$x_1 \backslash x_2$	0	1	2
0	0	0	1
1	0	1	1
2	1	1	2

(a)

$x_1 \backslash x_2$	0	1	2
0	0	1	1
1	1	1	1
2	1	1	2

(b)

x	F(x)
0	1
1	0
2	2

(c)

Fig. 5. $MT^*(3)$ elements for binary (a) AND*, (b) OR*, and (c) NOT* gates.

TABLE I

Decimal Equivalent	x_1	x_2	x_3	x_4	F	f_1	f_3	f_5	f_{15}
0	0	0	0	0	0	1	1	1	2
1	0	0	0	1	0	1	1	1	2
2	0	0	1	0	0	1	1	1	2
3	0	0	1	1	1	0	1	1	2
4	0	1	0	0	0	1	1	1	2
5	0	1	0	1	0	1	1	1	2
6	0	1	1	0	0	1	1	1	2
7	0	1	1	1	1	0	1	1	2
8	1	0	0	0	0	1	1	1	2
9	1	0	0	1	0	1	1	1	2
10	1	0	1	0	0	1	1	1	2
11	1	0	1	1	1	0	1	1	2
12	1	1	0	0	1	1	0	1	2
13	1	1	0	1	1	1	0	1	2
14	1	1	1	0	1	1	0	1	2
15	1	1	1	1	0	0	0	1	2

Next we should consider multiple s-a-2 faults. It is easy to show that one additional column should, then, be included in Table I covering the functions $f_{1,3} = f_{1,4} = f_{2,3} = f_{2,4} = 0$ (where $f_{i,j}$ results from w_i and w_j both s-a-2). All other multiple s-a-2 faults are accounted for by the functions f_1, f_3, f_5, and f_{15}. One more test becomes necessary and it can be chosen from the set {3, 7, 11, 12, 13, 14}.

From Theorem 2 it follows that all s-a-$\bar{2}$ faults can be detected using the input vector 2222. Therefore all multiple faults within the network are detectable with the test set {0000, 0011, 2222}.

It should be pointed out that the same network realized with Boolean gates requires four tests for single fault detection and five tests for multiple fault detection [7].

V. Conclusions

The preceding discussion suggested the use of additional logic values for fault detection in $MT(R)$ networks. This approach greatly reduces the number of tests that must be applied. It also simplifies the process of deriving adequate test sets.

The idea of using higher valued circuits to improve testability of binary networks was shown to be potentially promising in both sequential and combinational networks [8]. However, the resultant overhead in such cases can be considerable. The cost of additional circuitry becomes less important when circuits with larger radices are to be tested.

While little work has been done on the subject of testing many-valued circuits, it is important to recognize the many possibilities offered by such circuits, that do not arise in binary cases.

References

[1] S. Y. H. Su, "Symposium chairman's message," in *Proc. 6th Int. Symp. Multiple-Valued Logic*, Logan, UT, May 1976.
[2] R. J. Spillman and S. Y. H. Su, "Detection of single, stuck-type failures in multivalued combinational networks," *IEEE Trans. Comput.*, vol. C-26, pp. 1242–1251, Dec. 1977.
[3] D. R. Haring, "Multi-threshold threshold elements," *IEEE Trans. Electron. Comput.*, vol. EC-15, pp. 45–65, Feb. 1966.
[4] A. Druzeta, Z. G. Vranesic, and A. S. Sedra, "Application of multi-threshold elements in the realization of many-valued logic networks," *IEEE Trans. Comput.*, vol. C-23, pp. 1194–1198, Nov. 1974.
[5] T. T. Dao, L. K. Russell, D. R. Preedy, and E. J. McCluskey, "Multi-level I²L with threshold gates," in *Proc. 1977 IEEE Int. Solid-State Circuits Conf.*, Philadelphia, 1977, pp. 110–111.
[6] N. Friedman, C. A. T. Salama, F. E. Holmes, and P. M. Thompson, "Multivalued integrated-injection-logic (MI²L) full adder," *Electron. Lett.*, vol. 13, pp. 135–136, Mar. 1977.
[7] J. F. Poage, "Derivation of optimal tests to detect faults in combinational circuits," in *Proc. Symp. Mathematical Theory of Automata*, New York, Polytechnic Press, 1965, pp. 483–528.
[8] Z. G. Vranesic, "Multi-valued circuits in fault detection of binary logic circuits," *Microelectron. Rel.*, vol. 15, supplement, pp. 25–33, 1976.

Minimal TANT Networks of Functions with DON'T CARE's and Some Complemented Input Variables

H. A. VINK

***Abstract*—The minimization algorithm of Gimpel realizes a minimal TANT network for any Boolean function under a NAND gate cost criterion. A TANT network is a three-level network composed of AND-NOT (i.e., NAND) gates, having only true (i.e., uncomplemented) input variables.**

This correspondence extends the algorithm of Gimpel such that functions which can be minimized, may also be incompletely specified. It is shown that the incorporation of these DON'T CARE's cannot be done as easy as in the minimization method of Quine–McCluskey. Beside the prime implicants of the completely

Manuscript received December 23, 1975; revised November 17, 1976 and July 11, 1977.

The author is with the Department of Electrical Engineering, Delft University of Technology, Delft, The Netherlands.

0018-9340/78/1100-1073$00.75 © 1978 IEEE

specified switching function, some additional implicants are necessary during the first phase of the minimization method of Gimpel. Rules are given to generate a proper set of implicants. The minimization method of Gimpel is extended with these rules such that a minimal TANT network is obtained.

A second extension of the algorithm of Gimpel concerns the use of complemented input variables. Complemented input variables may further reduce a TANT network. These variables may be easily incorporated but cause an increase of the number of prime permissible implicants. Rules are presented which strongly reduce the number of additional implicants. The minimization method of Gimpel is extended accordingly and gives a minimal "TANT" network.

Index Terms—**DON'T CARE's, implicants, invertors, minimization, gate cost criterion.**

Introduction

In literature many algorithms have been presented which give a minimal or near-minimal network. In a specific design project an algorithm may be selected for every application. This may result in a large number of selected algorithms. A network with near optimal properties can be designed, however, by just selecting a small number of algorithms such that they as a whole are suited for this project. Criteria for the selection of these algorithms may be as follows: 1) the size of the switching function to realize (number of inputs, outputs, wild or structured logic); 2) practical restrictions which can be taken into account (fan-in, fan-out, logic level limitations, type of logic gates to be used); 3) the way the algorithm will be used: hand or computer execution; 4) the size of the possibly in quantity produced circuits; 5) the properties of the realized networks (a minimal or near-minimal number of gates, connections, levels of whatever a designer wants to minimize; hazardless).

Several special NAND algorithms [1]–[10], have been presented. This article extends the algorithm of Gimpel [3] such that DON'T CARE's and some complemented input variables beside the uncomplemented form are fully incorporated and a minimal three level network is obtained. Single output switching functions can be handled with small to intermediate sizes. The algorithm is suitable for hand and computer execution. If fan-in or fan-out restrictions must be solved, then this may be done by Su and Nam [7].

The algorithm of Gimpel is similar to the method of Quine–McCluskey [11], [12], which is one of the basic minimization techniques of logical circuits. Quine–McCluskey realize a two level network of minimal cost, where the cost is either the total number of gates or the total number of gate-inputs. This minimization method assumes that every input variable is available in its true and inverted form; otherwise a Boolean function cannot be implemented as a rule by a two level network. If a circuit does not give the complemented variables besides the true form, then an additional level, consisting solely of invertors, is necessary.

Gimpel [3], [13] has studied the minimization of logical circuits for which no complemented input variables are available. His method applies to three level AND–NOT networks, since three levels are just sufficient for realizing every Boolean function if no complemented input variables are available. Gimpel's algorithm realizes a minimal TANT network, where TANT stands for a three level network composed of AND–NOT gates, having only true input variables.

The method of Quine–McCluskey is easily extended to incompletely specified switching functions, as described in [11], [12], and [13].

This paper extends the minimization algorithm of Gimpel by allowing incompletely specified switching functions. The incorporation of DON'T CARE's proves to be more difficult than in the minimization method of Quine–McCluskey.

A second extension of the minimization method of Gimpel concerns the use of complemented input variables besides the true input variables. Complemented input variables generally reduce a TANT network. Although these variables are easily incorporated they cause an increase of the number of implicants which must be considered. Rules are presented which strongly reduce the number of additional implicants. This last extension of course makes the acronym "TANT" less applicable; the acronym "TAN" which stands for *three level network* composed of AND–NOT gates, would give a better description.

The Minimization Method of Gimpel

This correspondence is based on the definitions given by Gimpel [3]. In his minimization method the following implicants of a given function f, are determined successively:

1) Prime Implicants (PI's);
2) Maximum Permissible Implicants (MPI's);
3) Prime Permissible Implicants (PPI's).

Finally the Covering Closure (CC-table) Table, an extension to the prime implicant table [3], is filled with the augmented expressions of the PPI's.

This correspondence concentrates on the finding of the right inputs of the CC-table. The reduction of the CC-table, resulting in a minimal TANT network, is described by Gimpel [3] who lists the six reduction techniques of Grasselli and Luccio [14].

The minimization of TANT networks can be modified and extended with some reduction rules as presented in [15] and [16].

Incompletely Specified Switching Functions

Let the function regarding the 1-cells be denoted by $f1$, and the function regarding only the DON'T CARE's by $f2$. The prime implicants of the completely specified switching function $f1 + f2$ are the same as those of the incompletely specified switching function f. The prime implicant table contains, however, only the 1-cells of the function $f1$, because it is not necessary to cover the DON'T CARE cells. This prime implicant table is reduced by removal of prime implicants which cover solely DON'T CARE's.

The example of Fig. 1 shows that the algorithm of Gimpel cannot be extended in the same manner as the method of Quine–McCluskey.

1) Prime implicants of the function $f1$ (without DON'T CARE 12):

$$w'x'y',\ w'x'z',\ wxyz'.$$

The TANT network: $w'x'(yz)' \vee wxyz' = w'x'(yz)' \vee wxy(yz)'$ requires 6 NAND's.

2) Prime implicants of the function f (DON'T CARE 12 is regarded as an 1-cell):

$$w'x'y',\ w'x'z',\ wxz'.$$

The TANT network: $w'x'(yz)' \vee wxz'$ requires 7 NAND's. The incorporation of DON'T CARE: 12 causes implicant wxz' to be derived from $wxyz'$. Headfactor "y" of implicant $wxyz'$ offers third level gate sharing of $(yz)'$ and causes a reduction of the TANT network. Implicant wxz' does not offer this possibility, however, because its head does not contain headfactor "y."

Conclusion

The algorithm of Gimpel cannot be extended to incompletely specified switching functions f by just considering the prime implicants of the function f.

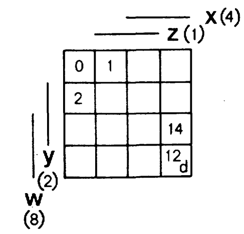

Fig. 1. Function $f(w, x, y, z)$ with DON'T CARE: $wxy'z'$.

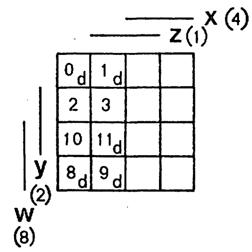

Fig. 2. Function $f(w, x, y, z)$ used to illustrate Theorems 1 and 2.

This extension can be performed, as will be shown, if some implicants of f which are not prime implicants are added to the prime implicants of f. These implicants which are added to the prime implicants of f will be denoted from now with "additional implicants."

In the minimization method of Gimpel, implicants are generated, besides the prime implicants, if these give a compound MPI or if these are necessary because of the third level gate sharing.

1) The additional implicants of an incompletely specified switching function f are not necessary to form a compound MPI because such an MPI will also be derived from the prime implicants of the function f.

a) If from an additional implicant $p1$ of f a prime implicant $p2$ of f may be derived by elimination of headfactors and if $p1$ together with some other implicants $p3, \cdots, pk$ give an MPI, then this MPI will also be derived from $p2$ and the implicants $p3, \cdots, pk$.

Example: Suppose function f, as shown in Fig. 1, is extended with a prime implicant $p3$ such that from $p3$ and wxz' an MPI with head wxy can be derived:

$$wxy \wedge (wxz' \vee p3) = wxyz' \vee wxyp3.$$

The additional implicant of function f: $wxyz'$ has been derived from prime implicant wxz'.

b) If from an additional implicant $p1$ of f a prime implicant $p2$ of f may be derived by elimination of tailfactors, then $p1$ will never be necessary to form an MPI:

Theorem 1: An implicant $p1$ of function f may be omitted if f has an implicant $p2$ which properly includes $p1$ but which has the same head as $p1$.

Appendix I contains the proof of this theorem.

Example: Implicant $x'y$ of the function f, shown in Fig. 2, can be derived from $w'x'y$. Theorem 1 proves that implicant $w'x'y$ may be omitted; it has the same head as $x'y$ but tailfactor w' additionally.

2) The additional implicants may be necessary because of the third level gate sharing. The example of Fig. 1 shows that an implicant which is included in a prime implicant of f and which has additional headfactors comparing with the prime implicant, must be added if it is necessary because of the third level gate sharing. This implicant may be included in a prime implicant of the function $f1$. Gimpel has proven [3] that a permissible implicant which is included in a prime implicant of the function $f1$, may be omitted. An implicant which is included in a prime implicant of f and which has additional headfactors compared with the prime implicant, may therefore be omitted if it is included in a prime implicant of $f1$.

Example: Implicant $w'x'yz'$ (2) of the function f shown in Fig. 1, can be omitted because it is included in $w'x'z'$ (2, 0).

Conclusion

The determination of the prime implicants of the function f must be followed by an addition of implicants which are included in prime implicants of f and have additional headfactors compared with these prime implicants, but which are not included in a prime implicant of $f1$.

This number of additional implicants may, however, be reduced.

Theorem 2: An implicant $p2$ of function f may be omitted if:

a) $p2$ is properly included, considering only the 1-cells, in an implicant $p1$ of f which has some additional headfactors comparing with $p2$ and if

b) $p2$ is properly included, considering the 1-cells and DON'T CARE's, in an implicant $p3$ of f.

Proof:

1) Implicant $p2$ achieves not more than $p1$ and will never require less NAND's because of the third level gate sharing.

2) Implicant $p2$ may give an MPI which, however, also can be derived from implicant $p3$.

3) Implicant $p3$ may give, moreover, an MPI which cannot be derived from $p2$. Q.E.D.

Example: Implicant $p1$: $x'yz$ of function $f1$ of Fig. 2 gives implicants $p2$: $x'z$ and $p3$: x' of f. Implicant $p2$ meets the requirements of Theorem 2 and can be omitted:

1) $x'z$ achieves the same as $x'yz$ and will never require less NAND's because of the third level gate sharing.

2) $x'z$ may give an MPI with a head not containing the variable y; this MPI can, however, also be derived from x'.

3) Implicant x' may give, moreover, an MPI with a head, not containing the variable z.

The determination of the prime implicants of f, will be followed by an addition of some implicants. The prime implicants which

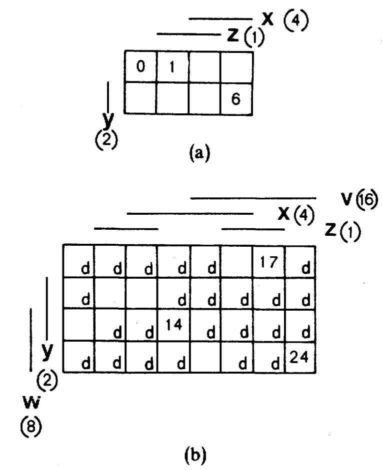

Fig. 3. (a) Specification of the function $f(w, x, y, z)$: $x'y' \vee xyz'$. (b) The extended Karnaugh map of the function shown in Fig. 3(a) with the complemented input variable of y and z.

cover solely DON'T CARE's cannot be omitted; they may give an MPI. After the determination of the MPI's, implicants which cover solely DON'T CARE's can be omitted.

Theorems 1 and 2 result in the following extended algorithm of Gimpel for the determination of the PPI's of an incompletely specified switching function:

1) Determine the prime implicants of f.

2) Adjust those implicants which are formed during phase 1 and which do not fall into one of the following categories. Implicants which

a) are properly included in a prime implicant of $f1$;

b) cover solely DON'T CARE's; and

c) meet the requirements mentioned in Theorems 1 and 2.

3) Determine the MPI's from the implicants of f. The reduction rule of Gimpel concerning an isolated quasi-simple MPI is still applicable if the DON'T CARE's are considered as 1-cells which are covered by other implicants. Consequently this rule is not applicable to an MPI which covers some DON'T CARE's.

4) Determine the PPI's from the MPI's.

5) Determine the augmented principal expressions from the PPI's and fill the CC-table.

Example: The prime implicants and additional implicants of the function, shown in Fig. 2, are determined:

prime implicant x'
additional implicants $x'yz$, $wx'y$, $x'y$

The minimal TANT network: $w'x'(yz)' \vee wxy(yz)'$ requires 6 NAND's.

FUNCTIONS WITH SOME COMPLEMENTED INPUT VARIABLES

The algorithm of Gimpel is extended by the use of some complemented input variables besides the true input variables. This extension will be illustrated with the function, shown in Fig. 3(a). It is assumed that the complemented input variables of y and z is given, these will be denoted with new names: $z' = w$ and $y' = v$.

The Karnaugh map is now 4 times as big as the original one. The cells which are based on the impossible logic combinations: $w'z'$, wz, $v'y'$, vy are considered as DON'T CARE's. Let these "implicants" be denoted with "empty" implicants. The original fundamental products are enlarged with the new variables.

0: $x'y'z'$ gives $vwx'y'z'$: 24
1: $x'y'z$ gives $vw'x'y'z$: 17
6: xyz' gives $v'wxyz'$: 14

The introduction of new variables causes an increase of the number of prime implicants. Gimpel has proven [3] that permissible implicants which are included in a prime implicant, can be omitted. The proof remains unchanged if some tailfactors contain complemented input variables. The implicants of the function shown in the extended Karnaugh map can therefore be determined easily using the symmetry in the extended Karnaugh map from the implicants of the function shown in the original Karnaugh map. This symmetry is introduced by the new variables.

Example:

1) The original prime implicants: $x'y'$ and xyz' give some new prime implicants:

$x'y'$ gives: $x'y'$, vx'
xyz' gives: xyz', wxy, $v'xz'$, $v'wx$.

2) The empty implicants are as follows:

$$w'z',\ wz,\ v'y',\ vy.$$

The introduction of new variables for the complemented input variables gives a straightforward solution of this minimization problem. The MPI's and PPI's are determined from the prime implicants of the expanded Karnaugh map of Fig. 3(b). The CC-table is filled with the augmented expressions of the PPI's after the substitution of the introduced variables by the corresponding complemented input variables and the removal of the empty implicants. The reduction of the CC-table results in a minimal TANT network.

The extension of the minimization method of Gimpel using complemented input variables can easily be performed in principal. Each complemented input variable, however, causes the Karnaugh map to redouble and the number of prime implicants to increase sharply. The determination of the MPI's and PPI's offers therefore an enormous amount of work.

This extension of the minimization method of Gimpel, however, can be strongly simplified. The following theorems which are also based on the introduction of a new name for every complemented input variable, prove that the Karnaugh map need not be expanded. They reduce the number of additional implicants. Moreover, these can be determined from the original prime implicants.

Theorem 3: Implicants which may be expressed as ab, where a is an uncomplemented input variable and b is the new variable introduced for the complemented input variable a (i.e., $b = a'$), can be omitted.

Proof:

1) Implicant ab will never give a compound MPI.

2) Implicant ab does not give PPI's.

3) Implicant ab will be removed after the substitution of the introduced variables by the complemented input variables because ab describes an empty implicant.

Omission of such an implicant ab implies that no MPI should be formed, having a head which is based on ab.

Example: In Fig. 3(a), the empty implicants are $w'z'$, wz, $v'y'$, vy. According to this theorem, wz and vy may be omitted.

Theorem 4: Implicants cannot be omitted if and only if they can be derived from an original implicant by replacing some headfactors by the corresponding introduced variables. An MPI which has a head containing one of the introduced variables, can be omitted.

Theorem 4 is proven in Appendix II.

Example: In Fig. 3(a), implicant xyz' gives: xyz', wxy, $v'xz'$, $v'wx$. According to Theorem 4, the implicants xyw and $v'wx$ can be omitted.

Application of Theorems 3 and 4 to the example of Fig. 3(a) results in the following prime implicants:

$x'y'$ gives: $x'y'$
xyz' gives: xyz' and $v'xz'$.

Theorems 3 and 4 have the following consequences:

1) The introduction of new variables (the origin of the empty implicants) causes that no dominant quasi-simple MPI will appear; such an MPI will never be isolated.

2) The determination of the MPI's and the PPI's requires no extended Karnaugh map or prime implicant table, but only a modified prime implicant table.

Complemented input variables cause the introduction of a new kind of augmented expressions. Let the input variables x and y be given in its primed and unprimed form, then $x'y'$ will have the following augmented expressions: $x'(x'y)'$ and $y'(y'x)'$. Theorem 5 proves that these expressions are also formed if a new variable is introduced for every complemented input variable and the augmented expressions are determined.

Theorem 5: If new names are introduced for complemented input variables of a function f and the PPI's of f are determined, then the augmented expressions will also be generated which have tailfactors being a product of complemented and uncomplemented input variables.

Appendix III contains the proof of this theorem.

Theorem 5 shows that the use of new variables for the complemented input variables causes that every augmented expression of the new kind will be generated. The formation of the augmented expressions on account of the headfactors has not changed.

The following example illustrates an augmented expression of still another kind.

Let function f of variables x, y, and z have one prime implicant: xy'. If the complemented variable of z is given, then the new variable w is introduced with $w = z'$ and the empty implicant $w'z'$. The determination of the MPI with head x gives:

$$xy' \vee xw'z' = x(yz)'(wy)' = x(yz)'(yz')' = xy'.$$

The PPI's of a function f with complemented input variables besides the true input variables, can be determined according to the following algorithm:

1) Determine the prime implicants of f.

2) Define the new variables and derive the new prime implicants.

3) Fill the prime implicant table with the new fundamental products out of that part of the Karnaugh map for which the introduced variables are in the low state.

4) Determine the MPI's in the same was as before, for every fundamental product which is multiply covered.

5) Determine the PPI's.

6) Substitute the original variables and adjust the implicants which are not dominated.

The minimization of a TANT network is illustrated with the function f with 1-cells: 4, 5, 7, and 12. Assume that the complemented variable of z is given.

A minimal TANT network for the function f is $xy' \vee xz(wz)'$.

A minimal "TANT" network for the function f with the complemented input variable z' is $(wz)' \wedge (yz')'$.

Conclusion

The algorithm of Gimpel is extended such that the functions which can be minimized, may also be incompletely specified. These DON'T CARE's may reduce a TANT network further. It is shown that the incorporation of DON'T CARE's in the minimization method of Gimpel cannot be done as easy as in the minimization method of Quine–McCluskey. In the minimization method of Gimpel some additional implicants are necessary besides the prime implicants. Rules are given to generate the proper set of implicants.

A second extension of the algorithm of Gimpel is the use of some complemented input variables besides the true input variables. These variables may further reduce a TANT network. Complemented input variables can easily be incorporated by the introduction of new variables. These variables cause the number of prime permissible implicants to increase. Rules are presented which strongly reduce the number of additional implicants and which give a minimal "TANT" network.

Appendix I

Theorem 1: An implicant $P1$ of function f may be omitted if f has an implicant $P2$ which properly includes $P1$ but which has the same head as $P1$.

Proof: The implicants $P1$ and $P2$ can be expressed as a product of a headterm and tailfactors: $HT1'T2' \cdots$. Here H is a product of uncomplemented input variables and each Ti is an input variable. Assume that $P2$: $HT1'T2' \cdots Tn'$ is derived from $P1$: $HT1'T2' \cdots Tn'Tn + 1'$, on account of DON'T CARE's. A realization of a function g with implicant $P1$ is compared with a realization of g with implicant $P2$.

1) Implicant $P1$ will never require less NAND's than $P2$ because of the third level gate sharing.

2) $P1$ and $P2$ may be combined with several other implicants: $P3$, $P4$, $\cdots$, Pm. The resulting compound MPI1 and MPI2 are compared with each other. The PPI's which are derived from MPI1 will also be derived from MPI2, or will be dominated by PPI's derived from MPI2 so MPI2 will never offer a more expensive solution.

3) Implicant $P2$ may give a MPI with a head containing the variable $Tn + 1'$, whereas $P1$ does not.

4) Formation of an MPI with a head different from H does not change the proof.

5) If implicant $P2$ differs in more than one tailfactor from $P1$, than the comparison of MPI1 with MPI2 leads to the same conclusion. Q.E.D.

Appendix II

Theorem 4: Implicants cannot be omitted if and only if they can be derived from an original implicant by replacing some headfactors by the corresponding introduced variables. An MPI which has a head containing one of the introduced variables can be omitted.

Proof: The symmetric, extended Karnaugh map shows that implicants which can be derived from an original implicant by replacing some headfactors by the corresponding introduced variables, must be added.

The following part contains the proof that a prime implicant which is derived from an original prime implicant, may be omitted if it has an introduced variable as headfactor. Moreover, an MPI which has a head containing one of the introduced variables, can be omitted. Two MPI's are defined and compared with each other.

MPI1 has a head containing some introduced variables and is determined from the prime implicants of the extended Karnaugh map.

MPI2 has the same head as MPI1 but without the introduced variables and is determined from the reduced list of prime implicants, according to Theorems 3 and 4.

MPI1 will now be dominated by MPI2:

1) If MPI1 has a head containing no introduced variables, then only the prime implicants of the reduced list are combined to form an MPI.

2) The proof is shown for an MPI with one introduced variable as headfactor. The method of the proof remains unchanged for every other combination of headfactors.

Assume function f is given with the complemented input variable of $v1$. Introduce the variable $v2$ with $v2 = v1'$. Two MPI's with the same head are determined. MPI1 has a head containing solely the introduced variable $v2$ and is determined from the prime implicants of the extended Karnaugh map. MPI2 has the same head as MPI1 but without the introduced variable $v2$ and is determined from the reduced list of prime implicants, according to Theorems 3 and 4. The prime implicants which will be combined to form the MPI's are partitioned into four distinct groups.

Prime implicants (PI's) of MPI1 are multiplied with the variable $v2$, because MPI1 is determined having a head containing the variable $v2$. The MPI's are compared with each other: Every PPI derived from MPI1 will also be derived from MPI2.

Q.E.D.

Appendix III

Theorem 5: If new names are introduced for complemented input variables of a function f and the PPI's are determined, then the augmented expressions will also be generated which have tailfactors being a product of complemented and uncomplemented variables.

Proof: Assume function f is given with its complemented variable $v1$. Introduce the new variable $v2$ with $v2 = v1'$. Combine the implicants with head H, except for the empty implicant $v1'v2'$. The resulting product A of tailfactors will now be combined with $H \wedge v1'v2'$ and gives product B. Assume that $T1$ is a product of variables not containing $v1$ or $v2$.

1) If a tailfactor of A contains the variable $v1$ or $v2$, then B will contain the same tailfactor.

2) If a tailfactor $(T1)'$ of A does not contain the variables $v1$ or $v2$ then B will contain the tailfactors $(T1 \wedge v1)'$ and $(T1 \wedge v2)'$.

After substitution of $v2$ by $v1'$, the concerning augmented expressions will arise. Q.E.D.

Acknowledgment

The author wishes to thank Dr. G. A. Blaauw for his continuous help in revising this correspondence.

References

[1] G. A. Maley and J. Earle, *The Logic Design of Transistor Digital Computers.* Englewood Cliffs, NJ: Prentice-Hall, 1963, ch. 6, pp. 114–159.
[2] D. T. Ellis, "A synthesis of combinational logic with NAND or NOR elements," *IEEE Trans. Electron. Comput.*, vol. EC-14, pp. 701–705, Oct. 1965.
[3] J. F. Gimpel, "The minimization of TANT networks," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 18–38, Feb. 1967.
[4] D. L. Dietmeyer and Y. H. Su, "Logic design automation of fan-in limited NAND networks," *IEEE Trans. Comput.*, vol. C-18, pp. 11–22, Jan. 1969.
[5] E. S. Davidson, "An algorithm for NAND decomposition under network constraints," *IEEE Trans. Comput.*, vol. C-18, pp. 1098–1109, Dec. 1969.
[6] K. K. Chakrabarti, A. K. Choudhury, and M. S. Basu, "Complementary function approach to the synthesis of three-level NAND network," *IEEE Trans. Comput.*, vol. C-19, pp. 509–514, June 1970.
[7] S. Y. H. Su and C. W. Nam, "Computer-aided synthesis of multiple-output multilevel NAND networks with fan-in and fan-out constraints," *IEEE Trans. Comput.*, vol. C-20, pp. 1445–1455, Dec. 1971.
[8] J. Frackowiak, "The synthesis of minimal hazardless TANT networks," *IEEE Trans. Comput.*, vol. C-21, pp. 1099–1108, Oct. 1972.
[9] H. C. Torng, *Switching Circuits. Theory and Logical Design.* Reading, MA: Addison-Wesley, 1972, ch. 8, pp. 103–130.
[10] S. Yajima and K. Inagaki, "Power minimization problems of logic networks," *IEEE Trans. Comput.*, vol. C-23, pp. 153–165, Feb. 1974.
[11] D. D. Givone, *Introduction to Switching Circuit Theory.* New York: McGraw-Hill, 1970.
[12] E. J. McCluskey, *Introduction to the Theory of Switching Circuits.* New York: McGraw-Hill, 1965.
[13] G. A. Blaauw, "Digitale techniek I," Twente University of Technology, P.O. Box 217, EF-library, Enschede, The Netherlands, no. 1261.0525.
[14] A. Grasselli and F. Luccio, "A method for minimizing the number of internal states in incompletely specified sequential networks," *IEEE Trans. Electron. Comput.*, vol. EC-14, pp. 535–541, Aug. 1965.
[15] H. A. Vink, "The analysis and synthesis of minimal TANT networks," Twente University of Technology, P.O. Box 217, EF-library, Enschede, The Netherlands, no. 1261.1545.
[16] H. A. Vink, B. van den Dolder, and J. Al, "Reduction of CC-tables using multiple implication," *IEEE Trans. Comput.*, to be published.

Decomposition of Polygons into Convex Sets

BRUCE SCHACHTER

***Abstract*—A method is presented for decomposing polygons into convex sets. The method is based upon a Delaunay tessellation of the polygon. It is implemented as a divide-and-conquer technique.**

***Index Terms*—Pattern recognition, polygon decomposition, tessellation.**

I. Introduction

This correspondence describes a method for decomposing polygons into convex sets. We are requiring that the edges of the decomposition start and end at vertices (Fig. 1).

The decomposition of polygons has a number of practical

Manuscript received December 19, 1977; revised March 8, 1978.
The author is with General Electric Co., Daytona Beach, FL 32015.