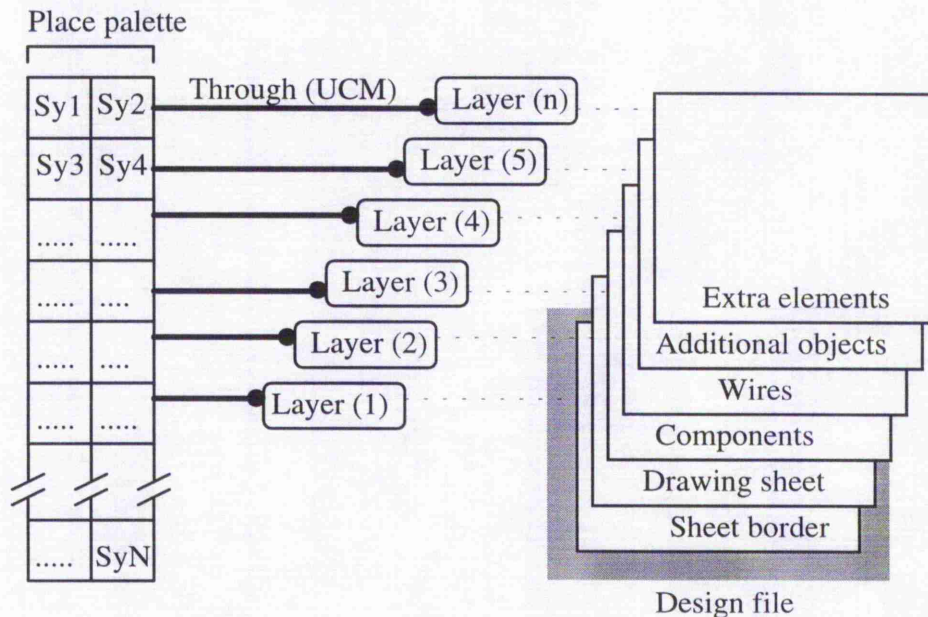


Figure 6.60 The concept of design by levels.

Schematics elements and objects can be displayed by selection push buttons in dialog box. Figure 6.61 shows levels control dialog designed as part of configuration tools (sheet 1). User commands used in assisting the search for design levels through file scan command.

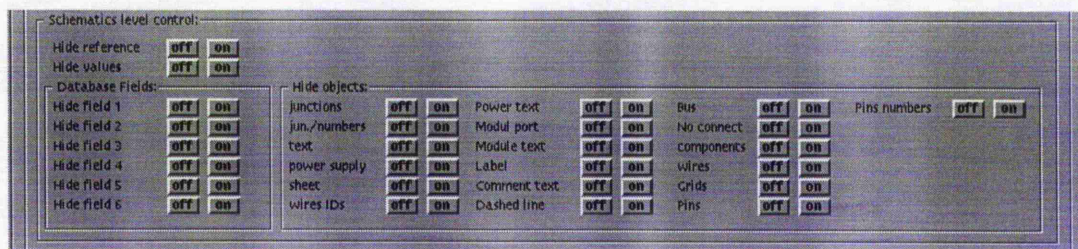


Figure 6.61 Schematics Levels control dialog.



Table [6.2]. Schematics level assignment.

Element number	Element name	Assigned level
1	Component reference	1
2	Component value	2
3	Database field 1	3
4	Database field 2	4
5	Database field 3	5
6	Database field 4	6
7	Database field 5	7
8	Database field 6	8
9	Junctions	9
10	Junction numbers	10
11	Text	11
12	Power supply	12
13	Sheet	13
14	Sheet border	14
15	Wires IDs	15
16	Power text	16
17	Module ports	17
18	Module ports text	18
19	Comment text	19
20	Dashed lines	20
21	Buses	21
22	No connect	22
23	Components	23
24	Wires	24
25	Pins	25
26	Pins numbers	26
27	Grid	27

## 6.9 DESIGN OF ASCII REPORTS TOOLS

ASCII reporting tools are designed to generate schematics reports. As it was mentioned in Chapter 2, data extraction is a process of extracting information using designed software tools. Design reports such as Connectivity, Drawing information, Components list and many other can quickly be generated. This method, however, is proved to be more efficient comparing to the use of database. Chapter 5 has stated, the use of database for generating different design reports is carried through two different steps (i.e. Store-Extract). Once these reports are generated, a review function is selected and data are displayed. Figure 6.62 shows design group of ASCII report generation dialog.

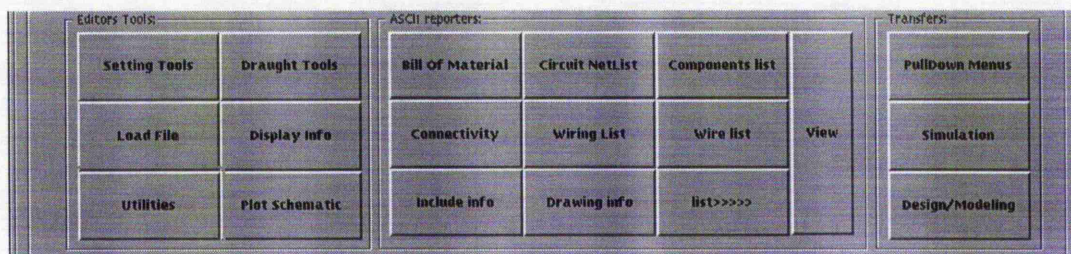


Figure 6.62 ASCII reports dialog.

## **6.10 DESIGN OF TRANSFERS TOOLS**

Transfers tools are used to transfer from one environment to another. Once the execution of these tools is invoked by means of UCMs programs. The involved program causes the associated tools to be displayed. The function of pull down menus, for example, causes pull down design tools to be displayed and automatically sets window viewing. These tools, however, does not process or create any files.

### **6.10.1 Pull-Down Menus Environment Design**

A more popular solution to the problem of effective menu selection is to have pull-down menus. They are designed to allow more room for the circuit schematics display. Commands are emerged together to perform a single command function options in hierarchical way known as (menu and sub-menu). Most CAD/CAE packages, however, make use of pull-down menus as they are used to cut down the amount of memorisation required by listing the range of available functions options. In addition, they can be changed to accommodate different applications.

Designing of pop-down menus is another way of developing GUI. Functions such as file loading, settings, Components different palettes, and database for design report generation and others are developed. Some of these menus contain items built from MicroStation basic command dictionary and by using the MDL and Marco language UCMs, these commands are combined to form functions commonly used in CAD activities. Figure 6.63 provides an illustration of hierarchical structure of pull down menus interface design. Figure 6.64 shows the corresponding dialog box, giving the access to all the commands, functions, settings, dialog boxes, etc. To be able to activate these functions two operations are provided by using the screen cursor or using assigned hot keys from the keyboard.

Based on the hierarchical structure shown in Figure 6.63, various selection of pull-down functions were developed. In this form pull-down menus design hold a large number of commands and single applications. Figures (6.65-6.69) show a bar of menu titles indicates the major command groupings selected for displaying.

In addition, control keys defined as part of pull-down menus to achieve high speed of interaction. Command styles, keystrokes are bound to individual commands, and simply by pressing these keys invokes the commands and applications are executed. Function keys have special assignment during certain mode of operations. This includes components placement, database entry dialog, etc. Table [6.3], provides an outline of data entry control and their relevant function keys.



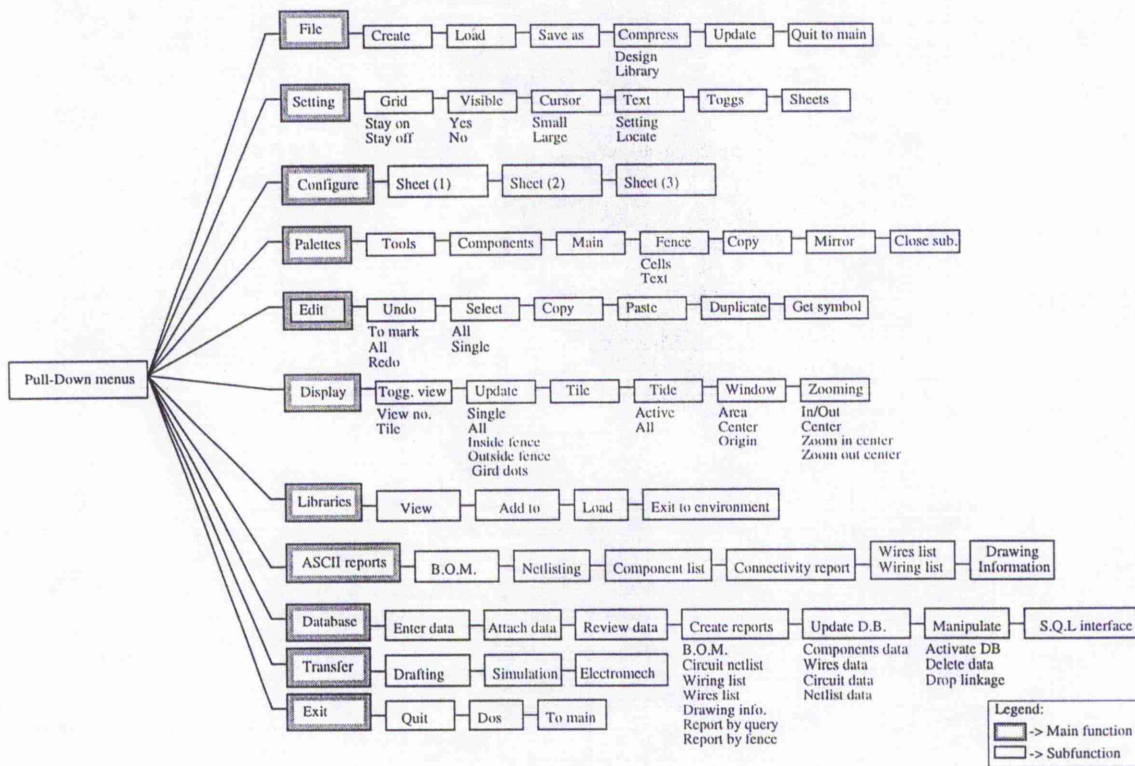


Figure 6.63 Pull-down menus tools design hierarchy.

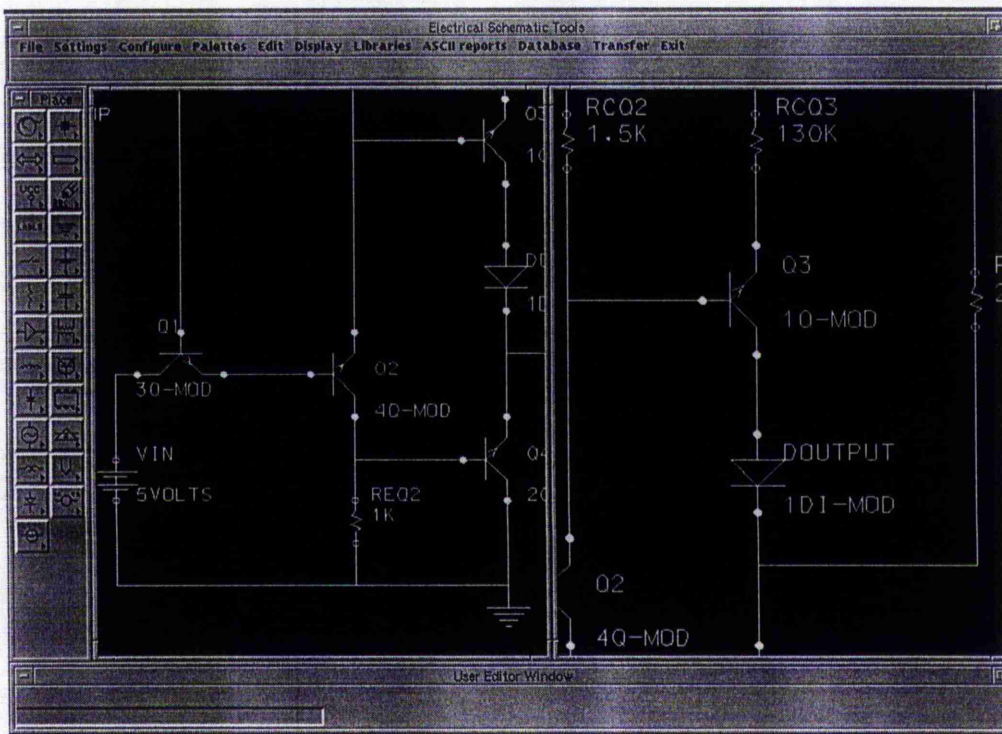


Figure 6.64 Pull-down menus environment.



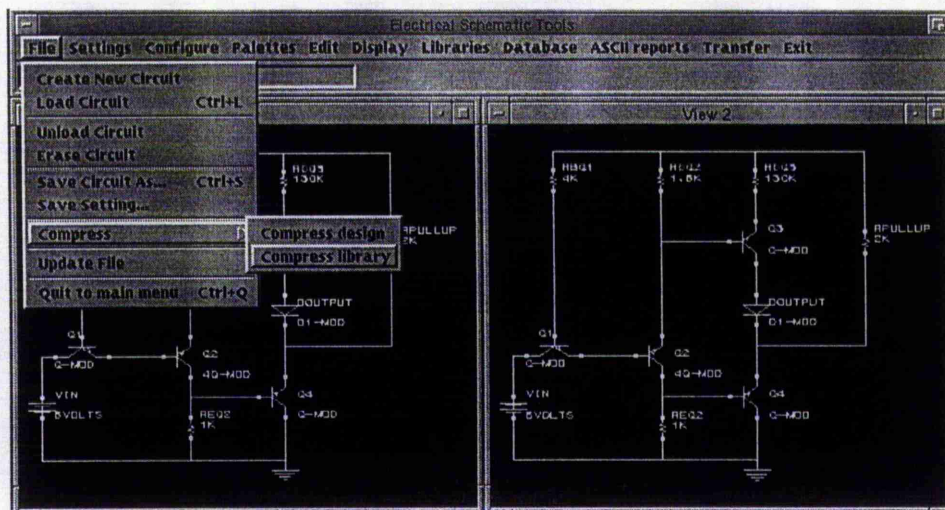


Figure 6.65 File tools submenus.

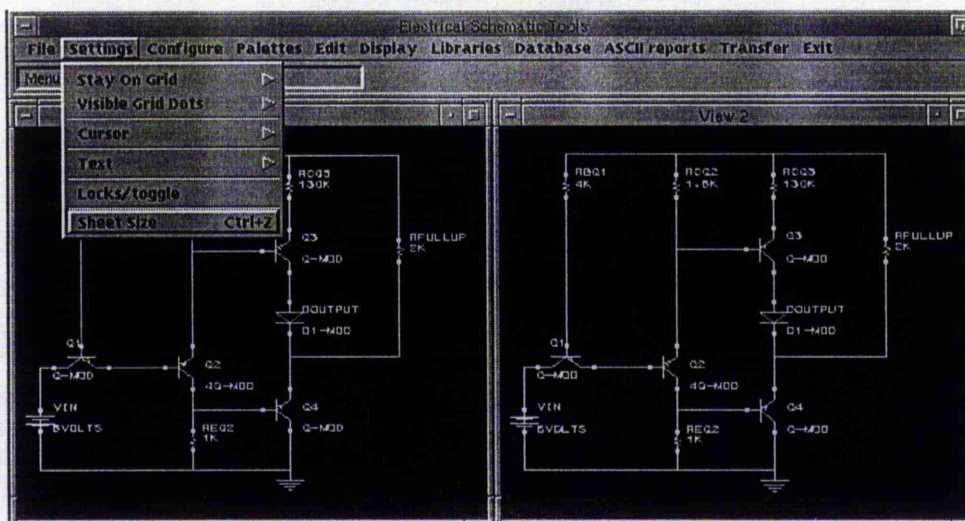


Figure 6.66 Settings tools submenus.

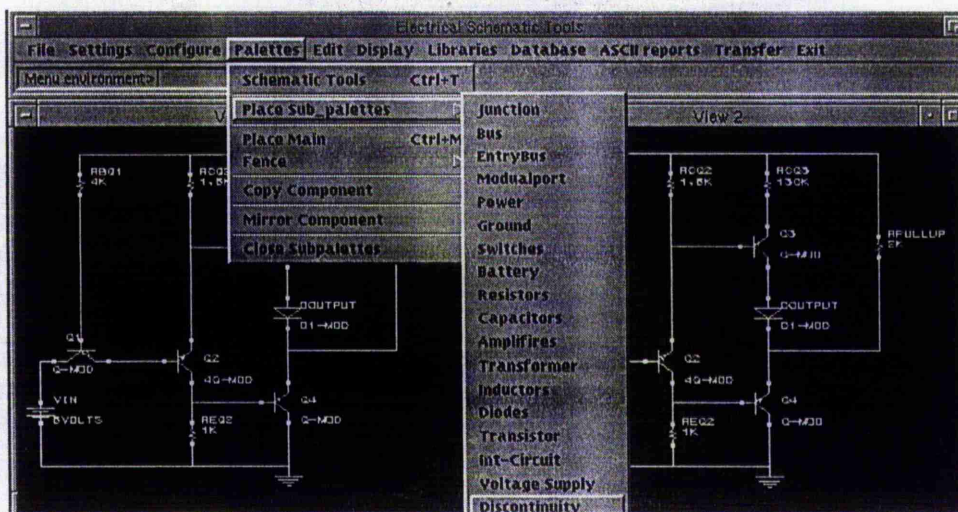


Figure 6.67 Palettes submenus.



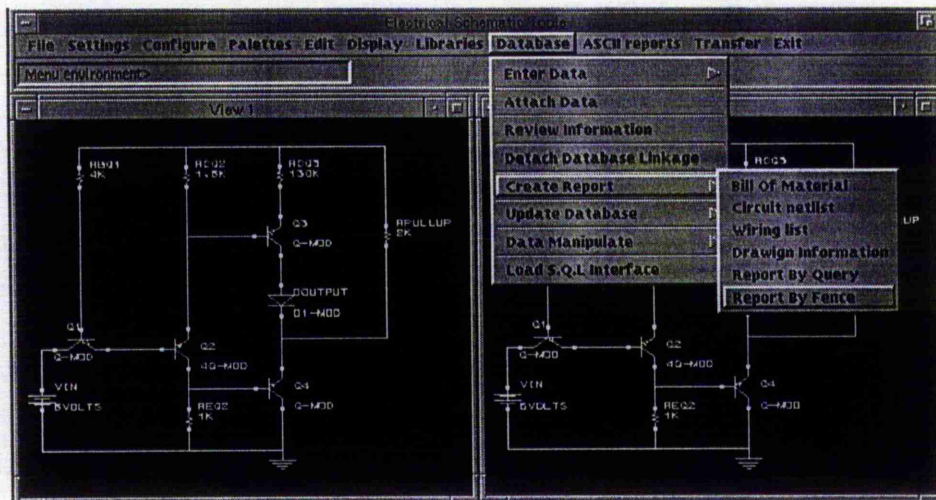


Figure 6.68 Database submenus.

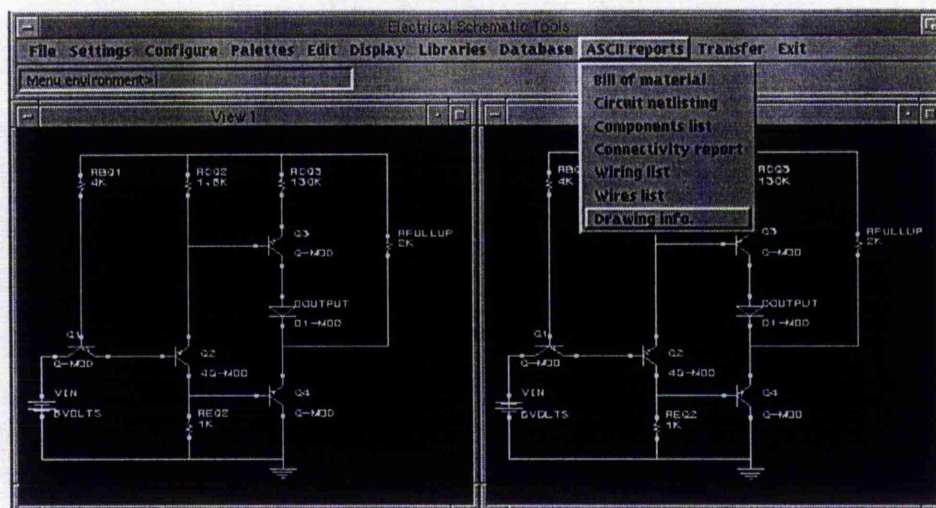


Figure 6.69 ASCII reports submenus.

Tabel [6.3]. Commands and their functions keys.

Defined key	Command	Action
Ctrl+L	dialog openfile	Activate load circuit dialog.
Ctrl+S	dialog saveas	Activate save design dialog.
Ctrl+Q	UCI=quit	Quit and load main interface
Ctrl+Z	mdl load sheet	Activate sheet size selection dialog.
Ctrl+C	mdl load config	Activate configuration first sheet
Ctrl+V	mdl load 2config	Activate configuration second sheet
Ctrl+B	mdl load 3config	Activate configuration third sheet
Ctrl+T	mdl load set	Activate schematic modify tools palette
Ctrl+M	mdl load place	Activate place main palette
Ctrl+D	mdl load draft	Activate draft working environment
Ctrl+U	dos/do	Activate PSPICE through dos command
Ctrl+N	mdl load design	Activate design/modelling environment
Ctrl+q	quit/exit	Quit system and exit microstation

### 6.10.2 3D Environment Design

3D design environment is designed and developed to construct 3D models and to facilitate the integration between 2D and 3D designs. It contains different design tools and functions ranging from file creation and library editing tools to database. For the advantages of speeding up the interaction, these tools are equipped with hot keys used for quick execution. Figure 7.70 illustrates the hierarchal structure of 3D modelling environment design.

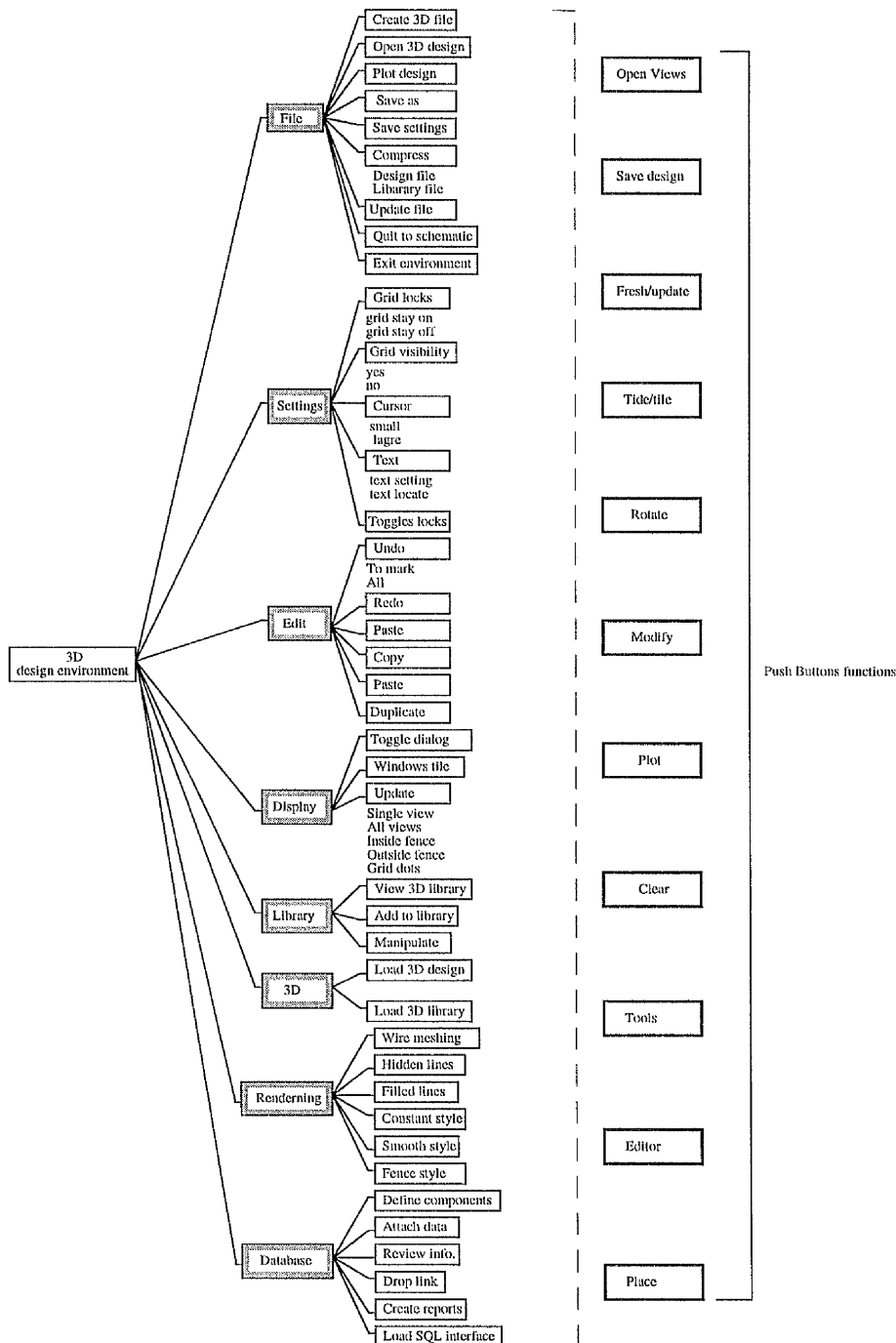


Figure 6.70 3D environment design hierarchy.



### Creating 3D Model

The process of creating a 3D model using this environment, starts by first setting up working space using the configuration tools explained earlier. This is followed by the loading of cell libraries that contain 3D components. Using the provided frame editor and tools, 3D cells can be added or deleted from the design workspace. To ease the creation of 3D model, four different views are preferably used so that cells are located in the precise position suitable for wiring and manipulation. Additional tools are also provided such as database tools used to interface with dbase(iv) environment. These tools are used for data attributes editing and other definitions. Figure 6.71 shows the designed 3D modelling environment and its tools.

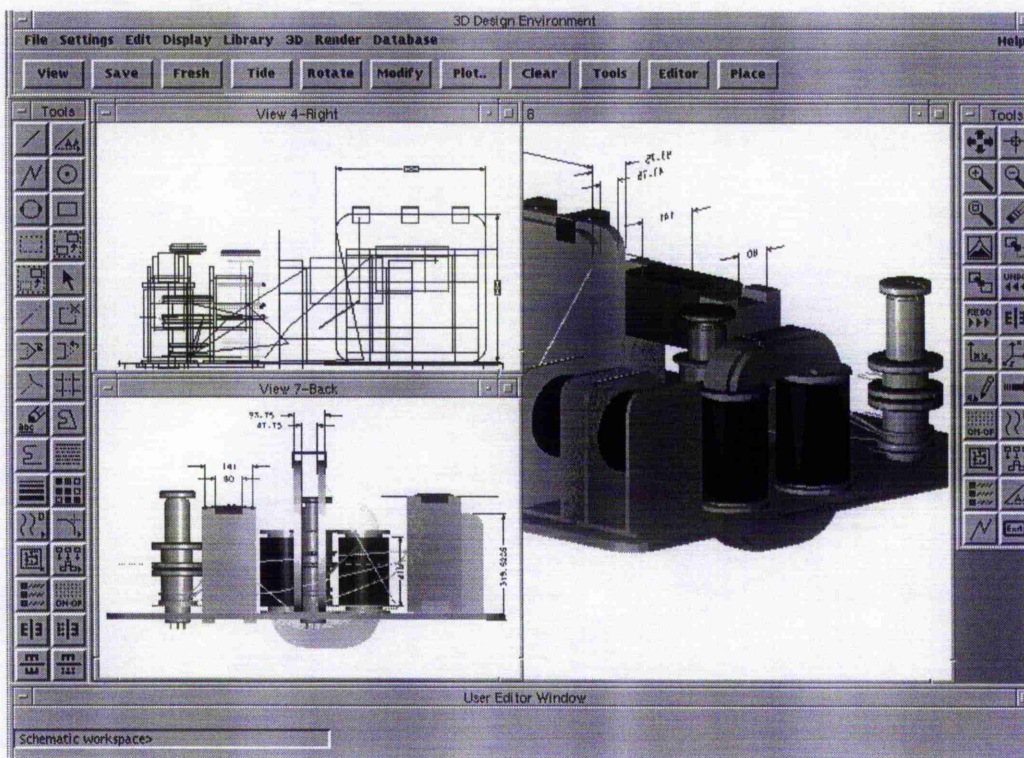


Figure 6.71 The 3D modelling environment.

### 6.11 DESIGN OF LIBRARIES TOOLS

Library management is one of the most critical elements to successful CAE. It is the effective development, distribution and control of symbol libraries. The use of accurate and complete libraries provides a platform for better schematics layout and documentation. In the preparation of components symbols, the storage of a CAD system can be very beneficial. It becomes necessary to construct a component symbol so that, it can be stored and retrieved as desired.

This section introduces schematic components libraries environment used to group and organise components that are used with the assistance of schematics tools. Figure 6.72 shows the hierarchy structure of the 2D library design environment. Library design environment entered through the selection of edit button from the 2D main interface.

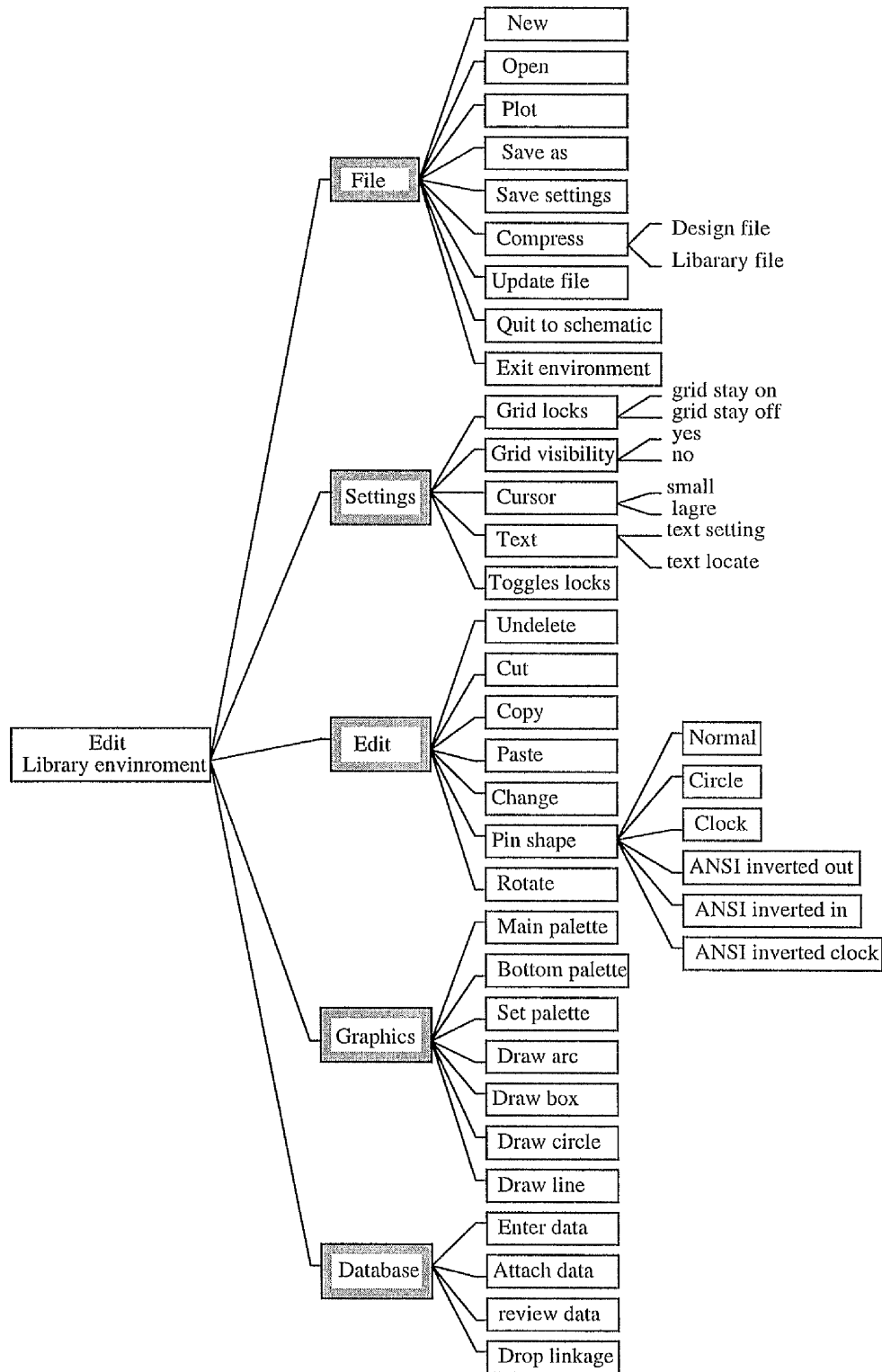


Figure 6.72 Libraries tools edit environment.



Figures (6.73-6.75) show the different tools that have been designed and developed to assist in components cells construction within 2D cells design environment.

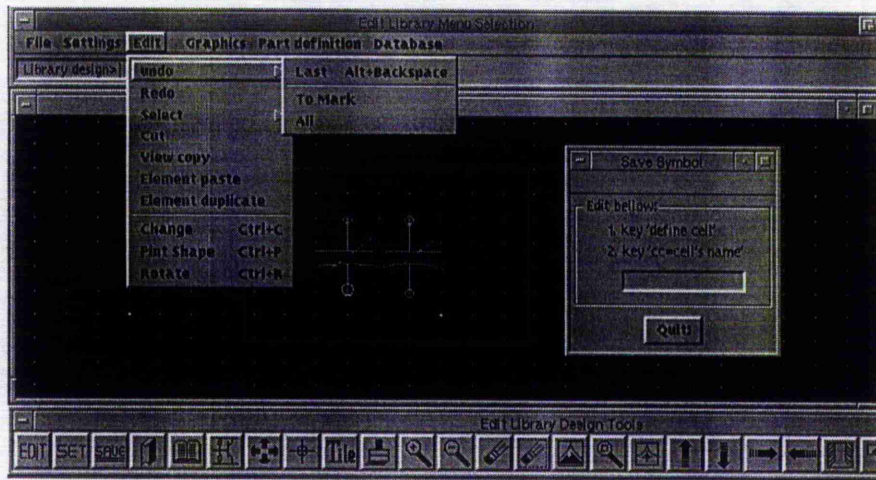


Figure 6.73 Library design tools (Edit submenus)

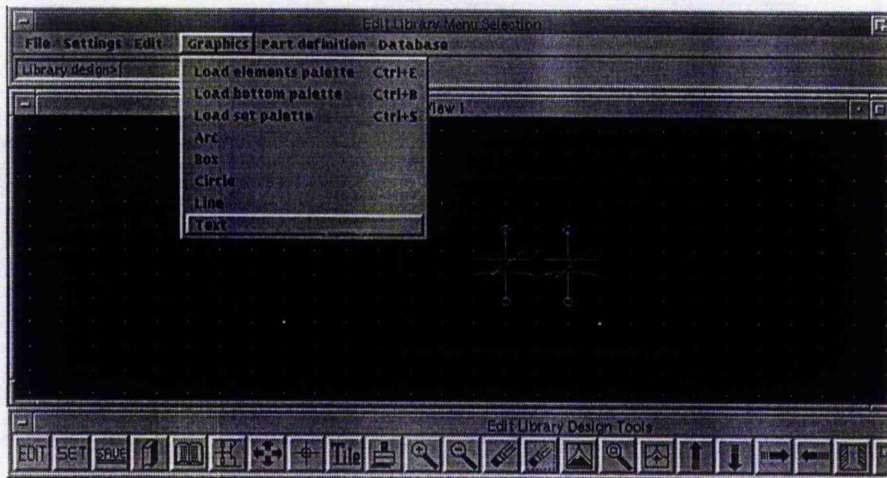


Figure 6.74 Library design tools (graphics submenus).

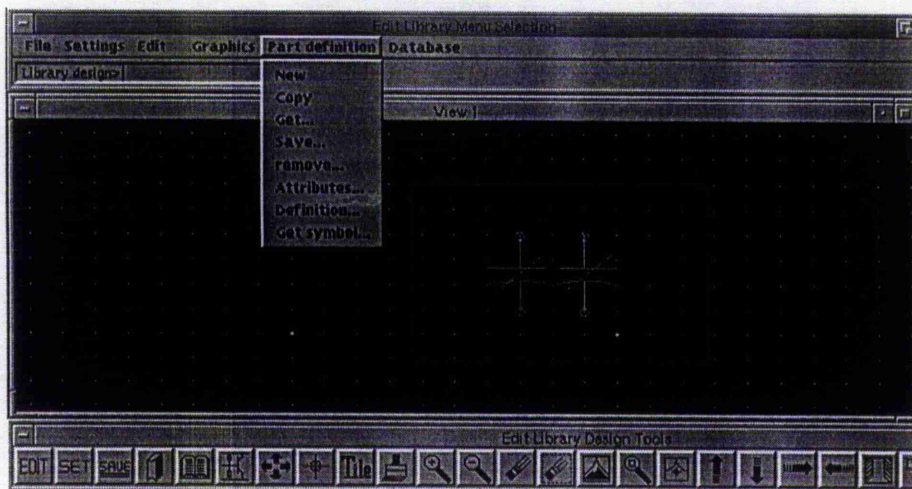


Figure 6.75 Library design tools (part definition submenus).



There is a fair selection of symbols provided to represent both components and schematics layout generally. For example, the place palette contains a variety of common used symbols for SPICE netlist generation. These symbols are presented as sub-palettes and are loaded and manipulated using UCMs programs linkage. They are organised into different libraries so that, they can be attached and called up as required. Figure 6.76 illustrates different libraries classification dialog.

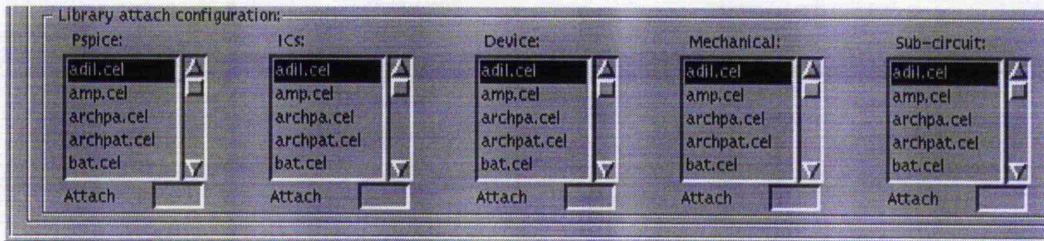


Figure 6.76 Components libraries classification.

### 6.11.1 Symbols Creation

Traditionally, drafting methods for creating a symbol, or graphical representation of standard parts, must be drawn individually using a template or other construction methods. For this purpose, library design environment equipped with its tools is designed to assist in creating customised or standard schematics symbols.

Symbol creation begins with the formation of the connect nodes (i.e. base points), which must be created as permanent points that are used for placement and positioning. Once the construction of the whole part is completed. A fence is built round it and the origin is specified and used as the reference point for future placement. The symbol is, therefore, stored using save symbol dialog box. Figure 6.77 provides an example of transformer symbol design.

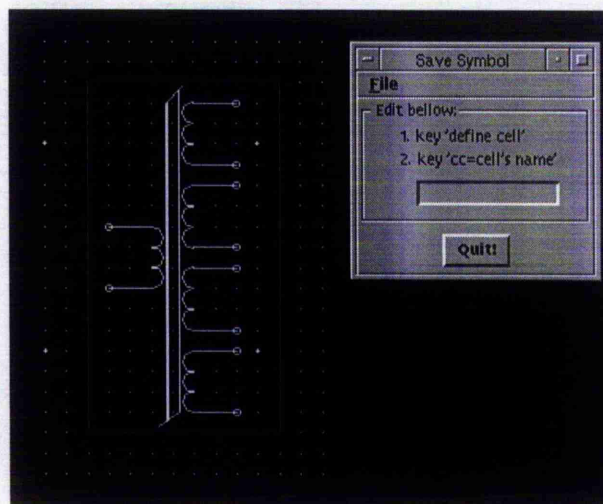


Figure 6.77 An example of transformer symbol design.

More complex components can be created and customised by nesting together individual symbols using the IEEE symbols main palette shown in Figure 6.78 and sub-palettes shown. Using modify tools main palette, these symbols may be recalled, changed, and resorted.

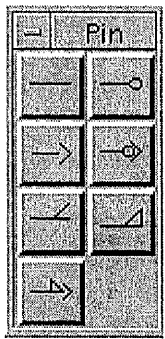
<i>Left column</i>		<i>Right column</i>
Line pin shape		Normal pin shape
Clock pin shape		Bubble clock pin shape
ANSI inverted out		ANSI inverted in
ANSI inverted clock		

Figure 6.78 Palette of components different pins.

Figures (6.79-6.84) show some of the common used components symbols stored as library cells.

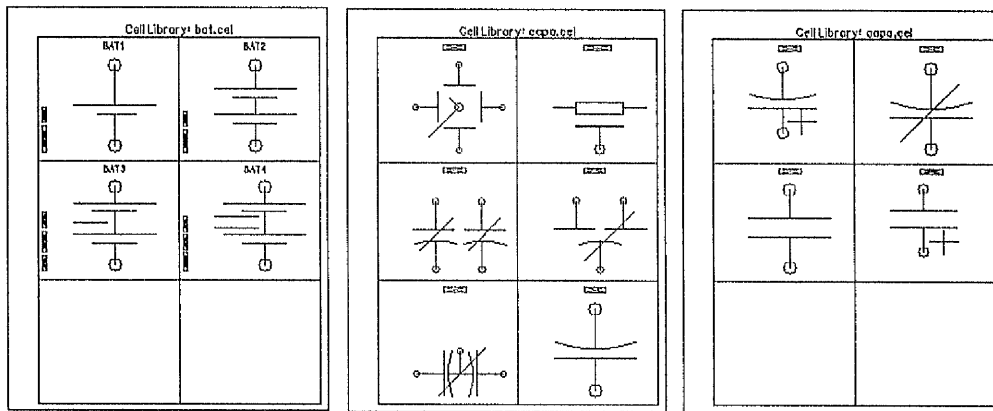


Figure 6.79 Batteries-capacitors symbols.

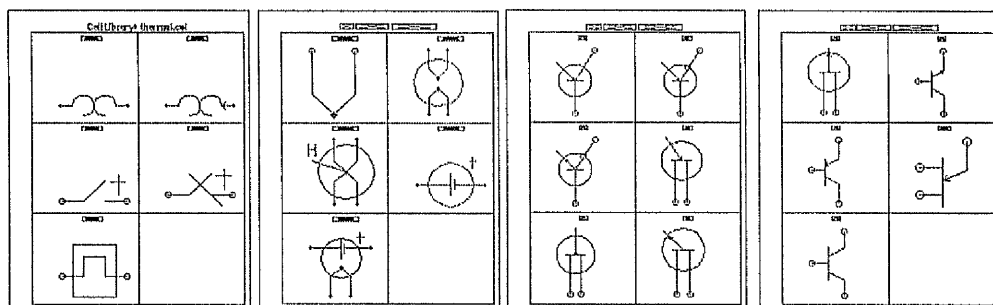


Figure 6.80 Thermal elements-transistors symbols.

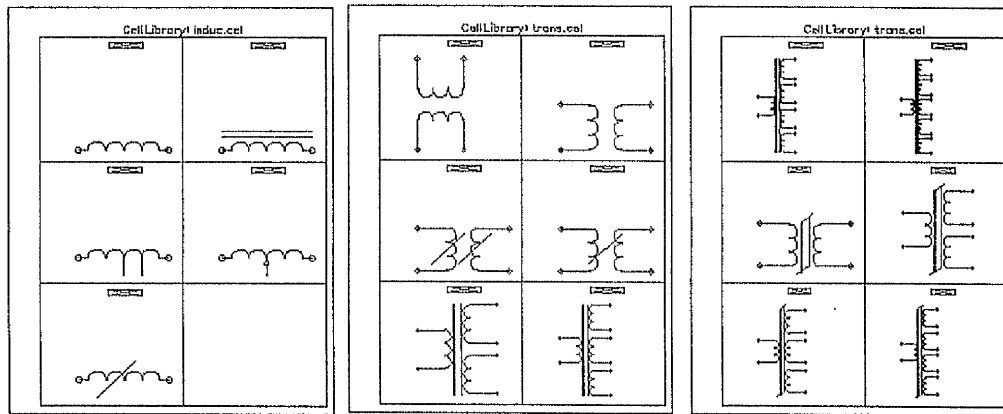


Figure 6.81 Inductors-transformers symbols.

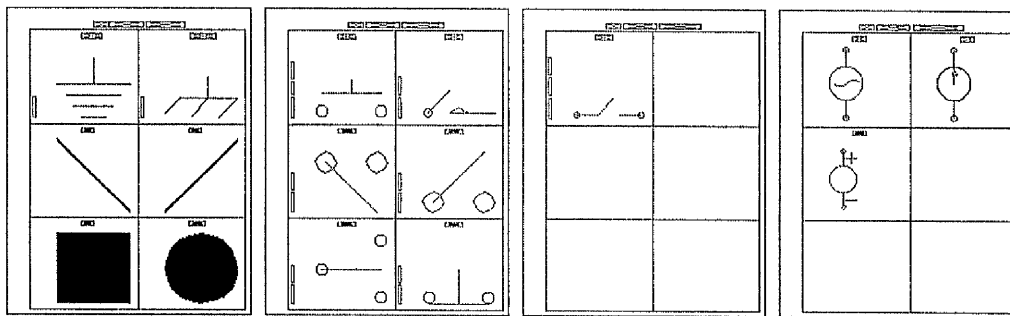


Figure 6.82 Schematic objects-switches symbols.

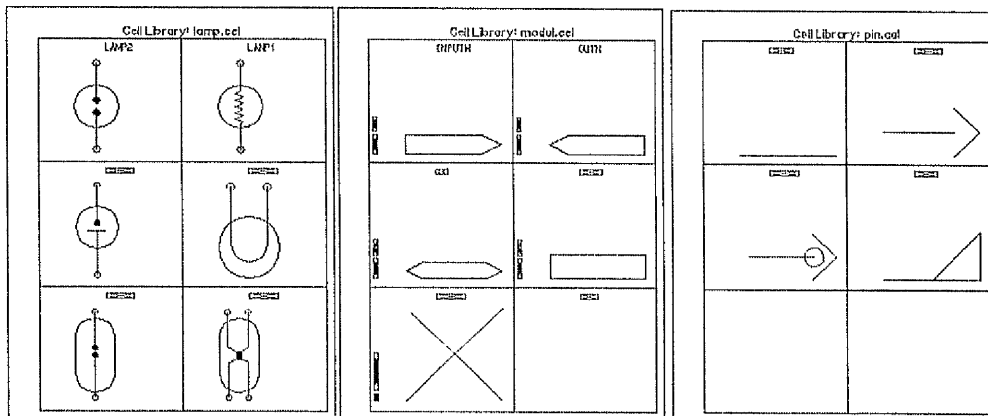


Figure 6.83 Module ports-pins symbols.



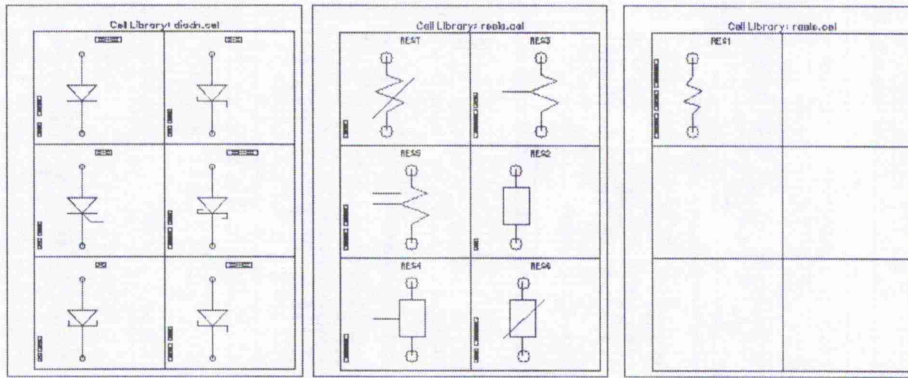


Figure 6.84 Diodes-resistors symbols.

## 6.12 DESIGN OF DATABASE TOOLS SET

Chapter 5 has shown that the extracted design data used to describe electronic CAD circuits. This would involve components data, circuit layouts, netlists, or other design documentation arise from different design steps using various design tools.

Generally, in CAD systems the exchange of design data between design tools is carried out by files which are designed and formatted in a manner to a specific CAD company. The design of database tools prompts the design using different text editors which assist in gathering inputs from visual and presents the results as an output format files that can be reviewed. This can be achieved once the interface between design environment and dbase(iv) is established through designed communication tools. These tools are responsible for accepting requests from the user data manipulation, transferring data back and forth. Figure 6.85 shows database tools set main dialog designed to input, retrieve, update, and manipulate circuits schematic design.

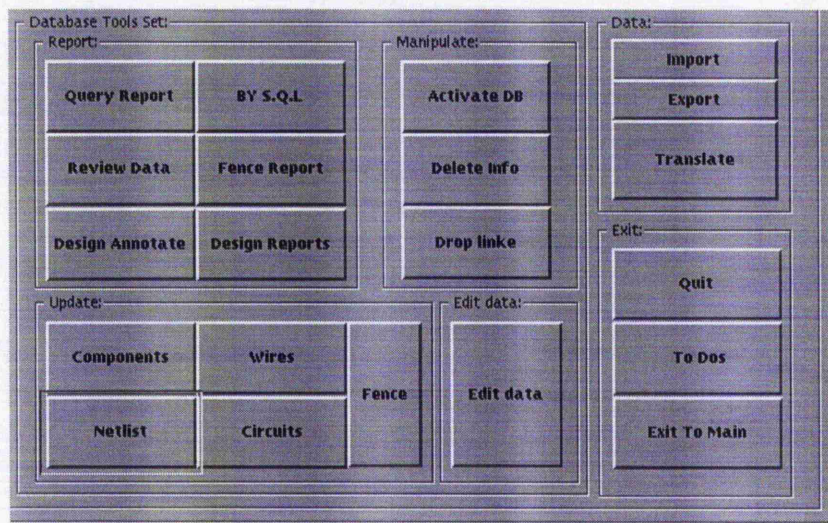


Figure 6.85 Database tools dialog.



Using the SQL software statements. Database tools allow different operations and descriptions to be performed. Table [5.2] shown in Chapter 5 Section 5.5, illustrates these statements. Components (i.e. cell libraries) and other circuit related elements contain design data that can be scanned and extracted. Figures (6.86-6.101) included in the following pages show dialog boxes activated once selecting the corresponding database push buttons. In addition, each dialog is accompanied with a descriptive table to show the purpose of each individual item of its contents.

### Data Query Main Dialog

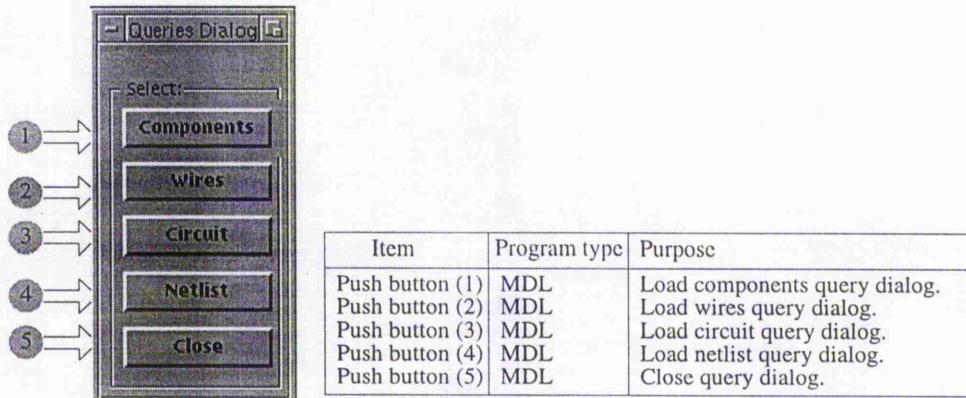


Figure 6.86 Database query main dialog.

### Components Query Main Dialog

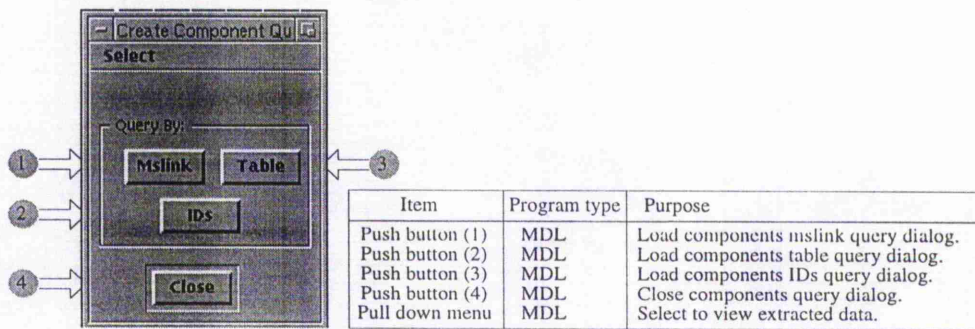


Figure 6.87 Components query main dialog.

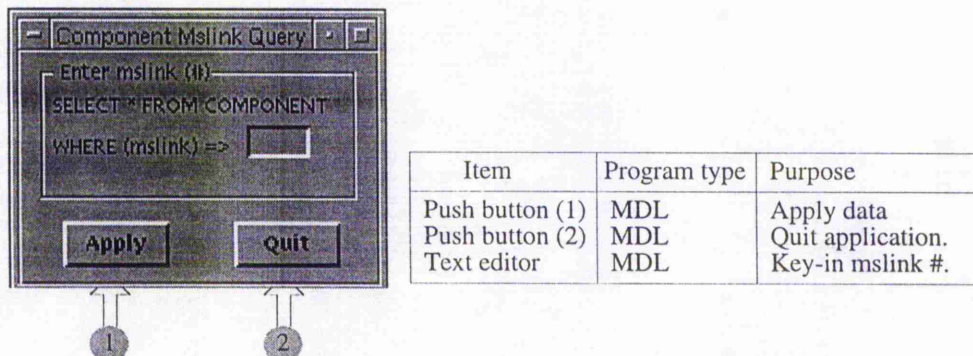


Figure 6.88 Components mslink query dialog.

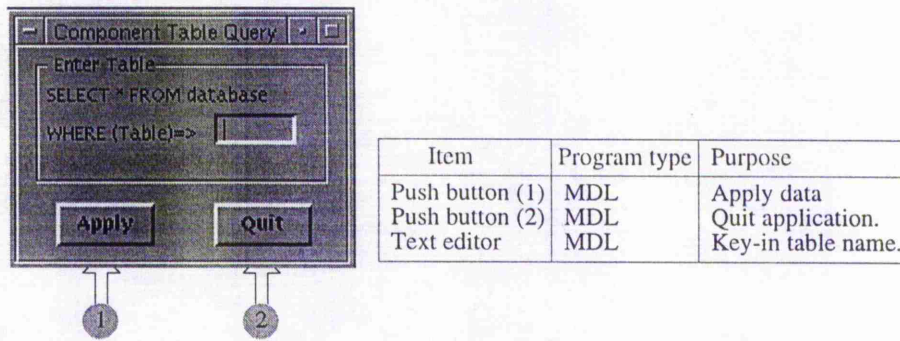


Figure 6.89 Components table query dialog.

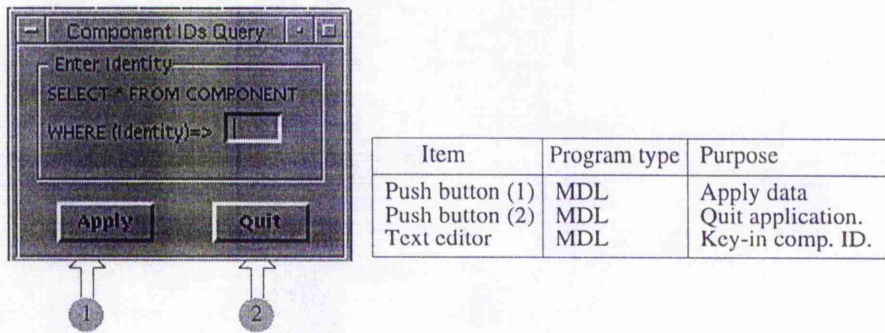


Figure 6.90 Components IDs query dialog.

### Wires Query Main Dialog

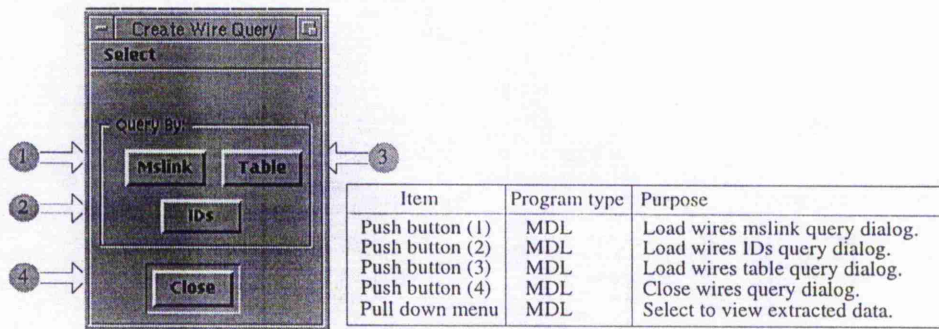


Figure 6.91 Wires query main dialog.

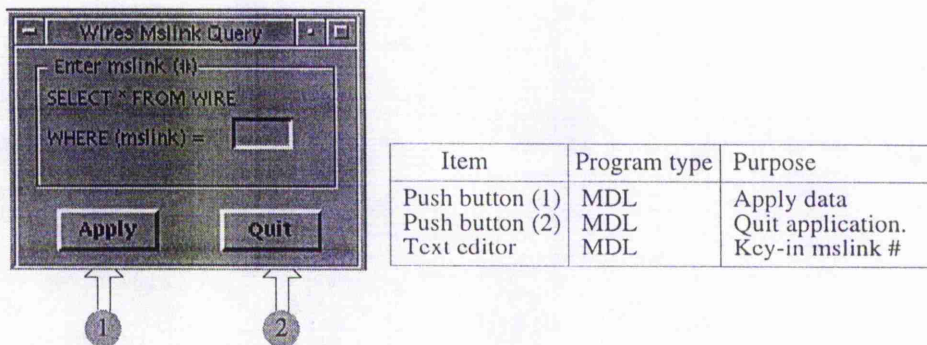


Figure 6.92 Wires mslink query dialog.



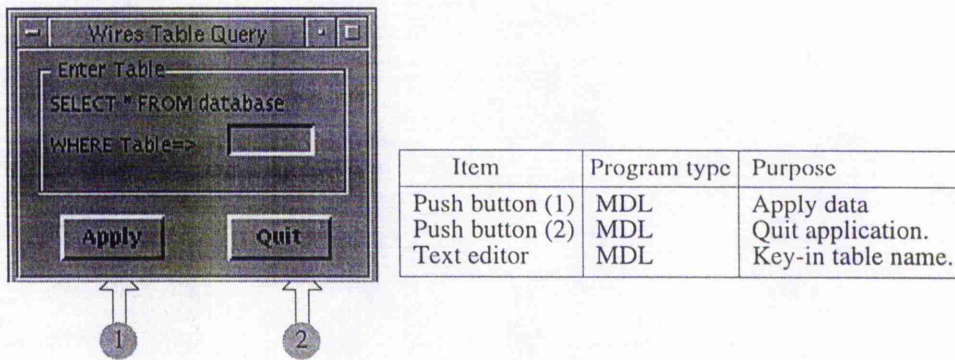


Figure 6.93 Wires table query dialog.

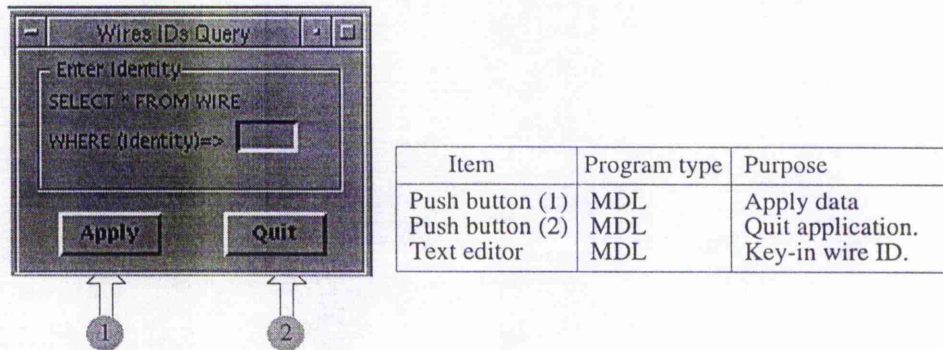


Figure 6.94 Wires IDS dialog.

### Netlist Query Main Dialog

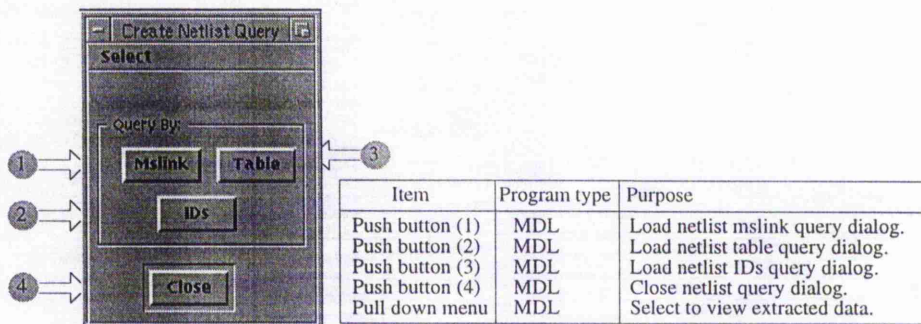


Figure 6.95 Netlist query main dialog.

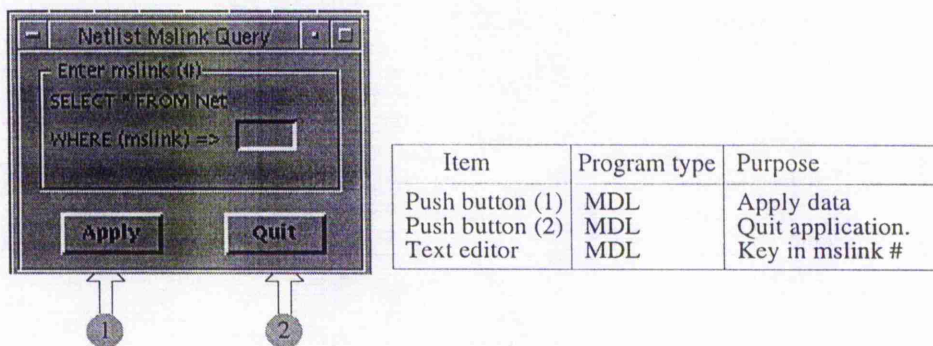


Figure 6.96 Netlist mslink query dialog.

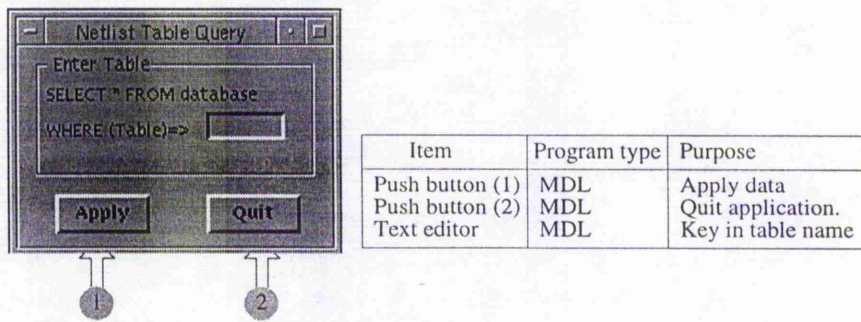


Figure 6.97 Netlist tables query dialog.

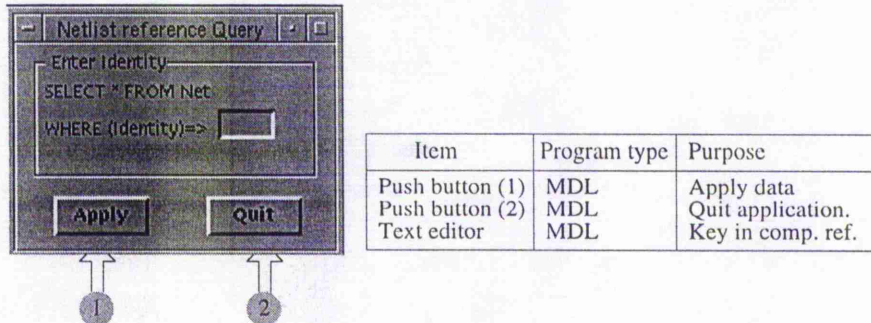


Figure 6.98 Netlist reference query dialog.

### Create Reports Main Dialog

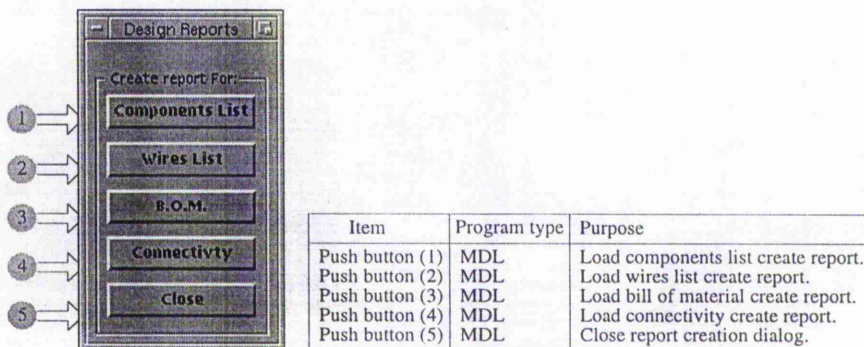


Figure 6.99 Reports creation main dialog.

### Retrieve Data Main Dialog

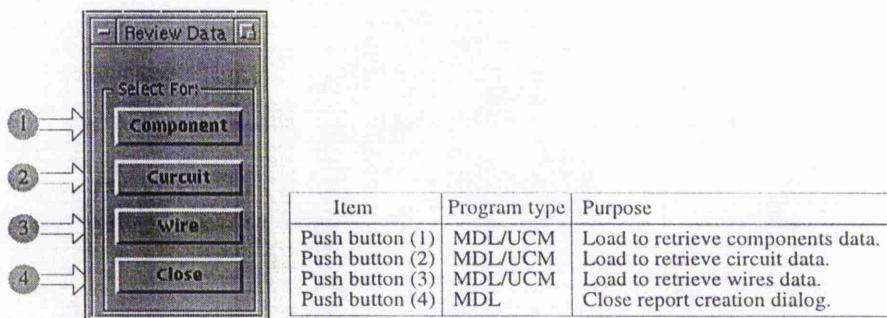


Figure 6.100 Data retrieval main dialog.



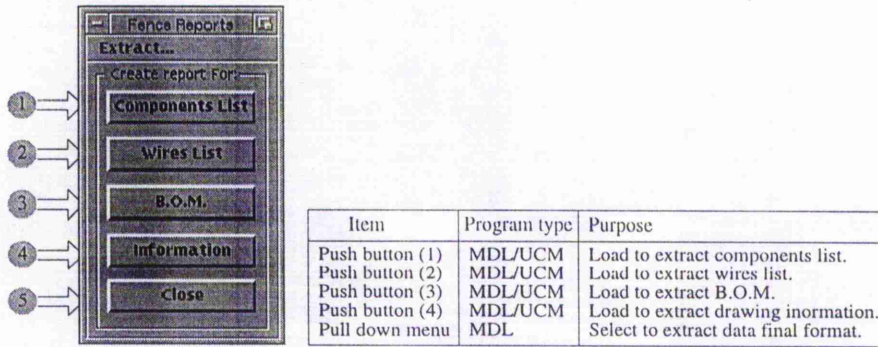
**Create Fence Reports Main Dialog**

Figure 6.101 Fence report generation dialog.

In addition to database queries dialog boxes, each schematic element has associated data. Data of an element are specified either by user through data editing dialog boxes, or automatically extracted by means of software programs. Components dialog, for example, differs from wires dialog boxes. Figures (6.102-6.107), show dialog boxes designed for schematics elements data definition.

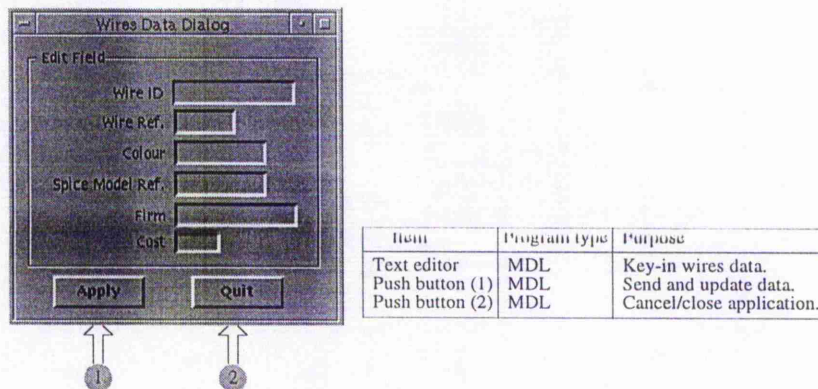
**Wires Data Main Dialog**

Figure 6.102 Wires editing data dialog.

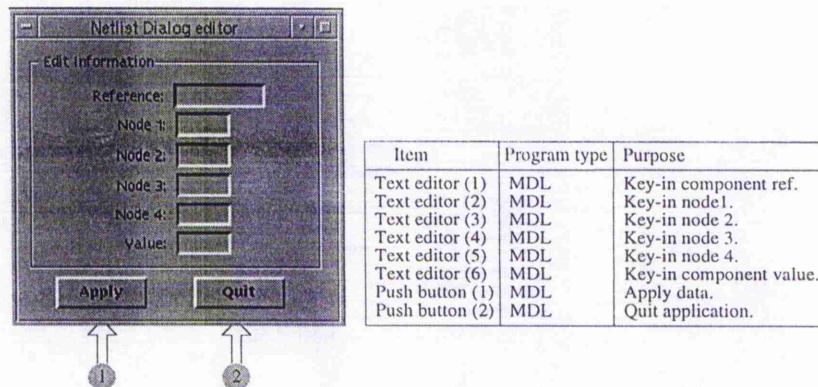
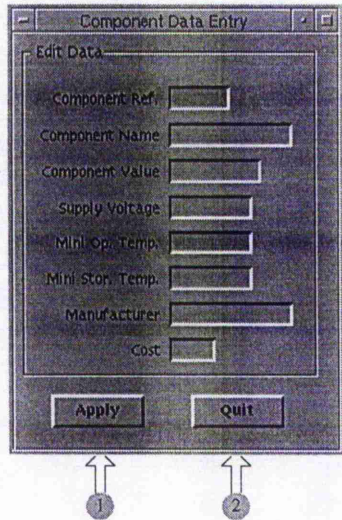
**Edit Netlist Dialog**

Figure 6.103 Netlist editor dialog.

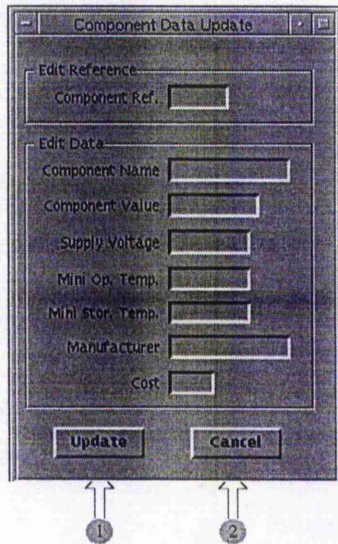
### Components Data Main Dialog



Item	Program type	Purpose
Text editor	MDL	Key-in components data.
Push button (1)	MDL	Send and enter data.
Push button (2)	MDL	Cancel/close application.

Figure 6.104 Components editing data dialog.

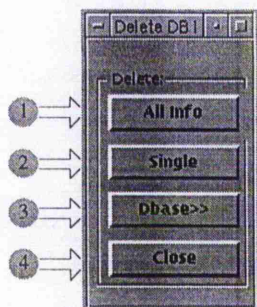
### Update Data Main Dialog



Item	Program type	Purpose
Text editor	MDL	Key in component reference.
Edit data (text)	MDL	Key in data to be updated.
Push button (1)	MDL	Send and update data.
Push button (2)	MDL	Cancel/close application.

Figure 6.105 Components update data dialog.

### Manipulate Database Main Dialog



Item	Program type	Purpose
Push button (1)	MDL	Load clear all tables main dialog.
Push button (2)	MDL	Load clear single records dialog.
Push button (3)	MDL	Load database (dbase IV) environment.
Push button (4)	MDL	Close/input delete dialog.

Figure 6.106 Clear database records main dialog.



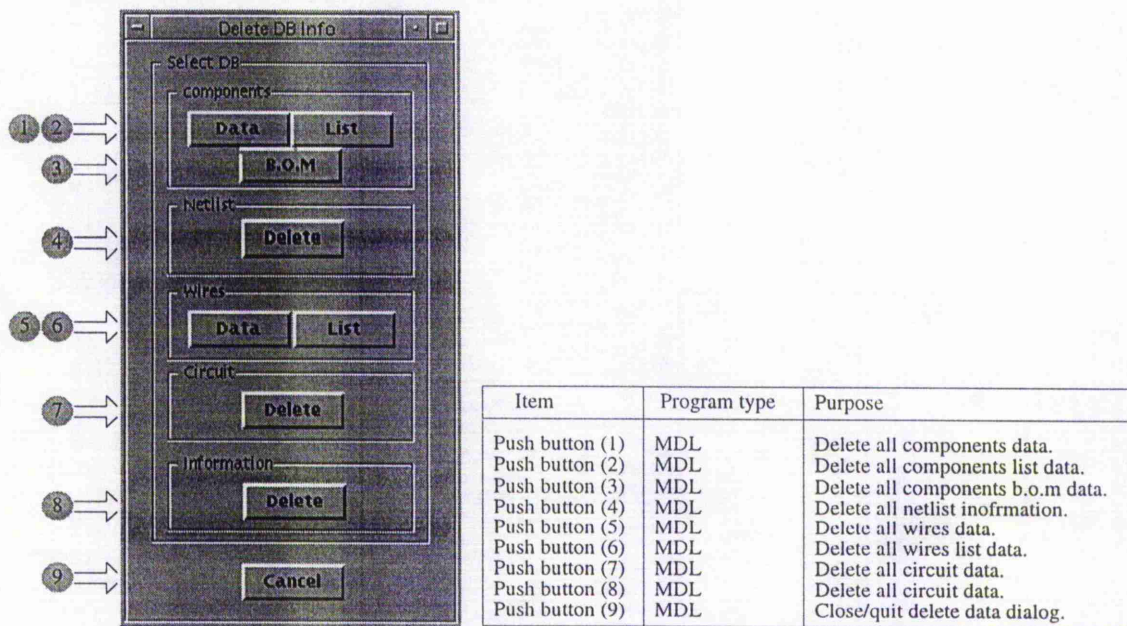


Figure 6.107 Clear database main dialog.

### Components Modelling Templates

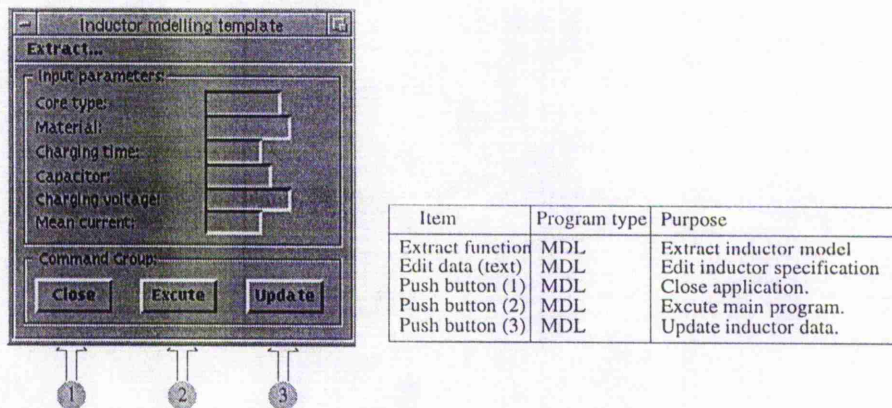


Figure 6.108 Inductor modelling template.

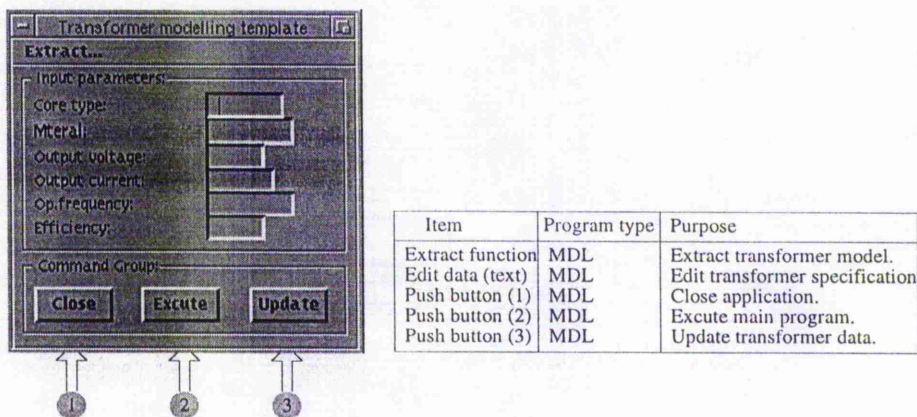


Figure 6.109 Transformer modelling template.

### **6.13 DESIGN OF DATA TOOLS**

These tools are designed to provide facilities such as importing and exporting of design data. Different design data formats can be imported and displayed into design file, while data translation function is used to translate different data formats during the extraction of database. Data translation ranging from netlist formats into B.O.M. extraction can be performed.

### **6.14 DESIGN OF EXIT TOOLS**

These tools are designed to perform three different independent actions. The quit function is used to quit the design environment and out of the MicroStation. The DOS function is used to view data or loading up other applications windows-based. The last is the exit to main, used to unload main interface and loading up the system main dialog.

### **6.15 CONCLUSION**

This chapter has shown the importance of providing GUI in system integration by reviewing a previously designed systems and their methodologies towards the construction of graphics tools. Using MicroStation supported languages a GUI has been designed and developed, gathering different tools and applications as a unified working environment. The design of GUI offers many benefits in environment integration. These benefits are listed in the conclusion of Chapter 7 of the software performance and evaluation.

### **REFERENCES**

- [6.1] Donald H., and Pauline M., "Computer graphics," Prentice-hall, Inc. ISBN 0-13-165382-2, 1986.
- [6. 2] Steven M., R., "Computer-Aided for VLSI design," ISBN 0-201-05824-3, 1987.
- [6.3] Stefan H., et al. "VISTA: User interface, Task Level, and Tool Interaction," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, Vol. 14, No. 10, pp. 1209-1222, October 1995.
- [6.4] Mark R., "PRIDE: An integrated design environment for semiconductor device simulation," IEEE Trans. on Computer-Aided Design, Vol. 10, No. 9, pp. 1163-1174, September 1991.

- [6.5] Wdward W., et al. "SIML-IPX: A utility-Based integrated system fro process simulation," IEEE Trans. on Computer-Aided Design, Vol. 11, No. 7, pp. 911-920, July 1992.
- [6.6] Len B., Joelle c., "Developing software for the user interface," ISBN 0-201-51046-4, 1991.
- [6.7] Marina Z., and Paolo G., "A conceptual model for design management," Computer-Aided Design Journal, Vol. 28, No. 1, pp. 33-49, January 1996.
- [6.8] James D., "An object oriented approach to CAD tool control," IEEE Trans. on Computer-Aided Design Journal, Vol. 10, No. 6, pp. 698-713, June 1991.
- [6.9] Pamela Z., L., "Methodologies in user interface design," IEEE ASE system magazine, pp. 14-20, August 1995.
- [6.10] Jeng L. G., et al. " Graphical block-diagram based programming environment for a DSP silicon," IEE Proceedings-G, Vol. 140, No. 5, pp. 313-318, October 1993.
- [6.11] Love D. M., "The design of a computerised component coding and classification systems for production applications," Computer-Aided Production Engineering, International conference, Edinburgh, April 1986.
- [6.12] Alan S., F., "CASE: Using software development tools," Second edition, ISBN 0-471-53042-5, 1991.
- [6.13] Ouslis C., and Sedra S., "Designing custom filters," IEEE Circuit and Devices Journal, Vol. 11, No. 3, pp. 29-37, May 1995.
- [6.15] Hardwick H. et al., "ROSE and CHIDE: user interface management system implementation as object-oriented database system application," Computer-Aided Design Journal , Vol. 22, No. 8, pp. 480-488, October 1990.
- [6.16] Louis G., and Sandra J., "Drafting for electronics," Second edition, ISBN 0-02-367342-7, 1993.
- [6.17] Allan F., "Powerful ESP," CAD/CAM Journal, Vol. 11, No. 7, pp. 28-31, July./August 1992.

# 7 Software performance and Evaluation

## 7.1 INTRODUCTION

This chapter demonstrates the software performance of the integrated design environment. The software described was designed to combine the flexibility essential for a design environment dealing with the integration of electro-mechanical systems. The tools developed have also been integrated with several commercial analysis tools to achieve a flexible CAE system which allows further adaptations and expansion of capabilities. This was done by interfacing PSPICE, database, and schematics capture facilities through a unified user interface. The examples used to illustrate functionality, are truly coupled problems which require iterative electrical/mechanical analysis. The results presented are compared against actual experimental performance data obtained on actual equipment.

## 7.2 PERFORMANCE EVALUATION

The purpose of this evaluation is to investigate the "performance" of the software, viz a viz.

- The creation and editing of 2D schematics.
- The ability of drawing the 3D models.
- The integration between the different workspace environments through a unified user interface.
- The ability of extracting design data that is used for simulation purposes (i.e netlist files, data models).

- Integration with a common database dbase(iv).
- Interfacing with PSPICE package for circuit simulation.
- The ability of extracting design information. This includes reports generation and data different queries.

### 7.3 A DESIGN EXAMPLE

To demonstrate the principles of integrating the design of different environments, a design example is selected to perform and evaluate the software.

The design operation starts with default interface settings. Its purpose is to provide feedback information about software functions and activities. The integration process has been divided into two main aspects, namely, electrical and mechanical. The application was selected to explore the integration of coupled design problem.

Electrical design describes the design of electrical circuit presented in 2D schematic form, which, details all components and connectivity. In addition, tools such as connectivity checking and nodes connection numbering have been demonstrated and the final netlist files are generated for PSPICE simulation. This section describes the relevant use of these tools to electrical designs and their applications.

#### 7.3.1 Circuit General Description

The example circuit, shown in Figure 7.1, is a part of laser charging pulser adopted from McDonald [7.1], and designed specifically for charging an excimer laser pulser forming line (PFL). The system is designed to produce a positive or negative resonant charging waveform up to a peak voltage of 20 kV in a cycle duration in the range of  $1\mu s$  to  $10\mu s$ , where the maximum pulse repetition frequency is 2kHz. Table [7.1] summarises the circuit input parameters.

Table [7.1]. Circuit general parameters.

Parameter	Name	Value	Units
Peak voltage	V2	20	kV
Charge cycle duration	$\tau$	1-10	$\mu s$
Maximum repetition frequency	F	2	kHz



### 7.3.2 2D Circuit Representation

Figure 7.1 shows the complete charging circuit drawing in the system schematic capture environment. The test circuit used to obtain the results is shown schematically in Figure 7.2. It has been constructed using the designed graphical libraries symbols (explained in Chapter 6).

#### Description of operation:

The intermediate capacitor denoted by  $C1$  is command charged to double the DC voltage via the inductors  $L1$  and  $L2$  (i.e. 20 kV from a 10 kV DC supply). The duration of  $C1$  charge cycle is set by the inductor  $L1$  and  $C1$  to be  $250\mu s$ . The charging cycle of the component  $C1$  is controlled by thyatron V1 (CX1836). The second inductor,  $L2$ , is included only as a means of grounding one side of  $C1$  and its value is selected to be smaller than  $L1$  so that  $L1$  sets the first stage charge cycle duration. The inductor  $L2$  is chosen to be much larger than the charging inductor,  $L3$ , so that  $L3$  sets the PFL charge cycle duration. The output pulse is generated by the discharge of  $C1$  and thyatron V2 which is represented by the component CX1747, through the output inductor  $L3$ .

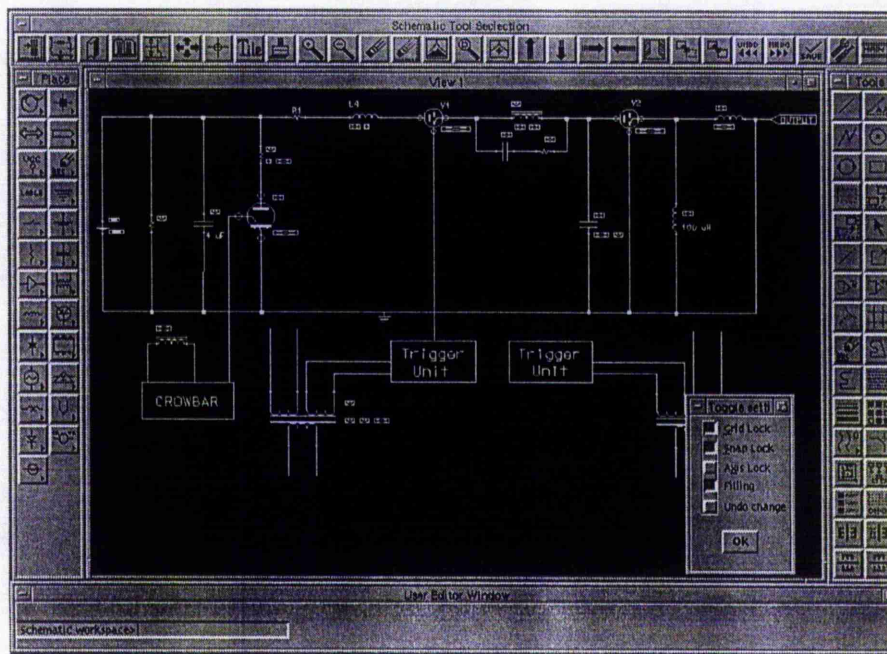


Figure 7.1 Resonant charging circuit drawing on 2D schematic environment.



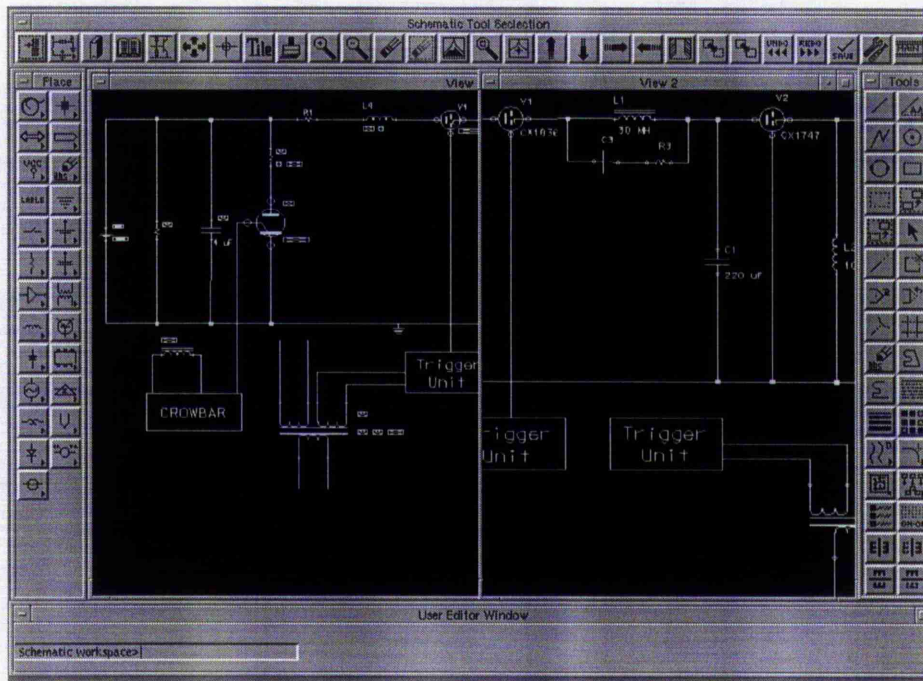


Figure 7.2 2D Schematic different views.

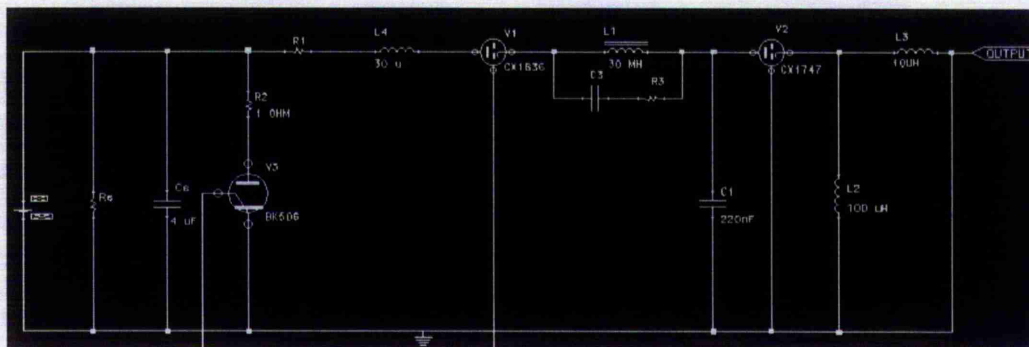


Figure 7.3 The 2D schematic of charging circuit.

## 7.4 CIRCUIT DESIGN AND ANALYSIS

The schematic circuit of Figure 7.3 satisfies in principle the requirements of the present application. A system design approach must be taken to ensure that all system components, the resonant power circuit, the HVDC supply and other control circuits are functionally compatible of particular importance is the calculation of fault current level which determine the design circuits (e.g. fault protection).

### 7.4.1 Design of First Stage

The difficulties associated with the design of command-charging stage are primarily associated with the design of the charging inductor, denoted as  $L1$ . This is because of the

high inductance value and therefore it requires a magnetic core that helps to keep the physical size down [7.1]. Figure 7.4 shows the circuit schematic of the first stage design while the schematic of Figure 7.5 shows the equivalent design model, where the capacitor C1 is charged via switch SW1, which also acts as a charging diode and provides isolation from the DC supply during the PFL charge cycle. From the circuit design specification previously described, the upper limit on the inductance  $L1$  is set by the maximum operating frequency of 2kHz and the second stage charging duration and the recovery times for thyratrons (i.e. CX1836 and CX1747).

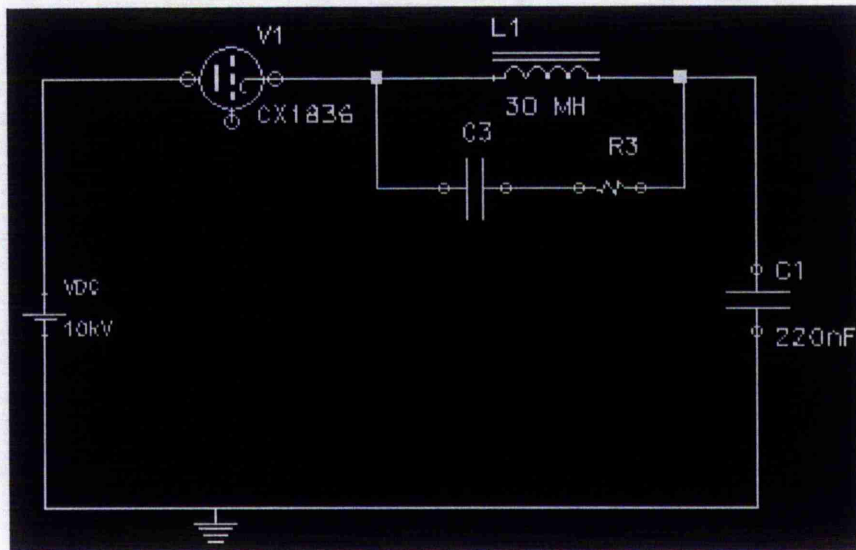


Figure 7.4 Schematic design of the first stage.

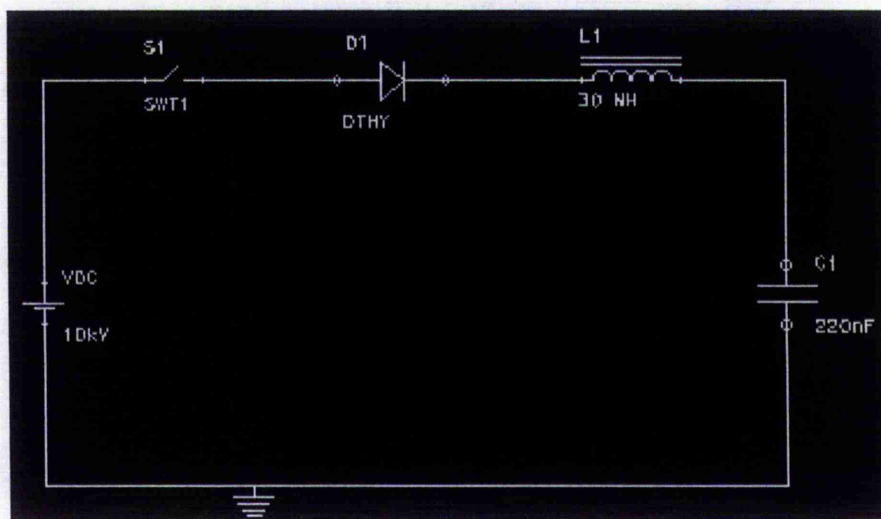


Figure 7.5 The equivalent design model of L1.

The task that it will be faced during the design is to ensure the charging inductance does not saturate during fault transient.

### Circuit Theory

The following theoretical analysis can be derived for the equivalent CLC circuit.

Assuming that  $L1$  does not saturate the default, the peak fault current would be given by:

$$\hat{I}_F = \frac{V_1(0)}{\sqrt{(L1/C1)}} \quad (1)$$

Comparing this with the normal peak current which given by:

$$\hat{I}_c = \frac{V_1(0)}{\sqrt{L1/C1}} \quad (2)$$

However, from the above two equations, it is clear that in this case the ratio of fault current equation (1) to normal charging equation (2) is simply equal to the ratio  $\sqrt{C_s/C_n}$ , Where is  $C_s$  is the smoothing capacitor and  $C_n$  is the network capacitor. A high value of smoothing capacitance, which is desired in order to minimise ripple voltage, is therefore not practicable as the peak fault currents during overloads can saturate the charging inductor.

The inductor  $L1$  is considered to be the focal point in designing this stage, as it must be designed for a constant AC inductance whilst carrying a mean DC current (i.e.  $I_c = 2C1V_{dc}$ ) for PRF of 2kHz. For the case of a gapped core with air gap of  $l_a$ , and core magnetic length of  $l_m$ , the inductance is therefore given by:

$$L = \frac{\mu_0 N^2 A_c}{l_a + \frac{l_m}{\mu_\Delta}} \quad (3)$$

where  $\mu_0 = 4 \times 10^{-7}$  is the permeability of free space,  $N$  is the number of turns,  $A_c$  is the core cross section area, and  $\mu_\Delta$  is the incremental permeability. Since that  $\mu_\Delta$  is a complex function consists of the magnitude and duty cycle of the charging current (i.e. depending on both peak amplitude and mean direct value), the inductance will also vary with mean charging current. This dependence can be eliminated if  $l_m / \mu_\Delta \ll l_a$  for grain-oriented silicon steel material (GOSS).

$$\frac{l_a}{l_m} = 0.06 \quad (4)$$

Equation (4) defines the minimum allowable air gap  $l_a$  for a given design core which will give a constant AC inductance independent of mean current. As it has been shown in Chapter 3 this condition is not always satisfied, as the increase of the gap size will increase the fringing which introduces additional losses in the coils by eddy currents. Also because of fringing, the effective air gap is smaller than the physical gap by factor calculated from.

$$F = 1 + \frac{l_a}{\sqrt{A}} \ln \left[ \frac{G}{l_a} \right] \quad (5)$$

where  $l_a$  is the design gap and  $G$  is the length of the core limb. Since that the core is gapped and relative permeability of GOSS material,  $\mu > 2000$ , most of the magnetic energy will be stored in the air gap. Therefore, the energy equation can be expressed as:

$$E = \frac{l}{2\mu_0} B^2 A_c l_a = \frac{1}{2} l \hat{I}^2 \quad (6)$$

Since, that the inductance of the gapped core is given by.

$$L = \frac{\mu_0 N^2 A_c}{l_a} \quad (7)$$

Therefore, it can be shown that the flux swing during the cycle is given by.

$$\Delta B = \frac{V_s \tau_c}{\pi N A_c} \quad (8)$$

Where  $\tau_c$  is the charging cycle duration. From equation (8) a suitable core is selected and the ratio of equation (4) is checked and in order to ensure that the constant inductance criterion.

### Theoretical calculations

To investigate and obtain the inductor  $L1$  final design parameters, the application for a  $250\mu s$  charging cycle duration was carried by assuming the following data input shown in Table [7.2].

Table [7.2]. First stage input parameters.

Parameter	Name	Value	Units
Peak voltage	V1	10	kV
Charge cycle duration	$\tau$	250	$\mu s$
Intermediate capacitor	C1	220	nF
Maximum repition frequency	F	2	kHz

From the above listed parameters values such  $L_1$  and peak current  $\hat{I}$  can be calculated as follows:

$$L_1 = \left( \frac{\tau}{\pi} \right)^2 \times \frac{1}{c} \quad (9)$$

$$= \left( \frac{250}{\pi} \right)^2 \times \frac{1}{220n} = 28.8mH$$

$$\hat{I} = \frac{V_1}{\sqrt{L/C}} = \frac{10}{\sqrt{28.8/220}} = 27.64A$$

Obviously, the above analysis gives a reasonable method with which to obtain design values for components. It is important for the designer to test his design and visualise circuit waveforms. this can be done easily by generating a circuit netlist and running this on a circuit simulator (PSPICE). the tools set developed permits partial generation of netlists via a window, which allows the designer to capture only the important elements.

#### 7.4.2 Simulation of First Stage

The basic objectives in carrying out the simulation procedure is to optimise circuit performance. Using the final extracted netlist files that represent the first stage circuit, simulation is performed in association to the inductor ( $L_1$ ), thus providing an effective visualisation capability suitable for obtaining the final design specification. The PSPICE package simulates the extracted netlist over a specified range of time (i.e. charging time) giving an output performances. Because of the need to produce an accurate design specification for the components involved, a repeated simulation procedures are normally required: PSPICE permits full Monte Carlo analysis and tolerance checking. Figure 7.6 shows the netlist generated for the first stage.



```
* Design: d:\ustation\dgn\default\charging.dgn
* This is Netlist File For PSPICE
* Date: 8/5/1997 Time: 14:32:45.91
Vin 1 0 pulse (0 10kV)
S1 1 2 100 0 swt1
D1 2 3 DTHY
L1 3 4 30mH
C1 4 0 220nF
Vc1 100 0 pulse (0 100ns)
.tran 10.0u 250.0u
.probe
.MODEL DTHY D[IS=1E-9 RS=0.001]
.MODEL SWT1 VSWITCH[RON=0.1 ROFF=1E12]
.END
```

Figure 7.6 Netlist file generated for the first stage.

Figure 7.7 shows screen dump of a simulation session with the 2D schematic editor and PSPICE probe display.

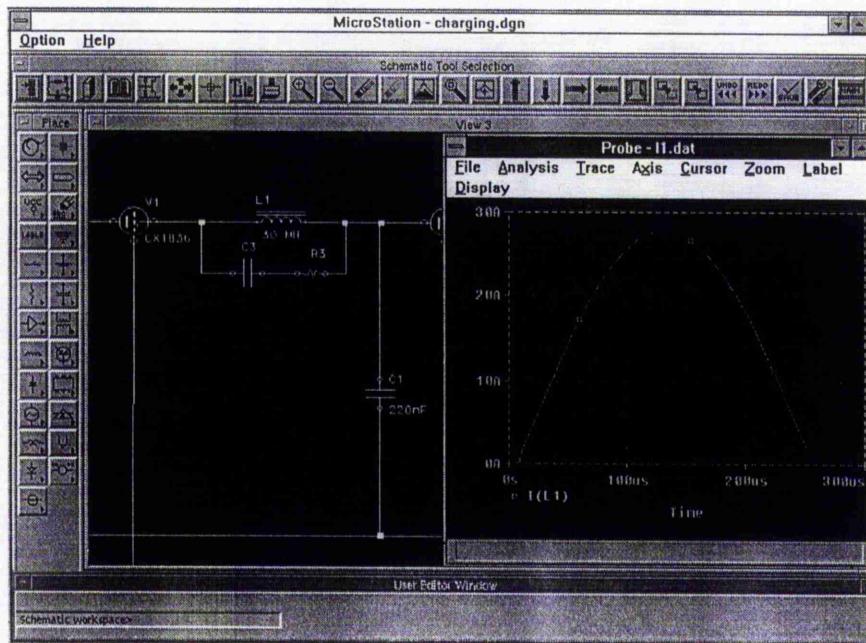


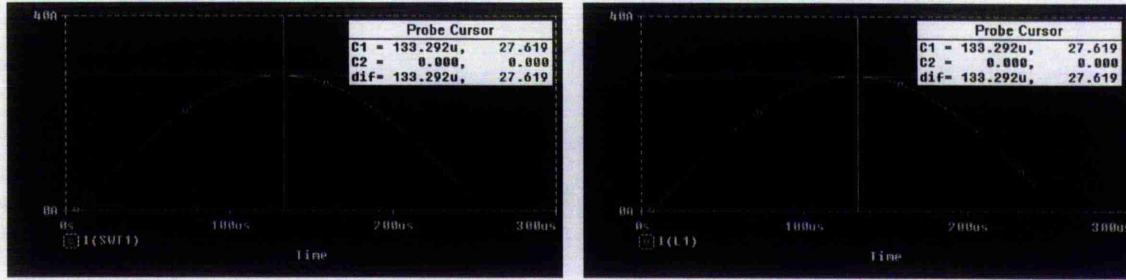
Figure 7.7 A display of the 2D schematic environment and PSPICE.

Different design parameters have been computed using the components modelling procedure previously described (refer to Chapter 3). These parameters include the obtained values, therefore decisions had to be made by a designer whether to continue proceeding an iterative process or terminating in case of reaching the desired solution. In addition, these values were checked against the results obtained by the modelling of each



component and traces of waveforms. Close agreement was found, confirming the validity of the integrated method investigated as part of this research.

The following figures are simulation results obtained using PSPICE. Figures 7.8a, 7.8b show the peak current across both component swt1 and inductor  $L_1$  confirming the result obtained by equation (1) which found to be 27.619A comparing to 27.64A by calculation.



(a)

(b)

Figure 7.8 Peak current  $\hat{I}$  through swt1 and  $L_1$

From design specification quoted in Table [7.1], it can be seen that the charging inductor is connected to voltage source of 10kV with charging period of  $250\mu s$ . Therefore, the output can be expressed as ( $V_o = 2V_{dc}$ ) for resonant charging which found to 19.972kV. Figure 7.9 shows the simulation result of the output voltage across the inductor  $L_1$ .

Figure 7.10 shows output waveform of the magnetic energy stored in the air gap of  $L_1$  confirming the result of equation (6) which found to be 10.985 J.

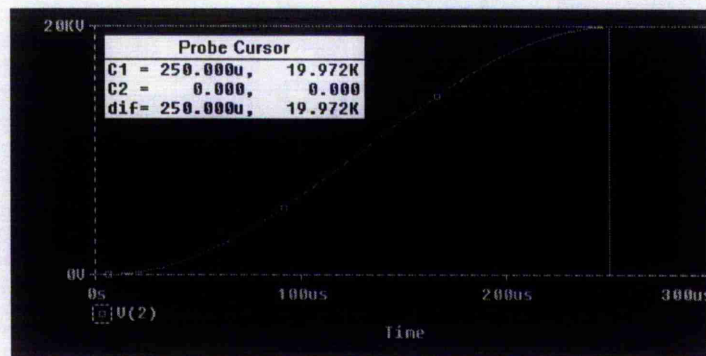
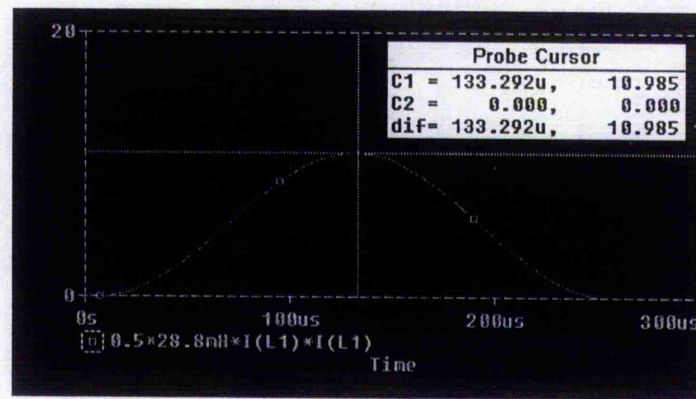


Figure 7.9 Output voltage  $V(0)$  through inductor  $L_1$ .

Figure 7.10 Energy stored in  $L_1$ .

### 7.4.3 Design Optimisation and Performance

Inductor  $L_1$  is the key component in designing the first stage of this circuit, as it has been mentioned in Section 7.5 of this chapter that the upper limit on the inductance  $L_1$  is set by the maximum PRF of 2kHz. With reference to the experimental data it was noticed that the maximum allowable value for  $L_1$  is approximately 74mH comparing to the initial calculated value of 28.8 mH.

### 7.4.4 Design of Second Stage

As previously mentioned, the purpose of designing the second stage is that it must resonantly charge the PFL to a maximum voltage of 20 kV in a period set by the inductor  $L3$ , as shown in Figure 7.11, while Figure 7.12 shows the equivalent design model of  $L3$ . For the case of  $C1 = Cn = 220\text{nF}$ , inductor  $L3$  will vary over the range  $1\mu\text{H}$  to  $10\mu\text{H}$ . The required range of charging period is from  $1\mu\text{s}$  to  $3.3\mu\text{s}$ . This suggests that several fixed, interchangeable, inductors will be required to cover the desired range of charging period.

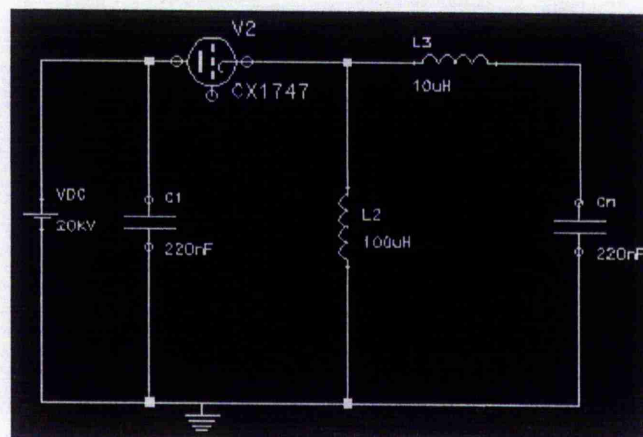


Figure 7.11 Schematic design of the second stage.



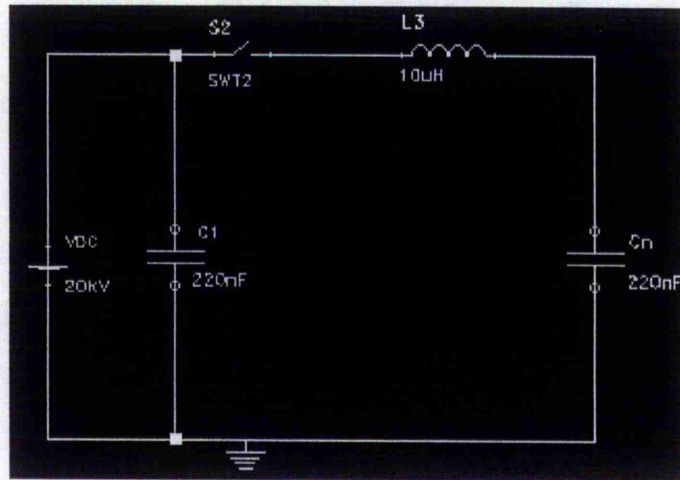


Figure 7.12 The equivalent design model of L3.

Under normal operation its likely that some residual voltage will remain on the PFL at the end of the short discharge pulse and that this charge will try to ring back through the CX1836 thyatron.

### Theoretical calculations

The given value of  $L3 = 10\mu H$ , therefore the charge cycle duration can be calculated as:

$$\begin{aligned}\tau &= \pi\sqrt{L \times C} \\ &= \pi\sqrt{10\mu \times 110n} = 3.295\mu s\end{aligned}$$

Since that the peak voltage is 20kV through the inductor  $L3$ , then the peak current can be calculated using equation (2).

$$\hat{I} = \frac{20 \times 10^3}{\sqrt{10\mu/110n}} = 2097.6A$$

Assuming no reverse conduction. The calculation are repeated and compared at both values of  $L3$  given the summarised results shown in Table [7.3].

Table [7.3]. Second stage input parameters.

Parameter	Name	Value		Units
Charge cycle duration	$\tau$	3.295us	1.042us	$\mu s$
Peak charging current	I	2.098	6.633	kA

The operating parameters of the second stage will be somewhat different to those quoted in Table [7.4], this is due to the thyatron losses and other circuit component losses.

Table [7.4]. Second stage design parameters.

Parameter	Name	Value		Units
Peak PFL voltage	V2	20	20	kV
Charge cycle duration	$\tau$	3.3	1	$\mu s$
Inductance	L3	10	1	$\mu H$
Peak charging current		2.098	6.268	kA
Thratron max di/dt		2	20	kA/ $\mu s$
Mean thrytron current		8.80	8.80	A

Table [7.5] details the revised circuit parameters when losses and the effect of the inductor  $L2$  are taken into account. The results are of a peak PFL charge voltage of 20kV and assume a circuit resistance of 0.1. It is also assumed that the PFL is instantaneously discharged at the peak of the charging waveform and that there is no reserve current in CX1747 component. The design of the second stage basically focus on the design of two components (i.e.  $L2$  and  $L3$ ). These were constructed from spiral winding of copper foil [7.10]. Therefore the inductance of this configuration can be calculated form.

$$L = \phi d N^2 \quad (\mu H) \quad (10)$$

Where the mean diameter,  $d$ , is in units of meters and

$$\phi = \frac{0.1 \times \pi^2}{0.45 + \alpha + \rho + \frac{2}{3} \alpha \rho \frac{(\alpha + 1)}{(\alpha + 2)}} \quad (11)$$

and the value of  $(\alpha, \rho, d)$  can be calculated from.

$$\alpha = \frac{1}{d} \quad \rho = \frac{c}{d} \quad d = \frac{do + di}{2} \quad (12)$$

In this case only small number of turns are required to provide the low inductance for  $L3$ . This is designed by selecting a suitable number of turns and therefore calculating the required diameter. Figure 7.13 provides the selecting graph for inductance versus former diameter and the equivalent number of turns. It also shows the results of such a computation over which the component  $L3$  must be varied, a former approximately 100mm diameter is satisfactory with a corresponding range of turns from 3 to 10. A

former of 110mm diameter was used as this gave almost the exact value of  $1\mu H$  and  $10\mu H$  with number of turns 3 and 9 respectively.

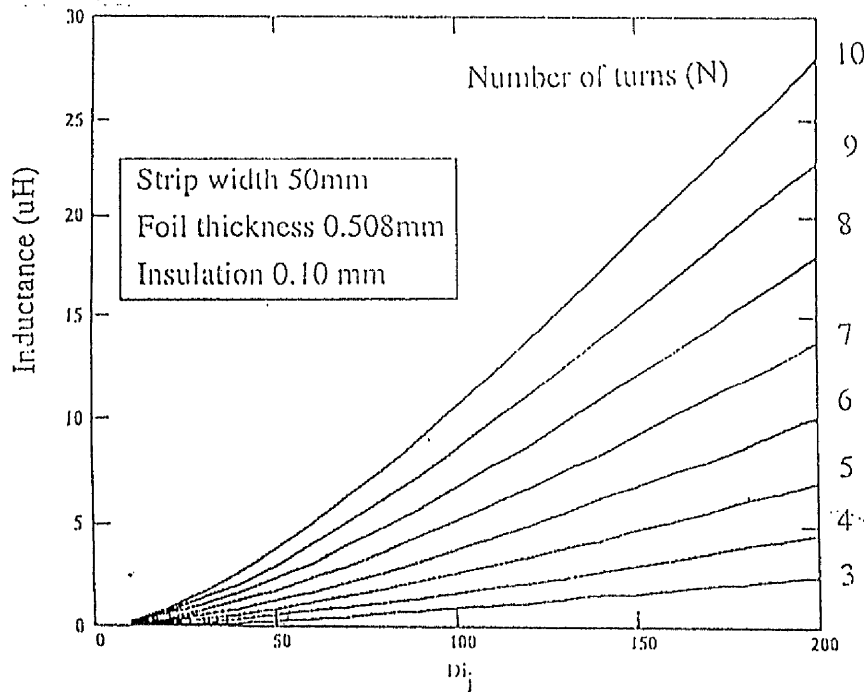


Figure 7.13 Graph of inductance versus former diameter and number of turns.

#### 7.4.5 Simulation of Second Stage

The design of the second stage focus on the inductor ( $L_3$ ) which was fabricated as spiral inductor. Figure 7.14 shows the extracted netlist for the second stage. Using PSPICE, the figures which follow show the measured and simulated results by varying the value of  $L_3$  over the range of  $1\mu H$  to  $10\mu H$  with required period time of  $1\mu s$  to  $3.3\mu s$ . The simulation results were obtained for two different rang of time.

```
* Design: d:\ustation\dgn\default\charging.dgn
* This is the Netlist File For PSPICE
* Date: 8/5/1997 Time:14:32:45.91
Vdc 1 0 pulse (0 20kV)
C1 0 1 220nF
* swt1 1 2 100 0 swt1
L3 1 2 1uH
Cn 2 0 220nF
.tran .3u 3.3u
.probe
.MODEL SWT1 VSWITCH[RON=0.1 ROFF=1E12]
.END
```

Figure 7.14 Netlist generated for the second stage.

Considering the value of  $L_3 = 10\mu H$ , Figures (7.15, 7.16) show the peak current and the output voltage through the inductor  $L_3$ .

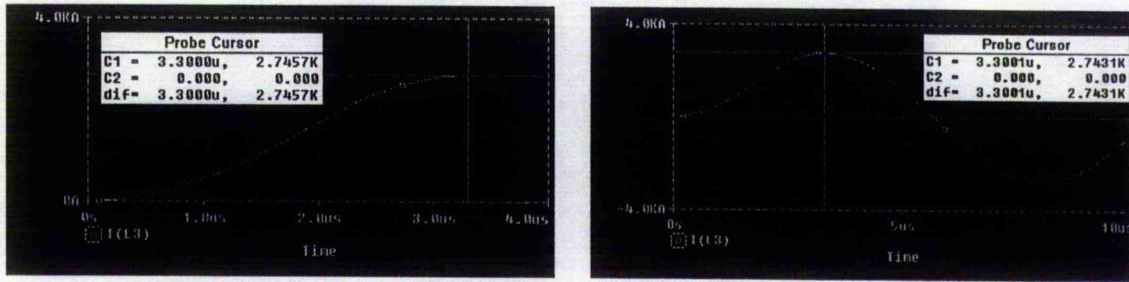


Figure 7.15 Peak current  $\hat{I}$  when  $L_3=10\mu H$ .

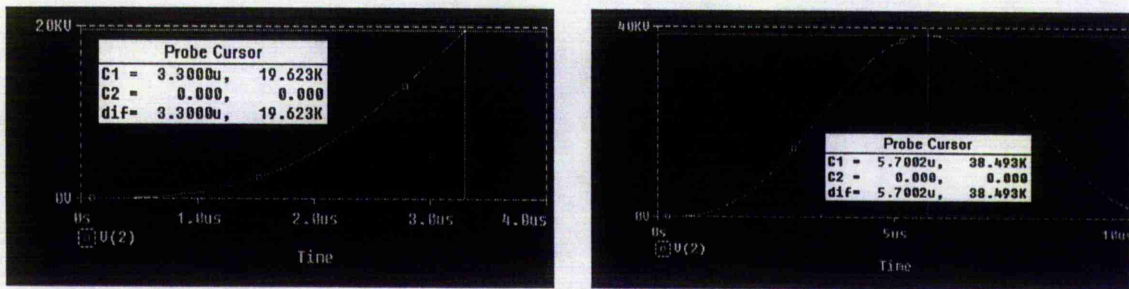


Figure 7.16 Output voltage ( $V_0$ ) when  $L_3=10\mu H$ .

The value of  $L_3$  was decreased to  $1\mu H$ . Figures (7.17, 7.18) is the obtained results for this value for both the peak current and the output voltage respectively.

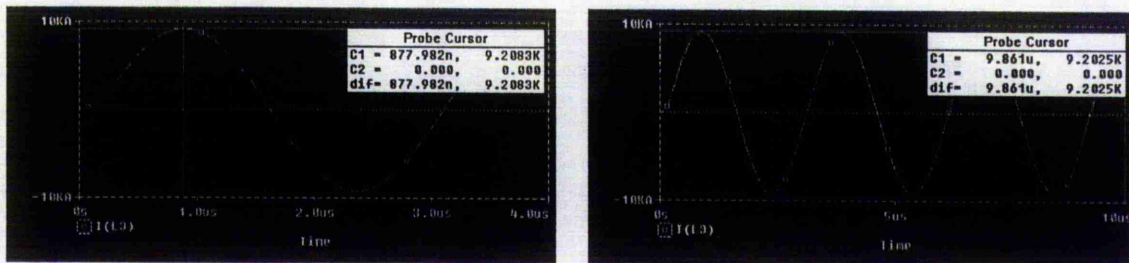
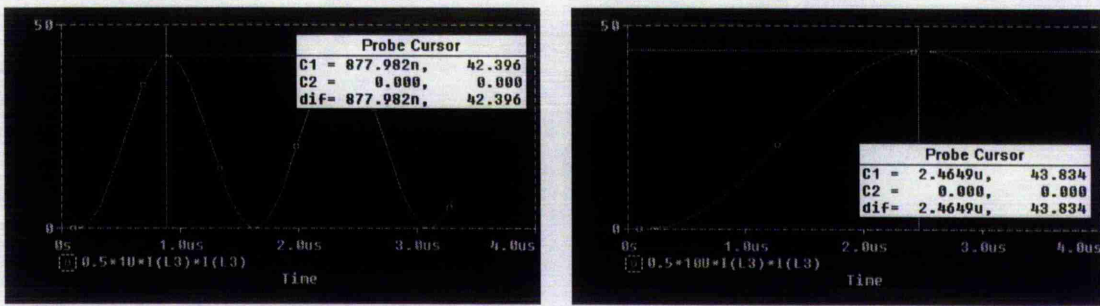


Figure 7.17 Peak current  $\hat{I}$  when  $L_3=1\mu H$ .



Figure 7.18 Output voltage ( $V_0$ )  $L_3=1\mu H$ .



Figure 7.19 Energy stored in ( $L_3$ ) when  $L_3 = 10\mu\text{H}$  and  $1\mu\text{H}$ 

## 7.5 MECHANICAL DESIGN

Mechanical design refers to the way in which the used components are arranged. In this design for example, the main high voltage circuit is contained within an oil tank with transformer oil used to insulate the components and to provide cooling for the thyristors and power components. With reference to 2D drawing shown in Figure 7.3, all components to the right of resistor R1 are contained in the oil tank.

This section highlights the use of mechanical design function and its aspects in relation to electrical design. This can be achieved by describing design components and their representation as 3D models. Photographs are shown to provide a clear idea of the system mechanical arrangement and the wiring between the components.

### 7.5.1 Components Design Description

Description to these components highlights the purpose of their use and their electrical and mechanical properties is explained as follows:

- Inductor L1: Designed to act as a charging inductance for the first stage of the resonant circuit. The value that it can hold can be varied depending on the cycle charging time. As it is important to consider the prospective fault currents which can develop and the requirements for protection of circuit components, the prospect of designing L1 holds a significant key issue in obtaining the desirable outcome of the first stage. Figure 7.20 shown below displays a 3D model of inductor L1.

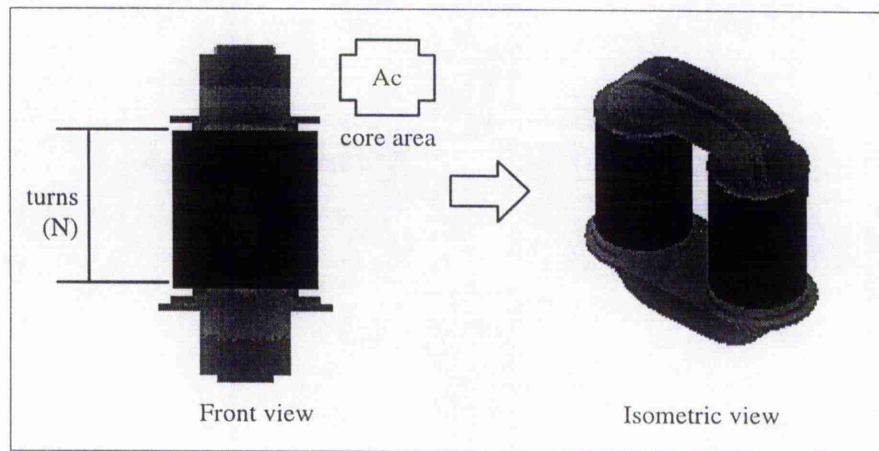


Figure 7.20 Inductor L1 design.

- Inductor L2: Is considered to be the focal point in the design of the second stage, it is included only as means of grounding one side of  $C1$  and is chosen smaller than  $L1$ , so that control can be taken towards the setting of the cycle duration of the first stage.
- Inductor L3: Since that the second stage must resonantly charge the PFL to maximum voltage of 20kV, its role to set a charging period ranging from  $1.0\mu s$  to  $3.3\mu s$  as its value will vary over the range  $1\mu H$  to  $10\mu H$ . Figure 7.21 shows a display of 3D model of (L2, L3) spiral inductors.

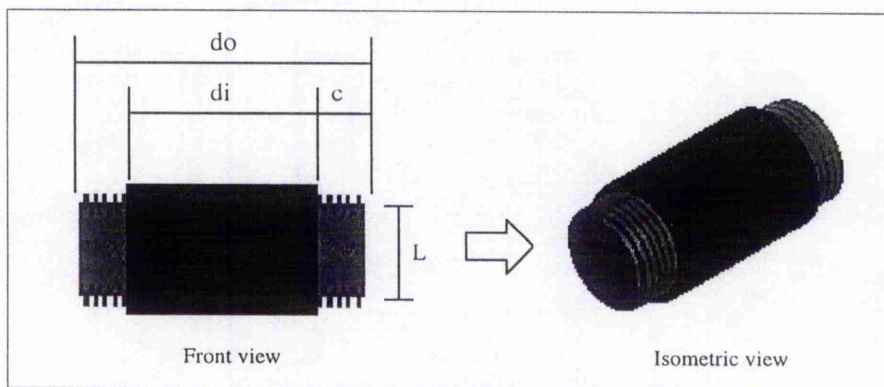


Figure 7.21 Spiral inductors (L2, L3) design.

- Capacitor C1: It consists of 60 TDK ceramic capacitors in which they are assembled of 30 in parallel. Its role in this design is to act as a charging diode and to provide a separation from the DC supply during the PFL (i.e. second stage) charge cycle. Its value is chosen to be equal to the PFL capacitance so that all of the charge stored will be transferred to the PFL in a specific time. Figure 7.22 shows a 3D display of capacitor (C1).



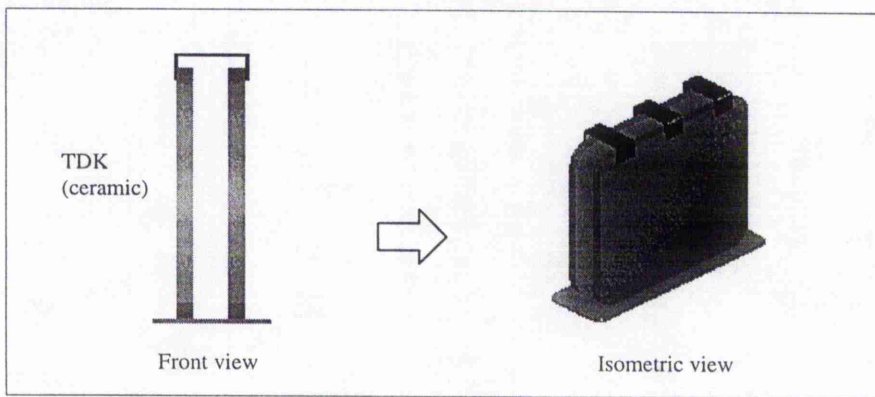


Figure 7.22 Capacitor (C1) design.

- CX1836: It is designed for air-cooled operation. One of its contents is a screen grid which provides a high level of immunity to spurious triggering. This kind of thyatron are suited for operation in circuits as they are required to fire independently. Figure 7.23 shows a 3D display of the CX1836 component.

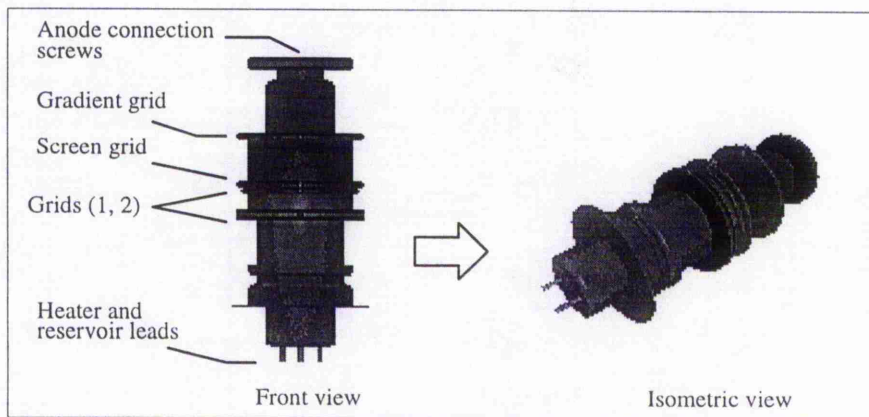


Figure 7.23 Thyatron CX1836 design.

- CX1747: It is designed to be suitable for switching high peak and average power at high pulse repetition rates with reserve current conduction capability. To maximise the gas pressure consistent with the required voltage, a reservoir operating from a separate supply is incorporated and resistor box settings can be adjusted within the specified limits. Figure 7.24 shows a 3D model of CX1747 component.

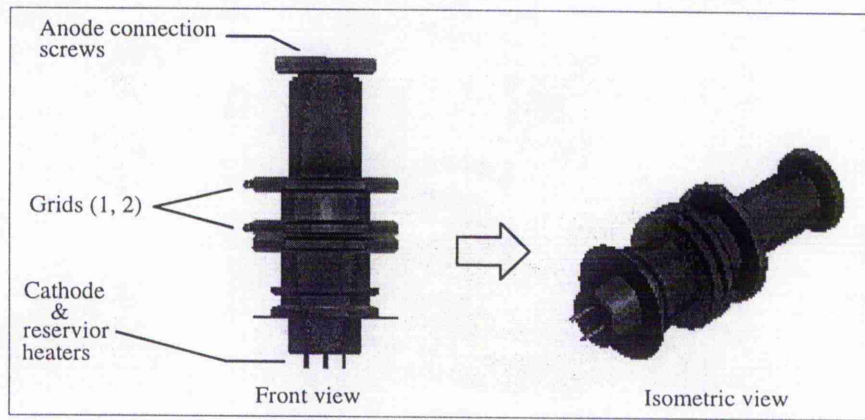


Figure 7.24 Thyatron CX1747 design.

The components used in constructing the 3D model are basically 3D library cells, and they can be loaded, modified and maintained as desired. Figure 7.25 shows the 3D components stored as library cells.

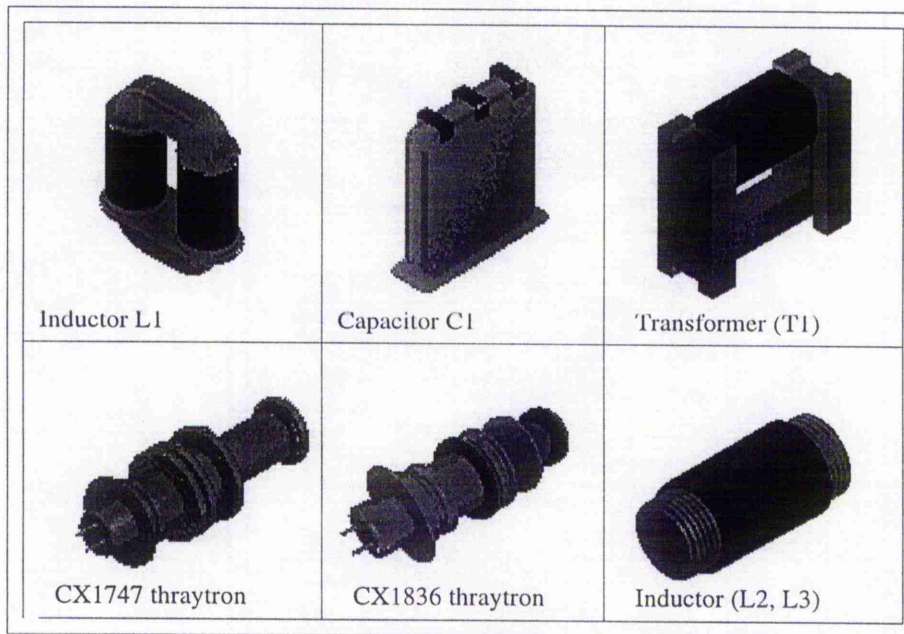


Figure 7.25 3D model components stored as library cells.

### 7.5.2 3D Layout Representation

Chapter 6 has described the method of creating 3D models which involves the settings of workspace and loading of 3D libraries cells. These cells which represent the circuit components were assembled into a complete system model. In this interaction process different colours were used for the inductors to represent the winding. Figure (7.26, 7.27) illustrate the complete 3D assembly model presented in different views drawn



within the 3D environment and by the assistance of the developed tools. The electrical connections between the components are shown. The involved components are arranged in a way that they can easily be wired. For example, Both the CX1747 and CX1836 components are positioned close to the isolating transformers which supply heater/reservoir current (and trigger circuit power).

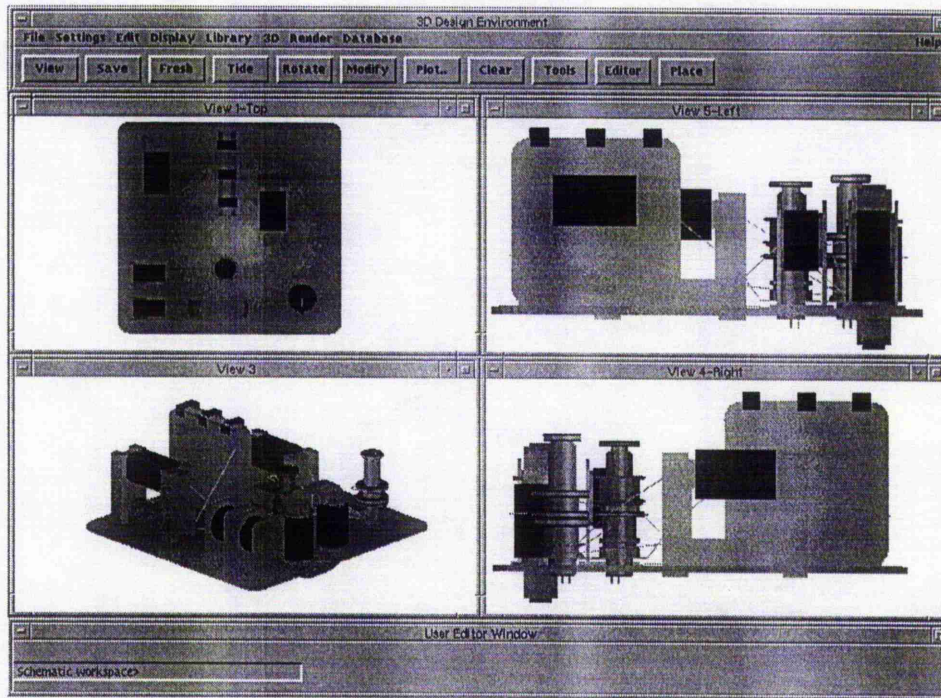


Figure 7.26 3D different views representation.

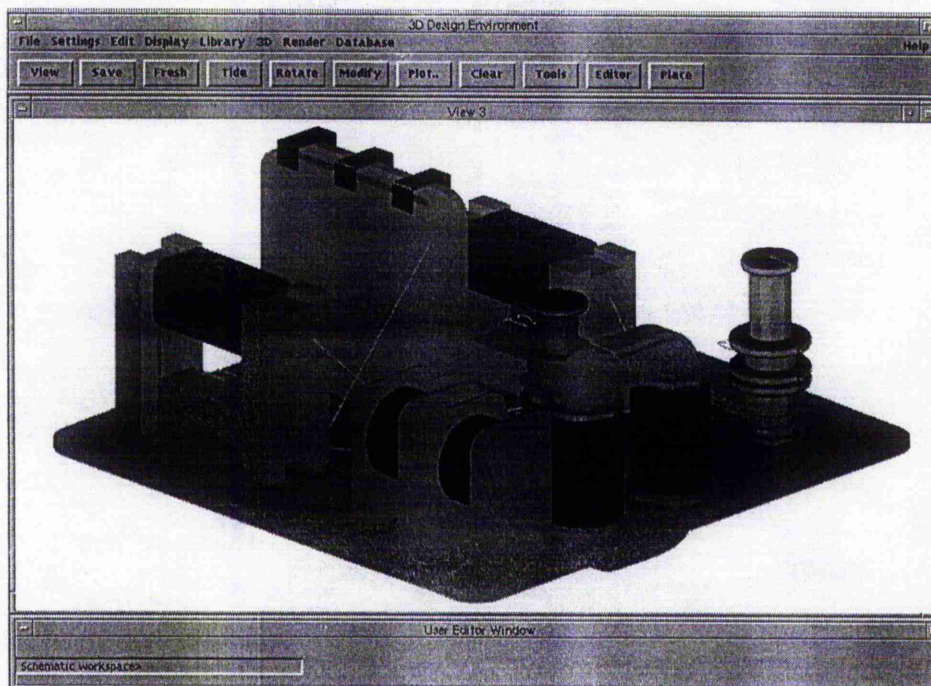
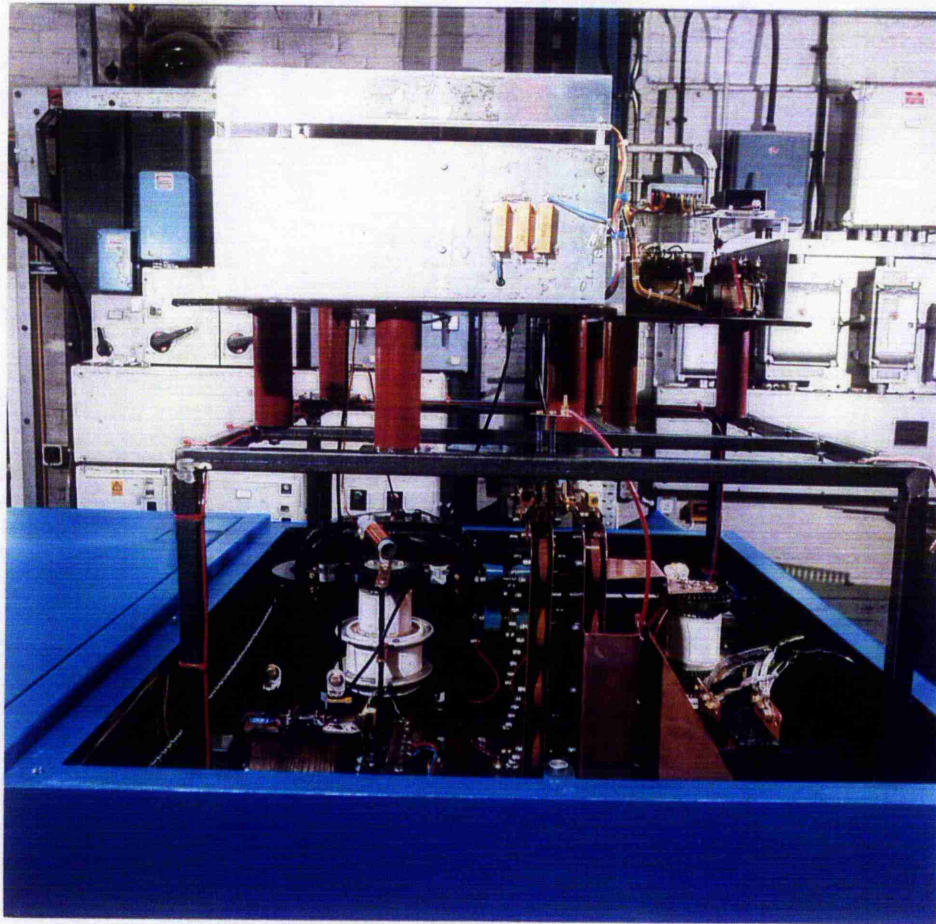


Figure 7.27 Isometric representation of the 3D model.



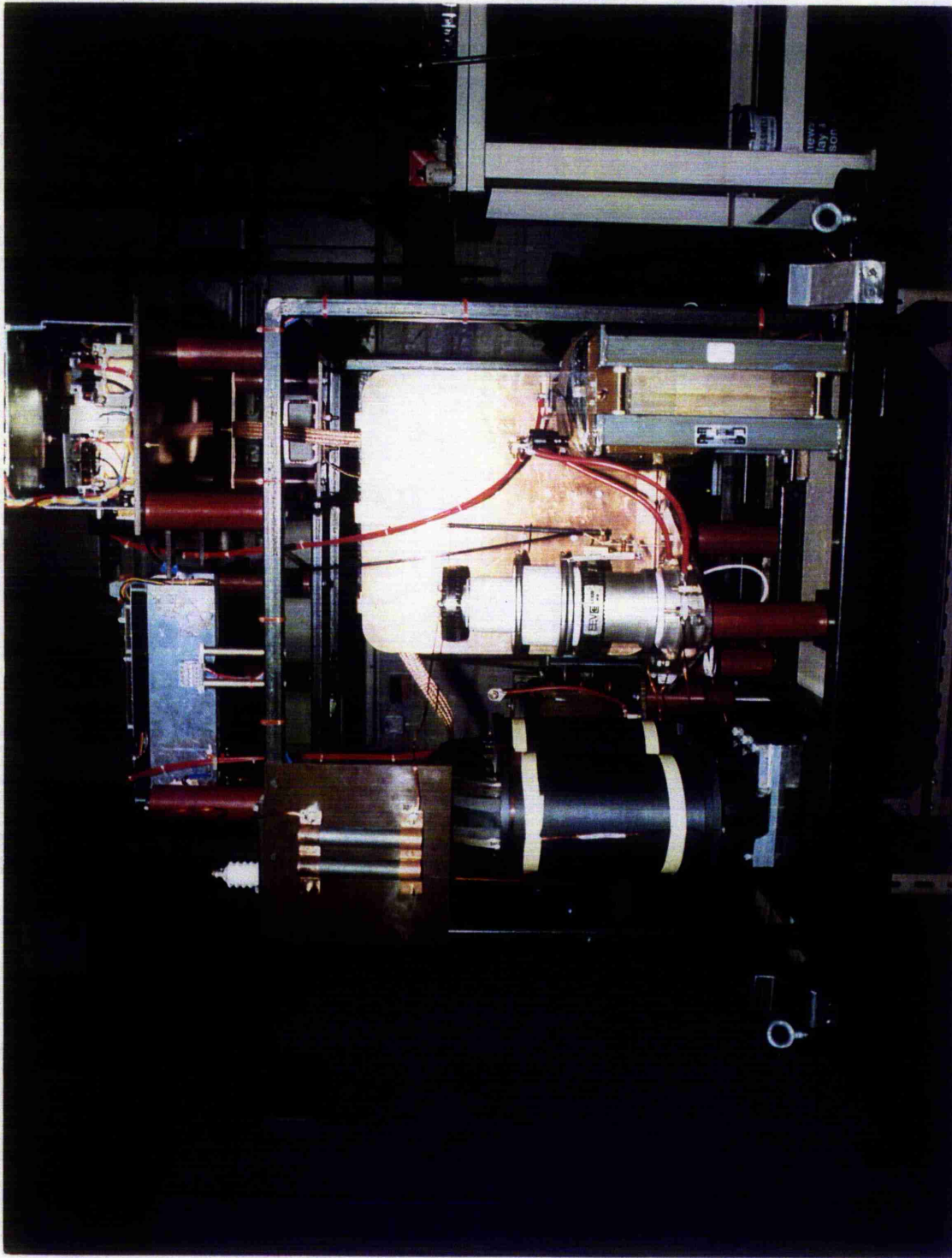
### 7.5.3 The Assembly Wiring Connections

With reference to photograph 7.1 and the 2D schematic diagram of Figure 7.3. The wiring between the components is represented as line string-type. The photograph shows the side view of the system where the component CX1836 is the centre connection between (L1) and the transformer. From the component recommendation sheet, it is recommended that separate supplies be used for cathode and reservoir heaters. A common transformer was used and the reservoir lead was connected to the mounting flange. As the regulation suggest that care should be taken to ensure excessive voltages are not applied to the reservoir heater circuit from the cathode heater, this is because of the high impedance cathode heater connections.

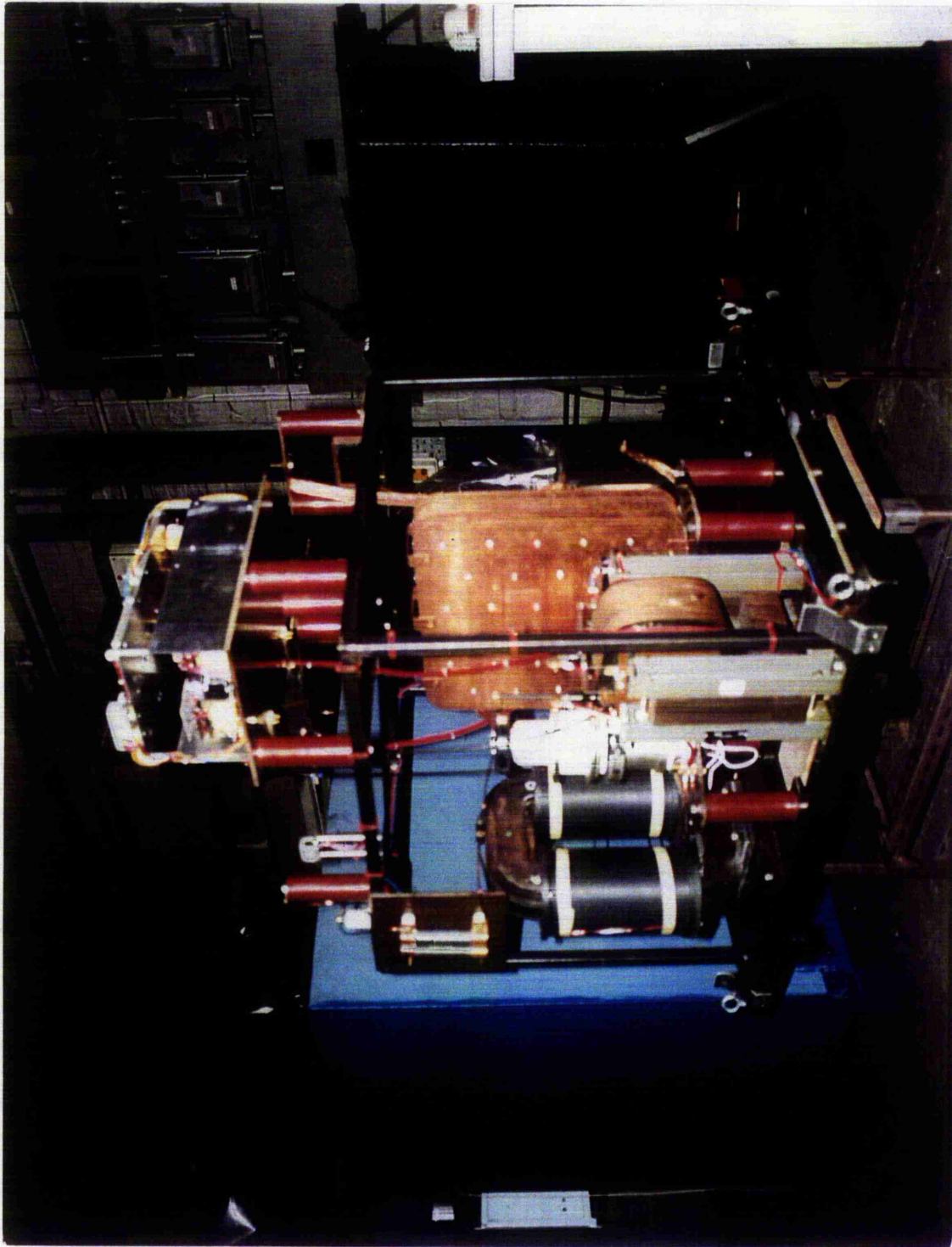


Photograph (7.1). Laser charging pulser.





Photograph 7.2 Laser charging pulser (side view)



Photograph 7.3 Laser charging pulser (isometric view)



## 7.6 ENVIRONMENT INTEGRATION

The illustration of Figure 7.28 shows the system's environments integration concept.

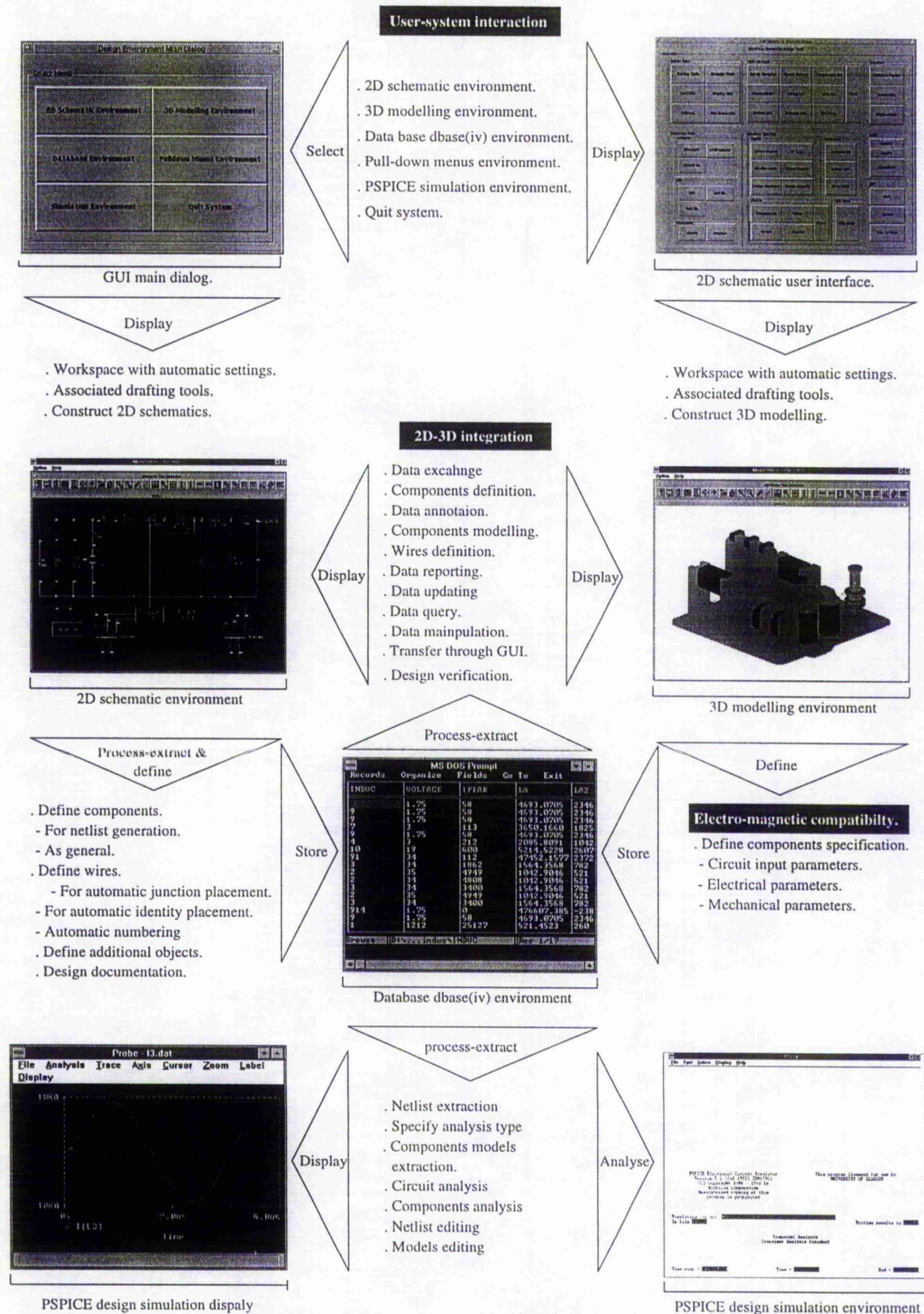


Figure 7.28 Illustration of environments integration.



### 7.6.1 The Interface between 2D and 3D layouts

Once the construction of the 2D schematics is complete, the process of interfacing between the two layouts takes place. It involves the conversion of 2D schematics into 3D format so that, they can be merged into a single design file or they are designed as separate files. In either case, the database is considered as a common element through which the designer interacts using components modelling templates.

The 3D models are updated using another developed function which basically scans through database table that contains components modelling parameters and annotates the computed values into the 3D layout. In this case, for example, the number of winding turns denoted by ( $N$ ) computed using design templates for the inductor L1 is updated and by executing update function, the new value is display on the 3D model. This suggests that using the database in this sense is another significant benefit as it establishes a direct communication between the two environments. Figure 7.29 illustrates the interface between 2D and 3D through common database.

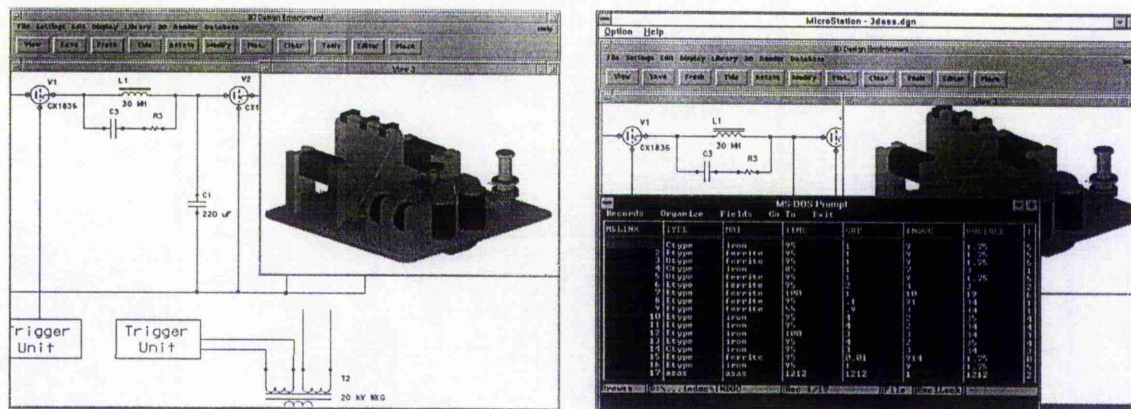


Figure 7.29 The interface between 2D and 3D using common database.

### 7.6.2 Components Modelling Procedure

The key issue in modelling the circuit components is the relationships that contain electrical and mechanical functions. These functions are expressed as mathematical equations that need to be solved by means of software programming routines (explained in Chapter 3). Design editors representing modelling templates were used to provide a way of obtaining model data files that can be used for simulation with PSPICE package. The templates have been adopted to allow reusability since they can be used to modify existing designs to achieve better performance and functionality. Each template requires design information which processed by programming tools using design editors. These editors were tested using the UCMs programming language and the database. As they

typically contain a set of input parameters through which the user interacts, an equivalent output parameters are therefore, calculated and stored in the database for further use (e.g. extracting the design final model).

Component model parameters are extracted from the database using MDL programs. The solving of these parameters based on the mathematical theory illustrated in Section 7.5. Considering the design components described previously, the modelling procedure taken towards the extraction of their design models is outlined as follows:

- The modelling of inductor L1: The main idea of designing the inductor L1 is to extract its inductance value and other relevant design parameters. Parameters such as core type, material, charging time ( $\tau$ ), capacitor (C1), and the charging voltage ( $V_{dc}$ ) are the input parameters. They are calculated by solving the equations given the outputs parameters (number of turns ( $N$ ), air gap required ( $l_a$ ), etc.) as a result. In this case, these parameters are divided into two types: parameters that need to be included in the model data file and the parameters that represent the inductor design specification. Figure 7.30 shows the dialog editor used for modelling the inductor L1 within 2D schematic environment.

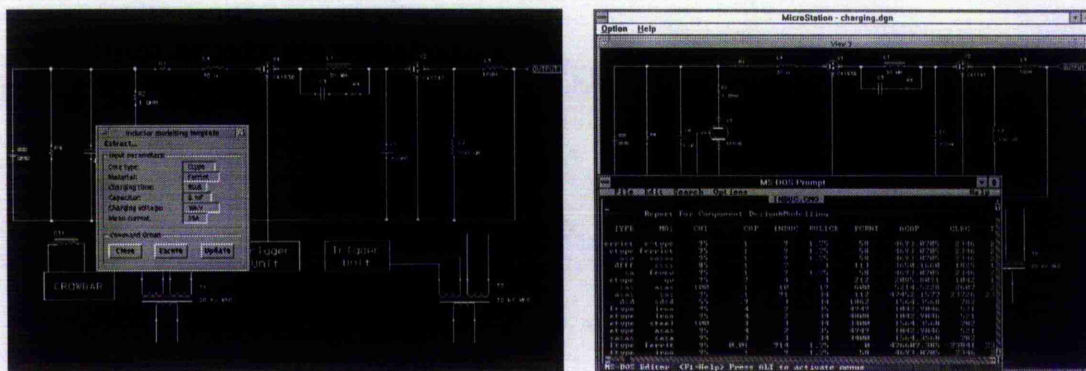


Figure 7.30 Dialog editor and L1 extracted model.

### 7.6.3 Integration with the Common Database

Generally, the main aim is to investigate the use of the database tools developed and fully described in Chapter 5. Therefore, it should provide meaningful explanation of the following features:

- The ability to integrate between the different design environments by means of interfacing with the database, thus, the flexibility of sharing and accessing the stored data can be achieved.



- The ability to integrate design data from a number of applications, with control across to these data, and also to ensure its integrity and security.
- The ability of the user to place additional information into data designed tables either using dialog editing or automatic extraction tools. Thus, this information can be retrieved and reviewed at any time.
- The ability to manipulate design data by the use of update, delete, and drop-link facilities.

Chapter 5 has mentioned that due to the database open structure, it allows the integration of data generated by the design tools. Based on these data, the database tools permit link creation as the initial step. This facility was used to establish the automatic linkage between both 2D and 3D workspaces, and also to assist the generation of the design different documents, which can be seen in Section 7.8. Using the resonant circuit previously described, this can be demonstrated by looking at the design different environments where various design issues were considered and different developed tools were tested. These issues are illustrated as follows:

- The integration with the schematic environment:  
This issue provides an illustration to show the database tools facilities operate. As it has been described in Chapter 5, database tools are classified into three main aspects (i.e. report, update, and manipulate). Further, each of these contain several tools and functions. Using the 2D schematic drawing of Figure 7.3, the design tools were used for various design tasks. These include components, wires, and circuit data definition using dialog editors, where the user interacts through elements recognition program (refer to Chapter 5), data updating either using dialog editors or fence defined, and reports generation and data different queries. Figure 7.31 shows component data editing dialog with the corresponding data retrieval using SQL dialog.

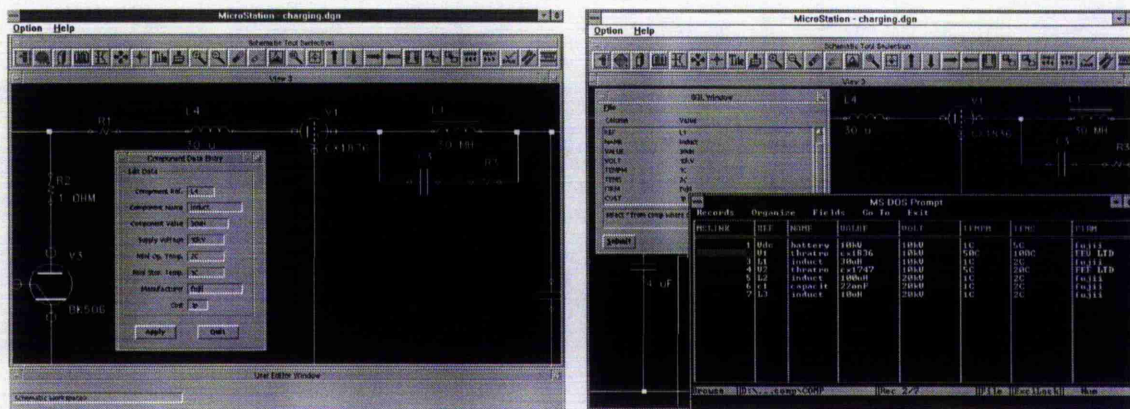


Figure 7.31 Components data editing and data retrieval.



Figure 7.32 is illustration of data updating dialog used to update components data which are already stored in the database. Here, for example, is used to update design data for the inductor  $L_1$ . Data can also be Updated using a user defined fence, which in this case data are automatically updated.

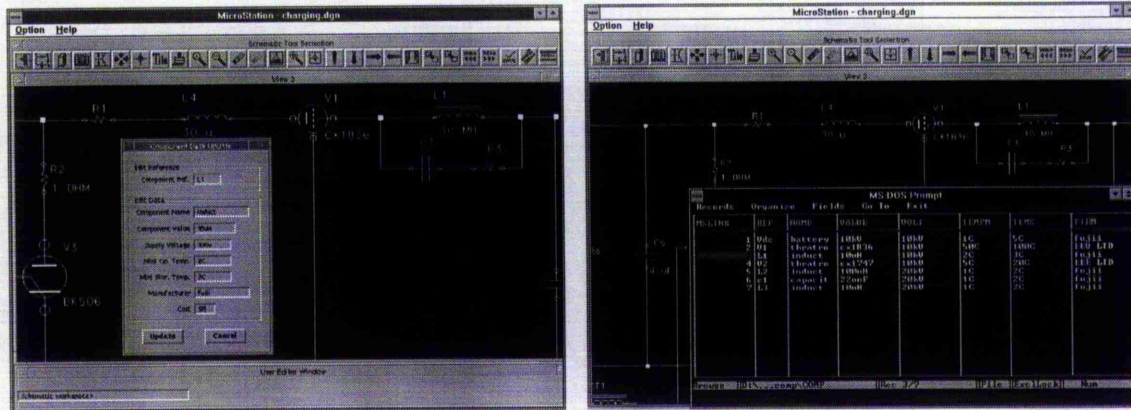


Figure 7.32 Components data updating.

## (2) The integration with the 3D modelling environment:

As previously explained that database played an important role in interfacing between 2D and 3D modelling environment. In this case, the procedure of integrating database with 3D environment was through 2D schematics design where the components are modelled using design template (explained above) and data are therefore stored and used for further use as for example, updating 3D models and generate complete modelling reports, as shown in Figure 7.32. Figure 7.33 shows stored modelling data for inductor  $L_1$  after it has been processed using the modelling procedure.



Figure 7.33 Display of inductor database storage.

## (3) The integration with PSPICE environment:

Database is integrated with the simulation package (PSPICE) through the storage of netlist information and design models. As previously mentioned these information are







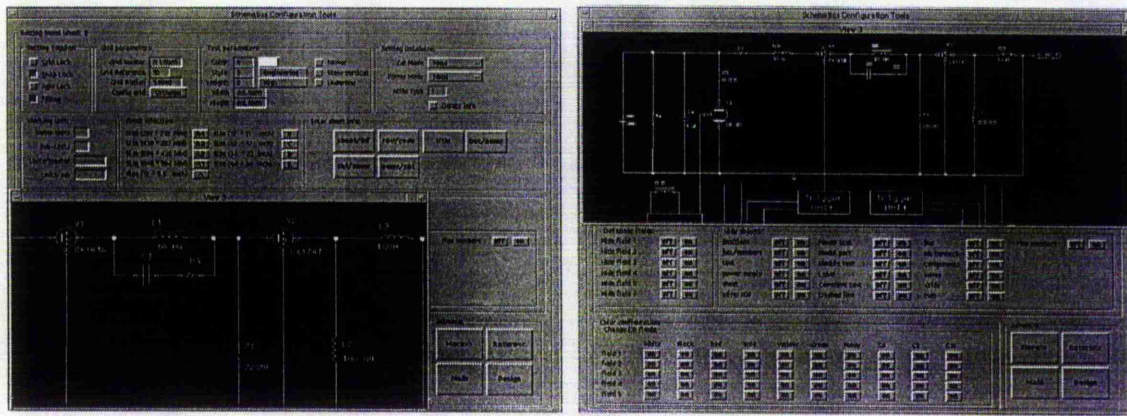


Figure 7.36 Display of settings and levels control tools.

## 7.8 DESIGN DOCUMENTATION

Chapter 2 Section 2.6.4 has described the different ways of extracting and representing design documentation using different automation tools. In order to test the capability of extracting these data. They are extracted by means of reports generation, design queries, and data retrieval. In this design, for example, Figure 7.37 shows a display of the different tools associated with document generation of both ASCII format and database, while Figure 7.39 shows data extracted as results of using these tools.

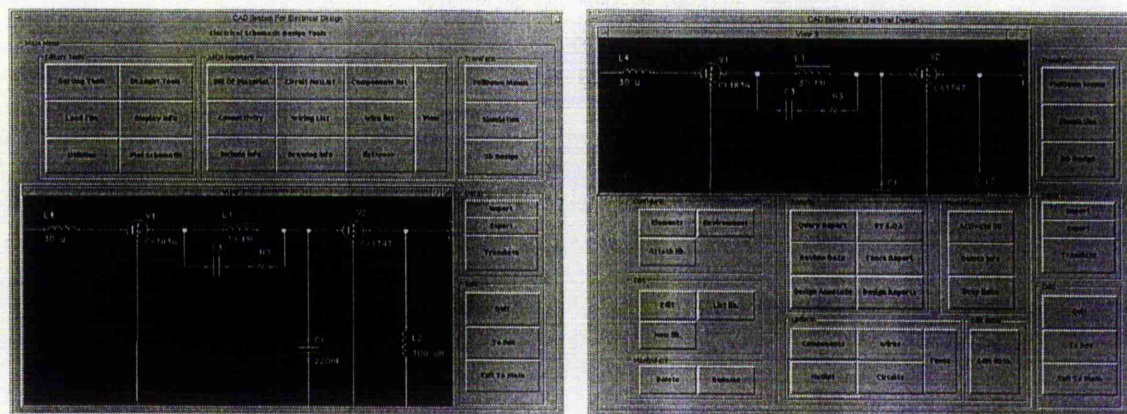


Figure 7.37 Display of Documents generation tools.

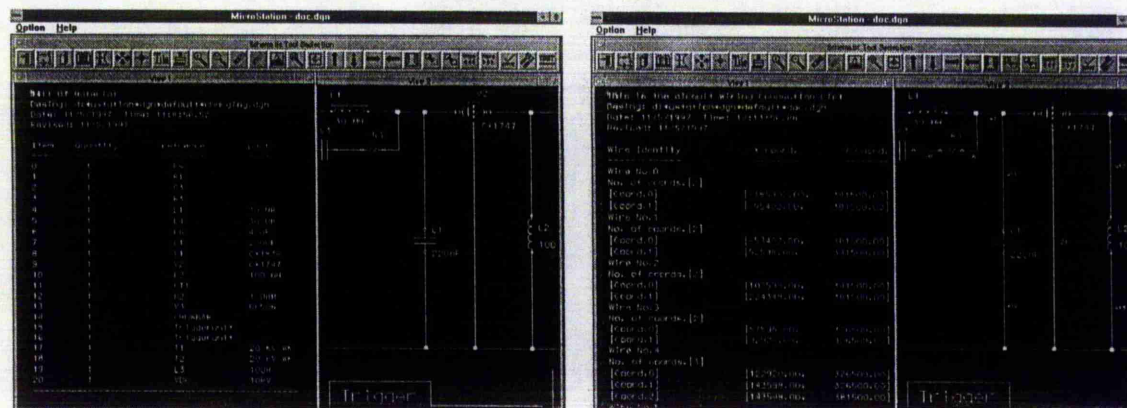


Figure 7.38 Circuit B.O.M and wiring list extraction.

## **7.9 CONCLUSIONS.**

The objectives of this chapter were to demonstrate the developed software tools and its integrated environment. It is an integrated environment that provides an approach to software development based on the concept of electro-mechanical design and analysis. The whole process lends itself to an iterative design cycle that facilitates the investigation of different design solutions.

For the purpose of investigating the approach of electro-mechanical design, a resonant charging circuit was designed using all the available tools. A set of design issues involving design testing, tools design capabilities, circuit-models simulations, and many others have been performed. This was carried by the assistance of other integrated environments (i.e. schematics, 3D layout, database dbase(iv), and PSPICE).

It has been shown possible to integrate the design tools for both electronic and mechanical design within an integrated framework. Transfer of design data back and forward between these two domains has been demonstrated. The benefits of this system approach are numerous:

- Reduced design time.
- Ease of verifying design performance.
- Facilitates design optimisation.
- Improved design management.
- Analysis of mechanical design constraints effecting electrical performance (i.e. interconnections (stray inductance/capacitance/resistance).
- Ability to link to standard CAE tools and databases.

Of particular importance to the utility of this software is the user interface. Using a MOTIF standard a full range of design tools and utilities are provided which gives the designer a very flexible environment to perform design, analysis and report generation. The major features of GUI include:

- It provides a convenient and efficient means to access design different functions.
- The ability to communicate with other software packages through import/export tools.
- The ability to maintenance design data through database tools.

- The ability to generate design different reports using both ASCII and database facilities.
- It helps the link with library facilities for 2D and 3D environment.
- It allow the configuration of drafting tools including workspace setting.

Following the design process progress, it became visible that other design tasks must be applied. For example, in the case of designing inductor L1, an iterative approach was required to reach the most possible solutions. The design is therefore, verified using the design tools for obtaining the final design documents, as they are obtained using ASCII format files and database tools.

Although there is much scope for further improvement in the design tools, the important design features of the system have been achieved. The "open" architecture of the system facilitates the development of design tools which are parameterically driven and link both to mechanical parameters and electrical parameters. It is therefore possible to optimise designs quickly when circuit performance is underdevelopment on both physical mechanical properties which effect electrical behaviour (i.e. stray capacitance/inductance). The design library tools could be expanded to cover a wide range of components which are at present time-consuming to design.

#### **REFERENCES.**

- [7.2] McDonald D., W., "10 kHz pulse repetition frequency  $CO_2$  laser fro processing high damage threshold material", PhD. Thesis, department of Mechanical Engineering, Glasgow University, October 1989.



# 8 General Conclusions and future work

The primary objective of this research was to design and develop an integrated environment for electro-mechanical system design and analysis, capable of examining the issues which are important in integrating both electrical and mechanical design domains. Of key importance was the identification of critical issues related to concurrent engineering.

To limit the workload involved in the project and to concentrate on the key issues of this research, a system was designed around a commercial CAD package, which offered an open architecture (MicroStation). Although this reduced considerably the engineering of the user interface, there were many shortcomings of this approach, in particular, extraction of information. The following summary and conclusions can be given which hopefully highlights the areas of progress and the major difficulties overcome in developing the software.

## 8.1 GENERAL CONCLUSION

An integrated design environment dealing with electro-mechanical design and analysis has been designed and developed. It provides the ability to design and examine different design issues, as well as to perform design analysis and performance evaluation. The system's main attraction is that it facilitates design of electro-mechanical systems and the management of design information, the storage and transferring information among these environments.

## **Software Developed**

A 2D schematic capture environment containing different design tools has been designed and developed for constructing and editing circuit schematics. One of the aims in constructing the 2D schematic diagrams was to show the functional relationships among the components (i.e. connectivity) allowing electrical circuits to be analysed for optimal performance. The other aim was to assist with the construction and updating of 3D models, allowing assemblies to be designed and modelled for best possible mechanical layout.

An automated design tool set has been developed to aid the construction of 2D schematics. The methodology employed was a file scanning technique in which various scanning functions were constructed to perform different data extraction. Some of these applications include automatic junction placement simply by selecting a wire, automatic wires identities placement, automatic junction numbering, and automatic element recognition allowing design data to be entered in response to selected element.

In association with 2D schematics several applications were also designed and developed to automate design different tasks. One of these applications was the generation of netlists format files, as these are the key element in carrying out circuit simulations. This, however, underlines the use of MicroStation in particular. With reference to the rules and conditions defined for this purpose (refer to chapter 4), it is clear that the principles of extracting a netlist for circuit simulation is based on wire-component relationship and connectivity establishment. As it has been described in chapter 2, the MicroStation's CAD engine contain various graphical elements, which carry the necessary data needed to specify components, however, there is no tracking of object connectivity. It was therefor necessary to develop a technique whereby the interconnection of objects placed was completed and arranged in a suitable form.

One of these techniques was the extract-process method. This technique was found to be time-consuming; due to complexity of circuit schematics and the spatial arrangement of components. In essence, the CAD engine does not easily allow the generation of connectivity reports on objects, although the labelling and creation of symbols was relatively straight forward. The other method employed was an edit-extract coupled with dbase(iv), which was found to be more efficient. Files describing a circuit and its connectivity are extracted giving output in a format suitable for simulation with any SPICE based modeller. The development of this software constituted a major portion of the work in this project.

Several problems associated with the extraction of netlist have been experienced. One of these problems was the application of extract-process technique. Since it was designed as a 2-stage procedure consisting of two subprograms, its performance was too slow. This was due to one reason: Although the process of extracting data schematic was extremely fast, the second stage which its task is to process the extracted data was the main problem due to data comparison needed between both wires and components co-ordinates, where many matrices were involved including the scanning of the node matrix. This problem was solved by using the second technique which involved the use of a database table. As a result, this has proved using dbase (iv) is a more efficient and flexible way of extracting schematics netlists.

One of the other main design issues addressed in designing this system was the ability to construct 3D models, which were linked to 2D schematic objects. The decision to choose MicroStation was based on a number of criteria specific to 3D modelling. The most important aspects were that solids could be used to design electrical components and that certain parts could be parameterised partly or completely, also the ability to render by taking into account the specified features such as bending radius, and ability to transfer 2D schematics drawing into 3D drawing format. The main feature with respect to 3D models was the use of the database, its role was to store design information and modelling parameters which are used to update components within the 3D models. Other facilities were also provided including assembly wiring construction, through connectivity establishment and the generation of different design reports and quires. Thus, a direct link between 2D and 3D domains has been created.

Chapter 3 described the design procedure for electro-mechanical components. The key element of developing this procedure was the concept of parametric design, enabling the modelling of each component where necessary within a circuit. The procedure works within both 2D or 3D modelling workspaces using design templates as parameters editors and dbase(iv) as a supportive tool. The procedure performance was tested for electro-magnetic components such as inductors and transformers and as a result, PSPICE syntax files were obtained suitable for simulation. Additional design information can also be extracted from the database suitable for design modifications and refinement (i.e. design optimisation).

Based on the conceptual model described in Chapter 5 Section 5.6, which illustrate the main source of these data. Tool sets for database auto-link were designed and developed providing a full integration between different design environments. By demonstrating the performance of these applications, it was observed that such an applications are necessary to bind the integrating environments together. These tools provide a number of benefits.

The most significant benefit is that through its integration and interface with 2D schematics environment, it provides the capability to manipulate stored data and the ability to update design parameters used to represent 3D components models. These models can be maintained in their final formats and used to support further design changes. Another significant benefit arises from the fact that the data stored in different database tables can be extracted into its final format very quickly. This facilitates the ease with which models can be transferred and interfaced with PSPICE for simulation and as result, reduces the time taken in editing netlist files manually. Therefore, it was concluded that using database in establishing the link between both 2D and 3D workspaces is the key element, ensuring the integration role in designing electro-mechanical components.

Due to the complexity in establishing the direct link between 2D schematics and dbase (iv). It was found that establishing the link with database was rather complicated when using MicroStation (ver. 4) in a way that many files needed to be created and additional information edited into dbase(iv) control file. This operation created interfacing errors every time the applications were executed, thus resulting time-consuming. In order to overcome this problem, a decision was made to use the other version of MicroStation (i.e. ver.5 ), and the problem was solved by creating a batch file containing the necessary files and by compiling this file, a direct link was achieved in much more flexible way.

Much work has been done towards design verification, which basically involved the extraction of design documentation and other relevant information. Most of this work was aimed at developing software programs and database applications. Chapter 7 show the performance of these tools and as a result, it was clear that these applications provided a vital design information and reporting including:

- Circuit design reports including (Bill Of Material, wiring list, etc.).
- List of components and models used for designing a circuit in 2D schematic forms.
- Connectivity reports ensuring the connectivity between components.
- Reports on component design models.

A unified user interface through which the different environments are integrated was designed and developed. It provides editing of operations depending on user request, and also provides various graphical tools used to facilitate the construction of different circuit schematics, 3D models for mechanical layout, and the ability to perform different design tasks and decisions in a user-friendly manner.

Attention was directed towards the design management tools which are important in managing the system design processes, thus, improving productivity and efficiency. Chapter 6 described the different levels and their use in achieving integrated design environment. One area in particular was drawing control utilities, where different configuration tools are managed and organised to carry out specific design changes and modifications, also to assist with the settings of design workspaces. This, however, gave a number of advantages:

- Taking control of how drawing functions operate simply by invoking the corresponding a function button.
- Element attributes are automatically configured and modified without selecting an element from design file. These include database fields, schematics objects, wires, and components attributes.
- Components libraries are organised and arranged for easy traceability, which provides quicker attachment and manipulation.
- The construction of circuit schematics includes the concept of design by levels for easy display and convenient element manipulation, and also allows to take control of view attributes which determines whether parts of schematic will be displayed.
- Improving time taking to construct circuit schematics by merging different MDL commands into a single functions which performs different design task, improving efficiency and flexibility.
- It controls the manner in which design files are displayed.

Another area specific to the construction of different libraries cells was considered, where standard symbols are used to represent 2D and 3D components of different kind. For this purpose, two separate design environments were designed and developed for both 2D and 3D cells construction. Each environment contain various tools and palettes to assist with the constructing of standard and customised symbols for both electrical and mechanical designs. These tools include tool palette of different pins type, palette of different graphical elements, database linking facilities, automatic element configuration, etc.

The other area aimed at was the method of how design data is managed. Chapter 5 has detailed the design of different database applications which were used for extracting design information. The primary objectives in developing these applications was to



organise data in a way that can easily be stored and retrieved. In Chapter 7, a detailed investigation was undertaken to evaluate the performance of these tools. It was recognised that there was a need for exchanging information between design environments. To facilitate the exchange and retrieval of these data, different SQL command tools were used for quick review. Generated documents are managed allowing easy tracking by using query editors dialog of different activities. In addition, reports can be generated to support application reviews of completion status and identification of problems.

It is clear from the literature and the present work, that designing such system highlights a number of areas which require further investigation. Therefore, recommendations for future development are discussed below.

## **8.2 FUTURE WORK**

There are several areas are recommended for further work. These are as follows:

- Links with other commercially available software packages:  
It would be interesting to achieve direct links with commercial software packages such as excel. Although, a procedure involving components design and modelling was developed, links with common software packages could be beneficial for component design and modelling. This would permit editing of parameters and mathematical equations. The achievement of this idea can be gained using special data handling interface called dynamic data exchange (DDE)- a mechanism allowing related or linked data existing in separate files to be updated simultaneously and make possible close integration of applications from separate vendors providing the systems are windows-based applications.
- Electro-magnetic capability (EMC):  
Research towards components modelling techniques for electro-magnetic is of great interest, therefore, further research should be directed towards this area. So far, various models have been formulated in an attempt to capture the mechanisms of design. These include the designer thought processes through using graphics templates, providing the interface with other environments such as database. Components models should be designed to contain information that describe the various phases of design process (e.g. material and core type selection, geometric descriptions, magnetic properties, etc.). These models could be designed and implemented using rule-based knowledge techniques, which will lead to the fundamental data structural in which further tasks can be performed to obtain an optimised solutions.

- Automated tools:

Further expansion to software programs that deal with element recognition (i.e. file scanning technique) to develop further tools specifically for 3D modelling environments is required. These tools could be designed to determine whether all connections between components are logical, and if the wires have been positioned in the required place of the associated housing by generating wiring diagrams. With reference to 2D design, automatic generation of circuit schematic could be performed using wiring list and bill of material extracted files. The program would involve 2-stage procedure providing the attachment of cells libraries. The advantage of this procedure is to help the designer to reconstruct a circuit schematic in case of losing the origin design file, thus, time-saving.

- Design management:

Despite the advantages obtained from the developed management tools, there is a demand for further management tools. Further investigation should be carried to determine the tools that are used most often and are the most effective in performing the various design tasks. Because of these preferences, the interface should include varied tools, methods of data transformation, and design entry. This would greatly enhance the system's effectiveness and efficiency. For example, it would be possible to add on-line help menus with helpful instruction to understand the purpose of functions and commands.

- Future CAD systems

Some of the major obstacles to implementing an integrated environments are the fact that many existing software systems are designed as stand-alone units which are not capable of communication in a general and flexible way. The concept of designing this kind of systems will need to be redesigned to allow both flexible processing and flexible communication.

Although, the current CAD systems are coupled with their tools and applications, there is much work in progress to bring design and analysis to achieve total integration, also there have been significant advances in the development of GUIs (e.g. many systems have been provided with NT-window based interfaces) for convenient and flexible use. This can be achieved by better software engineering and better understanding of the designer needs and faster hardware. Following is a list of research challenges that will be encountered as the future predictions in realising this framework:

(1) Designing CAD systems, which are capable of:

- Interactive information exchange with other software environments.

## *Chapter 8: General Conclusions and Future Work*

- Advanced parametric design techniques.
- Enhanced user interfaces and management tools for better performances.
- Advanced automated tools to facilitate designs such as electro-mechanical assembly layouts and detection.
- Modelling techniques for efficient engineering of complex systems by developing new software languages.

