MINIMIZATION OF SWITCHING FUNCTIONS

Layla Abou-Hadeed Abd-Allah
Department of Computer Science, Faculty of Engineering,
Alexandria University, Alexandria, Egypt.

ABSTRACT

This paper introduces a simple approach for absolute minimization for a single-output function. The approach is based on the classic approach; it consists of finding all the prime implicants of the function and selecting a set of them as a minimum sum of the function. Unlike Quine-McCluskey(Q-M)[2], the minterms are tabulated in their natural order. N passes are required for a function of N variables. Finding the prime implicants of the given function is based on partitioning the problem into smaller problems. In pass one, the problem is partitioned into $2^{(n-1)}$ problems of 2^1 minterms. Pass one finds the prime implicants for all these problems. In general, the output of pass (i-1) is input to pass i which solves $2^{(n-i)}$ problems of 2^i minterms. Consensus and subsumption operations are applied to get the prime implicants for each subproblem. Selecting the prime implicants does not need the prime implicant table; a smaller table called minterm table is used. It saves space in memory. Comparing our approach with the classic approaches results in the following: 1) natural order of minterms is required. 2) it introduces far fewer number of comparisons than that of Q-M. 3) it consumes less space in memory 4) far fewer number of repeated clauses are generated.

Keywords: logic design, Minimal sum of product, Prime implicant, Quine-McCluskey method.

I. INTRODUCTION

This paper addresses the problem of the minimization of switching functions. For few number of variables, Karnaugh map [26] can be used. For six or higher number of variables, several minimization procedures have been introduced. These procedures can be classified as follows:

1. with respect to generation of prime implicants:
   a. classic procedures that consist of generating all the prime implicants, then finding the minimum representation of the function [6,8,9,30].
   b. procedures that generate only the necessary prime implicants [10-12].
2. with respect to efficiency
   a. procedures for finding minimum representation [9,10,30].
   b. procedures for finding all/some irredundant normal forms [6,8,12].
3. with respect to form of input
   a. minterms m_i, where $f(m_i)=1$ or x (don't care) [10,11,30].
   b. an algebraic expression representing the function [6-9,12].

Our approach is a modification of the classic approaches for absolute minimization. The classic approach enhances each step separately. i.e. it is required to efficiently find all prime implicants, and then to efficiently select a set of prime implicants for absolute minimization. Finding the prime implicants of a function is an important point in its own. Several papers addressed only this problem [4,7,13-18,26]. [4,7,16-18,26] require the minterms representing the function as input, while [13-15] require an algebraic representation of the function and treat the problem algebraically. [16,17] solve the problem numerically while [4,18] use tabular methods. [7] solves it by recursively applying the so-called Pi operator and [26] by applying the sharp operator. The most popular classic approach is the tabular method by Quine-McCluskey [2]. Their

method leads to the generation of many repeated implicants(clauses) which consumes time and space. This problem was solved by Hwa's approach [4], but its drawback is the large number of comparisons between rows of adjacent groups. Morreale's algorithm [6] also avoids generating repeated clauses. Its drawback is the time elapsed to perform nonexhaustive search to get the pair of implicants that are reducible.

In this paper, the function's minterms are treated in their natural order. Comparisons between rows are applied in smaller domain as the problem is partitioned into subproblems. The final results are obtained by applying the Consensus operation to the results of the subproblems. This approach reduces the generation of repeated clauses to a large extent. The second step is also addressed separately in the literature [19-23,26-28]. [19,20,22,27,28] find the absolute minimization of the function.,while [23] finds the near-min representation of it. [21] generates the set of irredundant normal forms. Most of them [20-23] use tabular method to get the solution while [19] uses integer programming. [27,28] solve it algebraically. Roth'salgorithm [26] uses sharp operator to get essential prime implicant and then finds an irredundant cover of the function. It can also find a min-cost representation. In our approach, a small table called minterm table is used. The tabular representation proposed by [3] is adopted with some modifications. The rest of the paper will be as follows: section II introduces basic definitions. Section III presents the proposed approach and section IV is for the conclusion.

II. BASIC DEFINITIONS

Consensus operation (first proposed by Quine [1,26]):

Let P_1 and P_2 be two product terms of a function such that there exists one and only one variable x_i which is complemented in one product term and uncomplemented in the other. Let Q_1 and Q_2 be the product terms P_1 and P_2 respectively, after eliminating x_i and $\overline{x_i}$ from them, then the consensus of P_1 and P_2=(Q_1 AND Q_2).

Subsumption operation [5]:

It is the application of the following theorems:

$x+x=x$
$x+xy=x$

Tabular Representation of a function [3,5]:
A function of n variables represented in a sum of product terms can be represented in an n-column table of rows. Each row represents a product term and each column corresponds to a variable. A product term is represented as follows: column i = 0,1 or - for complemented, uncomplemented or absent variables.

Consensus and subsumption in tabular form [5]:

1. Let A and B be two rows. Let a_i, b_i denote the elements in column i in A and B respectively. The consensus of A and B ,if found, is row C, where element c_i in column i can be found as follows:

   if $a_i=b_i$ then $c_i=a_i$
   if a_i= - then $c_i=b_i$
   else if b_i= - then $c_i=a_i$
   if $a_i=\overline{b_i}$ then c_i= -

   Example: let A be 00-00 and B be 0001-, then C is 000-0.

2. A row A subsumes another row B if for each column $b_i \neq -, a_i=b_i$. The subsuming row is eliminated.

Iterative consensus:
Let $f(x_1,x_2,...,x_n)$ be a function represented in a sum of product form. i.e. $f=\sum_{i=1}^{s} P_i$ where s is the number of product terms. Iterative consensus method is to iteratively apply both consensus and consumption to each pair of products in f with the addition of any new terms to f. The application of iterative consensus (denoted it-con(f)) results in all prime implicants of f as proposed by McCluskey [25] and Dietmeyer [26]. Let PI(f) denotes the set of all prime implicants of f, then $PI(f)=it-con(\sum_{i=1}^{s} P_i)$.

Sharp product [26]:

The sharp product of two cubes A and B finds the minterms (vertices) of A not included in B. In this

paper, sharp product(denoted by #) is applied, as defined by Trabado et. al [3], with separation. Separation means that if the answer consists of more than one row, overlapping is eliminated. Example: Let A=--10, and B=001- then A#B={1-10,0110}

III. THE PROPOSED APPROACH

The approach consists of two steps; generation of all prime implicants, and selection of prime implicants that form the minimized function. The first step is solved by partitioning the problem into smaller problems. In pass one, the problem is partitioned into $2^{(n-1)}$ problems of 2^1 minterms. Pass one finds the prime implicants for all these problems. In general, the output of pass (i-1) is input to pass i which solves $2^{(n-i)}$ problems of 2^i minterms. Consensus and subsumption operations are applied to get the prime implicants for each subproblem.

Theorem 1

$PI(f)=it\text{-}con(PI(\overline{x}_1 f(0,x_2,..,x_n))+PI(x_1 f(1,x_2,..,x_n)))$

Proof:

By Shannon's expansion theorem [24]:

$f=\overline{x}_1 f(0,x_2,..,x_n)+x_1 f(1,x_2,..,x_n)$. i.e. f can be represented by two subfunctions. Each subfunction can be expressed as a sum of all its prime implicants.

So, $f=PI(\overline{x}_1 f(0,x_2,..,x_n))+PI(x_1 f(1,x_2,..,x_n))$. Applying the iterative consensus results in the prime implicants of the original function [25]. i.e.

$PI(f)=it\text{-}con(PI(\overline{x}_1 f(0,x_2,..,x_n))+PI(x_1 f(1,x_2,..,x_n)))$.

This means that the problem of finding all prime implicants of f is partitioned into two subproblems of finding prime implicants and the application of it-con operator to the two sets of prime implicants. In this case, a product term of one set is not required to be compared with other ones in the same set which saves time. This partitioning can be repeated recursively. i.e. partitioning each of the smaller problems into two smaller ones and so on. The last partition leads to finding the prime implicants of $2^{(n-1)}$ problems each having 2^1 minterms which is trivial.

It has to be noted that this idea is a form of a generalized consensus first proposed by Tison [8].

Theorem 2

Let $f=f_1+f_2$ as defined above by Shannon's expansion theorem, where f represents the original function or any intermediate subfunction. Let $PI(f_1)=\{A_1,...A_r\}$ and $PI(f_2)=\{B_1,...B_q\}$. If consensus of two terms A_i and B_j results in a new term C_k ,then applying consensus of C_k with any elements in $PI(f_1)$, $PI(f_2)$ or any new generated term will not result in any new terms.

Proof:

Let A_i be on the form $y_1y_2..y_r0^{**}..^*$ and let B_i be on the form $y_1y_2..y_r1^{**}..^*$ where variable y_i can be either 0 or 1 and * stands for 0,1, or -. Let C_k be the consensus of two terms A_i and B_j. Then C_k will be on the form $y_1y_2..y_r\text{-}^{**}..^*$. Consider the following three cases:

1. Consensus of any element of f_1 and C_k, if possible, results in a term on the form $y_1y_2..y_r0^{**}..^*$ which ,by definition of A_i, is an implicant or prime implicant of f_1.
2. Similarly the consensus of an element of f_2 and C_k does not generate any new term.
3. If the consensus operation between C_k and any new term C_k', which will be in the form $y_1y_2..y_r\text{-}^{**}..^*$ results in a term, then, by definition of consensus, there must exist one and only one column having a value 1 in one term and 0 in the other. i.e. without loss of generality, C_k and C_k' will be in the form $y_1y_2..y_r\text{-}^*0..^*$ and $y_1y_2..y_r\text{-}^*1..^*$ respectively. The consensus will be in the form $y_1y_2..y_r\text{-}^*\text{-}..^*$. This term is not new as it can result from the consensus of the following terms $y_1y_2..y_r0^*\text{-}..^*$ and $y_1y_2..y_r1^*\text{-}..^*$ that belong to f_1 and f_2 respectively. This imply that consensus of any pair of new terms does not generate a new one.

Algorithm:

First, to get all prime implicants of a function $f(x_1,x_2,...x_n)$, 2^n tables are constructed, where table (i) contains a row representing minterm m_i if m_i is a true/don't care minterm of the function, otherwise it contains NULL. To obtain all prime implicants, n passes are performed. Each pass combines the

tables in pairs such that table(i) is combined with table(i+1), i=0,2,... The combination operation generates all possible prime implicants with respect to the minterms covered by the involved tables. For example, if T(1) contains {00-00} and T(2) contains {0001-} -the dash denotes don't care- then the generated table T(12)={000-0,00-00,00010}. Pass n ends with one table containing all possible prime implicants.

Example: f(x,y,z)={0,1,2,3,4,5,7}

tables	initially	pass 1	pass 2	pass 3
	000	00-	0--	--1
	001	01-	1-1	-0-
	010	10-	10-	0--
	011	111		
	100			
	101			
	NULL			
	111			

i.e. the prime implicants are $z,\bar{y},\bar{x}$.

It has to be noted that NULL tables can be eliminated. In this case, each table is assigned a number corresponding to the partition it represents such that table i and the next table (table j) can be combined in pass k if and only if $j-i=2^{(k-1)}\forall i$ that is multiple of 2^k.

This approach is simpler than that of Q-M which performs the same number of passes but in pass i, the method generates all possible cubes composed of 2^i minterms. i.e. if there exists a cube of 2^k minterms, the method generates r1 rows where $r1=k*\{2^{(k-1)}+2^{(k-2)}+...+2^1\}+1$ while our method generates only r2 rows where $r2 = 2^{(k-1)}+2^{(k-2)}+...+2^1+1$. i.e. $r1=k*r2$ which reduces our tables compared to that of Q-M.

Selecting the prime implicants is performed without the need to construct the prime implicant table. This can be done by recursively selecting the essential prime implicants -using a smaller table- then applying the sharp operation between each prime implicant and each of the essential ones to reduce the number of prime implicants if possible. Applying this sequence recursively leads to the minimum representation of the function.

The steps can be summarized as follows:

Let DC denotes the set of don't care minterms. Let TI denotes the initial table which contains the set of all prime implicants.

1. Construct a small table which has three rows; MINTERM (as index), COVER, and FLAG. Assign a column to each true minterm.
2. For each prime implicants P, mark FLAG(minterm) and let COVER(minterm)=P for minterms that P covers.
   When collision occurs, unmark FLAG(minterm) where collision occurs and let COVER(minterm)=P. If a prime implicant is found to cover no minterm in the table, then it contains only don't care minterms, remove it from the set of prime implicants.
3. For the marked minterms, the COVER entry refer to an essential prime implicant. Add the minterms of them to DC.
4. If no essential prime implicant exists, exit (cyclic case). Otherwise, write all essential prime implicant in a separate table (TE) and eliminate them from the initial table (TI).
5. Apply the sharp operation to eliminate all minterms in DC from TI. Find the cost of each prime Implicant in TI.
6. If a row A of TI covers row B (the set of minterms of B is subset of that of A) and cost(A)<=cost (B), eliminate B. The problem now is reduced to a smaller one with less number of minterms.
7. Go to step 1.

A cyclic case is discovered if TE is NULL. In this case, some condition must be added to enforce a prime implicant to be added to TE. Otherwise Branching or Petrick's algorithm can be used [29].

It has to be noted that the sharp operation may lead to splitting a prime implicant into more than one row constituting the minterms that have not been covered yet. So, an identifier must be associated with each prime implicant (assigned to each row of it). Each group of rows having the same identifier are treated as an entity.

Testing for partitionability:

Through determining essential prime implicants, the problem is tested for partitionability. i.e. the prime implicants may be divided into disjoint sets such that every set covers a set S_i of minterms where $\cup_i S_i = f$ and $\cap_i S_i = \phi$. In this case, the problem is partitioned into several subproblems. The required expression is the sum of expressions resulting from solving these subproblems.

For large problems, the partitioning approach can be applied recursively. The algorithm for partitioning the problem can be summarized as follows:

While generating the minterm table do the following:

1. Get the next prime implicant (P).
2. Set a NULL identifier to it.
3. for each of its minterms: If FLAG(minterm) is unmarked and identifier(P)=NULL, set a new identifier to P. If FLAG(minterm) is marked, then let id-cover=identifier (COVER(minterm)). Let identifier(p) =id-cover$\forall$p where identifier(p) =identifier(P).
   At end, each group of prime implicants having an equal identifier constitute a set. The corresponding set of minterms is the union of all their minterms.

IV. CONCLUSION

The partitioning approach is introduced to obtain a minimum representation of a function(with/without don't care's) using tabular representation. It is a simple and efficient technique over many techniques in the literature. Minterms are used in their natural order. The approach reduces the generation of repeated clauses to a large extent. Also, the usage of consensus operation results in the generation of the smallest number of implicants (edges, faces, cubes,..etc.) that are necessary in generating a prime implicant. The principle of partitioning the second step of the problem ,if possible, may greatly affect the complexity of the problem.

REFERENCES

[1] A.R. Meo, "On the Determination of the ps Maximal Implicants of a Switching Function", *IEEE Trans. on Elect. Comp.* vol. EC-14, no. 6, pp. 830-840, Dec. 1965.

[2] Richard S. Sandige, *Modern Logic Design*, McGraw-Hill Int'l Editions, 1990.

[3] P.P. Trabado and A. Lioris-Ruis, "Solution of Switching Equations Based on a Tabular Algebra", *IEEE Trans. on Comp.*, vol. 42, no. 5, pp. 591-596, May 1993.

[4] H.R. Hwa, "A Method for Generating Prime Implicants of a Boolean Expression", *IEEE Trans. On Comp.*, vol. C-23, no. 6, pp. 637-641, June 1974.

[5] M.P. Marcus, :Switching Circuits for Engineers", 3rd edition, PrenticHall of India, New delhi, 1975.

[6] E. Morreale, "Recursive Operators for Prime Implicant and Irredundant Normal Form Determination", *IEEE Trans. on Comp.* vol. C-19, no. 6, pp. 504-509, June 1970.

[7] M. Ikram and D.A. Roy, "A Simple Technique to Improve the Pi-Algorithm for Prime Implicant Determination", *IEEE Trans. on Comp.*, vol C-25, no 11, pp. 1184-1187, Nov. 1976.

[8] P. Tison, "Generalization of Consensus Theory and Application to the Minimization of Boolean Functions", *IEEE Trans. on Elect. Comp.*, vol. EC-16, no. 4, pp. 446-456, Aug. 1967.

[9] J.R. Slagle, C.-L. Chang and R.C.T. Lee, "A New Algorithm for Generating Prime Implicants", *IEEE Trans. on Comp.*, vol. C-19, no. 4, pp. 304-310, Apr. 1970.

[10] V.T. Rhyne, P.S. Noe, M.H. Mckinney, and U.W. Pooch, "A New Technique for the Fast Minimization of Switching Functions", *IEEE Trans. on Comp.*, vol. C-26, no. 8, pp. 757-764, Aug. 1977.

[11] H.A. Curtis, "Adjacency Table Method of Deriving Minimal Sums", *IEEE Trans. on Comp.*, vol. C-26, no. 11, pp. 1136-1141, Nov. 1977 .

[12] D.M.Y. Chang and T.H. Mott, Jr. "Computing Irredundant Normal Forms

from Abbreviated Presence Functions" *IEEE Trans on Elect. Comp.* vol. EC-14, no. 3, pp. 335-342, June 1965.

[13] J.G. Bredeson and P.T. Hulina, "Generation of Prime Implicants by Direct Multiplication", *IEEE Trans. on Comp.*, vol. C-20, no. 4, pp. 475-476, Apr. 1971.

[14] S.R. Das and N.S. Khabra, "Clause-Column Table Approach for generating All the Prime Implicants of Switching Functions", *IEEE Trans. on Comp.*, vol. C-21, no. 11 ,pp 1239-1246, Nov. 1972.

[15] B.L. Hulme and R.B. Worrell, "A Prime Implicant Algorithm with Factoring", *IEEE Trans. on Comp.*, vol. C-24, no. 11, pp. 1129-1131, Nov. 1975.

[16] N.N. Necula,"A Numerical Procedure for Determination of the Prime Implicants of a Boolean Function", *IEEE Trans. on Elect. Comp.* vol. EC-16, no. 5, pp. 687-689, Oct. 1967.

[17] A. Svoboda, "Ordering of Implicants" *IEEE Trans. on Elect. Comp.* vol. EC-16, no. 1, pp. 100-105, Feb. 1967.

[18] E. Morreale, "Partitioned List Algorithms for Prime Implicant Determination from Cannical Forms", *IEEE Trans. on Elect. Comp.* vol. EC-16, no. 5, pp. 611-620, Oct. 1967.

[19] F. Luccio, "A Method for the selection of Prime Implicants", *IEEE Trans. on Elect. Comp.*, vol. EC-15, no. 2, pp. 205-212, Apr. 1966.

[20] S.U. Robinson III and R.W. House, "Gimpel's Reduction Technique Extended to the Covering Problem with Costs", *IEEE Trans. on Elect. Comp.* vol. EC-16, no. 4, pp. 509-514, Aug. 1967.

[21] V. Bubenik, "Weighting Method for the Determination of the Irredundant Set of Prime Implicants", *IEEE Trans. on Comp.*, vol. C-21, no. 12, pp. 1449-1451, Dec. 1972.

[22] J.F. Gimpel, "A Reduction Technique for Prime Implicant Tables", *IEEE Trans. on Elect. Comp.* vol. EC-14, no. 4, pp. 535-541, Aug. 1965.

[23] S.R. Das, "An approach for Simplifying Switching Functions by Utilizing the Cover Table Representation", *IEEE Trans. on Comp.* vol. C-20, no 3, pp. 355-359, Mar. 1971.

[24] A.D. Friedman, *Logical Design of Digital Systems*, Pitman, 1975.

[25] J.M. Mage, "Application of iterative Consensus to Multiple-Output Functions", *IEEE Trans. on Comp.* vol. C-19, no. 4, pp. 359, Apr. 1970.

[26] D.L. Dietmeyer, "Logical Design of Digital Systems", 2nd ed., Allyn and Bacon, 1978.

[27] S.J. Hong and S. Muroga, "Absolute Minimization of Completely Specified Switching Functions", *IEEE Trans. on Comp.* vol. 40, no. 1, pp. 53-65, Jan. 1991.

[28] R.B. Cutler and S. Muroga, "Derivation of Minimal Sums for Completely Specified Functions", *IEEE Trans. on Comp.* vol. C-36, no. 3, pp. 277-292, Mar. 1987.

[29] F.J. Hill and G.R. Peterson, *Introduction to Switching Theory and Logical Design*, John Wiley & Sons, Inc, 1968.

[30] N.N. Biswas, "Minimization of Boolean functions", *IEEE Trans. on Comp.*, vol. C-20, no. 8, pp. 925-929, Aug. 1971.