

AD-A073 383

RESEARCH TRIANGLE INST RESEARCH TRIANGLE PARK N C

F/8 9/5

AFAL SIMULATION FACILITY/CAPABILITY MANUAL. VOLUME II. EXECUTIV--ETC(U)

FEB 79 R L EARP

F33615-76-C-1308

UNCLASSIFIED

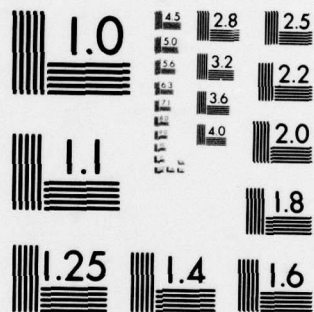
AFAL-TR-77-118-VOL-2

NL

1 OF 3

AD
A073383





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

② LEVEL III 98

AFAL-TR-77-118

A055591

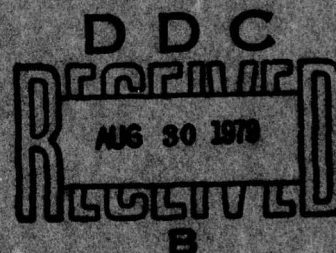


AD A 073383

AFAL SIMULATION FACILITY/CAPABILITY MANUAL
Volume II
Executive Summary and Electronic Technology Division:
Computer Aided Design for Microelectronics

RESEARCH TRIANGLE INSTITUTE
RESEARCH TRIANGLE PARK, NORTH CAROLINA 22709

FEBRUARY 1979



TECHNICAL REPORT AFAL-TR-77-118
Final Report for Period 30 June 1977 - 31 March 1978

DDC FILE COPY

Approved for public release; distribution unlimited.

AIR FORCE AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433


79 08 28 046

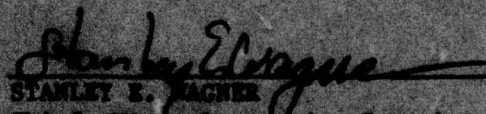
NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.


This report has been reviewed by the Information Office (IO) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.


J. S. RAWLINGS, II, CAPT, USAF
Project Engineer


STANLEY E. WAGNER
Chief, Microelectronics Branch

FOR THE COMMANDER


WILLIAM J. EDWARDS, Director
Electronic Technology Division
Air Force Avionics Laboratory

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify AFAL/DIE, 7-2 AFB, ON 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

18 AFAL

19 TR-77-118-VOL-2

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER 77 AFAL-TR-118, Volume II	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AFAL SIMULATION FACILITY/CAPABILITY MANUAL, EXECUTIVE SUMMARY AND ELECTRONIC TECHNOLOGY DIVISION: COMPUTER AIDED DESIGN FOR MICROELECTRONICS	5. TYPE OF REPORT & PERIOD COVERED Technical - Final Report 30 June 1977 - 31 March 1978	
7. AUTHOR(s) 10 Ronald L./Earp	8. CONTRACT OR GRANT NUMBER(s) 15 F33615-76-C-1308 ✓	
9. PERFORMING ORGANIZATION NAME AND ADDRESS Research Triangle Institute P.O. Box 12194 Research Triangle Park, North Carolina 27709	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62204F 60964001	
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Avionics Laboratory AFAL/AAF, Wright-Patterson AFB, Ohio 45433	12. REPORT DATE February 1979 13. NUMBER OF PAGES 204	
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE	
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release, distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) DDC RECEIVED AUG 30 1979 B		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Computer Aided Design Logic Simulation Logic Design Fault Analysis LSI Design Printed Circuit Structural Analysis Printed Circuit Board Layout Printed Circuit Thermal Analysis		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) The Air Force Avionics Laboratory (AFAL) at Wright-Patterson AFB is the focal point for development of new avionics technology for the Air Force. In order to carry out this responsibility, a significant capability to simulate physical avionics systems and components has been created by the AFAL divisions.		

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

304 400

79 08 28 046

TOP

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

This manual presents introductory and summary coverage of the AFAL Computer Aided Design (CAD) Facility in Section I and by presenting more detailed descriptive material in subsequent sections. The contents of Section I address the CAD Facility capabilities from a planning/management viewpoint by relating the Laboratory mission to present facility capability through the development of a conceptual simulation class structure. The contents of subsequent sections of this manual address specific facility/capability from a potential-user viewpoint. Specific programs/functions documented are as follows:

1. LOGIC4 - Logic simulation and fault analysis.
2. ASPEC - Analog circuit analysis
3. ASSIGN/PCPRA - Printed circuit board package assignment, placement, and wire routing.
4. CONVRT/PRF/CHPCHP/MCHPCK - Integrated circuit (IC) layout, route, and verification.
5. SCELL/MOSTRAN - IC cell design and analysis.
6. MAGNET - Microwave network design and analysis.
7. SAP IV - Structural analysis.
8. SINDA - Thermal analysis.
9. LIBBER - File/Subfile management.

The technical level of these sections is such that available capability can be determined and some insight can be gained regarding user interface.

ADDITION for	
NTIS	White Section <input checked="" type="checkbox"/>
DDC	Bull Section <input type="checkbox"/>
UNANNOUNCED	<input type="checkbox"/>
JUSTIFICATION	
BY	
DISTRIBUTION/AVAILABILITY CODES	
Dist. AVAIL. and/or SPECIAL	
A	

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

TABLE OF CONTENTS

		Page
	Executive Summary	xiii
1	SYSTEM OVERVIEW	1
1.1	INTRODUCTION	1
1.2	PROGRAM CAPABILITIES	2
1.2.1	Printed Circuit Board Design	2
1.2.1.1	Logic Simulation	2
1.2.1.2	Fault Analysis	4
1.2.1.3	ELEMENT and NET LIST Generation	4
1.2.1.4	Printed Circuit Board Layout	5
1.2.2	Custom LSI Design	5
1.2.2.1	The Basic Approach	6
1.2.2.2	Cell or Circuit Level Design	6
1.2.2.3	Chip or Logic Level Design	7
1.2.3	Microwave Network Design	8
1.2.4	Structural Analysis	8
1.2.5	Thermal Analysis	9
1.2.6	Data Base Management	9
1.3	FILE MANAGEMENT	10
1.3.1	File Assignments	10
1.3.2	Command Characteristics	10
2	DATA BASE MANAGEMENT PROGRAM - LIBBER	12
2.1	INTRODUCTION	12
2.2	PROGRAM PURPOSE	12
2.3	PROGRAM GENERAL DESCRIPTION	12
2.4	PROGRAM OPERATION	14
2.4.1	Program Access	14
2.4.2	LIBBER File Creating Commands	15
2.4.2.1	File Assignment	15
2.4.2.2	File Initialization	15
2.4.2.3	Subfile Creation	15
2.4.2.4	File Monitoring Commands	16
2.4.3	LIBBER Exit Command	16
2.5	LIBBER COMMAND DESCRIPTIONS	16
2.5.1	RUN LIBBER Command Description	16
2.5.2	File Assignment Command Description	16
2.5.3	HEL Command Description	17
2.5.4	TTY Command Description	17
2.5.5	TOC	17

TABLE OF CONTENTS (con.)

		Page
2.5.6	NEW Command Description	18
2.5.7	ADD Command Description	18
2.5.8	APP Command Description	19
2.5.9	DEL Command Description	19
2.5.10	LST Command Description	19
2.5.11	GET Command Description	19
2.5.12	SAV Command Description	20
2.5.13	RES Command Description	20
2.5.14	MRG Command Description	21
2.5.15	EDT Command Description	21
2.5.16	REN Command Description	22
2.5.17	UNT Command Description	22
2.5.18	REW Command Description	22
2.5.19	PRT Command Description	22
2.5.20	Card Headers	22
 3	 LOGIC SIMULATION AND FAULT ANALYSIS PROGRAM — LOGIC 4.	 24
3.1	INTRODUCTION AND GENERAL DESCRIPTION	24
3.2	FUNCTIONAL DESCRIPTION	27
3.2.1	Major Functions	27
3.2.2	Stimulus/Response Simulation	28
3.2.3	Test Stimulus Completeness Verification	29
3.2.4	Fault Isolation	29
3.2.5	Program Timing	30
3.2.5.1	Gate Time	30
3.2.5.2	Coarse Time	30
3.2.5.3	Gates/Coarse-Time	30
3.2.5.4	External Time	31
3.2.5.5	Output Time	31
3.2.5.6	Strobe Time	31
3.2.5.7	Normal Simulation	31
3.2.5.8	Fault Simulation	32
3.2.5.9	Example Program Timing	33
3.2.5.10	Summary	33
3.2.6	Program Inputs	33
3.2.6.1	Control Category	36
3.2.6.2	Run Header Category	36
3.2.6.3	Elements Category	36
3.2.6.4	Generator	36
3.2.6.5	Externals Category	37
3.2.6.6	Faults Category	37
3.2.6.7	Outputs Category	37
3.2.6.8	Memory Category	37
3.2.6.9	Definitions Category	37

TABLE OF CONTENTS (con.)

		Page
4	ELEMENTS/NET GENERATION PROGRAM — ASSIGN	38
4.1	INTRODUCTION	38
4.2	PROGRAM PURPOSE	38
4.3	PROGRAM GENERAL DESCRIPTION	38
4.3.1	String Generation	41
4.3.2	Element Generation	41
4.4	PROGRAM FUNCTIONAL DESCRIPTION	41
4.4.1	Program Initialization	43
4.4.2	Control Category	42
4.4.3	Backplane Description	43
4.4.4	Elements Category	44
4.4.5	Generator Specification	44
4.4.6	Externals Category	44
4.4.7	Equivalence Category	44
4.4.8	Bucket Layout	45
4.4.9	Card Descriptions	45
4.4.10	String List	46
4.4.11	Manual Definitions	46
4.4.12	Parts List	46
4.5	PROGRAM OPERATION	46
4.5.1	Manual Definition Phase	46
4.5.2	Tag Definition Phase	47
4.5.3	All Elements Phase	47
4.5.4	Gate Second Try Phase	47
4.6	PROGRAM OUTPUT DESCRIPTION	47
4.6.1	Input Data	48
4.6.2	Card Descriptions	48
4.6.3	List of Non-Fitting Elements	48
4.6.4	Bucket Map	48
4.6.5	Spare Elements	49
4.6.6	Signal Loading	49
4.6.7	Internal String List	49
4.6.8	Test Point List	49
4.6.9	Unused I/O Connector Pins or Unused Test Points	49
4.6.10	Allocated Elements	49
4.6.11	Parts Summary	50
4.6.12	Computer Written Parts List	50
5	INTRODUCTION TO PCPRA: PRINTED CIRCUIT PLACEMENT ROUTING AND ARTWORK PROGRAM	51
5.1	GENERAL DESCRIPTION OF PCPRA	51

TABLE OF CONTENTS (con.)

		Page
5.1.1	Printed Circuit Board Design Process	53
5.1.2	PCPRA Requirements	55
5.1.3	PCPRA Utilization Guidelines	57
5.2	PCPRA FUNCTIONAL DESIGN AND CAPABILITIES	59
5.2.1	PLACE Program	59
5.2.2	WORZ Program	61
5.2.3	PROUTE Program	64
5.2.4	MLPLOT Program	65
6	INTRODUCTION TO SCELL/MOSTRAN, CELL LAYOUT AND TRANSIENT ANALYSIS PROGRAMS	67
6.1	GENERAL DESCRIPTION	67
6.1.1	SCELL Program	67
6.1.2	MOSTRAN Program	69
6.2	FUNCTIONAL DESCRIPTION	73
6.3	SCELL PROGRAM FUNCTIONS	79
6.3.1	BULK MANIPULATION Category	79
6.3.2	CAPACITANCE Category	80
6.3.3	COMPOSITE CHARACTERS Category	80
6.3.4	CONTROL Category	80
6.3.5	DESIGN RULES Category	80
6.3.6	EXECUTE Directive	80
6.3.7	GLOBAL PARAMETERS Category	80
6.3.8	LIBRARY Category	81
6.3.9	NET LIST Category	81
6.3.10	PLACEMENT CONSTRAINTS Category	81
6.3.11	RUN HEADER Category	81
6.3.12	STOP Directive	81
6.4	MOSTRAN PROGRAM FUNCTIONS	81
6.4.1	APPLIED SIGNALS Category	82
6.4.2	CIRCUIT PARAMETERS Category	82
6.4.3	GENERAL PARAMETERS Category	82
6.4.4	INITIAL CONDITIONS Category	82
6.4.5	MERGE Category	82
6.4.6	MODIFY Category	82
6.4.7	NET LIST Category	83
6.4.8	OUTPUT CURVES Category	83
6.4.9	RUN HEADER Category	83
6.4.10	SIMULATE Directive	83
6.4.11	STOP Directive	83
6.4.12	TYPE/MODEL ASSIGNMENTS Category	83
6.4.13	UNIT NUMBERS Category	83

TABLE OF CONTENTS (con.)

		Page
6.5	PROGRAM OUTPUTS	83
7	IC LAYOUT PROGRAMS	87
7.1	INTRODUCTION	87
7.2	PROGRAM DESCRIPTIONS	87
7.3	PROGRAM OPERATION	87
7.4	INPUT DATA CONVERSION – CONVRT	88
7.5	CHIP LAYOUT	91
7.5.1	Component Sets	92
7.5.2	Level Sets	92
7.5.3	Automatic Placement	92
7.5.4	Chip Modification	93
7.5.5	User-Specified Placement	94
7.6	LAYOUT CHECKING – MCHPCK	97
7.6.1	Initial Net Check	98
7.6.2	Final Net Check	98
7.6.2.1	Orthogonal Interconnection	99
7.6.2.2	Diagonal Interconnection	99
7.6.2.3	Connections to Cell Terminals	100
7.6.2.4	Diagonal Connections to Cell Terminals	101
7.6.2.5	Terminal to Terminal Connections	101
7.6.2.6	Crossover (XOVER) Connections	101
7.6.2.7	BUS Connection	101
7.6.2.8	Logically Equivalent Inputs	102
7.6.2.9	Extraneous Material	102
7.6.2.10	Overall	102
7.6.3	Capacitance Loading Output	103
7.6.4	Output Net List	103
7.6.5	Design Rule Check	104
7.6.5.1	Cell to Cell Spacing	104
7.6.5.2	Cell to Interconnect Spacing	105
7.6.5.3	Interconnect to Interconnect Spacing and Width	105
7.6.5.4	Diagnostics	105
7.6.5.5	Overall	105
8	MICROWAVE NETWORK ANALYSIS AND OPTIMIZATION PROGRAM – MAGNET	106
8.1	INTRODUCTION AND GENERAL DESCRIPTION	106
8.1.1	INPUT Data	106
8.1.2	MAGNET Program	108

TABLE OF CONTENTS (con.)

		Page
8.1.3	OUTPUT Data	109
8.2	FUNCTIONAL DESCRIPTION	109
8.2.1	Starting Point	110
8.2.2	Network Analysis	110
8.2.3	Optimization Routine	111
8.2.4	Output Routine	113
8.3	PROGRAM MINIMUM REQUIREMENTS	114
8.3.1	Elements Coding	114
8.3.2	Program Control Coding	114
8.3.3	Program Outputs	114
8.3.4	Final Optimized Circuit	115
8.3.5	Composite S Parameters	115
8.3.6	Final Circuit Description	115
8.3.6.1	Starting Point Analysis	115
8.3.6.2	Optimized Results	115
8.3.6.3	Final Circuit	116
8.4	USAGE INFORMATION	116
8.4.1	Active Device S Parameters	116
8.4.2	Program Parameters	116
8.4.3	Frequency Response	116
8.4.4	Element Creation Statements	116
8.4.4.1	Active Device	117
8.4.4.2	Non-Active Elements	117
8.4.4.3	Repeat	117
8.4.5	Element Combination Statements	117
8.4.6	INV Statement	117
8.4.7	Network Definition Termination	117
8.4.8	Run MAGNET	117
8.4.9	File Assignment	118
8.5	OUTPUTS	118
8.6	RESTRICTIONS ON MAGNET	118
8.7	ABCD MATRIX	122
9	A STRUCTURAL ANALYSIS PROGRAM – SAP IV	123
9.1	GENERAL DESCRIPTION	125
9.2	PROGRAM ORGANIZATION	126
9.2.1	Nodal Point Input Data and Degrees of Freedom	126
9.2.2	Element Mass and Stiffness Calculations	127
9.2.3	Formation of Structure Stiffness and Mass	127

TABLE OF CONTENTS (con.)

		Page
9.3	THE ELEMENT LIBRARY	130
9.3.1	Three-Dimensional Truss Element	130
9.3.2	Three-Dimensional Beam Element	130
9.3.3	Plane Stress, Plane Strain and Axisymmetric Elements	130
9.3.4	Three-Dimensional Solid Element	132
9.3.5	Variable-Number Nodes Thick Shell and Three-Dimensional Element ..	132
9.3.6	Thin Plate and Shell Element	132
9.3.7	Boundary Element	133
9.3.8	Pipe Element	133
9.4	THE EQUILIBRIUM EQUATIONS FOR COMPLEX STRUCTURAL SYSTEMS	134
9.4.1	Element to Structure Matrices	134
9.4.2	Boundary Conditions	134
9.5	STATIC ANALYSIS	135
9.5.1	Solution of Equilibrium Equations	135
9.5.2	Evaluation of Element Stresses	136
9.6	CALCULATION OF FREQUENCIES AND MODE SHAPES	136
9.6.1	The Determinant Search Solution	138
9.6.2	The Subspace Iteration Solution	138
9.6.3	Dynamic Optimization	140
9.7	DYNAMIC ANALYSES	142
9.7.1	Response History Analysis by Mode Superposition	142
9.7.2	Response History Analysis by Direct Integration	143
9.7.3	Response Spectrum Analysis	143
9.7.4	Restart Capability in Mode Superposition Analysis	144
9.7.5	Mode Superposition Versus Direct Integration	145
9.8	INSTALLATION OF SAP IV ON A SYSTEM OTHER THAN A CDC COMPUTER	147
10	SINDA	149
10.1	INTRODUCTION	149
10.2	SYSTEM STRUCTURE DESCRIPTION	149
10.3	INPUT DECK	153
10.3.1	Title Block	155
10.3.2	Data Blocks	155
10.3.2.1	Node Data Block	157
10.3.2.2	Source Data Block	158
10.3.2.3	Conductor Data Block	158
10.3.2.4	Constants Data Block	162

TABLE OF CONTENTS (con.)

		Page
10.3.2.5	Array Data Block	164
10.3.3	Operations Blocks	164
11	ASPEC-ADVANCED SIMULATION PROGRAM FOR ELECTRONIC CIRCUITS	173
11.1	INTRODUCTION	173
11.2	PROGRAM ANALYSIS CAPABILITIES	173
11.3	PROGRAM DESCRIPTION	174
11.3.1	Components Specification	174
11.3.1.1	Elements	174
11.3.1.1.1	Resistors, capacitors, and inductors	174
11.3.1.1.2	Battery	174
11.3.1.1.3	Transconductance	175
11.3.1.1.4	Voltage-controlled switch	175
11.3.1.1.5	Coupled inductors	175
11.3.1.2	Sources	176
11.3.1.2.1	DC sources	176
11.3.1.2.2	Piecewise-linear source	176
11.3.1.2.3	Repeating piecewise-linear source	176
11.3.1.2.4	Piecewise-exponential source	176
11.3.1.2.5	Repeating piecewise-exponential source	177
11.3.1.2.6	Sinusoidal source	177
11.3.1.2.7	AC source value	178
11.3.1.3	Nonlinear devices	178
11.3.1.3.1	PN and Schottky diodes	178
11.3.1.3.2	Bipolar junction transistors (BJT)	178
11.3.1.3.3	Junction field-effect transistors (JFET)	178
11.3.1.3.4	MOS field-effect transistors (MOSFET)	179
11.3.2	Instruction Statements	179
11.3.2.1	•MODEL Statement	179
11.3.2.2	•PARAM Statement	180
11.3.3	Analysis Specification	180
11.3.3.1	DC Analysis	180
11.3.3.2	•TRAN Statement	180
11.3.3.3	•TFUN Statement	180
11.3.3.4	•FREQ Statement	181
11.3.4	OUTPUT Specification	181
11.3.5	Macro Definition and Use	182
11.3.5.1	•MACRO Statement	183
11.3.5.2	Xname Statement	183
11.3.5.3	Specification of Macro Output	184
11.3.6	Temperature Specification	184
11.3.7	Worst-Case Analysis	184
11.3.8	Noise Analysis Specification	184

TABLE OF CONTENTS (con.)

		Page
11.3.9	Print Control Specification	185
11.3.10	DC Node Voltage Specification	186
11.3.11	Breadboard Simulation	186
11.3.12	Analysis Control	186

LIST OF FIGURES

	Page
1.2-1 CAD General Functional Block Diagram	3
2.1-1 CAD Functional Block Diagram Emphasizing the LIBBER Program	13
3.1-1 CAD Functional Block Diagram Emphasizing the LOGIC 4 Program	25
3.2.5.9-1 Simulation Functions Performed for One Coarse Time	34
3.2.5.9-2 Fault Simulation Operation	35
4.0-1 CAD Functional Block Diagram Emphasizing the ASSIGN Program	39
4.3-1 ASSIGN Program Modes of Operation	40
5.0-1 CAD Functional Block Diagram Emphasizing the PCPRA Programs	52
5.1-1 PCPRA Program Flow	54
5.1.1-1 Exploded View of a Typical Multilayer Board	56
5.1.3-1 Typical Power/Ground Routing	58
6.0-1 CAD Functional Block Diagram Emphasizing the SCELL/MOSTRAN Programs	68
6.1.2-1 Time Window	71
6.1.2-2 The P-Channel and N-Channel Models	71
6.1.2-3 Time Step Sequence	74
6.2-1 Data Flow in a Typical Cell Design Effort	76
6.2-2 Cell Design Working Structure	77
7.2-1 Chip Layout Programs, Block Diagram	88
7.2-2 CAD Functional Block Diagram Emphasizing the Chip Layout Programs	89
7.5.4-1 Example of Height Contraction	95
7.5.4-2 Example of Height Expansion to Reduce Interconnect Crowding	96
7.6.2.2-1 Invalid Connections Between Orthogonal Segment A,B, and Diagonal Segment C,D	100
8.1-1 CAD Functional Block Diagram Emphasizing the MAGNET Program ..	107
8.1-2 MAGNET Program General Flow Diagram	108
8.2-1 Simplified Flow Chart of MAGNET	109
8.2-2 Element Type	111
8.2-3 Element Combination	112
8.2-4 Example Network Not Accepted by MAGNET	113
9.0-1 CAD Functional Block Diagram Emphasizing the SAP IV Program	124
9.2.3-1 Flow Chart Showing Calculation of Number of Equations in a Block	128
9.2.3-2 Flow Chart for Calculation of Structure Stiffness Matrix and Mass Matrix	129
9.3-1 Element Library of SAP IV	131
9.7.5-1 Amplitude Decay Wilson θ -Method	146
10.1-1 CAD Functional Block Diagram Emphasizing the SINDA Program	150
10.2-1 Detailed Internal Flow of the SINDA System	152
10.3-1 Basic SINDA Input Deck	156
10.3.2.1-1 Summary of Node Data Input Options	159
10.3.2.2-1 Summary of Source Data Input Options	160
10.3.2.3-1 Summary of Conductor Data Input Options	163
10.3.3-1 Basic Program Flow	166
10.3.3-2 Sample Flow Chart for the Execution Block	168
10.3.3-3 Nested Structure of the Operations Blocks	169
10.3.3-4 Flow Chart of Network Solution Subroutine CNFRWD	170
11.3.5-1 A Macro Example	182

LIST OF TABLES

		Page
1.1-1	CAD Programs	1
2.5.20-1	Card Holder Options.....	23
4.4-1	ASSIGN Input Data Category Functions	42
6.1.2-1	Symbols Used	70
6.5-1	SCCELL Program (Step 1) Output Description	84
6.5-2	SCCELL Program (Step 2) Output Description	84
6.5-3	SCCELL Program (Step 2A) Output Description	85
6.5-4	SCCELL Program (Step 3) Output Description	85
6.5-5	MOSTRAN Program Output Description	86
8.6-1	Element Combination Statements	119
9.5.1-1	Solution of Equations Using Sesol.....	137
9.6.1-1	Calculation of Frequencies and Mode Shapes Using Determinant Search Method	139
9.6.2-1	Calculation of Frequencies and Mode Shapes Using Subspace Interaction Method	141

EXECUTIVE SUMMARY

The technology of microelectronics continues to evolve toward higher and higher density circuitry with increasing performance requirements in terms of speed and reliability. In the military electronic systems arena, only limited quantities of any device are produced and the performance requirements are always pushing against the forefront of technology. To facilitate the continual exploration and evolution of new designs for electronics to meet new threats, the Electronic Technology Division of the Air Force Avionics Laboratory maintains a Computer Aided Design (CAD) system, resident on the AVSAIL DEC System 10.

The CAD programs couple the computer to digital design, manufacture, test, and documentation to lower cost and increase productivity. This is accomplished by having the computer handle the routine time consuming tasks in order to allow man more time to be creative.

This manual presents a general functional description of each of the CAD programs. To fully understand the routine tasks performed by the CAD programs, one must have a working knowledge of the processes used in the design, fabrication, test, and documentation of digital electronics. To comprehend all the details of each program is somewhat overwhelming and generally unnecessary as the average user will only be concerned with a small subset of the overall system and need not be confused by the number of program options and details which exist. The primary purpose of this manual, then, is to give enough of a functional description of each program to allow a person to comprehend the basic fundamentals of that program in order to determine if indeed it will perform the tasks required. Once this decision has been made, then the user can proceed to a specific subset of Users Manuals rather than having to deal with the entire program set. Since considerable knowledge of the system under design, in addition to job planning and control, is required to make the most cost effective use of the CAD programs, the user should be prepared with the necessary information required to work within Computer Aided Design. Successful use of the system requires strict adherence to standards. Many decisions made in the early stages of the job are not easy to change later if they are found to be inadvisable. Adequate resources such as computer time, mass storage

media, and trained personnel must be planned for and provided at appropriate times in the cycle.

The programs available to the designer simulate a logic design, link the logic design to a physical design, determine testability of the design, and generate layout information. These programs encompass a broad spectrum of software tools ranging from custom Large Scale Integrated Circuit (LSI) designs at a cell level to a complete Printed Circuit Board (PCB) layout. The programs used in the design of PCB's are LOGIC 4, ASPEC, ASSIGN, PCPRA, SCELL/MOSTRAN, CONVRT, PRF, CHPCHP, and MCHPCK. The primary functions of these programs are: logic simulation, fault analysis, chip layout, logic element association, and printed circuit board layout and routing. In addition, programs are available for analyzing and optimizing microwave designs and for providing thermal and structural analysis of any design. These programs are: MAGNET, SAP IV and SINDA. The data interaction of all of the CAD programs is controlled via the data base management program LIBBER. The interaction and generalized capabilities of the programs are described in the brief functional descriptions that follow.

The simulation program (LOGIC 4) is normally the first and perhaps most important program used. Inputs to this program consist of a description of the logic design to be simulated. This may be individual logic blocks such as gates or LSI chips and includes an input sequence to exercise the circuit described. Outputs are many useful lists of tables and a timing diagram so the designer can verify that the circuit is performing as intended. A Verification mode is included in which the program compares the specified output signal levels for a good circuit to the same outputs for a faulted circuit to determine if there is a difference in the output. If a difference is noted a detected fault is listed. A Fault Isolation mode is included in which the actual circuit locations of the fault are determined through the analysis of unique fault signatures or signature sets for each fault detected. The Advanced Simulation Program for Electronic Circuits (ASPEC) is designed to perform nonlinear dc, nonlinear transfer function, nonlinear transient and linear ac simulation of circuits containing independent sources, linear elements and nonlinear devices.

The ASSIGN program is normally run after the design has been proven to be of sound logical structure, and serves as the introduction and the link to the physical design process. The program assembles the logical design information, previously supplied to the simulation programs, into lists for use in component interconnection on the board.

The PCPRA program is a printed circuit wire routing optimization program for production of two-sided or multilayer printed circuit boards. It consists of four parts that are normally used in order, yet can be used independently. The placement part (PLACE) selects a placement of the packages to be used on a PCB so as to minimize total printed wire length. The organizer part (WORGZ) accepts the output data from PLACE, organizes it, and formats it for input to the router part (PROUTE). The router performs printed circuit wire routing for two-sided or multilayer boards. The artwork (or printer) part (MLPLOT) presents a printer plot of the routing.

The MOS Cell Layout Program (SCELL) is a general purpose cell layout program for designing metal-oxide-silicon (MOS) cells. Based on a library of basic device configurations, defined for each new technology, it provides a rapid and accurate means of defining cell geometries; that is, the placement and interconnection of devices that form a cell. The SCCELL program is designed to interface with the MOS Circuit Transient Analysis (MOSTRAN) program for a circuit transient analysis test of the cell layout. The MOSTRAN program provides a simulation of circuit dynamic operations. This program is an automated design tool for use in analyzing MOS transient circuit performance and displays the output either as a list or as a plot of current (I), voltage (V), or power (P).

The chip layout programs CONVRT, PRF, CHPCHP, and MCHPCK are used to aid the design engineer in generating the data necessary for automated chip fabrication, starting with a logic design and utilizing cell definitions in the cell parameter library. The chip layout programs perform three primary functions: conversion of input data (CONVRT), chip layout with automatic placement (PRF), chip layout with user placement (CHPCHP), and checking of chip layout (MCHPCK).

The Microwave Network Analysis and Optimization Program (MAGNET) allows computerized analysis and optimization of a large class of linear microwave networks for gain, VSWR, phase linearity and noise figure. It is capable of modeling any network that can be built from pairs of linear two-ports. MAGNET is designed primarily for microwave networks; however, it may also be used for classical filter design, if suitable scaling is performed.

The Structural Analysis Program for Static and Dynamic Response of Linear Systems (SAP IV) is a computer program designed specifically for

structural analysis and has the capacity to analyze very large three-dimensional systems. The systems to be analyzed may be composed of combinations of a number of different structural elements such as three-dimensional truss and beam elements, pipe and boundary elements, two and three-dimensional solids, as well as plane stress and strain elements.

The Systems Improved Numerical Differencing Analyzer (SINDA) is a software system suited for solving lumped parameter representations of physical problems governed by diffusion-type equations. The system consists of two main pieces: (1) the preprocessor which accepts problems written in the SINDA language and converts them to the FORTRAN language, and (2) the library which consists of a collection of commonly needed pre-written FORTRAN subroutines.

The LIBBER program is a data base management program which is used for the maintenance of all standard data libraries and archive files. It provides the basic capability to store and retrieve data in standard card image format either as a stand-alone program, or as an integral part of many of the programs at both the cell and chip levels of design.

SECTION 1 SYSTEM OVERVIEW

1.1 INTRODUCTION

The Computer Aided Design (CAD) Users Manuals are reference manuals. They describe the details of format and functions of the computer programs that aid electronic hardware design engineers in the design, manufacture, test, and documentation of digital printed circuit boards (PCB's), hybrids and integrated circuits. The purpose of this section is to provide an overview which enables the users to understand the broad scope of the system without becoming involved with the actual program details. Users that require more detailed information on a specific topic should refer to the appropriate section describing the program. Table 1.1-1 is a listing of the CAD programs which are described and the corresponding section containing the detailed description.

TABLE 1.1-1 CAD Programs

SECTION	PROGRAM	PROGRAM FUNCTION
2.0	LIBBER	Data Base Management
3.0	LOGIC 4	Logic Simulation and Fault Analysis
4.0	ASSIGN	Element and Net List Generation
5.0	PCPRA	Multilayer Board Chip Placement and Routing
6.0	SCELL/MOSTRAN	Cell Layout and Transient Analysis
7.0	CONVRT/PRF/ CHPCHP/MCHPCHK	IC Chip Layout
8.0	MAGNET	Microwave Network Analysis and Optimization
9.0	SAPIV	Structural Analysis
10.0	SINDA	Thermal Analysis
11.0	ASPEC	Electronic Circuit Simulation

1.2 PROGRAM CAPABILITIES

The CAD system is a set of programs available to the designer to simulate a logic design, link the logic design to a physical design, determine the testability of the design, and generate layout information. These programs encompass a broad spectrum of software capability ranging from custom LSI design at a cell level to a complete PCB layout. In addition, programs are available for analyzing and optimizing microwave design and providing thermal and structural analysis of any design. The interaction and generalized capabilities of these programs are shown in Figure 1.2-1 and described in the general functional descriptions that follow.

1.2.1 Printed Circuit Board Design

There are three programs in the CAD system which are used in the design of PCB's. These are LOGIC 4, ASSIGN, and PCPRA. These programs perform four main functions which are:

- a. Logic simulation,
- b. Fault analysis,
- c. ELEMENT and NET LIST generation,
- d. Printed Circuit Board Layout and routing.

1.2.1.1 Logic Simulation

The simulation program (LOGIC 4) is normally the first used. Inputs to this program consist of a description of the logic design to be simulated. This may be individual logic blocks such as gates or LSI chips and includes an input sequence to exercise the circuit described. Outputs are many useful lists of tables and a timing diagram so the designer can verify that the circuit is performing as intended.

The simulation program can access a large library of LSI devices which have been defined previously and are documented in system libraries. The actual delays of the simulated devices do not automatically correspond to the physical devices; therefore, it is necessary to check critical timing paths in the design. The timing diagram represents a logic simulation in such a way as to check pulse by pulse (semi-static) operations.

1.2.1.2 Fault Analysis

The fault analysis option of the LOGIC 4 simulation program provides fault verification and isolation information for use in automatic board testing. The verification option uses the input sequence from simulation and the tagged ELEMENTS deck from ASSIGN to determine the percentage of possible user specified faults on the board which can be detected at the output connector. The user specifies the type of faults to simulate (inputs and/or outputs stuck at one and/or zero). The isolation option expands on these results to create a list of faults which can be isolated to a user specified number of IC's. If many fault conditions look the same at the I/O connector, it is impossible to determine which of the possible faults is occurring on the board. Therefore, the testability of the design is enhanced by judicious test-point selection and an input sequence which properly exercises the logic.

1.2.1.3 ELEMENT and NET LIST Generation

The ASSIGN program serves as the introduction and the link to the physical design process. The program assembles the logical design information, previously supplied to the simulation programs, into string-lists (NET LISTS) for use in component interconnection on the board. The program is capable of doing a random layout constrained only by the order in which the data is arranged in the card deck. The ASSIGN program does not determine geometry; therefore, placement and routing information is dependent upon the manual specifications of the designer.

The data-base necessary for ASSIGN consists of a library of physical characteristics of the LSI being simulated. These parameters are used to prepare the net lists, a dc loading table for all signal names on the board, a computer written parts list, and a table of unused logic and logic that does not fit.

The ASSIGN program allows the user to punch cards or create disk files of all output data which is used in other CAD programs. These outputs may be:

- a. A tagged ELEMENTS file which has the position for each element in the tag field.
- b. A string-list for board interconnection for use in the PCPRA placement and organizer functions.

1.2.1.4 Printed Circuit Board Layout

The PCPRA program produces the physical layout data for two-sided or multilayer printed circuit boards. This program consists of four parts which are normally run in sequence but may be used independently. The first part of PCPRA is the placement function (PLACE) which accepts the BUCKET and STRING-LIST data from ASSIGN, TYPE-MODEL and MODEL data from LIBRARIES, and user input data then selects the placement of the component modules to be used on the PCB. This program optimizes the placement of the modules in order to minimize the total printed wire length and produces a GEOMETRY DATA file formatted for input to the second phase.

The second phase (WORGZ) accepts the output data from PLACE and the STRING-LIST data from ASSIGN, organizes the data and formats it for input to the third phase. The output from WORGZ is a COVERFILE consisting of GEOMETRY DATA reformatted into SMIP statements, and an EXTENDED NET LIST which is computer ordered for routing.

The third phase (PROUTE) performs printed circuit wiring for two-sided or multilayer boards. PROUTE accepts the COVERFILE from WORGZ and produces a SOLUTION FILE which may be used by a plotter to produce the actual PCB. This file may also be applied to the fourth phase of the program to produce a computer printer plot.

The printer plot output from the last phase (MLPLOT) shows the placement of all component modules and the interconnecting wiring for each layer produced by PROUTE. MLPLOT will also accept the COVERFILE output from WORGZ and produce a printer plot showing only the placement of the component modules.

1.2.2 Custom LSI Design

The majority of PCB design performed by the users of the CAD programs can be accomplished using standard TTL logic; however, the system also provides the capabilities to design custom LSI chips for an even broader spectrum of electronic equipment application.

1.2.2.1 The Basic Approach

The basic approach to custom LSI design is based on the cell system. A cell is any arbitrary partition of the design where primary emphasis is placed on circuit design. LSI chips are constructed of cells with primary attention being given to logic design.

Standard Cells are families of circuits which are designed to meet the requirements of a series of similar equipment. They are designed to ensure that the circuits perform over a wide range of voltage, speed, temperature, and drive requirements. The two cell families implemented with the custom LSI design program (dynamic two-phase silicon gate PMOS and static single-guarded metal gate CMOS) are examples of standard cell families with the additional advantage of being designed around industry standard design rules.

Custom Cells are circuits which are designed to meet specific objectives such as high performance, high volume production, or implementation of a unique or unusual process. In this case, cells are designed for a specific chip where performance and size are optimized for the particular chip design. Using custom cells, the design begins at the circuit level, progressing through artwork generation largely under manual control.

1.2.2.2 Cell or Circuit Level Design

The CELL program is a general-purpose cell layout program. Based on a library of basic device configurations, defined for each new technology, it provides a means of defining cell geometries. Design rules including width and spacing, both for masks taken individually, and for other unique combinations of masks taken together, are checked.

Interconnect nets are checked to ensure connectivity, to detect shorts, and to identify extraneous material. During the net check, process variables such as latent diffusion and mask misalignment are simulated; and electrical parameters, including actual device geometries, load capacitances, and coupling capacitances, are calculated. These electrical parameters are used as input to the transient circuit analysis program, MOSTRAN, which provides a simulation of the dynamic operation of the circuit.

The Cell level design aids may be used for either the Standard Cell or the Custom Cell approaches to LSI design. Limitations of design complexity affixed to these programs are computer run time and core limiting factors.

The basic limitation in the CELL program is 35 levels of artwork, four of which may be interconnect. Built-in models in the MOSTRAN program are restricted to MOS, but models may be added by writing Fortran subroutines to simulate other technologies.

The final output of the cell level system is a Technology Data Base to be used in subsequent chip design. First, all input data to CELL and MOSTRAN is retained for future analysis or modification of any cell design on a master design file. Second, the details of the geometries internal to the cells are saved on a master geometry library. A third file, the master parameter library, contains the terminal characteristics of the cells; that is, all the information necessary for chip level programs to lay out, modify, analyze, and check chip layouts. Finally, a fourth file, master MACRO library, contains MACRO or subroutine descriptions, test sequence generation, and fault analysis at the chip level.

1.2.2.3 Chip or Logic Level Design

Relying on the Technology Data Base, either for a standard cell family or for a completely custom design using a custom cell family, a number of programs are available to simulate, layout, and analyze LSI chip designs. The system logic analysis program, LOGIC 4, is a general-purpose program for the logic simulation and fault analysis of LSI systems. It is a hierarchical, functional simulator capable of simulation or analysis at any level of design from initial equipment specification down to detailed logic.

Once a designer is satisfied with the functional correctness of his design and has assured himself that the design is testable, chip layout may begin. For standard cell designs, the automatic layout program, PRF, places and wires his cells in accordance with the design rules for that technology. The user has the option of using fully automatic placement, manual placement, or a combination of the two. Output from the PRF program includes final placement, net loading, and detailed chip geometries, which are all cataloged for that chip design under the Equipment Data Base.

For a Custom Cell layout, a program called CHPCHP is provided for short-hand input of placement and routing. This program may be used to originate layouts of unique or unusual designs, or to alter or append layouts generated by PRF. It also interfaces with the Equipment Data Base and other related programs via the detailed chip geometries.

To ensure accuracy and complete conformity to design rules, a checking program called MCHPCK is provided. Here, exhaustive net checks are performed comparing the original LOGIC 4 definition of the chip against the final chip geometries. The total LSI chip design capability is limited to 300 functional cells.

1.2.3 Microwave Network Design

The MAGNET program allows computerized analysis and optimization of a large class of linear microwave networks for gain, VSWR, phase linearity and noise figure. It is capable of modeling any network that can be built from pairs of linear two ports. The program is used to minimize input VSWR, output VSWR, phase deviation from linearity, noise figure, and gain error, in any combination, for a network in a 50-ohm system. MAGNET is designed primarily for microwave networks; however, it may also be used for classical filter design if suitable scaling is performed. The MAGNET program is used primarily for design of linear two port microstrip; however, it may also be used in stripline and lumped element design.

1.2.4 Structural Analysis

SAP IV is a computer program designed specifically for structural analysis and has the capacity to analyze very large three-dimensional systems. It has available a new variable-number-nodes thick shell and three-dimensional element, and out-of-core direct integration for time history analysis. The structural systems to be analyzed may be composed of combinations of a number of different structural elements. The program presently contains the following types:

- a. Three-dimensional truss element,
- b. Three-dimensional beam element,
- c. Plane stress and plane strain element,
- d. Two-dimensional axisymmetric solid,
- e. Three-dimensional solid,
- f. Variable-number-nodes thick shell and three-dimensional element,

- g. Thin plate or thin shell element,
- h. Boundary element,
- i. Pipe element (tangent and bend).

These structural elements can be used in a static or dynamic analysis and are carried out in the same manner. The static analysis is continued by solving the equations of equilibrium followed by the computation of element stresses. In a dynamic analysis the choice is between:

- a. Frequency calculations only,
- b. Frequency calculations followed by response history analysis,
- c. Frequency calculations followed by response spectrum analysis,
- d. Response history analysis by direct integration.

1.2.5 Thermal Analysis

SINDA, the Systems Improved Numerical Differencing Analyzer, is a software system suited for solving lumped parameter representations of physical problems governed by diffusion-type equations. The system consists of two main pieces: (1) the preprocessor, and (2) the library. The SINDA preprocessor is a program which accepts problems written in the SINDA language and converts them to the FORTRAN language. The SINDA library consists of many pre-written FORTRAN sub-routines which perform a large variety of commonly needed actions and which reduce the programming effort which might have been required to solve a given problem.

One of the most outstanding features of SINDA is that it accepts "program-like" logic statements and subroutine calls as data which, ultimately, permit the user to tailor the program to suit his particular problem.

1.2.6 Data Base Management

The overall system relies heavily on installation dependent software for many functions relating to the creation and maintenance of file structures, bulk movement and manipulation of data, and for on-line editing and program execution. Use of these functions requires a reasonable familiarity with the computer system being used and will not be treated in detail here.

The LIBBER program is a machine independent data base management program which is used for the maintenance of all standard data libraries and archive files. It provides the basic capability to store and retrieve data in standard card image format either as a stand-alone program, or as an integral part of many of the programs at both the cell and chip levels of design. All on-line data storage is maintained in packed form to optimize the use of disk space. Off-line storage is expanded to standard 80-column card images for permanent storage or for intermachine transfer.

1.3 FILE MANAGEMENT

The CAD system programs perform several interrelated tasks and require frequent interaction for the complete process. It is essential; therefore, that the user have a complete list of the unit numbers and files associated with each program.

1.3.1 File Assignments

To access any of the AFAL CAD programs the user must first use the required system commands to initiate system operation, after which each program must be accessed by the appropriate program select command. When a program is accessed the system responds with a request for file assignment (UNIT, FILE?) at which time the user enters the appropriate commands to reference all files accessed by the program for input data of files created to store output data.

1.3.2 Command Characteristics

One or more files may be assigned by the user; however the END field is used only after the last file assignment is made. If the optional END field is not used on the last assignment, an END must appear as the next input. If UNIT 5 is assigned, it must be the last unit number assigned in a series of assignments as the system will then expect all following inputs to be on logical UNIT 5. In addition, the END statement must appear in the appropriate field on the data entry for UNIT 5 assignments otherwise the program will not terminate the file assignment mode.

When a program is initiated, it uses UNIT 5 for the input file and UNIT 6 for the output file. After each file assignment terminated by the DONE and carriage return, the system responds with another request for file

assignment. By reassigning UNIT 6 all program outputs are put on the assigned disk file. The user must therefore enter the next file assignment without any additional prompting.

The user must define all input files he will require during that session. In addition, the user should define all output files required. Assignment of output files are not required; however, if not assigned the system creates a FORTRAN file name that may be destroyed by any other program.

Upon completion of a program run, the program releases all file assignments and returns to the monitor mode. Units 5 and 6 are then returned to their normal input/output peripherals. The system then prints any remaining user information on UNIT 6.

SECTION 2 DATA BASE MANAGEMENT PROGRAM—LIBBER

2.1 INTRODUCTION

The programs contained in the CAD system requires frequent interaction for a complete circuit design; therefore, it is essential that a data base be established that can be called upon for many applications. The LIBBER program provides for this data base. Figure 2.1-1 depicts the functional relation between the LIBBER program and the remaining CAD programs. This section provides sufficient information to describe the purpose and capabilities of LIBBER and a general description of the various commands and options available for use. Users need have no programming experience to use the LIBBER program although some background is helpful.

2.2 PROGRAM PURPOSE

The LIBBER program is used to perform such conventional maintenance tasks as addition, deletion, or modification of subfiles used with other programs in the AFAL-CAD system. The program also allows control over the use of the data on any CAD system file. The program provides aids for the associated CAD system programs, such as displaying data-category and card-image heading to assist in coding input categories.

2.3 PROGRAM GENERAL DESCRIPTION

The LIBBER program is a machine independent, data-base management program that uses random access reads and writes to build a LIBBER file and may be run in either the interactive or batch mode. Data is stored in LIBBER files in a packed format to make maximum use of disk space. The data stored on the file can be restored to 80-column card-image format for use by other CAD programs or intermachine transfer.

The LIBBER program is capable of saving the data base information on backup tape as off-line storage. The program is used primarily for creating new subfiles or manipulating existing subfiles as follows:

- a. Creates new subfiles.
- b. Edits existing subfiles such as:

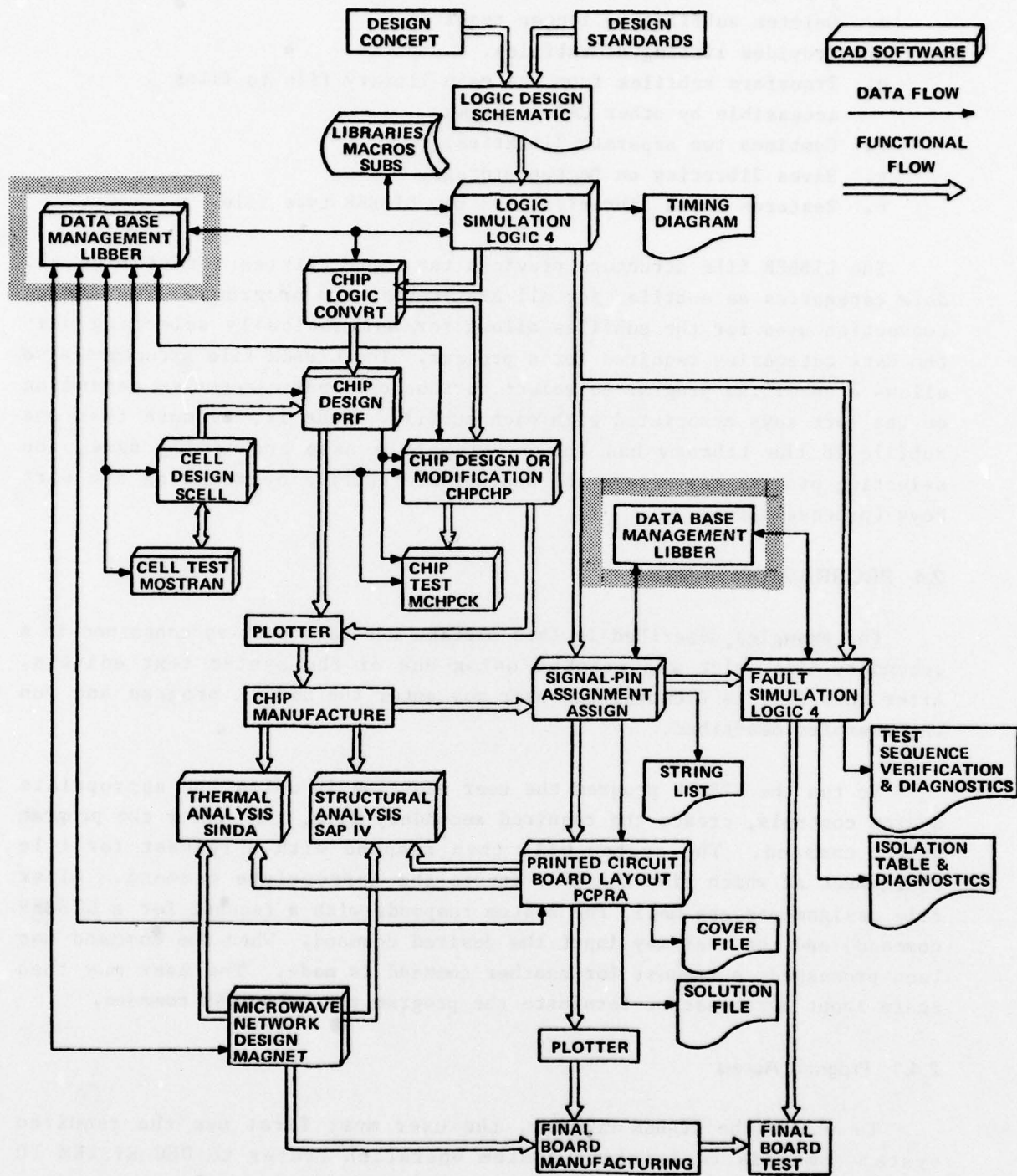


Figure 2.1-1. CAD Functional Block Diagram Emphasizing the LIBBER Program

- (1) append additional data,
- (2) performs line editing.
- c. Deletes subfiles no longer required.
- d. Provides listing of subfiles.
- e. Transfers subfiles from the main library file to files accessible by other CAD programs.
- f. Combines two separate libraries.
- g. Saves libraries on backup storage.
- h. Restores saved libraries to a new LIBBER type file.

The LIBBER file structure provides the capabilities for storing all data categories as subfiles for all AFAL-CAD system programs. The naming convention used for the subfiles allows for automatically selecting only the data categories required for a program. The LIBBER file structure also allows a specified program to select various optional categories depending on the sort keys associated with each subfile. That is, if more than one subfile in the library has the same subfile name and family name, the selecting program may select the appropriate subfile by matching the sort keys (process names).

2.4 PROGRAM OPERATION

The examples described in this section use data entries contained in a secondary file which was created using one of the system text editors. After this file is created, the user may enter the LIBBER program and run the examples described.

To run the LIBBER program the user must log in using the appropriate system controls, create the required secondary file, then enter the program select command. The system will then respond with a request for file assignment at which time the user inputs the appropriate command. After file assignments are made, the system responds with a request for a LIBBER command, and the user may input the desired command. When the command has been processed, a request for another command is made. The user may then again input a request or terminate the program run by an END command.

2.4.1 Program Access

To access the LIBBER Program, the user must first use the required system commands to initiate system operation (refer to DEC SYSTEM 10

Operating System Command Manual DEC-10-OSCMA-A-D). After the system has been initiated, the LIBBER program is accessed by a program select RUN command.

2.4.2 LIBBER File Creating Commands

To create a new LIBBER file two types of commands are required. The first type establishes the file name and file type of the LIBBER file and the input file. The second type initializes the LIBBER file and sets up the file structure. These are both performed through use of the UNIT, FILE command.

2.4.2.1 File Assignment

Two file assignment commands are used which provide the user access to a library file to logical Unit 15 with a file name selected by the user and enables the LIBBER program to access the secondary file. The name assigned to the LIBBER file may be any legal operating system name and extension. The commands also define the file types.

All LIBBER commands manipulate the file on logical Unit 15 unless preceded by a unit number change command. Therefore, when creating a LIBBER file the unit number specified should always be Unit 15. In addition, all LIBBER files are RANDOM access and should be specified as such by the file type input.

2.4.2.2 File Initialization

The file assignment commands are followed by an initialization command that determines the file size and structure by allocating a specified amount of file space for the Library. Initialization is performed by use of the NEW command.

2.4.2.3 Subfile Creation

The ADD command creates new subfiles on the master file, initialized by the NEW command. The subfiles are created from data contained in the secondary file. Files may also be created from data supplied by the user immediately following the command input.

2.4.2.4 File Monitoring Commands

The LIBBER program provides two file monitoring type commands (TOC and LST) that enable the user to obtain a printout of the master file table of contents or a listing of an individual subfile. The line numbers are program generated during the LST process and are not part of the actual subfile. These may be used for editing purposes using the LIBBER edit function.

2.4.3 LIBBER Exit Command

To exit the LIBBER program the user must input an END command when the program is expecting a LIBBER type command. This terminates the LIBBER program execution at which time the user may enter another CAD program, perform system operations, or log off the system.

2.5 LIBBER COMMAND DESCRIPTIONS

The following sections provide a general description of the LIBBER program functions by documenting the use and capability of each LIBBER command.

2.5.1 RUN LIBBER Command Description

The RUN LIBBER command is the program select command that accesses the LIBBER program and allows the user to perform the various functions allowed.

2.5.2 File Assignment Command Description

The UNIT, FILE command defines which files and associated units are assigned or accessed by a particular LIBBER program run. Each command specifies the unit number, file name, and file type for each assignment.

The user must define all input files he will require during that session. In addition, the user should define all output files required. Assignment of output files is not required; however, if not assigned the system creates a FORTRAN file name that may be destroyed by any other program. For example, if the user outputs to UNIT 23 the file created will be FOR23.DAT and any other user that writes on UNIT 23 will destroy the previously created file. Possible needs for file assignments are:

- a. LIBBER Files (Random access only).
- b. Input files.
- c. Output files.
- d. SAVE files (read/write).
- e. Restore files.
- f. Files to be merged.

The program default units for files not assigned are:

- a. Unit 15 (LIBBER files).
- b. Unit 5 Input.
- c. Unit 6 Output.

2.5.3 HEL Command Description

The HEL (Help) command is provided only as an aid for the inexperienced user. This command provides a listing of all the LIBBER commands available.

2.5.4 TTY Command Description

The TTY command, used only in the interactive mode, causes the program to eliminate echoing reprints of operator supplied inputs. The default mode of the LIBBER program is to reprint each input command prior to execution of that command. In addition, the system prints all card image inputs to the program. For example, when the ADD command is used to create LIBBER subfiles from a secondary file, each input card image of the subfile is printed on the terminal. The TTY command eliminates this print out, thereby, speeding up terminal interaction.

2.5.5 TOC

The TOC (Table of Contents) command prints out a complete list of all subfiles, the position of each subfile on the master file unit, and date and time of each subfile creation. The printout gives a summary of the master file unit configuration, number of data blocks, size of data blocks and number of words per table of contents entry.

The LIBBER files are in packed format and the system has 36 bits per word with seven bits per character. Therefore, the LIBBER files contain

five characters per word in the packed form. The TOC print out lists the block number containing the subfile, the first word number of the subfile, and the number of characters in each subfile.

The DATE TIME reading on the TOC is the creation date time record for the subfile. This may be the original creation record or the update record if a RES function is performed.

2.5.6 NEW Command Description

The NEW command is used to initialize a newly created LIBBER file allocating a specific amount of disk space for the master file. This must be the first LIBBER command following the file assignment commands when a new file is to be created.

Default allocation creates 7 data blocks, 1024 words per block, and 14 words per TOC entry in the TOC block. If defaults are not acceptable, the NEW command should be specified with a Start block and End block operand.

The Start block for a new LIBBER file should always be one. The END block is the total number of blocks desired for that file.

Additional data blocks can be appended to a previously established LIBBER file by specifying the Start block as one less than the existing number of data blocks in the master file. The Ending block would then be specified as the new total number of blocks desired after NEW is complete.

The EXTENDABLE NEW command is used to append additional data blocks to the last unused (empty) data block of an existing file. This command can not be used on files that do not have an empty data block. To add data blocks in this case the SAV command and RES command should be used.

2.5.7 ADD Command Description

The ADD command creates new subfiles on a master library file from data supplied by the user immediately after the ADD, or from a peripheral unit specified by the ADD. The LIBBER program classifies the subfile as a device or as a data category dependent upon the first data input. Naming of the subfile is dependent upon its classification.

The program provides the capability of creating multiple data category subfiles with a single ADD command. However, multiple subfile creation is not possible for categories that are classified as devices. An ADD command must be provided for each device subfile added from either the user peripheral or the storage peripheral. In addition, if a device data record follows multiple data categories on a file, the ADD function is terminated by the data record and the device is not added to the master file.

2.5.8 APP Command Description

The APP (Append) command appends data supplied by the user to existing subfiles on the master file immediately after the APP, or from a peripheral unit specified by the APP.

Command format of the APP command is similar to that of the ADD command. Operands of the APP command must correspond exactly to those established by the ADD command. The TOC command may be used to obtain a list of the subfiles contained in a master file.

2.5.9 DEL Command Description

The DEL (Delete) command deletes specified subfiles from the master file. The DEL command operands must correspond exactly to those established by the ADD command. The TOC command may be used to obtain a list of the subfiles contained in a master file.

2.5.10 LST Command Description

The LST (List) command produces a listing, with appended line numbers, of a specified subfile contained in a master file. This command provides the user with a list of the subfile title, date and time of creation, and a card-image listing of the subfile. Line numbers are also affixed to the list that can be used for the EDT function. The LST command operands must correspond exactly to those established by the ADD command.

2.5.11 GET Command Description

The GET command produces a card-image listing of a subfile on a peripheral storage unit. This command provides the user with a file for editing purposes or for input to any other AFAL-CAD programs. Subfiles may

be assigned sequentially to a storage unit by means of successive GET commands.

The GET command operands must correspond exactly to those established by the ADD command. The unit number and associated file name must be defined by the file assignment command and there must be one GET command for each subfile listed on the peripheral unit.

2.5.12 SAV Command Description

The SAV command saves a complete LIBBER file, subfile-by-subfile, on a user specified peripheral storage unit for backup storage or editing only and not for use with other programs. SAV offers the ability to obtain a card image print out of the LIBBER file. The date and time record of each subfile may also be optionally saved.

The SAV command creates one ADD type card-image command for each subfile in the LIBBER file. The date/time record request also produces a one word record, preceding each ADD, to save the subfile creation date and time. Following the ADD card-image, are the data records for each item in the subfile.

2.5.13 RES Command Description

The RES command creates a LIBBER file from a card-image file contained on a peripheral storage unit. This function is normally used with a file produced by the SAV function. However, any file containing the appropriate ADD type card-images and date/time records may be restored.

If a file is saved without the optional date/time record, the restore must be performed without the record. If however, the file is saved with the date/time option, the restore may be performed with the record but is not required. When the option is used, the date and time of the subfile creation is restored with each subfile. If the option is not used, the date and time of restore is attached to each subfile.

The RES command creates a NEW type command to initialize a new LIBBER file. The library size is determined by the start block, end block, words/block, and words/TOC parameters specified by the RES command. This enables the SAV and RES commands to be used to extend the size of an existing LIBBER file when all data blocks contain data and the extendable NEW cannot be used.

If the restore is performed on the library unit (Unit 15), only the RES command is required. If the restore is performed on another unit, the RES command must be preceded by a UNT command. In addition, the file name associated with this unit must be defined by the file assignment command.

2.5.14 MRG Command Description

The MRG (Merge) command creates a new LIBBER file from two existing LIBBER files stored on separate peripheral units. The new file is written on a third unit on a subfile-by-subfile basis.

The merge function is performed on a subfile-by-subfile basis. If duplicate subfile names are encountered, an automatic delete is performed on the subfile with the earliest date/time record. The most recent subfile is entered on the merged file unit.

2.5.15 EDT Command Description

The EDT (Edit) command allows basic line editing functions of the subfiles contained in a master file. The program performs the functions specified by the text-editing input immediately following the EDT command.

The EDT command functions to delete text, insert new text, or a combination of both. The functions are performed on a complete subfile record rather than on individual characters. The text-editing inputs reference the line numbers appended to the card-images by the LST command.

The text editing mode is terminated by an END command, thus allowing multiple text-editing functions. Because the END terminates the EDT function, addition of an END is not allowed.

An additional feature of the text-edit function is the (=) command. This is used when data is added to a subfile whose first non-blank character is a minus sign, such as -12v. The program would normally interpret the -12v as an update command to insert new data after line 12. The (=) command allows the update character (-) to be changed to any other character. After this command the update character is that specified and remains the new character until changed back by another occurrence of the (=).

Multiple updates made to a subfile must be made in ascending line number order. For example, the user could not delete line 19 and later insert data after line 6. The program performs diagnostic verification on the text-editing functions and restores the non-edited subfile if an error is made. These errors may be updates made out of order, updates made beyond the subfile limits, or updates terminated without an END command.

2.5.16 REN Command Description

The REN (Rename) function allows an existing subfile to be renamed by a second input containing a new subfile and/or family, process, and password names. The REN function only changes the subfile names and has no effect on the data contained in the subfile.

2.5.17 UNT Command Description

The UNT (Unit) command changes the library unit number (Unit 15) on a one time basis. That is, this command only applies to the LIBBER command immediately following the UNT input, and each subsequent LIBBER command references the original unit.

2.5.18 REW Command Description

The REW (Rewind) command rewinds a user specified logical unit (tape or disk). For example, the REW command may be used prior to a RES command to reposition the peripheral that contains the saved file.

2.5.19 PRT Command Description

The PRT (Print) command specifies a particular unit to be the print unit for all LIBBER outputs. When used all outputs produced are printed on the unit specified.

2.5.20 Card Headers

The card header options are special features of the LIBBER program that are used in the interactive mode. These commands provide the user with grid images of the card fields used in the associated CAD program data categories. The commands and a brief description are listed in table 2.5.20-1.

TABLE 2.5.20-1 Card Holder Options

COMMAND	USE
\$ARR	PCPRA Array category.
\$BOA	PCPRA Board category.
\$CAR	ASSIGN Card Description.
\$DIS	PCPRA Description category.
\$ELE	LOGIC \$, ASSIGN, OR CONVERT Element description category.
\$LOC	PCPRA Local category.
\$NET	PCPRA Net List category.
\$PLA	PCPRA Place category.
\$PRF	Placement Router Function PRF input categories.
\$XTR	PCPRA External Connectors.
\$(SM	PCPRA SMIP category.

SECTION 3 LOGIC SIMULATION AND FAULT ANALYSIS PROGRAM—LOGIC 4

3.1 INTRODUCTION AND GENERAL DESCRIPTION

LOGIC 4 is a Computer Aided Design (CAD) logic circuit analysis program to be used for circuit simulation, fault detection, isolation, and verification. The program is written in a high level language and does not require previous programming experience for use. By use of full event-oriented simulation, LOGIC 4 allows verification of basic circuit concepts, fundamental machine structure, and potential interface and timing problems. The ability to simulate all or part of an equipment at any level of refinement builds confidence in the ultimate correctness of the final product. Figure 3.1-1 depicts the functional relation between the LOGIC 4 program and the remaining CAD programs.

The LOGIC 4 program provides logic simulation and fault analysis functions by means of three basic functions. These functions are:

- a. Stimulus/Response Simulation
- b. Test Stimulus Completeness Verification
- c. Fault Isolation

A basic concept imbedded in LOGIC 4 is a hierarchical simulation technique which assimilates many levels of complexity within one automation system. The LOGIC 4 system is based on entities called logic blocks. A logic block is a functional model representing an arbitrary portion of an equipment. Logic blocks may be defined to represent any convenient partition including analog circuitry, logic circuitry, memory, or even software. Logic blocks may be expanded into more refined functional operations, such as more detailed logic blocks, or eventually into models for basic gates of which an equipment is to be constructed if time and core requirements are not too severe.

Logic analysis may be performed at any level. The system may be simulated as a collection of logic blocks, with one or more of the blocks expanded, or by comparing any logic block against its expansion. The

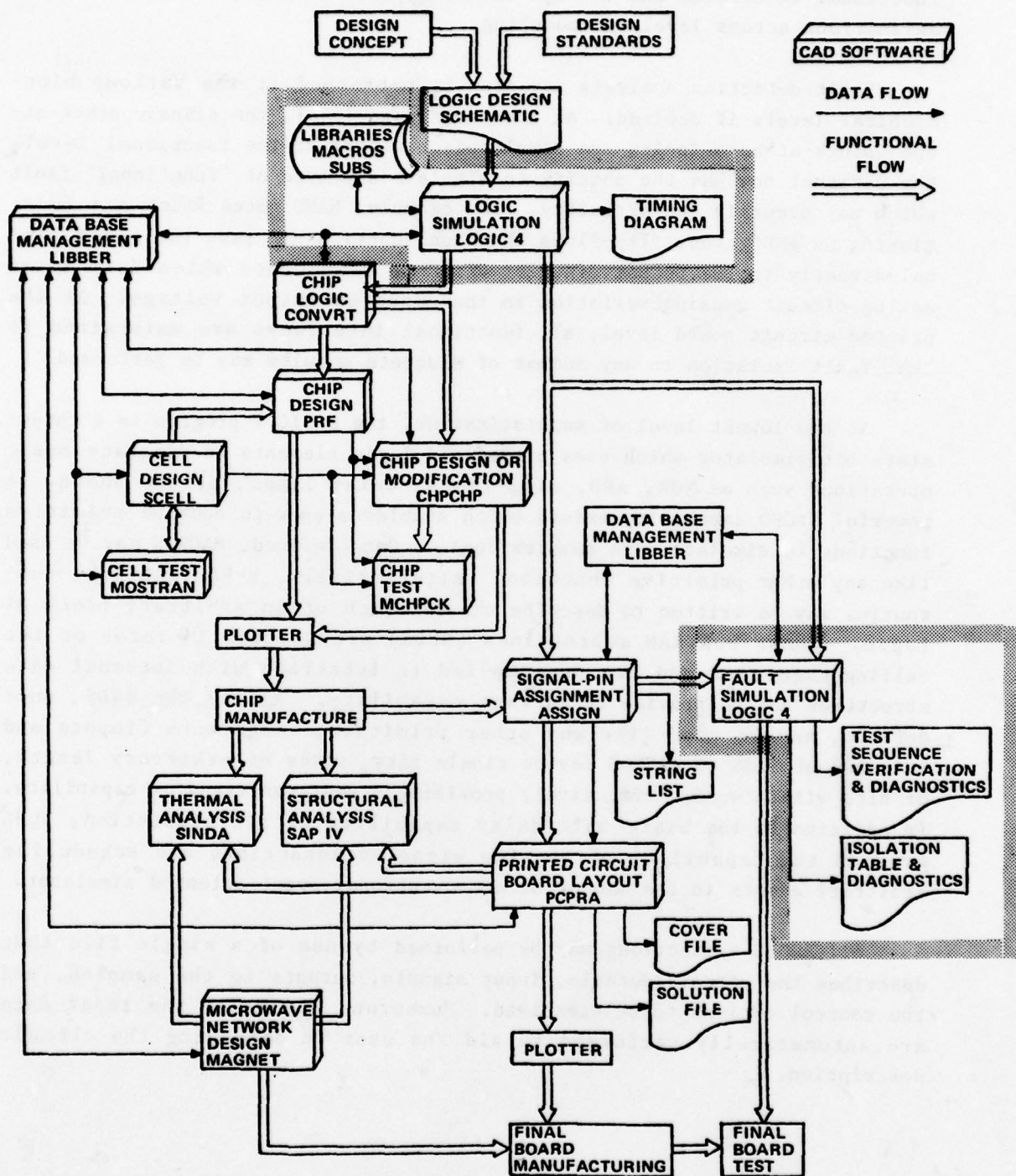


Figure 3.1-1. CAD Functional Block Diagram Emphasizing the LOGIC 4 Program

accuracy of the simulation is maintained through automatic handling of functional interfaces and through automated aids for comparing functional definitions across levels of modeling.

Fault detection analysis may also be performed at the various hierarchical levels if desired. At the basic gate level, the classic stuck-at-one, stuck-at-zero faults are simulated; however, at the functional level, the designer now has the ability to simulate any type of "functional" fault which may occur in the circuitry. For example, NAND gates which are functioning as AND gates, flip-flops which malfunction and pass the clock signal directly to the output line, or discrete components which fail in an analog circuit causing variation in the expected output voltage. At the printed circuit board level, all functional interfaces are maintained so that fault isolation to any number of discrete modules may be performed.

At the lowest level of sophistication, the LOGIC 4 program is a three-state bit simulator which uses predefined logic elements to simulate basic operations such as NOR, AND, flip-flops, delay lines, or one shots. A powerful MACRO capability exists which enables a user to combine primitive functions to simulate more complex logic. Once defined, MACROS may be used like any other primitive function. Alternatively, a FORTRAN-like subroutine may be written to describe the function of an arbitrary block of logic. These FORTRAN subroutines (SUBS) are written in terms of the calling arguments and are precompiled to interface with internal data structures and to provide re-entrant capability. Again, the SUBS, once defined, may be used like any other primitive. Arguments (inputs and outputs) of SUBS or MACROS may be single bits, words of arbitrary length, or bits within words, effectively providing a register transfer capability. In addition to the basic unit delay capability of bit simulation, SUBS provide the capability of sensing signal transactions and scheduling arbitrary events in the manner of a conventional event-oriented simulator.

All LOGIC 4 functions may be performed by use of a single file that describes the circuit details, input signals, outputs to be sampled, and the control options to be exercised. Numerous checks of the input data are automatically performed to aid the user in debugging the circuit description.

LOGIC 4 offers the following advantages:

- a. Saves design development costs by finding design errors before hardware is fabricated.
- b. Provides documentation for the actual design.
- c. Tabulates hardware requirements in the accounting statistics for each type of element and the loading of each element.
- d. Facilitates repair or replacement of hardware by isolating faults to the desired component level.
- e. Redundancies in the circuit design may become apparent to the designer as the circuit is coded for program input.
- f. Allows the designer to see many signals simultaneously on a timing diagram output as opposed to the limited number of traces available on oscilloscopes.
- g. The timing diagrams printed by the computer often show logic spikes, which may not be apparent on oscilloscopes.
- h. Top down design is possible through the functional model capability. Simulation is possible even though the hardware implementation has not been designed.
- i. Subroutines allow the simulator to be technology independent.

The program can simulate systems containing "Block Boxes" specified by their terminal behavior only through the use of FORTRAN subroutines. As the detailed design of these boxes becomes available, the subroutines can be replaced by the actual element descriptions, thus allowing design verification in a controlled way and of successively finer levels of detail.

3.2 FUNCTIONAL DESCRIPTION

Before a design is committed to hardware, it can be analyzed to determine design validity. This is accomplished by simulating the circuit and comparing the response to the expected response for a given input. The design may be changed and resimulated until the simulated outputs agree with the expected outputs. Later in the design process, the completeness of a given test plan can be determined by means of the fault simulation capability of LOGIC 4.

3.2.1 Major Functions

The major functions of LOGIC 4 are as follows:

- a. Stimulus/Response Simulation in which the program accepts a description of the logic circuit (ELEMENTS), the circuit input signal specifications (EXTERNALS), simulates the logic operation of the circuit, and produces a timing diagram showing the state of each sampled output (OUTPUTS) at specified times.
- b. Test Stimulus Completeness Verification in which the program compares the specified outputs signal levels for a good circuit to the same outputs for a faulted circuit to determine if there is a difference in the output. If a difference is noted, a fault is considered detected. This mode also checks the user test sequences for completeness of verification.
- c. Fault Isolation is the mode in which the actual circuit locations of the fault are determined through the analysis of unique fault signatures or signature sets for each fault detected.

3.2.2 Stimulus/Response Simulation

The purpose of this simulation function is to verify that a design is performing properly prior to committing the design to hardware manufacturing. This verification process is accomplished by calculating the circuit outputs for a given set of circuit inputs. These calculated or simulated outputs can be compared with the outputs desired by the designer. If the desired results are not obtained, the circuit can be modified and re-simulated until the desired outputs are obtained.

The circuit design is defined in terms of logic elements whose inputs and output signals have three possible states: logic 0, logic 1, or don't know (X). The program calculates the outputs for each building block element for any set of inputs. The simulator functions by applying a set of circuit input signal values to the design as externals. The value of each input signal is compared with the previous value of that signal. Any signal whose value has changed is put on a list. Then all elements which are fed by signals on the list are simulated and the output signal values of those elements are compared to their previous values. Any signals which have changed are put on a new list. Then all elements fed by the new entries are simulated. This process continues until no element outputs change or until the number of such repetitions exceed a user supplied threshold. After the threshold is reached, all signals which are still trying to change are set to the Don't Know state (X). This can result from actual circuit oscillation or because the threshold was too low. When the simulation process does terminate for a given input, the resulting signals

on all the elements are used as the initial conditions for the next set of input values applied. The sequence of element calculations may be thought of as a "wave front" emanating from the inputs which change. All elements in a particular wave are considered to change state at the same instant of time. Adjacent wave fronts are one unit of time apart (one gate time).

3.2.3 Test Stimulus Completeness Verification

The LOGIC 4 program simulates faults within a logic design by changing signal interconnections according to the type of fault. An open input to an AND gate is simulated by reconnecting that input to logical one. An element output stuck fault is simulated by altering the description for the element so that the output signal always remains at the stuck value. The circuit, altered for a particular fault, is then processed in the same manner as a fault free circuit. The only additional action that takes place is the comparison of outputs with those of the fault free circuit and cataloging of the fault signatures.

This function verifies the completeness of the test sequence to ensure that faults in the logic design can be detected at the circuit outputs. Faults are defined as element input or output signals stuck at a logic zero or logic one. In this mode, the program simulates the circuit as if the requested faults were present. Faults are considered detected when the output for a faulty circuit differ from that of a fault free circuit. The user can select the signals or signal classes to be faulted according to element inputs, element outputs, externals, signals internal to complex devices, or individual signal names.

3.2.4 Fault Isolation

The Fault Isolation mode determines the actual location of the fault in reference to individual elements. This function is used to produce specific diagnostic or hardware location information for the detected faults. This provides the user with the means to further evaluate test sequence and the selection of test points. This is important since electrical circuits are typically composed of replaceable modules and fault isolation may be specified in terms of the number of replaceable modules indicated by a given faulty output. LOGIC 4 handles this problem automatically by analyzing the data produced during fault simulation. The number of modules is defined by the user. For each fault signature, at each coarse time, the program compares the number of replaceable modules

containing the faults with the isolation criteria. Isolation can be accomplished if the number is less than or equal to the criterion. If comparison is greater than the criterion, simulation of these faults is continued.

3.2.5 Program Timing

The functions performed by the LOGIC 4 program operate in relation to simulation time. Effective use of LOGIC 4 depends on a clear understanding of how time is handled within the program. The following paragraphs define the basic unit of simulation time and the various multiples of this unit. The basic time units used in LOGIC 4 are:

- a. Gate Time
- b. Coarse Time
- c. Gates/Coarse Time
- d. External Time
- e. Output Time
- f. Strobe Time.

3.2.5.1 Gate Time

The basic task involved in logic simulation is the calculation of changes in an elements output signal value whenever a change occurs in the elements input signal value. The time required for the elements output to reflect the change at its input, that is the time required for a signal to propagate through one element, is called a gate time. This is the basic unit of time in LOGIC 4.

3.2.5.2 Coarse Time

A second level of time measurement in the LOGIC 4 program is COARSE TIME. This is a user defined multiple of GATE TIME and is usually determined by the number of gate times required to simulate the flow of a signal from a circuit input to the circuit output.

3.2.5.3 Gates/Coarse-Time

The GATES/COARSE-TIME statement defines the user determined coarse time to the program in terms of a ratio between gate time and coarse time. This time unit enables the program to determine circuit oscillation. This

is done by examining all elements fed by a set of user defined externals and simulating those elements. The program calculates the element outputs and if their outputs are not changed simulation is terminated for that coarse time. If some of these outputs did change, the program simulates the elements driven by the changed outputs and the process is repeated. If all signals have not settled within the defined GATES/COARSE-TIME, the program assumes circuit oscillation and the signals that are still changing are set to the don't know state.

3.2.5.4 External Time

The External time is that time at which the input externals are applied to the circuit design. This time is specified by the user as the start time on the external input statements.

3.2.5.5 Output Time

Each signal listed in the OUTPUTS category and/or desired signal classes specified by the SAMPLE option is sampled at the output time and reflected on the program timing diagram for each coarse time simulated. This is usually the last gate time in each coarse time; however, the user has the option of specifying the output time for his design by the SAMPLE statement in the CONTROL category.

3.2.5.6 Strobe Time

The Strobe time is the time at which the circuit measurable outputs are tested for fault analysis and/or the Quick Test (QT) option.

3.2.5.7 Normal Simulation

The normal simulation function is performed by applying a circuit input signal or a set of input signals (EXTERNALS) to the logic design then simulating and analyzing the operation of each element which has an input influenced by the applied externals (gate time one). Each element whose output is changed as a result of this input is applied to the next element in the logic structure. Those elements driven by this output are then simulated and their output analyzed (gate time two). This process continues until no more signals change value. If this process exceeds the specified number of GATES/COARSE-TIME oscillation is assumed and all subsequent signals are set to the DON'T KNOW state. If the effect ends before

the specified GATES/COARSE-TIME is reached, the simulation function terminates for that coarse time. The program then skips ahead to (SAT) the next gate time when an external is applied and starts a second simulation function for the next coarse time. This process continues for the number of strobe times specified by the user. Therefore, for good simulation it is important that the user analyze his design prior to simulation and specify a sufficient number of GATES/COARSE-TIME to prevent the program from assuming oscillation when signal changes will settle. This should include all possible feedback loop effects.

3.2.5.8 Fault Simulation

When Fault Simulation is requested by the user, the program first performs normal simulation for one coarse time and saves the measurable outputs at the specified strobe time. The program then resets all signals and memory values and returns to the first gate time in that coarse time and inserts the first specified fault condition. The program performs simulation at each gate time as described for normal simulation. The measurable outputs are compared with the response of the normal machine and the fault signature is formed. The program again resets all signals and memory, returns to the first gate time, inserts the next fault condition, and performs element simulation a second time. After all fault conditions have been simulated the faulted output states are compared to the normal outputs. If a difference is noted, a fault signature is formed. Only two types of differences will produce a fault signature in order that a fault is considered detected. These are a zero-one or one-zero. That is, if a signal was a don't know and is now a one or zero or the signal was a one or zero and is now a don't know, a fault signature is not formed and the fault is not considered detected.

At the completion of fault analysis for each COARSE-TIME the program compares the fault signatures formed. If there are less identical fault signatures than replaceable modules for which isolation is to be performed, the module is considered isolated and put on the output listing. If more identical fault signature than replaceable modules specified are formed, the module is not listed and the program will continue to attempt isolation for this module at the completion of the next COARSE-TIME.

3.2.5.9 Example Program Timing

Figure 3.2.5.9-1 shows the operations performed for one coarse-time of simulation. The COARSE-TIME is defined as being 14 gate times by the GATES/COARSE-TIME 14 statement. The Outputs are printed on the Timing Diagram at the completion of simulation for GATE time 6 as specified by the SAMPLE 0+6, 14 statement. Finally the circuit measurable outputs are saved for fault analysis at gate time 10 as specified by the TEST 0+10, 14 statement. The 14 indicates that the test function is to be performed every 14 gate times; thereby, occurring on the tenth gate time of each succeeding COARSE-TIME. The TEST statement enables the user to define the Strobe time to the program.

Figure 3.2.5.9-2 shows the operations performed during a fault simulation function. The VERIFY INPUTS ONE OUTPUTS BOTH statement states the type of faults to be inserted on each element of simulation. For example an inverter would be faulted three times; i.e., stuck to a logic one at the input, stuck to a logic one at the output, and then stuck to a logic zero at the output. The VERIFY ISOLATE TWO statements tells the program the maximum number of replaceable modules to which the program must isolate. The process shown for fault analysis in figure 3.2.5.9-2 is repeated for the specified number of Coarse-Times simulation is to be performed.

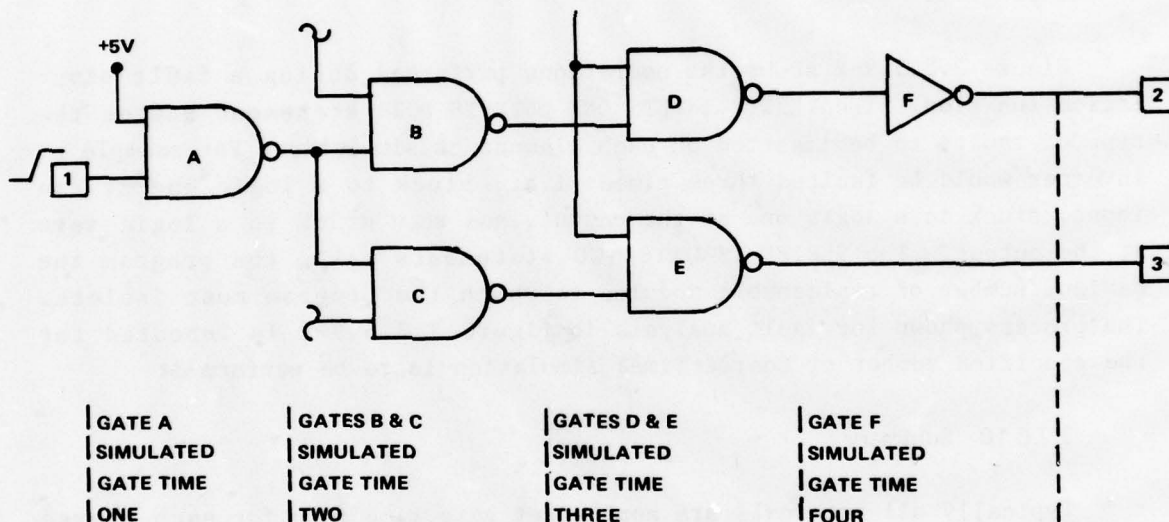
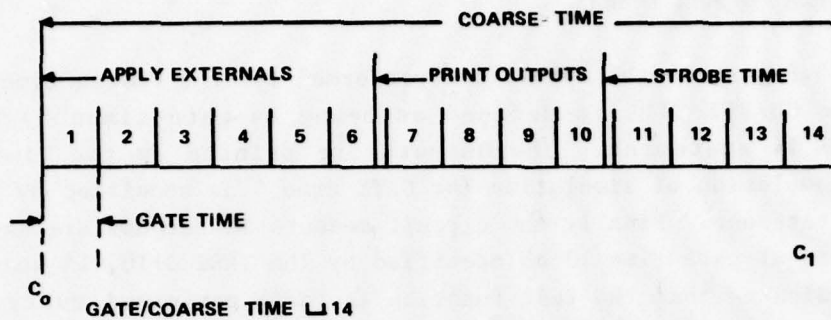
3.2.5.10 Summary

Typically all externals are applied at gate time zero for each coarse time and outputs are sampled for the timing diagram at the strobe time. Also the strobe time is the first gate time in each coarse time. Each of these, however, may be manipulated by the user.

3.2.6 Program Inputs

The program inputs describe the circuit logic design input control signals, and select the expected outputs and functions to be performed. There are nine input data categories in LOGIC 4. These are:

- a. CONTROL which controls the overall operation of the program
- b. RUN HEADER which prints header statements on each printout page.



SAMPLE $\square 0+6,14$
TEST $\square 0+10,14$

1. OUTPUTS ARE PRINTED AT THE COMPLETION OF SIMULATION FOR GATE TIME 6 AND REPEATED FOR EACH COARSE TIME AT GATE TIME 6.
2. OUTPUTS ARE SAVED FOR FAULT ANALYSIS AFTER COMPLETION OF SIMULATION FOR GATE TIME 10 AND REPEATED FOR EACH COARSE TIME AT GATE TIME 10.

Figure 3.2.5.9-1. Simulation Functions Performed For One Coarse Time

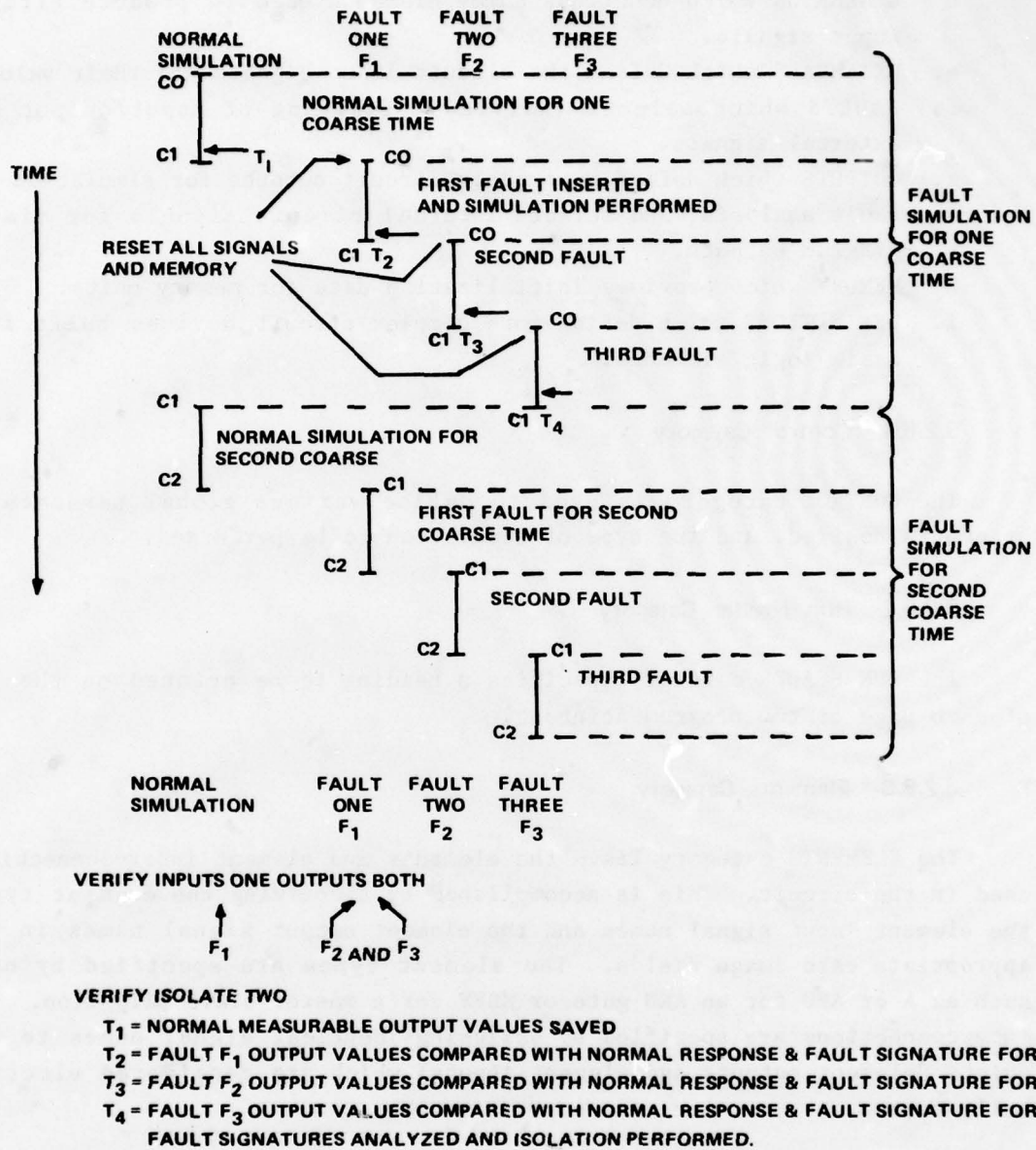


Figure 3.2.5.9-2. Fault Simulation Operation

- c. ELEMENTS which lists and defines all elements used in the circuit design.
- d. GENERATOR which describes dummy elements used to produce circuit input signals.
- e. EXTERNALS which define the circuit input signals and their values.
- f. FAULTS which selects individual faulting of input/output and external signals.
- g. OUTPUTS which define measurable circuit outputs for simulation and fault analysis, and selects internal circuit signals for timing diagram outputs.
- h. MEMORY which provides initialization data for memory units.
- i. DEFINITIONS which define more complex circuit devices built from basic logic elements.

3.2.6.1 Control Category

The CONTROL category is used to define various global parameters, printouts desired, and the type of simulation to be performed.

3.2.6.2 Run Header Category

The RUN HEADER category specifies a heading to be printed on the top of each page of the program printout.

3.2.6.3 Elements Category

The ELEMENTS category lists the elements and element inter-connections used in the circuit. This is accomplished by specifying the element type, the element input signal names and the element output signal names in the appropriate card image fields. The element types are specified by name such as A or AND for an AND gate or MSFF for a master slave flip flop. The interconnections are specified by assigning identical signal names to all points (element outputs and element inputs) which are considered electrically common.

3.2.6.4 Generator

The GENERATOR category is a dummy elements category used to describe elements which produce circuit input signals. These elements are excluded from fault analysis. Such elements may be used to provide interfacing logic with a circuit being simulated.

3.2.6.5 Externals Category

The EXTERNALS category is used to supply the circuit input signals to be simulated, and their value for each strobe time. This is accomplished by specifying the circuit input signal name and the discrete values or an input sequence formula for the value of the signal for each strobe time. The program then uses the first value assigned for each external as the first input to simulate. The second set of values is used next providing the user with complete control of the simulation process.

3.2.6.6 Faults Category

The FAULTS category enables the user to specify faulting for individual signals. This option may be used in addition to or to override the faults specified in the CONTROL category. The FAULTS category allows control of a signal in general or only as an input, output or external.

3.2.6.7 Outputs Category

The OUTPUTS category defines the circuit measurable outputs and lists the internal circuit signals that are to appear on the program output timing diagram. The order in which the signals are listed in the category is the order in which they appear on the timing diagram.

3.2.6.8 Memory Category

The MEMORY category enables the user to initialize the memory elements (ROMS and RAMS) used in the circuit design.

3.2.6.9 Definitions Category

The DEFINITIONS category is used to define complex circuit devices that are not already included in the program libraries. This new element is defined in terms of elements recognizable by LOGIC 4. In addition, a new name such as F314 is assigned to the new device and is then referred to as any other element.

SECTION 4 ELEMENTS/NET GENERATION PROGRAM—ASSIGN

The design and development of an electronic circuit involves a transition from original concepts, to diagrams on paper, and finally to actual construction of the circuit. The CAD ASSIGN program provides an interface between the circuit diagram and the realization of the circuit design. Figure 4.0-1 depicts the functional relation between the ASSIGN program and the remaining CAD programs.

4.1 INTRODUCTION

This section provides sufficient information to describe the purpose and general capabilities of ASSIGN and a general description of the functional areas, program operation and program output available for use. Users need have no programming experience to use the ASSIGN program although some background is helpful.

4.2 PROGRAM PURPOSE

Through the use of ASSIGN, an association is made between each logic element depicted in a schematic diagram and the physical element performing that logic function. In addition to logic element associations, all interconnection of element information illustrated on schematic diagrams is translated by ASSIGN into pin interconnections and passed on to other CAD programs which perform the physical placement and wiring of the various devices.

4.3 PROGRAM GENERAL DESCRIPTION

The ASSIGN program is normally run after the design has been proven to be of sound logical structure, and may be run in either the interactive or batch method. The program has two major modes of operation. These are the normal ASSIGN (String Generate) mode and the Element Generate mode. Inputs to the program for either mode of operation are of two classes: logic information and hardware information (refer to Figure 4.3-1).

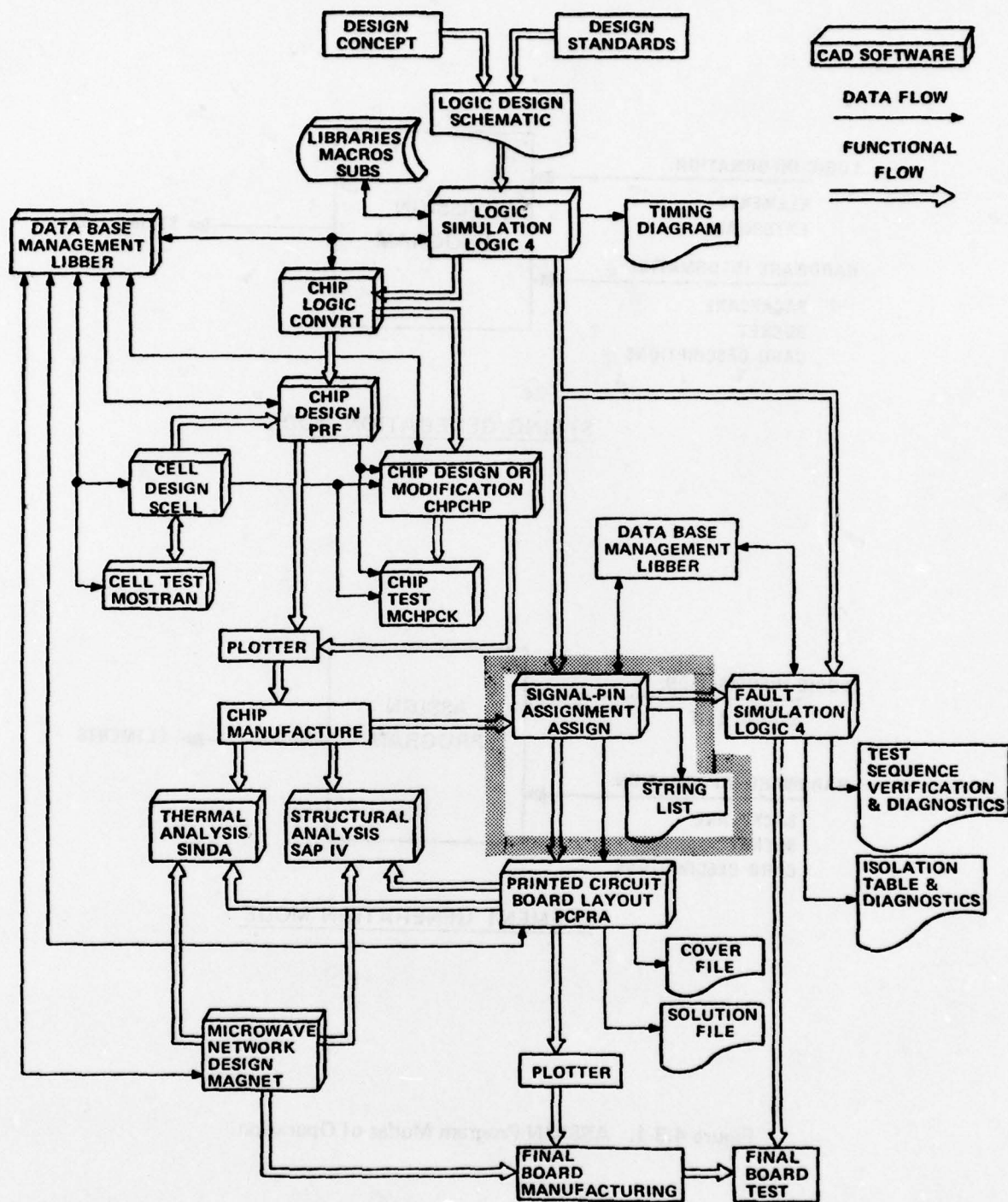
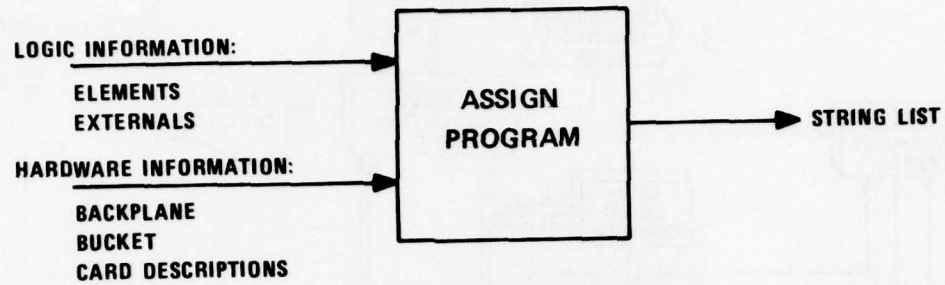
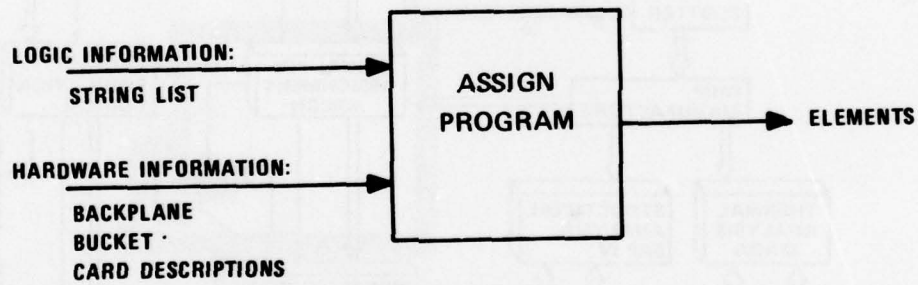


Figure 4.0-1. CAD Functional Block Diagram Emphasizing the ASSIGN Program



STRING GENERATION MODE



ELEMENT GENERATION MODE

Figure 4.3-1. ASSIGN Program Modes of Operation

4.3.1 String Generation

In the normal ASSIGN mode, logic information in the form of logic element interconnections, as supplied by the ELEMENTS data input, is matched with hardware information in the form of CARD DESCRIPTIONS data input. Use of ASSIGN in this mode includes assignment of a gate or flip-flop to a DIP, a DIP to a printed circuit board, or a printed circuit board to a card rack. Outputs from the String Generate mode include a String list and/or file containing information on all device pin interconnections according to signal name.

4.3.2 Element Generation

The Element Generate mode is used to adjust the ASSIGN data base to match any changed DIP and/or input/output (I/O) pin locations that are made after the initial assignment by the ASSIGN program. DIP and/or I/O pin locations may change because:

- a. Resultant outputs of a CAD program performed after the ASSIGN program; or
- b. Design changes occurring after the ASSIGN program is run.

In the Element Generate mode, logic information, as supplied by the STRING LIST of device pin interconnections, is matched with hardware information in the form of CARD DESCRIPTIONS data input. Outputs from the Element Generation mode include a description of the element interconnections with an additional tag that indicates which element is in which device.

4.4 PROGRAM FUNCTIONAL DESCRIPTION

The following is the sequence of the input data categories to the ASSIGN program. The order of these categories is fixed and is always as shown. Certain data categories may be omitted, but the sequence must remain the same. A summary of input data categories is presented in Table 4.4-1.

TABLE 4.4-1 ASSIGN Input Data Category Functions

CATEGORY	FUNCTION
CONTROL	Specifies program control options.
BACKPLANE	Describes the sockets, external connectors, and test points located on the backplane.
ELEMENTS	Provides the program with circuit inter-connection information.
GENERATOR	Special case of the EXTERNALS category, the importance of which is in the simulation aspect of design development.
EXTERNALS	Designates the input stimulus signals to the design.
EQUIVALENCE	Provides the capability to change hardware devices with ease.
BUCKET LAYOUT	Designates the card to be placed in a socket on the backplane.
CARD DESCRIPTIONS	Provide data on each card device used in the design.
STRING LIST	Provides for interconnection of pins on the backplane by signal name.
MANUAL DEFINITIONS	Provides the ability to manually place signal names on pins in the backplane.
PARTS LIST	Provides data for a complete, documented Computer Generated Parts List of the design.

PROGRAM INITIALIZATION
CONTROL CATEGORY
BACKPLANE DESCRIPTION

ELEMENTS CATEGORY
GENERATOR SPECIFICATION
EXTERNALS CATEGORY
EQUIVALENCE CATEGORY
BUCKET LAYOUT
CARD DESCRIPTIONS
STRING LIST
MANUAL DEFINITIONS
PARTS LIST

4.4.1 Program Initialization

To initialize the ASSIGN program the user must first set up the required operating system conditions (refer to DEC SYSTEM 10 DEC-10-OSDMA-A-D). After the system has been set up, the ASSIGN program is accessed by the program select RUN command. The system will then respond with a request for file assignment at which time the user must input the appropriate commands to reference all files to be accessed or created by the ASSIGN program by use of the UNIT, FILE command. After file assignments are made, the program is ready to accept the CONTROL input data category.

4.4.2 Control Category

This category specifies the desired program options. These include information on program control, I/O units, and general program parameters.

4.4.3 Backplane Description

The BACKPLANE input category contains the description of the assembly in which the cards, described by the CARD DESCRIPTION data category, are inserted. This may be a multichip hybrid package (MHP), in which chips are placed; a planar array (printed circuit board), in which DIP's are placed; or a card rack, in which printed circuit boards are placed. This category is limited to the description of sockets, test point pins, and I/O connector pins, which are to be located on the Backplane. Requirements for this type of hardware should be understood before coding the BACKPLANE DESCRIPTION input category.

NOTE

Note that the ASSIGN program is not a hardware placement program and actual placement of hardware on the backplane is done in other CAD programs.

4.4.4 Elements Category

The Elements category provides the description of the logic schematic diagram in a form understood by the program. The interconnections of each logic element used in the circuit are provided by specifying signal names to be associated with inputs and/or outputs of each element. The same signal name is used for all inputs and outputs which are to be interconnected. The ELEMENTS category contains a listing of each logic element by type, and all associated input and output signal names. Information required to code the ELEMENTS input data category can be obtained directly from a logic schematic diagram if properly marked with signal names.

4.4.5 Generator Specification

The GENERATOR specification is a special case of the EXTERNALS data category and describes all logic used to generate input signals other than EXTERNALS. The GENERATOR specification is used by the simulation portion of the CAD system. If included in ASSIGN, it will be considered as an EXTERNALS category entry. For a detailed description of the GENERATOR category refer to the CAD Users Manual for the LOGIC 4 program.

4.4.6 Externals Category

The EXTERNALS data category defines those signal names which are external inputs to the system being processed and are the input stimulus to the design. Signals of this nature should be apparent on the logic schematic diagram.

4.4.7 Equivalence Category

Hardware changes are necessary many times during the design of a circuit. The EQUIVALENCE category provides the ability to change the hardware in a circuit with relative ease. The EQUIVALENCE data category contains equivalences, or synonyms, of device names. When this category is used, the old names in the BUCKET LAYOUT category are replaced by the new names.

4.4.8 Bucket Layout

The BUCKET LAYOUT category specifies the packages (chips, DIPs, or PC boards) used and the sockets in which they are inserted. Each package listed in the BUCKET LAYOUT must be described by an entry in the CARD DESCRIPTIONS category, and each socket listed must match a socket defined in the BACKPLANE DESCRIPTION category.

The order of assignment of elements to physical packages is determined by the order of the packages in the BUCKET LAYOUT. To provide the most efficient assignment by the program, the suggested, but not required, ordering of packages in the BUCKET LAYOUT is to place non-pin-sensitive devices ahead of pin-sensitive devices. In addition, ordering within each of these device types should be by order of increasing number of inputs.

4.4.9 Card Descriptions

CARD DESCRIPTIONS data category contains descriptions of the physical packages (chips, DIPs or PC cards) processed. The header, required for each package described, contains information on the generic device name, number of pins, power dissipation, failure rate data, weight, cost, and part identification number. The rest of the CARD DESCRIPTION lists the elements contained in the device and their pin numbers along with the load or drive capability of each. These elements and pin numbers are matched with the elements and signal names of the ELEMENTS category to generate the device pin to signal name association, on which the ASSIGN program functions. It is important that the element type and input/output fields used in the ELEMENTS category match the element type and input/output fields used in the CARD DESCRIPTIONS if proper association is to be achieved. Device power and ground pins are also specified in the CARD DESCRIPTIONS category.

The information required to code the CARD DESCRIPTIONS category can be obtained from manufacturer specifications on the device to be used. Data books, catalogs, and reference manuals are all issued to supply such technical information.

4.4.10 String List

The STRING LIST data category defines connections between pins on the backplane. This is done by specifying a signal name and all socket, test point, and external connector pins associated with that signal name.

4.4.11 Manual Definitions

The MANUAL DEFINITIONS category is used to define a signal name on a particular socket and pin in the bucket. Thus, an element can be assigned to a particular package by defining the output signal name for that element to the output pin of a package. Options to assign special pins, such as test point and external connector pins, are also possible using the MANUAL DEFINITIONS category.

4.4.12 Parts List

The PARTS LIST category is used to describe items, such as capacitors, resistors, or wire, which are needed for a complete parts list but are not included in either the BUCKET LAYOUT or the CARD DESCRIPTION data categories. Header information for the Computer Written Parts List (CPL) is also entered in the PARTS LIST category.

4.5 PROGRAM OPERATION

The ASSIGN program produces a stored 'list' of the sockets with one data record reserved for each pin on each socket. The sockets in the 'list' are stored in the same order in which they appear in the BUCKET LAYOUT data category. When a signal name is placed on a pin, either by the MANUAL DEFINITION category or as the result of assignment by the program, that signal name reference number is placed in the location corresponding to that pin. The ASSIGN program makes a number of passes through the ELEMENTS category to complete the pin assignment. These passes may be divided into four phases of program operation executed in the order presented and correlated with the input category involved.

4.5.1 Manual Definition Phase

All manually defined names are placed on the specified socket pin. The program examines each socket in the Bucket Layout in the order in which they appear in the input data. If an output pin has a signal name placed on it, the program finds the first element of the corresponding type that has that name as an output and tries to assign it. It goes through all the

positive pins in ascending numerical order and then the negative pins in descending numerical order during this phase. Thus, if in assigning an element, a name is placed on an output pin of another element which is processed later, then a feedback process takes place and the other element is then assigned.

4.5.2 Tag Definition Phase

This phase does not occur unless a TAG-DEFINITIONS card is placed in the CONTROL data category. During this phase, the TAGS field of all previously unassigned elements is compared with the socket names in the bucket and, if a match occurs, an attempt is made to assign the element to the first unused circuit in that socket.

4.5.3 All Elements Phase

All devices are processed in the order in which they appear in the input data. For pin sensitive devices (flip-flops as opposed to gates) the cards are searched in the order specified in the BUCKET LAYOUT and the device is assigned to the first unused element of the same type. Gate devices are not assigned during this pass unless the assignment results in all input and output pins of the element being used.

4.5.4 Gate Second Try Phase

The second pass processes all gate devices not assigned in the first pass. This is done in the order of appearance in the input data. The cards are searched in the order specified in the BUCKET LAYOUT and the device is assigned to the first unused element of the same type found. Note that for non-pin-sensitive (gate) devices, inputs are interchanged in an attempt to fit a device into a particular element. This is of use where two or more elements in the same package have a common pin, resulting in a signal name being placed on an input pin of an otherwise unused device.

4.6 PROGRAM OUTPUT DESCRIPTION

Following is a description of the possible printed output generated by the ASSIGN program.

4.6.1 Input Data

ASSIGN is capable of generating an output listing of the input data in both unaltered format and expanded format. The unaltered format is an exact duplication of all input data categories as supplied to the program.

Expanded format is a listing of the input data categories incorporating all Repeat, Synonym, and Equivalence options, where used. (Refer to the CAD ASSIGN Users Manual for detailed descriptions of these options.) Therefore, a listing would be generated for each of the following categories.

- EXPANDED CONTROL CARDS
- EXPANDED BACKPLANE DESCRIPTION
- EXPANDED ELEMENTS
- EXTERNAL VALUES
- BUCKET LAYOUT
- MANUAL DEFINITIONS

4.6.2 Card Descriptions

The CARD DESCRIPTION printout lists the device type, number of pins, and each element on the package by type, (giving input pin number, load, output pin number, and drive capability) and power and ground pin numbers. Only those card devices listed in the BUCKET LAYOUT will be listed in the CARD DESCRIPTIONS.

4.6.3 List of Non-Fitting Elements

Any element in the ELEMENTS category that does not match an element provided in the CARD DESCRIPTIONS category is listed under the heading LIST OF NON-FITTING ELEMENTS. Any entry in this list should be considered as a diagnostic, warning the user of the failure of the program to associate a hardware device with a logic device that required such a hardware assignment.

4.6.4 Bucket Map

The BUCKET MAP provides a listing, by socket connector and device type, of all pins on that connector and device type, of all pins on that connector, and the signal name which was assigned to that pin.

4.6.5 Spare Elements

Unassigned elements within partially used card devices are listed in the generated output list of SPARE ELEMENTS. The listing shows socket, card type, element type, whether input and/or output pins are not assigned, and output pin numbers.

4.6.6 Signal Loading

Each device pin that is assigned a signal name provides a signal drive capability, or load imposed on the signal as specified by the CARD DESCRIPTION data. The total loads are subtracted from the drive and the results are listed in the SIGNAL LOADING output. System input or output signals are indicated as such. As no drive capability is listed for an input signal, a drive of zero is assumed for tabulation purposes.

4.6.7 Internal String List

The INTERNAL STRING LIST provides all information necessary to wire the assembly into which the backplane is to be inserted. Information included is the signal name, type, and external connector and pin number.

4.6.8 Test Point List

The TEST POINT LIST describes the backplane test points by listing the signal name and test point connector name and associated pin number of each.

4.6.9 Unused I/O Connector Pins or Unused Test Points

Any I/O connector pins or test points that were defined in the BACKPLANE DESCRIPTION but not assigned a signal name are listed by connector name and pin number.

4.6.10 Allocated Elements

The ALLOCATED ELEMENTS listing relists the EXPANDED ELEMENTS listing with the addition of the following information: The socket to which a particular element was assigned is listed below each element type and also

particular element was assigned is listed below each element type and also in the tags field, replacing any previous tags. Also, the pin to which a signal name was assigned is listed below the signal name.

4.6.11 Parts Summary

The PARTS SUMMARY provides technical information for each device used in the design. Listed is device type, quantity used, power dissipation, failure rate, weight, cost, and part number along with totals for the complete design.

4.6.12 Computer Written Parts List

The COMPUTER WRITTEN PARTS LIST (CPL) lists each part required for the design circuit and any ordering information as provided in the CARD DESCRIPTION or PARTS LIST data categories.

SECTION 5

INTRODUCTION TO PCPRA: PRINTED CIRCUIT PLACEMENT ROUTING AND ARTWORK PROGRAM

In designing printed circuit boards, there is generally a great need to minimize the use of layers and the length of printed wiring paths. The Printed Circuit Placement Routing and Artwork Program (PCPRA) is a printed circuit wiring routing optimization program for production of two-sided or multilayer printed circuit boards (PCB's). As such, it is a major tool in the design of PCB's. It can be applied to problems that can be modeled as components with wiring on a PCB.

PCPRA consists of four parts that are normally used in order, yet can be used independently. The placement part (PLACE) selects a placement of the packages to be used on a PCB so as to minimize total printed wire length. The organizer part (WORGZ) accepts the output data from PLACE, organizes it, and formats it for input to the router part (PROUTE). The router performs printed circuit wire routing for two-sided or multilayer boards. The artwork (or printer) part (MLPLOT) presents a printer plot of the routing. Figure 5.0-1 depicts the functional relation between the PCPRA Programs and the remaining CAD Programs.

5.1 GENERAL DESCRIPTION OF PCPRA

The four programs PCPRA uses to accomplish a goal of minimum use of layers and shortest printed wiring paths possible are:

- a. PLACE - Inputs the BUCKET LAYOUT Category, the STRING LIST from the ASSIGN program of CAD, and other information to automatically place component sockets on the PCB. A complete GEOMETRY DATA file is created.
- b. WORGZ - Inputs and organizes the GEOMETRY DATA and additionally inputs and organizes the NET LIST of signals and pins that should be interconnected. The NET LIST may be generated by a designer or by the ASSIGN program of the CAD system, and the GEOMETRY DATA may be generated by a designer or by the PLACE program of PCPRA. The output of WORGZ is the COVER FILE.

- c. PROUTE - Inputs the COVER FILE from WORZ and performs the printed wire routing to connect the pins in the NET LIST. The output is the SOLUTION FILE.
- d. MLPLOT - Inputs the COVER FILE and/or the SOLUTION FILE and prints a line printer plot representing the PCB with the COVER FILE information and, depending on the user's choice, a printer plot of the PCB with the SOLUTION FILE information.

Figure 5.1-1 shows the logical flow of the inputs and outputs for the PCPRA programs.

PCPRA uses the GEOMETRY DATA and NET LIST to produce the wiring paths for the various layers of a multilayered board. The program initially defines a PCB as having only two layers. The top is layer 1, while the bottom is layer 2. Electrical connections between the two layers are made possible by plated thru holes called vias or feed-thrus. The wire routing program uses a PCPRA modified version of the NET LIST, called the EXTENDED NET LIST (ENL), and information in the GEOMETRY DATA to make component interconnections within the physical dimensions of the initial two-layer PCB. If the ENL is not exhausted by the routing process, PCPRA creates another two-layer PCB. This new board has images of all the pins and feed-thrus from the previous board. The router uses these if possible to complete the residue ENL connections on the top layer of the new board which will be layer three of the multilayer board. If it is not possible, either the feed-thrus created on the first board are used to connect to layer 4 (the bottom of the new board) or new ones are established. This process continues until either all the connections are made or the maximum number of layers allowed is reached. Any residue or unconnected signals are printed in a failure listing. By enlarging the spacing (layering) available for routing on the PCB, the failure list can be eliminated. A line printer plot is produced for each layer of the PCB with or without the wiring depending on the options used.

5.1.1 Printed Circuit Board Design Process

The physical make-up of the PCB is like a sandwich in which each layer is separated from adjacent layers by an insulation layer. Figure 5.1.1-1 depicts an exploded view of a typical four layer multilayer board. Layers one and two are called outside layers and any other layers are called inside layers. Component sockets are located only on layer one, but their

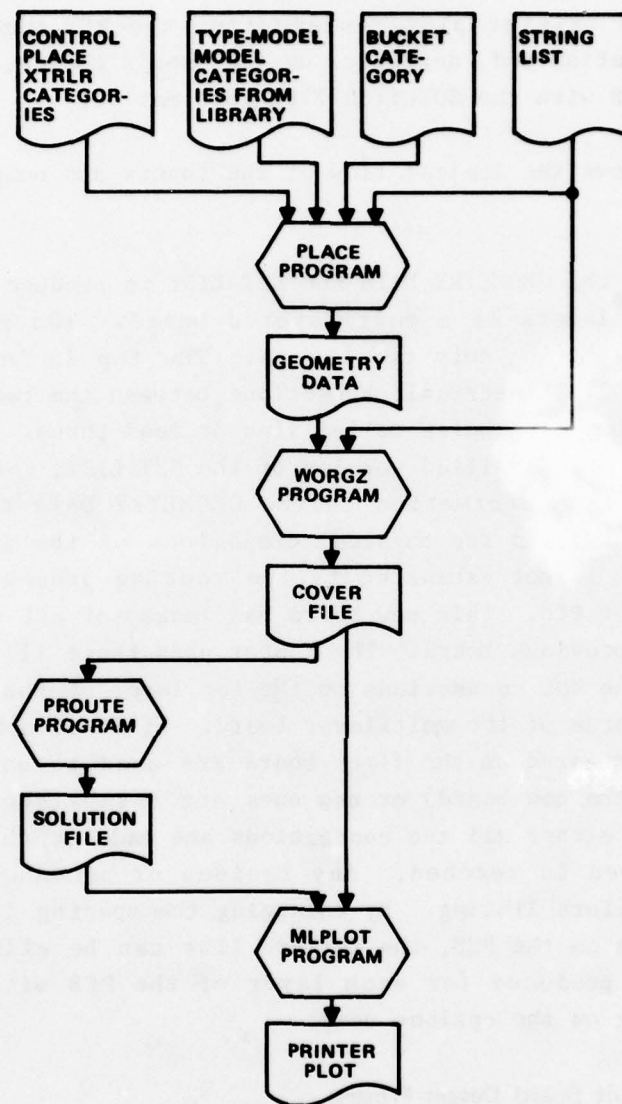


Figure 5.1-1. PCPRA Program Flow

pins, as well as test point, connector, and user-defined pins, pass through all layers. Feed-thrus are also passed through all layers.

5.1.2 PCPRA Requirements

Specific information is required in order to use the Printed Circuit Placement Routing and Artwork (PCPRA) programs. The typical successful designer will have the following:

- a. A reasonably accurate, dimensioned drawing of the bare P.C. board showing tooling holes, board puller areas if board pullers are used, physical connector and test point locations, electrical connector and test point locations.
- b. Indication of the outline of the area where automatically generated wiring may be placed defined by straight horizontal and vertical segments.
- c. Indication of the outline of the rectangular area where components (sockets) may be placed. This area must be inside the area where automatically generated wiring may be placed.
- d. The NET LIST (usually input from the output of the ASSIGN program of CAD) that specifies the desired interconnection. The NET LIST is made up of a series of nets. Each net must have a unique signal name. All socket pins having the same signal name will be electrically connected. A socket pin does not have a name if the socket pin is not connected. A socket pin cannot have more than one name. A typical net description is shown below.

CLOCK	J01007	U01009	U02005
-------	--------	--------	--------

This typical net indicates that the signal with the name CLOCK is to be electrically connected to connector J01 pin 7 and dual-in-line pack U01 pin 9 and to dual-in-line pack U02 pin 5.

- e. The geometric connector and socket information. For example, if U01 is a 54H108 DIP, the manufacturer's catalog provides information that this is a 14 pin dual-in-line package with a JEDEC (TO-116) socket geometry (i.e., two rows of 7 pins with row spacing of 0.3 inch and pin spacing of 0.1 inch).
- f. Choice of socket placement at specific locations on the P.C. board or use of the automatic placement feature of PCPRA. The automatic placement will tend to minimize total track length and thus maximize the probability of 100% automatic routing. Manual placement may be used if there are mechanical or thermal

- (A) LAYER ONE CONTAINS ALL COMPONENTS
- (B) LAYERS ARE CONNECTED BY PINS OR FEED THRUS
- (C) LAYERS ONE & TWO ARE CALLED OUTSIDE LAYERS, THREE & FOUR ARE INNER LAYERS
- (D) LAYERS ARE SEPARATED BY INSULATION
- (E) COMPONENTS ARE CONNECTED BY PRINTED WIRE TRACKS USING LAYER PAIRS

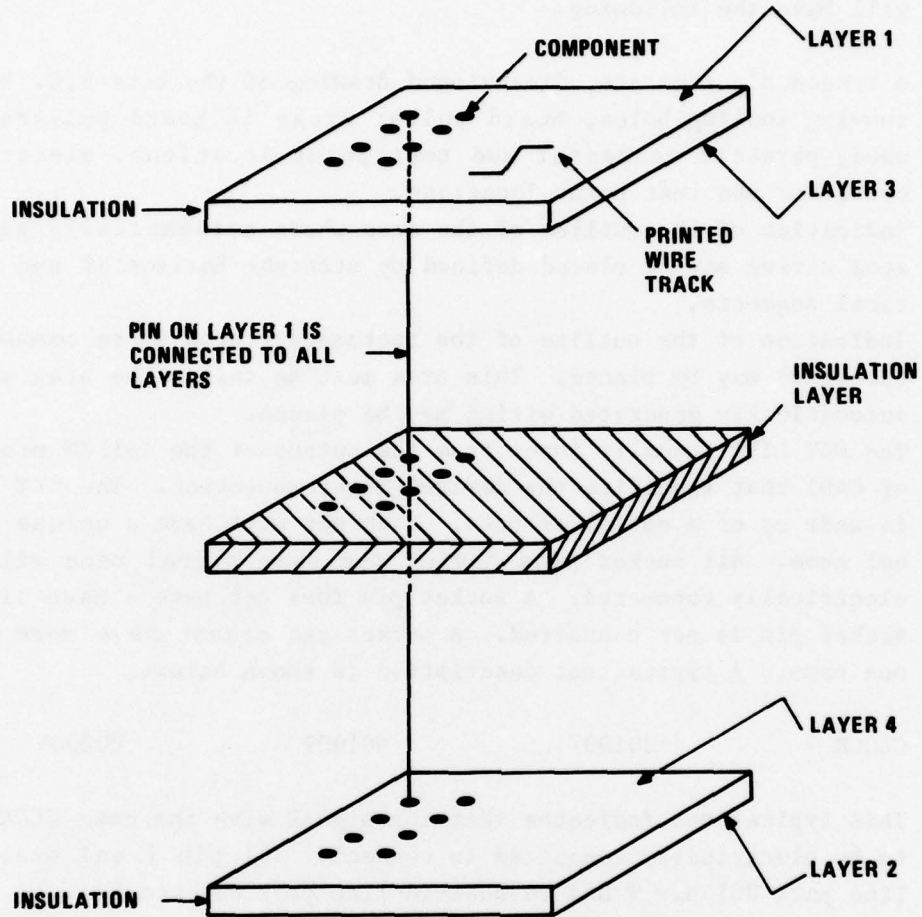


Figure 5.1.1-1. Exploded View of a Typical Multilayer Board

constraints. Manual placement may also be necessary if there are critical signals or manufacturing constraints.

- g. PCPRA operates on a grid system. In principle, the grid does not have dimensions until the time when the final artwork is plotted. However, knowledge of the physical relationship of the grid to the real world is required. Socket pins exist at the intersection of a horizontal and vertical grid line. Tracks are centered about horizontal and vertical grid lines. Routing is done horizontally and vertically along grid lines. An example of choosing a grid is as follows: Assume DIP's are to be placed on a P.C. board and one track is allowed between a pair of DIP pins. The DIP pins must be on the grid and the track must be on the grid. DIP pins are spaced 0.1 inch apart and a routing lane for a track must fall between the pins. Thus the grid would be chosen to have $1/2 \times 0.1 = 0.05$ inch spacing.

5.1.3 PCPRA Utilization Guidelines

Some useful guidelines to successful use of PCPRA follow:

- a. The most difficult task for PCPRA is wiring ground and power nets. When manufacturing permits, a ground plane and a power plane should be used. When using ground planes and power planes the power and ground should not be in the NET LIST. If power and ground are already in the NET LIST then the IGNOR statement of WORZ is used to ignore them.
- b. If power and ground tracks must be on layers along with signal tracks then manual aid to PCPRA is recommended. Figure 5.1.3-1 shows the typical use of multiple connector pins for power and ground at opposite ends of the connector. A power track is run vertically on one side of the board and a ground track is run vertically on the other side. Horizontal runs are then made with short stubs to the socket power and ground pins. This is desirable because the automatic routine will be in a predominately horizontal, x-direction, on layer one.
- c. The next most difficult task for PCPRA is wiring signals to the connector(s) when a majority of the connector pins are used. The designer has the choice of assigning signals to connector pins or allowing PCPRA to make the assignment. In either case, both placement and routing give priority to routing signals to the connector(s). If the connector plug is manufactured separately

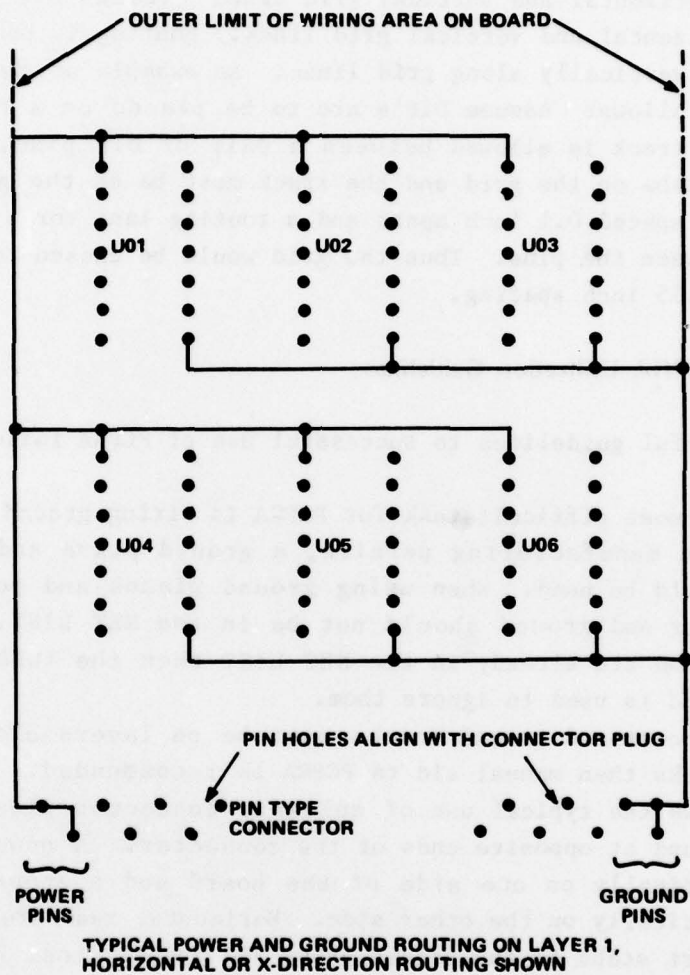


Figure 5.1.3-1. Typical Power/Ground Routing

and mounted on the board with pins, and the connector holes can be defined by using a connector input statement, then no further effort is required. This is because PCPRA is pin/hole oriented and a connector that uses pins to plug into a hole (much as a component into a socket) adapts to this orientation. If the connector is made by plating on both sides of the PCB, manual assistance to PCPRA is recommended. Connectors that require strips on the top and bottom of a two layer board to make a connection need some manual intervention to create the strips. They are typically generated on an interactive graphics machine.

The PLACE program is an optional automatic generator of the GEOMETRY DATA. The user may choose not to use it and instead create appropriate GEOMETRY DATA for use by WORGZ. If PLACE is not chosen, then the user may want to run WORGZ and MLPLOT only. WORGZ will find most of the errors in the input data and MLPLOT will plot a picture of the board and the hardware on it without the printed wiring. Otherwise the order for running PCPRA is:

- a. PLACE
- b. WORGZ
- c. PROUTE
- d. MLPLOT

5.2 PCPRA FUNCTIONAL DESIGN AND CAPABILITIES

Following is a general description of the four major functions of PCPRA and their capabilities. Each function is treated individually along with a discussion of each of the function capabilities.

5.2.1 PLACE Program

The placement program is the interface between the ASSIGN program and the organizer program of PCPRA. The placement program connects the logical interconnection of physical component sockets (which the ASSIGN program of CAD has generated) into specific geometric placement of component sockets on the PCB. PLACE uses component socket pin spacing (as defined by the library) and board size (as input by the user) to determine an initial placement of component sockets on the PCB. The NET LIST (STRING LIST of the ASSIGN program of CAD) is used to successively improve the placement through trial interchanges of component sockets. An improved placement

means there is a shorter total PCB printed wire length. Component socket rotations in ninety degree increments are tried as well as component socket interchanges.

The input to the PLACE program is:

- a. The size of the board given as a rectangle.
- b. The routing outline and part outline within the board rectangle to account for board pullers and mounting structures.
- c. The NET LIST.
- d. The BUCKET LAYOUT category containing the NET LIST component name to generic component name (for example U31 is a SN5404, U26 is a SN5410).
- e. The library containing the spacing requirements and geometric pin location information for generic components (for example a SN5404 and a SN5410 are 14 pin DIPS).

Input to the PLACE program is separated into data categories. Each category contains a specific type of data.

Data categories are useful for several reasons. First, data categories help the user organize the data and reduce the possibility that important data is omitted. Second, data categories allow modular software development and maintenance by having one subroutine to process one specific data category. And finally, data categories may be input directly by the user or any specific data categories may be automatically retrieved from library files. The following categories are used by PLACE:

- a. The CONTROL category is used to specify overall control of the computer run. Print options may be specified. Any data category may be specified to be input from a unit other than the main input unit (Unit 5).
- b. The STRING category is used to obtain the interconnection information. This data category may come from the ASSIGN program or be user generated.
- c. The BUCKET category relates the socket designations in the string list to physical components, i.e., U03 to 5404.
- d. The TYPE-MODEL assignment category relates the physical component to the package geometry, i.e., SN5404 to a 14 pin DIP. This data category is typically input from a library file (PLAMOD.LIB).

- e. The MODEL category has the component size and pin locations for many packages. For example: 14, 16, and 24 pin DIPS are described, as are flat packs, resistors and capacitors. This data category is typically input from a library file (PLAMOD.LIB).
- f. The PLACE category must be present in order to specify the PCB size and the area where parts may be placed. Optional input in the PLACE category is the definition of the available routing area. As a convenience, all of the types of data cards that WORZ accepts may be input free-format in the PLACE category. These statements are just passed along to WORZ in proper format.
- g. The XTRLR category may be used to input the pin locations for test points and input/output connectors.

The output of PLACE is the GEOMETRY DATA formatted for use by WORZ.

5.2.2 WORZ Program

The WESTINGHOUSE organizer (WORZ) program is the interface between the placement and routing phases of the board design system. The WORZ program processes and checks the input data, assigns input/output connector and test point pins to signals not assigned in the NET LIST, and organizes the NET LIST into the EXTENDED NET LIST. By using computer calculated figures of merit that determine an order for routing signals, WORZ attempts to minimize layering.

The input to WORZ is the GEOMETRY DATA and NET LIST along with some control statements. The NET LIST and GEOMETRY DATA should be created on a file by the user before input to WORZ. The PLACE program is one means of generating the GEOMETRY DATA file.

The WORZ program requires the input of the GEOMETRY DATA and the NET LIST. The UNITS statement of WORZ contains the unit numbers where the GEOMETRY DATA and NET LIST are located, and they must agree with the unit numbers of the files assigned by the user during the file assignment portion of WORZ. Any valid DEC system unit number may be used in assigning these files. If no output file is assigned, WORZ writes a default file named FOR19.DAT which is the COVER FILE on unit 19.

The following statements are used to control and input data to WORZ:

UNITS - allows input of the GEOMETRY DATA and NET LIST files from the system peripherals.

- BOARD - contains information on the maximum X (length) and Y (width) values for the PCB to be modeled. The Board statement also specifies print options for the GEOMETRY DATA and NET LIST inputs and the printing of the unused connectors on the board.
- ARRAY - Used to define component geometry (the pin placements) on the board. Each statement defines a row or column of pins and the associated pin numbers. Coordinates are given in terms of a local coordinate system using a user-selected local reference point. More than one statement may be used to define a particular component.
- PLACE - uses the pin configuration defined by a set of ARRAY statements to place a component on the board at a user-designated location. PLACE uses the local origin (as defined in ARRAY statements) to place the array of pins at an X,Y board location.
- DISC - used to define and locate discrete components of up to three pins. The discrete statement is used when there is a small number of components which do not warrant the use of an ARRAY type definition.
- XTRLR - used to define pin locations for external I/O connectors and test points. These arrays are treated as components with a special feature. If a connector is called out in the NET LIST and a specific pin number is not given, the router selects the nearest available pin. These arrays of pins extend through each layer of the board in the same manner as pins.
- (SMIP - used to define router restrictions on the board. Such items as tooling holes and board outlines are defined with (SMIP statements. These restrictions are at the board level. Another statement (LOCAL) is used to define restrictions at the component level.
- LOCAL - used to define router restrictions at the component level, such as no feed-thrus allowed under a socket or no lines allowed under a socket. The LOCAL statements refer back to ARRAY statement names so that whenever a PLACE statement uses an ARRAY statement and has restrictions put on it by the LOCAL statement, then the LOCAL restrictions will appear in the area called for by the PLACE statement.

- IGNOR - used to delete unwanted signals from the NET LIST before the routing process.
- END - used to signal the termination of the GEOMETRY DATA input.
- NET LIST - This is not a statement in itself but a name for a listing of signal names and associated component sockets, connectors, and test points with their respective pin numbers assigned to those signal names. The NET LIST may be input manually or by file.
- END - used to signal the termination of the NET LIST.

WORZG generates the COVER FILE for the routing phase containing the environment for the multilayer board and the EXTENDED NET LIST which lists the interconnection requirements.

The WORZG output file is developed by the following sequence of operations:

- a. Input statements are read and tables are established defining all pin locations on the board.
- b. The placement information is read and a table is constructed defining sockets, connector and test point pin positions, and unused pins.
- c. The NET LIST file is read, board coordinates are assigned to the pins, and the number of pins and the NET HALF PERIMETER are computed and recorded.
- d. The NET LIST is then sorted for subsequent processing.
- e. Each signal net from the sorted NET LIST file is processed in the following manner:
 1. Assign connector pin(s) where required, based on the connector pin proximity to a net pin.
 2. Analyze the net and issue printer diagnostics for single pin nets and unidentifiable signal/pin combinations.
- f. The GEOMETRY DATA is converted to (SMIP statements. The (SMIP statements reflect all pins and any router restrictions.
- g. The pin list is sorted again to produce the EXTENDED NET LIST (ENL). The (SMIP statements and ENL are written to the WORZG output file and WORZG terminates.

5.2.3 PROUTE Program

The printed circuit router (PROUTE) program generates socket, connector, and test point interconnections from the COVER FILE information. Signals are considered for routing in the order determined by WORZ. The router works with layer pairs of a board using the second layer only when necessary. The program routes as many signals as it can on these two layers and then creates another layer pair to make the remaining connections. This process continues until 1) all the signals are routed, 2) the user-picked layer limit is reached, or 3) the program layer limit of 16 layers is reached. If the router fails to complete the interconnections in the number of layers assigned by the user, a detailed FAIL LIST is generated. For example, if the user restricts the router to two layers and at least three are required to route all the signals, a FAIL LIST would appear listing the signals and unmade connections.

PROUTE is capable of two directions of routing:

- a. X/Y - in which routing for layer one is primarily in the X board direction (except for one grid changes in the Y board direction); and for layer two routing is primarily in the Y board direction (except for one grid changes in the X board direction). All layers other than these use routing in both the X and Y directions.
- b. FOUR DIRECTIONAL - in which routing is in both X and Y board directions.

The user has the option of choosing the method used. The designer should oversee the operation to assure good design techniques. PROUTE is also capable of HIGH DENSITY or STANDARD DENSITY routing. HIGH DENSITY with X/Y directional routing provides the most efficient use of the board space.

The success of automatic computer layout is influenced by the restrictions used and the routing area available. When restrictions are imposed, manual intervention may be required to assist PROUTE. As restrictions become severe, intervention may approach the amount of effort required for a totally manual design. The main factors involved are the number of layers involved; number of component sockets, connectors, test points used; the size of the printed circuit board; and the permission to place feed-thrus.

There are three options available for feed thrus:

- a. PROUTE is given freedom to place feed-thrus where they are needed.
- b. PROUTE is limited to user-defined locations.
- c. No feed-thrus are allowed. Layer to layer interconnections are made by the component socket pins, connectors, and test points (since they extend to all layers).

Best results are obtained by allowing PROUTE to place feed-thrus where needed.

The input to PROUTE is the output from WORGZ plus some control information such as the maximum number of layers for the PCB.

The output is the SOLUTION FILE containing the X-Y board coordinates of all the interconnecting segments generated. The form is signal name to component socket of the origin of the route, the component socket of the target of the route, and a post processor field. A statistical summary is printed containing the number of signal nets input, the number of connections to be made, the number of connections made, and the number of connections not made (fails).

5.2.4 MLPLOT Program

The multiple layer plotter (MLPLOT) assembles a line printer plot of the board layout showing the routing of all interconnections and any user applied routing restrictions. Any layer or all layers are printed depending on the PARAMETERS input. A group of statistics showing the number of cells used, total number of cells, and the percentage of the PCB is printed after each total plot. Also shown are the number of feed-thrus, pins, line segments drawn, and signal nets.

The input to MLPLOT is the output from WORGZ and PROUTE plus some control parameters. The plot parameters control the type output, layers to be printed, location of X,Y axes, board origin location, size of board, and whether restrictions are to be printed.

The following types of printouts may be obtained from the MLPLOT program when used in conjunction with the WORGZ program or with the WORGZ and PROUTE programs:

- a. WORGZ and MLPLOT only produce a plot of the board components sockets, connectors, and test points without interconnections;
- b. WORGZ, PROUTE, and MLPLOT produce:
 1. Plot of board component sockets, connectors, and test points connected;
 2. Plot of board component sockets, connectors, and test points unconnected.

SECTION 6

INTRODUCTION TO SCELL/MOSTRAN, CELL LAYOUT AND TRANSIENT ANALYSIS PROGRAMS

This chapter describes the purpose and capabilities of the Computer Aided Design (CAD) SCELL/MOSTRAN programs used in the design of Integrated Cell Layout and Circuit Transient Analysis. Figure 6.0-1 depicts the functional relation between the SCELL/MOSTRAN programs and the remaining CAD programs.

6.1 GENERAL DESCRIPTION

The purpose of the SCELL/MOSTRAN programs is to provide a tool to assist in the geometry design and circuit simulation analysis of integrated circuit metal-oxide-silicon (MOS) cells. Designed MOS cells are cataloged and stored along with all necessary cell geometry and circuit parameters onto a master library to be later used by the integrated circuit chip layout programs

6.1.1 SCELL Program

The SCELL program is a general purpose cell layout program for designing metal-oxide-silicon (MOS) cells used in integrated circuits. Based on a library of basic device configurations, defined for each new technology, it provides a rapid and accurate means of defining cell geometries, that is, the placement and interconnection of devices that form a cell.

Since the SCELL program maintains a library of commonly used, completely described devices, the user is required only to input the desired spacing, placement constraints, device parameters, and valid interconnections. The program builds a picture of the cell in memory and performs a series of automated checks. At all stages of development in the cell design, the user controls the actual functions performed, selecting those necessary to each phase of design. The SCELL program is not a synthesis program, but is a design verification program.

Automatic design rule checks and net checks are the most powerful functions of the SCELL program. Design rules including width and spacing, both for masks taken individually and unique combinations of masks taken together, are rigorously checked.

Interconnect nets are checked to ensure connectivity, detect shorts, and identify extraneous material. During the net check, process variables such as lateral diffusion and mask misalignment are simulated; and electrical parameters including actual device geometries, load capacitances, and coupling capacitances are calculated. These electrical parameters are used as inputs to the MOSTRAN program.

6.1.2 MOSTRAN Program

The MOSTRAN program provides a highly accurate simulation of circuit dynamic operations. This program is an automated design tool for use in analyzing MOS transient circuit performance and displays the output either as a list or as a plot of current (I), voltage (V), or power (P).

The dynamic simulation of MOS circuitry involves the simultaneous solution of a set of nonlinear differential equations. In the case of MOSTRAN, these are nodal current Kirchhoff's equations. The method of solution is a numerical integration technique utilizing a variable time increment to achieve reasonable accuracy within reasonable computing times.

While the user is not required to have any programming knowledge, or even a detailed familiarity with the method of solution of the differential equations, an awareness of the basic modeling equations is valuable because of inconsistencies in the values of process constants across industry. The evaluation equations are included here so that the user may determine his process parameters according to the equations used. The symbols used throughout this discussion are defined in Table 6.1.2-1.

The evaluation is based on the four time samples shown graphically in Figure 6.1.2-1. During normal operation, the voltages for each network of devices tied together by the user-defined net list is calculated by determining the total current charging the load capacitance of that network. The basic equation is:

$$dv = (I_{DC}dt_3 + Cdv)/CL$$

where I_{DC} is the dc current contribution into the net and Cdv is transient coupling into the net due to parasitic capacitances. All components of these two terms are based on known values of current and voltage at times t_2 and t_3 to calculate new values at t_4 .

TABLE 6.1.2-1 Symbols Used

SYMBOL	DEFINITION
I_{pc}	dc current into a net in mA
I_{ps}	Drain - source current in mA
C_{dv}	Parasitic capacitance times the voltage differential from times t_2 to t_3
t_1, t_2, t_3, t_4	Four times comprising time window in nsec
dt_1, dt_2, dt_3	Time differentials between time samples
C_L	net load capacitance in pF
C_{GS}, C_{GD}	Gate-source, gate-drain parasitic capacitances in pF
R_p, R_s	Drain and source resistors in kohms
V_G, V_s, V_D	Gate, source, drain voltages in volts
V_T, V_{Ti}	Threshold voltage, initial threshold voltage in volts
K, K_i	Gain, initial gain in mohos/volt ²
W, L	Width, length of device in hundredths of a mil
d	Lateral diffusion in hundredths of a mil

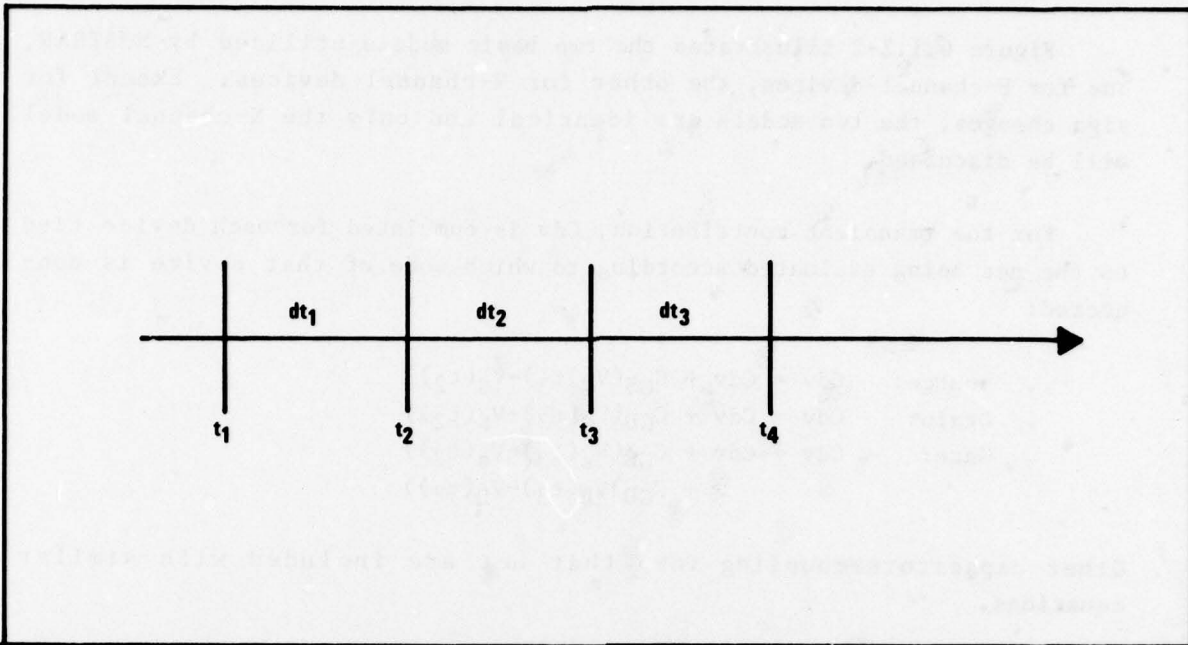


Figure 6.1.2-1. Time Window

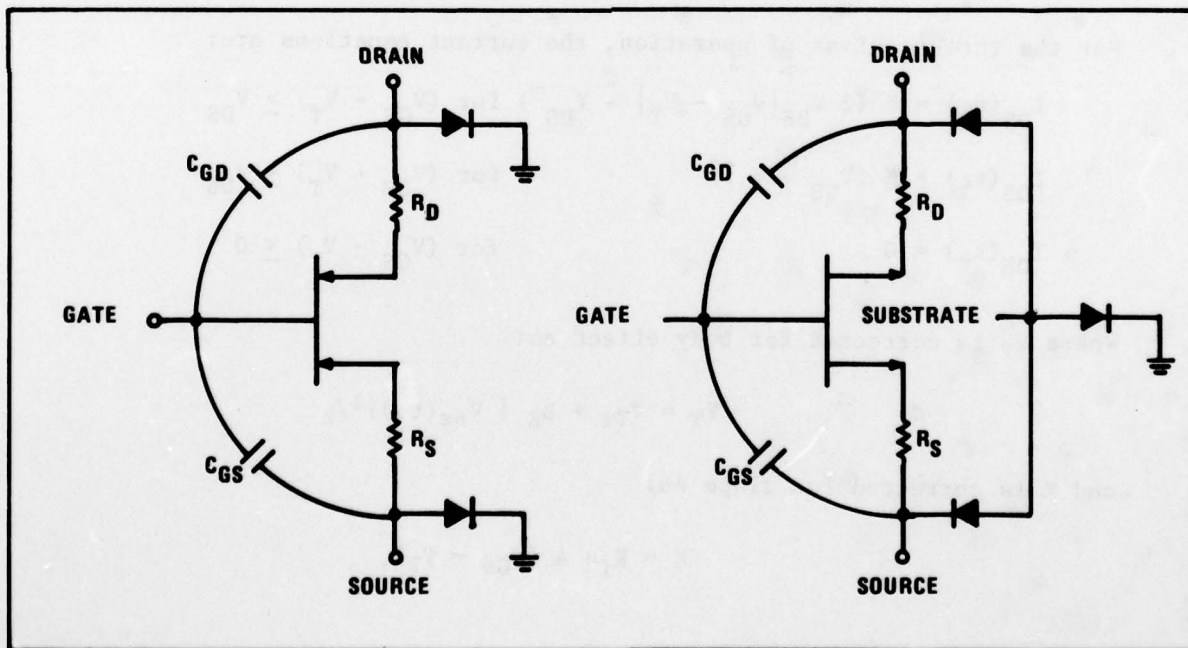


Figure 6.1.2-2. The P-Channel and N-Channel Models

Figure 6.1.2-2 illustrates the two basic models utilized by MOSTRAN, one for P-channel devices, the other for N-channel devices. Except for sign changes, the two models are identical and only the N-channel model will be discussed.

For the transient contribution, Cdv is cumulated for each device tied to the net being evaluated according to which mode of that device is connected:

$$\begin{aligned} \cdot \text{ Source: } & Cdv = Cdv + C_{GS}(V_G(t_3) - V_G(t_2)) \\ \cdot \text{ Drain: } & Cdv = Cdv + C_{GD}(V_G(t_3) - V_G(t_2)) \\ \cdot \text{ Gate: } & Cdv = Cdv + C_{GS}(V_S(t_3) - V_S(t_2)) \\ & + C_{GD}(V_D(t_3) - V_D(t_2)) \end{aligned}$$

Other capacitors coupling into that net are included with similar equations.

The dc current contribution is based on the equations developed by T.C. Sah, modified to account for source/drain resistors.

$$\begin{aligned} V_{GS} &= V_G - V_S - I_{ds}(t_3)R_s \\ V_{DS} &= V_D - V_S - I_{DS}(t_3)(R_s + R_D) \end{aligned}$$

For the three regions of operation, the current equations are:

$$\begin{aligned} I_{DS}(t_4) &= K (2 V_{DS} |V_{GS} - V_T| - V_{DS}^2) \text{ for } (V_{GS} - V_T) \geq V_{DS} \\ I_{DS}(t_4) &= K (V_{GS} - V_T)^2 \text{ for } (V_{GS} - V_T) \leq V_{DS} \\ I_{DS}(t_4) &= 0 \text{ for } (V_{GS} - V_T) \leq 0 \end{aligned}$$

Where V_T is corrected for body effect as:

$$V_T = V_{T1} + B_E |V_{ss}(t_3)|^{1/2}$$

and K is corrected for slope as:

$$K = K_1 e A (V_{GS} - V_T).$$

The initial value of K is corrected for lateral diffusion as:

$$K_1 = K^1 (W/L - 2d)$$

The equations for the diodes are simple exponentials and the discrete resistor model is evaluated with a straight forward $I = (V_1 - V_2) R$ equation.

Figure 6.1.2-3a illustrates how the time window is moved forward as the analysis proceeds in time with no error detected. Error control is based on a recalculation of the ΔV_2 based on the new currents just calculated. The difference between the old and new values of ΔV_2 is then compared against a two-sided threshold. If the error is less than the minimum threshold for all nets, the next dt is increased as shown in Figure 6.1.2-3b. If the error is greater than the maximum threshold for any net, however, the time window is stepped back as shown in Figure 6.1.2-3c, the dt is halved, and processing continues again from that point. Maximum dt is limited by the program to one-tenth of the smallest bit time or 10 nsec, whichever is smaller. The user may input initial dt , the maximum dt , and the two thresholds under General Parameters if he desires.

6.2 FUNCTIONAL DESCRIPTION

The user should be aware of three basic types of data when using the SCELL and MOSTRAN programs. These data types are:

- a. Rules data, which is data unique to a technology and is applied uniformly to all designs under that technology. It is initially defined, carefully checked, and then centrally stored for use by all designers. The integrity of this data is paramount since it affects all designs.
- b. Design data, which is unique to each cell design. It is defined during the design process and is changed frequently until a design is complete. Once a designer completes his design, this data must be protected against inadvertent change.
- c. Control data, which is transient in nature and changes from run to run. It controls program usage and is not retained once a design is complete.

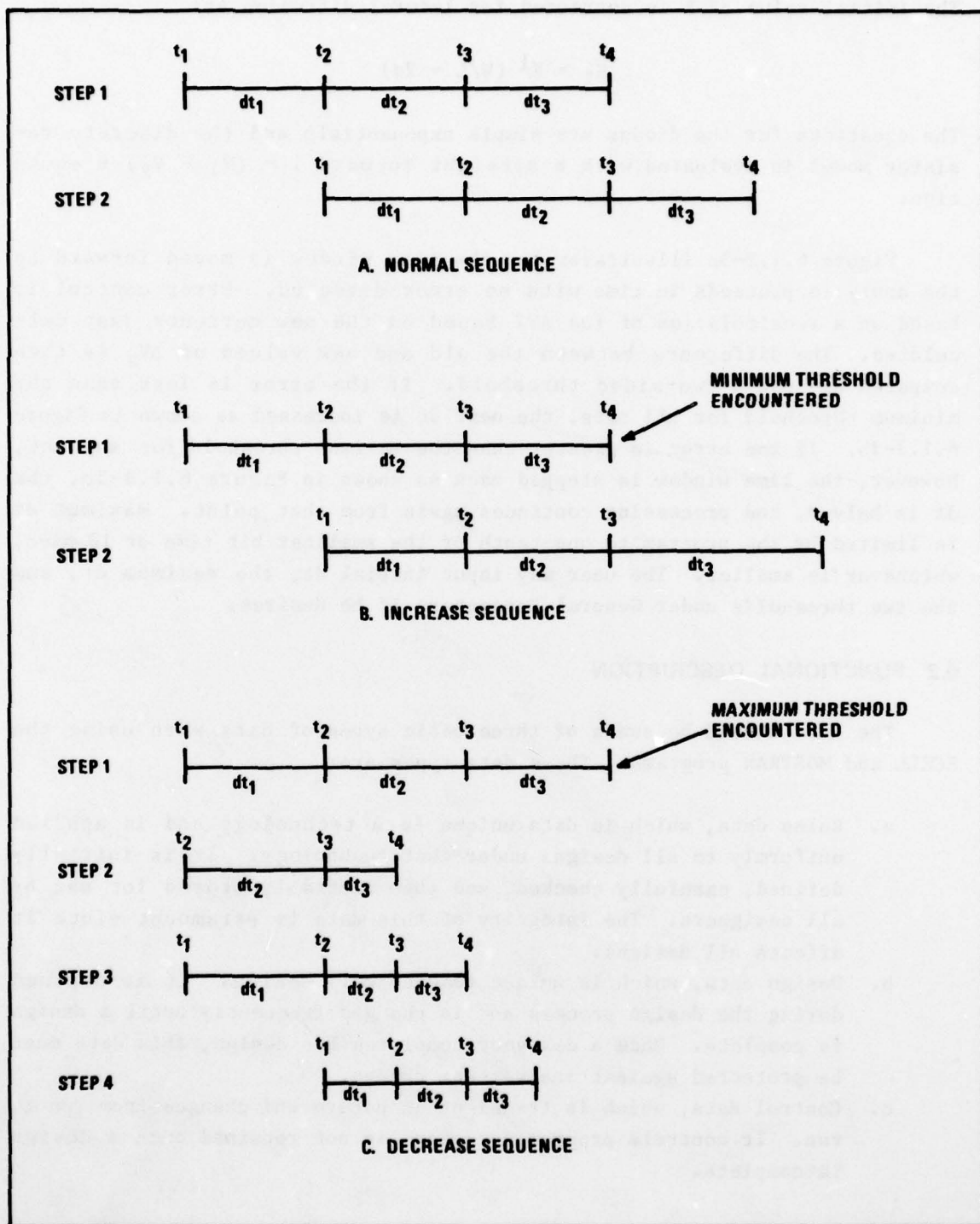


Figure 6.1.2-3. Time Step Sequence

Typical use of the SCELL and MOSTRAN programs is illustrated in the flow diagram shown in Figure 6.2-1.

Design of a cell family begins with the definition and testing of Rules Data. The following data categories are loaded in a MASTDESIGN file for the Technology Data Base for use by all the designers:

- a. CAPACITANCE
- b. COMPOSITE CHARACTERS
- c. DESIGN RULES
- d. GENERAL PARAMETERS
- e. GLOBAL PARAMETERS
- f. LIBRARY OF DEVICES
- g. TYPE/MODEL ASSIGNMENTS

Retrieval of this data is based on keys to the data base used at each phase of design.

After the MASTDESIGN library is established, the cell design (see Figure 6.2-1) can proceed based on the following rules:

- a. Each designer defines a design structure for each cell as shown in Figure 6.2-2.
- b. Each design is cataloged under the cell name with the following types of files allocated:
 - (1) DESIGN file -- working space for data defining the cell under design including PLACEMENT CONSTRAINTS, NET LIST, CIRCUIT PARAMETERS, INITIAL CONDITIONS, APPLIED SIGNALS, and OUTPUT CURVES Categories.
 - (2) GEOMLIB file -- cell geometries as output from the SCELL program.
 - (3) PARALIB file -- cell terminal descriptions as output from the SCELL program.
 - (4) XCAPS file -- cell electrical characteristics as output from the SCELL program.

A designer initiates his design typically by loading MOSTRAN input data on the DESIGN file for a rough approximation of circuit performance based on his estimates of the circuit parameters. If the designer has access to graphics equipment, his NET LIST can be derived automatically from the circuit schematic.

- a. Design Rule checks.
- b. Detailed Net check.
- c. Calculation of detailed Circuit Parameters, including all load and coupling capacitances. This data is written onto the XCAPS file for future use.

If there are no design rule or net violations, the designer can now replace his rough approximation of the Circuit Parameters by moving those on the XCAPS file to the DESIGN file and making a final MOSTRAN program run to determine if performance is as expected. The step 2 SCELL program run and the MOSTRAN program run are retained as final documentation of the design. The XCAPS file is no longer needed (its data having been incorporated into the DESIGN file) and can be released.

The design is technically correct at this point, and all that remains is to build the entries pertaining to this cell on the Technology Data Base. A final SCELL program execution (step 3) performs the following:

- a. Loads the DESIGN file onto the MASTDESIGN file from which all subsequent data is derived.
- b. Calculates total net capacitance and builds a Cell Parameter entry on the PARALIB file.
- c. Generates final artwork for the cell on the GEOMLIB file.

The step 3 SCELL program run is retained for final documentation. Since it derived its input data from the MASTDESIGN file, where it was filed under its family name, the final version of the data is retained for archival purpose. The PARALIB and GEOMLIB files include traces that identify the data and run number of the step 3 execution for configuration control. The DESIGN file, no longer needed since data now resides on the MASTDESIGN file, is now released.

Data on the PARALIB and GEOMLIB files are finally merged with the master files. MASTPARALIB, and MASTGEOMLIB, respectively; and the temporary files are released. The MASTMACROLIB file, containing functional descriptions of the cells for use in logic simulation, is prepared manually and carefully checked for accuracy. At the conclusion of a cell family design effort, all data finally resides in these four files on the Technology Data Base:

AD-A073 383

RESEARCH TRIANGLE INST RESEARCH TRIANGLE PARK N C
AFAL SIMULATION FACILITY/CAPABILITY MANUAL. VOLUME II. EXECUTIV--ETC(U)
FEB 79 R L EARP

F/G 9/5

F33615-76-C-1308

UNCLASSIFIED

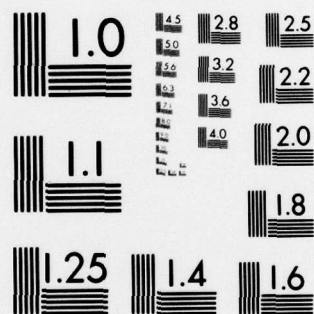
AFAL-TR-77-118-VOL-2

NL

2 OF 3

AD
A073383





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

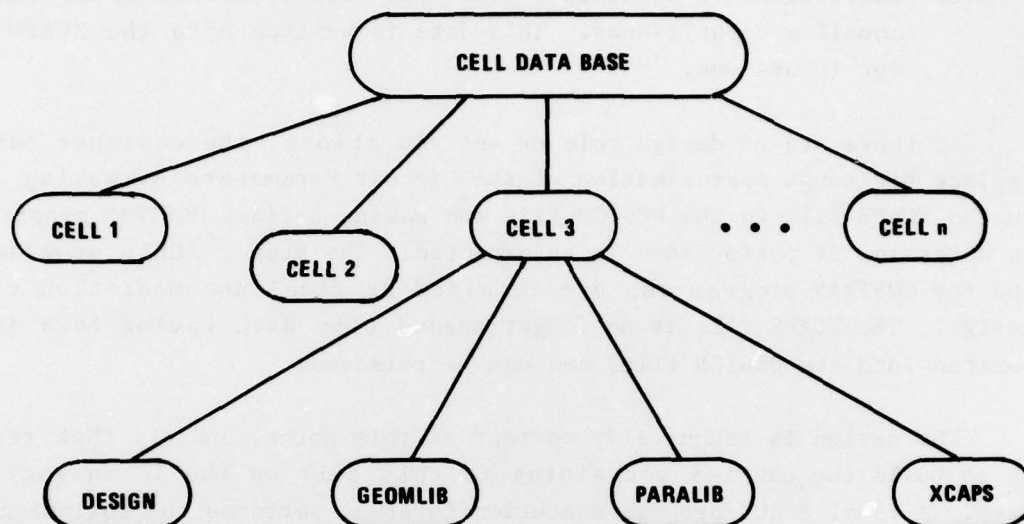


Figure 6.2-2. Cell Design Working Structure

Once the designer is satisfied with the estimates of transistor sizes and other parametric data, placement data is added to the DESIGN file manually from a hand-down layout. An initial program run (step 1) is now made through the SCELL program to perform:

- a. Input format checks.
- b. Initial net consistency checks.
- c. Printer plot.
- d. Generation of artwork on the GEOMLIB file for a pen and ink plot.

The major concern at this point is to determine if the cell is coded as intended before more detailed checks are performed.

Having successfully coded the cell placement, the designer proceeds to a more detailed analysis to the cell (step 2) by performing the following functions:

- a. Design Rule checks.
- b. Detailed Net check.
- c. Calculation of detailed Circuit Parameters, including all load and coupling capacitances. This data is written onto the XCAPS file for future use.

If there are no design rule or net violations, the designer can now replace his rough approximation of the Circuit Parameters by moving those on the XCAPS file to the DESIGN file and making a final MOSTRAN program run to determine if performance is as expected. The step 2 SCELL program run and the MOSTRAN program run are retained as final documentation of the design. The XCAPS file is no longer needed (its data having been incorporated into the DESIGN file) and can be released.

The design is technically correct at this point, and all that remains is to build the entries pertaining to this cell on the Technology Data Base. A final SCELL program execution (step 3) performs the following:

- a. Loads the DESIGN file onto the MASTDESIGN file from which all subsequent data is derived.
- b. Calculates total net capacitance and builds a Cell Parameter entry on the PARALIB file.
- c. Generates final artwork for the cell on the GEOMLIB file.

The step 3 SCELL program run is retained for final documentation. Since it derived its input data from the MASTDESIGN file, where it was filed under its family name, the final version of the data is retained for archival purpose. The PARALIB and GEOMLIB files include traces that identify the data and run number of the step 3 execution for configuration control. The DESIGN file, no longer needed since data now resides on the MASTDESIGN file, is now released.

Data on the PARALIB and GEOMLIB files are finally merged with the master files. MASTPARALIB, and MASTGEOMLIB, respectively; and the temporary files are released. The MASTMACROLIB file, containing functional descriptions of the cells for use in logic simulation, is prepared manually and carefully checked for accuracy. At the conclusion of a cell family design effort, all data finally resides in these four files on the Technology Data Base:

- a. MASTDESIGN file -- Contains all rules and design data accumulated during design. Saved for archival purposes.
- b. MASTPARALIB file -- Contains terminal characteristics of the cells for use as rules type data by the chip design programs.
- c. MASTGEOMLIB file -- Contains geometric descriptions of the cells for use in generating chip artwork.
- d. MASTMACROLIB file -- Contains logical or functional descriptions of the cells for use in simulation, test word generation, and test word verification of chips.

Data for the cell notebook is derived from these master files. Any data generated as a result of program execution contains traces that identify the job number and date of the run generating that data. This is retained in the final documentation.

6.3 SCELL PROGRAM FUNCTIONS

The following categories and directives are used in the SCELL program:

- a. BULK MANIPULATION Category
- b. CAPACITANCE Category
- c. COMPOSITE CHARACTERS Category
- d. CONTROL Category
- e. DESIGN RULES Category
- f. EXECUTE Directive
- g. GLOBAL PARAMETERS Category
- h. LIBRARY Category
- i. NET LIST Category
- j. PLACEMENT CONSTRAINTS Category
- k. RUN HEADER Category
- l. STOP Directive

6.3.1 BULK MANIPULATION Category

The BULK MANIPULATION category permits the orientation of an entire cell to be changed, or some portion of the cell to be altered in height or width.

6.3.2 CAPACITANCE Category

The CAPACITANCE category provides information needed to verify electrical interconnections within the cell and to calculate net-to-substrate and net-to-net capacitances. It also contains a description of how the as-drawn geometries are modified during actual processing.

6.3.3 COMPOSITE CHARACTERS Category

The COMPOSITE CHARACTERS category supplies data needed to produce a graphic representation of the cell on the line printer.

6.3.4 CONTROL Category

The CONTROL category specifies unit numbers of all input and output data files accessed or created by the program. This category also specifies which program options are desired. These options include information on program control, I/O units and general program parameters.

6.3.5 DESIGN RULES Category

The DESIGN RULES category checks that the geometries of a cell are valid within the constraints imposed by the minimum and/or maximum spacing between boundaries. This category also checks for invalid overlays of mask levels.

6.3.6 EXECUTE Directive

The EXECUTE directive specifies the end of input data, enabling the processing to begin. This directive also specifies the access keys to control the retrieval of data from the MASTDESIGN library.

6.3.7 GLOBAL PARAMETERS Category

The GLOBAL PARAMETERS category permits the user to specify device dimensional variables in addition to those specified on the first device description statement listed in the Library of Devices.

6.3.8 LIBRARY Category

The LIBRARY category permits the user to use the LIBBER library manipulation package during the cell input phase.

6.3.9 NET LIST Category

The NET LIST category specifies the electrical interconnections of devices, terminals, and buses in the cell by listing all the nodes connected to one another.

6.3.10 PLACEMENT CONSTRAINTS Category

The PLACEMENT CONSTRAINTS category determines where and in what orientation, devices, interconnects, buses, feed-thrus, terminals, polygons, and rectangles are placed to form a cell, and also provides the scale and size of the cell.

6.3.11 RUN HEADER Category

The RUN HEADER category identifies the cell being run.

6.3.12 STOP Directive

The STOP directive signifies the end of the program and terminates execution.

6.4 MOSTRAN Program Functions

The following categories and directives are used in the MOSTRAN program:

- a. APPLIED SIGNALS Category
- b. CIRCUIT PARAMETERS Category
- c. GENERAL PARAMETERS Category
- d. INITIAL CONDITIONS Category
- e. MERGE Category
- f. MODIFY Category

- g. NET LIST Category
- h. OUTPUT CURVES Category
- i. RUN HEADER Category
- j. SIMULATE Directive
- k. STOP Directive
- l. TYPE/MODEL ASSIGNMENTS Category
- m. UNIT NUMBERS Category

6.4.1 APPLIED SIGNALS Category

The APPLIED SIGNALS category describes the types of signals that serve as inputs to the circuit being simulated.

6.4.2 CIRCUIT PARAMETERS Category

The CIRCUIT PARAMETERS category contains all the parameter data about the circuit to be simulated.

6.4.3 GENERAL PARAMETERS Category

The GENERAL PARAMETERS category contains all the parameters common to a process and shared by the various designs using that process.

6.4.4 INITIAL CONDITIONS Category

The INITIAL CONDITIONS category permits internal nodes to be assigned an initial voltage.

6.4.5 MERGE Category

The MERGE category signals the MOSTRAN program to begin input from library device and to interconnect cells as specified by the SCELL statements.

6.4.6 MODIFY Category

The MODIFY category modifies a circuit just run without duplicating or reprocessing the entire input of statements.

6.4.7 NET LIST Category

The NET LIST category contains a set of strings of device node pairs describing the electrical connections between components. This category is shared with the SCELL program.

6.4.8 OUTPUT CURVES Category

The OUTPUT CURVES category describes the basic types of output requests that are permitted.

6.4.9 RUN HEADER Category

The RUN HEADER category identifies the cell being analyzed.

6.4.10 SIMULATE Directive

The SIMULATE directive specifies the end of input data and proceed with the setup and simulation phases of execution.

6.4.11 STOP Directive

The STOP directive signifies the end of the program and terminates execution.

6.4.12 TYPE/MODEL ASSIGNMENTS Category

The TYPE/MODEL ASSIGNMENTS category simulates the electrical characteristics of the devices on file in the Library of Devices.

6.4.13 UNIT NUMBERS Category

The UNIT NUMBERS category specifies the FORTRAN Logical Unit number to be used for input or output.

6.5 PROGRAM OUTPUTS

The expected outputs from the SCELL and MOSTRAN programs are listed in Tables 6.5-1 through 6.5-5 along with a description of the outputs.

TABLE 6.5-1 SCELL Program (Step 1) Output Description

OUTPUT	DESCRIPTION
PLACEMENT	Checks the PLACEMENT CONSTRAINTS category and provides warnings for improper PLACEMENT CONSTRAINTS .
INITIAL NET CHECK	Makes an initial net check of the program.
CELL COMPOSITE	Provides a graphic representation of the cell using the COMPOSITE CHARACTERS category.
ARTWORK GENERATION	Provides the values for each level of artwork generation.

TABLE 6.5-2 SCELL Program (Step 2) Output Description

OUTPUT	DESCRIPTION
PLACEMENT	Checks the PLACEMENT CONSTRAINTS category and provides warnings for improper PLACEMENT CONSTRAINTS .
INITIAL NET CHECK	Makes an initial net check of the program.
CELL COMPOSITE	Provides a graphic representation of the cell using the COMPOSITE CHARACTERS category.
FINAL NET CHECK	Checks Net-to-Substrate capacitances and circuit parameters.
DESIGN RULE CHECK	Checks the DESIGN RULES category and provides warnings for improper DESIGN RULES .

TABLE 6.5-3 SCELL Program (Step 2A) Output Description

OUTPUT	DESCRIPTION
PLACEMENT	Checks the PLACEMENT CONSTRAINTS category and provides warnings for improper PLACEMENT CONSTRAINTS.
INITIAL NET CHECK	Makes an initial net check of the program.
FINAL NET CHECK	Checks Net-to-Net Capacitances and Circuit Parameters taking lateral diffusion into account.

TABLE 6.5-4 SCELL Program (Step 3) Output Description

OUTPUT	DESCRIPTION
TOC	Lists the Table of Contents for Library unit specified.
PLACEMENT	Checks the PLACEMENT CONSTRAINTS category and provides warnings for improper PLACEMENT CONSTRAINTS.
INITIAL NET CHECK	Makes an initial net check of the program.
CELL COMPOSITE	Provides a graphic representation of the cell using the COMPOSITE CHARACTERS category.
FINAL NET CHECK	Checks Net-to-Net Capacitance and Circuit Parameters.
ARTWORK GENERATION	Provides final values for each level of artwork generation.

TABLE 6.5-5 MOSTRAN Program Output Description

OUTPUT	DESCRIPTION
<u>SETUP PHASE:</u>	
GENERAL PARAMETERS	Lists General Parameters and their values.
CIRCUIT PARAMETERS	Lists Circuit Parameters for devices used in design.
PARASITIC CAPACITORS	Lists the capacitors and device numbers.
TOTAL NET CAPACITANCE	Lists the capacitance values of the different nets.
<u>EXECUTION PHASE:</u>	
DELTA TIME	Gives minimum Delta time and maximum Delta time.
AVE POWER CONSUMED FOR PERIOD	Lists average current, peak current, and time for each applied signal.
LIST OF NODE VOLTAGES	Lists the node voltages for the different devices.
LIST OF DEVICE CURRENTS	Lists the device currents for the different devices.
PLOT OF NODE VOLTAGES	Provides a graphic plot of the node voltages.
PLOT OF DEVICE CURRENTS	Provides a graphic plot of the device currents.

SECTION 7 IC LAYOUT PROGRAMS

7.1 INTRODUCTION

The Computer Assisted Design (CAD) chip layout programs are designed to aid the design engineer in generating the data necessary for automated chip fabrication, starting with a logic design and utilizing cell definitions in the cell parameter library.

7.2 PROGRAM DESCRIPTIONS

The chip layout programs perform three primary functions: conversion of input data, chip layout, and checking of chip layout. These functions are performed by four programs:

- a. CONVRT (input data conversion)
- b. PRF (chip layout with automatic placement)
- c. CHPCHP (chip layout with user placement)
- d. MCHPCK (layout checking)

The interaction between these programs is illustrated in the block diagram shown in figure 7.2-1. Figure 7.2-2 depicts the functional relation between these four programs and the remaining CAD programs.

7.3 PROGRAM OPERATION

Data required to run the programs is fed to CONVRT to be processed and put in a form acceptable to PRF. The PRF program then accesses the files produced by CONVRT and processes these files to place the cells, connect them as required, provide input and output pads, and to provide data necessary for artwork preparation and plotter display input, and to provide a line printer display of chip layout. The CHPCHP program may be used to modify the PRF placement or, alternatively, to allow the user to design his own chip. The resulting output of PRF (or CHPCHP) is then applied to the MCHPCK program, which tests these results for calculation of capacitance values of all wires and cell inputs.

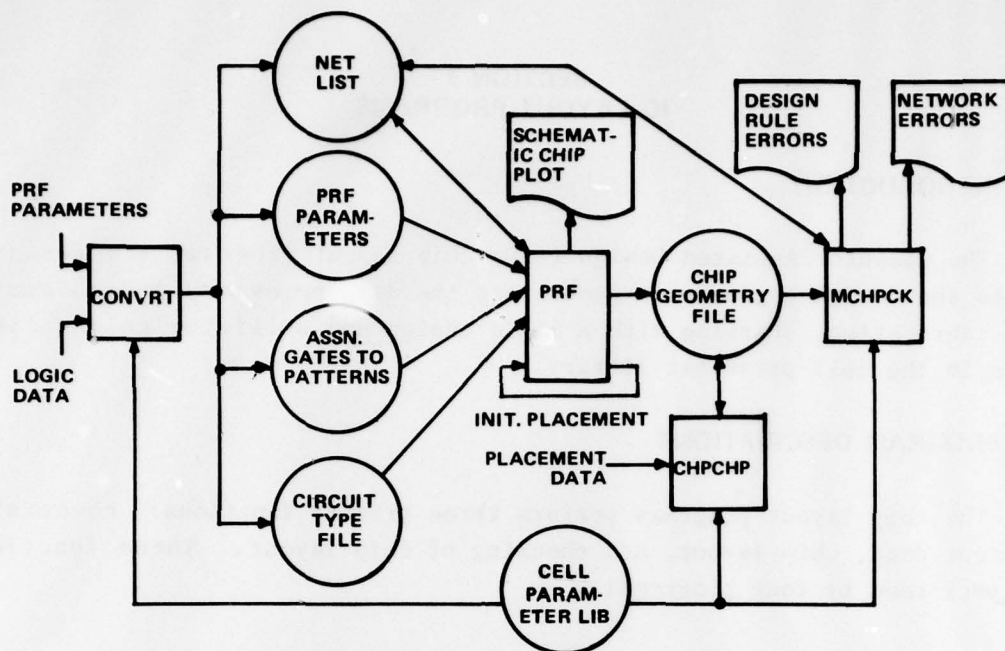


Figure 7.2-1. Chip Layout Programs, Block Diagram

7.4 INPUT DATA CONVERSION—CONVRT

When the engineer is sufficiently satisfied with his logic design and simulation to begin laying out a chip, he must select cells from the cell parameter library to perform the logical operation required by each element of his design. During simulation in LOGIC 4, signal names are assigned to each interconnection between elements, chip inputs, and chip outputs. Also, a unique tag is assigned to each element, chip input, and chip output. Although the data used for logic simulation may be sufficient to define a chip when simulation is completed, it is not structured properly for input to the chip layout programs. The resulting cell types (pattern numbers), signal names, and tags constitute the logic file which is input to CONVRT. Restructuring these data is the function of the input data conversion program, CONVRT.

The inputs to CONVRT are the logic deck, the PRF input parameters, and the cell parameter library. These inputs are processed by CONVRT to

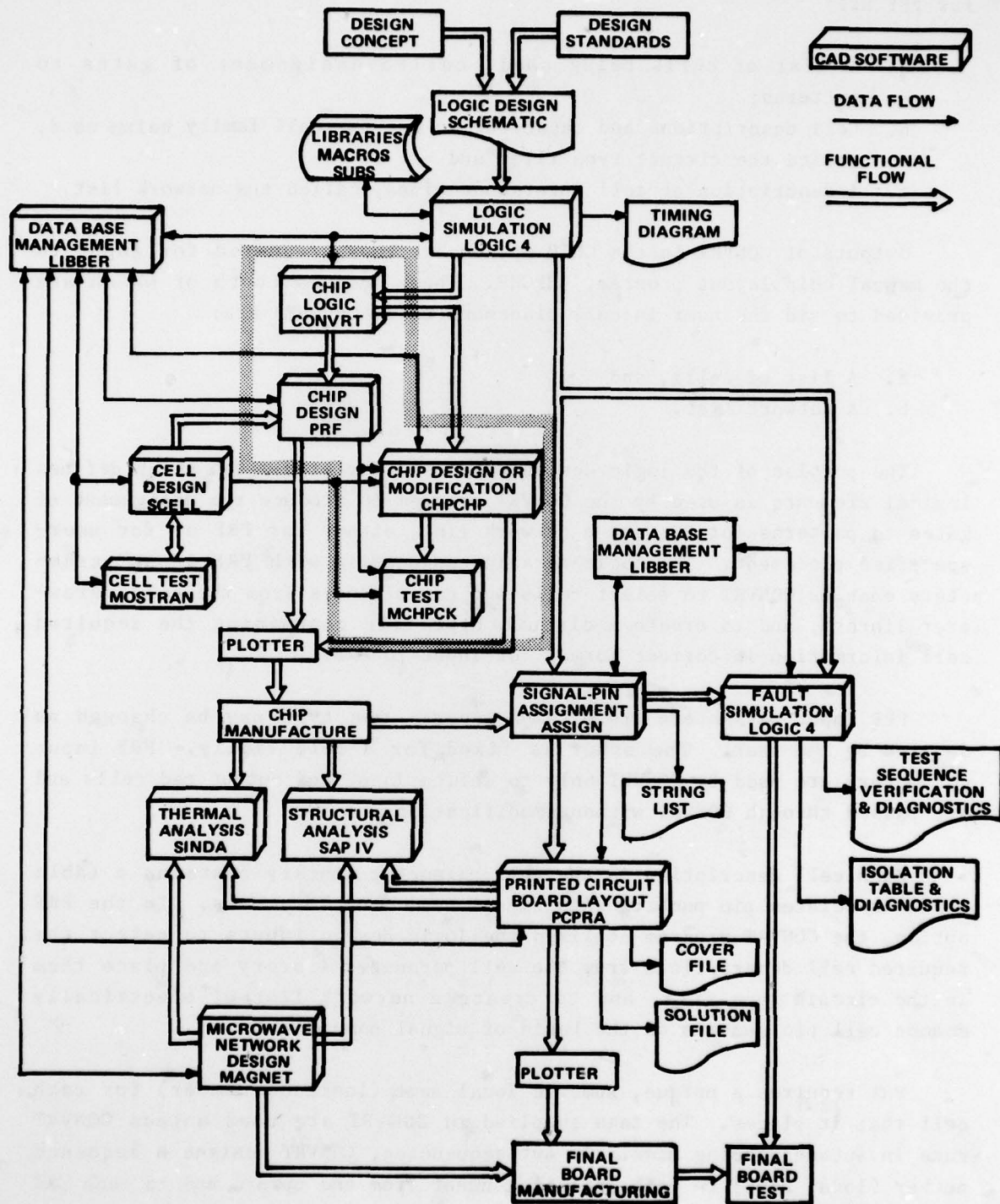


Figure 7.2-2. CAD Functional Block Diagram Emphasizing the Chip Layout Programs

produce inputs to the PRF or CHPCHP program. Outputs of the CONVRT program for PRF are:

- a. A list of cells being used, called assignment of gates to patterns;
- b. Cell descriptions and capacitances for the cell family being used, called the circuit type file; and
- c. A description of cell interconnections, called the network list.

Outputs of CONVRT in the CHIP option are those required for input to the manual chip layout program, CHPCHP. These outputs (both of which are provided to aid the user in cell placement using CHPCHP) are:

- a. A list of cells, and
- b. A network list.

The portion of the logic design data used for simulation which defines logical elements is used by the CONVRT program to produce the assignment of gates to patterns for PRF and a network list, either for PRF or for user-specified placement. The logic data in conjunction with PRF input parameters enables CONVRT to select cells and capacitances from the cell parameter library, and to create a circuit type file containing the required cell information in correct format for input to PRF.

PRF input parameters are of two types. One type may be changed as desired by the user. The other is fixed for a cell family. PRF input parameters are used by CONVRT only to define input and output pad cells and are passed through CONVRT without modification.

Each cell description in the cell parameter library contains a table which associates pin numbers with signal name field numbers. In the PRF option, the CONVRT program utilizes the logic design inputs to select the required cell descriptions from the cell parameter library and place them in the circuit type file, and to create a network list of electrically common cell pin numbers on the basis of signal name fields.

PRF requires a unique, numeric local name (instance number) for each cell that it places. The tags supplied to CONVRT are used unless CONVRT runs in autosequencing mode. In autosequencing, CONVRT assigns a sequence number (local name) to each logical element from one upward and to each pad from 298 downward (for a maximum of 298 calls/gates per chip).

The logic design data supplied to CONVRT must include a statement for each logical element and pad, defining a signal name for each element input and output. Logical elements are named using cell types in the cell parameter library. Input and output pads are called IPIN and OPIN, respectively. If an output buffer is included in a pad cell, the OPIN and buffer cell statements are given the same tag number, and CONVRT defines a single pad cell. Otherwise, standard pad cells from the PRF input parameters are used.

The CONVRT program may also be run with the CHIP option. In this option, CONVRT produces only a network list which can be used with MCHPCK, and a list of cells required. The user must supply placement data to CHPCHP.

7.5 CHIP LAYOUT

Chip layout may involve any of three processes: automatic cell and interconnect placement performed by PRF, chip modification aided by CHPCHP, and user-specified placement aided by CHPCHP.

PRF performs automatic cell placement and wire routing on the basis of input parameters for a particular cell family technology.

The initial inputs to PRF are all outputs of CONVRT: circuit type file, assignment of gates to patterns, network list, and PRF input parameters. The following outputs are available from PRF:

- a. Chip geometry file, used in chip fabrication.
- b. Network list, which includes network capacitances. Due to automatic reassignment of electrically equivalent cell pins, this list may be different from the input list.
- c. Initial placement matrix and interchange map, for use in iterative runs of PRF.
- d. A schematic plot of the chip.

CHPCHP can be used either to modify a chip layed out by PRF or to design a custom chip based on manually created input placement data. If CHPCHP is used to modify a chip, a chip geometry file created by PRF is supplied to CHPCHP. Additional data to modify the chip must be supplied by the user. To create a chip using CHPCHP, the user must supply data defining locations to allow placement of cells from the cell parameter

library and to place interconnect material. The output of CHPCHP is a chip geometry file.

Chip geometries are defined in terms of X-Y coordinates in an orthogonal grid. Chip geometry data consists of component sets and level sets.

7.5.1 Component Sets

Component sets specify the placement of cells (logical elements, via holes, and pads) on the chip. Each entry of a component set defines the X and Y chip coordinates of the cell reference point and cell orientation. Orientation, specified by a digit between 0 and 7 refers to the rotation or reversal (mirror-imaging) of the cell with respect to its definition in the cell parameter library.

7.5.2 Level Sets

Level Sets - which include line, shape, and symbol sets - specify the placement of material other than standard cells on each level. This includes interconnect material, borders, and legends. A line set consists of a line width followed by pairs of entries, each pair giving the coordinates of the end points of a line. A shape set consists of the coordinates of the corners of a polygon. The polygon is formed by drawing lines between consecutive coordinates, and finally by connecting the last coordinate with the first. All lines in a shape set must be parallel to the X or Y axis. Symbol sets and complete text editing capabilities are available for placing legends on the chip artwork.

7.5.3 Automatic Placement

PRF places cells and wiring automatically on the basis of rules for a standard cell family technology which are selected by the PRF input parameters. Automatic placement of cells is accomplished on a 0.1-mil XY coordinate system; resistance is in ohms, and capacitance is in pF times 10^{-4} . (Note that capacitances listed in the circuit type file are in pF times 10^{-3} .) Placement and routing is performed in three steps: preplacement, final placement, and wire routing.

PRF preplacement is optional. If performed, it results in an initial placement matrix and an interchange map. The initial placement matrix is a matrix of cell local names showing cell relative position. Cell rows are shown vertically in the matrix. It always has an even number of cell rows,

and may contain dummy rows. The interchange map is a corresponding matrix of digits indicating which cells are permitted to be interchanged during final placement. Any cell positions with identical, non-zero digits on the interchange map may be interchanged. Cells within groups may be interchanged within the group (but not with cells outside the group) by assigning a unique digit to cells in the group. A zero prohibits interchange. Pad cells must have a value of two on the interchange map.

During final placement, cells as well as logically equivalent cell pins are interchanged by the program to improve wire routing. The wire routing routines assume two levels of wiring, one predominantly horizontal and one predominantly vertical. When placement and routing are completed, a new initial placement matrix and interchange map are generated reflecting placement results. These may be used as inputs to a second run of PRF, bypassing automatic preplacement, to attempt to further improve the layout.

The PRF input network list produced by CONVRT includes a weight for each net, initially set to 100. If optimum wiring for certain nets is critical, increasing weights for these nets will cause PRF to give them special attention.

PRF prints an output net list, reflecting final routing and including net capacitances between levels, for the total of cell pin and wiring capacitances. Network capacitances are derived from capacitance per unit area inputs from the cell parameter library.

As already explained, the circuit type file and network list inputs to PRF are generated by CONVRT, and PRF input parameters are passed through CONVRT without change. Most PRF input parameters are fixed for a technology. Parameters are in card image format, and consist of 11 cards with 18 four-column, right-justified fields per card. Only certain parameters may be changed for a given technology by the user.

7.5.4 Chip Modification

The CHPCHP program may be used to modify the chip geometry file output of PRF. CHPCHP has two capabilities for the modification of chip geometry, bulk manipulation, and additional placement inputs.

Bulk manipulation provides the capability to re-orient the chip, or to expand or contract the chip area in height or width with respect to a cut-line through the chip. A cut-line consists of horizontal and vertical line segments and is specified by the X-Y coordinates of points where the segments meet. A height cut-line is drawn from left to right and a width cut-line is drawn from top to bottom of the chip. Height is changed above the cut-line and width is changed to the right of the cut-line. An example is shown in figure 7.5.4-1.

Figure 7.5.4-2 illustrates another use of the bulk manipulation commands in CHPCHP. In this example it is assumed that excessive crowding of interconnect material has resulted in design rule spacing violations. To reduce the crowding along the X-axis at Y=3, the chip is being expanded along a cut line X=0, Y=3. After chip expansion, CHPCHP automatically spreads out the existing interconnect material to utilize the additional chip space.

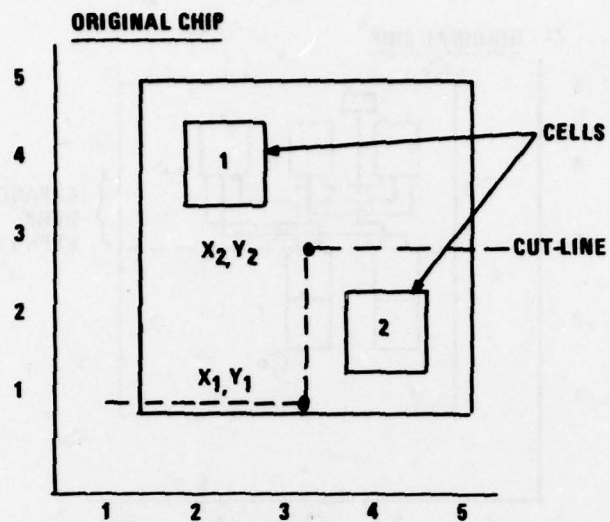
Chip orientation is specified in the same manner as cell orientation with a digit between 0 and 7.

Additional cells (logical elements, pads, and via holes) and interconnections may be placed on a chip by means of placement inputs to CHPCHP. The details of placement inputs are described below under user-specified placement. Unused components to be removed from the chip must be edited out of the geometry data by a text editor prior to inputting the file to CHPCHP. All placement inputs are incorporated into the chip geometry file before bulk manipulation is performed.

Since placement inputs are incorporated prior to bulk manipulation, it may be advantageous to use successive runs of CHPCHP when adding cells to a chip: one run to expand the chip, and a second run to place new cells, etc. Proceeding in this fashion eliminates some diagnostics which might otherwise result.

7.5.5 User-Specified Placement

By providing placement inputs to CHPCHP, the user can specify exactly the placement of material on a chip through a more flexible process than direct specification of component sets, line sets, and shape sets in the chip geometry file. Placement specifications to CHPCHP define cells and



AMOUNT OF CONTRACTION: 2

CUT-LINE SPECIFICATION:

3	1	3	3
x_1	y_1	x_2	y_2

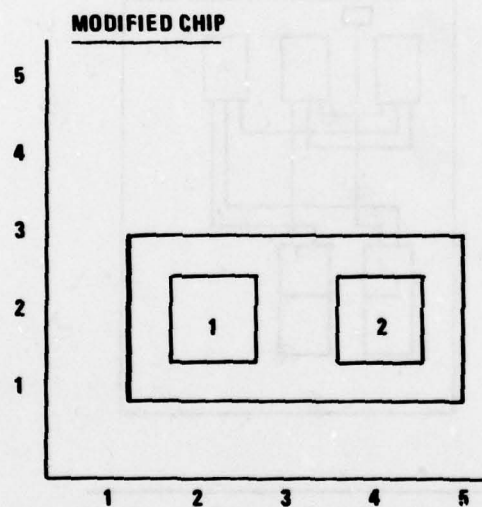


Figure 7.5.4-1. Example of Height Contraction

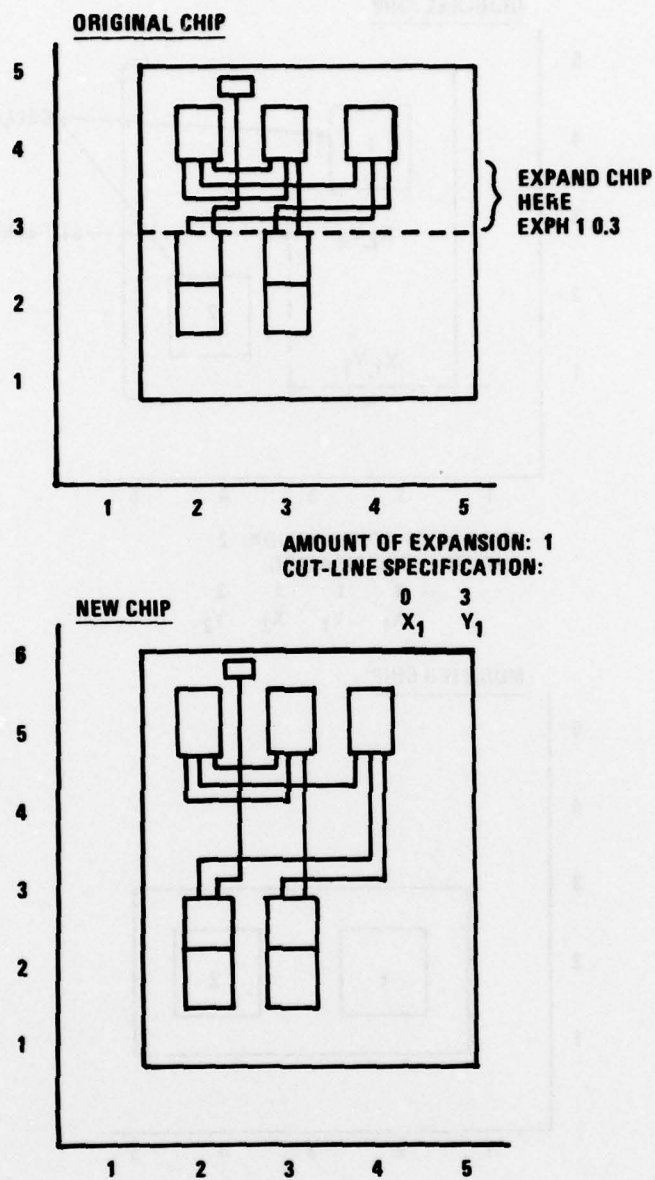


Figure 7.5.4-2. Example of Height Expansion to Reduce Interconnect Crowding

describe locations on the chip for both cells and interconnect material. User placement allows for defining diagonal line segments, thereby possibly shortening interconnect length; whereas, automatic placement places interconnect material horizontally or vertically and, in general, does not use diagonal line segments.

Cell definitions consist of a local name (instance number), library name (cell type, pattern number), and orientation digit. The location of cells or interconnect material may be specified either absolutely with respect to the chip origin, or relatively with respect to a previously-defined location.

Relative cell placement is specified in terms of the X and Y offsets from a previously-placed cell origin or node (i.e., pin).

Relative interconnect placement (either pin-to-pin connections or bus material placement) is specified in terms of successive points along the interconnection. Points may be specified with the following terms:

- a. TER -- a cell pin or bus node (bus nodes are numbered downward from 99).
- b. OFS -- X or Y offsets from the preceding point.
- c. VIA -- a via hole placed at the location of the preceding point.

In relative bus placement, the TER specification causes material to be placed so that it crosses the specified cell. For cell pin interconnections, placement with the TER specification does not cross the cell.

7.6 LAYOUT CHECKING--MCHPCK

After a chip has been layed out by PRF or by CHPCHP, the designer will probably wish to check the layout for obvious errors, either with the aid of the schematic plot supplied by PRF or with a plot produced by a graphic plotter if available. For a more detailed check, MCHPCK is available to check the chip layout against design rules defining spacing constraints for the technology being used.

In addition to the design rules, MCHPCK must be supplied with the chip geometry file, the network List, and cell family capacitances. If CHPCHP is used to lay out the chip, the network list must be generated by CONVRT or be created manually, since CHPCHP does not produce a network list.

MCHPCK supplies error listings for both design rule errors and network errors, as well as network capacitances. On the basis of these outputs the designer can modify the chip layout and recheck it until it is satisfactory.

Checks performed by MCHPCK are as follows:

- a. Initial Net Check - A consistency check between the network list describing the logic being implemented and the component set.
- b. Final Net Check - A detailed trace of the line sets and shape sets to determine if the physical connections agree with the network list, computation of capacitance for each net, reporting of all stray matter, and construction of a net list showing actual connections.
- c. Design Rule Checks - Checks of cell to cell, cell to interconnect, and interconnect to interconnect spacing and interconnect material width.

It should be noted that certain simplifying assumptions are made in these various checks, and the user should be aware of these assumptions and the limitations of the program with regard to checks it performs. Under certain conditions, manual checking is desirable and necessary. Assumptions made are described in the following paragraphs.

7.6.1 Initial Net Check

The initial net check is a one-to-one consistency check between the input network (net) list and the component set produced by PRF or CHPCHP. Each cell/terminal pair in the network list is checked to determine if a corresponding terminal exists for the instance of that cell in the component set. Conversely each terminal for each cell in the component set is checked to determine if a corresponding cell/terminal pair exists in the network list. Uniqueness is required. Duplication of cell instance names in the component set and duplication of cell/terminal pairs in the network list are diagnosed as errors.

7.6.2 Final Net Check

The final net check consists of a detailed tracing of the line and shape sets of the chip geometry file (built by PRF or CHPCHP) to determine if the physical locations of the cell nodes and interconnecting material

provide electrical connections as specified by the net list. The nets are checked for the following:

- a. Continuity (no fragmentation).
- b. Short circuits between nets.
- c. Existence of extraneous material (matter not connected in any net).

Error messages are printed for all fragmented or connected (shorted) nets, and network capacitances are calculated and printed. Additionally, the locations of all stray matter, which is not a part of a net, is listed.

7.6.2.1 Orthogonal Interconnection

Horizontal or vertical interconnect segments defined by line or shape sets must touch or overlap (on the same level) to be electrically connected. For line sets, the centerline and width is defined in the chip geometry file. MCHPCK treats each line segment as a rectangle surrounding the centerline of the segment.

Coordinates specified in shape sets are treated as the outline of a polygon. All material is inside this outline. All shape sets are constrained to be orthogonal: that is, no diagonal lines are permitted in shape sets.

7.6.2.2 Diagonal Interconnection

The only non-orthogonal geometries permitted are line sets. The test for connectivity of diagonal lines is not as rigorous as for orthogonal lines: only the two end points are checked. For two entities to be electrically connected, one end point of the diagonal segment must touch material on the same level. For two diagonal segments to be considered to be electrically connected by the program, their end points must exactly coincide. Figure 7.6.2.2-1 illustrates a weakness of this assumption where two segments actually touch but are diagnosed as an open circuit. Where connectivity was desired, this is a valid assumption since it guarantees that no gaps exist at the point of connection such as the one shown in the illustration. Where connectivity was not desired, this is not a valid assumption since a short does exist between the illustrated line segments.

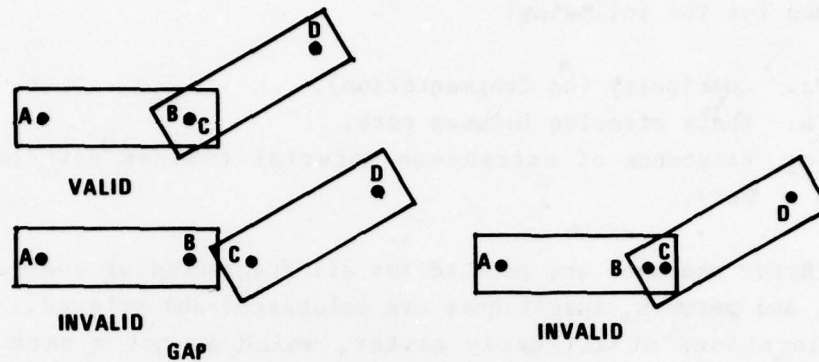


Figure 7.6.2.2-1. Invalid Connections Between Orthogonal Segment A, B, and Diagonal Segment C, D.

A second weakness of the end-point-coincidence assumption occurs where two diagonal segments cross each other on the same level, but are diagnosed as unconnected. This case and the previously described case must be diagnosed manually.

7.6.2.3 Connections to Cell Terminals

The location of cell terminals is identified by TER card images in the cell parameter library. Cell terminals are points of electrical interconnection (interconnect) material where levels of valid connection are specified in the overlay field (field 3) of the TER card image. For electrical connectivity, an orthogonal interconnect must touch or surround the point specified in the TER card image and be on the level specified in the overlay field.

A weakness of the assumption is that, within the cell, the terminal is actually interconnect material having some dimensions. Thus, a valid electrical connection can exist without actually touching the point specified as the terminal; whereas, the program would diagnose the connection as an open circuit. On the other hand, the designer might inadvertently place an interconnect too close to a terminal, but not touch the point specified by the terminal. In this case, the program would not detect a possible short. In general, any time the designer crosses a cell boundary with interconnect

material which does not connect directly to a terminal in the normal sense, he is constrained to manual checking for complete assurance of a valid connection.

7.6.2.4 Diagonal Connections to Cell Terminals

Connections between diagonal line segments and cell terminals must exactly coincide since only end points are considered for diagonal line segments. If the end point of a diagonal segment misses the terminal point, or if the interconnect is not on a level specified on the overlay field for that terminal, the apparent connection is diagnosed as an open circuit. Again, a valid connection can exist without actually touching the end point of the line and the terminal.

7.6.2.5 Terminal to Terminal Connections

Occasionally connectivity between cells is established by physically overlapping cells. For this case, a terminal on one cell must exactly coincide with a terminal on the other, and at least one level specified in the overlay for each must agree. Any other situation results in a diagnostic identifying an open circuit.

7.6.2.6 Crossover (XOVER) Connections

Within a cell, two points that are electrically common are specified in the cell parameter library as an XOVER. Each point can have only one level of legitimate electrical connection associated with it. Rules for the two points are identical to those for terminals.

7.6.2.7 BUS Connection

BUS connections to a cell are specified in the cell parameter library as a connection extending completely across the cell on a specific level of interconnect. For a valid electrical connection, the interconnect material must completely cover the line defined in the BUS card image and must be orthogonal. The interconnect material must also be on the level specified on the BUS card. Any deviation from this results in a diagnostic.

Two weaknesses exist as a result of these assumptions. First, the designer having knowledge of the internal geometries of the cell may elect to not completely cross the cell with interconnect material. If he does

so, he should manually inspect the chip geometry outputs to account for the resulting diagnostic. Second, the requirement that the interconnect merely cover the bus line may allow an interconnect that is slightly askew to pass without a diagnostic. The designer should check to insure that the center-line of the interconnect exactly coincides with the bus line and that it is of the proper width.

7.6.2.8 Logically Equivalent Inputs

It is frequently desirable to interchange physical connections to logically equivalent inputs of a cell (such as a two-input NAND gate) to improve chip layout. However, if such a change is made, the MCHPCK program will produce diagnostic messages unless the net list used by MCHPCK is the one produced by the PRF run which also produced the chip geometry file. Two other alternatives are to interchange signal names in the LOGIC 4 deck or to manually account for such diagnostic messages.

7.6.2.9 Extraneous Material

Any interconnect material which is not identified as part of a net is diagnosed at the conclusion of the final net check. The (X, Y) location of the lower left-hand corner of the material is given along with any capacitance for that material. The capacitance output also serves to identify what level the material is on. Unused terminals of cells are also identified as extraneous material with their capacitance listed under the library column.

7.6.2.10 Overall

As long as a designer places and interconnects cells in a straightforward manner, the final net check is extremely valuable in verifying the logical correctness of the physical chip layout. Manual checks should be unnecessary unless the designer uses shortcuts such as connecting to a cell at a place other than a terminal or crossing diagonal lines. To the maximum extent possible, the program diagnoses a problem if uncertainty exists. The user is cautioned, however, that this is not always possible as pointed out in the previous paragraphs.

7.6.3 Capacitance Loading Output

The final net check provides as an output the detailed loading on each net. The capacitive loading calculation is based on data describing capacitance as a function of the overlay of the masks normally stored in the cell parameter library for each technology.

The capacitive loading output identifies each component contributing to the total loading: capacitance internal to the cell as well as capacitance for each level of interconnect material taken separately. The sum of these for a given net is the total capacitance of that net. Where a net is fragmented, the capacitance output listed is only for the identifiable portion of the net. The user should check the list of extraneous material for any additional stray portions of that net if he needs to know the total capacitance of the net. (Note that for uniformly defined line widths, the resistance of each net may also be derived from the capacitance output.)

Network capacitances are computed from capacitance per unit area between each interconnect level and the substrate for the cell family being utilized. Capacitance per unit area is normally defined in the cell parameter library in terms of overlays.

An overlay is a means of defining the presence or absence of material at a point on each level. Overlay definition is necessary because intervening material may effect capacitance between an interconnect level and the substrate.

7.6.4 Output Net List

The net list output of MCHPCK is constructed using the actual physical layout of the chip as defined by the chip geometry file. This list agrees exactly with input net list if the chip is completely and accurately laid out. In the event of wiring problems, however, this list differs from the input list and shows exactly what is interconnected and is useful in diagnosing problems. A user may elect to leave power busses out of his net list since every cell is connected together. A check of the output net list then shows whether all cells were tied together. (Note that for this case, each power bus is identified as extraneous material.)

7.6.5 Design Rule Check

The design rule check inspects spacing in the chip geometry file using the rules for the cell family being utilized. All violations in wiring width and cell-to-cell, cell-to-wiring, and wiring-to-wiring spacing are printed.

The design rules are global in nature and apply equally to all cells. Where specific rules apply to a unique cell, such as a wiring via or bus crossing cell, design rules for the cell are included in the cell parameter library for that cell and override global rules. The following design rule checks are made:

- a. Cell to Cell Spacing.
- b. Cell to Interconnect Spacing.
- c. Interconnect to Interconnect Spacing and Width.

7.6.5.1 Cell to Cell Spacing

Minimum spacing between cells is specified in the BDRY card in the cell parameter library for each edge (North, South, East, and West), once in the design rule data category for all cells, and then for each cell that differs from the global rules. Spacing is assumed to be uniform along each edge, and the value listed represents the worst case spacing from what is known to be inside that edge of the cell to any edge of any other cell. For example, if diffusion to diffusion spacing of .4 mil is the limiting case for a given edge and a diffusion exists .15 mil inside that edge, then the permissible cell to cell spacing for that edge is .25 mil. Any cell spaced closer than that results in a diagnostic message. A finite space must exist in order for the cell-to-cell spacing check to be performed. Overlapping or exactly coincident cells are not checked. Negative spacings are not permitted.

Sometimes a designer, knowing what is internal to a cell, reduces component spacing below that specified by the rules. Likewise, designers may overlap cells. Reduced or overlapping placement is generally used for fairly simple structures such as wiring vias, end caps and bus crossers. For other cells, the designer often uses (or should use) terminals in any area of overlap where electrical connection is required. These terminals

will serve as alignment points to insure proper placement as a consequence of the final net check. In all cases of reduced spacing, manual checks are required.

7.6.5.2 Cell to Interconnect Spacing

Cell to interconnect spacing is also specified in the BDRY card images for each edge of a cell. This spacing can be more accurately defined since the actual material adjacent to a cell is known. Once again uniformity is assumed along each edge and no checks are performed for coincidence or overlap. It should be noted, however, that spacing is not checked for diagonal interconnect segments. For each of these, manual checks are required.

7.6.5.3 Interconnect to Interconnect Spacing and Width

Interconnect spacing and width are specified in the design rule data and are applied uniformly across the chip. Overlapping line or shape sets are permissible and are treated as a resulting larger polygon. Only minimum spacings and width are checked. A designer wishing to increase the width of certain interconnects, to reduce resistance for example, must check these larger dimensions manually. No checks are performed on diagonal line segments and these segments should also be checked manually.

7.6.5.4 Diagnostics

The exact coordinates at which a violation occurred is always printed with as much other pertinent information as possible: for example, which cell and which level of interconnect. Cells not having unique instance names, bus crossers for example, may cause some confusion. Use of unique names in the component set is encouraged, but lacking this, the coordinates uniquely identify the geometries in question.

7.6.5.5 Overall

Just as for the final net check, the designer who performs his cell placement and wiring in a straightforward manner will find the design rule checks to be extremely valuable diagnostic tools. To the maximum extent possible, the program diagnoses a problem if uncertainty exists. The user is cautioned, however, that this is not always possible as indicated above. (For a more detailed description of design rules, refer to the handbook for the cell family being used.)

SECTION 8 MICROWAVE NETWORK ANALYSIS AND OPTIMIZATION PROGRAM—MAGNET

8.1 INTRODUCTION AND GENERAL DESCRIPTION

The MAGNET program allows computerized analysis and optimization of a large class of linear microwave networks for gain VSWR, phase linearity and noise figure. The MAGNET program provides the microwave designer with the means to optimize and analyze his design. The program does not actually design the microwave circuit; it only relieves the user of much of the complex mathematics involved in the design. Figure 8.1-1 depicts the functional relation between the MAGNET program and the remaining CAD programs.

The MAGNET program is capable of modeling any network that can be built from pairs of linear two ports. The program is used to minimize input VSWR, output VSWR, phase deviation from linearity, noise figure, and gain error, in any combination, for a network in a 50-ohm system. MAGNET is designed primarily for microwave networks; however, it may also be used for classical filter design, if suitable scaling is performed. The MAGNET program is used primarily for design of linear two port microstrip; however, it may also be used in strip-line and lumped element design.

The use of MAGNET involves three main parts which are shown in Figure 8.1-2. The main parts are:

- a. Input Data.
- b. MAGNET Program.
- c. Output Data.

8.1.1 INPUT Data

The Network Description and device S parameters are input to the program via input data files. The primary file contains the network description and desired performance criteria. The device S parameters may also be input as part of the primary file or called by the primary file from a secondary file.

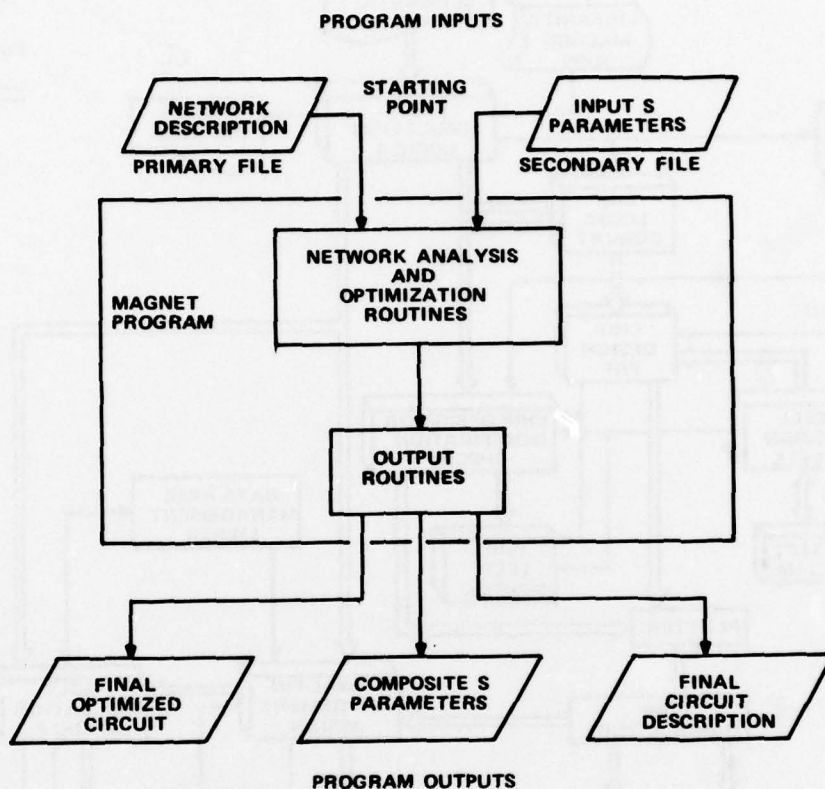


Figure 8.1-2. MAGNET Program General Flow Diagram

8.1.2 MAGNET Program

The program consists of network analysis, optimization, and output routines. The circuit input data is first analyzed for the desired performance then optimized toward the desired results. After the optimization function is complete, the output routines produce three output data files.

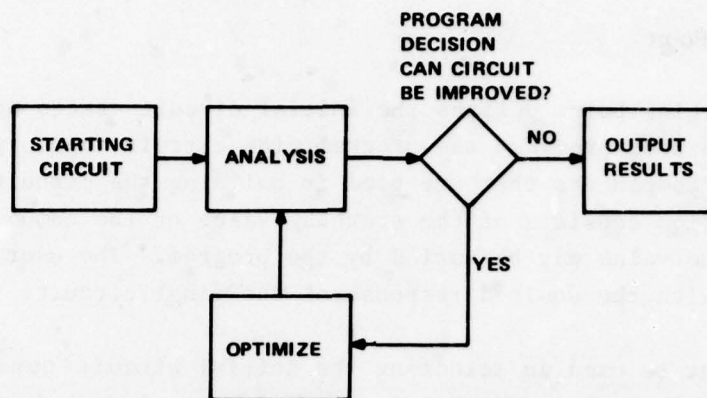


Figure 8.2-1. Simplified Flow Chart of MAGNET

8.1.3 OUTPUT Data

The outputs produced by the program are:

- a. Final optimized circuit file which may be input to other programs.
- b. Composite S parameters for the entire network.
- c. Final circuit description including the analysis and optimization results.

8.2 FUNCTIONAL DESCRIPTION

The MAGNET program takes a circuit from a starting point, analyzes it, compares the desired response with the actual and calculates the difference between them. The optimization routine then varies the circuit elements in an attempt to reduce or eliminate this calculated difference in order to approximate the desired network response. When specific requirements are met, the optimization routine terminates and the "improved" circuit is output. Figure 8.2-1 shows a simplified functional flow of the MAGNET program including the OPTIMIZE feedback loop.

8.2.1 Starting Point

The starting point defines the initial circuit design and the desired results. The user provides the program with circuit topology and descriptions of the components that are used in building the circuit. The component description consists of the starting value of the component and limits over which the value may be varied by the program. The user also provides the program with the desired response of the final circuit.

Care must be used in selecting the initial circuit topology and setting the starting values of the circuit. If the circuit is too complex, it may be difficult to layout, assemble, and test; if the circuit is poorly configured, it may be very critical of device variations. The program operates to minimize the error function (the difference between the actual and the desired response) and as a result it will progress from the starting point to the nearest minimum. Therefore, the final result is a direct function of the selected starting point.

8.2.2 Network Analysis

The network analysis routine is the heart of MAGNET. It determines which circuit topologies and what circuit element types are allowed. The routine performs matrix manipulation on each of the elements to allow their combination to be put into an overall circuit.

The program allows pairs of linear two-port elements to be connected together in almost any fashion to become a new two-port element. The elements used to make the composite element (which may have been made up of two-port elements) are not lost and may be used again. Examples of element types and interconnections allowed are given in Figures 8.2-2 and 8.2-3.

During all matrix manipulations, a noise matrix is carried along and updated for each circuit element. Each noise matrix contains all of the information to completely characterize the noise performance of its corresponding element. Through the use of this technique, MAGNET is capable of accurately computing the noise figure of any network constructed within its format. This includes all of the examples shown in Figures 8.2-2 and 8.2-3.

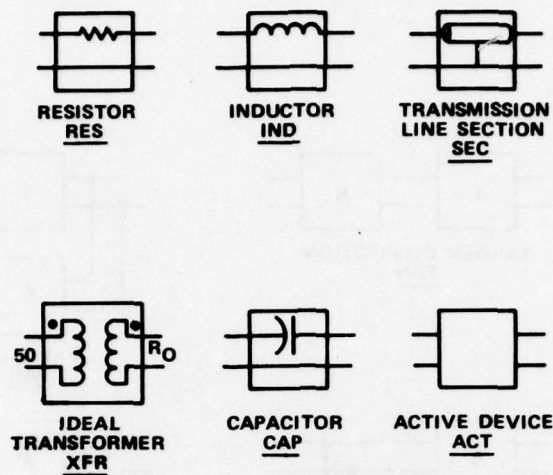


Figure 8.2-2. Element Type

There are networks that cannot be modeled by MAGNET due to intermeshed feedback loops. Figure 8.2-4 shows a circuit with such feedback loops which cannot be built from two ports taken in pairs.

8.2.3 Optimization Routine

This algorithm sets up a rectangular neighborhood about the given values, then examines the boundary of the neighborhood in an orderly but pseudorandom way to find a direction that will lower the error function. When this direction is found, a vector is created to represent the direction. The program values are then set along the vector in an attempt to find a minimum error function value. The minimum is used as a new starting point to estimate a new direction, and the process is repeated. If no point on the boundary can be found that lowers the error function, a smaller neighborhood is set up, and the process is repeated. Each cycle is defined as one iteration. The program continues until the number of iterations, set by the user, is exceeded, or the neighborhood is less than 0.01 percent of the given circuit values.

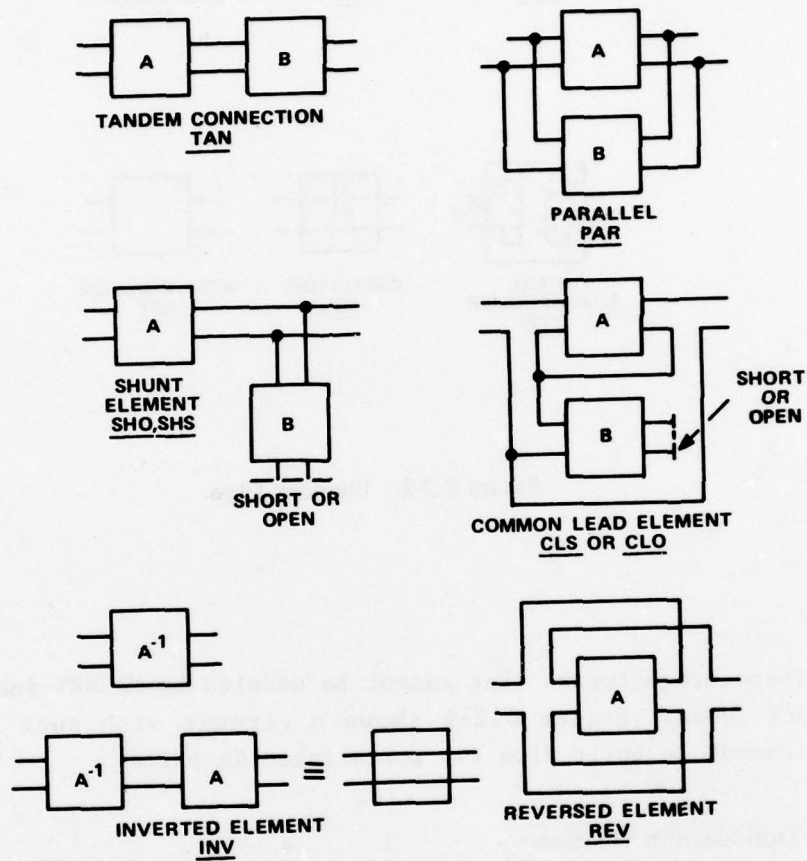


Figure 8.2-3. Element Combination

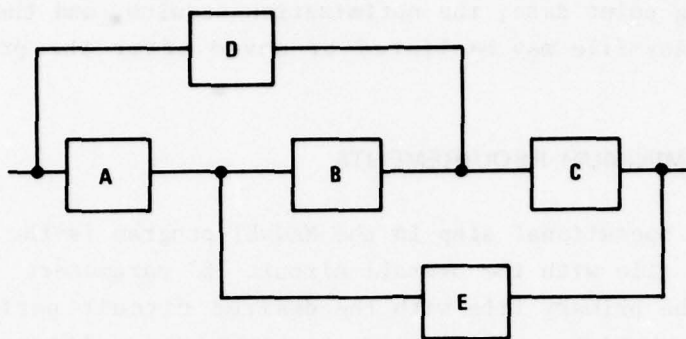


Figure 8.2.4. Example Network Not Accepted by MAGNET

8.2.4 Output Routine

The output routine supplies the user with an analysis of the starting circuit and of the final optimized circuit erected by MAGNET. The analysis includes:

- a. Power gain,
- b. Input VSWR,
- c. Output VSWR,
- d. Phase linearity,
- e. Noise figure.

The MAGNET program provides a printout of circuit topology and of the values arrived at for the elements of the optimized circuit. The program also prints the dimensions for microstrip implementation of the various line sections used. If other mediums are used (strip-line or lumped elements design) the dimensions are not output. It is important that the user be aware that these dimensions do not include compensation for such effects as line thickness or end effects.

The MAGNET program also creates three files that contain data in a format that can be used in this and other programs. One file (FOR23.DAT) contains the input data modified by the optimization routine which may be used as a starting point for further optimization used to find stability

and gain. The third file (printed on Unit 6) contains the program analysis of the starting point data, the optimization results, and the final circuit description. Any file may be listed or saved after the program is complete.

8.3 PROGRAM MINIMUM REQUIREMENTS

The first operational step in the MAGNET program is the creation of a secondary data file with the overall circuit "S" parameter. The next step is to create the primary file with the desired circuit performance data, component description, and element combinations. After the files are created and saved the user may begin the main sequence of instructions.

8.3.1 Elements Coding

There are two types of element coding statements; element creation and element combination. The element creation statements define the element type. The element combination statements define the electrical connection for the elements. These statements must be input to the program in the order of element numbers which are defined on the design from left to right.

8.3.2 Program Control Coding

The program control statements initiate the MAGNET program and open the input data files which contain the network description. The optimized circuit output is written on Unit 23 under the file name FOR23.DAT. This file may be re-input to the program on a second run for further optimization.

8.3.3 Program Outputs

The program creates two new files containing data in a format that can be used in this and other programs. These files are normally output as:

- a. FOR23.DAT (Final Optimized Circuit).
- b. FOR24.DAT (Composite Circuit Parameters).

In addition, the program outputs the final circuit description on Unit 6.

8.3.4 Final Optimized Circuit

The final optimized circuit file contains optimized input data which may be re-input to the MAGNET program for further optimization. This file contains the combined primary network description, desired performance data, and the secondary S parameter data. The remaining data in the file are the input element and combination statements and the optimized performance ranges for each element.

8.3.5 Composite S Parameters

The composite S parameters file lists the overall amplifier S parameters. That is the combined S parameters for both active devices in the circuit design.

8.3.6 Final Circuit Description

The final circuit description file printed on Unit 6 contains:

- a. Starting point analysis data.
- b. Optimization results of the input data.
- c. Final circuit description containing the required information for circuit layout.

8.3.6.1 Starting Point Analysis

The program analyzes the input data and prints the actual gain, VSWR, phase deviation, and noise figure data for the circuit as described on the input data. The printout lists the initial error function, the initial step size (DELTA) and frequencies specified on the input data. In addition, for each frequency specified the desired gain, as coded in the input data, and the actual gain at the starting point is listed. Also listed is the input and output VSWR, phase deviation, and noise figure for each frequency specified.

8.3.6.2 Optimized Results

After the program analysis function is complete, the program performs optimization for the number of iterations specified and prints the error function after each iteration.

At the completion of the optimization routine the program prints the optimized results. These results again list each frequency specified and the desired gain at each frequency. The actual gain, input and output VSWR, phase deviation and noise figure are then listed for the optimized circuit.

8.3.6.3 Final Circuit

The final circuit description lists each element type, the characteristic impedance of the element, and its dimensions. Width is given in mils, length in degrees and mils. Also listed are all element combination statements used in the design.

8.4 USAGE INFORMATION

As basic input the user provides the MAGNET program with a circuit topology and descriptions consist of the starting values of the components and the limits over which the value may be varied by the program. Also provided are the desired responses for the final circuit.

8.4.1 Active Device S Parameters

The active device S parameter (scattering parameters) data may be applied to the program as part of the primary file or as a secondary data file.

8.4.2 Program Parameters

The program parameters are specified on the first two data entries in the primary file.

8.4.3 Frequency Response

The frequency response inputs set the desired levels at which the network is analyzed and optimized.

8.4.4 Element Creation Statements

The element creation statements define the type of elements contained in the circuit design.

8.4.4.1 Active Device

Active device creation statements are followed by parameter statements which may be in the primary file or in a secondary file called by the primary file.

8.4.4.2 Non-Active Elements

The program optimizes the element parameters for all variable elements and has no affect on fixed elements.

8.4.4.3 Repeat

The repeat function is used as an aid to tell the program that the element (N) is the same as a previously described element (M).

8.4.5 Element Combination Statements

Element combination statements combine two discrete elements, which have been previously defined to make a new element.

8.4.6 INV Statement

The INV statement is used primarily for de-imbedding.

8.4.7 Network Definition Termination

The network termination statement tells the program that the circuit description is complete. This statement ends the circuit description input and enables the program to run.

8.4.8 Run MAGNET

The run MAGNET command is the program select command that accesses the MAGNET program and allows network optimization.

8.4.9 File Assignment

The file assignment commands are used to assign file names to the system unit numbers which are assigned or accessed by the MAGNET program. This statement must be used to assign the appropriate primary file name to unit 21. The program automatically writes three output files on units 23, 24, and 6. If these files are to be saved for future use, file names should be assigned for each to prevent a second MAGNET run from destroying the existing files.

8.5 OUTPUTS

The program output routine produces two files containing the final optimized circuit description, and the composite S parameters of the active devices. A listing of these files may be obtained by using the operating systems TYPE or PRINT commands.

8.6 RESTRICTIONS ON MAGNET

The following list contains restrictions on the use of the MAGNET program.

- a. Element numbers must be in order. MAGNET does not check for this. (When operating on element #n the program gets the nth element in the list, ignoring the element number.)
- b. In all cases j and k in Table 8.6-1 must be less than N.
- c. In ACT, when two port parameters are listed in the primary file, there must be only one line of data for each frequency. This also applies to the noise figure data. No interpolation is performed. This does not apply if "ALL" or 'AUTO' is used.
- d. When a secondary file is used, MAGNET will interpolate but not extrapolate.
- e. The S matrix and "ABCD" matrix must exist for all 'ACT' commands. If 'AUTO' is used the Y matrix must also exist.
- f. In all cases the ABCD matrix must exist. When 'PAR' is called the Y matrix of each element must also exist. The Z matrix must exist when 'CLO' or 'CLS' is called.
- g. In the noise figure data R_n is normalized to 50-ohms.
- h. A maximum number of 100 elements, 10 active devices, and 20 frequencies may be used.

TABLE 8.6-1 Element Combination Statements

ELEM TYPE	STATEMENT	SYMBOLIC CONNECTION	COMMENTS
'TAN'	$L_n \sqcup N \sqcup \text{'TAN'} \sqcup i \sqcup k$		Element N is equal to series connection of j and k.
'PAR'	$L_n \sqcup N \sqcup \text{'PAR'} \sqcup i \sqcup k$		Element N is equal to parallel connection of j and k.
'REV'	$L_n \sqcup N \sqcup \text{'REV'} \sqcup i \sqcup \phi$		Element N input and output ports are reversed from element j.

TABLE 8.6-1 Element Combination Standards (Continued)

ELEM TYPE	STATEMENT	SYMBOLIC CONNECTION	COMMENTS
'SHS'	$L_n \sqcup N \sqcup \text{'SHS'} \sqcup i \sqcup k$		Element N is equal to elements j and k with the shunt, k, on the output port and the shunt shorted.
'SHO'	$L_n \sqcup N \sqcup \text{'SHO'} \sqcup i \sqcup k$		Element N is the same as 'SHS' except the shunt is not shorted.
'CLS'	$L_n \sqcup N \sqcup \text{'CLS'} \sqcup i \sqcup k$		Element N equals common lead connection between elements j and k with the output of k shorted.

TABLE 8.6-1 Element Combination Standards (Continued)

ELEM TYPE	STATEMENT	SYMBOLIC CONNECTION	COMMENTS
'CLO'	$L_n \sqcup N \sqcup \text{'CLO'} \sqcup i \sqcup k$		<p>Element N same as 'CLS' except output of k is open.</p>
'INV'	$L_n \sqcup N \sqcup \text{'INV'} \sqcup i \sqcup \phi$		<p>Element N is equal to the inverse of j $ABCD$ of $N = ABCD^{-1}$ of j.</p>

- i. (Number of variable ELE) times (Number at Freq.) times (Number of iterations) should be less than 1000 to keep a reasonable running time.
- j. MAGNET assumes a 50-ohm system for all two-port parameter representations.

8.7 ABCD MATRIX

Most of the communication between MAGNET and the design is with S parameters. Internally, however, the program represents every element in its ABCD form. This is done to minimize computation in converting from one matrix form to another.

SECTION 9

A STRUCTURAL ANALYSIS PROGRAM--SAP IV

The development of an effective computer program for structural analysis requires a knowledge of three scientific disciplines--structural mechanics, numerical analysis and computer application. The development of accurate and efficient structural elements requires a modern background in structural mechanics. The efficiency of a program depends largely on the numerical techniques employed and on their effective computer implementation. With regard to programming techniques, an optimum allocation of high- and low-speed storage is necessary.

A most important aspect of a general purpose computer program is, however, the ease with which it can be modified, extended and updated; otherwise, it may very well be that the program is obsolete within a few years after completion. This is because new structural elements are developed, better numerical procedures are available, or new computer equipment which requires new coding techniques is produced.

The structural analysis program (SAP) was designed to be modified and extended by the user. Additional options and new elements may easily be added. The program has the capacity to analyze very large three-dimensional systems; however, there is no loss in efficiency in the solution of smaller problems. Also, from the complete program, smaller special purpose programs can easily be assembled by simply using only those sub-routines which are actually needed in the execution. This makes the program particularly usable on small size computers.

The current program version SAP IV for the static and dynamic analysis of linear structural systems is the result of several years' research and development experience. The program has proven to be a very flexible and efficient analysis tool. The program is coded in FORTRAN IV and operates without modifications on the CDC 6400, 6600 and 7600 computers. Figure 9.0-1 depicts the functional relation between the SAP IV Program and the remaining programs.

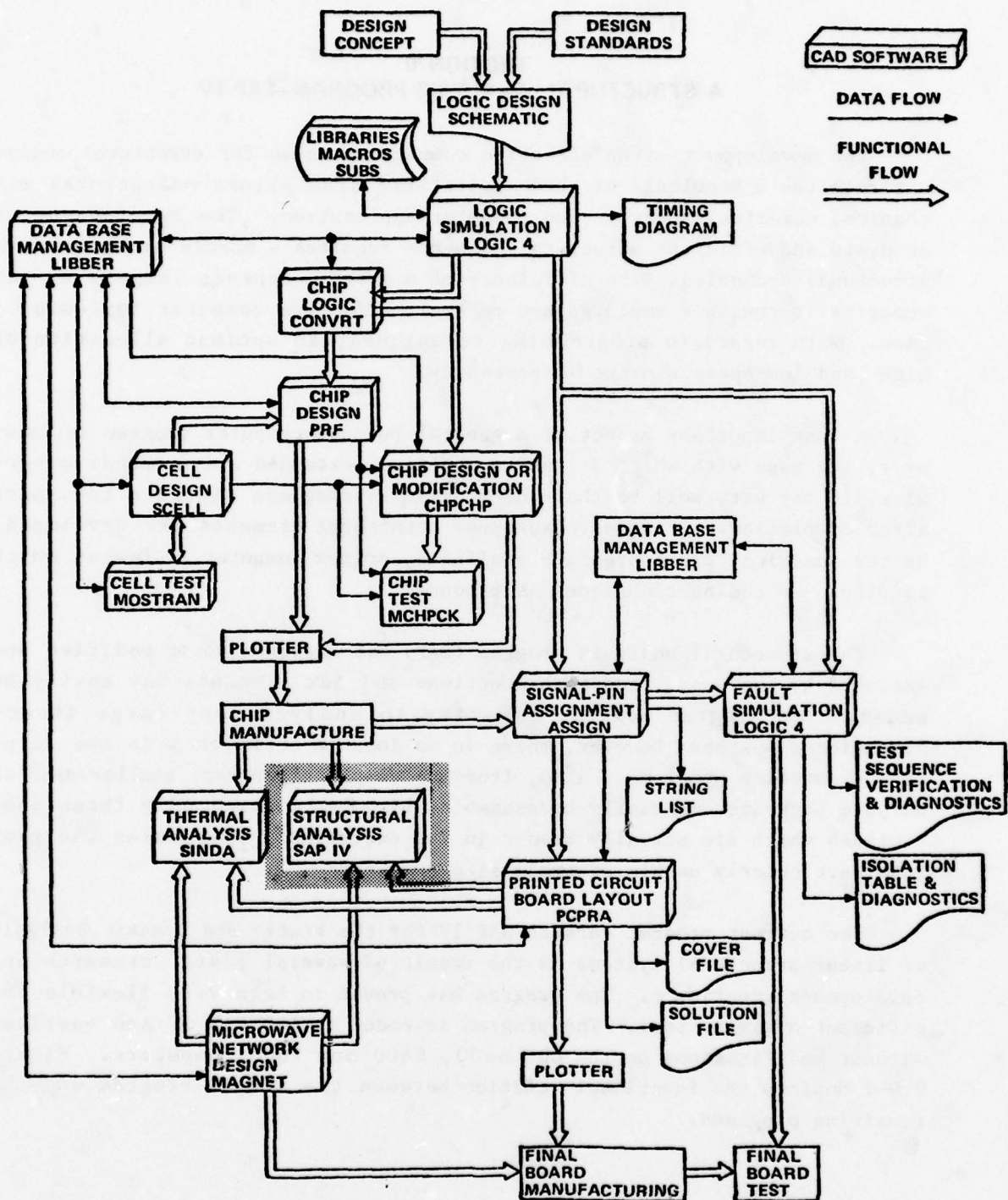


Figure 9.0-1. CAD Functional Block Diagram Emphasizing the SAP IV Program

9.1 GENERAL DESCRIPTION

The structural systems to be analyzed may be composed of combinations of a number of different structural elements. The program presently contains the following element types:

1. three-dimensional truss element,
2. three-dimensional beam element,
3. plane stress and plane strain element,
4. two-dimensional axisymmetric solid,
5. three-dimensional solid,
6. variable-number nodes thick shell and three-dimensional element,
7. thin plate or thin shell element,
8. boundary element,
9. pipe element (tangent and bend).

These structural elements can be used in a static or dynamic analysis. The capacity of the program depends mainly on the total number of nodal points in the system, the number of eigenvalues needed in the dynamic analysis and the computer used. There is practically no restriction on the number of elements used, the number of load cases or the order and bandwidth of the stiffness matrix. Each nodal point in the system can have from zero to six displacement degrees of freedom. The element stiffness and mass matrices are assembled in condensed form; therefore, the program is equally efficient in the analysis of one-, two- or three-dimensional systems.

The formation of the structure matrices is carried out in the same way in a static or dynamic analysis. The static analysis is continued by solving the equations of equilibrium followed by the computation of element stresses. In a dynamic analysis the choice is between:

1. frequency calculations only,
2. frequency calculations followed by response history analysis,
3. frequency calculations followed by response spectrum analysis,
4. response history analysis by direct integration.

To obtain the frequencies and vibration mode shapes, solution routines are used which calculate the required eigenvalues and eigenvectors directly without a transformation of the structure stiffness matrix and mass matrix to a reduced form. In the direct integration an unconditionally stable

integration scheme is used, which also operates on the original structure stiffness matrix and mass matrix. This way the program operation and necessary input data for a dynamic analysis is a simple addition to what is needed for a static analysis.

9.2 PROGRAM ORGANIZATION

The calculation of the structure stiffness matrix and mass matrix is accomplished in three distinct phases:

1. The nodal point input data is read and generated by the program. In this phase the equation numbers for the active degrees of freedom at each nodal point are established.
2. The element stiffness and mass matrices are calculated together with their connection arrays; the arrays are stored in sequence on tape (or other low-speed storage).
3. The structure stiffness matrix and mass matrix are formed by addition of the element matrices and stored in block form on tape.

It should be noted that three basic steps can be applied to any of the element types and are the same for either a static or dynamic analysis.

9.2.1 Nodal Point Input Data and Degrees of Freedom

The capacity of the program is controlled by the number of nodal points of the structural system. For each nodal point six boundary condition codes (stored in the array ID), three coordinates (stored in the arrays X, Y, Z) and the nodal point temperatures (stored in the array T) are required (generation capability is provided). All nodal point data is retained in high-speed storage during the formation of the element stiffness and mass matrices. Since the required high-speed storage for the element subroutines is relatively small, the minimum required storage for a given problem is a little larger than ten times the number of nodal points in the system.

It is noted that the user should allow only those degrees of freedom which are compatible with the elements connected to a nodal point. The program always deals with six possible degrees of freedom at each nodal point, and all non-active degrees of freedom should be deleted so as to

decrease the order of the structure matrices. Specifically, a "1" in the ID array denotes that no equation shall be associated with the degree of freedom, whereas a "0" indicates that this is an active degree of freedom.

9.2.2 Element Mass and Stiffness Calculations

With the coordinates of all nodal points known and the equation numbers of the degrees of freedom established, the stiffness, mass and stress-displacement transformation matrices for each structural element in the system are calculated. As pointed out earlier, little additional high-speed storage is required for this phase since these matrices are formed and placed on tape storage at the same time as the element properties are read. Together with the matrices pertaining to the element, the corresponding element connection array, vector LM, is written on tape. The vector LM is established from the ID matrix and the specified structure nodal points pertaining to the element.

The element matrices are calculated in groups, i.e., all elements being in one group together, thus calling the corresponding element subroutine only once for each element group. After all element matrices have been established, the ID and X, Y, Z arrays are not needed any more, and the corresponding storage area is used for the formation of the structure matrices and later for the solution of the equations of equilibrium.

9.2.3 Formation of Structure Stiffness and Mass

The stiffness matrix and mass matrix of the structure are formed in blocks. The number of equations per block depends on the available high-speed storage and is calculated in the program as indicated in Figure 9.2.3-1. It is noted that on reasonable size computers very large systems can be analyzed for static and dynamic response. With the number of equations per block known, the stiffness and mass matrix are assembled two blocks at a time by direct addition of the element matrices. In this process it is necessary to pass through the element matrices which are stored on tape. In order to minimize tape reading, in each pass, element matrices which pertain to the next several blocks are written on another tape. This way the tape reading necessary for the formation of these blocks is reduced significantly.

A flow diagram of the program organization for the calculation of the structure stiffness matrix and mass matrix is shown in Figure 9.2.3-2.

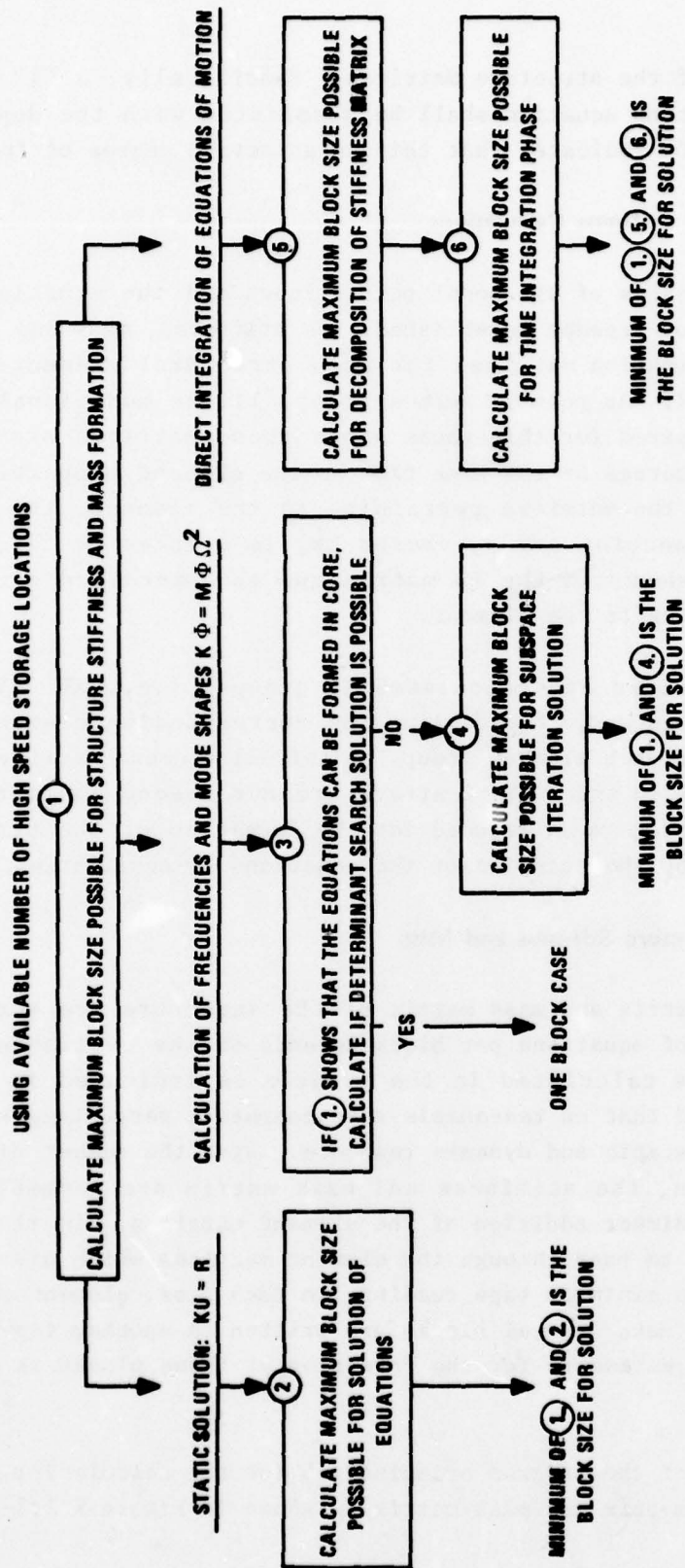


Figure 9.2.3-1. Flow Chart Showing Calculation of Number of Equations in a Block

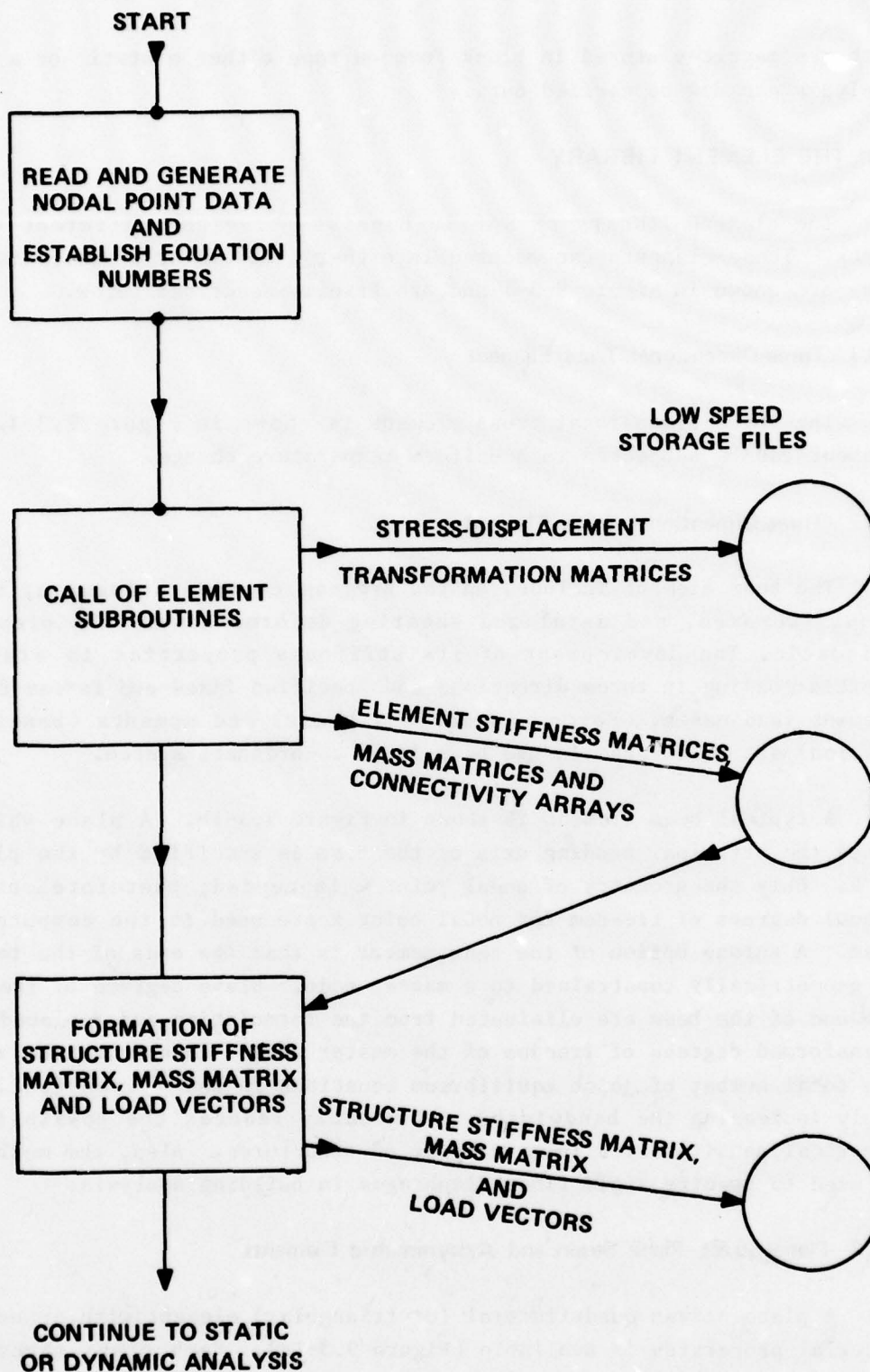


Figure 9.2.3-2. Flow Chart for Calculation of Structure Stiffness Matrix and Mass Matrix

With the matrices stored in block form on tape either a static or a dynamic analysis can now be carried out.

9.3 THE ELEMENT LIBRARY

The element library of SAP IV consists of eight different element types. These elements can be used in either a static or dynamic analysis. They are shown in Figure 9-3-1 and are briefly described below.

9.3.1 Three-Dimensional Truss Element

The three-dimensional truss element is shown in Figure 9.3-1a. The element can be subjected to a uniform temperature change.

9.3.2 Three-Dimensional Beam Element

The beam element included in the program considers torsion, bending about two axes, and axial and shearing deformations. The element is prismatic. The development of its stiffness properties is standard. Inertia loading in three directions and specified fixed-end forces form the element load cases. Forces (axial and shear) and moments (bending and torsion) are calculated in the beam local co-ordinate system.

A typical beam element is shown in Figure 9.3-1b. A plane which defines the principal bending axis of the beam is specified by the plane i, j, k . Only the geometry of nodal point k is needed; therefore, no additional degrees of freedom for nodal point k are used in the computer program. A unique option of the beam member is that the ends of the beam can be geometrically constrained to a master node. Slave degrees of freedom at the end of the beam are eliminated from the formulation and replaced by the transformed degrees of freedom of the master node. This technique reduces the total number of joint equilibrium equations in the system (while possibly increasing the bandwidth) and greatly reduces the possibility of numerical sensitivities in many types of structures. Also, the method can be used to specify rigid floor diaphragms in building analysis.

9.3.3 Plane Stress, Plane Strain and Axisymmetric Elements

A plane stress quadrilateral (or triangular) element with orthotropic material properties is available (Figure 9.3-1c). Each plane stress element may be of different thickness and may be located in an arbitrary plane

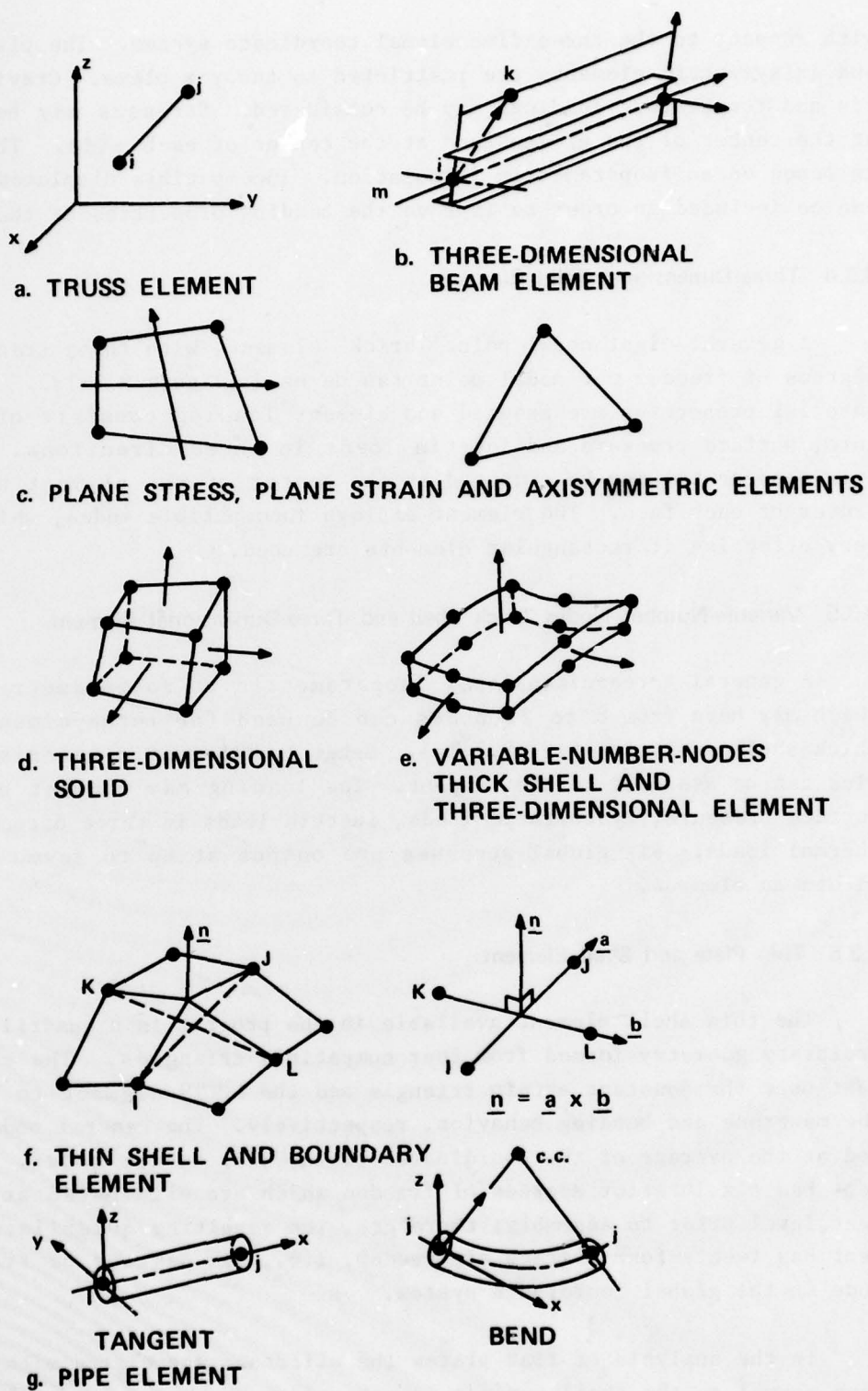


Figure 9.3-1. Element Library of SAP IV

with respect to the three-dimensional coordinate system. The plane strain and axisymmetric elements are restricted to the y-z plane. Gravity, inertia and temperature loadings may be considered. Stresses may be computed at the center of the element and at the center of each side. The element is based on an isoparametric formulation. Incompatible displacement modes can be included in order to improve the bending properties of the element.

9.3.4 Three-Dimensional Solid Element

A general eight nodal point "brick" element, with three translational degrees of freedom per nodal point can be used (Figure 9.3-1d). Isotropic material properties are assumed and element loading consists of temperature, surface pressure and inertia loads in three directions. Stresses (six components) may be computed at the center of the element and at the center of each face. The element employs incompatible modes, which can be very effective if rectangular elements are used.

9.3.5 Variable-Number Nodes Thick Shell and Three-Dimensional Element

A general three-dimensional isoparametric or subparametric element which may have from 8 to 21 nodes can be used for three-dimensional or thick shell analysis (Fig. 9.3-1e). General orthotropic material properties can be assigned to the element. The loading may consist of applied surface pressure, hydrostatic loads, inertia loads in three directions, and thermal loads. Six global stresses are output at up to seven locations within an element.

9.3.6 Thin Plate and Shell Element

The thin shell element available in the program is a quadrilateral of arbitrary geometry formed from four compatible triangles. The shell element uses the constant strain triangle and the LCCT9 element to represent the membrane and bending behavior, respectively. The central node is located at the average of the coordinates of the four corner nodes. The element has six interior degrees of freedom which are eliminated at the element level prior to assembly; therefore, the resulting quadrilateral element has twenty-four degrees of freedom, i.e., six degrees of freedom per node in the global coordinate system.

In the analysis of flat plates the stiffness associated with the rotation normal to the shell surface is not defined; therefore, the rotation

normal degree of freedom must not be included in the analysis. For curved shells, the normal rotation need be included as an extra degree of freedom. In case the curvature is very small, the degree of freedom should be restrained by the addition of a "boundary element" with a small normal rotational stiffness, say of less or about 10 percent of the element bending stiffness.

9.3.7 Boundary Element

The boundary element, shown in Figure 9.3-1f, can be used for the following:

1. in the idealization of an external elastic support at a node;
2. in the idealization of an inclined roller support;
3. to specify a displacement, or
4. to eliminate the numerical difficulty associated with the "sixth" degree of freedom in the analysis of nearly flat shells.

The element is one-dimensional with an axial or torsional stiffness. The element stiffness coefficients are added directly to the total stiffness matrix.

9.3.8 Pipe Element

The pipe element (Figure 9.3-1g) can represent a straight segment (tangent) or a circularly curved segment (bend); both elements require a uniform section and uniform material properties. Elements can be directed arbitrarily in space. The member stiffness matrices account for bending, torsional, axial and shearing deformations. In addition, the effect of internal pressure on the stiffness of curved pipe elements is considered.

The types of structure loads contributed by the pipe elements include gravity loading in the global directions and loads due to thermal distortions and deformations induced by internal pressure. Forces and moments acting at the member ends (i, j) and at the center of each bend are calculated in coordinate systems aligned with the member's cross section.

The pipe element stiffness matrix is formed by first evaluating the flexibility matrix corresponding to the six degrees of freedom at end j. With the corresponding stiffness matrix, the equilibrium transformations are used to form the complete element stiffness matrix. Distortions due to

element loads are premultiplied by the stiffness matrix to compute restrained nodal forces due to thermal, pressure or gravity loads.

9.4 THE EQUILIBRIUM EQUATIONS FOR COMPLEX STRUCTURAL SYSTEMS

9.4.1 Element to Structure Matrices

The nodal point equilibrium equations for a linear system of structural elements can be derived by several different approaches. All methods yield a set of linear equations of the form

$$M\ddot{u} + C\dot{u} + Ku = R, \quad (1)$$

where M is the mass matrix, C is the damping matrix and K is the stiffness matrix of the element assemblage. The vectors \ddot{u} , \dot{u} , u and R are the nodal displacements, velocities, accelerations and generalized loads, respectively. The structure matrices are formed by direct addition of the element matrices. For example,

$$K = \sum K_m, \quad (2)$$

where K_m is the stiffness matrix of the m th element. Although K_m is formally of the same order as K , only those terms in K_m which pertain to the element degrees of freedom are nonzero. The addition of the element matrices can therefore be performed by using the element matrices in compact form, together with identification arrays which relate element to structure degrees of freedom. The algorithm used in the program is described in Section 9.2.3.

In the program, the structure stiffness matrix and a diagonal mass matrix are assembled. Therefore, a lumped mass analysis is assumed where the structure mass is the sum of the individual element mass matrices plus additional concentrated masses which are specified at selected degrees of freedom. The damping is assumed to be proportional and is specified in the form of a modal damping factor.

9.4.2 Boundary Conditions

If a displacement component is zero, the corresponding equation is not retained in the structure equilibrium equations, Eq. (1), and the corresponding element stiffness and mass terms are disregarded. If a nonzero

displacement is to be specified at a degree of freedom i , say $U_i = x$, the equation

$$ku_i = kx \quad (3)$$

is added into Eq. (1), where $k \gg k_{ii}$. Therefore, the solution of Eq. (1) must give $u_i = x$. Physically, this can be interpreted as adding at the degree of freedom i a spring of large stiffness k and specifying a load which, because of the relatively flexible structure at this degree of freedom, produces the required displacement x .

9.5 STATIC ANALYSIS

A static analysis involves the solution of the equilibrium equations

$$Ku = R \quad (4)$$

followed by the calculation of element stresses.

9.5.1 Solution of Equilibrium Equations

The load vectors R have been assembled at the same time as the structure stiffness matrix and mass matrix are formed. The solution of the equations is obtained using the large capacity linear equation solver SESOL. This subroutine uses Gauss elimination on the positive-definite symmetrical system of equations. The algorithm performs a minimum number of operations; i.e., there are no operations with zero elements. In the program, the L^TDL decomposition of K is used; hence, Eq. (4) can be written as

$$L^T v = R, \quad (5)$$

and

$$v = DLu, \quad (6)$$

where the solution for v in Eq. (5) is obtained by a reduction of the load vectors; the displacement vectors u are then calculated by a back-substitution.

In the solution, the load vectors are reduced at the same time as K is decomposed. In all operations it is necessary to have at any one time the required matrix elements in high-speed storage. In the reduction, two

blocks are in high-speed storage (as was also the case in the formation of the stiffness matrix and mass matrix); i.e., the "leading" block, which finally stores the elements of L and D, and in succession those blocks which are affected by the decomposition of the "leading" block. Table 9.5.1-1 gives some typical solution times.

9.5.2 Evaluation of Element Stresses

After the nodal point displacements have been evaluated, sequentially the element stress-displacement matrices are read from low speed storage and the element stresses are calculated.

9.6 CALCULATION OF FREQUENCIES AND MODE SHAPES

The dynamic analysis of a structural system using mode superposition requires as the first step the solution of the generalized eigenvalue problem

$$K\phi = \omega^2 M\phi \quad (7)$$

where ϕ and ω are free vibration frequency and mode shape, respectively. As was described in Section 9.2.3, the program stores the stiffness and mass matrix in blocks on tape. The mass matrix is diagonal with partly zero diagonal elements. The program assumes that only the lowest p eigenvalues and corresponding eigenvectors are needed. The solution of Eq. (7) can therefore be written as

$$K\phi = M\phi\Omega^2 \quad (8)$$

where Ω^2 is a diagonal matrix with the p smallest eigenvalues; i.e., $\Omega^2 = \text{diag}(\omega_1^2)$, and ϕ stores the corresponding M -orthonormalized eigenvectors $\phi_1, \phi_2, \dots, \phi_p$. Two different solution procedures are used in the program, a determinant search technique or a subspace iteration solution. The determinant search solution is carried out when the stiffness matrix can be contained in high-speed storage in one block. Therefore, for systems of larger order and bandwidth, the subspace iteration method is used. Both solution techniques solve the generalized eigenvalue problem directly without a transformation to the standard form.

TABLE 9.5.1-1 Solution of Equations Using Sesol

NUMBER OF EQUATIONS	HALF BANDWIDTH	CENTRAL PROCESSOR SEC	COMPUTER USED
8036	544	1786 [†]	CDC 6600
2696	488	1260	CDC 6600
4214	205	31	CDC 7600

[†]The inner DO - loop in the factorization of the stiffness matrix has been coded in machine language for this solution.

9.6.1 The Determinant Search Solution

The determinant search technique is best suited for the analysis of large systems in which K and M have small bandwidths. Basically, the solution algorithm combines triangular factorization and vector inverse iteration in an optimum manner to calculate the required eigenvalues and eigenvectors; these are obtained in sequence starting from the least dominant eigenpair ω_1^2, ϕ_1 . An efficient, accelerated secant iteration procedure which operates on the characteristic polynomial

$$p(\omega^2) = \det(K - \omega^2 M) \quad (9)$$

is used to obtain a shift near the next unknown eigenvalue. The eigenvalue separation theorem is used in this iteration. Each determinant evaluation requires a triangular factorization of the matrix $K - \omega^2 M$. Once a shift near the unknown eigenvalue has been obtained, inverse iteration is used to calculate the eigenvector. The eigenvalue is then obtained by adding the Rayleigh quotient correction to the shift value. Table 9.6.1-1 shows typical solution times.

9.6.2 The Subspace Iteration Solution

When the system is too large to be completely contained in high speed storage, i.e., more blocks than one are used, the subspace iteration solution is carried out. The iteration can be interpreted as a repeated application of a method in which the computed eigenvectors from one step are used as the trial basis vectors for the next iteration until convergence to the required p eigenvalues and eigenvectors is obtained.

The solution is carried out by iterating simultaneously with q linearly independent vectors, where $q > p$. In the k th iteration the vectors span the q -dimensional subspace and "best" eigenvalue and eigenvector approximations are calculated; i.e., when the vectors span the p -dimensional least dominant subspace, the required eigenvalues and eigenvectors are obtained.

Let V_0 store the starting vectors, then the k th iteration is described as follows:

Solve for vectors \bar{V}_k which span ξ_k

$$K\bar{V}_k = M\bar{V}_{k-1} \quad (10)$$

TABLE 9.6.1-1 Calculation of Frequencies and Mode Shapes Using Determinant Search Method

SYSTEM	SYSTEM ORDER n	MAXIMUM HALF BAND WIDTH	NUMBER OF REQ'D. FREON. & MODE SHAPES p	COMPUTER USED	CENTRAL PROCESSOR SEC
PLANE FRAME	297	30	3	CDC 6400	40
PIPING SYSTEM	566	12	7	CDC 6600	11
BUILDING	340	32	7	CDC 6600	20
CONTAINER	265	65	40	CDC 7600	58

Calculate the projections of K and M onto ξ_k (i.e., the generalized stiffness matrix and mass matrix corresponding to ξ_k)

$$K_k = \bar{V}_k^T K \bar{V}_k \quad (11)$$

$$M_k = \bar{V}_k^T M \bar{V}_k \quad (12)$$

Solve for the eigensystem of K_k and M_k

$$K_k Q_k = M_k Q_k \Omega_k^2 \quad (13)$$

and calculate the k'th improved approximation to the eigenvectors

$$V_k = \bar{V}_k Q_k \quad (14)$$

Provided that the starting subspace is not orthogonal to any of the required eigenvectors, the iteration converges to the desired result; i.e., $\Omega_k^2 \rightarrow \Omega^2$ and $V_k \rightarrow \Phi$ as $k \rightarrow \infty$.

The number of vectors q used in the iteration is taken greater than the desired number of eigenvectors in order to accelerate the convergence of the process. The number of iterations required to achieve satisfactory convergence depends, of course, on the quality of the starting vectors V_0 . Unless requested otherwise (see Section 9.6.3), the program generates q starting vectors where $q = \min(2p, p + 8)$, which has proven to be effective in general applications. At convergence, a Sturm sequence check can be requested to verify that the lowest p eigenvalues have been found.

Table 9.6.2-1 lists a few typical solution times using the program generated starting vectors.

9.6.3 Dynamic Optimization

The solution of the eigenvalue problem may be required when a good estimate of the required eigensystem is already known, such as in dynamic optimization. In this case the subspace iteration method is ideally suited for solution. The number of iteration vectors q and the vectors V_0 , together with the maximum number of iterations, can in this case be specified by the user. Also, in case the number of eigenvalues and vectors required is increased, the already calculated eigenvectors can be specified as part of the starting iteration vectors in order to accelerate convergence.

TABLE 9.6.2-1 Calculation of Frequencies and Mode Shapes Using Subspace Interaction Method

SYSTEM	SYSTEM ORDER n	MAXIMUM HALF BAND WIDTH	NUMBER OF REQ'D. FREQN. & MODE SHAPES p	COMPUTER USED	CENTRAL PROCESSOR SEC
PLANE FRAME	297	30	3	CDC 6400	25
PIPING SYSTEM	566	12	28	CDC 6600	142
BLDG. WITH FOUNDATION	1174	138	45	CDC 6600	890
3-DIM BLDG. FRAME	468	156	4	CDC 6400	160

9.7 DYNAMIC ANALYSES

In dynamic response analysis the solution of the equations

$$M\ddot{u} + C\dot{u} + Ku = R(t) \quad (15)$$

is required, where $R(t)$ can be a vector of arbitrary time varying loads, or of effective loads which result from ground motion. Specifically, in the case of ground motion, if it is assumed that the structure is uniformly subjected to the ground acceleration \ddot{u}_g , the equilibrium equations considered are

$$M\ddot{u}_r + C\dot{u}_r + Ku_r = -M\ddot{u}_g \quad (16)$$

where u_r is the relative displacement of the structure with respect to the ground; i.e., $u_r = u - u_g$.

The program can carry out a history analysis for solution of Eqs. (15) or (16), or a response spectrum analysis for solution of Eq. (16). The history analysis can be carried out using mode superposition or direct integration. The response spectrum analysis necessitates, of course, first the solution of the required eigensystem.

9.7.1 Response History Analysis by Mode Superposition

In the mode superposition analysis, it is assumed that the structural response can be described adequately by the p lowest vibration modes, where $p \ll n$. Using the transformation $u = \phi X$, where the columns in ϕ are the p M-orthonormalized eigenvectors, Eq. (15) can be written as

$$\ddot{X} + \Delta\dot{X} + \Omega^2 X = \phi^T R \quad (17)$$

where

$$\Delta = \text{diag}(2\omega_1 \xi_1); \quad \Omega^2 = \text{diag}(\omega_1^2) \quad (18)$$

In Eq. (18) it is assumed that the damping matrix C satisfies the modal orthogonality condition

$$\phi_i^T C \phi_j = 0 \quad (i \neq j) \quad (19)$$

Equation (17) therefore represents p uncoupled, second-order differential equations. These are solved in the program using the Wilson θ -method, which is an unconditionally stable step-by-step integration scheme. The same time step is used in the integration of all equations to simplify the calculation of stress components at preselected times.

In the case of prescribed ground motion $u_r = \phi X$ and in Eq. (17) the right-hand side is given by $-\phi^T \ddot{M} \ddot{u}_g$, where the ground acceleration is considered as the sum of the components in the x , y and z directions as described in Section 9.7.3.

9.7.2 Response History Analysis by Direct Integration

The solution of the equations of motion, Eqs. (15) and (16), can be obtained by direct integration. In the program, the Wilson θ -method is used, which is unconditionally stable. It need be noted that Rayleigh damping is assumed. This form of damping is easily taken into account in the analysis, because no storage and no multiplications for a damping matrix are required.

9.7.3 Response Spectrum Analysis

In this analysis the ground acceleration vector in Eq. (16) is written as

$$\ddot{u}_g = \ddot{u}_{gx} + \ddot{u}_{gy} + \ddot{u}_{gz} \quad (20)$$

where \ddot{u}_{gx} , \ddot{u}_{gy} and \ddot{u}_{gz} are the ground accelerations in the x , y and z directions, respectively. The equation for the response in the r th mode is therefore

$$\ddot{x}_r + 2\xi_r \omega_r \dot{x}_r + \omega_r^2 x_r = r_{rx} + r_{ry} + r_{rz} \quad (21)$$

where x_r is the r th element in X , and

$$r_{rx} = -\phi_r^T \ddot{M} \ddot{u}_{gx}; \quad r_{ry} = -\phi_r^T \ddot{M} \ddot{u}_{gy}; \quad r_{rz} = -\phi_r^T \ddot{M} \ddot{u}_{gz} \quad (22)$$

Using the definition of the spectral displacement, the maximum absolute modal displacements of the structure subjected to an acceleration into the x direction are

$$u_{rx}^{(max)} = \phi_r \left| \phi_r^T M I_x \right| S_x(\omega_r) \quad (23)$$

where $S_x(\omega_r)$ is the spectral displacement into the x direction corresponding to the frequency ω_r , and I_x is a null vector, except that those elements are equal to one which correspond to the x-translational degrees of freedom. Similarly, for the responses due to a ground acceleration into the y and z directions

$$u_{ry}^{(max)} = \phi_r^T M I_y S_y(\omega_r); u_{rz}^{(max)} = \phi_r^T M I_z S_z(\omega_r) \quad (24)$$

and the total maximum response in the rth mode is assumed to be

$$u_r^{(max)} = u_{rx}^{(max)} + u_{ry}^{(max)} + u_{rz}^{(max)}. \quad (25)$$

Program SAP IV calculates the maximum responses in each of the p lowest modes, where the spectra (displacements or accelerations) into the x, y and z directions are assumed to be proportional to each other. The total response for displacements and stress resultants is calculated as the square root of the sum of the squares of the modal maximum responses.

9.7.4 Restart Capability in Mode Superposition Analysis

The most expensive phase in mode superposition analysis is usually the calculation of frequencies and mode shapes. However, once the required eigensystem has been solved for, it can be used to analyze the structure for different loading conditions. Also, in a design process the history or spectrum analysis for the same loading can be carried out economically a few times, for example, to study the stress history in different parts of the structure.

In the program, at completion of the eigensystem solution, all variables required for a response history or response spectrum analysis, together with the frequencies and mode shapes, are written on low-speed storage. The program execution may be stopped at this stage and the information on low-speed storage be copied onto a physical tape. Later, this tape would be copied back to low-speed storage before starting a response analysis. If, after a number of response analyses using the eigensystem on the tape, it is decided that more frequencies and mode shapes need be calculated, the information on the tape can be used to reduce the cost of the new eigensystem solution as described in Section 9.6.3.

9.7.5 Mode Superposition Versus Direct Integration

For an effective response history analysis the user must decide appropriately whether to use mode superposition or direct integration. It should be realized that the direct integration is equivalent to a mode superposition analysis in which all the eigenvalues and vectors have been calculated and the uncoupled equations in Eq. (17) with $p = n$ are integrated with a common time step Δt . Naturally, the integration can only be accurate for those modes for which Δt is smaller than a certain fraction of the period T . Using the Wilson θ -algorithm, the integration errors effectively "filter" the high mode response, for which $\Delta t/T$ is large, out of the solution. This filtering is due to the amplitude decay observed in the numerical solution when $\Delta t/T$ is large. As an example, Figure 9.7.5-1 shows the amplitude decay for the initial value problem indicated.

The effective filtering of the high frequency response from the solution may be beneficial. Integration accuracy cannot be obtained in the response of the modes for which $\Delta t/T$ is large, and the filtering process allows one to obtain a total system solution in which the low mode response is accurately observed.

It is therefore noted that the direct integration is quite equivalent to a mode superposition analysis in which only the lowest modes of the system, but a sufficient number to take proper account of the applied loading, are considered. The exact number of modes effectively included in the analysis depends on the time step size Δt and the distribution of the periods.

The advantages of mode superposition are essentially that frequencies and mode shapes are obtained and that a variety of response history and response spectrum analyses can be carried out with relatively small additional cost. Also, if the structure is slightly changed or more eigenvalues and vectors are required, i.e., the frequency domain to be considered shall be extended, the eigensystem solved for already can be used to reduce the cost of the new eigensystem solution (see Section 9.7.4).

The direct step-by-step integration, however, is more effective when many modes need be included in the analysis and the response is required over relatively few time steps, such as in shock problems. It should be

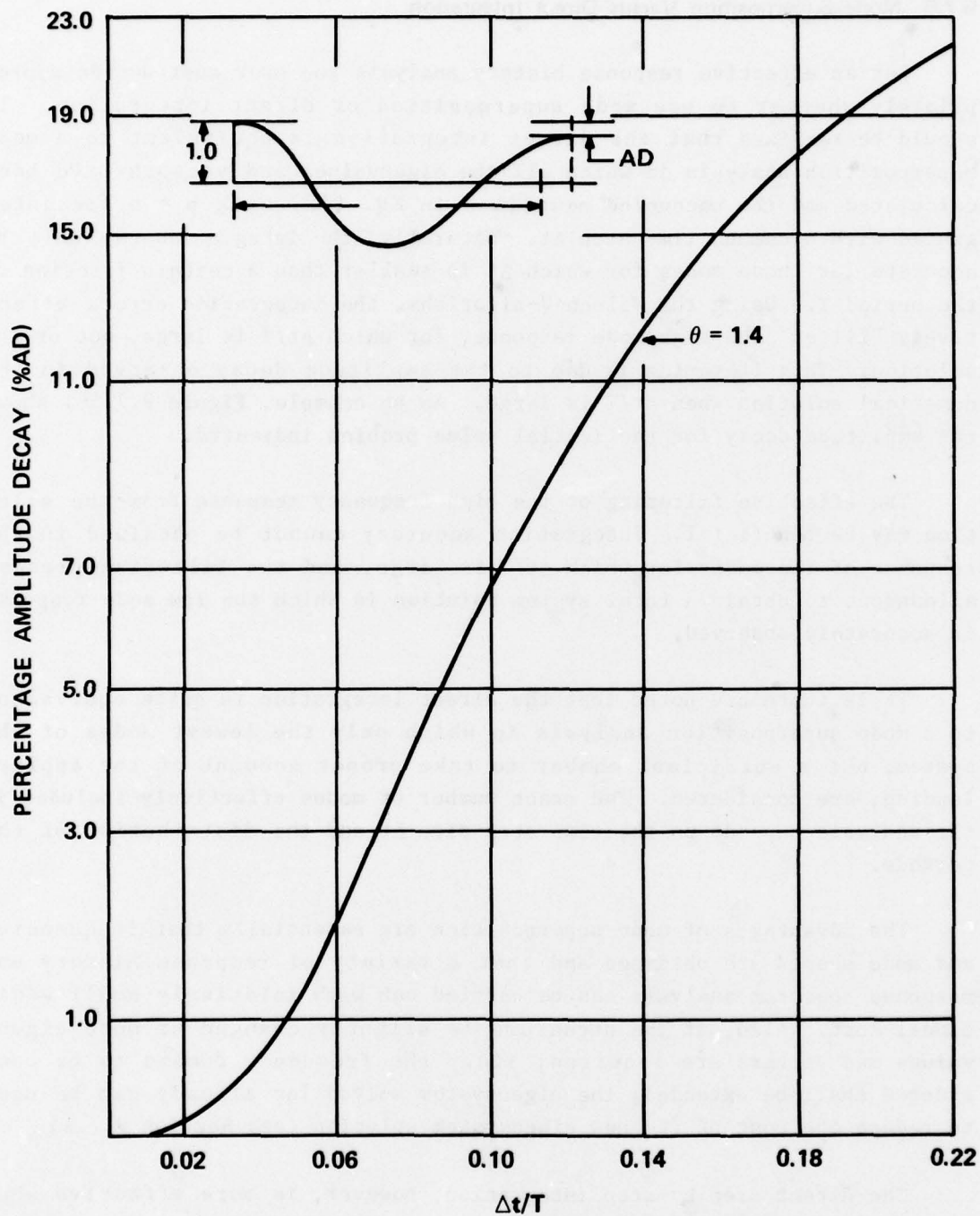


Figure 9.7.5-1. Amplitude Decay Wilson θ -Method

noted that the tape reading required in the direct integration analysis of large out-of-core systems can be costly, because in the solution for the response in each time step, the triangularized effective stiffness matrix must be taken into high speed storage.

9.8 INSTALLATION OF SAP IV ON A SYSTEM OTHER THAN A CDC COMPUTER

SAP IV is written using FORTRAN IV and has been developed on a CDC computer. The program has also been installed with relatively little effort on IBM and UNIVAC machines.

The program or parts of it can essentially be used on any reasonably sized computer. SAP IV consists of about 14,000 cards, and is organized in a standard Fortran overlay structure to reduce the required high-speed storage for program execution. The main overlay essentially consists of the main program. The secondary overlays are, respectively, the element routines, the equation solver, the eigenvalue routines, the mode superposition history analysis program, the spectrum analysis program and the direct integration routine. Using only specific overlays, efficient special purpose programs are obtained. For example, using the main overlay, plus the secondary overlays of the pipe element, the eigenvalue routines, and the response history analysis, a special purpose pipe response history analysis program by mode superposition is obtained. On the CDC 6400 of the University of California, Berkeley, the complete program with 12000₁₀ high-speed storage locations allocated for solution processing (the blank common block A has a length of 12000) requires a field length of about 114000₈ for execution.

On installation of SAP IV on other machines than the CDC series, it must be observed that arithmetic calculations should be performed using about 14 digit words. This means that, for example, on IBM and UNIVAC machines double precision need be used. The calculations to be performed in double precision by the static and dynamic analysis are the formation of structure stiffness matrices. These are the main steps in the solution of the equations of motion, namely, the solution of $Ku = R$, the solution of the generalized eigenvalue problem $K\phi = \omega^2 M\phi$, and by direct integration, the solution of the effective displacements u_t^* (see Table 4). These calculations need primarily be performed in double precision because of truncation errors occurring when too few digits are used, which can cause large errors in the solution and numerical instabilities.

With regard to the use of back-up storage, to keep the program system independent, sequential accessing is used throughout. Therefore, since no advantage is taken of efficient buffering and direct access techniques, it need be noted that the use of secondary storage can be much improved when tailored to a specific system.

SECTION 10 SINDA

10.1 INTRODUCTION

SINDA, the Systems Improved Numerical Differencing Analyzer, is a software system which possesses capabilities which make it well suited for solving lumped parameter representations of physical problems governed by diffusion-type equations. The system was originally designed as a general thermal analyzer, accepting resistor-capacitor (R-C) network representations of thermal systems; however, with due attention to units and thermally oriented peculiarities, SINDA will accept R-C networks representing other types of systems.

The SINDA system consists of two main pieces: the preprocessor and the library. The SINDA preprocessor is a program which accepts problems written in the SINDA language and converts them to the FORTRAN language. The SINDA library consists of many pre-written FORTRAN sub-routines which perform a large variety of commonly needed actions and which reduce the programming effort which might have been required to solve a given problem. These routines are fully compatible with the FORTRAN routines produced by the preprocessor from the user's input.

One of the most outstanding features of SINDA is that, in addition to accepting network description cards and other relevant values as input data, it also accepts "program-like" logic statements and subroutine calls (requesting some specific operation from the library) as data, which, ultimately, permit the user to tailor the program to suit his particular problem. Figure 10.1-1 depicts the functional relation between the SINDA program and the remaining programs.

10.2 SYSTEM STRUCTURE DESCRIPTION

In the usual engineering environment, a programmer is commissioned to prepare an applications program which is subsequently made available to the engineer on a production basis. The engineer supplies input data and receives output data.

Changes to the logic and equations are difficult for the program user to implement conveniently since they must be written in a computer oriented language and must be submitted through a formal programming organization. When SINDA is used, however, the engineer need only call on the programmer to supply a standard deck of computer oriented "control cards" which will call the various elements of the system into action in the proper sequence. The engineer then formulates his problem in the engineering oriented SINDA language, assembling both data and solution techniques (i.e., logic and equations) into this card deck, which then serves as the complete input to the SINDA system. Programmer support has been minimized since the bulk of the programming effort is already built into SINDA preprocessor and library. The engineering user need only specify the data and the order and type of "program building blocks" which he deems necessary for the solution of his problem.

It should then be evident that the SINDA system is much more than an applications program. It has, in fact, all of the functions and capabilities of a special purpose operating system. Since most computers in current use in engineering environments already have operating systems built around a FORTRAN compiler, the SINDA system is designed to augment the existing FORTRAN system. Hence, the SINDA library serves as an extension to the existing FORTRAN library, and the SINDA program serves as a preprocessor to (i.e., it precedes) the existing FORTRAN compiler. This augmentation arrangement is illustrated in Figure 10.2-1.

When using the full capability of SINDA, the engineer will be required to exert a programming effort of sorts, to a major extent in the engineering-oriented SINDA language, and to a limited extent in the FORTRAN language. This, together with the wide variety of options and features offered by the system, suggests an appropriate word of caution: SINDA is a comprehensive system which cannot be mastered overnight. The prospective user should not assume that a cursory review of the Instruction Manual will lead to immediate success, nor should he assume that this manual represents a "cookbook" which will eventually yield to a plodding and rigid adherence to each and every rule. In presenting instructions on the use of a computer program, it is not possible to completely avoid some "cookbook-like" sections; however, every effort has been made to explain the "why" and "how" behind each rule, option, and feature, with the intent of encouraging the reader to think about and understand SINDA in depth.

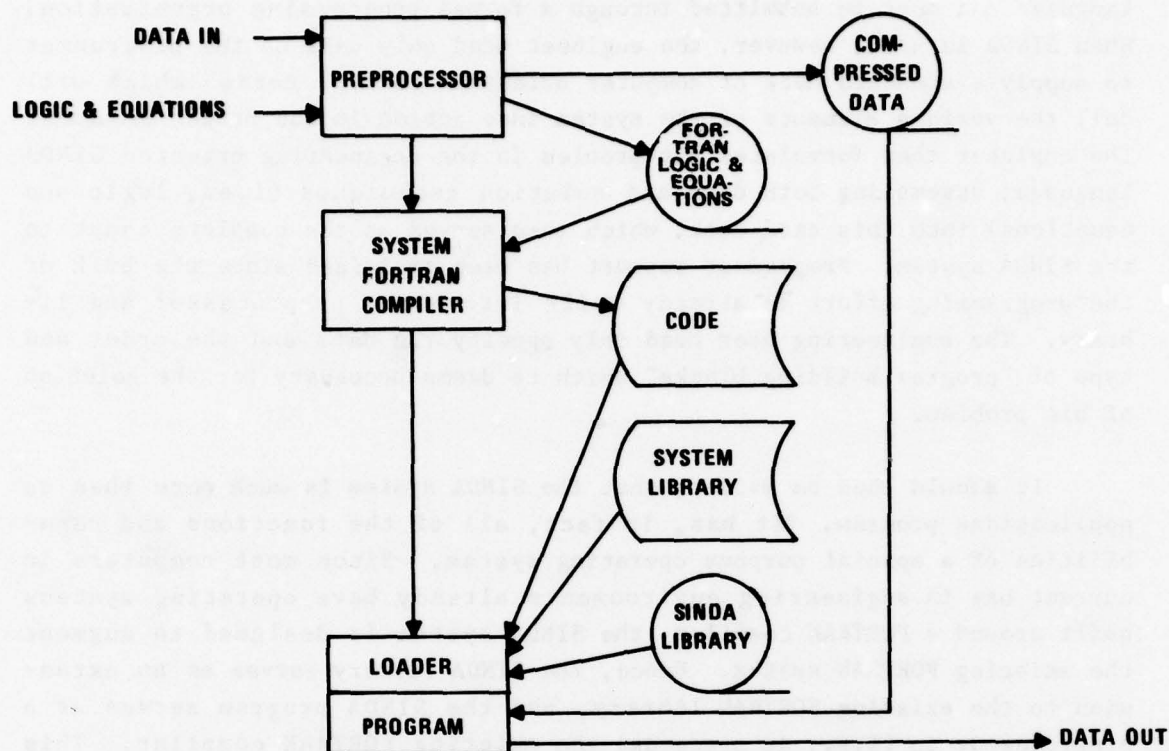


Figure 10.2-1. Detailed Internal Flow of the SINDA System

Empirical methods require mention only for the harm they have done through the limited nature of the understanding which they have produced. By "empirical methods" is meant a kind of wishful thinking (unsupported by fundamental understanding) that the solution for a given, unique problem exists somewhere within the body of previously written SINDA statements residing in the discarded printouts of prior users of the program. The methodical use of SINDA opposes this approach with the assertion that the engineer will have little confidence in the answer to his problem until he understands every detail of its solution. However, the successful "this

worked for him" or "I did it this way last time" engineer is not to be criticized for his approach, but is to be encouraged by clear expositions to observe how his special knowledge is merely a valid part of a more extensive and more useful general practice. The SINDA system is a complex synthesis of engineer, programmer, system oriented languages, and features which can be used to full advantage only as the result of a fundamental understanding of the total system. Without exception, every effort to understand SINDA will be rewarded by increased ability to use SINDA.

10.3 INPUT DECK

Two general types of problems may be handled by SINDA: thermal network solutions and general mathematical manipulations.

Thermal problems require the user to specify a network of thermal modeling elements (i.e., lumped parameters). Three DATA BLOCKS, (groups of cards), called NODE DATA, SOURCE DATA, and CONDUCTOR DATA, are provided to satisfy this requirement. Using various special formats described later, the user prepares NODE, SOURCE, and CONDUCTOR DATA cards which uniquely establish the characteristics and interconnections of the elements of his thermal network. The preprocessor will assemble and save the element characteristics data in large tables stored in the computer's memory. For example, when the preprocessor reads the NODE DATA cards, it will construct a table of initial temperatures and a table of nodal capacitances. The preprocessor also assembles the element interconnections data into an internally coded numerical list called the PSEUDO-COMPUTE-SEQUENCE (PCS). Operationally, the PCS serves two purposes: (1) it specifies the order of computation to be used when performing network heat transfer calculations, and (2) it supplies indices into the data tables where the network element characteristics may be found. For example, to compute the heat transferred during some delta-time, the computer must be instructed to perform the calculation for node-X connected to node-Y through conductor-Z, and it must be informed as to where it can locate the characteristics (temperature, capacitance, conductance, etc.) of these specific network elements.

Two additional data blocks, called CONSTANTS DATA and ARRAY DATA, are provided in the SINDA input deck to allow the user to include in his problem numerical values which are not strictly classifiable as thermal network element characteristics. The preprocessor assembles CONSTANTS and ARRAY data into tables in memory for later use in the problem solving computations. Since problems of the GENERAL type do not refer to a thermal

network, only the CONSTANTS and ARRAY data blocks may be used for this type of problem. THERMAL problems, however, require that all data blocks be present in the input deck.

In addition to the five groups of cards designated as data blocks, the SINDA input deck requires four groups of cards designated as OPERATIONS BLOCKS. These four blocks, called EXECUTION, VARIABLES 1, VARIABLES 2, and OUTPUT CALLS, are translated by the preprocessor into four FORTRAN subroutines, and serve, therefore, to specify the individual instructions required to solve the user's problem. The subroutine which results from the translation of the EXECUTION block is called by a main program (also constructed by the preprocessor) whose only purpose is to communicate to the subroutine the length and location of the various tables assembled from the five data blocks. For a simple thermal problem, the user need not concern himself with the complexities of computer programming, because the SINDA Subroutine Library contains several versatile "canned" routines for solving thermal networks. To solve such a problem, the user needs only to select an appropriate routine, punch the name of this routine on a card, and insert this card in the input deck as his EXECUTION block. This simple procedure, however, belies the extensive computational potential and versatility of the SINDA system.

In addition to a call on one or more of the standard network solution routines, the EXECUTION block may contain calls to other routines in the SINDA Library. The Library contains a wide variety of thermal, mathematical, matrix, input/output, and utility routines which may be "pieced together" within the SINDA operations blocks in order to form a complete, specialized "program" for the solution of the user's problem. For example, SINDA works in the Fahrenheit temperature system*; to produce a table of node temperatures in degrees Rankine, the user need only insert three cards in his EXECUTION block: one to call for the network solution, one to call for the addition of 460.0 to each temperature, and one to call for the printout of the node temperature table.

*Although all other units may be chosen arbitrarily as long as they are consistent, SINDA presently imposes the FARENHEIT system on all temperatures. This is due to the fact that the offset to absolute zero, 460°s, is built into the network solution routines for the purposes of evaluating radiation heat flow (which is a function of absolute temperature to the fourth power). Since current engineering practice indicates an increasing use of the metric system of units, this temperature offset will probably be converted to a user-selectable option in the near future.

Clearly, the SINDA system makes it possible for the user to avoid the programming details of such things as "DO-loops, DIMENSION statements and INDEX VARIABLES," and yet still produce a solution "program" which is tailor-made for his problem. If the user does understand FORTRAN, he may include FORTRAN statements in the operations blocks along with any necessary SINDA subroutine calls. The translated subroutines would then contain the user's FORTRAN statements in addition to the FORTRAN statements generated by the preprocessor from SINDA statements.

The network solution routines which may be called from the EXECUTION block are made quite versatile through the use of the other three operations blocks. These blocks allow the user to interject sequences of operations, specific to the problem at hand, at certain points in the midst of the "canned network solution calculations. Briefly, the subroutine resulting from the translation of the VARIABLES 1 operations block is called just prior to the network computations for a delta-time step; the VARIABLES 2 subroutine is called just after the computations for a delta-time step; and OUTPUT CALLS subroutine is called at a time interval specified by the user.

The basic structure of the SINDA input deck is shown in Figure 10.3-1. This figure illustrates the five data blocks and the four operations blocks in the correct input sequence. As shown, they are preceded by the INPUT CONTROL CARD and TITLE BLOCK, and they are followed by parametric run decks, if used, and an END OF DATA CARD.

10.3.1 Title Block

The TITLE BLOCK consists of a problem specification card (which serves as the block header card), followed by any number of optional title cards. The problem specification card informs the preprocessor of the type of problem which the rest of the input deck represents. Title cards give the user the opportunity to specify a problem title which will be printed on each page of output.

10.3.2 Data Blocks

There are five types of data blocks, each consisting of data in either integer, floating point, or Hollerith format. As shown in Figure 10.3-1, there are the node data block, source data block, conductor data block, constant data block and array data block. Only the first three (node, source and conductor) are used in thermal-type problem applications.

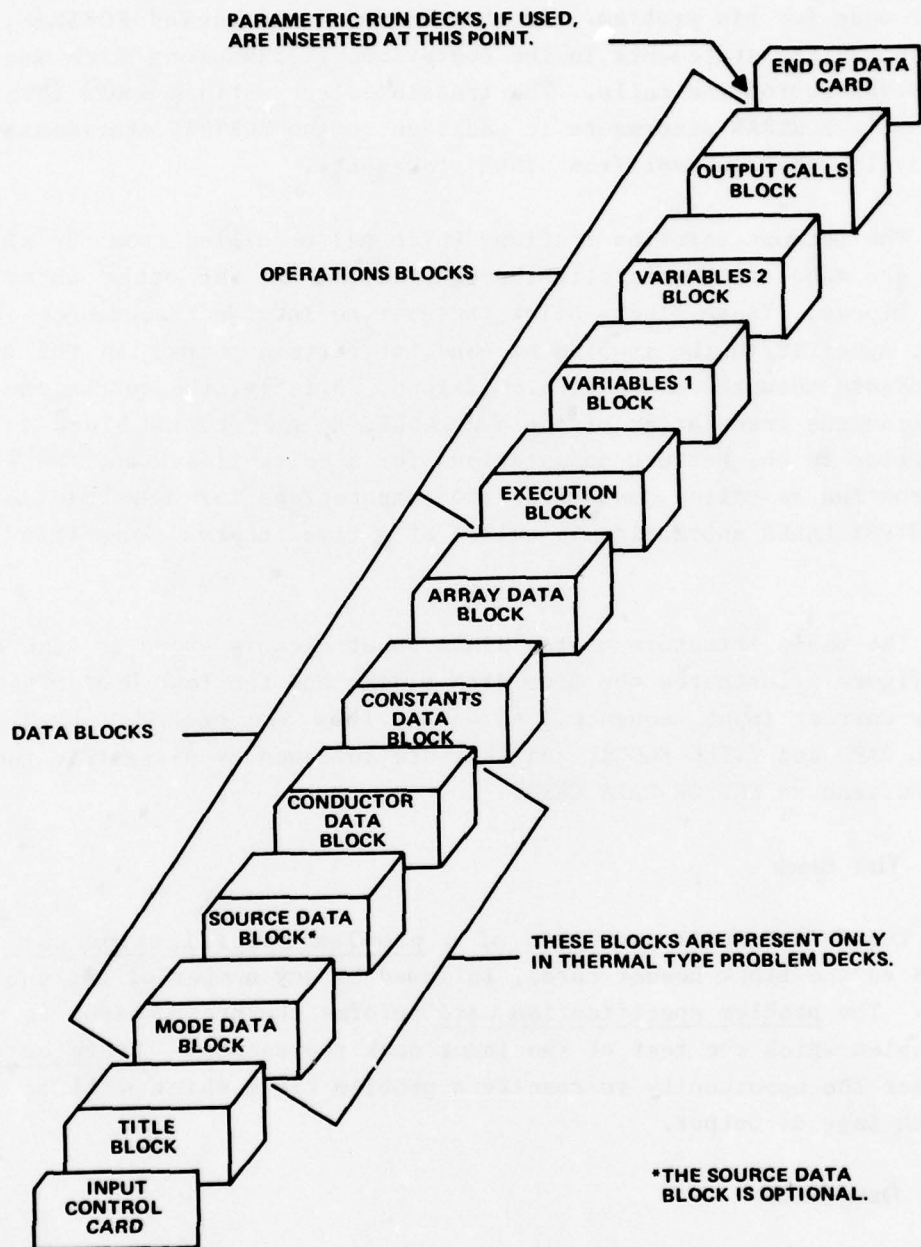


Figure 10.3-1. Basic SINDA Input Deck

10.3.2.1 Node Data Block

Three types of nodes may be defined and input by the user: diffusion, arithmetic, and boundary. Diffusion nodes have a positive capacitance and thus store energy. In the network solution routines, diffusion node temperatures are calculated by using a finite difference representation of the parabolic, differential heat transfer equation. Three locations in core memory are reserved for each diffusion node input by the user: one location to store the temperature of the node, one to store its capacitance, and one for the heat source (if any) impressed on the node. Diffusion node data input options are provided to accommodate capacitance values which are not constant (i.e., vary with temperature, etc.).

Arithmetic nodes have a capacitance of zero. The temperatures of arithmetic nodes are calculated using a finite difference representation of Poisson's equation. Since there is no capacitance value to store, only two core locations are reserved for each arithmetic node: one for the temperature and one for the impressed heat source (if any).

Boundary nodes have no capacitance and may not receive an impressed heat source. A single core location is reserved to store the temperature of each boundary node. These temperatures are not altered by the network solution routines, but may be modified, as desired, by the user.

Figure 10.3.2.1-1 summarizes the node data input options which are available to the user. Two points should be clarified. Impressed heat sources are not input with node data; they are input in the SOURCE DATA block, in which case they are transferred to the source locations automatically when needed, or they may be entered in the source locations directly with appropriate operations in the VARIABLES 1 block. Secondly, the tables of nodal capacitances are always accessible to the user in the operations blocks, and hence, a "constant capacitance value," from the standpoint of node data input, need not be held constant during the entire course of a problem's solution.

10.3.2.2 Source Data Block

This block provides the user with a convenient means for defining heat sources which are to be impressed upon the nodes defined in the NODE Data block. Sources are always input in units of:

$$\frac{\text{ENERGY}}{\text{TIME}}$$

As previously described, one memory location in the heat source table is reserved for each diffusion and arithmetic node that is defined. These locations may be referenced from the operations blocks by using the form: Q_n , where $n = \text{actual node number}$. All Q locations are set to zero by the network solution routines at the beginning of each time-step iteration. It is appropriate to use operations in the VARIABLES 1 block to reload the Q locations with the required source values because this block is called just prior to performing the heat transfer calculations. By providing a "shorthand" method for defining the most common types of sources (i.e., constant, time variant, and temperature variant), the SOURCE DATA block relieves the user of the burden of preparing source computation algorithms and inputting them in the VARIABLES 1 block.

It should be clearly understood that the preprocessor will not make any direct entries in the heat source table (Q locations) based on data in the SOURCE DATA block but, rather, it will provide for appropriate operations following the VARIABLES 1 block which will automatically add the specified values to the Q locations during each time-step iteration of the network solution. Since sources defined in the SOURCE DATA block are ADDED to the Q locations after the VARIABLES 1 operations are completed, the user need not fear that these SOURCE DATA values will erase any Q location entries made directly from the VARIABLES 1 block.

The source data input options and their associated card codes are summarized in Figure 10.3.2.2-1.

10.3.2.3 Conductor Data Block

Two basic types of conductors may be defined and input by the user: (1) LINEAR and (2) RADIATION. The conductance of a linear conductor is input in units of

OPTION (CODE)	NODE TYPE			DESCRIPTION
	D	A	B	
3 blanks	x	x	x	TO INPUT A SINGLE NODE WHERE THE CAPACITANCE IS GIVEN AS A SINGLE, CONSTANT VALUE.
CAL	x			TO INPUT A SINGLE NODE WHERE THE CAPACITANCE WILL BE CALCULATED BY THE PREPROCESSOR FROM FOUR FACTORS INPUT BY THE USER.
GEN	x	x	x	TO GENERATE AND INPUT A GROUP OF NODES; EACH HAVING THE SAME INITIAL TEMPERATURE AND THE SAME CAPACITANCE.
SIV SPV	x			TO INPUT A SINGLE NODE WHERE THE CAPACITANCE VARIES WITH TEMPERATURE. FOR SIV, THE CAPACITANCE IS FOUND BY INTERPOLATING ON AN ARRAY OF TEMPERATURE VS CAPACITANCE. FOR SPV, THE CAPACITANCE IS FOUND BY COMPUTING AN N-TH ORDER POLYNOMIAL FUNCTION OF TEMPERATURE.
SIM SPM	x			TO GENERATE AND INPUT A GROUP OF NODES, EACH HAVING THE SAME INITIAL TEMPERATURE AND THE SAME TEMPERATURE VARYING CAPACITANCE. FOR SIM, C IS FOUND BY INTERPOLATING ON AN ARRAY OF T VS C. FOR SPM, C IS FOUND BY COMPUTING A POLYNOMIAL IN T.
DIV DPV	x			TO INPUT A SINGLE NODE CONSISTING OF TWO MATERIALS WHICH HAVE DIFFERENT TEMPERATURE VARYING CAPACITANCES. FOR DIV, C1 AND C2 ARE TAKEN FROM ARRAYS OF T VS C. FOR DPV, C1 AND C2 ARE COMPUTED FROM POLYNOMIALS IN T.
DIM DPM	x			TO GENERATE AND INPUT A GROUP OF NODES, EACH OF WHICH CONSISTS OF THE SAME TWO MATERIALS HAVING DIFFERENT TEMPERATURE VARYING CAPACITANCES. FOR DIM, C1 AND C2 ARE TAKEN FROM ARRAYS OF T VS C. FOR DPM, C1 AND C2 ARE COMPUTED FROM POLYNOMIALS IN T.
BIV	x			TO INPUT A SINGLE NODE WHERE THE CAPACITANCE IS A FUNCTION OF TIME AND TEMPERATURE. THE CAPACITANCE IS FOUND BY INTERPOLATING ON AN ARRAY OF TIME AND TEMPERATURE VS CAPACITANCE.

D = DIFFUSION A = ARITHMETIC B = BOUNDARY

Figure 10.3.2.1-1. Summary of Node Data Input Options

OPTION (Card Code)	DESCRIPTION
(3 blanks)	TO IMPRESS A CONSTANT HEAT SOURCE ON A SINGLE NODE.
GEN	TO IMPRESS THE SAME CONSTANT HEAT SOURCE ON SEVERAL NODES.
SIV	TO IMPRESS A TEMPERATURE VARYING HEAT SOURCE ON A NODE.
SIT	TO IMPRESS A TIME VARYING HEAT SOURCE ON A NODE.
DIT	TO IMPRESS THE SUM OF TWO TIME VARYING HEAT SOURCES ON A NODE.
DTV	TO IMPRESS THE SUM OF A TIME VARYING SOURCE AND A TEMPERATURE VARYING SOURCE ON A NODE.
CYC	TO IMPRESS A CYCLIC TIME VARYING SOURCE ON A NODE.

Figure 10.3.2.2-1. Summary of Source Data Input Options

$$\frac{\text{ENERGY}}{\text{TIME} \bullet ^\circ \text{F}} \quad (1)$$

and the heat flow rate through such a conductor is calculated in the network solution routines as:

$$\dot{Q} = G \bullet (T_i - T_j) \quad (2)$$

where: \dot{Q} = Heat rate (ENERGY/TIME)
 G = Conductance
 T = Temperature

Several types of physical heat transfer mechanisms can be modeled as linear conductors. For heat transfer by conduction, the conductance should be computed as:

$$G = \frac{K \bullet A}{L} \quad (3)$$

where: k = Thermal conductivity of the material (ENERGY/LENGTH-TIME-°F)
 A = Cross sectional area of the conduction path (LENGTH²)
 L = Length of the conduction path (LENGTH)

For heat transfer by convection, the conductance should be computed as:

$$G = h \cdot A \quad (4)$$

where: h = Convective film coefficient (ENERGY/LENGTH²-TIME-°F)
 A = Surface area (LENGTH²)

For heat transfer by mass flow, the conductance should be computed as:

$$G = \dot{m} \cdot C_p \quad (5)$$

where: \dot{m} = Mass flow rate (MASS/TIME)
 C_p = Specific heat of the flowing material (ENERGY/MASS-°F)

The conductance of a radiation conductor is input in units of

$$\frac{\text{ENERGY}}{\text{TIME} \cdot \text{R}^4} \quad (6)$$

and the heat flow rate through such a conductor is calculated in the network solution routines as:

$$\dot{Q} = G \cdot (T_i - T_j) \quad (7)$$

However, the value that is input as the conductance of a radiation conductor should be computed as:

$$G_{\text{input}} = \sigma \epsilon F A \quad (8)$$

where: σ = Stephan-Boltzman constant (ENERGY/LENGTH²-TIME-°R⁴)
(e.g., 0.1714×10^{-8} BTU/FT²-HR-°R⁴)
 ϵ = Emissivity
 F = Shape factor
 A = Surface area (LENGTH²)

The network solution routines automatically premultiply the input conductance value by $((T_i + 460) + (T_j + 460))((T_i + 460)^2 + (T_j + 460)^2)$ each time a

radiation conductor is processed so that all subsequent calculations may use the same heat rate equation used for linear conductors. However, this premultiplication utilizes scratch memory so that the input radiation conductance value, $\sigma \epsilon F A$, remains unaltered in the table of conductances.

To facilitate the modeling of fluid loops, SINDA allows any conductor to be specified as a ONE-WAY CONDUCTOR. One-way conductors permit the realistic modeling of heat transfer by fluid (mass) flow, and their conductances are always computed as mCp . Such a conductor is defined by prefixing the node number of one of the adjoining nodes with a minus sign. The node so designated will not be allowed to lose or gain heat through the conductor, even though the temperature of the node will be used to calculate a heat flow. In other words, the solution subroutines will compute the heat transferred through a one-way conductor as though it were an ordinary conductor. This heat will not, however, be allowed to enter (or leave) the node prefixed by the minus sign; it will be allowed to leave (or enter) the unsigned node.

Figure 10.3.2.3-1 summarizes the conductor data input options available to the user. It should be remembered that the table of conductance values is always available to the user in the operations blocks, and hence, a "constant conductance value" from the standpoint of conductor data input need not be held constant during the entire course of a problem's solution.

10.3.2.4 Constants Data Block

The purpose of the CONSTANTS DATA block is to provide a means for defining and initializing SIMPLE VARIABLES. A simple variable requires one core storage location and is referenced within a program by a symbolic name or identifier. The name of a simple variable is associated, not with a specific data value, but with the address of the memory location where the current value of the variable is stored.

There are two types of constants which may be defined in the CONSTANTS DATA block: CONTROL constants and USER constants. Control constants have preassigned names and are used primarily for communicating various parameters to the network solution routines. Some constants, called DUMMY control constants, do not have preassigned uses and may therefore be utilized for temporary storage of values whose significance is peculiar to the user's problem.

OPTION (CODE)	DESCRIPTION
3 blanks	TO INPUT A SINGLE CONDUCTOR WHERE THE CONDUCTANCE IS GIVEN AS A SINGLE, CONSTANT VALUE.
CAL	TO INPUT A SINGLE CONDUCTOR WHERE THE CONDUCTANCE WILL BE CALCULATED BY THE PREPROCESSOR FROM FOUR FACTORS INPUT BY THE USER.
GEN	TO GENERATE AND INPUT A GROUP OF CONDUCTORS, EACH HAVING THE SAME CONDUCTANCE.
SIV SPV	TO INPUT A SINGLE CONDUCTOR WHERE THE CONDUCTANCE VARIES WITH TEMPERATURE. FOR SIV, THE CONDUCTANCE IS FOUND BY INTERPOLATING ON AN ARRAY OF TEMPERATURE VS CONDUCTANCE. FOR SPV, THE CONDUCTANCE IS FOUND BY COMPUTING AN N-TH ORDER POLYNOMIAL FUNCTION OF TEMPERATURE.
SIM SPM	TO GENERATE AND INPUT A GROUP OF CONDUCTORS, EACH HAVING THE SAME TEMPERATURE VARYING CONDUCTANCE. FOR SIM, G IS FOUND BY INTERPOLATING ON AN ARRAY OF T VS G. FOR SPM, G IS FOUND BY COMPUTING A POLYNOMIAL IN T.
DIV DPV	TO INPUT A SINGLE CONDUCTOR REPRESENTING A PATH THROUGH TWO MATERIALS WHICH HAVE DIFFERENT TEMPERATURE VARYING CONDUCTANCES. FOR DIV, G1 AND G2 ARE TAKEN FROM ARRAYS OF T VS G. FOR DPV, G1 AND G2 ARE COMPUTED FROM POLYNOMIALS IN T.
DIM DPM	TO GENERATE AND INPUT A GROUP OF CONDUCTORS, EACH REPRESENTING THE SAME PATH THROUGH TWO MATERIALS WHICH HAVE DIFFERENT TEMPERATURE VARYING CONDUCTANCES. FOR DIM, G1, AND G2 ARE TAKEN FROM ARRAYS OF T VS G. FOR DPM, G1 AND G2 ARE COMPUTED FROM POLYNOMIALS IN T.
BIV	TO INPUT A SINGLE CONDUCTOR WHERE THE CONDUCTANCE IS A FUNCTION OF TIME AND TEMPERATURE. THE CONDUCTANCE IS FOUND BY INTERPOLATING ON AN ARRAY OF TIME AND TEMPERATURE VS CONDUCTANCE.
PIV PIM	TO INPUT A CONDUCTOR WHERE THE CONDUCTANCE IS A FUNCTION OF BOTH AN ARBITRARY TIME VARIABLE AND TEMPERATURE. PIV DEFINES A SINGLE CONDUCTOR; PIM DEFINES SEVERAL SIMILAR CONDUCTORS.

Figure 10.3.2.3-1. Summary of Conductor Data Input Options

If a particular routine called in an operations block requires a value for a certain control constant, and if the user has not specified some reasonable value, an appropriate error message will be printed and the program will be terminated. The user should check the description of each subroutine being used to determine the control constants required and the acceptable range of values for each.

USER constants are defined and input as required for a particular problem. Each such constant is assigned a positive integer reference number (1 to 5 digits) by the user.

10.3.2.5 Array Data Block

An array is an ordered list of data values occupying sequential locations in memory. In SINDA, an array is defined by entering a sequence of three or more data values in the ARRAY DATA block. The first data value must be a positive integer which will be interpreted as the user-assigned array reference number. The last data value must be the Hollerith value 'END '. All intervening data values will be accepted as being the ELEMENTS of the list which constitute the array.

Numerous SINDA subroutines available in the operations blocks, as well as the interpolation and polynomial evaluation options available in the NODE, CONDUCTOR, and SOURCE DATA blocks, require that the exact number of data values in an array be specified as an integer. In order to reduce the number of input parameters and the chance of error, the preprocessor counts the number of data values in each array and enters this integer count as the zero-th element of the array.

SINDA ARRAY specifications may take the form of singlet arrays, doublet arrays, bivariate arrays, trivariate arrays, and matrices.

10.3.3 Operations Blocks

An operations block is a group of cards, preceded by a block header card and followed by an END card, which specifies a sequence of operations to be performed on the data previously input in the five SINDA data blocks. Each block is translated by the preprocessor into a corresponding FORTRAN subroutine, which in turn, is translated by the system FORTRAN compiler into machine executable code. Thus, an operations block prepared by the user becomes a program executable by the computer. To facilitate the discussion

in the following sections, the name of each subroutine and its corresponding operations block name is shown below:

<u>Operations Block Name</u>	<u>FORTTRAN Subroutine Name</u>
EXECUTION	EXECTN
VARIABLES 1	VARBL1
VARIABLES 2	VARBL2
OUTPUT CALLS	OUTCAL

Three basic types of operations may be included in an operations block, as follows:

1. SINDA statements
2. F-type FORTRAN statements
3. M-type FORTRAN statements

A SINDA statement is a simplified form of a subroutine call which dispenses with unnecessary key words and permits arguments to be specified using reference forms keyed to the actual numbering system developed in the data blocks. An F-type statement is any valid FORTRAN statement. An M-type statement is similar to an F-type statement, except that the actual numbering system may be used to reference data. That is, F-type statements are FORTRAN statements which are not translated by the preprocessor, whereas M-type statements are FORTRAN statements which are modified by the preprocessor to reflect the same translation from the actual to the relative numbering system that is applied to SINDA statements. Each of the three types of statements is illustrated below:

```

SINDA STATEMENT:      STFSEP (A9+2, T3, G18, K4, A7, Q10)
F-TYPE FORTRAN STATEMENT: CALL STFSEP (A(84), T(46), G(6), K(27), A(61),
                        Q(32)) T(46) = G(6)
M-TYPE FORTRAN STATEMENT: CALL STFSEP (A(9+2), T3, G18, K4, A7, Q10)
                        T3 = G18

```

The following discussion of the four SINDA operations blocks requires the introduction of several concepts which may be foreign to the prospective user who is not familiar with computer programming techniques. This situation arises because these blocks are the vehicle through which the user specifies the sequence of operations (mathematical and otherwise) to be performed by the computer in order to solve the user's problem, and, as such, they constitute a "computer program." Just as the data blocks are

used to tell the plug-and-grind computer WHAT to plug, the operations blocks are used to tell it HOW to grind.

After the preprocessor has processed the data blocks and translated the operations blocks, the compressed data is placed on a magnetic tape (thus releasing core memory for other uses) and the resulting subroutines are passed to the system FORTRAN compiler. Following compilation, the resulting program is loaded into core and executed as shown in the flow chart in Figure 10.3.3-1.

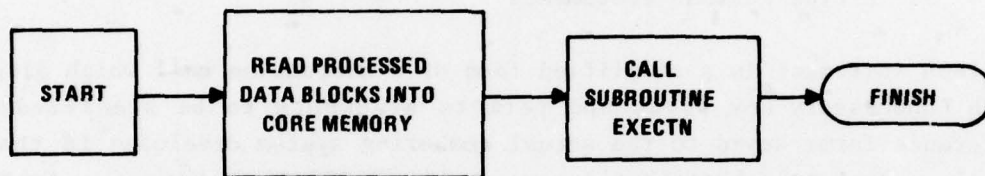


Figure 10.3.3-1. Basic Program Flow

This simple flow chart reveals and implies several things:

1. The actions depicted in the flow chart take place within the framework of a "main program." Since this "main program" is fabricated entirely by the preprocessor, the actions therein occur automatically from the standpoint of the user (i.e., the user has no control over these actions).
2. The first action performed automatically is the reading of the processed data from the magnetic tape back into core memory.
3. The next and final action which is performed automatically is a transfer of control to SUBROUTINE EXECUTN.
4. Clearly, the first opportunity for the user to specify operations which will lead to the solution of his problem occurs in the EXECUTION block. That is, if the user includes no operations in his EXECUTION block,

then subroutine EXECTN will be empty (i.e., it will indicate that nothing is to be performed) and, for all intents and purposes, his program will do nothing. Conversely, exactly and only those operations included by the user in his EXECUTION block will appear and will be performed in subroutine EXECTN when it is called.

5. The basic flow chart includes no explicit reference to subroutine VARBL1, VARBL2, or OUTCAL.
6. The basic flow chart applies to both THERMAL- and GENERAL-type problems.

No general flow chart of the EXECUTION block (and, hence, subroutine EXECTN) can be presented here because everything in the block must be placed there by the user and the sequence of operations will always be specific to a particular problem. However, the flowchart for a typical EXECUTION block is shown, for illustrative purposes, in Figure 10.3.3-2. The last line in each block of this flowchart shows the SINDA operation which will accomplish the action described there in words. The various subroutines referenced by name (i.e., D1DEGL, STFSEP, CNFRWD, and PRNTMP) exist in pre-written, "canned" form in the SINDA library.

It will be noticed that this flowchart still makes no explicit mention of subroutines VARBL1, VARBL2, or OUTCAL. This situation will be true, in general, for any EXECUTION block flowchart prepared by the user, because these subroutines are called automatically from within the pre-written network solution subroutines (e.g., CNFRWD). They are not, in general, called directly by the user. In other words, EXECTN, as directed by the user, calls CNFRWD. CNFRWD, at certain points in the sequence of heat transfer calculations, calls VARBL1, VARBL2 and OUTCAL. Each of these routines in turn, again as directed by the user, calls upon various library subroutines to perform operations which are unique to the problem at hand. Thus, the operations included in the VARIABLES 1, VARIABLES 2, and OUTPUT CALLS blocks are used to "customize" the pre-written, "canned" network solution routines so that they can accommodate any and all of the peculiarities which are specific to a given user's problem. Figure 10.3.3-3 will aid the user in visualizing that which has just been described--namely, that the operations entered by the user in the VARIABLES 1, VARIABLES 2, and OUTPUT CALLS blocks serve as "customized additives" to the pre-written operations contained in the various thermal network solution subroutines (of which, the most commonly used is subroutine CNFRWD).

The flowchart in Figure 10.3.3-4 details the sequence of operations contained in a typical network solution routine, CNFRWD. The flowcharts

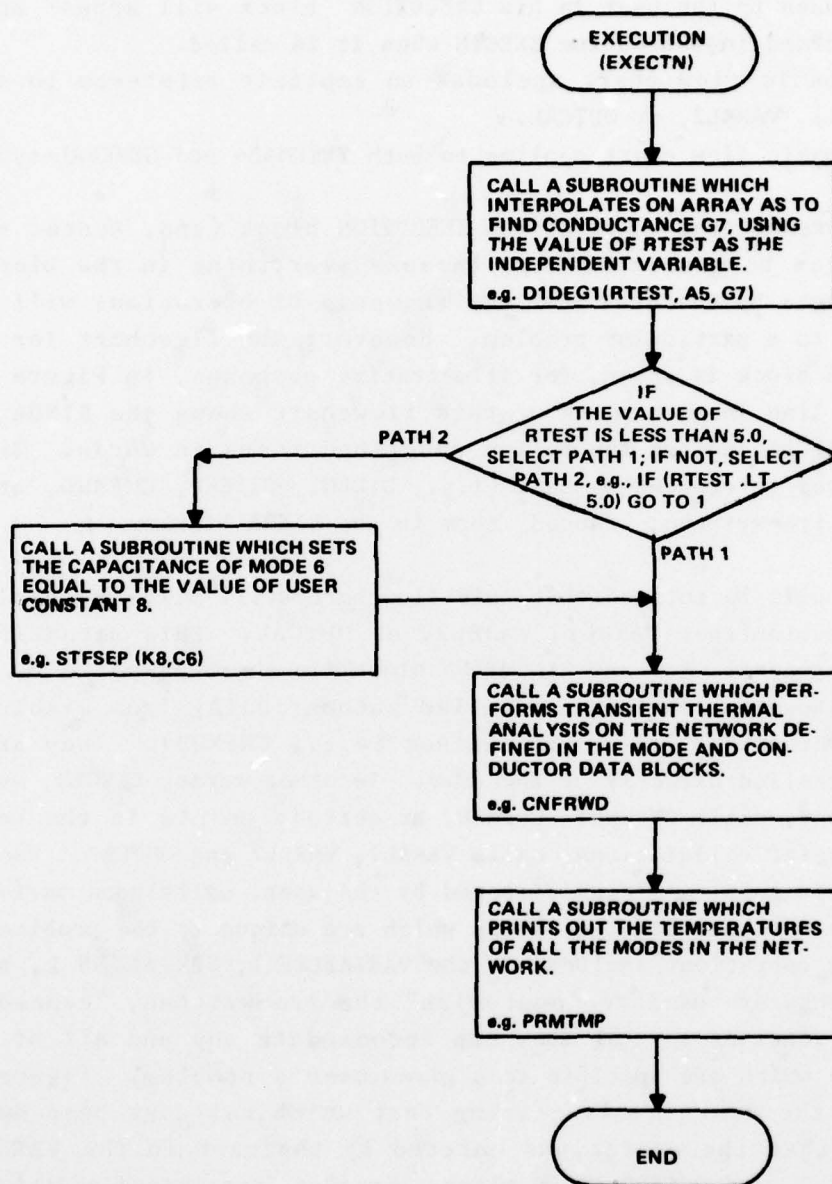


Figure 10.3.3-2. Sample Flow Chart for the Execution Block

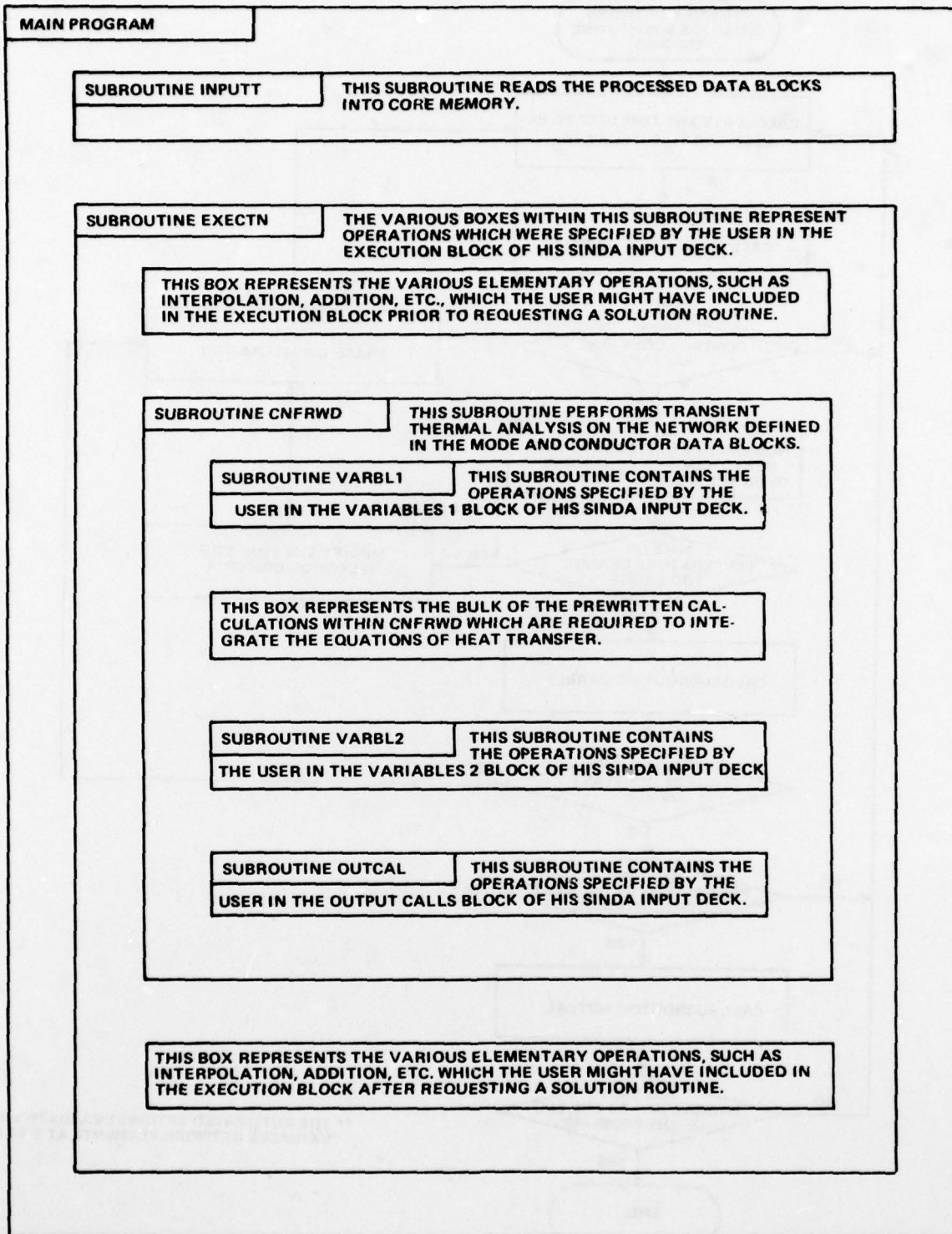


Figure 10.3.3-3. Nested Structure of the Operations Blocks

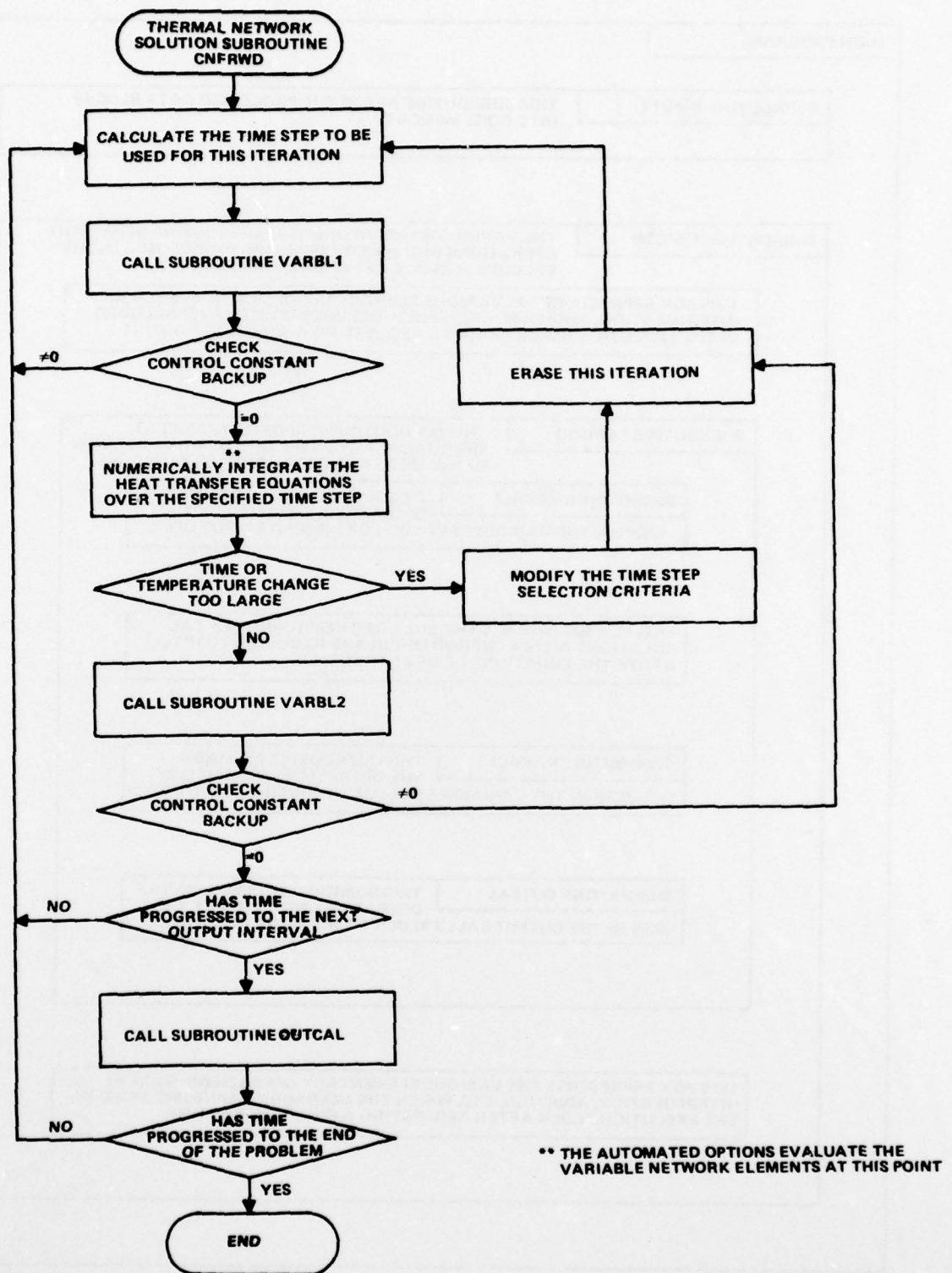


Figure 10.3.3-4. Flow Chart of Network Solution Subroutine CNFRWD

for the other solution routines available in the SINDA library are similar to the one shown here. Careful examination of this flowchart will reveal the extensive versatility which may be built into the solution routines through the creative use of the VARIABLES 1, VARIABLES 2, and OUTPUT CALLS blocks. The VARIABLES 1 block allows the user to interject operations which will be performed after the time step for each iteration is computed, but before the actual heat transfer equations are integrated. The VARIABLES 2 block allows the user to interject operations which will be performed after the equations are integrated, and the OUTPUT CALLS block allows the user to interject operations which will be performed only when the problem time has progressed to a multiple of some specified interval.

The flowchart also reveals how the SINDA CONTROL CONSTANTS can be used to achieve program flow control within the network solution routines. For example, the check on control constant BACKUP is built into subroutine CNFRWD. However, since control constants are accessible from all the operations blocks, the user could, for example, set the value of BACKUP equal to 1.0 in his VARIABLES 2 block, and thus cause CNFRWD to automatically erase the previous iteration. Control constant OUTPUT is used to specify the time interval at which subroutine OUTCAL will be performed. Other control constants provide checks on the time-step used, the maximum temperature change calculated, etc. By examining and modifying these constants in the VARIABLES 1, VARIABLES 2, and OUTPUT CALLS blocks, the user is able to effect complete control over the flow of the network solution routines, in addition to interjecting his own operations into the pre-written sequence of calculations contained therein.

The sample flowchart for the EXECUTION block (Figure 10.3.3-2) represents the result of applying three basic concepts (i.e., flowchart development, subroutine usage, and program flow control) to a specific problem. These concepts are equally applicable to the VARIABLES 1, VARIABLES 2, and OUTPUT CALLS blocks. The structure of the flowchart indicates the exact sequence of operations which the user has established as being necessary for the solution of his problem. The discrete operations detailed in each box are mechanized through the usage of pre-written subroutines from the SINDA library, and the FORTRAN "IF" statement (in the diamond) is used to accomplish program flow control (i.e., the selection of alternate sequences of operations based on certain values and criteria). These three concepts must be applied by the user to each individual problem which he desires to solve. Various guidelines for structuring the flow of an operations block exist, but the user must exercise a certain degree of judgement in applying

them to his problem. In addition, the user should recognize that the SINDA system provides him with the capability and freedom to innovate in those cases where general guidelines cannot be applied. It is up to the user to select the routines which are appropriate for his problem, as well as to supply them with appropriate actual arguments. And in the same fashion, the responsibility for selecting and correctly applying these methods rests with the user.

AD-A073 383

RESEARCH TRIANGLE INST RESEARCH TRIANGLE PARK N C
AFAL SIMULATION FACILITY/CAPABILITY MANUAL. VOLUME II. EXECUTIV--ETC(U)
FEB 79 R L EARP

F/6 9/5

F33615-76-C-1308

UNCLASSIFIED

AFAL-TR-77-118-VOL-2

NL

3 OF 3

AD
A073383



END
DATE
FILMED
10-79
DDC

SECTION 11 ASPEC-ADVANCED SIMULATION PROGRAM FOR ELECTRONIC CIRCUITS

11.1 INTRODUCTION

ASPEC is a machine-independent computer program designed to perform nonlinear dc, nonlinear transfer function, nonlinear transient and linear ac simulation of circuits containing independent sources, linear elements and nonlinear devices.

11.2 PROGRAM ANALYSIS CAPABILITIES

Nonlinear dc analysis determines the quiescent state of the circuit. The user can print out a table of the node voltages, and the source, element and device currents and power dissipations in the dc state. If the circuit does not have a well-defined dc state, the user can optionally define fixed dc node voltages as initial conditions.

Nonlinear transfer function analysis is a series of dc analyses performed while sweeping the value of a source, element or device model. The user may print out tables or produce line printer plots of voltages or currents versus the changing value. Furthermore, "snapshots" of the complete circuit operating state may also be printed at selected values.

Nonlinear transient analysis predicts the circuit response versus time. Charge storage effects of capacitors and inductors are simulated. The user may print tables or line printer plots of voltages or currents with respect to time. A "snapshot" capability is also available to print out the complete operating state of the circuit at selected time points.

Linear ac analysis predicts the frequency response of the circuit to "impulse" source inputs. The linearized models of nonlinear devices are automatically calculated by the program based upon the dc state of the circuit and the model information supplied. Tables and line printer plots of selected complex outputs can be printed versus frequency.

11.3 PROGRAM DESCRIPTION

Application of ASPEC relies on the specification of circuit components and program instruction statements.

11.3.1 Components Specification

Circuit components can generally be described in terms of sources, elements and devices. As such, input to ASPEC must include the description of these components.

11.3.1.1 Elements

Allowed linear elements are resistors, capacitors, inductors, transconductances, voltage-controlled piecewise-linear switches and batteries. In addition, coupled inductors are allowed.

11.3.1.1.1 Resistors, capacitors, and inductors

Linear resistors, capacitors, and inductors are all specified in the same format. Specification must include a unique identifying name, directional node numbers for positive branch current, the nominal resistance, capacitance or inductance of the element at 25 degrees C, and optional first and second order temperature coefficient values which define how the element value changes over temperature.

11.3.1.1.2 Battery

The battery element is intended primarily for simulation of Zener diodes which are always in breakdown, although it can be used anywhere that a constant voltage is required. Specification must include a unique name, directional node numbers defining positive branch current, the nominal voltage of the battery at 25 degrees C, the internal resistance of the battery (default of 10 ohms), and the optional value of the first order temperature coefficient in parts per degree C of the battery voltage.

11.3.1.1.3 Transconductance

The specification of a transconductance element must include a unique identifying name and directional connecting node numbers of the element. In addition, description of the element must include directional controlling node numbers of the gm element, the transconductance of the gm element at a temperature of 25 degrees C, and optional first and second order temperature coefficients in parts per degree C which define how the transconductance value changes over temperature.

11.3.1.1.4 Voltage - controlled switch

This element is actually a resistor whose value changes during transient analysis, depending upon the voltage across two nodes. Specification must include a unique identifying name, directional node numbers across which the switch is connected, directional controlling voltage nodes, and the break voltage value which determines the resistance of the switch during transient simulation. Also included must be the switch type. During DC and linear AC analysis, a switch which is normally open (NO) will be assigned the largest resistance value. A switch which is normally closed (NC) will be assigned the smallest resistance value. Also to be specified is the switch resistance when the voltage across the controlling voltage nodes is less than or equal to the break voltage and also when it is greater than the break voltage.

11.3.1.1.5 Coupled inductors

Any two inductors in the data may be defined as having a mutual coupling factor. Coupled inductors are specified by a unique identifying name of the coupled inductor pair and unique names of the two inductors which are mutually coupled. Note that each of these two inductors must have their node numbers and values assigned elsewhere in the data. Either or both of the inductors may be coupled to any other inductors in the circuit. Specification will also include the mutual inductance value between the two inductors.

11.3.1.2 Sources

Independent voltage and current sources may be dc, piecewise-linear, piecewise-exponential and sinusoidal with respect to time for DC and transient analysis. For linear AC analysis, sources may have a finite ac amplitude.

11.3.1.2.1 DC sources

A dc source remains constant for all time values and is specified as either a voltage or current source. Specification should include a unique identifying name, directional node numbers, and the value of voltage (in volts) or current (in amps). Source values may be positive, negative, or zero.

11.3.1.2.2 Piecewise-linear source

A piecewise-linear source is defined with value-time "breakpoint pairs." Between each of the breakpoint pairs, the source value is linearly interpolated. Specification consists of a unique name for the voltage or current source, directional node numbers, source value at time = 0 and the voltage-time or current-time breakpoint pairs. After a specified time, the source value remains constant.

11.3.1.2.3 Repeating piecewise-linear source

Repetitive piecewise-linear sources may most easily be specified by using a slight modification to the regular piecewise linear source specification. Specification is identical to the regular piecewise-linear source except that the repeating source does not remain constant, and therefore requires that a repeat time be specified.

11.3.1.2.4 Piecewise-exponential source

A piecewise-exponential source is somewhat similar to a piecewise-linear source except that the source value rises and falls exponentially. Instead of specifying value-time breakpoints, the user defines start times and limit values.

Piecewise-exponential source values (e.g., for a voltage source) are calculated as follows:

The dc source value (V_{dc}) is used from time $T=0$ until the first time-point (t_1). The value then begins exponentially approaching the value V_1 according to the equation:

$$V = V_{dc} + (V_1 - V_{dc}) * (1 - \exp((T - t_1)/\tau))$$

where " τ " is either t_{rise} or t_{fall} depending upon whether $V_1 > V_{dc}$ or $V_1 < V_{dc}$, respectively. The source value continues asymptotically to approach V_1 until time t_2 is reached. Defining V_x as the source value at time t_2 , the source now begins changing towards the value V_2 according to the equation:

$$V = V_x + (V_2 - V_x) * (1 - \exp((T - t_2)/\tau))$$

At time t_3 , the source begins changing towards the value V_3 and so forth up through the last value-time points (V_n, t_n). After time t_n , the value asymptotically approaches V_n .

11.3.1.2.5 Repeating piecewise-exponential source

Repetitive piecewise-exponential source waveforms may be specified in a manner similar to that used for repeating piecewise-linear sources.

The specification is identical to the regular piecewise-exponential sources except that a repeat time must be defined. The source will be treated exactly like a normal piecewise-exponential source up until the last time breakpoint. After that time, the source value does not asymptotically approach the voltage or current breakpoint. Rather, the same limit values and time intervals (beginning from the specified repeat time) are continuously repeated.

11.3.1.2.6 Sinusoidal source

Specification consists of a unique identification name of the voltage or current source, directionally numbered nodes, the source value at time = 0., the peak voltage of current amplitude, the oscillation frequency in Hertz of the voltage or current source, the oscillation delay time, and the exponential damping factor.

11.3.1.2.7 AC source value

For linear AC analysis, the AC source values default to zero unless otherwise specified (i.e., voltage source nodes are grounded and current sources are effectively removed). If a source is to have a finite AC value, it must be specified. The AC values may be specified in conjunction with DC, piecewise linear, exponential or sinusoidal values.

11.3.1.3 Nonlinear devices

Nonlinear device types are p-n diodes, Schottky diodes, BJTs, JFETs and MOSFETs. The device model structures and equations are generally fixed; the user defines values to be entered into the device model equations. The more parameters the user specifies, the more complex the model used by the program.

11.3.1.3.1 PN and Schottky diodes

Specification for PN and Schottky diodes includes the diode device name, node numbers, and model name. This model name is used to refer to a specific .MODEL instruction statement which is discussed in section 11.3.2.1. Additional optional specifications include the area scaling factor, the periphery scaling factor, an optional dc current value for the initial estimate of the diode operating point, and a optional dc voltage value.

11.3.1.3.2 Bipolar junction transistors (BJT)

Specification for both Ebers-Moll and Gummel-Poon BJT devices consist of the transistor name, numbered nodes of the collector, base and emitter nodes, model name, area factor, and optional initial collector current and collector-emitter voltage values. When dc analysis is performed, these initial values are used to calculate the initial estimate of the dc operating point.

11.3.1.3.3 Junction field-effect transistors (JFET)

The JFET device specification consists of the device identification name, numbered nodes of the drain, gate and source, model name, area

scaling factor, and optional initial drain current and drain-source voltage values. For DC analyses, these initial surface values are used to calculate the first guess of the dc operating point.

11.3.1.3.4 MOS field-effect transistors (MOSFET)

Specification for the MOSFET device consists of the device identification name, the drain, gate, and source node numbers of the MOSFET, the model name, the drawn width and length values, and the optimal initial drain-source, gate-source and bulk-source voltage values. These voltage source values are used as an initial estimate of the DC operating point.

11.3.2 Instruction Statements

In addition to a description of the components, ASPEC requires instruction statements to control program operation. Instruction statements are identified by unique names beginning with a period. The period serves to avoid any name conflicts with possible component names. Example instruction cards are .PRINT, .FREQ, and .TEMP.

11.3.2.1 .MODEL Statement

The .MODEL card is used to define the model parameters associated with each different type of device. Diode, BJT, JFET and MOSFET devices all have different model parameter names and values. However, all have the same specification format. Each requires a model name (as referenced on the device specification), device type, and model parameters. The ten possible types of device models are as follows:

- PN diode
- Shottky diode
- NPN Ebers-Moll BJT
- PNP Ebers-Moll BJT
- NPN Gummel-Poon BJT
- PNP Gummel-Poon BJT
- N-type JFET
- P-type JFET
- N-channel MOSFET
- P-channel MOSFET

Each device type has parameters specific to that device.

11.3.2.2 .PARAM Statement

ASPEC allows for the assignment of numerical values indirectly through the use of parameters. For example, rather than explicitly specifying a resistor with a value of 5.3K, it can be specified with a parameter (e.g., RVAL); and elsewhere in the data, the value of RVAL can be defined by a .PARAM statement. Parameter names can be used for source, element and device specification as well as on .MODEL statements.

11.3.3 Analysis Specification

In addition to regular DC analysis, there are three types of analysis statements, .TRAN, .TFUN, and .FREQ, to request nonlinear transient, nonlinear transfer function and linear AC analysis, respectively. Note that these only request the analysis; they do not specify printed output.

11.3.3.1 DC Analysis

The program always performs a DC analysis prior to nonlinear transient or linear AC analysis. If a simple DC analysis only is desired, with no follow-on analysis, an .OP card (see next section) must be used to cause the program to perform the DC analysis and print the outputs.

11.3.3.2 .TRAN Statement

The .TRAN statement is used to control the printing timestep during transient analysis. As such, timestep and timepoint value pairs must be specified.

Note that the .TRAN card only controls the timestep used for printout. During transient analysis, the program may use considerably smaller timesteps than those specified for printout.

11.3.3.3 .TFUN Statement

For transfer function analysis, the .TFUN card defines: 1) What is to be changed (temperature or parameter value) and 2) The increments by which either the temperature or the value is to be changed.

11.3.3.4 .FREQ Statement

The .FREQ statement is used to control the frequency print points during linear AC analysis. Specification must include the type and values of the frequency increment. There are three types of frequency increment specifications allowed.

DEC - Causes 1 points per decade to be printed logarithmically between start and finish frequencies

OCT - Causes 1 points per octave to be printed logarithmically between start and finish frequencies

LIN - Causes a total of 1 points to be printed linearly between start and finish frequencies.

11.3.4 OUTPUT Specification

The .OP, .PRINT, .PLOT and .OUTPUT cards are used to define which specific node voltages, branch currents or power dissipations are to be printed and/or plotted for DC, transfer function, nonlinear transient or linear ac analysis.

Although there is a limit on the number of different outputs, multiple uses of an identical output count only as a single output. Thus, if the same output is to be both printed and plotted, or plotted more than once, it is still considered to be only one of the total allowable outputs.

The .OP card can be used to print the complete operating condition of the circuit in the DC state and at selected points during transfer function or transient analysis.

For each .PRINT card, a tabular listing of the specified outputs will be printed versus time, a parameter value, temperature or frequency.

For each .PLOT card, a line printer plot containing the waveforms of the specified outputs will be printed versus time, a parameter value, temperature, or frequency.

For each .OUTPUT card, the numerical value of one output will be printed alongside a plot of the specified outputs.

11.3.5 Macro Definition and Use

A macro is a group of sources, elements and devices that are defined as a single functional block. This block can then be used like any other circuit component.

A simple example of a macro is the differential amplifier shown in Figure 11.3.5-1. The circuit on the left may be thought of as a functional block, which can be connected to other functional blocks or to basic components, as shown on the right.

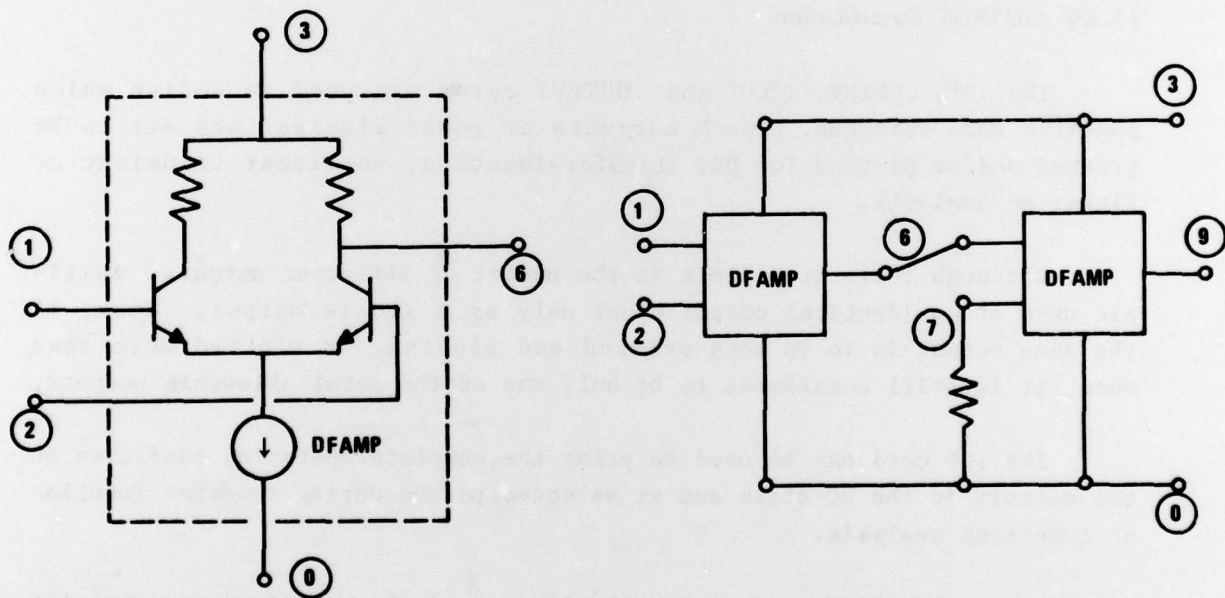


Figure 11.3.5-1. A Macro Example

Some other examples of macros are a Schottky-clamped transistor, a CMOS memory cell, a TTL gate and a functional op amp model. Whenever a circuit contains several topologically identical sections, the use of macros can greatly reduce the amount of input data necessary to describe the circuit.

The use of any macro requires both a macro definition, and one or more macro expansions.

The MACRO DEFINITION assigns a unique name to the functional block, defines the components of which it is composed, and indicates which nodes in the block are connected to outside circuit components. Also, the macro definition may indicate certain source, element or device values which will be different each time the macro is expanded.

MACRO EXPANSION involves creating a new set of circuit components, interconnected as described in the macro definition, but with generally different outside nodes. Selected component values may also be changed. It should be understood that using macros does not save computer storage; components created during macro expansion use the same amount of storage as they would if they were entered normally into the data file.

11.3.5.1 .MACRO Statement

The .MACRO statement is used to define a macro. As such, it is placed immediately preceding the data cards describing the macro components. Its specification consists of the macro name, the external macro node numbers used only to influence the nodes of the components within the macro definition, and the parameter names and values. These names and values refer to parameter names used in place of component values within the macro definition.

11.3.5.2 Xname Statement

An Xname card is used to expand a macro. Each macro may be expanded as many times as necessary up to the allowed limit, as long as the number of nodes or components does not exceed the program limits.

Specification includes node numbers, the name referencing a specific macro definition which indicates which macro is to be expanded, and optional parameter name-value definition.

When the macro is expanded, these node numbers replace, on a one-for-one basis, the external node numbers specified in the macro definition. Any new internal nodes will be automatically assigned by the program.

The parameter name-value definition, if used, will replace the parameter name used on the referenced macro statement.

When the macro is expanded, the new parameter values will replace the values used in the macro definition.

11.3.5.3 Specification of Macro Output

Normally, a macro expansion is considered to be a "black box" for which only the external nodes are of interest. However in some cases it is useful to obtain the node voltages and currents inside a macro. When an ".OP" print is specified, the internal voltages and currents are printed automatically. Specified voltages and currents can also be obtained for transient, transfer function and linear ac analysis using the .PRINT, .PLOT, and .OUTPUT cards as described below.

11.3.6 Temperature Specification

Normally, the program performs only one simulation at an assumed temperature of 25 degrees C (298 degrees K). If desired, the simulation may be run at several different temperatures by using the .TEMP card. Prior to each simulation, element values and device parameters are modified depending upon user-specified temperature coefficients.

11.3.7 Worst-Case Analysis

Worst-case analysis allows the user to multiply all resistors, capacitors, inductors, or battery element values and certain diode, BJT, JFET, or MOSFET device model parameter values by given factors prior to analysis.

11.3.8 Noise Analysis Specification

Noise analysis enables the user to predict the thermal noise and noise vs. frequency voltages across any two nodes in the circuit. Also, the equivalent input noise vs. frequency is calculated. Noise analysis

output is requested by using the .NOISE card in conjunction with a .FREQ card.

The .NOISE card is used to request calculation of the output noise voltages at each of the frequency points specified on a .FREQ card. Note that noise analysis will not be performed unless a .FREQ card is also specified.

Normally, the .NOISE card causes the program to generate plots of the output noise and reflected input noise at those frequencies specified on a .FREQ card. In addition, a variation of the .OP card allows the user to request printing of a table of the noise contribution of each resistor, diode and transistor at specified frequency points.

Note: This feature may be removed at a future date.

11.3.9 Print Control Specification

By use of the .PC statement, the user can control the amount and format of the program data printout. Options available are as follows:

- BRIEF - Suppresses reprinting of the data card images. All data cards following the .PC card are affected. Thus, if no data cards are to be printed, the .PC card must be the first card in the data deck after the Title Card.
- REPRINT - Requests that the instruction card, source, element, device and .MODEL card data as interpreted by the program be printed. This reprint will also include all components created by macro expansions. The data reprint allows checking to ensure that the input data values were interpreted correctly by the program.
- VERIFY - Causes the program to print a node connection table, iteration control parameters and program storage usage. The node connection table shows which circuit components are connected to each node in the circuit.
- ACVER - Causes the program to print the linearized Diode, BJT, JFET, and MOSFET models to be used during the linear ac analysis. ACVER is ignored if no ac analysis is requested.
- NP - (Narrow Paper) specifies that all output is to be printed assuming a 72 column page width. This option allows for output print on interactive data terminals. (If not specified, the program assumes a 132 column page width.)

11.3.10 DC Node Voltage Specification

The .DCVOLT card allows the user to define node voltage values which will be maintained throughout the DC analysis (including transfer function analysis). The effect of this card is to put a grounded voltage source on certain specified nodes during any DC analysis.

Caution should be exercised when using .DCVOLT cards since the circuit could be forced into an unrealistic DC state. The "sources" set up by the .DCVOLT card will sink or source any amount of current necessary to maintain the specified node voltages. This could result in severe start-up transients when the transient analysis begins. Linear AC analysis might also give erroneous answers if the circuit is forced into an incorrect DC state.

11.3.11 Breadboard Simulation

The .BBOARD card causes the program to attach a fixed grounded capacitor to each external node of every diode, BJT, FJET, and MOSFET in the circuit. Note that if more than one Diode, BJT, JFET, or MOSFET device is connected to a node, more than one capacitor will be added to that node. The .BBOARD card allows a convenient first-order simulation of a circuit in breadboard (rather than integrated) form.

Note that during nonlinear transient and linear AC analysis, the grounded capacitors may shunt currents to ground and thus the sum of currents into each node may not appear to equal zero.

11.3.12 Analysis Control

Analysis control allows the user to modify the program's DC and transient analysis iteration scheme. It should be used with considerable discretion since changes in the standard values could result in greatly increased execution times with no appreciable improvement in accuracy.