TECHNICAL REPORT NUMBER AU-T-16

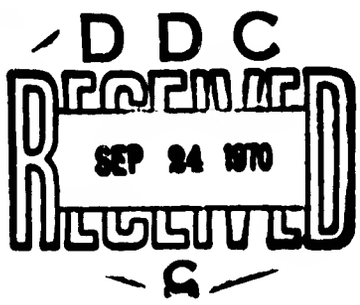
AN ALGORITHM FOR FAST BOOLEAN FUNCTION MINIMIZATION USING PROPERTIES OF THE CELLULAR N-CUBE

PROJECT THEMIS: INFORMATION PROCESSING

PREPARED BY
DIGITAL SYSTEMS LABORATORY
C. C. CARROLL, PROJECT LEADER

AUGUST, 1970

CONTRACT DAAH01-68-C-0296
ARMY MISSILE COMMAND
HUNTSVILLE, ALABAMA



PROJECT THEMIS
AUBURN UNIVERSITY

TECHNICAL REPORT NUMBER AU-T-16

AN ALGORITHM FOR FAST BOOLEAN FUNCTION MINIMIZATION USING PROPERTIES OF THE CELLULAR N-CUBE

Prepared by

DIGITAL SYSTEMS LABORATORY

C. C. CARROLL, PROJECT LEADER

August, 1970

PROJECT THEMIS: INFORMATION PROCESSING

Contract DAAH01-68-C-0296

Army Missile Command

Huntsville, Alabama

Approved and Submitted by:

C. C. Carroll
Professor and Head
Electrical Engineering
Themis Project Director

Ben T. Lanham, Jr.
Vice President for Research
Auburn University

FOREWORD

This report is a technical summary reporting the progress of a study conducted by the Digital Systems Laboratory of the Electrical Engineering Department, Auburn University. The study is focused toward fulfillment of Contract No. DAAH 01-68-C-0296, granted to Auburn University, Auburn, Alabama, by the Army Missile Command, Huntsville, Alabama.

ACKNOWLEDGEMENT

The authors wish to express appreciation to Mr. George E. Jordan for the valuable assistance he rendered in connection with formulating the algorithm which was used as the basis of this report.

AN ALGORITHM FOR FAST BOOLEAN FUNCTION MINIMIZATION

USING PROPERTIES OF THE CELLULAR N-CUBE

C. C. Carroll and W. A. Hornfeck

ABSTRACT

Properties of the cellular n-cube representation are used to advantage in developing a fast algorithm for finding the prime implicants of a Boolean function. The algorithm is discussed and several examples are included showing computer solutions to selected Boolean function minimization problems. The complete PL/I source program listing for the automated algorithm is included.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

The problem of determining the prime implicants of a Boolean function [1], [2], [3], [4], was treated by C. C. Carroll and G. E. Jordan in a previous report [5]. An automated design algorithm was developed which could quickly produce a list of prime implicants in a form from which the minimum subset of prime implicants that cover the function could be chosen. The algorithm was based on properties of the cellular n-dimensional cubic representation [6], theorems pertaining to the containment of cells by other cells and a three-valued representation of variables in the function [7]. These representations along with the containment relations permitted the fast determination of the prime implicants of a function expressed in cellular notation.

In this report, the original FORTRAN source program has been converted to a PL/I source program and has been modified so that it can also reduce Boolean functions containing "don't care" terms. In addition, the present program is capable of handling Boolean functions of order 13 without special computer requirements if the number of minterms is not excessive; added storage requirements may be needed if the number of minterms is large and the function is of order 12 or 13. Adjustable length arrays have been used in a manner which minimizes the amount of storage area needed at any given point in the program.

Chapter II outlines the automated algorithm used for the minimization program. A general flowchart of the entire algorithm is included. The computer program is explained using this general flowchart along with additional flowcharts which are used to show portions of the general program in greater detail. The basic algorithm is the same as that used in the original FORTRAN version of the minimization program. Chapter III presents several examples of Boolean function minimization problems which have been solved using the programmed algorithm.

Two appendices are included in the report. Appendix 1 contains a discussion of the cellular representation and properties of Boolean functions. This appendix is a condensed version of a more complete account given by Carroll and Jordan in Chapter II of the original report. Appendix 2 contains the complete PL/I source program listing of the Boolean function minimization program.

II. THE ALGORITHM

The theorems which are outlined in Appendix 1 are used to construct a computer algorithm which quickly determines the prime implicants of a Boolean function. The algorithm also provides for automatically checking for the essential prime implicants. Finally, constraint equations are automatically generated for all minterms which are not included in essential prime implicants. The constraint equations are actually a listing of all prime implicants which contain a particular minterm.

A flowchart illustrating the main procedures used in the algorithm is shown in figure II-1. In the following paragraphs, we will discuss this general flowchart and each step of the general procedure will be broken down in further detail in subsequent flowcharts.

Referring to figure II-1, the first block of the programmed algorithm is the declaration of attributes for all variables and arrays to be used in the program. The storage attribute CONTROLLED will be used for all arrays so that storage can be made available for a particular array only during that part of the program in which the information is needed. In addition, all arrays will be of adjustable length so that the amount of storage needed can be adjusted according to the order of the Boolean function and the number of minterms in the function. A listing of all declared variables is included in the flowchart of figure II-2.

The second step (block no. 2) is to input the data which describes the Boolean function to be minimized. The data includes the order of

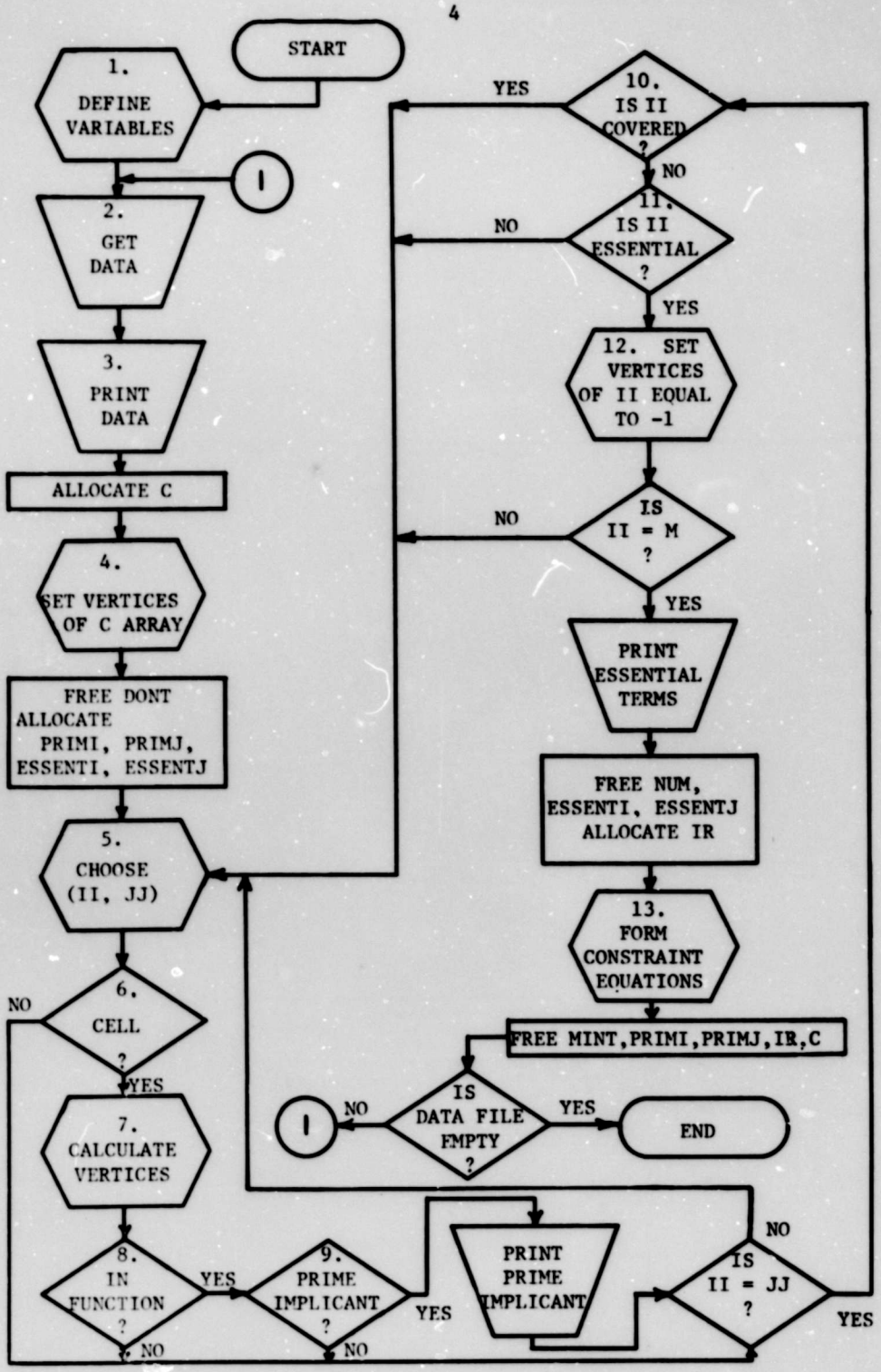


FIG. II-1. GENERAL FLOWCHART OF MINIMIZATION PROGRAM.

the function, N, the total number of minterms (including "don't cares") in the function, M, the number of don't cares in the function, D; the minterms (including don't cares) of the function which are stored as the one-dimensional array MINT, and finally the don't cares of the function which are stored as the one-dimensional array DONT. The arrays MINT and DONT must store the minterms and don't cares in numerically increasing order, which requires that the data be listed in numerically increasing order. After the data is stored, block no. 3 gives an output listing of the order of the function, the minterms (including don't cares) of the function, and the don't cares of the function (see figure II-2).

The one-dimensional array C is now allocated and block no. 4 is executed to set up the C array. The elements of C correspond to vertices of the cellular n-cube (or squares of the n-variable Karnaugh map). An element of C is given the value "1" if it corresponds to a minterm of the function; "-1" if it corresponds to a don't care term; and "0" if it does not correspond to either a minterm of a don't care term. The procedure for setting the values of vertices of C is shown in the flowchart of figure II-3.

The program is now ready to check all the different possible pairings of minterms to determine whether they form prime implicant cells (block no. 5). Coordinate pairs are denoted (II, JJ) and are chosen such that each minterm (beginning with the smallest), II, is paired with all other minterms, (beginning with the largest), JJ, which are

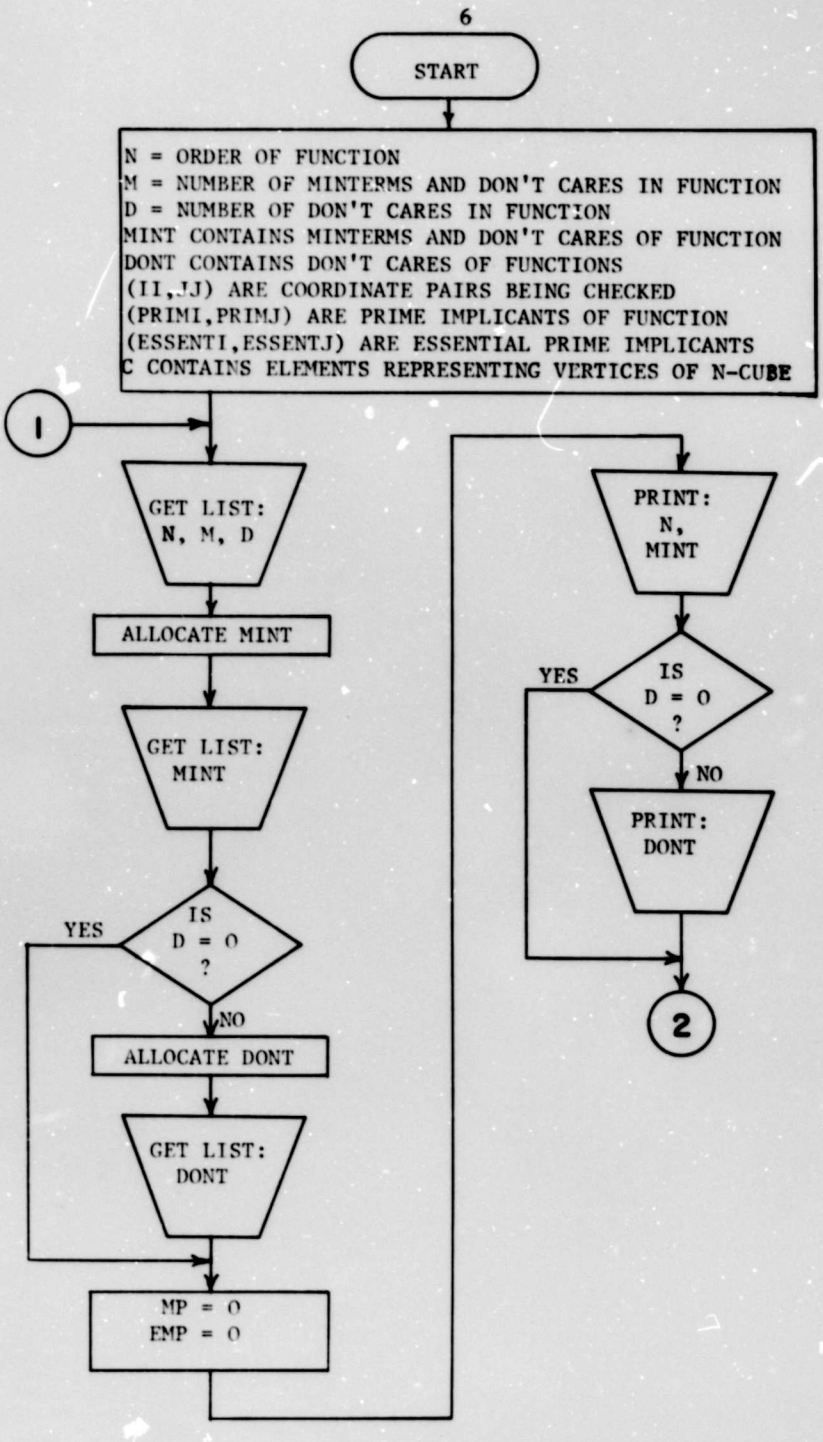


FIG. II-2. DEFINITION OF VARIABLES AND INPUT OF FUNCTION.

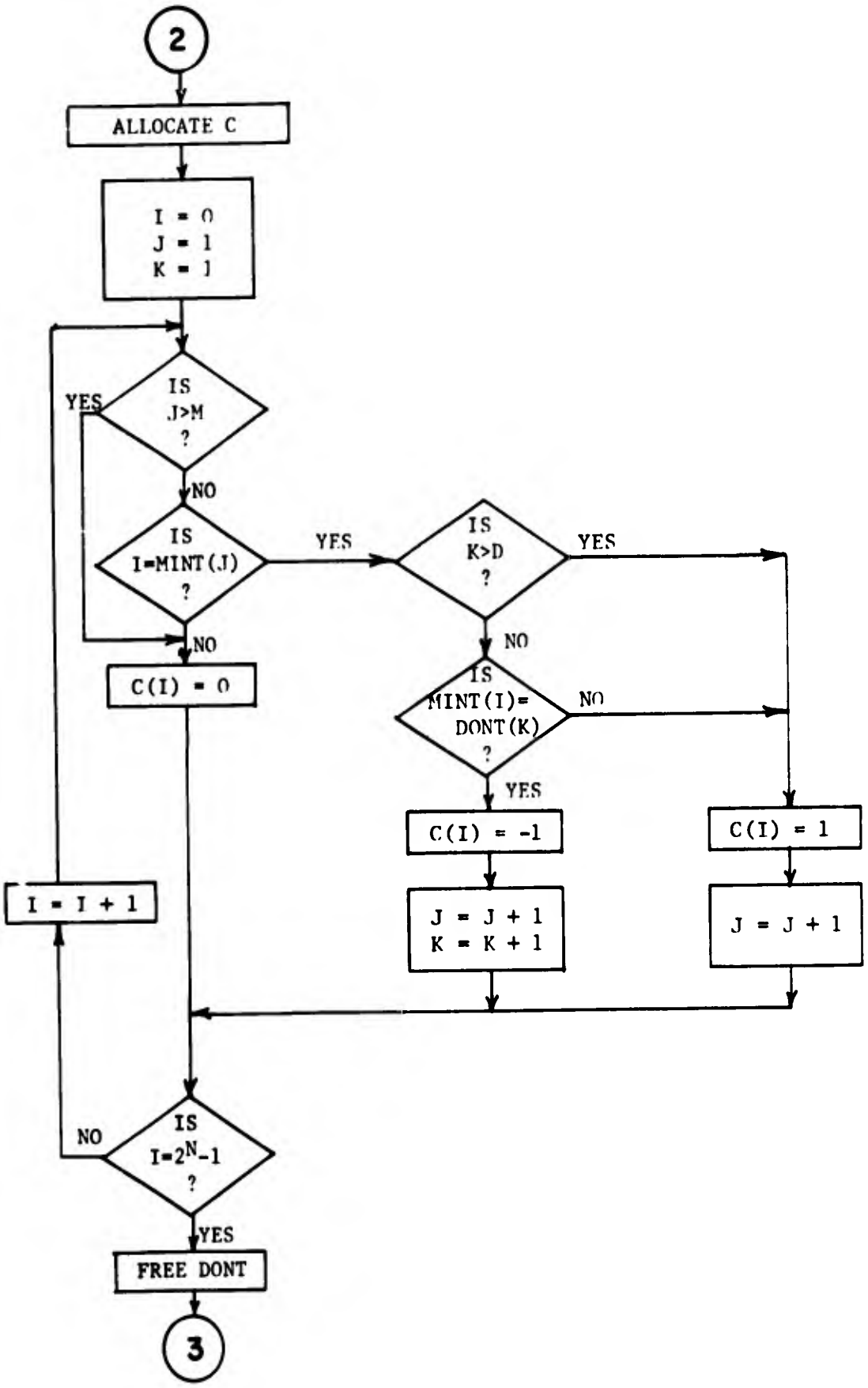


FIG. II-3. ASSIGNMENT OF VALUES TO THE ELEMENTS OF THE C ARRAY.

greater than or equal to II. After the coordinate pair (II, JJ) is chosen, it is first checked in block no. 6 to determine if it forms a cell by using Theorem 1 of Appendix 1. The above two steps are shown in detail as part of figure II-4 (a).

If (II, JJ) is a cell, block no. 7 of the program proceeds to calculate all the vertices which are contained in the cell (by applying Theorem 2 of Appendix 1) and, using the C array, decides whether the cell is a cell of the function as indicated by block no. 8. Blocks 7 and 8 of the program are described as part of the flowchart in figures II-4 (a) and (b).

If (II, JJ) is a cell and is a cell of the function, block no. 9 of the program then checks the array of all prime implicants to determine if the cell (II, JJ) is contained in one of the prime implicants found previously. If not, Lemma 4.1 of Appendix 1 allows us to decide at this point that (II, JJ) is indeed a prime implicant and (II, JJ) is added to the table of prime implicants. The flowchart of Figure II-5 illustrates this portion of the program.

After minterm II has been paired with all other minterms which are greater than or equal to II, block no. 10 of the program uses the C array to determine whether minterm II is either a don't care term or is included in an essential prime implicant. If it is not, block no. 11 of the program finds the number of prime implicants which contain II by using Theorem 3 of Appendix 1. If it is contained in only one prime implicant, then that prime implicant is essential and is added

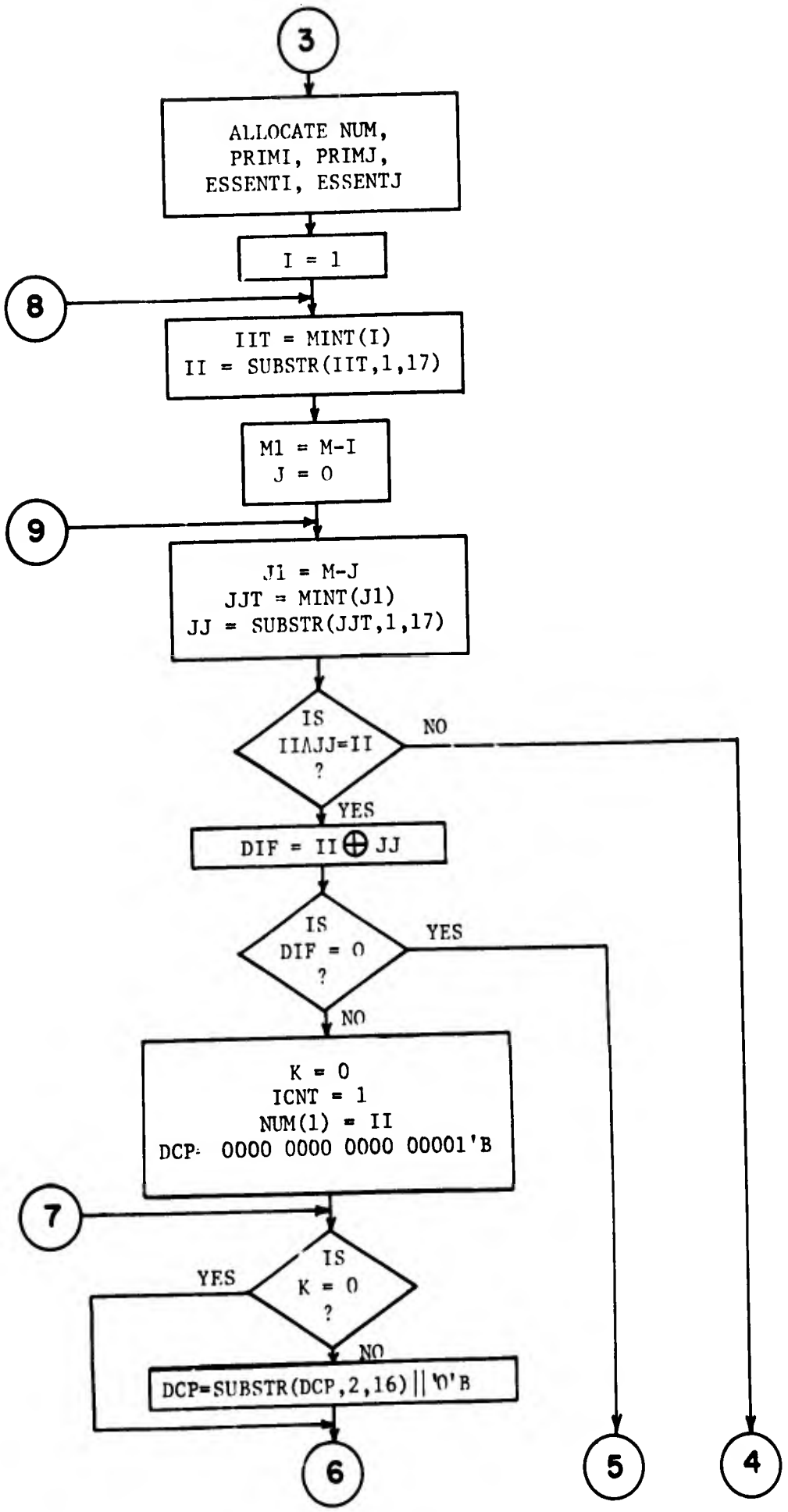


FIG. II-4a. CHECK IF (II,JJ) IS A CELL.

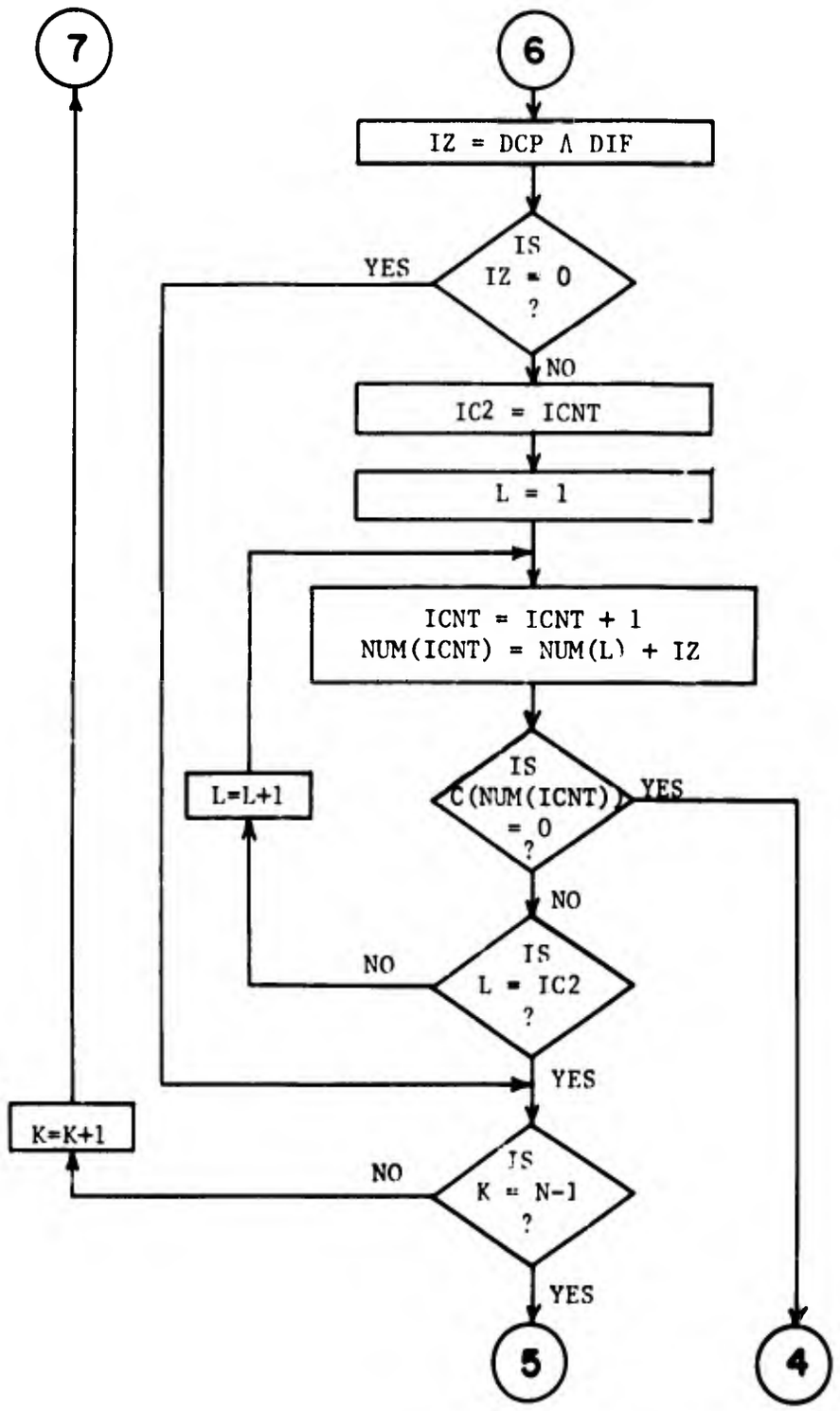


FIG. II-4b. CHECK IF CELL (II,JJ) IS A CELL OF THE FUNCTION.

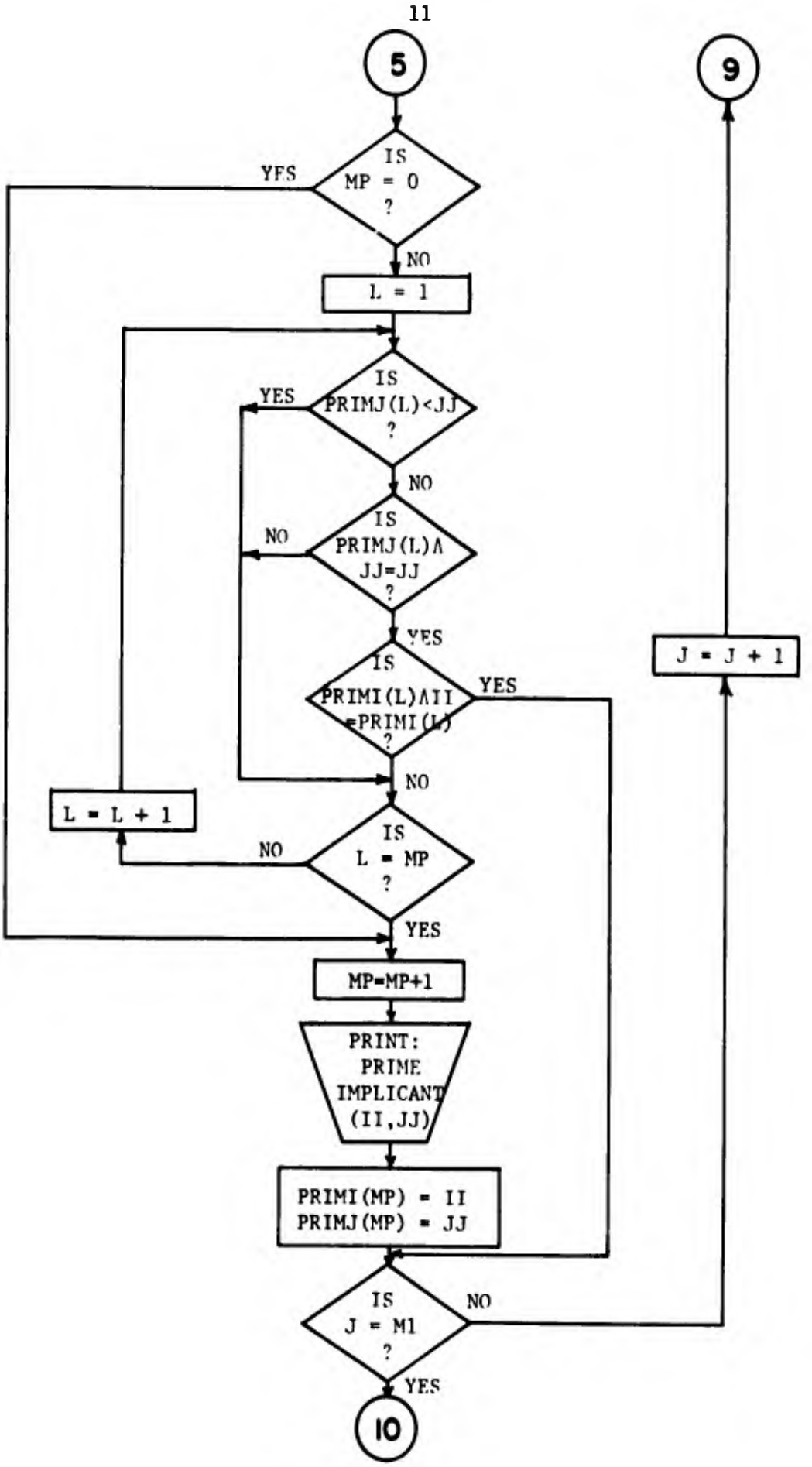


FIG. II-5. IS CELL A PRIME IMPLICANT?

to the essential prime implicant table. See Figure II-6 for the flowchart describing this procedure.

If the program finds that vertex II requires an essential prime implicant, block no. 12 is then executed. This portion of the program again makes use of Theorem 2 to calculate all the vertices contained in the essential prime implicant and sets these vertices equal to "-1" in the C array. Block no. 12 is shown in detail in Figure 7(a) and 7(b).

After all possible coordinate pairs are checked and all prime implicants and essential prime implicants are determined, the program checks the values of all vertices in the C array (block no. 13) which correspond to minterms of the function. If the value of a particular vertex in the C array is "-1" then either that minterm is a don't care or is included in one of the essential prime implicants. If the value of the vertex is "1" then the program must find the prime implicants which contain that vertex. This portion of the program uses Theorem 3 and Lemma 4.1 to determine all of the prime implicants which contain the minterm in question. This part of the program is broken down in Figure 8 (a). From the list of n prime implicants which include a particular minterm (which is not included in an essential prime implicant), a constraint equation can be set up in the form

$$(P.I.)_1 + (P.I.)_2 + \text{---} + (P.I.)_n \geq 1$$

where $n \geq 2$. From the set of all such constraint equations, one can find the minimum set of prime implicants which, along with the essential prime implicants, are necessary to completely cover the function.

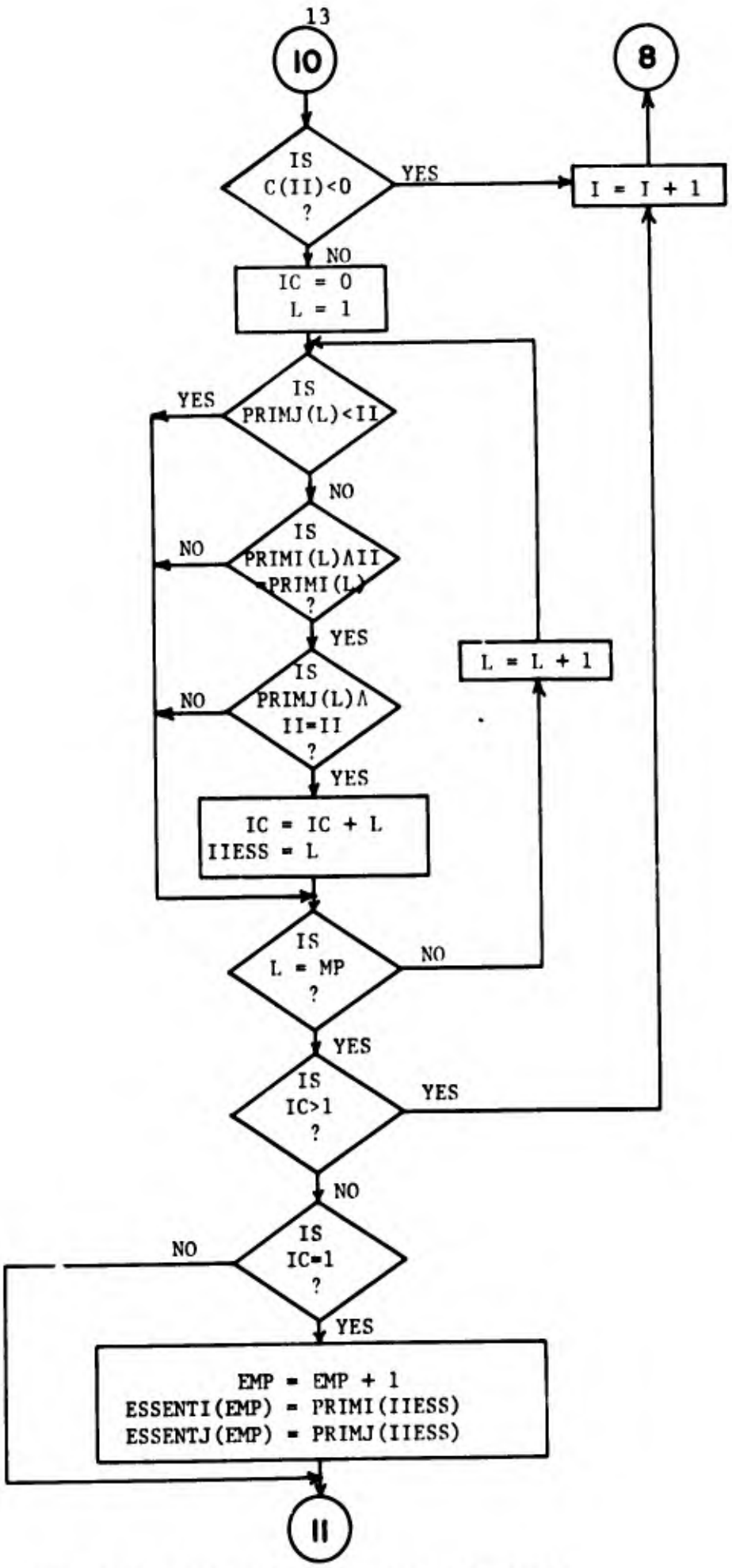


FIG. II-6. ESSENTIAL PRIME IMPLICANT CHECK.

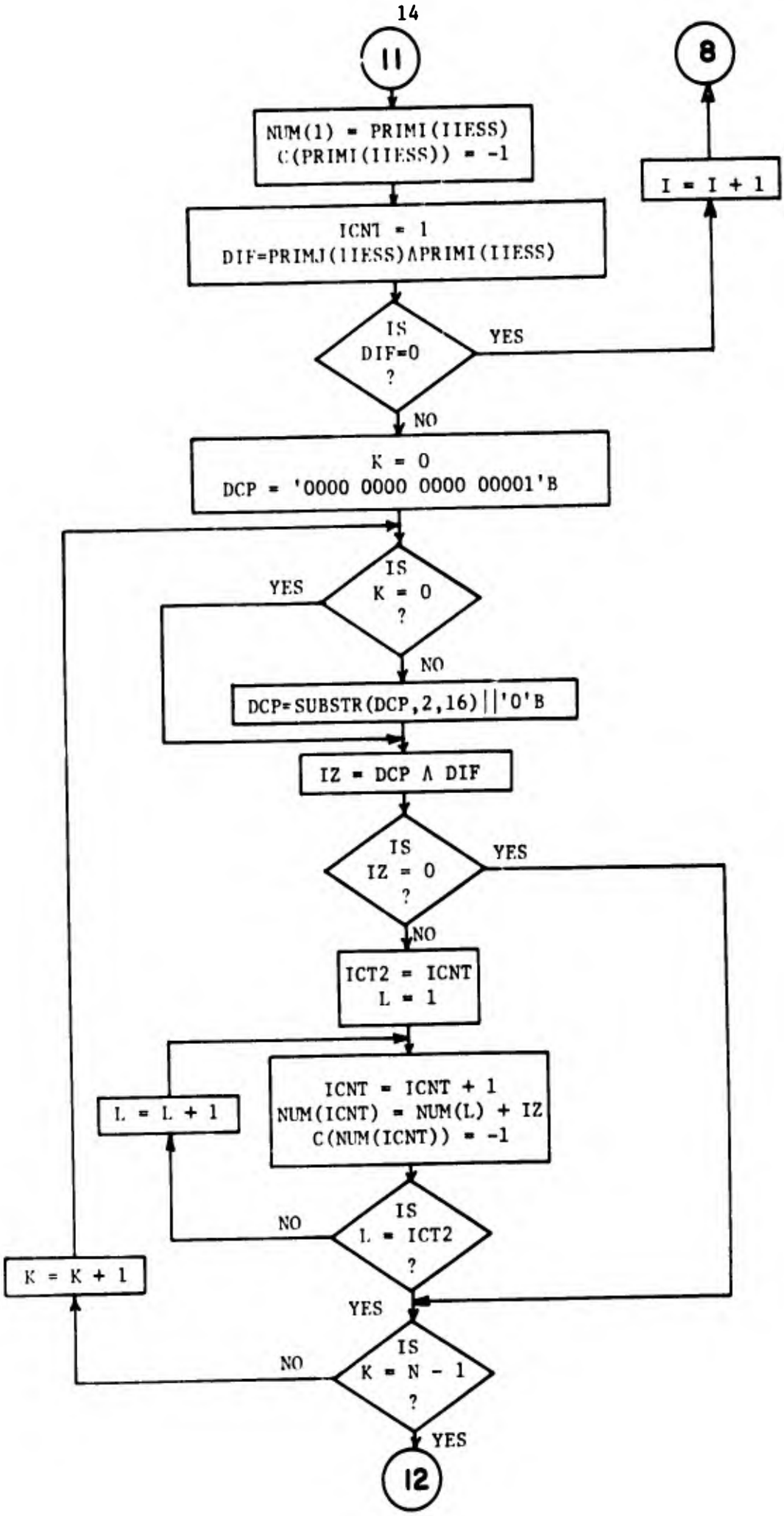


FIG. 7a. SET VERTICES OF ESSENTIAL PRIME IMPLICANT EQUAL TO -1.

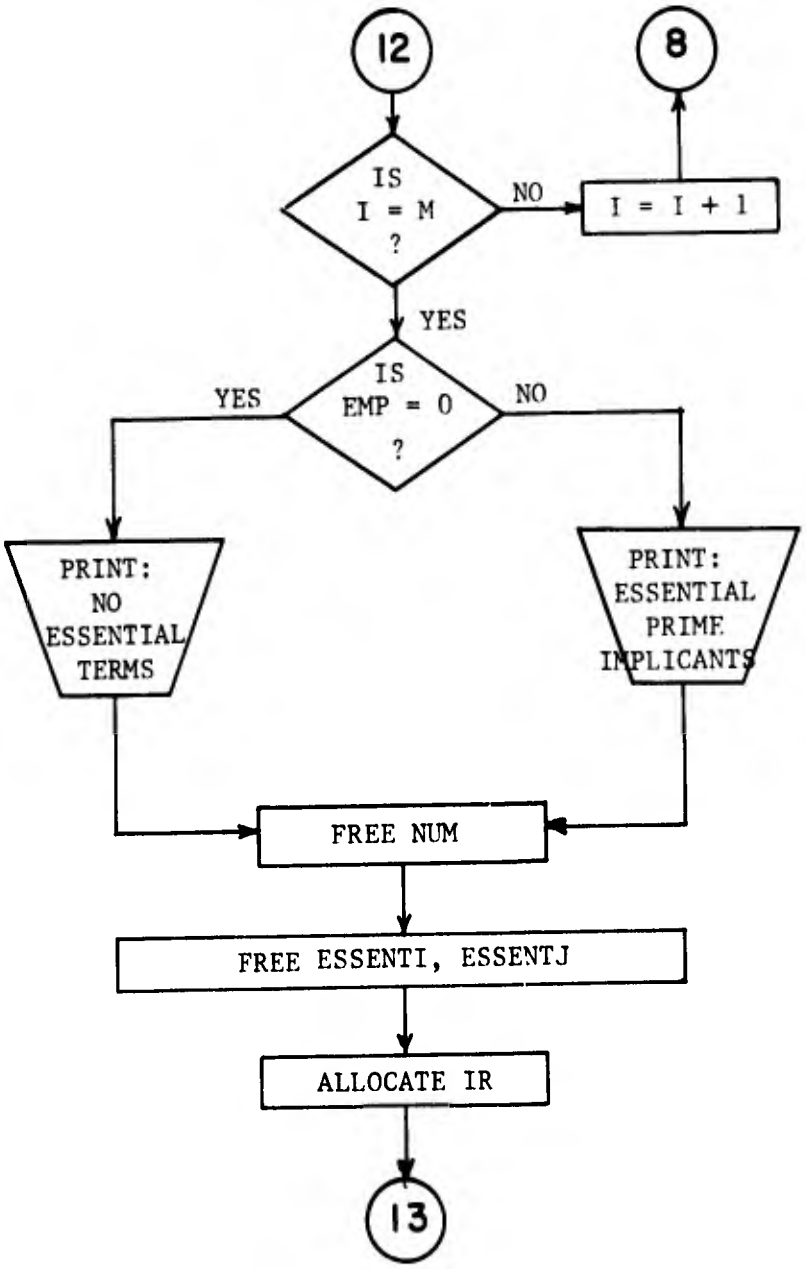


FIG. 7b. IF ALL MINTERMS HAVE BEEN CHECKED, PRINT THE ESSENTIAL PRIME IMPLICANTS AND FREE STORAGE ALLOCATED TO VARIABLES NO LONGER NEEDED.

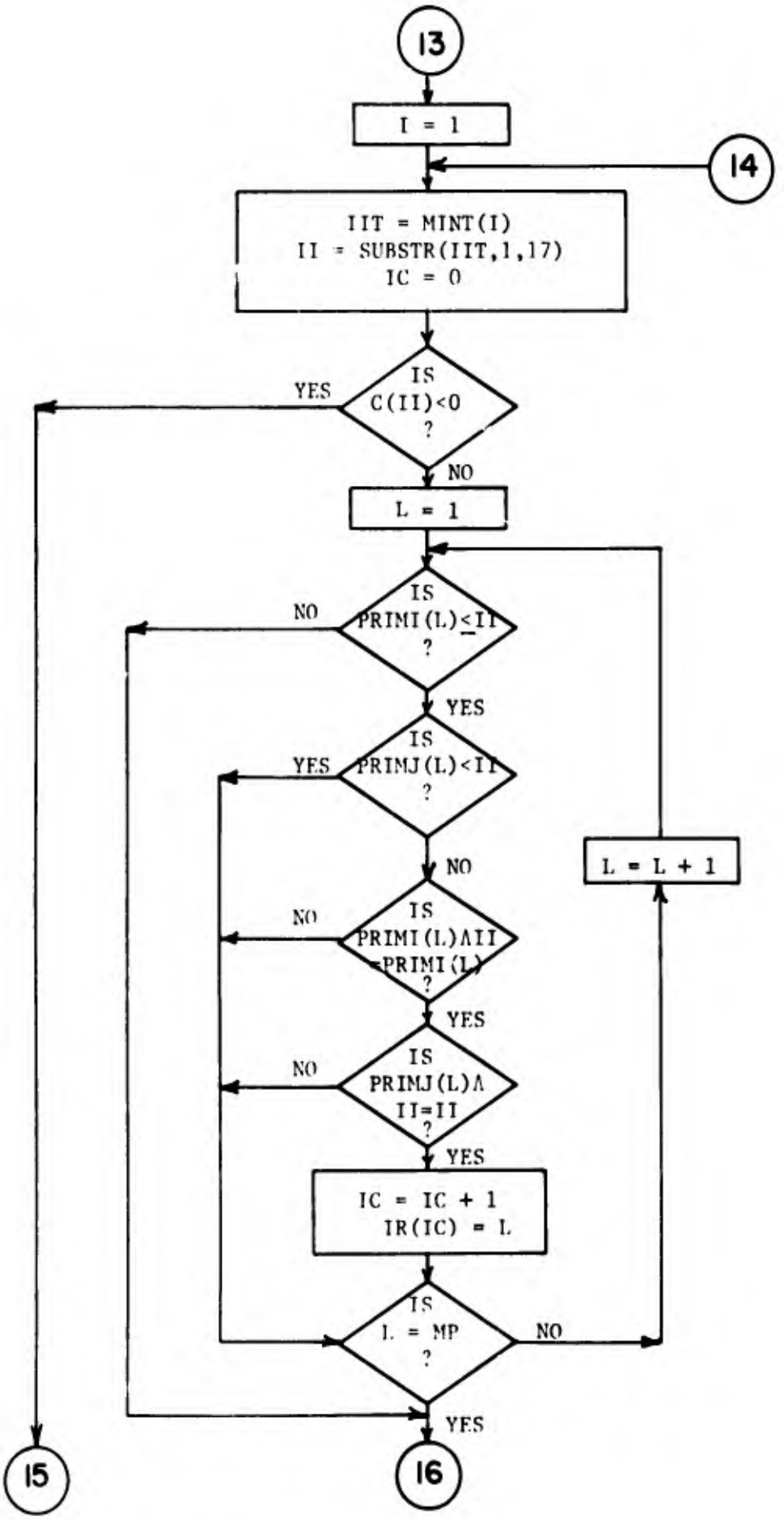


FIG. 8a. CALCULATION OF CONSTRAINT EQUATIONS.

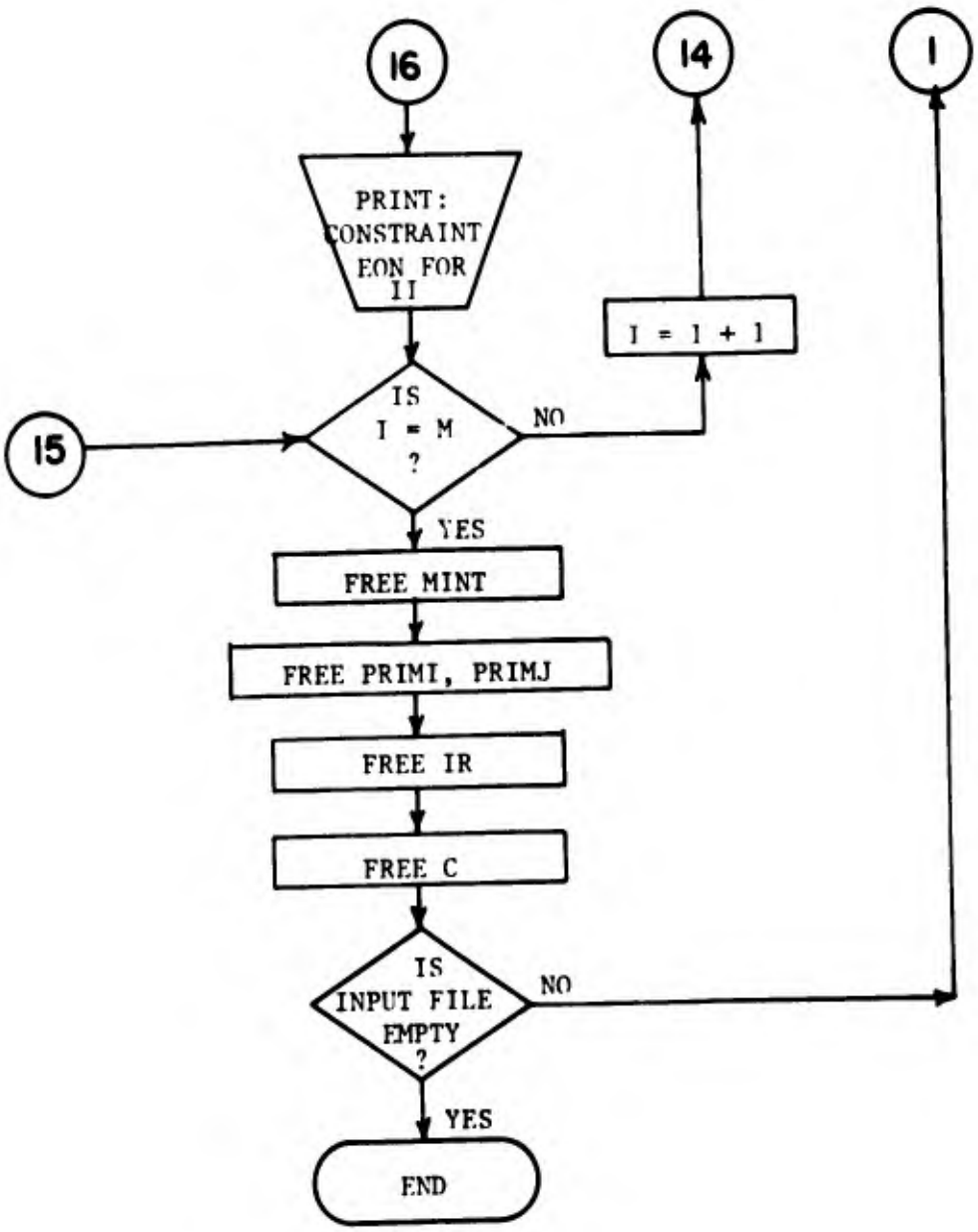


FIG. 8b. COMPLETION OF PROCEDURE WHICH FINDS CONSTRAINT EQUATIONS AND TERMINATION OF PROGRAM.

After each minterm has been checked as described in the preceding paragraph and all necessary constraint terms have been found, the program will free any storage which is still being allocated for the function, which at this point has been minimized. The program then will return to block no. 2 in order to input the next function which is in the data file. If the input data file does not contain another function to be minimized, the program will terminate. Figure 8(b) shows these final steps of the program in detail.

The discussion up to this point, along with the accompanying flowcharts, completely describes the fast computer algorithm. The reader will notice that it is basically a Quine-McCluskey type procedure; however, great savings in computer time and storage requirements are realized by modifying the Quine-McCluskey method using containment properties of the cellular n-cube representation. The next chapter will present some examples of Boolean function minimization performed by the programmed algorithm. The first example will illustrate each step performed by the program. Three more examples will then be presented which illustrate the normal output information provided by the program. Two of these examples are fairly trivial and the third gives the output for an 8th order function containing 202 minterms.

III. EXAMPLE PROGRAMS

Before presenting examples of the output of the computer algorithm in its final form, the computer output of a modified version of the minimization program will be illustrated. This program is for illustrative purposes and is shown in Figure III-1. It is seen from the output listing that each minterm is successively compared with all other minterms which are greater than or equal to itself. As each coordinate (minterm) pair is examined, the program checks

A. Each Coordinate Pair:
   (1) Is the coordinate pair a cell?
   (2) Is the cell a cell of the function?
   (3) Is the cell which is in the function, a prime implicant?
   (4) Is the prime implicant essential?

B. Each Minterm:
   (1) Is the minterm a don't care term or is the minterm included in an essential prime implicant?
   (2) Find all prime implicants which contain the minterm that is not covered.

These different actions taken by the program can be checked using the 4th order function which is minimized in the figure. It turns out that the function contains three essential prime implicants, (0, 6), (4, 15) and (9, 15), which represent the groupings of minterms {0,2,4,6}, {4,5,6,7,12,13,14,15} and {9,11,13,15} respectively. Minterm 10 is the only minterm which is not covered by the essential terms. The program therefore lists the prime implicants which contain minterm 10, which

```
THE BOOLEAN FUNCTION HAS ORDER  4

MINTERMS OF FUNCTION ARE
    0,    2,    4,    5,    6,    7,    9,   10,   11,   12,   13,   14,
   15,

DON'T CARES OF FUNCTION ARE
    7,   11,   12,   13,   14,   15,

COORDINATE PAIRS******CELL*FUNCTION**PRIME IMPLICANT**ESSENTIAL*****
(    0,   15)        YES         NO
(    0,   14)        YES         NO
(    0,   13)        YES         NO
(    0,   12)        YES         NO
(    0,   11)        YES         NO
(    0,   10)        YES         NO
(    0,    9)        YES         NO
(    0,    7)        YES         NO
(    0,    6)        YES         YES       YES
(    0,    5)        YES         NO
(    0,    4)        YES         YES         NO
(    0,    2)        YES         YES         NO
(    0,    0)        YES         YES         NO          (    0,    6)
(    2,   15)        YES         NO
(    2,   14)        YES         YES       YES
(    2,   13)        NO
(    2,   12)        NO
(    2,   11)        YES         NO
(    2,   10)        YES         YES         NO
(    2,    9)        NO
(    2,    7)        YES         NO
(    2,    6)        YES         YES         NO
(    2,    5)        NO
(    2,    4)        NO
(    2,    2)        YES         YES         NO
(    4,   15)        YES         YES       YES
(    4,   14)        YES         YES         NO
(    4,   13)        YES         YES         NO
(    4,   12)        YES         YES         NO
(    4,   11)        NO
(    4,   10)        NO
(    4,    9)        NO
(    4,    7)        YES         YES         NO
(    4,    6)        YES         YES         NO
(    4,    5)        YES         YES         NO
(    4,    4)        YES         YES         NO
(    5,   15)        YES         YES         NO
(    5,   14)        NO
(    5,   13)        YES         YES         NO
(    5,   12)        NO
(    5,   11)        NO
(    5,   10)        NO
(    5,    9)        NO
(    5,    7)        YES         YES         NO
(    5,    6)        NO
(    5,    5)        YES         YES         NO          (    4,   15)
(    6,   15)        YES         YES         NO
(    6,   14)        YES         YES         NO
(    6,   13)        NO
(    6,   12)        NO
```

NOT REPRODUCIBLE

FIG. III-1. COMPUTER PRINTOUT OF MODIFIED PROGRAM USED FOR EXAMPLE 1.

(6,	11)	NO					
(6,	10)	NO					
(6,	9)	NO					
(6,	7)	YES	YES	NO			
(6,	6)	YES	YES	NO			
(7,	15)	YES	YES	NO			
(7,	14)	NO					
(7,	13)	NO					
(7,	12)	NO					
(7,	11)	NO					
(7,	10)	NO					
(7,	9)	NO					
(7,	7)	YES	YES	NO			
(9,	15)	YES	YES	YES			
(9,	14)	NO					
(9,	13)	YES	YES	NO			
(9,	12)	NO					
(9,	11)	YES	YES	NO			
(9,	10)	NO					
(9,	9)	YES	YES	NO	(9,	15)	
(10,	15)	YES	YES	YES			
(10,	14)	YES	YES	NO			
(10,	13)	NO					
(10,	12)	NO					
(10,	11)	YES	YES	NO			
(10,	10)	YES	YES	NO			
(11,	15)	YES	YES	NO			
(11,	14)	NO					
(11,	13)	NO					
(11,	12)	NO					
(11,	11)	YES	YES	NO			
(12,	15)	YES	YES	NO			
(12,	14)	YES	YES	NO			
(12,	13)	YES	YES	NO			
(12,	12)	YES	YES	NO			
(13,	15)	YES	YES	NO			
(13,	14)	NO					
(13,	13)	YES	YES	NO			
(14,	15)	YES	YES	NO			
(14,	14)	YES	YES	NO			
(15,	15)	YES	YES	NO			

TERMS OF CONSTRAINT FOR FOR MINTERM 10 ARE (2, 14),
(10, 15),

FIG. III-1. (CON'T)

are (2, 14) and (10, 15), representing the groupings {2,6,10,14} and {10,11,14,15} respectively. Then, since the terms of the constraint equation

$$(2, 14) + (10, 15) \geq 1$$

are both of equal cost, we arbitrarily choose either one of the terms which, along with the three essential terms, realizes the function.

Figure II-2 shows the abbreviated output listing which is used in the normal program. The output represents the solution to a minimization problem involving a 5th order Boolean function containing no don't care terms. The program gives us the eight prime implicants of the function, the five essential prime implicants, and the four constraint equations:

$$\begin{aligned}
&\text{minterm } 2: \quad (0, 10) + (2, 11) \geq 1,\\
&\text{minterm } 3: \quad (2, 11) + (3, 15) \geq 1,\\
&\text{minterm } 10: \quad (0, 10) + (2, 11) \geq 1,\\
&\text{minterm } 11: \quad (2, 11) + (3, 15) \geq 1.
\end{aligned}$$

The optimum solution to this set of equations is of course the choice of the prime implicant (2, 11).

The solution to a 6th order Boolean function minimization problem is shown by the computer output in figure III-3. This function contains a number of don't care terms and the solution shows eight prime implicants, four of which are essential. There are no constraint equations in the solution, which indicates that all minterms are covered by the four essential terms.

Finally, the solution to an 8th order Boolean function minimization problem is shown in Figure III-4. This function contains 202

```
THE BOOLEAN FUNCTION IS OF ORDER  5

MINTERMS OF FUNCTION ARE
    0,    2,    3,    5,    7,    8,   10,   11,   13,   15,   16,   17,
   20,   23,   24,

PRIME IMPLICANTS OF FUNCTION, IN CELLULAR NOTATION, ARE
        (    0,   24)
        (    0,   10)
        (    2,   11)
        (    3,   15)
        (    5,   15)
        (    7,   23)
        (   16,   20)
        (   16,   17)

THE ESSENTIAL PRIME IMPLICANTS ARE
        (    5,   15)
        (   16,   17)
        (   16,   20)
        (    7,   23)
        (    0,   24)

TERMS OF CONSTRAINT EQN FOR MINTERM     2 ARE
        (    0,   10)
        (    2,   11)

TERMS OF CONSTRAINT EQN FOR MINTERM     3 ARE
        (    2,   11)
        (    3,   15)

TERMS OF CONSTRAINT EQN FOR MINTERM    10 ARE
        (    0,   10)
        (    2,   11)

TERMS OF CONSTRAINT EQN FOR MINTERM    11 ARE
        (    2,   11)
        (    3,   15)
```

FIG. III-2. COMPUTER PRINTOUT OF SOLUTION TO 5TH ORDER FUNCTION OF EXAMPLE 2.

```
THE BOOLEAN FUNCTION HAS ORDER  6

MINTERMS OF FUNCTION ARE
    0,    3,    7,   11,   12,   13,   14,   15,   19,   23,   27,   28,
   29,   30,   31,   32,   35,   39,   43,   44,   45,   46,   47,   48,
   49,   50,   51,   52,   53,   54,   55,   56,   57,   59,   60,   61,
   62,   63,

DON'T CARES OF FUNCTION ARE
    0,   11,   13,   23,   30,   32,   43,   47,   51,   54,   61,

PRIME IMPLICANTS OF FUNCTION, IN CELLULAR NOTATION, ARE
        (    0,   32)
        (    3,   63)
        (   12,   63)
        (   32,   48)
        (   48,   61)
        (   48,   55)
        (   49,   63)
        (   52,   63)

THE ESSENTIAL PRIME IMPLICANTS ARE
        (    3,   63)
        (   12,   63)
        (   48,   55)
        (   48,   61)
```

FIG. III-3. COMPUTER PRINTOUT OF SOLUTION TO 6TH ORDER FUNCTION OF EXAMPLE 3.

```
THE BOOLEAN FUNCTION HAS ORDER  8

MINTERMS OF FUNCTION ARE
    0,   10,   11,   12,   13,   14,   15,   17,   20,   21,   23,   25,
   26,   27,   28,   29,   30,   31,   32,   34,   35,   38,   39,   42,
   43,   44,   45,   46,   47,   48,   49,   53,   54,   58,   59,   60,
   61,   62,   63,   64,   65,   66,   70,   71,   72,   73,   74,   75,
   76,   77,   78,   79,   84,   85,   86,   87,   90,   91,   92,   93,
   94,   95,   98,   99,  100,  101,  102,  106,  107,  108,  109,  110,
  111,  114,  115,  116,  117,  118,  119,  122,  123,  124,  125,  126,
  127,  132,  133,  134,  135,  136,  137,  138,  139,  140,  141,  142,
  143,  151,  152,  153,  154,  155,  156,  157,  158,  159,  160,  161,
  162,  163,  164,  165,  166,  167,  168,  169,  170,  171,  172,  173,
  174,  175,  176,  177,  178,  179,  180,  181,  182,  183,  184,  185,
  186,  187,  188,  189,  190,  191,  192,  193,  194,  195,  196,  197,
  198,  199,  200,  201,  202,  203,  204,  205,  206,  207,  208,  209,
  210,  211,  212,  213,  214,  215,  216,  217,  218,  219,  220,  221,
  222,  223,  224,  225,  226,  227,  228,  229,  230,  231,  232,  233,
  234,  235,  236,  237,  238,  239,  240,  241,  242,  243,  244,  245,
  246,  247,  248,  249,  250,  251,  252,  253,  254,  255,

DON'T CARES OF FUNCTION ARE
    0,   10,   11,   12,   13,   14,   15,   26,   27,   28,   29,   30,
   31,   42,   43,   44,   45,   46,   47,   58,   59,   60,   61,   62,
   63,   74,   75,   76,   77,   78,   79,   90,   91,   92,   93,   94,
   95,  106,  107,  108,  109,  110,  111,  122,  123,  124,  125,  126,
  127,  138,  139,  140,  141,  142,  143,  154,  155,  156,  157,  158,
  159,  160,  161,  162,  163,  164,  165,  166,  167,  168,  169,  170,
  171,  172,  173,  174,  175,  176,  177,  178,  179,  180,  181,  182,
  183,  184,  185,  186,  187,  188,  189,  190,  191,  192,  193,  194,
  195,  196,  197,  198,  199,  200,  201,  202,  203,  204,  205,  206,
  207,  208,  209,  210,  211,  212,  213,  214,  215,  216,  217,  218,
  219,  220,  221,  222,  223,  224,  225,  226,  227,  228,  229,  230,
  231,  232,  233,  234,  235,  236,  237,  238,  239,  240,  241,  242,
  243,  244,  245,  246,  247,  248,  249,  250,  251,  252,  253,  254,
  255,

PRIME IMPLICANTS OF FUNCTION, IN CELLULAR NOTATION, ARE
        (    0,   64)
        (    0,   32)
        (   10,  255)
        (   12,  255)
        (   17,   53)
        (   17,   29)
        (   20,   93)
        (   21,  125)
        (   21,   95)
        (   23,  223)
        (   25,  159)
        (   32,  176)
        (   32,  162)
        (   34,  238)
        (   34,  235)
        (   34,  175)
        (   38,  254)
        (   48,  177)
        (   49,  181)
```

FIG. III-4. SOLUTION TO 8TH ORDER FUNCTION OF EXAMPLE 4.

```
(   53,  253)
(   64,  202)
(   64,  201)
(   66,  238)
(   70,  254)
(   70,  223)
(   72,  207)
(   84,  255)
(   98,  254)
(   98,  251)
(  100,  254)
(  100,  253)
(  114,  255)
(  132,  239)
(  135,  255)
(  136,  255)
(  160,  255)
(  192,  255)


THE ESSENTIAL PRIME IMPLICANTS ARE
(   20,   93)
(   34,  175)
(   38,  254)
(   64,  201)
(   70,  223)
(  100,  253)
(  132,  239)
(  136,  255)

TERMS OF CONSTRAINT EQN FOR MINTERM     17 ARE
(   17,   53)
(   17,   29)

TERMS OF CONSTRAINT EQN FOR MINTERM     23 ARE
(   [illegible]
(   23,  223)

TERMS OF CONSTRAINT EQN FOR MINTERM     25 ARE
(   17,   29)
(   25,  159)

TERMS OF CONSTRAINT EQN FOR MINTERM     32 ARE
(    0,   32)
(   32,  176)
(   32,  162)

TERMS OF CONSTRAINT EQN FOR MINTERM     48 ARE
(   32,  176)
(   48,  177)

TERMS OF CONSTRAINT EQN FOR MINTERM     49 ARE
(   17,   53)
(   48,  177)
(   49,  181)

TERMS OF CONSTRAINT EQN FOR MINTERM     53 ARE
(   17,   53)
(   21,  125)
(   49,  181)
```

FIG. III-4. (CON'T)

```
(   53,  253)

TERMS OF CONSTRAINT EQN FOR MINTERM    66 ARE
(   64,  202)
(   66,  238)

TERMS OF CONSTRAINT EQN FOR MINTERM    98 ARE
(   34,  238)
(   34,  235)
(   66,  238)
(   98,  254)
(   98,  251)

TERMS OF CONSTRAINT EQN FOR MINTERM    99 ARE
(   34,  235)
(   98,  251)

TERMS OF CONSTRAINT EQN FOR MINTERM   114 ARE
(   98,  254)
(   98,  251)
(  114,  255)

TERMS OF CONSTRAINT EQN FOR MINTERM   115 ARE
(   98,  251)
(  114,  255)

TERMS OF CONSTRAINT EQN FOR MINTERM   119 ARE
(   84,  255)
(  114,  255)

TERMS OF CONSTRAINT EQN FOR MINTERM   151 ARE
(   23,  223)
(  135,  255)
```

FIG. III-4. (CON'T)

minterms, 157 of which are don't care terms. The computer output lists 37 prime implicants, 8 essential prime implicants, and terms for the constraint equations for minterms 17, 23, 25, 32, 48, 49, 53, 66, 98, 99, 114, 115, 119 and 151, which are not covered by the essential terms. The essential terms therefore cover 31 of the 45 minterms of the function which are not don't care terms. Prime implicants must be chosen which cover the remaining 14 minterms with minimum cost. These prime implicants are easily calculated to be (17, 53), (23, 223), (25, 159), (32, 176), (66, 238), (84, 255), and (98, 251).

IV. SUMMARY

The representation of a Boolean function in terms of the vertices of the cellular n-dimensional cube offers many advantages in a computer solution to the Boolean function minimization problem. This concept was used by Carroll and Jordan in an earlier report, "A Fast Algorithm for Boolean Function Minimization." In this report, the computer algorithm has been modified so as to handle a wider range of Boolean functions and the original FORTRAN program has been converted to a PL/I source program which is computible with the IBM 360 system.

A thorough explanation of the computer algorithm has been given and computer outputs for a number of different Boolean function minimization problems have been illustrated. The final example showed the solution to a nontrivial 8th order minimization problem in which a total of 202 minterms and don't care terms were handled. To give the reader some idea of the speed of the computer algorithm, we note that this 8th order function was processed using 3 minutes 12.87 seconds computer time. If we assume that approximately 45 seconds of this time was required to compile the program, then the actual time required for computation was approximately two and one-half minutes.

It is felt that the computer algorithm described in this report would be very appropriate for use in any automated logic design procedure. The program is both fast and flexible and could be used with any computer system capable of handling PL/I source programs.

REFERENCES

1. W. V. Quine, "The Problem of Simplifying Truth Functions," American Math. Monthly, vol. 59, pp. 521-531, October 1952.

2. W. V. Quine, "A Way to Simplify Truth Functions," American Math. Monthly, vol. 62, pp. 621-631, Nov. 1955.

3. E. J. McCluskey, Jr., "Minimization of Boolean Functions," Bell System Tech. Journal, vol. 35, Nov. 1956.

4. E. J. McCluskey, Jr., Introduction to the Theory of Switching Circuits, New York, McGraw-Hill, 1965.

5. C. C. Carroll and G. E. Jordan, "A Fast Algorithm for Boolean Function Minimization," Project Themis Tech. Report No. AU-T-3, Dec. 1968.

6. R. E. Prather, Introduction to Switching Theory: A Mathematical Approach, Boston, Allyn & Bacon, 1967.

7. H. Mott and C. C. Carroll, "Numerical Procedures for Boolean Function Minimization," IEEETEC, Aug. 1964.

APPENDIX 1

THE CELLULAR STRUCTURE AND PROPERTIES OF BOOLEAN FUNCTIONS

A convenient representation of variables in a Boolean product is a parenthetical exponent representation [6] such that

$$x_i^{(o)} = \bar{x}_i$$

$$x_i^{(1)} = x_i$$

$$x_i^{(I)} = 1 \text{ (or absence of } x_i).$$

Then a Boolean product is

$$p = x_n^{(c_n)} \cdot x_{n-1}^{(c_{n-1})} \cdots x_1^{(c_1)} : c_i \epsilon C \ (i = 1, 2, \cdots, n).$$

This Boolean product will be shown to correspond to a cell in the n-Cube, C^n.

The n-Cube

The partially ordered set $(C^n, \geq)$

$$C^n = \left\{ (c_n, c_{n-1}, \cdots, c_1);\ c_i \in C \right\} \quad (1 \leq n < \infty)$$

with the order relation

$$(c_n, c_{n-1}, \cdots, c_1) \geq (c'_n, c'_{n-1}, \cdots, c'_1) \Longleftrightarrow c_i \geq c'_i$$

in C is called the cellular n-Cube. The set C is the partially ordered set

$$C = \left\{ 0, 1, I \right\}$$

with the order relation $\geq$ defined for C as

$$I \geq 1,\ I \geq 0,\ 1 \geq 1, \text{ and } 0 \geq 0.$$

The order of a cell in C^n is equal to the number of components c_i such that $c_i = I$. Thus a "one cell" contains one component equal to I; a "two cell" contains two components equal to I; etc. A cell with order zero is called a vertex of C^n. Clearly there are 2^n vertices in the n-Cube, C^n.

Two vertices that are extremely useful for representing a cell of C^n are the minimum and maximum vertices of a cell c. The minimum vertex of c where $c = (c_n, c_{n-1}, \cdots, c_1)$ is

$$\min (c) = (v_n, v_{n-1}, \cdots, v_1)$$

where

$$v_i = c_i \text{ if } c_i \in B, \quad B = \{0,1\}$$

$$v_i = 0 \text{ if } c_i = I.$$

The maximum vertex of c is

$$\max (c) = (v_n', v_{n-1}', \cdots, v_1')$$

where

$$v_i' = c_i \text{ if } c_i \in B,$$

$$v_i' = 1 \text{ if } c_i \in I.$$

The Decimal Transform

In order to obtain a useful decimal characterization of the cells of the n-Cube, the idea of the decimal transform is used. Since each cell of the n-Cube can be represented by its minimum and maximum vertices, we may use these coordinates in a two-dimensional representation. The decimal transform is defined as

$$D: C^N \Longrightarrow Z_2n \times Z_2n,$$

such that

$$D(c) = (\partial(\min (c)), \partial(\max (c))).$$

The function ∂ is defined as

$$\partial: B^n \Longrightarrow Z_2n$$

such that

$$\partial(b_n, b_{n-1}, \cdots, b_1) = \sum_{i=1}^{n} b_i 2^{i-1}$$

Example: Consider the cell c = (0, I, 1, I, 0, 1) in C^6.

$$D(c) = (\partial(\min(c)), \partial(\max(c)))$$

$$= (\partial(0, 0, 1, 0, 0, 1), \partial(0, 1, 1, 1, 0, 1))$$

$$= (9, 29)$$

It is seen that if all the vertices of c are determined then $\partial(\min(c)) \leq \partial(v) \leq \partial(\max(c))$ for every $v \subseteq c$. The vertices of c are:

$$\{(0,0,1,0,0,1),\ (0,0,1,1,0,1),\ (0,1,1,0,0,1),\ (0,1,1,1,0,1)\}$$

$\partial(0,0,1,0,0,1) = 9$

$\partial(0,0,1,1,0,1) = 13$

$\partial(0,1,1,0,0,1) = 25$

$\partial(0,1,1,1,0,1) = 29.$

It is seen that $9 < 13 < 25 < 29$.

In Fig. A-1 the cells of the 4-Cube are plotted on a two-dimensional grid. This chart is useful as an aid to the understanding of the theorems presented and the computer algorithms developed in Chapter II.

Some of the properties of the decimal representation of C^n are useful in defining relations existing between cells and vertices. Those properties which are useful in arriving at an algorithm for determining a minimum non-redundant covering of a subset of cells are presented. Operations that are defined will prove to be easily computed on most digital computers.

The AND operation "·" is defined for the set B = {0,1} as:

$$b_i \cdot b_j = 1 \text{ if } b_i = b_j = 1$$

otherwise,

$$b_i \cdot b_j = 0$$

This corresponds to the operation table as follows:

·	0	1
0	0	0
1	0	1

The operation "$\wedge$" is defined for B^n as

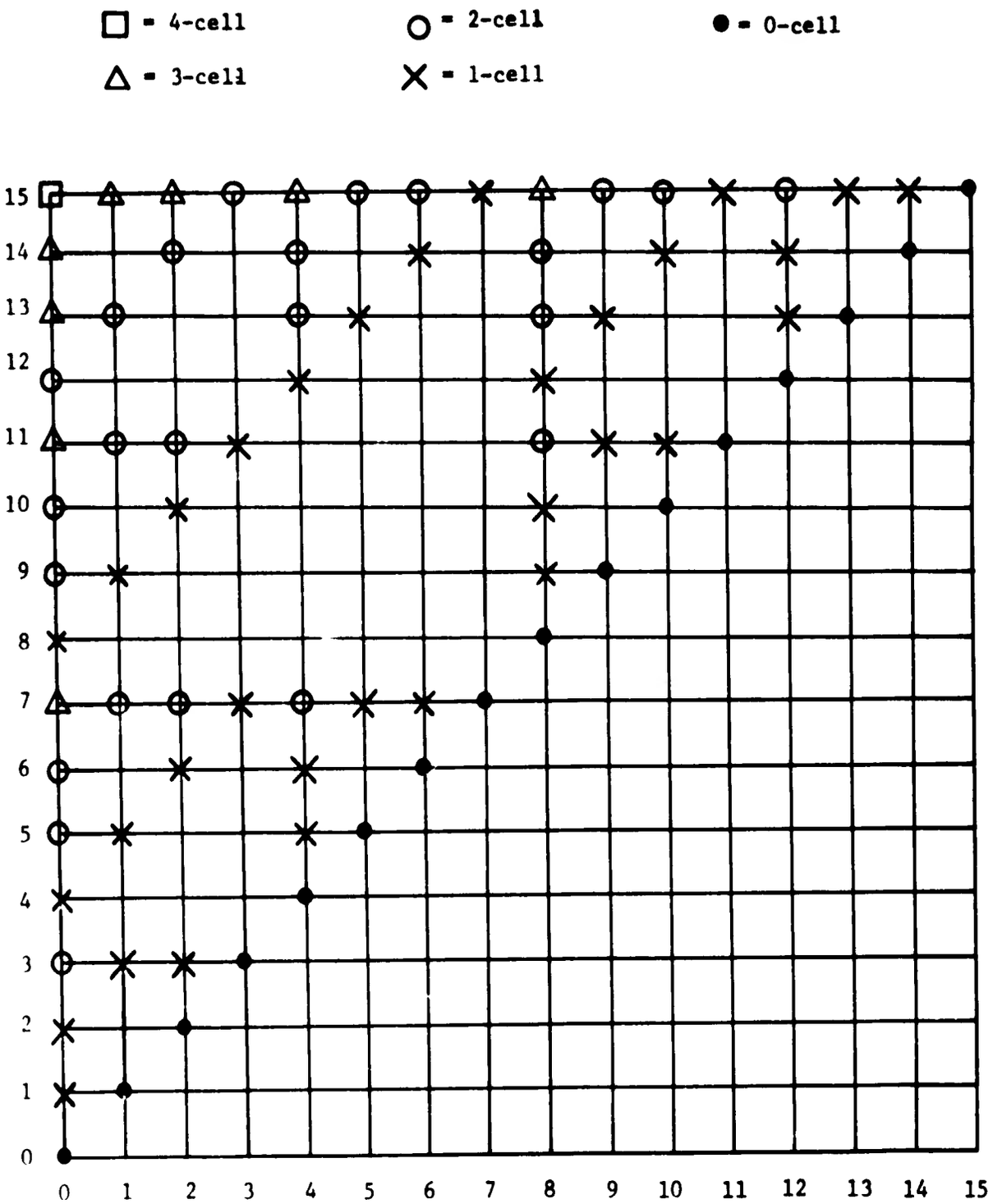


FIG. A-1. TWO-DIMENSIONAL CHART OF C^4.

$$(b_n, b_{n-1}, \cdots, b_1) \wedge (b'_n, b'_{n-1}, \cdots, b'_1)$$

$$= (b_n \cdot b'_n, b_{n-1} \cdot b'_{n-1}, \cdots, b_1 \cdot b'_1)$$

This operation is essentially a logical intersection of two n-element sets, such that if both sets contain a "one" in a given position, then the resulting set will contain a one in that position. This operation is provided as a basic operation on most general purpose digital computers.

If two vertices v_1 and v_2 of an n-cube are such that $v_1 \wedge v_2 = v_1$, then define v_1 to be related to v_2 by the relation $\leftarrow$; $v_1 \leftarrow v_2$ can be thought of to mean that v_1 is contained in v_2 since for every coordinate of v_1 equal to 1 the corresponding coordinate of v_2 will be equal to 1.

Theorem 1: If $c \in C^n$, then min (c) $\leftarrow$ max (c).

The above theorem provides a method to determine if some element of $Z_{2^n} \times Z_{2^n}$ maps onto a cell of C^n. By choosing any coordinate pair from $Z_{2^n} \times Z_{2^n}$ and performing the operation $\wedge$ on the coordinates, it can be determined if they are related by the relation $\leftarrow$.

Theorem 2: $v \subseteq c$ iff $v \leftarrow$ max (c) and min (c) $\leftarrow v$.

Theorem 2 provides a method to determine easily which vertices are contained in a cell c given the corresponding element of $Z_{2^n} \times Z_{2^n}$.

Theorem 3: If min (c) $\nleq$ min (c') and max (c') $\nleq$ max (c), then $c \subseteq c'$.

The above theorem gives the criterion for containment of two cells of C^n given the corresponding elements in $Z_{2^n} \times Z_{2^n}$.

Theorem 4: If $B_1 \nleq B_2$ then $\partial(B_1) \leq \partial(B_2)$

This theorem states the requirement that for one vertex to contain another, its decimal transform must be greater than the contained vertex.

Lemma 4.1: If $c_1 \supseteq c_2$ then $\partial(\min(c_1)) \leq \partial(\min(c_2))$ and

$$\partial(\max(c_1)) \geq \partial(\max(c_2)).$$

This Lemma provides an ordering of the cell which will greatly reduce the number of comparisons that will have to be made to eliminate cells that are contained in other cells.

APPENDIX 2

SOURCE PROGRAM

The following five pages show the complete PL/I source program listing for the Boolean function minimization program described in this report.

```
BOOLMIN0 PROC OPTIONS(MAIN);

   /*****N IS THE ORDER OF THE FUNCTION*************************************/
   /*****M IS THE NUMBER OF MINTERMS AND DON'T CARES IN THE FUNCTION*******/
   /*****D IS THE NUMBER OF DON'T CARES IN THE FUNCTION********************/
   /*****MINT CONTAINS THE MINTERMS AND DON'T CARES OF THE FUNCTION********/
   /*****DONT CONTAINS THE DON'T CARES OF THE FUNCTION*********************/
   /*****(II,JJ) ARE THE COORDINATE PAIRS BEING CHECKED********************/
   /*****(PRIMI,PRIMJ) ARE THE PRIME IMPLICANTS OF THE FUNCTION************/
   /*****(ESSENTI,ESSENTJ) ARE THE ESSENTIAL PRIME IMPLICANTS**************/

DCL (M,N,IZ,D) FIXED DEC
DCL (MINT,NUM,IR) (M) FIXED DEC CONTROLLED;
DCL DONT(D) FIXED DEC CONTROLLED;
DCL C(2**N) FIXED DEC(1,0) CONTROLLED;
DCL (PRIMI,PRIMJ,ESSENTI,ESSENTJ) (M) BIT(17) CONTROLLED;
DCL (III,JJJ) BIT(21);
DCL (III,JJJ,CCP,DIF) BIT(17);

   /*****GET DATA***********************************************************/

        ON ENDFILE(SYSIN) GO TO NOF;
NEWF:   GET LIST(N,M,D);
        ALLOCATE MINT;
        GET LIST((MINT(I) DO I=1 TO M));
        IF D=0 THEN GO TO PROG;
        ELSE DO;
        ALLOCATE DONT;
        GET LIST((DONT(I) DO I=1 TO D));
        END;
PROG:   MP=0;
        EMP=0;

   /*****PRINT THE ORDER OF THE FUNCTION, THE MINTERMS OF THE FUNCTION, ****/
   /*****AND THE DON'T CARES OF THE FUNCTION*******************************/

        PUT EDIT('THE BOOLEAN FUNCTION HAS ORDER ',N) (PAGE,A(21),F(2));
        PUT EDIT('MINTERMS OF FUNCTION ARE') (SKIP(3),A(24));
        K=0;
        DO I=1 TO M;
        IF I=(12*K)+1 THEN DO;
        PUT EDIT(MINT(I),',') (SKIP(1),F(5),A(1));
        K=K+1;
        END;
        ELSE PUT EDIT(MINT(I),',') (F(5),A(1));
        END;
        IF D=0 THEN GO TO HEAD;
        ELSE DO;
        PUT EDIT('DON''T CARES OF FUNCTION ARE   ') (SKIP(3),A(30));
        K=0;
        DO I=1 TO D;
        IF I=(12*K)+1 THEN DO;
        PUT EDIT(DONT(I),',') (SKIP(1),F(5),A(1));
        K=K+1;
        END;
        ELSE PUT EDIT(DONT(I),',') (F(5),A(1));
        END;
```

FIG. A-2. PL/I SOURCE PROGRAM.

```
        END;
HEAD:   PUT EDIT('PRIME IMPLICANTS OF FUNCTION, IN CELLULAR NOTATION, ARE')
        (SKIP(3),A(60));

   /*****SET VERTICES OF C CORRESPONDING TO MINTERMS EQUAL TO ONE, ********/
   /*****VERTICES OF C CORRESPONDING TO DON'T CARES EQUAL TO MINUS ONE,****/
   /*****AND ALL OTHER VERTICES EQUAL TO ZERO*****************************/

        ALLOCATE C;
        J=1;
        K=1;
        DO I=0 TO (2**N)-1;
        IF J>M THEN GO TO NCH;
        IF I=MINT(J) THEN GO TO VERT;
NCH:    C(I)=0;
        GO TO NCVERT;
VERT:   IF K>D THEN GO TO MVERT;
        IF MINT(J)¬=DCNT(K) THEN GO TO MVERT;
        C(I)=-1;
        K=K+1;
        J=J+1;
        GO TO NCVERT;
MVERT:  C(I)=1;
        J=J+1;
NCVERT: END;

   /*****CHOOSE COORDINATE PAIR (II,JJ)***********************************/

        FREE DONT;
        ALLOCATE NUM;
        ALLOCATE PRIMI;
        ALLOCATE PRIMJ;
        ALLOCATE ESSENTI;
        ALLOCATE ESSENTJ;
        DO I=1 TO M;
        IIT=MINT(I);
        II=SUBSTR(IIT,1,17);
        M1=M-1;
        DO J=0 TO M1;
        J1=M-J;
        JJT=MINT(J1);
        JJ=SUBSTR(JJT,1,17);

   /*****CHECK IF COORDINATE PAIR IS A CELL********************************/

        IF BOOL(II,JJ,1)¬=II THEN GO TO NOCELL;

   /*****CALCULATE VERTICES OF CELL***************************************/

        DIF=BOOL(JJ,II,6);
        IF DIF=0 THEN GO TO IMPCHEK;
        ICNT=1;
        NUM(1)=II;
        DCP='00CCCCCCCCCCCCCC01'B;
        DO K=0 TO N-1;
        IF K=0 THEN GO TO DCPOK1;
        ELSE DCP=SUBSTR(DCP,2,16)||'0'B;
```

FIG. A-2. (CON'T)

```
DCPCK1:   IZ=BCOL(DCP,DIF,1);
          IF IZ=0 THEN GO TO NDCP;
          IC2=ICNT;
          DO L=1 TO IC2;
          ICNT=ICNT+1;
          NUM(ICNT)=NUM(L)+IZ ;

   /*****CHECK IF CELL IS IN FUNCTION*************************************/

          IF C(NUM(ICNT))=0 THEN GO TO NOCELL;
          END;
NDCP:     END;

   /*****CHECK IF CELL IS COVERED BY ONE IN PRIME IMPLICANT TABLE*********/

IMPCHEK:  IF MP=0 THEN GO TO INCRMP ;
          DO L=1 TO MP;
          IF PRIMJ(L)<JJ THEN GO TO NOTCON;
          IF BCOL(PRIMJ(L),JJ,1)¬=JJ THEN GO TO NOTCON ;
          IF BCOL(PRIMI(L),II,1)=PRIMI(L) THEN GO TO NOCELL ;
NOTCON:   END;
INCRMP:   MP=MP+1;

   /*****PRINT PRIME IMPLICANT********************************************/

          PUT EDIT('(',II,',',JJ,')') (COLUMN(9),A(1),F(5),A(1),F(5),A(1));
          PRIMI(MP)=II;
          PRIMJ(MP)=JJ;
NOCELL:   END;

   /*****CHECK IF II IS INCLUDED IN AN ESSENTIAL TERM**********************/

          IF C(II)<0 THEN GO TO RECYCLE ;

   /*****CHECK FOR ESSENTIAL TERMS FOR VERTEX II**************************/

          IC=0 ;
          DO L=1 TO MP ;
          IF PRIMJ(L)<II THEN GO TO ONCV;
          IF BCOL(PRIMI(L),II,1)¬=PRIMI(L) THEN GO TO ONCV ;
          IF BCOL(PRIMJ(L),II,1)¬=II THEN GO TO ONCV;
          IC=IC+1;
          IIESS=L;
ONCV:     END ;
          IF IC >1 THEN GO TO RECYCLE;
          IF IC=1 THEN DO;
          EMP=EMP+1 ;
          ESSENTI(EMP)=PRIMI(IIESS);
          ESSENTJ(EMP)=PRIMJ(IIESS);
          END ;

   /*****CHECK OFF ALL VERTICES OF ESSENTIAL TERM**************************/

          NUM(1)=PRIMI(IIESS);
          C(PRIMI(IIESS))=-1;
          ICNT=1;
          DIF=BCOL(PRIMJ(IIESS),PRIMI(IIESS),6);
```

FIG. A-2. (CON'T)

```
        IF DIF=0 THEN GO TO RECYCLE;
        DCP='0000000000000001'B;
        DO K=0 TO N-1;
        IF K=0 THEN GO TO DCPOK2;
        ELSE DCP=SUBSTR(DCP,2,16)||'0'B;
DCPOK2: IZ=BCOL(DCP,DIF,1);
        IF IZ=0 THEN GO TO NDCP2;
        ICT2=ICNT;
        DO L=1 TO ICT2;
        ICNT=ICNT+1;
        NUM(ICNT)=NUM(L)+IZ;
        C(NUM(ICNT))=-1;
        END;
NDCP2:  END;
RECYCLE: END;

   /*****PRINT ESSENTIAL PRIME IMPLICANTS**********************************/

        IF EMP=0 THEN PUT EDIT('THERE ARE NO ESSENTIAL PRIME IMPLICANTS')
        (SKIP(3),A(40));
        ELSE PUT EDIT('THE ESSENTIAL PRIME IMPLICANTS ARE') (SKIP(3),A(40));
        DO I=1 TO EMP;
        PUT EDIT('(',ESSENTI(I),',',ESSENTJ(I),')');
        (COLUMN(9),A(1),F(5),A(1),F(5),A(1));
        END;

   /*****FIND ALL MINTERMS NOT INCLUDED IN ESSENTIAL PRIME IMPLICANTS******/

        FREE NUM;
        FREE ESSENTI;
        FREE ESSENTJ;
        ALLOCATE IR;
        DO I=1 TO M;
        IIT=MINT(I);
        II=SUBSTR(IIT,1,17);
        IC=0;
        IF C(II)<0 THEN GO TO COVERED;
        ELSE DO L=1 TO MP WHILE(PRIMI(L)<=II);
        IF PRIMJ(L)<II THEN GO TO DNCV2;
        IF BCOL(PRIMI(L),II,1)¬=PRIMI(L) THEN GO TO DNCV2;
        IF BCOL(PRIMJ(L),II,1)¬=II THEN GO TO DNCV2;
        IC=IC+1;
        IR(IC)=L;
DNCV2:  END;

   /*****PRINT TERMS OF CONSTRAINT EQUATION********************************/

        PUT EDIT('TERMS OF CONSTRAINT EQN FOR MINTERM ',II,' ARE')
        (SKIP(2),A(36),F(5),A(4));
        DO L=1 TO IC;
        PUT EDIT('(',PRIMI(IR(L)),',',PRIMJ(IR(L)),')')
        (COLUMN(9),A(1),F(5),A(1),F(5),A(1));
        END;
COVERED: END;

   /*****FREE STORAGE AND GET NEXT FUNCTION OR END EXECUTION***************/
```

FIG. A-2. (CON'T)

```
          FREE MINT;
          FREE PRIMI;
          FREE PRIMJ;
          FREE C ;
          FREE IR ;
          GO TC NEWF;
NCF:      END BCOLMIN;
```

FIG. A-2. (CON'T)

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)
Vice President for Research
Auburn University
Auburn, Alabama 36830

2a. REPORT SECURITY CLASSIFICATION
Unclassified

2b. GROUP
NA

3. REPORT TITLE
An Algorithm for Fast Boolean Function Minimization Using Properties of the Cellular N-Cube

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical report

5. AUTHOR(S) (First name, middle initial, last name)
Chester C. Carroll
William A. Hornfeck

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
August, 1970	52	7

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
DAAH01-68-C-0246	AU-T-16
b. PROJECT NO. NA	
c. NA	9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)
d. NA	None

10. DISTRIBUTION STATEMENT
Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
None	Army Missile Command

13. ABSTRACT

Properties of the cellular n-cube representation are used to advantage in developing a fast algorithm for finding the prime implicants of a Boolean function. The algorithm is discussed and several examples are included showing computer solutions to selected Boolean function minimization problems. The complete PL/I source program listing for the automated algorithm is included.

DD FORM 1 NOV 68 1473 REPLACES DD FORM 1473, 1 JAN 64, WHICH IS OBSOLETE FOR ARMY USE.

UNCLASSIFIED

Security Classification

UNCLASSIFIED
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Logic design						
Boolean function minimization						
Cellular n-cube						
Prime implicants						