General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by the NASA Center for Aerospace Information (CASI)

355

Yes

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

AUG 1970
RECEIVED
NASA STI FACILITY
INPUT BRANCH

INTERNAL NOTE MSC-EB-S-68-4024-U

COMPUTER AIDED DIGITAL SYSTEMS DESIGN (CADSS)

FACILITY FORM 602

N70-35990 (ACCESSION NUMBER)	(THRU)
41 (PAGES)	1 (CODE)
TMX-65071 (NASA CR OR TMX OR AD NUMBER)	08 (CATEGORY)

MANNED SPACECRAFT CENTER
HOUSTON, TEXAS

LIBRARY COPY

August 12, 1968

INTERNAL NOTE MSC-EB-S-68-4024-U

COMPUTER AIDED DIGITAL SYSTEMS DESIGN (CADSS)

PREPARED BY

E. Franke

APPROVED BY

O. Patterson
Head, Data Systems Development Section

J. Overton
Chief, Data Systems Development Branch

P. Vavra
Chief, Information Systems Division

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
MANNED SPACECRAFT CENTER
HOUSTON, TEXAS
August 12, 1968

AUTHOR'S BIOGRAPHICAL NOTE

Dr. Franke is Associate Professor in the Department of Electrical Engineering at Texas A&I University. After receiving BS and MS degrees from Texas A&I, Dr. Franke obtained a Ph.D. ~~Degree from~~ Case-Western Reserve University in 1967. His Doctoral Dissertation is entitled: "Automated Functional Design of Digital Systems." He is presently engaged in research in computer aided design of digital systems. The following report was completed by him while participating in the 1968 NASA-ASEE Summer Faculty Fellowship Program at MSC.

C A D S S

Computer Aided Digital Systems Design

Preliminary Specifications

E. A. Franke, EB/4

NASA - MSC

Houston, Texas

August 12, 1968

TABLE OF CONTENTS

TITLE	PAGE
1 THE DESIGN OF DIGITAL SYSTEMS	1
1.1 CONCEPTUAL DESIGN	1
1.2 FUNCTIONAL DESIGN	3
1.3 LOGICAL DESIGN	3
1.4 IMPLEMENTATION DESIGN	4
1.5 CONSTRUCTION	4
1.6 COMPUTER-AIDED DESIGN	4
1.7 THE CADSS SYSTEM	6
2 THE FLOW CHART DESCRIPTION LANGUAGE	8
2.1 GENERAL CONVENTIONS	8
2.2 UNITS AND DECLARATIONS	9
2.3 CONDITION LISTS	10
2.4 OPERATION LISTS	12
2.5 STATEMENTS	15
2.6 LABELS	16
2.7 PROGRAM BLOCK STRUCTURE	16
2.7.1 BOOLEAN BLOCKS	18
2.7.2 SEQUENCED BLOCKS	18
2.7.3 CONCURRENT BLOCKS	19
2.8 SERIAL BINARY MULTIPLIER EXAMPLE	19
3 THE CADSS SYSTEM IMPLEMENTATION	24
3.1 COMPILATION	24
3.2 FLOW CHART SIMULATION	25

TABLE OF CONTENTS

TITLE	PAGE
3.3 TABLE GENERATION	25
3.4 STATE ASSIGNMENT	26
3.5 LOGIC GENERATION	28
3.6 LOGIC SIMULATION	28
3.7 LOGIC PLACEMENT AND WIRE LIST GENERATION	29
4 THE CADSS MODEL OF A DIGITAL SYSTEM	30

TABLE OF ILLUSTRATIONS

TITLE		PAGE
FIGURE 1.1	THE PROCESS OF DIGITAL SYSTEM DESIGN	2
FIGURE 2.1	MULTIPLIER BLOCK DIAGRAM	20
FIGURE 2.2	MULTIPLIER FLOW CHART	22
FIGURE 2.3	CADSS DESCRIPTION OF MULTIPLIER	23
FIGURE 4.1	FIRST MODEL OF A DIGITAL SYSTEM	30
FIGURE 4.2	SECOND MODEL OF A DIGITAL SYSTEM	31
FIGURE 4.3	THIRD MODEL OF A DIGITAL SYSTEM	32

TABLE OF REFERENCES

TITLE	PAGE
ORR, W.K. AND SPITZE, J.M., DESIGN AUTOMATION UTILIZING A MODIFIED POLISH NOTATION, PROCEEDINGS OF THE FALL JOINT COMPUTER CONFERENCE, 1964, PP 643-650.	4
AAKHUS, SEEMAN, AND PTAK, ACCLAIM - A COMPUTER AIDED DESIGN SYSTEM, COMPUTER DESIGN, VOLUME 7, NUMBER 5, PP. 64-71, MAY,1968	4
PROCTOR, R.M., A LOGIC DESIGN TRANSLATOR EXPERIMENT DEMONSTRATING RELATIONSHIPS OF LANGUAGE TO SYSTEMS AND LOGIC DESIGN, IEEE TRANSACTIONS ON ELECTRONIC COMPUTERS, VOL EC-13, PP. 422-430, AUGUST, 1964.	5
SCHLAEPPI, H.P., A FORMAL LANGUAGE FOR DESCRIBING MACHINE LOGIC, TIMING, AND SEQUENCING (LOTUS), IEEE TRANSACTIONS ON ELECTRONIC COMPUTERS, VOL EC-03, PP. 439-448, AUGUST, 1964	5
SCHORR, H., COMPUTER-AIDED DIGITAL SYSTEM DESIGN AND ANALYSIS USING A REGISTER TRANSFER LANGUAGE, IEEE TRANSACTIONS ON ELECTRONIC COMPUTERS, VOL. EC-13, PP. 730-737, AUGUST, 1964.	5
THE ADD SYSTEM (AUTOMATED DESIGN DOCUMENTATION SYSTEM) USERS MANUAL, PHILCO-FORD CORPORATION, WDL DIVISION, PALO ALTO, CALIFORNIA	7
FRANKE AND MERGLER, COMPUTER AIDED FUNCTIONAL DESIGN OF DIGITAL SYSTEMS, SWIEECO CONFERENCE PROCEEDINGS, 1968.	8

1 THE DESIGN OF DIGITAL SYSTEMS

A COMPLETE SYSTEM FOR THE AUTOMATED DESIGN OF DIGITAL SYSTEMS SHOULD BEGIN WITH AN ORIGINAL SPECIFICATION OF A SYSTEM OPERATION AND CARRY THE DESIGN PROCESS THROUGH TO A HARDWARE LAYOUT. IDEALLY THE SYSTEM SHOULD BE ABLE TO FUNCTION WITHOUT OPERATOR INTERVENTION BUT PROVISIONS SHOULD BE INCLUDED TO ALLOW THE DESIGNER TO MONITOR THE SYSTEM AS THE DESIGN EVOLVES AND TO MODIFY THE RESULTS AT ANY POINT IN THE DESIGN PROCESS.

IN ORDER TO ALLOW THE DESIGNER TO EXAMINE, CHECK, AND MODIFY THE DESIGN, THE DESIGN AUTOMATION SYSTEM MUST PRODUCE DESCRIPTIVE OUTPUTS AS THE DESIGN OF A SYSTEM PROCEEDS. CONSIDERATION OF THE COMPLETE PROCESS OF DIGITAL SYSTEM DESIGN INDICATES THAT THE PROCESS MAY BE BROKEN INTO FIVE TASKS AS SHOWN IN FIGURE 1.1.

THE OUTPUT OF EACH DESIGN TASK IS A DESCRIPTION OF THE SYSTEM THAT IS MEANINGFUL TO A DESIGNER. THE DESCRIPTION PRODUCED AT ANY POINT CAN THEN BE ANALYZED, MODIFIED AND USED AS THE INPUT TO THE NEXT DESIGN TASK.

1.1 CONCEPTUAL DESIGN

THE FIRST PHASE IN THE DESIGN OF A DIGITAL SYSTEM BEGINS WITH THE RECOGNITION OF A PROBLEM TO BE SOLVED AND A DECISION TO SOLVE THE PROBLEM BY DIGITAL TECHNIQUES. THE FIRST TASK OF THE DESIGNER IS TO DETERMINE A METHOD OR CONCEPT OF SOLUTION. THIS WILL CONSIST OF SPECIFYING SUCH THINGS AS THE TYPE OF NUMBER REPRESENTATION TO BE USED, THE NUMBER OF INTERNAL STORAGE REGISTERS TO BE PROVIDED, AND THE RELATION OF THE SYSTEM TO EXTERNAL INPUTS AND OUTPUTS.

THE TASK OF DESCRIBING THE OPERATION OF A DIGITAL SYSTEM IS COMPLICATED BY THE FACT THAT UNITS SUCH AS REGISTERS, COUNTERS, OR ADDERS ARE USED FOR DIFFERENT PURPOSES AT DIFFERENT TIMES. INFORMATION IS TRANSFERRED TO OR FROM VARIOUS REGISTERS DEPENDING ON THE PARTICULAR OPERATION BEING CARRIED OUT. THE SEQUENCE OF OPERATIONS TO BE PERFORMED WILL OFTEN DEPEND ON THE RESULTS OF PREVIOUS OPERATIONS AS WELL AS THE CONDITION OF THE INPUTS TO THE SYSTEM.

CONTROL LOGIC WILL EVENTUALLY BE USED TO SEQUENCE THE OPERATIONS OF UNITS BUT DURING THE CONCEPTUAL DESIGN PHASE THE DESIGNER SHOULD BE FREE TO CONCENTRATE ON THE OPERATION OF THE SYSTEM AS A WHOLE. THE DETAILS OF CONTROL LOGIC AND OPERATION TIMING SHOULD REMAIN UNSPECIFIED UNTIL THE OVERALL STRUCTURE OF THE MACHINE IS DETERMINED. REPRESENTATION OF

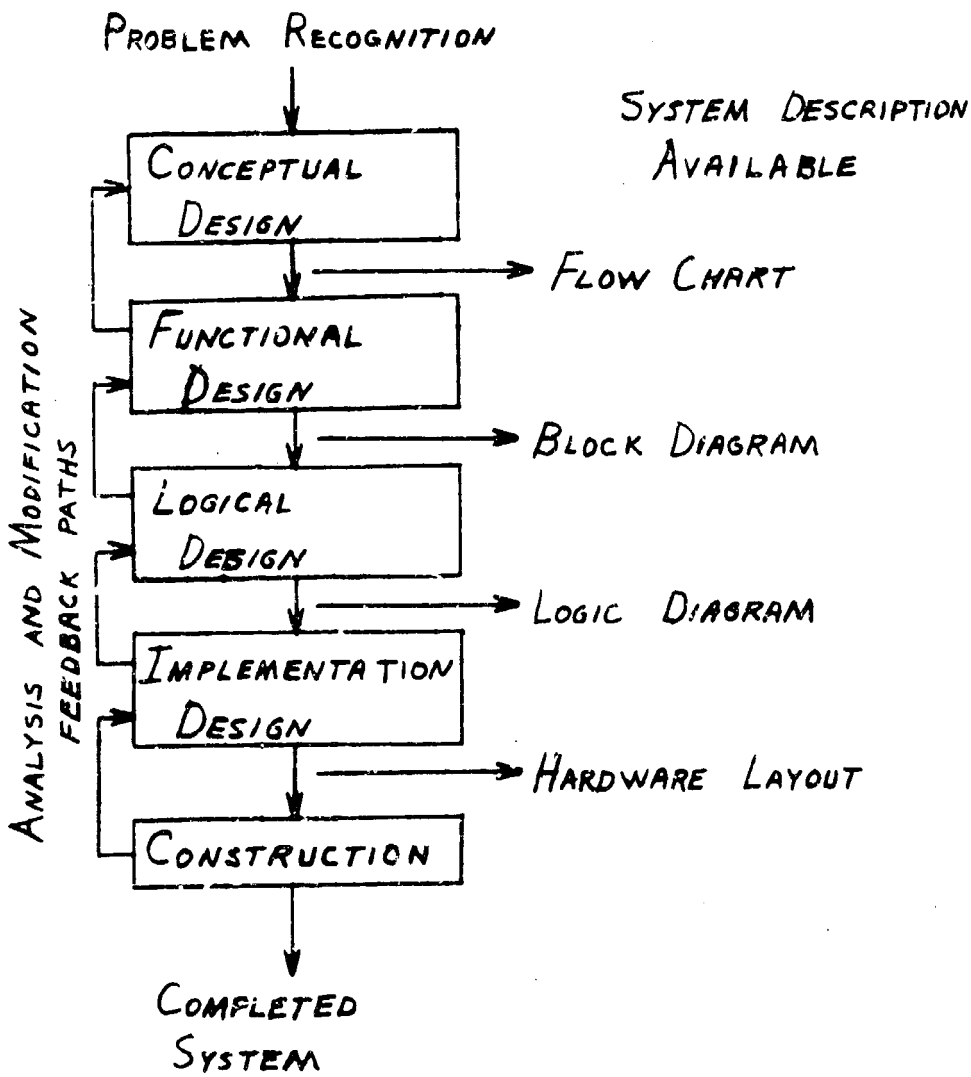

FIGURE 1.1

THE PROCESS OF DIGITAL SYSTEM DESIGN

THE SYSTEM AS A PROCESS FLOW CHART IS AN EFFECTIVE WAY TO FREE THE DESIGNER FROM CONSIDERATION OF THESE DETAILS AND STILL DESCRIBE THE RELATIONS BETWEEN INPUT, OUTPUT, AND INTERNAL OPERATIONS IN A SEQUENTIAL MANNER.

THE ENTRIES IN EACH OPERATION BLOCK OF A FLOW CHART DESCRIBE THE OPERATION OF THE ENTIRE SYSTEM AT A SPECIFIC TIME (OR FOR A SPECIFIC STATE OF THE MACHINE). THE PATHS BETWEEN OPERATION BLOCKS AND THE CONDITIONAL BRANCH BLOCKS CONTAIN THE INFORMATION THAT WILL EVENTUALLY BE TRANSLATED INTO CONTROL LOGIC. ALTHOUGH THE SEQUENCING OF OPERATIONS IS SPECIFIED, NO TIMING INFORMATION IS INCLUDED IN THE FLOW CHART. THE DESCRIPTION OF THE SYSTEM, UPON COMPLETION OF THE CONCEPTUAL DESIGN PHASE, CONSISTS OF A PROCESS FLOW CHART SPECIFYING THE OPERATIONS TO BE PERFORMED BY THE SYSTEM AND IMPLYING THE NECESSARY FUNCTIONS OF THE CONTROL LOGIC.

1.2 FUNCTIONAL DESIGN

THE SECOND PHASE OF THE DESIGN PROCESS INVOLVES THE TASK OF EXPRESSING THE IMPLIED CONTROL LOGIC EXPLICITLY AND LISTING THE OPERATIONS TO BE PERFORMED BY EACH UNIT (REGISTER, COUNTER, ETC.) IN THE SYSTEM. COMPLETION OF THE FUNCTIONAL DESIGN PHASE RESULTS IN A BLOCK DIAGRAM SHOWING THE CONDITIONS (IN THE FORM OF BOOLEAN EQUATIONS) NECESSARY FOR AN OPERATION TO BE PERFORMED IN A GIVEN UNIT AND A STATE TABLE REPRESENTING THE CONTROL LOGIC OF THE SYSTEM.

1.3 LOGICAL DESIGN

THE BLOCK DIAGRAM OF THE SYSTEM, SHOWING UNITS AND THE CONDITIONS FOR UNIT OPERATIONS, SERVES AS THE INPUT TO THE DETAILED LOGICAL DESIGN PHASE. DURING THIS PHASE OF DESIGN A PARTICULAR FAMILY OF LOGIC MUST BE SELECTED AND AN OPERATING SPEED DETERMINED ON THE BASIS OF THE NUMBER OF OPERATIONS TO BE PERFORMED AND THE TIME AVAILABLE.

THE STATE TABLE MUST BE REDUCED, AND THE CONTROL STATES SPECIFIED MUST BE ASSIGNED TO MEMORY ELEMENT CONFIGURATIONS IN A REASONABLY EFFICIENT MANNER. IF THE BOOLEAN EQUATIONS FROM THE BLOCK DIAGRAM ARE NOT IN MINIMAL FORM FOR THE PARTICULAR FAMILY OF LOGIC CHOSEN, IT IS NECESSARY TO MINIMIZE THEM. THE MINIMIZED BOOLEAN EQUATIONS MUST THEN BE CONVERTED INTO CONNECTIONS OF LOGIC ELEMENTS WITH CONSIDERATION BEING GIVEN TO PROBLEMS OF TIMING AND THE FAN-IN/FAN-OUT CAPABILITIES OF THE PARTICULAR LOGIC ELEMENTS CHOSEN. COMPLETION OF THIS PHASE OF DESIGN ACTIVITY RESULTS IN A DETAILED LOGIC DIAGRAM DESCRIBING THE SYSTEM.

1.4 IMPLEMENTATION DESIGN

THE COMPLETE LOGIC DIAGRAM OF THE SYSTEM PROVIDES THE INPUT TO THE IMPLEMENTATION, OR HARDWARE DESIGN PHASE. AT THIS TIME THE DESIGNER MUST CONSIDER SUCH THINGS AS POWER REQUIREMENTS, HEAT DISSIPATION, METHOD OF CONSTRUCTION, COMPONENT PLACEMENT, AND WIRE ROUTING. COMPLETION OF THIS PHASE OF THE DESIGN PROCESS WILL RESULT IN A HARDWARE LAYOUT DESCRIPTION AND A WIRING LIST.

1.5 CONSTRUCTION

ALTHOUGH CONSTRUCTION IS NOT NORMALLY CONSIDERED AS A PART OF THE DESIGN PROCESS, IT IS INCLUDED HERE BECAUSE OF THE APPLICATIONS OF AUTOMATION IN THE CONSTRUCTION OF DIGITAL SYSTEMS. THE TECHNIQUES OF NUMERICALLY CONTROLLED WIRE WRAPPING AND COMPONENT INSERTION ARE WELL KNOWN AND WIDELY APPLIED. CURRENT APPLICATION OF AUTOMATED CONSTRUCTION IS LIMITED TO HIGH VOLUME DESIGNS BECAUSE OF THE TIME AND EXPENSE REQUIRED TO GENERATE THE CONTROLLING INSTRUCTIONS AND TO DEBUG THE SYSTEM. MUCH OF THIS TIME AND EXPENSE COULD BE ELIMINATED IF A COMPUTER AIDED DESIGN SYSTEM PRODUCED THE INSTRUCTIONS (PUNCHED PAPER TAPE OR OTHER MEDIUM) AS THE FINAL DESCRIPTION OF THE SYSTEM BEING DESIGNED.

1.6 COMPUTER-AIDED DESIGN

THE FIRST ATTEMPTS IN THE APPLICATION OF COMPUTERS TO THE PROBLEM OF DESIGN AUTOMATION WERE IN THE AREAS OF IMPLEMENTATION AND CONSTRUCTION. COMPUTER PROGRAMS ARE CURRENTLY AVAILABLE TO PLACE LOGIC CARDS OR MODULES IN AN ARRAY SO AS TO MEET GIVEN CONSTRAINTS AND MAXIMIZE A GIVEN CRITERIA FUNCTION. PROGRAMS FOR CHECKING THE LOADING ON LOGIC ELEMENTS AND FOR PRODUCING PAPER TAPE FOR A WIRE WRAPPING MACHINE ARE ALSO AVAILABLE. WORK IS CONTINUING IN THE AREA OF PRINTED-CIRCUIT BOARD LAYOUT AND SOME PROGRAMS OF THIS TYPE HAVE BEEN WRITTEN.

ALTHOUGH SOME RESEARCH HAS ALSO BEEN DONE ON THE AUTOMATION OF THE INITIAL PHASES OF THE DESIGN PROCESS, THE RESULTS IN THIS AREA HAVE BEEN LESS SATISFACTORY. SOME OF THE TECHNIQUES IN USE (ORR AND SPITZE), (AAKHUS, SEEMAN, AND PYAK) PERFORM LOGIC MINIMIZATION BUT REQUIRE THE INITIAL DESCRIPTION OF THE SYSTEM TO BE ENTERED IN THE FORM OF BOOLEAN EQUATIONS. THE TASK OF GENERATING THE BOOLEAN EQUATIONS IS THUS LEFT TO THE DESIGNER.

IN ORDER TO REDUCE THE AMOUNT OF PREPARATION REQUIRED BEFORE THE COMPUTER AIDED PORTION OF DESIGN, SEVERAL LANGUAGES FOR DESCRIBING THE OPERATION OF A DIGITAL SYSTEM HAVE BEEN PROPOSED (PROCTOR), (SCHLAEPPI), (SCHORR, H). THE PROPOSED LANGUAGES ARE TAILORED SPECIFICALLY FOR LARGE STORED PROGRAM COMPUTER SYSTEMS. THE DESCRIPTION LANGUAGES THEMSELVES ARE COMPOSED OF CONCEPTS MORE FAMILIAR TO PROGRAMMERS THAN LOGIC DESIGNERS THUS REDUCING THE APPLICATION OF THESE TECHNIQUES.

SINCE THE FIRST AVAILABLE DESCRIPTION OF SYSTEM IS IN THE FORM OF A FLOW CHART, THE INPUT LANGUAGE TO THE COMPUTER-AIDED DESIGN SYSTEM SHOULD BE CAPABLE OF DESCRIBING SUCH A FLOW CHART. THE LANGUAGE SHOULD BE TAILORED TO THE CONCEPTS AND TERMINOLOGY OF ENGINEERING DESIGNERS WITHOUT A GREAT DEAL OF PROGRAMMING EXPERIENCE. IN ADDITION, THE LANGUAGE SHOULD PERMIT THE IMPLICIT DESCRIPTION OF THE OPERATION OF A SYSTEM WITHOUT EXPLICITLY DESCRIBING THE CONTROL LOGIC AND OPERATION TIMING.

THE COMPUTER AIDED DESIGN SYSTEM SHOULD THEN BE ABLE TO ACCEPT THE DESCRIPTION OF THE DESIRED SYSTEM, COMPILE THE VERBAL DESCRIPTION INTO AN INTERNAL STRUCTURE OF LISTS AND TABLES, AND PERFORM THE FOLLOWING FUNCTIONS

1. SIMULATION OF THE DESCRIBED DIGITAL SYSTEM. ALTHOUGH SIMULATION OF A SYSTEM DESCRIBED ON THE FLOW CHART LEVEL WILL NOT DETECT HAZARDS OR RACE CONDITIONS IT MAY BE USED TO VERIFY THE DESCRIPTION ITSELF.

2. TRANSLATION FROM A FLOW CHART DESCRIPTION TO A BLOCK DIAGRAM DESCRIPTION. THIS WILL PRODUCE THE STATE TABLE AND THE LOGICAL OPERATION TABLE FOR THE SYSTEM.

3. TRANSLATION FROM A BLOCK DIAGRAM DESCRIPTION TO A LOGIC DIAGRAM DESCRIPTION. THIS REQUIRES MINIMIZATION OF THE LOGIC EQUATIONS AND THE SELECTION OF LOGIC ELEMENTS FROM A PARTICULAR FAMILY OF LOGIC.

4. TRANSLATION FROM A LOGIC DIAGRAM DESCRIPTION TO A HARDWARE LAYOUT DESCRIPTION. THIS INVOLVES ASSIGNING LOGIC ELEMENTS TO PARTICULAR LOCATIONS AND GENERATING THE WIRING LIST FOR THE CHOSEN METHOD OF CONSTRUCTION.

1.7 THE CADSS SYSTEM

THE PRESENT DEVELOPMENT OF THE COMPUTER AIDED DIGITAL SYSTEMS SYNTHESIS (CADSS) SYSTEM REPRESENTS AN ATTEMPT TO APPLY COMPUTER-AIDED DESIGN TECHNIQUES TO THE DESIGN OF SEQUENTIAL LOGIC SYSTEMS. THE INITIAL SCOPE OF THE STUDY IS LIMITED TO THE DEVELOPMENT OF A FLOW CHART DESCRIPTION LANGUAGE AND THE PRODUCTION OF COMPUTER PROGRAMS FOR SIMULATION AND TRANSLATION TO THE BLOCK DIAGRAM DESCRIPTION. IN ORDER TO PROVIDE A FLEXIBLE SYSTEM, THE SIMULATION INPUT AND OUTPUT AS WELL AS THE STATE AND UNIT OPERATION TABLES WILL BE AVAILABLE ON EITHER THE LINE PRINTER OR THE DIGITAL TELEVISION DISPLAY SYSTEM.

IT IS ANTICIPATED THAT FUTURE DEVELOPMENT WILL BE DIRECTED TOWARD COMPLETION OF THE CADSS SYSTEM ALONG THE FOLLOWING LINES

1. DEVELOPMENT OF A PROGRAM TO REDUCE THE STATE TABLE, ASSIGN STATES TO MEMORY ELEMENT CONFIGURATIONS, REDUCE THE BOOLEAN LOGIC EQUATIONS AND ASSIGN LOGIC FUNCTIONS TO PARTICULAR LOGIC ELEMENTS. THIS WILL RESULT IN A DETAILED LOGIC DIAGRAM FOR A PARTICULAR FAMILY OF LOGIC (PROBABLY PHILCO WDL).

2. DEVELOPMENT OF A PROGRAM TO OUTPUT THE LOGIC DIAGRAM ON THE DTDS HARD COPIER OR THE GERBER PLOTTER.

3. DEVELOPMENT OF A SIMULATOR TO VERIFY THE OPERATION OF THE LOGIC DESIGN. AT THIS PHASE OF THE DESIGN PROCESS MANY TIMING ERRORS CAN BE DETECTED THAT WILL NOT BE FOUND BY SIMULATION OF THE FLOW CHART DESCRIPTION.

4. SINCE THE DESIGNS GENERATED BY THE COMPUTER WILL TEND TOWARD STANDARD LOGIC CONFIGURATIONS, IT WILL BE DESIRABLE TO ALLOW AN EXPERIENCED DESIGNER TO MODIFY THE DESIGN, CORRECTING ERRORS DETECTED BY SIMULATION AND POSSIBLY REDUCING THE TOTAL AMOUNT OF LOGIC. THIS CAN BE DONE BY DISPLAYING THE LOGIC DIAGRAM ON THE DTDS AND ALLOWING DELETION AND INSERTION OF LOGIC ELEMENTS AND CONNECTIONS. THE DESIGNER CAN THEN MODIFY THE DESIGN AS DESIRED AND CHECK EACH MODIFICATION BY SIMULATION.

5. A PROGRAM SHOULD BE DEVELOPED TO ASSIGN LOGIC ELEMENTS TO PARTICULAR TRAY LOCATIONS AND GENERATE A WIRING LIST. IF THE OUTPUT OF THIS PROGRAM IS COMPATABLE WITH THE ADD INPUT FORMAT (ADD SYSTEM USERS MANUAL), THE THE PHILCO ADD SYSTEM CAN BE USED FOR DOCUMENTATION OF THE FINAL SYSTEM.

THE COMPLETE CADSS SYSTEM OUTLINED ABOVE WILL PROVIDE A POWERFUL TOOL FOR THE DESIGN OF SEQUENTIAL SYSTEMS. MUCH OF THE TEDIOUS DOCUMENTATION AND ANALYSIS WILL BE PERFORMED AUTOMATICALLY THUS ALLOWING THE DESIGNER TO SPEND A GREATER PART OF HIS TIME ON CREATIVE WORK.

2 THE FLOW CHART DESCRIPTION LANGUAGE

THE STRUCTURE OF THE INPUT LANGUAGE FOR THE COMPUTER AIDED DIGITAL SYSTEMS SYNTHESIS (CADSS) SYSTEM WAS DEVELOPED WITHOUT REGARD TO LINGUISTIC ASPECTS. THE MOST IMPORTANT FACTORS CONSIDERED DURING THE DEVELOPMENT OF THE LANGUAGE ARE SUMMARIZED BELOW

1. THE CADSS ~~LANGUA~~GE MUST DESCRIBE THE OPERATION OF A DIGITAL SYSTEM AT THE FLOW CHART LEVEL.

2. THE CADSS LANGUAGE MUST BE TAILORED TO THE CONCEPTS AND TERMINOLOGY OF DIGITAL SYSTEMS ENGINEERING DESIGNERS WITHOUT A GREAT DEAL OF PROGRAMMING EXPERIENCE.

3. IN ADDITION TO SPECIFYING THE OPERATION OF A SYSTEM, THE CADSS LANGUAGE MUST SERVE AS THE INPUT TO A SIMULATOR. THIS REQUIRES THE INCLUSION OF SIMULATION CONTROL COMMANDS SUCH AS PRINT, READ, AND PAUSE.

4. SINCE MANY OF THE PROBLEMS IN DIGITAL SYSTEMS DESIGN ARISE FROM INTERACTIONS OF SEPARATE MACHINES, THE LANGUAGE MUST BE CAPABLE OF DESCRIBING TWO OR MORE CONCURRENT INTERACTING SYSTEMS.

THE FLOW CHART DESCRIPTION LANGUAGE PRESENTED HERE IS A MODIFIED FORM OF A PREVIOUS ATTEMPT IN THIS AREA OF RESEARCH (FRANKE AND MERGLER). USE OF THE PREVIOUS DESIGN SYSTEM BROUGHT TO LIGHT SEVERAL DEFICIENCES IN THE ORIGINAL VERSION. FURTHER APPLICATION OF THE CADSS LANGUAGE TO DESIGN PROBLEMS IS NECESSARY TO DETERMINE HOW SUCESSFUL THE MODIFICATIONS WILL BE.

2.1 GENERAL CONVENTIONS

THE INPUT MEDIUM FOR THE CADSS PROGRAMS IS PUNCHED CARDS. ALL 80 COLUMNS OF A CARD MAY CONTAIN INFORMATION. ANY CARD WITH A C IN THE FIRST COLUMN IS A COMMENT CARD. COMMENT CARDS WILL BE PRINTED OUT WITH THE PROGRAM LISTING BUT ARE OTHERWISE IGNORED BY THE PROCESSOR. COMMENT CARDS MAY APPEAR ANYWHERE IN A PROGRAM.

ALL SYNTACTIC UNITS OF THE PROGRAM SUCH AS DECLARATIONS OR STATEMENTS MUST BE TERMINATED BY A SPECIAL TERMINATION CHARACTER ($). STATEMENTS MAY BE CONTINUED FROM ONE CARD TO THE NEXT BUT THE CONTINUATION MUST OCCUR AT A BLANK, I.E.,

NAMES, WORDS, AND NUMBERS MAY NOT BE DIVIDED. SINCE THE END OF A STATEMENT IS MARKED BY A SPECIAL TERMINATION CHARACTER IT IS NOT NECESSARY TO INDICATE CONTINUATION CARDS. WHEN A TERMINATION CHARACTER APPEARS ON A CARD THE REMAINDER OF THE CARD IS NOT PROCESSED. THIS ALLOWS ADDITIONAL COMMENTS TO BE PLACED ON A CARD AFTER THE TERMINATOR, BUT THE NEXT STATEMENT MUST BEGIN ON A NEW CARD.

EXCEPT AS NOTED ABOVE THE CARDS ARE PUNCHED IN FREE-FORMAT FORM. AT LEAST ONE BLANK IS REQUIRED AS A SEPARATOR BETWEEN SYNTACTIC ELEMENTS SUCH AS WORDS OR NUMBERS BUT ANY NUMBER OF BLANKS MAY BE USED.

IN THE DESCRIPTION OF THE LANGUAGE WHICH FOLLOWS THE SLASH (/) IS USED INSTEAD OF LEFT AND RIGHT ANGLE BRACKETS TO DENOTE METALINGUISTIC VARIABLES. SEQUENCES OF CHARACTERS ENCLOSED BETWEEN SLASHES REPRESENT VARIABLES WHOSE VALUES ARE ELEMENTS OF THE CADSS LANGUAGE. FOR EXAMPLE, THE EXPRESSION

/UNIT TYPE/

IS A VARIABLE WHICH MAY HAVE AS ITS VALUE ANY OF THE UNIT TYPES DEFINED IN THE CADSS LANGUAGE.

2.2 UNITS AND DECLARATIONS

THE UNITS OF A CADSS DESCRIPTION CORRESPOND TO HARDWARE UNITS SUCH AS MEMORY ELEMENTS OR TO FUNCTIONALLY CONNECTED GROUPS OF MEMORY ELEMENTS SUCH AS COUNTERS OR REGISTERS. ALL UNITS OF A SYSTEM MUST BE DECLARED IN THE FIRST SECTION OF THE CADSS PROGRAM. THE FORMAT OF THE DECLARATION IS

/UNIT TYPE/. NAME1, NAME2, . . . NAMEN $

THE DEFINED UNIT TYPES ARE

1. INPUT - INPUTS ARE THE ONLY UNITS WHICH MAY BE CHANGED FROM OUTSIDE THE SYSTEM. THE SYSTEM IS NOT ALLOWED TO CHANGE THE VALUE OF AN INPUT.

2. OUTPUT - OUTPUTS ARE TREATED AS MEMORY ELEMENTS SUPPLYING INFORMATION FROM THE SYSTEM.

3. REGIST - REGISTERS ARE GROUPS OF MEMORY ELEMENTS THAT MAY BE SHIFTED RIGHT OR LEFT, SET TO ANY DESIRED VALUE, OR INCREMENTED OR DECREMENTED IN BINARY SEQUENCE.

4. DELAY - DELAYS ARE MEMORY ELEMENTS THAT MAY BE SET BY AN INPUT AND WILL BE RESET AUTOMATICALLY AFTER A SPECIFIED TIME INTERVAL. THE TIME INTERVAL (IN TENTHS OF MICROSECONDS) IS SPECIFIED AS A SUBSCRIPT.

THE UNIT TYPE MUST BE FOLLOWED BY A COLON OR PERIOD.

FOLLOWING THE COLON IS A LIST OF NAMES THAT ARE TO BE DECLARED AS A PARTICULAR TYPE. A NAME CONSISTS OF A SEQUENCE OF ALPHABETIC AND/OR NUMERIC CHARACTERS BEGINNING WITH AN ALPHABETIC. NAMES MAY BE OF ANY LENGTH BUT THE FIRST SIX CHARACTERS OF ANY TWO NAMES MAY NOT BE THE SAME.

IF THE UNIT DECLARED CONSISTS OF MORE THAN ONE MEMORY ELEMENT (OR IF IT CONSISTS OF SEVERAL INPUT/OUTPUT LINES CONSIDERED AS ONE UNIT) THE NAME IS SUBSCRIPTED WITH THE NUMBER OF MEMORY ELEMENTS IN THE UNIT. IN THE CASE OF COUNTERS AND REGISTERS THE LEAST SIGNIFICANT BIT IS BIT 1 AND IT IS LOCATED IN THE LEFTMOST MEMORY ELEMENT OF THE UNIT. THE NAMES IN A DECLARATION LIST ARE SEPARATED BY COMMAS AND THE LIST IS TERMINATED BY A $.

EXAMPLES

```
INPUT. START, CLEARA, DATAINPUT(8) $
OUTPUT. READY, DATAOUT(10) $
REGIST. BUFFER(10), BUSY, STATE(3) $
```

ALL DECLARATIONS MUST APPEAR IN THE HEADING OF THE CADSS FLOW CHART DESCRIPTION.

2.3 CONDITION LISTS

THE BASIC ELEMENT OF A CADSS PROGRAM IS THE STATEMENT WHICH IS COMPOSED OF A CONDITION LIST AND AN OPERATION LIST. THE TWO LISTS WILL BE DISCUSSED FIRST AND THEN THE USE OF THE LISTS IN THE FORMATION OF STATEMENTS WILL BE CONSIDERED.

THE CONDITION LIST IS A WELL FORMED BOOLEAN EXPRESSION OF DEFINED UNITS OF THE SYSTEM, BOOLEAN LITERALS, AND THE FOLLOWING OPERATORS

1.	-	LOGICAL NOT
2.	+	INCLUSIVE OR
3.	*	LOGICAL AND
4.	=	LOGICAL EQUALITY
5.	'	TRANSITION TO TRUE
6.	'-	TRANSITION TO FALSE

THE DEFINED UNITS OF THE SYSTEM MAY BE USED AS OPERANDS IN THREE WAYS

1. IF THE ENTIRE UNIT IS TO BE USED AS THE OPERAND THE UNSUBSCRIPTED NAME IS USED

   BUFFER

2. AN OPERAND CONSISTING OF A SINGLE BIT OF A UNIT IS INDICATED BY THE NAME WITH A SINGLE SUBSCRIPT SUCH AS

   BUFFER(1)

3. IF A SECTION (SEVERAL ADJACENT BITS) OF A UNIT IS TO BE USED AS A SUBSCRIPT THIS IS INDICATED BY AN EXTENDED SUBSCRIPT. FOR EXAMPLE, THE FIRST FOUR BITS OF BUFFER WOULD BE REFERRED TO AS

   BUFFER(1-4)

THE LOGICAL OPERATORS (-,+,*) ARE USED IN THE CONVENTIONAL MANNER. EACH OPERATOR REQUIRES TWO OPERANDS OF ONE BIT EACH. THE LOGICAL EQUALITY (=) OPERATOR REQUIRES TWO OPERANDS WHICH MAY BE EITHER BOOLEAN LITERALS OR DEFINED UNITS OF THE SYSTEM. THE DIMENSIONS OF BOTH OPERANDS MUST BE THE SAME. THE VALUE OF AN EQUALITY OPERATION ON TWO OPERANDS IS TRUE WHEN ALL THE CORRESPONDING BITS OF THE OPERANDS ARE EQUAL. FOR EXAMPLE, IF THE FIVE BIT REGISTER A HAS BEEN SET,

(A = 00000)

HAS THE LOGICAL VALUE TRUE. AS AN ADDITIONAL EXAMPLE, IF C AND D ARE ONE BIT UNITS, THEN THE EXPRESSION

-(C = D)

IS EQUIVALENT TO THE EXCLUSIVE OR OPERATION ON C AND D.

THE TRANSITIONAL OPERATORS (' AND '-) ARE SPECIAL UNARY OPERATORS WHICH ARE TRUE WHEN THE SINGLE ONE BIT OPERAND CHANGES VALUE. THE USE OF THE TRANSITION OPERATORS IS ILLUSTRATED IN THE FOLLOWING EXAMPLES

('CLEAR)	TRUE WHEN CLEAR CHANGES FROM FALSE TO TRUE
('-BUFFER(3))	TRUE WHEN THE THIRD BIT OF BUFFER CHANGES FROM TRUE TO FALSE

THE NORMAL ORDER IN WHICH OPERATIONS ARE PERFORMED IN A CONDITION LIST IS

1. = EQUALITY OPERATION
2. ' TRANSITION TO TRUE OPERATION
3. '- TRANSITION TO FALSE OPERATION
4. - NEGATION OPERATION
5. * AND OPERATION
6. + OR OPERATION

THE NORMAL PRECEDENCE ORDERING OF OPERATIONS MAY BE MODIFIED BY PARENTHESIS AS DESIRED.

THE REQUIREMENT THAT ALL CONDITION LISTS BE WELL-FORMED BOOLEN EXPRESSIONS MEANS ESSENTIALLY THAT EACH OPERATOR MUST BE ASSOCIATED WITH THE CORRECT NUMBER OF OPERANDS IN THE CORRECT ORDER. THIS IS ILLUSTRATED BY THE FOLLOWING EXAMPLES USING THE UNITS

```
INPUT. SWITCH(4) $
OUTPUT. A, B, C, D $
REGIST. AREG(10), BREG(4) $
```

EXAMPLES OF CONDITION LISTS

```
A*B*(C+D)
SWITCH(1) * (BREG = 1001)
'SWITCH(4) * -AREG(5)
A*(B+(C*SWITCH(1))+(ACTR(1)=BREG(1))+('SWITCH(1)
     *(AREG(1)+BREG(1))))
```

2.4 OPERATION LISTS

THE OPERATION LIST IS A LIST OF COMMANDS SPECIFYING THE OPERATION OF THE SYSTEM DURING SOME INCREMENT OF TIME. THREE TYPES OF COMMANDS ARE DEFINED IN THE CADSS INPUT LANGUAGE. THE FIRST TYPE IS THE UNIT CONTROL COMMAND WHICH SPECIFIES THE OPERATION OF A UNIT OF THE SYSTEM. THE DEFINED UNIT CONTROL COMMANDS ARE DESCRIBED BELOW

1. SET /UNIT NAME/
   THIS SPECIFIES THAT THE UNIT NAMED IS TO BE SET TO TRUE (ALL ONES).

2. RESET /UNIT NAME/
   THIS SPECIFIES THAT THE UNIT NAMED IS TO BE RESET TO A LOGICAL FALSE VALUE (ALL ZEROS).

3. RSHIFT /REGISTER NAME/ ADDING /BIT NAME/
   THIS SPECIFIES THAT THE REGISTER NAMED IS TO BE SHIFTED RIGHT ONE PLACE. THE VALUE OF THE BIT NAMED IS SHIFTED INTO THE LEFTMOST POSITION OF THE REGISTER.

4. LSHIFT /REGISTER NAME/ ADDING /BIT NAME/
   THIS SPECIFIES THAT THE REGISTER NAMED IS TO BE SHIFTED LEFT ONE POSITION. THE VALUE OF THE BIT NAMED IS PLACED IN THE RIGHTMOST POSITION OF THE REGISTER.

5. INCREMENT /REGISTER NAME/
   THIS SPECIFIES THAT THE REGISTER NAMED IS TO BE INCREMENTED IN BINARY SEQUENCE.

6. DECREMENT /REGISTER NAME/
   THIS SPECIFIES THAT THE NAMED REGISTER IS TO BE DECREMENTED IN BINARY SEQUENCE.

7. TRANSFER /NAME1/ TO /NAME2/
   THIS SPECIFIES THAT THE CONTENTS OF /NAME1/ ARE TO BE TRANSFERRED TO /NAME2/. THE DIMENSIONS OF THE NAMED UNITS MUST BE THE SAME.

8. TRANSFER /BINARY LITERAL/ TO /NAME/
   THIS SPECIFIES THAT THE VALUE OF THE UNIT NAMED IS TO BE SET TO THE BINARY LITERAL. THE DIMENSIONS OF THE OPERANDS MUST BE THE SAME.

ANY COMMAND WHICH ATTEMPTS TO CHANGE THE VALUE OF AN INPUT IS ILLEGAL. NAMES USED IN THE COMMANDS MAY BE EITHER AN UNSUBSCRIPTED NAME INDICATING A COMPLETE UNIT, A NAME WITH A SINGLE SUBSCRIPT INDICATING A SINGLE BIT OF A UNIT OR A NAME WITH AN EXTENDED SUBSCRIPT INDICATING A SECTION OF A UNIT.

THE SECOND TYPE OF COMMAND DEFINED IS USED FOR PROGRAM SEQUENCE CONTROL. THE SINGLE COMMAND OF THIS TYPE IS

GO TO /LABEL/

THIS SPECIFIES THAT THE NORMAL SEQUENCE OF SYSTEM OPERATIONS IS TO BE INTERRUPTED AND TRANSFERRED TO THE POINT IN THE PROGRAM INDICATED BY THE LABEL (SEE SECTION 2.6).

THE THIRD TYPE OF COMMAND DEFINED IS USED FOR SIMULATION CONTROL. SINCE THE SYSTEM IS TO BE SIMULATED IT IS NECESSARY FOR THE USER TO SPECIFY WHEN INPUTS ARE TO BE CHANGED AND WHEN THE CONTENTS OF SPECIFIED UNITS ARE TO BE PRINTED OUT. THE SIMULATION CONTROL COMMANDS DEFINED ARE

1. ACCEPT
   THIS SPECIFIES THAT ONE DATA CARD IS TO BE READ AND THE INFORMATION ON THE CARD TRANSFERRED TO THE UNITS DECLARED AS INPUTS. THE BINARY NUMBER IN THE FIRST COLUMN OF THE CARD IS TRANSFERRED TO THE FIRST BIT DECLARED AS INPUT, THE NUMBER IN THE SECOND COLUMN IS TRANSFERRED TO THE SECOND BIT, ETC. IF ANY COLUMN OF THE CARD IS BLANK THE CORRESPONDING INPUT IS NOT CHANGED.

2. DISPLAY
   THIS SPECIFIES THAT THE CONTENTS OF THE UNITS NAMED IN THE PRINT LIST ARE TO BE PRINTED. THE PRINT LIST IS A LIST OF UNIT NAMES ENTERED DURING SIMULATOR INITIALIZATION WHICH MAY BE CHANGED WITHOUT RECOMPILING THE PROGRAM.

3. PAUSE
   THIS SPECIFIES THAT THE SIMULATION PROCESS IS TO STOP UNTILL THE COMPUTER RUN SWITCH IS PRESSED. IN ADDITION TO NORMAL TERMINATION OF A SIMULATION THIS MAY BE USED TO GIVE THE OPERATOR TIME TO OBSERVE THE CURRENT CONDITION OF THE SYSTEM AND PREPARE DATA CARDS ACCORDINGLY. THIS IS PARTICULARILY USEFUL WHEN THE SIMULATED SYSTEM IS TO INTERACT WITH AN OPERATOR, PROCESS, OR MACHINE.

AN OPERATION LIST IS CONSTRUCTED BY FORMING A SEQUENCE OF COMMANDS SEPARATED BY COMMAS. IF MORE THAN ONE COMMAND APPEARS IN A LIST THE ORDER IN WHICH THE COMMANDS ARE EXECUTED IS

1. ALL UNIT CONTROL COMMANDS (SIMULTANEOUSLY)
2. ACCEPT COMMAND
3. DISPLAY COMMAND
4. PAUSE COMMAND
5. SEQUENCE CONTROL (GO TO) COMMAND

IN ORDER TO ILLUSTRATE THE FORMATION OF OPERATION LISTS ASSUME THAT THE FOLLOWING UNITS ARE DEFINED

```
INPUT. SWITCH(4) $
OUTPUT. A, B, C, D $
REGIST. AREG(10), BREG(4) $
```

THEN EXAMPLES OF OPERATION LISTS ARE

SET A, TRANSFER AREG(1-4) TO BREG, TRANSFER BREG TO AREG(1-4), ACCEPT, DISPLAY

IN THIS CASE A IS SET TRUE, THE CONTENTS OF THE FIRST FOUR ELEMENTS OF AREG ARE INTERCHANGED WITH THE CONTENTS OF BREG, NEW VALUES ARE READ INTO THE FOUR ELEMENTS OF SWITCH, AND THE CONTENTS OF THE UNITS SPECIFIED BY THE PRINT LIST ARE PRINTED.

LSHIFT AREG ADDING AREG(1), INCREMENT BREG, RESET A

HERE AREG IS CIRCULATED LEFT, BREG IS INCREMENTED IN BINARY SEQUENCE AND A IS RESET.

PAUSE, GO TO L1, SET B, RESET AREG, DISPLAY

IN THIS EXAMPLE B IS SET AND ALL 10 BITS OF AREG ARE RESET. THE CONTENTS OF THE PRINT LIST ARE THEN PRINTED DURING SIMULATION AND THE COMPUTER SIMULATION PROCESS STOPS. WHEN THE RUN SWITCH IS PRESSED SIMULATION WILL RESUME AT THE POINT IN THE PROGRAM INDICATED BY THE LABEL L1. THIS ILLUSTRATES THAT THE ORDER IN WHICH COMMANDS ARE EXECUTED IS INDEPENDANT OF THEIR ORDER IN THE LIST.

2.5 STATEMENTS

THE OPERATION AND CONDITION LISTS MAY BE COMBINED TO FORM THREE TYPES OF STATEMENTS. THE FORMAT OF THE UNCONDITIONAL TYPE OF STATEMENT IS

THEN /OPERATION LIST/

THIS SPECIFIES OPERATIONS THAT ARE TO BE PERFORMED REGARDLESS OF CONDITIONS EXISTING IN THE SYSTEM.

IF THE PERFORMANCE OF CERTAIN OPERATIONS IS DEPENDANT ON THE CONDITION OF THE SYSTEM, A CONDITIONAL STATEMENT MAY BE USED

WHEN /CONDITION LIST/ THEN /OPERATION LIST/ $

IN THIS CASE THE COMMANDS IN THE OPERATION LIST ARE EXECUTED ONLY WHEN THE BOOLEAN CONDITION LIST IS TRUE.

IN MANY CASES A SYSTEM IS TO PERFORM EITHER OF TWO SETS OF OPERATIONS WITH THE PARTICULAR SET BEING DETERMINED BY CONDITIONS EXISTING IN THE SYSTEM. THIS SITUATION IS DESCRIBED BY AN OPTION STATEMENT

```
WHEN /CONDITION LIST/ THEN /OPERATION LIST/
     ELSE /OPERATION LIST/ $
```

THUS IF THE CONDITION LIST IS TRUE THE COMMANDS IN THE FIRST OPERATION LIST ARE EXECUTED. IF THE CONDITION LIST IS FALSE THE COMMANDS IN THE SECOND OPERATION LIST ARE EXECUTED.

EXAMPLES OF STATEMENTS (USING UNITS DEFINED FOR PREVIOUS EXAMPLES)

```
WHEN SWITCH(1) THEN RSHIFT AREG ADDING 0, RESET A $
WHEN -A=D*'-AREG(5) THEN RESET BREG,TRANSFER 1001100100
     TO AREG, SET B, DISPLAY $
WHEN SWITCH(2) THEN RESET AREG, GO TO L1 ELSE
     INCREMENT AREG $
THEN RESET A, RESET B, SET C, ACCEPT, DISPLAY, PAUSE $
THEN TRANSFER AREG(1-4) TO BREG $
```

2.6 LABELS

A LABEL CONSISTS OF A SEQUENCE OF ALPHABETIC AND NUMERIC CHARACTERS BEGINNING WITH AN ALPHABETIC. LABELS MAY BE OF ANY LENGTH BUT THE FIRST SIX CHARACTERS MUST FORM A UNIQUE LABEL. EITHER THE COLON OR PERIOD IS USED AS A LABEL SEPARATOR. A LABEL MAY BE ATTACHED TO ANY STATEMENT WITH THE FORMAT

```
/LABEL/. /STATEMENT/
```

LABELS MAY BE USED AS SUBTITLES TO ENHANCE THE MEANING OF A PROGRAM. ANY STATEMENT WHICH IS REFERRED TO BY A SEQUENCE CONTROL COMMAND (GO TO) MUST BE LABELED. IN ADDITION ANY STATEMENT WHICH IS TO BE THE INITIAL STATEMENT IN THE DESCRIPTION OF A SYSTEM MUST BE LABELED SO THAT IT MAY BE REFERRED TO DURING SIMULATOR INITIALIZATION.

2.7 PROGRAM BLOCK STRUCTURE

THE STATEMENTS OF A CADSS PROGRAM CORRESPOND ROUGHLY TO THE BLOCKS MAKING UP THE FLOW CHART OF THE SYSTEM BEING DESCRIBED. IN THE CASE OF A FLOW CHART DESCRIPTION OF A SYSTEM THE SEQUENCE OF OPERATIONS FOLLOWS FLOW PATHS FROM BLOCK TO BLOCK SO THAT AT A GIVEN TIME ONLY THE OPERATIONS

OF ONE BLOCK MAY OCCUR. THIS BLOCK MAY BE SAID TO BE ACTIVE AT THAT PARTICULAR TIME WHILE ALL OTHERS ARE INACTIVE.

IN THE SAME WAY THE SEQUENCE OF OPERATIONS IN ANCADSS PROGRAM FOLLOWS PATHS DEFINED BY THE STRUCTURE OF THE PROGRAM. AT A GIVEN TIME ONE (OR PREHAPS SEVERAL) OF THE STATEMENTS ARE PERMITTED TO PERFORM OPERATIONS. THE STATEMENTS THAT MAY PERFORM OPERATIONS AT A PARTICULAR TIME ARE SAID TO BE POTENTIALLY ACTIVE AT THAT TIME. THE ORDER IN WHICH STATEMENTS BECOME POTENTIALLY ACTIVE IS DEFINED BY THE BLOCK STRUCTURE OF THE PROGRAM AND THE SEQUENCE CONTROL COMMANDS.

FOR PURPOSES OF DESCRIPTION IT IS USEFUL TO CONSIDER THE STATEMENT OR STATEMENTS WHICH ARE POTENTIALLY ACTIVE AS BEING IN CONTROL OF THE SYSTEM. 'CONTROL' MAY BE CONSIDERED AS A POINTER (OR POINTERS) WHICH MOVES THROUGH THE PROGRAM, ALWAYS SPECIFYING THE POTENTIALLY ACTIVE STATEMENT (OR STATEMENTS). THE TRANSFER OF CONTROL COMMAND (GO TO) MAY BE INTERPETED AS SPECIFING THE LABEL OF THE NEXT POTENTIALLY ACTIVE STATEMENT.

A BLOCK OF A CADSS PROGRAM CONSISTS OF ONE STATEMENT WHICH MAY BE LABELED FOLLOWED BY ANY NUMBER OF UNLABELED STATEMENTS. SINCE THREE DIFFERENT TYPES OF BLOCKS ARE DEFINED IT IS NECESSARY TO INDICATE A CHANGE IN BLOCK TYPE BY MEANS OF A BLOCK TYPE SPECIFICATION. THE FORMAT OF A BLOCK IS

```
/BLOCK TYPE/.  /LABEL/.  /STATEMENT/  $
     /STATEMENT/ $
     /STATEMENT/ $
     /STATEMENT/ $
      .
      .
      .
```

THE BLOCK TYPE SPECIFICATION MAY BE OMITTED IF THE TYPE IS THE SAME AS THAT OF THE PRECEEDING BLOCK. IF THE TYPE SPECIFICATION IS PRESENT THE STATEMENT LABEL MAY BE OMITTED.

THE END OF A BLOCK (AND THE BEGINNING OF THE FOLLOWING BLOCK) IS INDICATED BY A BLOCK TYPE SPECIFICATION, A STATEMENT LABEL, OR BOTH. ONE OBVIOUS RESULT OF THIS METHOD OF SPECIFING BLOCKS IS THAT NO BLOCK MAY CONTAIN ANOTHER.

2.7.1 BOOLEAN BLOCKS

THE BLOCK TYPE SPECIFICATION FOR A BOOLEAN BLOCK IS

BOOLEAN.

THE ONLY STATEMENTS PERMITTED IN A BOOLEAN BLOCK ARE BOOLEAN STATEMENTS HAVING THE FORMAT

/UNIT NAME/ = /CONDITION LIST/ $

THE UNIT NAMED ON THE LEFT OF THE EQUAL SIGN IS TRUE WHENEVER THE CONDITION LIST IS TRUE AND FALSE WHENEVER THE LIST IS FALSE. BOOLEAN STATEMENTS MAY BE USED TO DEFINE SIGNALS TO BE USED IN THE SYSTEM OR TO DEFINE CONNECTIONS BETWEEN REGISTERS AND OUTPUTS.

EXAMPLES OF BOOLEAN STATEMENTS

```
      INPUT. SWITCH(4) $
      OUTPUT. A,B,C,D,BOUT(4) $
      REGIST. AREG(10),BREG(4) $
      BOOLEAN.
            BOUT = BREG $
C     THIS SPECIFIES THAT THE OUTPUT BUS BOUT IS
C     CONNECTED TO THE REGISTER BREG.
            B = AREG(1)*AREG(2)*SWITCH(1) $
C     THE OUTPUT LINE B IS FORMED BY ANDING THE
C     FIRST TWO BITS OF AREG AND THE FIRST BIT
C     OF SWITCH.
            PARITY = -BREG(1)-BREG(2)-BREG(3)-
               BREG(5) $
C     THIS DEFINES A LINE CALLED PARITY WHICH IS
C     1 WHEN THE NUMBER OF 1'S IN BREG IS ODD.
```

ONLY OUTPUTS OR PREVIOUSLY UNDEFINED UNITS (OF ONE BIT) MAY APPEAR ON THE LEFT OF THE EQUAL SIGN OF A BOOLEAN STATEMENT.

2.7.2 SEQUENCED BLOCKS

THE BLOCK TYPE SPECIFICATION FOR A SEQUENCED BLOCK IS

SEQUENCED.

ONLY ONE STATEMENT IN A SEQUENCED BLOCK MAY BE POTENTIALLY ACTIVE AT A GIVEN TIME. IF THE CONDITION LIST OF A POTENTIALLY ACTIVE STATEMENT IS TRUE THE STATEMENT BECOMES ACTIVE AND THE COMMANDS IN THE OPERATION LIST ARE EXECUTED.

IF THE CONDITION LIST IS FALSE THE STATEMENT REMAINS POTENTIALLY ACTIVE. AFTER THE OPERATION LIST OF A STATEMENT IS EXECUTED, THE STATEMENT BECOMES INACTIVE AND THE NEXT STATEMENT IN THE BLOCK BECOMES POTENTIALLY ACTIVE.

THUS IN THE ABSENCE OF ANY TRANSFER OF CONTROL COMMANDS STATEMENTS BECOME POTENTIALLY ACTIVE IN SEQUENCE. WHEN THE LAST STATEMENT OF A SEQUENCED BLOCK IS REACHED, CONTROL PASSES TO THE NEXT BLOCK. THIS NORMAL SEQUENCE OF CONTROL WILL BE INTERRUPTED WHEN AN OPERATION LIST CONTAINING A TRANSFER OF CONTROL COMMAND IS EXECUTED.

2.7.3 CONCURRENT BLOCKS

THE BLOCK TYPE SPECIFICATION FOR A CONCURRENT BLOCK IS

CONCURRENT.

WHEN CONTROL IS TRANSFERRED TO A CONCURRENT BLOCK ALL STATEMENTS IN THE BLOCK BECOME POTENTIALLY ACTIVE. THE CONDITION LISTS OF ALL POTENTIALLY ACTIVE STATEMENTS ARE THEN CHECKED TO DETERMINE WHICH OF THE STATEMENTS ARE TO BECOME ACTIVE. THE COMMANDS IN THE OPERATION LISTS OF THE ACTIVE STATEMENTS ARE EXECUTED CONCURRENTLY. IF A TRANSFER OF CONTROL COMMAND WAS NOT EXECUTED ALL STATEMENTS IN THE BLOCK BECOME POTENTIALLY ACTIVE AND THE PROCESS IS REPEATED. CONTROL REMAINS AT A CONCURRENT BLOCK UNTIL A TRANSFER OF CONTROL COMMAND IS EXECUTED.

2.8 SERIAL BINARY MULTIPLIER EXAMPLE

A BINARY MULTIPLIER WAS SELECTED TO PROVIDE A SIMPLE EXAMPLE OF THE CADSS LANGUAGE. THE MULTIPLIER IS TO BE DESIGNED TO ACCEPT TWO SIGNED TEN BIT NUMBERS AND A START SIGNAL. IT IS THEN TO PERFORM THE MULTIPLICATION AND TRANSMIT A FINISH SIGNAL ON COMPLETION. THE MULTIPLICATION IS TO BE PERFORMED BY REPEATED SERIAL ADDITION SO TWO INTERNAL COUNTERS ARE NECESSARY FOR CONTROLLING THE OPERATIONS. A BLOCK DIAGRAM OF THE MULTIPLIER IS SHOWN IN FIGURE 2.1.

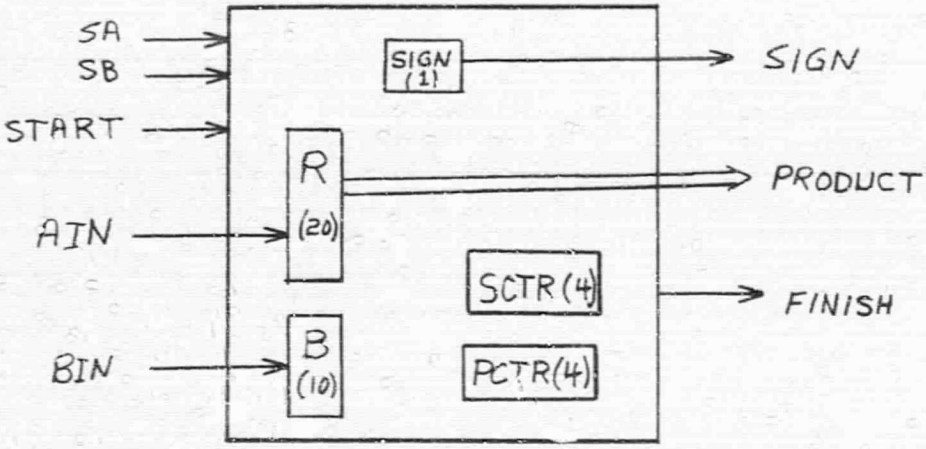

FIGURE 2.1 MULTIPLIER BLOCK DIAGRAM

THE FLOW CHART DESCRIBING THE MULTIPLIER OPERATION IS PRODUCED BY CONSIDERING THE SEQUENCE OF OPERATIONS THAT MUST BE PERFORMED. UPON RECEIVING A START SIGNAL THE FOLLOWING INITIALIZATION OPERATIONS MUST OCCUR.

1. RESET FINISH SIGNAL
2. ACCEPT SIGNALS ON INPUT LINES
   AIN TO R(11-20) REGISTER
   BIN TO B REGISTER
3. CLEAR UNUSED PART OF RESULT (R REGISTER)
4. SET SIGN BIT TO SIGN OF RESULT
5. INITIALIZE PRODUCT COUNTER TO 10

THESE OPERATIONS MAY OCCUR CONCURRENTLY AND SO ARE PLACED IN ONE BLOCK OF THE FLOW CHART.

IT IS THEN NECESSARY TO INITIALIZE THE ADD OPERATION BY CLEARING THE CARRY BIT AND DECREMENTING THE PRODUCT COUNTER. THE ADD OPERATION IS PERFORMED ONLY IF BIT 11 OF THE RESULT REGISTER (THE LEAST SIGNIFICANT BIT OF THE MULTIPLIER) IS 1. IF AN ADDITION IS NECESSARY THE SHIFT COUNTER IS FIRST SET TO 10 AND THE SUM AND CARRY ARE FORMED. THE R REGISTER IS SHIFTED. THE SUM IS PLACED IN THE MOST SIGNIFICANT BIT AND THE SHIFT COUNTER IS DECREMENTED. THIS PROCESS IS REPEATED

UNTIL THE SHIFT COUNTER IS ZERO. FOLLOWING EACH ADDITION IT IS NECESSARY TO RESTORE THE R REGISTER BY 10 ADDITIONAL SHIFTS.

AFTER EACH ADDITION (OR SKIPPED ADDITION) THE R REGISTER IS SHIFTED ONE BIT, THE PRODUCT COUNTER IS DECREMENTED, AND THE R(11) BIT EXAMINED TO DETERMINE IF AN ADDITION IS TO BE PERFORMED. UPON COMPLETION (PCTR = 0) THE FINISH SIGNAL IS SET AND THE MULTI .IER RETURNS TO WAIT FOR THE NEXT INPUT. THE FLOW CHART PR DUCED IN ACCORDANCE WITH THIS DESCRIPTION IS SHOWN IN FIGURE 2.2.

THE CADSS DESCRIPTION OF THE MULTIPLIER (FIGURE 2.3) IS WRITTEN DIRECTLY FROM THE FLOW CHART. FIRST ALL UNITS OF THE SYSTEM ARE DEFINED AS INPUTS, OUTPUTS, OR REGISTERS. THE COMBINATORIAL LOGIC USED TO GENERATE THE SUM, CARRY AND SIGN AS WELL AS THE OUTPUT DEFINITION IS LISTED IN A BOOLEAN BLOCK. THE OPERATION OF THE SYSTEM IS THEN DESCRIBED IN SEQUENCED OR CONCURRENT BLOCKS.

$$\text{SUM} = R(1) \oplus B(1) \oplus \text{CARRY}$$
$$N_{\text{CARRY}} = R(1)\cdot B(1) + R(1)\cdot \text{CARRY} + B(1)\cdot \text{CARRY}$$

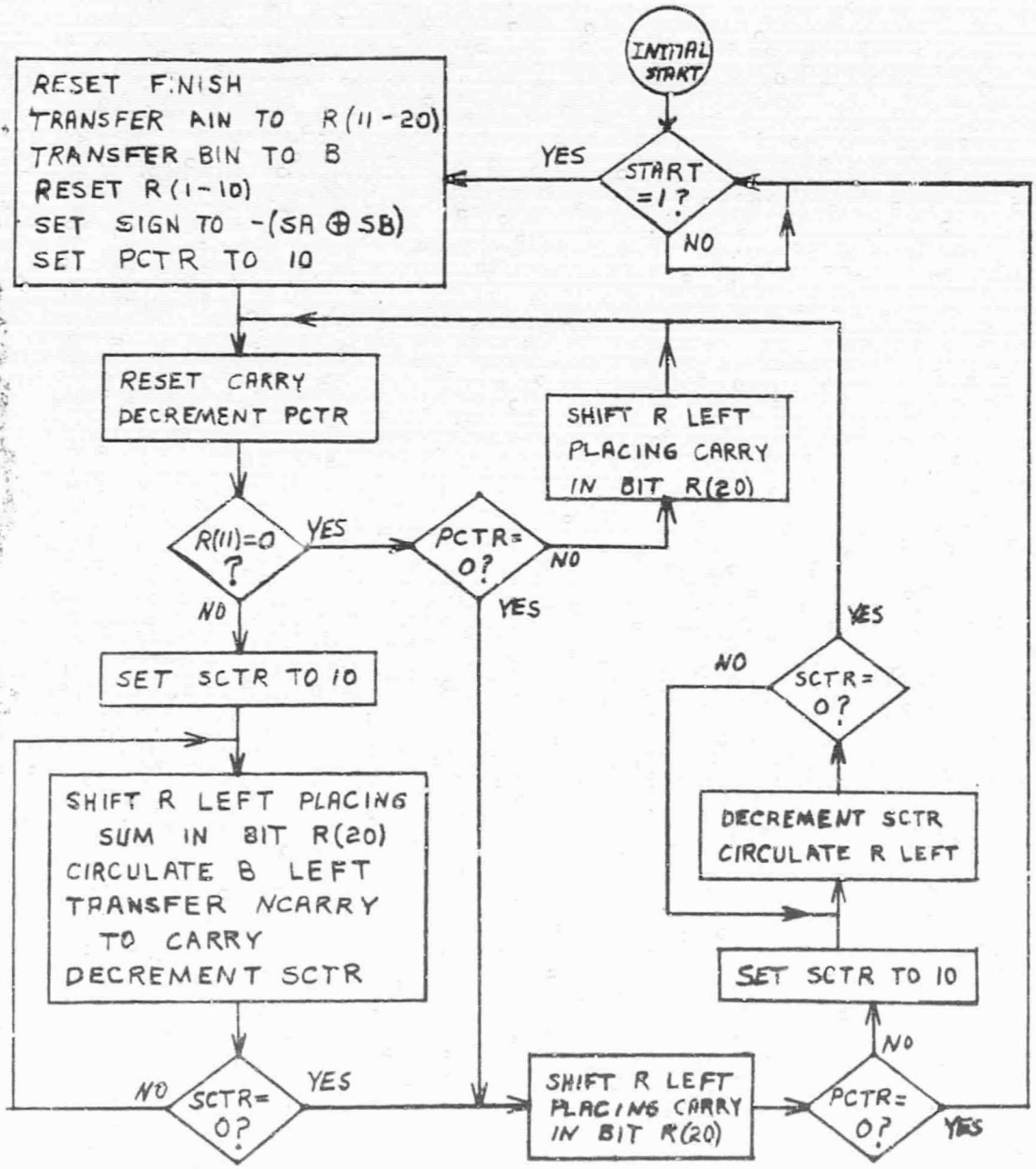

FIGURE 2.2 MULTIPLIER FLOW CHART

```
      OUTPUT. FINISH, SIGN, PRODUCT(20) $
      INPUT. START, SA, SB, AIN, BIN $
      REGIST. R(20), B(10)8 SCTR(4), PCTR(4), CARRY $

      BOOLEAN.
           SUM = (R(1)+B(1)*-CARRY + (R(1)+CARRY)*-B(1) +
                 (B(1)+CARRY)*-R(1) $
C     ALTERNATIVLY THIS COULD BE WRITTEN
           SUM = R(1)=B(1)=CARRY $
           NCARRY = R(1)*B(1) + R(1)*CARRY + B(1)*CARRY $
           ISGN = (SA*SB)+(-SA*-SB) $
           PRODUCT = R $
C     INITIALIZATION OCCURS WHEN A START SIGNAL IS RECEIVED.

      SEQUENCED. STARTPOINT.
           WHEN 'START THEN RESET FINISH, RESET R(1-10),
                 TRANSFER AIN TO A(11-20), TRANSFER BIN TO
                 B, TRANSFER ISGN TO SIGN, TRANSFER 0101
                 TO PCTR $
C     SET UP THE ADDITION AND CHECK TO SEE IF AN ADDITION
C     IS TO BE PERFORMED.

      CONCURRENT. SKIPCHECK.
           THEN RESET CARRY, DECREMENT PCTR $
           WHEN -R(11) THEN GO TO SKIP ELSE TRANSFER
                 0101 TO SCTR, GO TO ADD $
        ADD.  WHEN SCTR = 0000 THEN GO TO SHIFT ELSE
              LSHIFT R ADDING SUM, TRANSFER NCARRY TO
              CARRY, DECREMENT SCTR, LSHIFT B ADDING B(1) $
C     AFTER SKIPPING AN ADDITION IT IS NECESSARY TO CHECK
C     THE PRODUCT COUNTER FOR COMPLETION.

      SEQUENCED. SKIP.
           WHEN -PCTR = 0000 THEN LSHIFT R ADDING CARRY,
                 GO TO SKIPCHECK ELSE GO TO SHIFT $
        SHIFT.  THEN LSHIFT R ADDING CARRY $
           WHEN PCTR = 0000 THEN SET FINISH, GO TO
                 STARTPOINT ELSE TRANSFER 0101 TO SCTR $
C     THE R REGISTER MUST BE RESTORED TO ITS CORRECT POSITION
C     AFTER EACH ADDITION.

      CONCURRENT. RESTORE.
           WHEN SCTR = 0000 THEN GO TO SKIPCHECK ELSE LSHIFT
                 R ADDING R(1), DECREMENT SCTR $
```

FIGURE 2.3 CADSS DESCRIPTION OF MULTIPLIER

3 THE CADSS SYSTEM IMPLEMENTATION

IN GENERAL METHODS FOR PERFORMING THE DESIGN OF DIGITAL SYSTEMS MAY BE SEPARATED INTO TWO GROUPS

1. OPTIMUM METHODS USING FUNDAMENTAL MATHEMATICAL RELATIONS TO PRODUCE DESIGNS HAVING THE LEAST POSSIBLE COMPONENT COST.

2. HEURISTIC METHODS USING APPROXIMATIONS TO PRODUCE DESIGNS WHICH WILL BE COMPRABLE IN COST TO THOSE OF EXPERIENCED HUMAN DESIGNERS.

IN MANY CASES OPTIMUM METHODS ARE NOT KNOWN AND WHEN THEY ARE KNOWN THEY REQUIRE A MUCH GREATER AMOUNT OF DESIGN TIME THAN HEURISTIC METHODS. IN ADDITION HEURISTIC DESIGNS, BEING SOMEWHAT REDUNDANT, ARE USUALLY EASIER TO CHECK OUT AND MAINTAIN AND ARE MORE RELIABLE. WHEN A LIMITED NUMBER OF SYSTEMS ARE TO BE CONSTRUCTED THE COMPONENT SAVING OF A MINIMAL DESIGN IS USUALLY NOT SUFFICIENT TO JUSTIFY THE INCREASED DESIGN AND CHECK OUT TIME. FOR THESE REASONS THE CADSS SYSTEM IS TO USE HEURISTIC DESIGN TECHNIQUES. IN ORDER TO UTILIZE THE DESIGN EXPERIENCE OF THE ENGINEERING DESIGNER THE CADSS SYSTEM WILL PRODUCE MEANINGFUL DESCRIPTIONS OF THE SYSTEM AS THE DESIGN PROGRESSES. THE DESIGNER MAY THEN ANALYZE THESE DESCRIPTIONS OF THE PARTIALLY DESIGNED SYSTEM AND MODIFY THEM AS DESIRED.

IN ORDER TO PROVIDE AS MUCH MACHINE INDEPENDANCE AS POSSBILE THE PROGRAMS MAKING UP THE CADSS SYSTEM ARE TO BE WRITTEN IN FORTRAN IV. SOME FUNCTIONS SUCH AS WORD PACKING AND USE OF AUXILIARY STORAGE WILL BE MACHINE DEPENDANT. THESE WILL BE WRITTEN AS SUBROUTINES TO FACILITATE CONVERSION. IN ORDER TO ACHIEVE MAXIMUM UTILIZATION INPUT/OUTPUT ROUTINES ARE TO BE PROVIDED FOR CARDS, LINE PRINTERS, GRAPHIC TERMINALS AND X-Y PLOTTERS.

THE COMPLETE CADSS SYSTEM WILL CONSIST OF 7 PROGRAMS LINKED BY DATA STORAGE ON AUXILIARY MEMORY.

3.1 COMPILATION

THE PURPOSE OF THE COMPILER PROGRAM IS TO TRANSLATE THE ENGLISH DESCRIPTION OF A SYSTEM INTO A TABLE AND LIST STRUCTURE FOR FURTHER COMPUTER PROCESSING. THE INPUT TO THE COMPILER CONSISTS OF PUNCHED CARDS. THE COMPILER PROGRAM GENERATES NAME AND LABEL TABLES, PROCESSES THE CONDITION AND OPERATION LISTS, AND PRODUCES DIAGNOSTIC MESSAGES. THE GENERATED TABLES AND LISTS DESCRIBING THE OPERATION OF THE SYSTEM ARE PLACED ON AUXILIARY MEMORY AND MAY BE PRINTED AS

A USER OPTION. AFTER COMPILATION THE FLOW CHART SIMULATOR OR THE TABLE GENERATOR PROGRAMS MAY BE RUN.

3.2 FLOW CHART SIMULATION

THE PURPOSE OF THE FLOW CHART SIMULATOR IS TO VERIFY THAT THE DESCRIPTION OF THE DESIRED SYSTEM OPERATION IS ACCURATE. THE INPUT TO THE SIMULATOR CONSISTS OF THE TABLES GENERATED BY THE COMPILER AND A DESCRIPTION OF THE INPUT SIGNALS FOR WHICH THE SIMULATION IS TO BE RUN. THE INPUTS MAY BE ENTERED ON CARDS OR GRAPHICALLY AND THE DESIRED OUTPUTS MAY BE OBTAINED ON THE LINE PRINTER OR ON A GRAPHIC DISPLAY. AFTER ANALYSIS OF THE OUTPUTS THE DESIGNER CAN EITHER MODIFY THE ORIGINAL DESCRIPTION OF THE SYSTEM AND REPEAT COMPILATION OR PROCEED TO THE GENERATION OF THE STATE AND UNIT OPERATION TABLES.

3.3 TABLE GENERATION

THE PURPOSE OF THE TABLE GENERATOR PROGRAM IS TO TRANSLATE FROM A TIME ORIENTED FLOW CHART DESCRIPTION OF A SYSTEM TO A UNIT ORIENTED BLOCK DIAGRAM DESCRIPTION. THE TABLE GENERATOR READS THE LISTS AND TABLES FROM AUXILIARY MEMORY AND GENERATES THE STATE AND UNIT OPERATION TABLES FOR THE SYSTEM.

THE STATE TABLE IS GENERATED BY ASSIGNING STATES TO GROUPS OF STATEMENTS AND THEN ANALYZING THE STATEMENTS TO DETERMINE THE NEXT ACTIVE STATEMENT IN EACH CASE. GENERATION OF THE UNIT OPERATION TABLE IS PERFORMED BY SORTING THE OPERATION LISTS SO THAT ALL OPERATIONS ON THE SAME UNIT ARE GROUPED TOGETHER. THE CONDITIONS NECESSARY FOR THE PERFORMANCE OF EACH OPERATION ARE THEN OR'ED TO FORM A BOOLEAN EXPRESSION SPECIFYING WHEN EACH PARITICULAR OPERATION IS TO OCCUR.

THE PROGRAM WILL DISPLAY THE TABLES GENERATED (LINE PRINTER, DIGITAL TELEVISION DISPLAY SYSTEM OR PLOTTER) AND WILL ACCEPT MODIFICATIONS. ANY MODIFICATIONS ENTERED WILL CAUSE ALL TABLES AND LISTS DESCRIBING THE SYSTEM TO BE UPDATED. UPON COMPLETION OF TABLE GENERATION THE USER CAN PROCEED TO THE STATE ASSIGNMENT PROGRAM.

3.4 STATE ASSIGNMENT

THE STATE ASSIGNMENT PROGRAM READS THE STATE TABLE FROM AUXILIARY MEMORY AND ASSIGNS MEMORY ELEMENT CONFIGURATIONS TO STATES. ALTHOUGH SOME TECHNIQUES EXIST FOR OPTIMUM STATE ASSIGNMENT IT IS DIFFICULT TO GENERALIZE TO THE MULTI-INPUT/MULTI-OUTPUT CASE. IN ADDITION THE DESIGN TIME REQUIRED USUALLY PROHIBITS THE USE OF AN OPTIMUM METHOD.

THE COST OF THE CONTROL LOGIC OF A SYSTEM IS DETERMINED BY

1. THE NUMBER OF MEMORY ELEMENTS USED.

2. THE AMOUNT OF LOGIC REQUIRED TO GENERATE CONTROL SIGNALS TO THE FUNCTIONAL UNITS OF THE SYSTEM.

3. THE AMOUNT OF LOGIC REQUIRED TO GENERATE THE INPUT EQUATIONS DETERMINING THE NEXT STATE CONFIGURATIONS.

4. THE AMOUNT OF TIME REQUIRED TO DESIGN THE CONTROL LOGIC.

THE MINIMUM NUMBER OF MEMORY ELEMENTS WILL BE USED IF THE STATES ARE MAPPED INTO A BOOLEAN N-CUBE. THE LOGIC REQUIRED TO GENERATE THE CONTROL SIGNALS WILL BE MINIMIZED IF THE MAPPING IS SUCH THAT THE OCCURANCE OF A GIVEN CONDITION IN ANY STATE OF A MAXIMAL SUB-CUBE LEADS TO THE SAME SET OF CONTROL SIGNALS. THE LOGIC REQUIRED FOR THE NEXT STATE SIGNALS WILL BE MINIMIZED IF THE MAPPING IS SUCH THAT ALL TRANSITIONS FROM STATES OF A SUB-CUBE GO TO CORRESPONDING STATES OF OTHER SUB-CUBES.

IN GENERAL IT WILL NOT BE POSSIBLE TO MINIMIZE BOTH THE CONTROL LOGIC AND THE NEXT STATE LOGIC WITH THE SAME STATE ASSIGNMENT. IT WILL THEN BE NECESSARY TO CONSIDER SEVERAL POSSIBLE ASSIGNMENTS IN ORDER TO PICK THE BEST ONE.

ONE HEURESTIC METHOD OF STATE ASSIGNMENT THAT MAY BE EMPLOYED IS TO GROUP STATES HAVING THE SAME SET OF CONTROL SIGNALS FOR THE SAME SYSTEM CONDITIONS TOGETHER AND TO WEIGHT STATE PAIRS BY THE NUMBER OF TRANSITIONS BETWEEN PAIRED STATES. STATES MAY THEN BE ASSIGNED TO MEMORY ELEMENT CONFIGURATIONS BY STARTING WITH THE STATES HAVING THE GREATEST WEIGHT AND FOLLOWING THE PROCEDURE BELOW.

1. FIND THE STATE MOST CONNECTED TO A PREVIOUSLY ASSIGNED STATE.

2. ASSIGN THAT STATE TO A CONFIGURATION A UNIT HAMMING DISTANCE AWAY FROM THE OTHER STATE OF THE PAIR.

3. IF THE NEW STATE IS IN ONE OF THE GROUPS OF STATES ASSIGN ALL STATES IN THE GROUP SO AS TO FORM A SUB-CUBE AND MARK THE ASSIGNED STATES AND MEMORY ELEMENT CONFIGURATIONS.

4. REPEAT UNTIL ALL STATES ARE ASSIGNED.

THE ASSIGNMENT PRODUCED IN THIS WAY WILL USE THE MINIMUM POSSIBLE NUMBER OF MEMORY ELEMENTS. THE AMOUNT OF LOGIC REQUIRED WILL BE REDUCED (BUT NOT MINIMIZED).

IT IS POSSIBLE TO ARRIVE AT A DIFFERENT APPROACH TO THE STATE ASSIGNMENT PROBLEM BY THE FOLLOWING REASONING.

A. CONTROL LOGIC MAY BE REDUCED IF FOR SEVERAL STATES AND THE SAME SYSTEM CONDITIONS THE SAME CONTROL SIGNALS ARE REQUIRED AND THE SAME BITS OF THE MEMORY ELEMENTS ARE TO BE CHANGED. IN THIS CASE STATES ARE ASSIGNED TO SUB-CELLS OF THE N-CUBE AND IT IS NOT NECESSARY TO DECODE ALL STATES COMPLETELY.

B. IN TYPICAL SYSTEMS HAVING SEVERAL INPUTS, REGISTERS, COUNTERS, ETC. THERE WILL BE MANY DIFFERENT SYSTEM CONDITIONS AND MANY DIFFERENT CONTROL SIGNALS.

C. THUS IN A TYPICAL SYSTEM NEARLY ALL STATES WILL HAVE TO BE COMPLETELY DECODED AND VERY LITTLE LOGIC REDUCTION WILL BE POSSIBLE.

D. IN CURRENT INTEGRATED CIRCUIT LOGIC FAMILIES THE COST OF MEMORY ELEMENTS IN BUFFER REGISTER CONFIGURATIONS IS COMPARABLE TO THAT OF GATES.

E. THEREFORE IT MAY BE PREFERABLE TO MINIMIZE THE LOGIC REQUIRED TO DECODE THE STATES RATHER THAN TO MINIMIZE THE NUMBER OF MEMORY ELEMENTS REQUIRED.

ONE METHOD OF MINIMIZING THE DECODE LOGIC REQUIRED IS TO USE ONE MEMORY ELEMENT FOR EACH STATE. THIS WILL ELEMINATE ALL LOGIC REQUIRED TO DECODE THE STATES AND WILL GREATLY REDUCE THE DESIGN TIME. THE ADDITIONAL LOGIC FOR GENERATING THE CONTROL AND NEXT STATE SIGNALS WILL DEPEND ON THE PARTICULAR SYSTEM BUT SHOULD BE ROUGHLY COMPARABLE TO THAT REQUIRED FOR THE PREVIOUS METHOD. A PRELEMINARY ANALYSIS OF THIS TECHNIQUE OF STATE ASSIGNMENT INDICATES THAT THE COST OF LOGIC ELEMENTS AND CONSTRUCTION WILL ALMOST ALWAYS BE LESS THAN FOR THE MINIMUM MEMORY ELEMENT METHOD.

SINCE THE DESIGN TIME WILL ALWAYS BE LESS AND SINCE THIS IMPLEMENTATION WILL BE EASIER TO TROUBLESHOOT IT APPEARS TO BE PREFERABLE.

3.5 LOGIC GENERATION

THE PURPOSE OF THE LOGIC GENERATOR PROGRAM IS TO TRANSLATE THE BOOLEAN LOGIC EQUATIONS GENERATED BY THE PREVIOUS PROGRAMS INTO A LOGIC DIAGRAM. THE PROGRAM READS THE STATE AND UNIT OPERATION TABLES AND THE STATE ASSIGNMENT FROM AUXILIARY MEMORY. THE PROGRAM ALSO ACCEPTS A DESCRIPTION OF THE LOGIC ELEMENTS AVAILABLE FROM CARDS OR AUXILIARY MEMORY. THE SYSTEM IS THEN IMPLEMENTED USING THE DESCRIBED LOGIC ELEMENTS.

THE GENERAL PROBLEM OF FINDING A MINIMUM HARDWARE IMPLEMENTATION USING A SPECIFIC GROUP OF LOGIC ELEMENTS AND MULTILEVEL GATING IS UNSOLVED. THE APPROACH TO BE TAKEN IN THE CADSS SYSTEM IS TO FACTOR OUT THE MOST USED FUNCTIONS. IMPLEMENT THESE AND REPEAT. IT IS ANTICIPATED THAT THE RESULTS WILL BE NEAR MINIMAL BECAUSE OF THE NATURE OF THE CONDITION LISTS IN THE ORIGINAL DESCRIPTION.

AFTER A DESIGN IS COMPLETED IT WILL BE DISPLAYED (DTDS OR GERBER) TO ALLOW THE USER TO ANALYZE THE RESULTS. IN MOST CASES THE USER WILL BE ABLE TO REDUCE THE LOGIC OR IMPROVE THE SYSTEM IN SOME WAY. THE CADSS SYSTEM WILL THEN ACCEPT ANY DESIRED MODIFICATIONS AND UPDATE THE INTERNAL LOGIC DESCRIPTION ACCORDINGLY. THE RESULTING LOGIC DIAGRAM MAY ALSO BE DISPLAYED.

3.6 LOGIC SIMULATION

AFTER A LOGIC DIAGRAM IS OBTAINED THE LOGIC SIMULATOR PROGRAM MAY BE USED TO VERIFY THE OPERATION OF THE SYSTEM. THE LOGIC SIMULATOR READS THE DESCRIPTION OF THE LOGIC DIAGRAM FROM AUXILIARY MEMORY AND ACCEPTS A DESIRED INPUT SEQUENCE FROM THE USER. THE SYSTEM WILL THEN BE SIMULATED BY STANDARD TECHNIQUES AND THE DESIRED OUTPUTS DISPLAYED. AFTER SIMULATION THE USER MAY RETURN TO THE LOGIC GENERATION PROGRAM TO MAKE ADDITIONAL MODIFICATIONS TO THE SYSTEM OR PROCEED TO THE LOGIC PLACEMENT PROGRAM.

3.7 LOGIC PLACEMENT AND WIRE LIST GENERATION

THE LOGIC PLACEMENT PROGRAM READS THE LOGIC DIAGRAM PRODUCED BY THE LOGIC GENERATOR PROGRAM AND ACCEPTS A DESCRIPTION OF THE HARDWARE CONFIGURATION FROM THE USER. THE SYSTEM WILL ALLOW ANY DESIRED ELEMENTS TO BE PREPOSITIONED OR GROUPED TOGETHER.

THE CADSS PROGRAM WILL THEN POSITION THE REMAINING ELEMENTS FOLLOWING A HEURISTIC PROCEDURE TO REDUCE THE TOTAL WIRE LENGTH SUBJECT TO CONSTRAINTS SUCH AS THE NUMBER OF CONNECTIONS TO A TERMINAL AND THE MAXIMUM ALLOWABLE LENGTH OF A SIGNAL PATH. AFTER COMPLETION THE LOGIC PLACEMENT WILL BE DISPLAYED AND MAY BE MODIFIED BY THE USER.

WHEN AN ACCEPTABLE PLACEMENT IS FOUND THE CADSS SYSTEM WILL LIST THE LOGIC MODULES WHICH ARE TO BE PLACED IN EACH LOCATION AND WILL PRINT THE WIRE CONNECTION LIST FOR THE SYSTEM.

4 THE CADSS MODEL OF A DIGITAL SYSTEM

CONSIDERABLE INSIGHT INTO THE PROBLEM OF LOGIC GENERATION MAY BE OBTAINED BY ANALYZING THE DIGITAL SYSTEM MODEL ASSUMED BY THE CADSS SYSTEM.

THE MOST COMMON MODEL OF A DIGITAL SYSTEM IS OBTAINED BY SEPARATING THE MEMORY (STATE) ELEMENTS AND THE COMBINATORIAL LOGIC ELEMENTS AS SHOWN IN FIGURE 4.1.

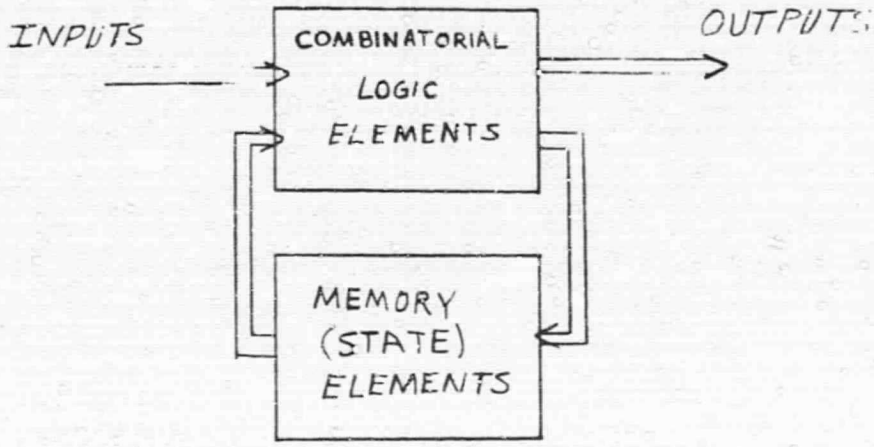

FIGURE 4.1 FIRST MODEL OF A DIGITAL SYSTEM

DIGITAL SYSTEMS OF TYPICAL COMPLEXITY NORMALLY CONTAIN A GROUP OF MEMORY ELEMENTS FUNCTIONALLY CONNECTED AS UNITS SUCH AS REGISTERS OR COUNTERS. WHILE THESE FUNCTIONAL UNITS MAY HAVE SOME PART IN DETERMINING THE OVERALL STATE OF THE SYSTEM (FROM THE VIEWPOINT OF AUTOMATA THEORY) THEY ARE NOT GENERALLY CONSIDERED AS STATE ELEMENTS. IF THE FUNCTIONAL UNITS ARE SHOWN SEPARATELY A SECOND MODEL OF A DIGITAL SYSTEM IS OBTAINED AS SHOWN IN FIGURE 4.2.

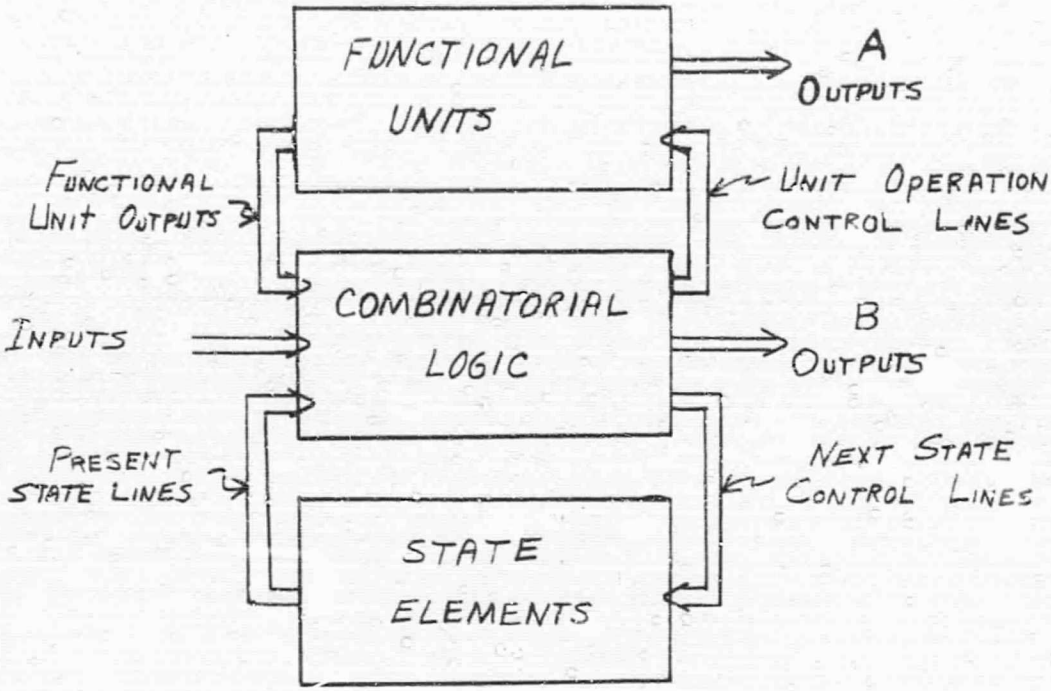

FIGURE 4.2 SECOND MODEL OF A DIGITAL SYSTEM

IN THIS MODEL THE COMBINATORIAL LOGIC RECIEVES INPUTS FROM EXTERNAL SOURCES, THE STATE ELEMENTS AND THE FUNCTIONAL UNITS. THE COMBINATORIAL LOGIC GENERATES THE SIGNALS DEFINING THE NEXT STATE OF THE SYSTEM, THE UNIT OPERATION COMMANDS (RESET, SHIFT, INCREMENT, ETC) AND ANY OUTPUTS NOT TAKEN DIRECTLY FROM THE FUNCTIONAL UNITS.

THE THIRD MODEL IS FORMED BY SEPARATING THE COMBINATORIAL LOGIC INTO TWO LEVELS AS SHOWN IN FIGURE 4.3. THE OUTPUTS OF THE FUNCTIONAL UNITS ARE FIRST COMBINED WITH THE EXTERNAL INPUTS TO FORM SIGNALS DEFINING THE CONDITION OF THE SYSTEM. IN THE SECOND LEVEL OF LOGIC THE CONDITION DEFINING SIGNALS ARE COMBINED WITH THE STATE OUTPUTS TO GENERATE THE UNIT OPERATION, NEXT STATE, AND OUTPUT SIGNALS.

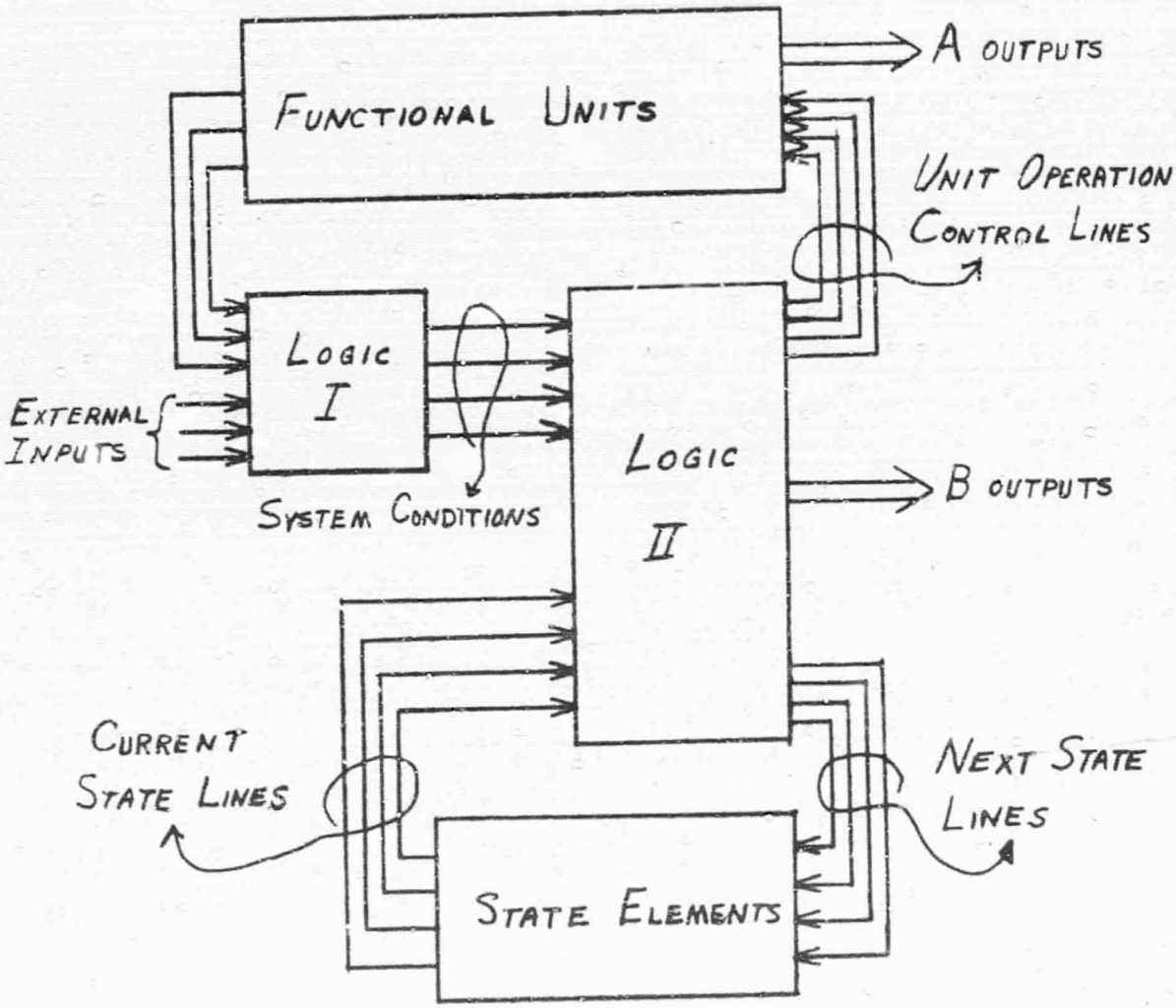

FIGURE 4.3

THIRD MODEL OF A DIGITAL SYSTEM

THE THIRD MODEL CAN BE RELATED TO THE CADSS DESCRIPTION OF A DIGITAL SYSTEM IN THE FOLLOWING WAYS

1 THE INPUT LINES CORRESPOND TO THE INPUT DEFINITIONS OF THE CADSS DESCRIPTION.

2 THE OUTPUT LINES OF THE MODEL CORRESPOND TO THE DEFINED OUTPUTS OF THE CADSS DESCRIPTION. BOOLEAN STATEMENTS ARE USED TO SPECIFY THE UNITS CONNECTED TO THE OUTPUTS (A OUTPUTS) OR THE GATING REQUIRED TO GENERATE THE OUTPUT (B OUTPUTS).

3 THE SYSTEM CONDITION LINES OF THE MODEL CORRESPOND TO THE CONDITION LISTS OF THE CADSS DESCRIPTION. THERE WILL BE ONE CONDITION LINE FOR EACH DIFFERENT CONDITION LIST.

4 THE UNIT OPERATION CONTROL LINES CORRESPOND TO THE COMMANDS IN THE UNIT OPERATION LISTS OF THE CADSS DESCRIPTION. THE BOOLEAN EQUATION FOR EACH OPERATION CONTROL LINE MAY BE OBTAINED BY 'ANDING' THE CONDITION LIST WITH THE STATE FOR EACH STATEMENT CONTAINING THE COMMAND AND THEN 'ORING' THE FUNCTIONS SO FORMED.

5 THE NEXT STATE LINES OF THE MODEL DO NOT APPEAR EXPLICITLY IN THE CADSS DESCRIPTION OF A SYSTEM BUT MAY BE DETERMINED FROM THE GENERATED STATE TABLE. THE BOOLEAN EQUATION FOR EACH NEXT STATE LINE IS OBTAINED BY 'ANDING' THE STATE WITH THE CONDITION FOR EACH STATEMENT CONTAINING A TRANSFER TO THE PARTICULAR NEXT STATE UNDER CONSIDERATION. ALL OF THE FUNCTIONS FORMED IN THIS WAY ARE THEN 'ORED' TO FORM THE COMPLETE EQUATION FOR THAT PARTICULAR NEXT STATE LINE.

ALTHOUGH THIS MODEL IS USED IN THE INITIAL TRANSLATION FROM THE CADSS DESCRIPTION TO THE BOOLEAN EQUATIONS DESCRIBING THE SYSTEM THE FINAL LOGIC DESIGN WILL NOT BE PARTITIONED IN THE SAME WAY. LOGIC REDUCTION BY FACTORING WILL REMOVE THE DISTINCTION BETWEEN THE TWO LEVELS OF LOGIC GATING. SOME STATE ASSIGNMENTS WILL RESULT IN 'BIT CHANGE LINES' RATHER THAN NEXT STATE LINES. THE MODEL SERVES ONLY AS A TOOL IN TRANSLATING THE INITIAL DESCRIPTION INTO A MATHEMATICAL FORM.