1994-05

Multiple-valued Logic Operations with Universal Literals

Dueck, Gerhard W.; Butler, Jon T.

IEEE

Dueck, Gerhard W. "Multiple-valued logic operations with universal literals." IEEE Proc. 24th Int. Symp. on Multiple-Valued Logic. 1994.
https://hdl.handle.net/10945/70586

Multiple-Valued Logic Operations with Universal Literals†

Gerhard W. Dueck
Dept. of Math. and Comp. Sci.
St. Francis Xavier University
Antigonish, N. S. B2G 1C0

Jon T. Butler
Dept. of Elec. and Comp. Eng.
Naval Postgraduate School
Monterey, CA 93943-5121

Abstract

We propose the use of universal literals as a means of reducing the cost of multiple-valued circuits. A universal literal is any function on one variable. The target architecture is a sum-of-products structure, where sum is the truncated sum and product terms consist of the minimum of universal literals. A significant cost reduction is demonstrated over the conventional window literal. The proposed synthesis method starts with a sum-of-products expression. Simplification occurs as pairs of product terms are merged and reshaped. We show under what conditions such operations can be applied.

1 Introduction

The goal of this paper is to develop the theoretical framework needed to provide more efficient implementations of multiple valued logic circuits. Towards this end, we propose the use of more complex literal functions than have been used in the past. Previous literal functions include the so-called window literal. We replace this with the universal literal, which is any logic function on a single variable. We propose a synthesis method that combines and divides product terms. While this makes synthesis more complicated, it results in more efficient realizations. There are two contributions of this paper 1. establishing a theoretical basis for the new operations and 2. demonstrating their efficiency.

Many multiple-valued logic minimization algorithms use the direct cover method [POM81, BES86, DUE87]. In direct cover minimization, a minterm is selected according to some criteria. From all the implicants that cover the selected minterm, the best one is chosen to be part of the solution. This process is iterated until all minterms are covered. Several direct cover algorithms have been implemented in the PLA minimization tool HAMLET [YUR90]. These algorithms use window literals. However, in current-mode CMOS, window literals do not provide as efficient implementation as universal literals [DUE92b]. Also, the cost-table was shown to produce more cost effective implementations. Most minimization procedures using cost-tables in the past have been limited to one or two input variables [LEI91, LEE83, ABD88]. This restriction makes these minimization procedures inept for most practical functions. Dueck [DUE92b] proposed an algorithm which combines cost-tables with direct cover minimization that produces good results for functions with up to four variables. Unfortunately, functions with more than four variables require excessive CPU time.

There are two serious limitations inherent in the direct cover method. One is that it operates on minterms. This implies that the function, even when it is given in a near minimal form, has to be expanded into minterms. Memory requirements become very large when the number of input variables increases. Also, the number of implicants that have to be considered to cover a given minterm may be very large. This occurs when a function can be covered by a few large product terms.

Recently, Dueck *et al.* [DUE92a] proposed an algorithm that manipulates product terms directly without breaking them into minterms. The algorithm makes use of the operations: merge, sharp, and reshape. These operations are applied in a nondeterministic fashion, guided by the simulated annealing principle. Results from this algorithm, which has been incorporated into HAMLET, are encouraging.

We want to be able to manipulate product terms in a sum-of-product expression consisting of universal literals. In this paper, we redefine the primitive operations that have been successfully used with window literals and apply them to universal literals. This provides a basis for algorithms operating on universal literals.

2 Definitions and Notation

Let x_i be a variable that can assume any logic value in the set $R = \{0,1,\cdots,r-1\}$, where r denotes the *radix*. Let $X = \{x_1, x_2, \cdots, x_n\}$ be a set of n variables. An r-valued function is a mapping $f: R^n \to R$. The *universal literal*

† Research supported by the Natural Sciences and Engineering Research Council of Canada and by the Naval Research Laboratory, Washington, DC through direct funds at the Naval Postgraduate School, Monterey, CA.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAY 1994**	2. REPORT TYPE	3. DATES COVERED
4. TITLE AND SUBTITLE **Multiple-valued Logic Operations with Universal Literals**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Department of Electrical and Computer Engineering,Monterey,CA,93943**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
We propose the use of universal literals as a means of reducing the cost of multiple-valued circuits. A universal literal is any function on one variable. The target architecture is a sum-of-products structure, where sum is the truncated sum and product terms consist of the minimum of universal literals. A significant cost reduction is demonstrated over the conventional window literal. The proposed synthesis method starts with a sumof products expression. Simplification occurs as pairs of product terms are merged and reshaped. We show under what conditions such operations can be applied.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**7**	

$< a_0 a_1 \cdots a_{r-1} >_{x_i}$ is a one-variable function $f(x_i)$, such that $f(j) = a_j$. For example, the identity function $f(x_1) = < 0123 >_{x_1}$ has the property that $x_1 = 0$ yields $f = 0$, $x_1 = 1$ yields $f = 1$, $x_1 = 2$ yields $f = 2$, and $x_1 = 3$ yields $f = 3$.

A *product term* of literals $P(x_1, x_2, \cdots, x_n) = < a_{10} a_{11} \cdots a_{1r-1} >_{x_1} < a_{20} a_{21} \cdots a_{2r-1} >_{x_2} \cdots < a_{n0} a_{n1} \cdots a_{nr-1} >_{x_n}$ has the property, that for each assignment α of values to the variables $x_1, x_2, \cdots, x_n$, P is the minimum of the values achieved by the literals for α. For example, $< 0123 >_{x_1} < 0123 >_{x_2}$ is the minimum function; i.e. $P(x_1, x_2) = \mathrm{MIN}(x_1, x_2)$.

A product term representation is not unique. For example, for $P_1(x_1, x_2) = < 3210 >_{x_1} < 1220 >_{x_2}$ and $P_2(x_1, x_2) = < 2210 >_{x_1} < 1230 >_{x_2}$, we can write $P_1 = P_2$. This product term is shown in Figure 1. There, blank squares correspond to assignments where the function is 0. Two product terms are said to be *equivalent* if they realize the same function. Thus, $< 3210 >_{x_1} < 1220 >_{x_2}$ and $< 2210 >_{x_1} < 1230 >_{x_2}$ are equivalent.

$x_2 \backslash x_1$	0	1	2	3
0	1	1	1	
1	2	2	1	
2	2	2	1	
3				

Figure 1. A two-variable 4-valued function.

Let $a_i^{\max} = \mathrm{MAX}(a_{i0}, a_{i1}, \cdots, a_{ir-1})$, where $1 \le i \le n$. A product term is said to be normalized if all $a_i^{\max}$ are equal. For example, for $P_1(x_1, x_2) = < 3210 >_{x_1} < 1220 >_{x_2}$ and $P_2(x_1, x_2) = < 2210 >_{x_1} < 1230 >_{x_2}$, we have $a_1^{\max}, a_2^{\max}$ = 3,2 and 2,3, respectively. Neither is normalized. However, $P_3(x_1, x_2) = < 2210 >_{x_1} < 1220 >_{x_2}$ which is equivalent to $P_1(x_1, x_2)$ and $P_2(x_1, x_2)$, is normalized with $a_1^{\max} = a_2^{\max} = 2$.

Let $a^{\min\max} = \mathrm{MIN}(a_1^{\max}, a_2^{\max}, \cdots, a_n^{\max})$. A product term P can be converted into a normalized one equivalent to P by replacing all $a_{ij} > a^{\min\max}$ with $a^{\min\max}$. Note that the normalized product term is unique. For the remainder of this paper, we will only consider normalized product terms on universal literals.

Let a *minterm* be a function $f(x_1, x_2, \cdots, x_n)$ that is nonzero for exactly one assignment of values to the variables. A minterm can be represented by a product term, where each literal consists of all 0 components except one.

The *size* of a product term is the number of assignments of values to variables for which the product term is nonzero. For example, the size of the product term $< 2210 >_{x_1} < 1220 >_{x_2}$ is 9.

Function f_1 *covers* f_2, if $f_1(\beta) \ge f_2(\beta)$, for all assignments of values β, where $\ge$ is the greater-than-or-equal-to operator with logic values viewed as integers. For example, the function $< 1220 >_{x_1} < 0221 >_{x_2}$ covers $< 0120 >_{x_1} < 0210 >_{x_2}$.

The truncated sum operation, denoted +, is

$$f_1(\beta) + f_2(\beta) = \mathrm{MIN}[r-1, f_1(\beta) + f_2(\beta)],$$

where + on the right is arithmetic addition with logic values are viewed as integers.

We say that two universal literals $< a_{i0} a_{i1} \cdots a_{ir-1} >_{x_i}$ and $< b_{i0} b_{i1} \cdots b_{ir-1} >_{x_i}$ *intersect* iff

$$\sum_{j=0}^{r-1} a_{ij} b_{ij} \ne 0.$$

For example, literals $< 1030 >_{x_1}$ and $< 1010 >_{x_1}$ intersect, while $< 0223 >_{x_1}$ and $< 1000 >_{x_1}$ do not. The *distance* between two product terms is the number of variables for which the corresponding literals do not intersect. For example, the distance between product terms $P_4(x_1, x_2) = < 1030 >_{x_1} < 0223 >_{x_2}$ and $P_5(x_1, x_2) = < 1010 >_{x_1} < 1000 >_{x_2}$ is 1, since there is one variable, x_2, where the literals do not intersect.

3 Product Term Operations

In this section, we describe operations that can be performed on product terms. These operations have been defined elsewhere for window literals [DUE92]. Here, they are extended to universal literals. Most operations are intuitively easy to understand, but the conditions under which they apply and their implementation in computer programs are not trivial.

3.1 The Merge Operation

A fundamental operation in our proposed method is the merging of two product terms on universal literals. Specifically, we ask under which conditions the truncated sum of two product terms, A and B, can be expressed as a single product C. The first result below specifies the form of C.

Lemma 1: If the truncated sum of two product terms, A and B, is expressible as a single product term $A + B = C = < c_{10} c_{11} \cdots c_{1r-1} >_{x_1} < c_{20} c_{21} \cdots c_{2r-1} >_{x_2} \cdots < c_{n0} c_{n1} \cdots c_{nr-1} >_{x_n}$, then

$$c_{ij} \le a_{ij} + b_{ij}, \tag{1}$$

where a_{ij} and b_{ij} are components in the literals of A and B, respectively, for $1 \le i \le n$ and $0 \le j \le r-1$.

Proof: Consider a c_{ij}, and choose an assignment $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ of values to variables, such that $\alpha_i = j$ and $\alpha_{i'} = j'$, where $i' \ne i$ and $c_{i'j'} = c^{\min\max}$. Because the product term is normalized, we can always choose $c_{i'j'}$ as $c^{\min\max}$. It follows that $C(\alpha) = \mathrm{MIN}(c_{1\alpha_1}, c_{2\alpha_2}, \cdots, c_{n\alpha_n}) = c_{ij}$. On the contrary, if

$c_{ij} > a_{ij} + b_{ij}$, then from $A(\alpha) \le a_{k\alpha_k}$ and $B(\alpha) \le b_{k\alpha_k}$, $C(\alpha) = c_{ij} > a_{ij} + b_{ij} \ge A(\alpha) + B(\alpha)$, which contradicts $C = A + B$. Q.E.D.

Example 1. Let $A = <1030>_{x_1} <0223>_{x_2}$ and $B = <1010>_{x_1} <1000>_{x_2}$. See Figure 2. $C = A + B$ is expressible as a single product term $C = <1030>_{x_1} <1223>_{x_2}$, which has the property $c_{k\alpha_k} = a_{k\alpha_k} + b_{k\alpha_k}$, except for $k = 1$ and $\alpha_k = 0$, where $c_{10} < a_{10} + b_{10}$.

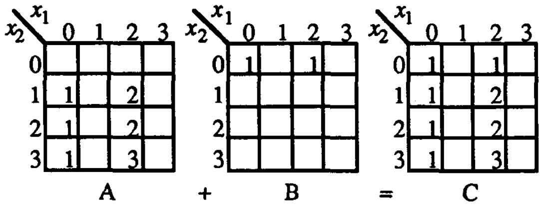

Figure 2. Merging of two product terms.

Example 1 shows that the inequality of (1) cannot be replaced by equality. For many examples, equality holds in (1). For such cases, we can show necessary and sufficient conditions for the merging of two product terms.

Lemma 2: Let A and B be two product terms. $A + B$ is expressible as a single product term C, where

$$c_{j\alpha_j} = a_{j\alpha_j} + b_{j\alpha_j}, \qquad (2)$$

for all $1 \le j \le n$ and $0 \le \alpha_j \le r-1$ iff for all assignments $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ of values to variables $x_1, x_2, \cdots$, and x_n either

a) There exists an i such that both $A(\alpha) = a_{i\alpha_i}$ and $B(\alpha) = b_{i\alpha_i}$

or b) $A(\alpha) + B(\alpha) = r - 1$.

Proof: (if) Assume a) holds. Then, $C(\alpha) = A(\alpha) + B(\alpha) = a_{i\alpha_i} + b_{i\alpha_i} \le a_{j\alpha_j} + b_{j\alpha_j}$ for all $1 \le j \le n$. The latter inequality is the condition for expressibility by a single product term. Assume b) holds. We claim there is no j such that $a_{j\alpha_j} + b_{j\alpha_j} < r - 1$. On the contrary, if so, then $A(\alpha) + B(\alpha) < r - 1$. But $a_{j\alpha_j} + b_{j\alpha_j} \ge r - 1$ for all $1 \le j \le n$ implies $C(\alpha) = r - 1$ and is the condition for expressibility by a single product term.

(only if) Assume that neither a) nor b) hold for some assignment $\alpha' = (\alpha'_1, \alpha'_2, \cdots, \alpha'_n)$. We show that $A + B$ is not expressible as a single product term such that (2) holds. Since A and B are product terms, $A(\alpha') = a_{p\alpha'_p} \le a_{j\alpha'_j}$ and $B(\alpha') = b_{s\alpha'_s} \le b_{j\alpha'_j}$, for all $1 \le j \le n$. Because a) does not hold, $a_{p\alpha'_p} < a_{s\alpha'_s}$ or $b_{s\alpha'_s} < b_{p\alpha'_p}$. Because b) does not hold, $a_{p\alpha'_p} + b_{s\alpha'_s} < r - 1$. Thus, for all $1 \le i \le n$, $a_{i\alpha'_i} + b_{i\alpha'_i} > a_{p\alpha'_p} + b_{s\alpha'_s}$. Further, $r - 1 > C(\alpha') = a_{p\alpha'_p} + b_{s\alpha'_s} < a_{i\alpha'_i} + b_{i\alpha'_i} \le c_{i\alpha'_i}$, which contradicts the condition that C is a single product term satisfying (2). Q.E.D.

Example 2. From Example 1, if $A = <1030>_{x_1} <0223>_{x_2}$ and $B = <1010>_{x_1} <1000>_{x_2}$, then $C = A + B$ is expressible as a single product term $C = <1030>_{x_1} <1223>_{x_2}$. Recall that $c_{k\alpha_k} \le a_{k\alpha_k} + b_{k\alpha_k}$, except for one case, $1 = c_{10} < c_{10} + c_{10} = 1 + 1$. Since this does satisfy (2) in Lemma 2, then it must not satisfy a) or b) of Lemma 2. This can be seen as follows. Consider $\alpha = (0,2)$. Since $A(\alpha) + B(\alpha) = 1$, b) is not satisfies. But neither is a), as follows. For $i = 1$, $A(\alpha) \ne a_{10}$, and for $i = 2$, $B(\alpha) \ne a_{22}$.

Definition 1: Two product terms, A and B, can be *merged* iff there is a product term, C, such that $C = A + B$, where + is the truncated sum.

For example, the product terms A and B in Example 2 can be merged into C. Note that Lemmas 1 and 2 give conditions under which two product terms can be merged. A third condition follows.

Lemma 3: If product terms A and B can be merged, then the distance between them is no greater than 1.

Proof: On the contrary, assume A and B are distance two or more apart. Thus, at least two of the universal literals in A do not intersect with their corresponding literals in B. Let i and j denote the indices of these literals. That is, for $A = \cdots < a_{i0} a_{i1} \cdots a_{ir-1} >_{x_i} \cdots < a_{j0} a_{j1} \cdots a_{jr-1} >_{x_j} \cdots$ and $B = \cdots < b_{i0} b_{i1} \cdots b_{ir-1} >_{x_i} \cdots < b_{j0} b_{j1} \cdots b_{jr-1} >_{x_j} \cdots$, we have $a_{ik} b_{ik} = 0$ and $a_{jk} b_{jk} = 0$ for $0 \le k \le r-1$. Let $\alpha = \alpha_1 \alpha_2 \cdots \alpha_n$ and $\beta = \beta_1 \beta_2 \cdots \beta_n$ be nonzero minterms in A and B, respectively. It follows that $a_{1\alpha_1} > 0$, $a_{2\alpha_2} > 0, \cdots a_{n\alpha_n} > 0$ and $b_{1\beta_1} > 0$, $b_{2\beta_2} > 0, \cdots b_{n\beta_n} > 0$. Assume A and B can be merged into one product term C. Thus, for $C = \cdots < c_{i0} c_{i1} \cdots c_{ir-1} >_{x_i} \cdots < c_{j0} c_{j1} \cdots c_{jr-1} >_{x_j} \cdots$, we have $C = A + B$, and it follows that $c_{1\alpha_1} > 0$, $c_{2\alpha_2} > 0, \cdots c_{n\alpha_n} > 0$ and $c_{1\beta_1} > 0$, $c_{2\beta_2} > 0$, $\cdots c_{n\beta_n} > 0$. Consider the assignment $\gamma = \alpha_1 \alpha_2 \cdots \alpha_{i-1} \beta_i \alpha_{i+1} \cdots \alpha_n$. It follows that $C(\gamma) > 0$, since $c_{1\alpha_1} > 0, \cdots, c_{i-1\alpha_{i-1}} > 0, c_{i\beta_i} > 0, c_{i+1\alpha_{i+1}} > 0, \cdots, c_{n\alpha_n} > 0$. By the nonoverlap of A and B, we have $a_{i\alpha_i} b_{i\alpha_i} = 0$ and $a_{j\beta_j} b_{j\beta_j} = 0$. Since $a_{i\alpha_i} > 0$ and $b_{j\beta_j} > 0$, it follows that $b_{i\alpha_i} = 0$ and $a_{j\beta_j} = 0$. But, the former implies that $B(\gamma) = 0$, while the latter implies $A(\gamma) = 0$. Thus, $C(\gamma) = A(\gamma) + B(\gamma) = 0$, a contradiction. Q.E.D.

We now establish the conditions under which two product terms at distance 1 or less may be merged. We consider three different cases.

Case 1: Two product terms at distance 1 can be merged if the conditions in Lemma 4 are satisfied.

Lemma 4: Let A and B be two product terms at distance 1, and let x_i be the variable for which the corresponding literals do not intersect. A and B can be merged iff for all $1 \le j \le n$, such that $i \ne j$ and $0 \le k \le r - 1$,

a) if $a_{jk} > b_{jk}$, then $b_{jk} = b^{\min\max}$,

b) if $b_{jk} > a_{jk}$, then $a_{jk} = a^{\min\max}$.

Example 3. Figure 3a below illustrates an A and B that satisfy the conditions in Lemma 4. In Figure 3a, $A = <0130>_{x_1} <1300>_{x_2}$ and $B = <0120>_{x_1} <0022>_{x_2}$. Their truncated sum $C = <0130>_{x_1} <1322>_{x_2}$. However, the functions $D = <0310>_{x_1} <1300>_{x_2}$ and $E = <0120>_{x_1} <0022>_{x_2}$ shown in Figure 3b, cannot be merged, because they don't satisfy the conditions in Lemma 4.

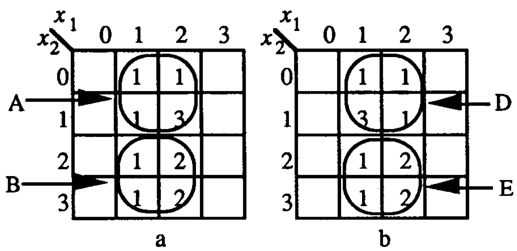

Figure 3. Illustration for Lemma 4.

Proof: (if) Assume that the conditions hold. Let $c_{jk} = MAX(a_{jk}, b_{jk})$. We show that for all assignments α of values to variables that $C(\alpha) = A(\alpha) + B(\alpha)$. There are two cases; either $C(\alpha) = 0$ or $C(\alpha) \neq 0$. If $C(\alpha) = 0$, then for at least one j and one k, $c_{jk} = 0$, and thus $a_{jk} = a_{jk} = 0$ (since $c_{jk} = MAX(a_{jk}, b_{jk})$). Thus, $C(\alpha) = A(\alpha) + B(\alpha)$. Now consider, $C(\alpha) \neq 0$. Since A and B are at distance 1 either $A(\alpha) = 0$ or $B(\alpha) = 0$. Without loss of generality, assume that $B(\alpha) = 0$. We must show that $C(\alpha) = A(\alpha)$. We have $C(\alpha) = MIN\{c_{j\alpha_j}\} = MIN\{MAX(a_{j\alpha_j}, b_{j\alpha_j})\}$ and $b_{i\alpha_i} = 0$ since $A(\alpha) \neq 0$. If $a_{j\alpha_j} \geq b_{j\alpha_j}$, then $MAX(a_{j\alpha_j}, b_{j\alpha_j}) = a_{j\alpha_j}$ this hold for $i = j$. If $a_{j\alpha_j} < b_{j\alpha_j}$ then condition b) applies, i.e. $a_{j\alpha_j} = a^{\min\max}$, however $a_{i\alpha_i} \leq a^{\min\max} < b_{j\alpha_j}$ (in other words, it will never be the minimum) and this implies that $MIN\{MAX(a_{j\alpha_j}, b_{j\alpha_j})\} = MIN\{a_{j\alpha_j}\}$.

(only if) Assume that A and B can be merged. That is, there exists a product term C such that $C=A+B$. Assume, on the contrary, that the conditions are not satisfied. Specifically, assume a) is not satisfied; the argument for b) is similar. That is, suppose there is a j and a k such that $a_{jk} > b_{jk}$, but that $b_{jk} \neq b^{\min\max}$. Since the product terms are normalized, the latter inequality implies that $b_{jk} < b^{\min\max}$. Consider an assignment $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_n)$ of values to the variables, such that $\alpha_j = k$ and, for $l \neq j$, $\alpha_l = h$, where $b_{lh} = b^{\min\max}$. Because the product term is normalized, all $b_i^{\max}$ are the same, and $B(\alpha) = b_{jk}$. But, A and B do not intersect, and $A(\alpha) = 0$. Thus, if A and B can be merged into a single product term C, then $C(\alpha) = A(\alpha) + B(\alpha) = b_{jk}$. It follows that $MIN(c_{1\alpha_1}, c_{2\alpha_2}, \cdots, c_{n\alpha_n}) = b_{jk}$. Consider the assignment $\alpha' = (\alpha_1', \alpha_2', \cdots, \alpha_n')$, such that $\alpha_j' = \alpha_j$, except for $j=i$, in which case $\alpha_j' = b^{\min\max}$. It follows that $C(\alpha') = A(\alpha') + B(\alpha') = b^{\min\max}$ and that $c_{jk} = b_{jk}$.

Consider an assignment $\beta = (\beta_1, \beta_2, \cdots, \beta_n)$ of values to the variables, such that $\beta_j = k$ and, for $l \neq j$, $\beta_l = h$, where $a_{lh} = a^{\min\max}$. Because the product term is normalized, all $a_i^{\max}$ are the same, and $A(\beta) = a_{jk}$. But, A and B do not intersect, and $B(\beta) = 0$. Thus, if A and B can be merged into a single product term C, then $C(\beta) = A(\beta) + B(\beta) = a_{jk}$. It follows that $MIN(c_{1\alpha_1}, c_{2\alpha_2}, \cdots, c_{n\alpha_n}) = a_{jk}$. Consider the assignment $\beta' = (\beta_1', \beta_2', \cdots, \beta_n')$, such that $\beta_j' = \beta_j$, except for $j=i$, in which case $\beta_j' = a^{\min\max}$. It follows that $C(\beta') = A(\beta') + B(\beta') = a^{\min\max}$ and $c_{jk} = a_{jk}$. But $a_{jk} > b_{jk}$, and there are contradictory requirements on c_{jk}. It follows that A and B cannot be merged. Q.E.D.

Case 2: Consider two product terms A and B at distance 0, such that B covers A. The merged product term C is obtained as follows:

$C \leftarrow B$
for each minterm α included in A
modify C such that $C(\alpha) = A(\alpha) + B(\alpha)$

Once C has been obtained we have to verify that it is indeed equal to $A + B$.

for each minterm α included in C
we must have $C(\alpha) = A(\alpha) + B(\alpha)$

otherwise A and B cannot be merged.

Example 4. Given $A = <0321>_{x_1} <0323>_{x_2}$ and $B = <0010>_{x_1} <0100>_{x_2}$ we obtain $C = <0331>_{x_1} <0323>_{x_2}$. But $C \neq A + B$ since $C(2,3) \neq A(2,3) + B(2,3)$. The left hand side of the inequality evaluates to 3, whereas the right hand side is equal to 2.

Finally, we consider product terms at distance 0 that do not fall into Case 2.

Case 3: If A can be expressed as $A = A_1 + A_2$, such that the pair (A_1, B) falls into Case 1 and (A_2, B) falls into Case 2, then A and B can be merged if the following 2 conditions hold:

1) A_2 and B can be merged (let $C_1 = A_2 + B$)
2) A_1 and C_1 can be merged (let $C = C_1 + A_1$)

If the decomposition $A = A_1 + A_2$ exists, then it is unique. If such decomposition does not exist, then A and B cannot be merged. The following condition must hold for A to be expressed as $A_1 + A_2$, as described above

$[(a_{ik} = 0) \text{ and } (b_{ik} = 0)]$ or $[(a_{ik} \neq 0) \text{ and } (b_{ik} \neq 0)]$,

for all $0 \leq k < r$, $1 \leq i < j$, and $j < i \leq n$ given $1 \leq j \leq n$. Essentially, we split the product term along the j^{th} variable.

Example 5. Consider the product terms $A = <2100>_{x_1} <1200>_{x_2} <0112>_{x_3}$ and $B = <3200>_{x_1} <0133>_{x_2}$

$<0223>_{x_3}$. We can express A as a sum of two product terms, split along x_2, $A = A_1 + A_2 = <1100>_{x_1} <1000>_{x_2} <0111>_{x_3} + <2100>_{x_1} <0200>_{x_2} <0112>_{x_3}$. B can be merged with A_2 (Case 2) as

$$A_2 + B = <3200>_{x_1} <0333>_{x_2} <0223>_{x_3} = C_1.$$

Finally, C_1 can be merged with A_1 (Case 1) as

$$C_1 + A_1 = <3200>_{x_1} <1333>_{x_2} <0223>_{x_3} = A + B.$$

We conclude that A and B can be merged.

3.2 The Consensus Operation

Informally, the *consensus* term C of two product terms A and B is a largest product term that includes minterms from both, such that $A + B$ covers C. The distance between A and B must be either 1 or 0. We consider two cases.

Case 1: First, we consider product terms A and B at distance 1. Unfortunately, two product terms may have more than one consensus term—according to the definition given above. We illustrate this with the following example.

Example 6. Consider the product terms $A = <3300>_{x_1} <3210>_{x_2}$ and $B = <0023>_{x_1} <0123>_{x_2}$ (see Figure 4.) We have the following seven normalized consensus terms of size 8: $<1111>_{x_1} <0110>_{x_2}$, $<1112>_{x_1} <0120>_{x_2}$, $<1121>_{x_1} <0120>_{x_2}$, $<1122>_{x_1} <0120>_{x_2}$, $<1211>_{x_1} <0210>_{x_2}$, $<2111>_{x_1} <0210>_{x_2}$, and $<2211>_{x_1} <0210>_{x_2}$.

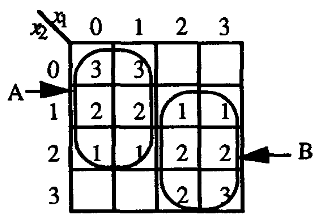

Figure 4. Function used in Example 6.

We give the following definition, which uniquely defines the consensus term of two product terms at distance 1. Let x_i be the variable for which the corresponding literals do not intersect. We define A' as follows;

$$a'_{ik} = a_{ik}, \text{ for } 0 \le k \le r-1$$

and

$$a'_{jk} = \begin{cases} a_{jk} & \text{if } b_{jk} \ne 0 \\ 0 & \text{otherwise} \end{cases},$$

for $0 \le k \le r-1$, $1 \le j < i$, and $i < j \le n$. Similarly, we define B'. Normalize A' and B'. If A' and B' are mergable, then the consensus $C = A' + B'$. If A' and B' are not mergable, then let B'' be the following product term

$$b''_{ik} = \mathrm{MIN}(a'_{ik}, b'_{ik}), \text{ for } 0 \le k \le r-1$$

and

$$b''_{jk} = b'_{jk}, \; 0 \le k \le r-1, \; 1 \le j < i, \text{ and } i < j \le n.$$

The consensus term $C = A' + B''$.

We illustrate our definition of consensus with the following two examples.

Example 7. Consider the product terms $A = <3300>_{x_1} <3110>_{x_2}$ and $B = <0023>_{x_1} <0123>_{x_2}$. We have $A' = <1100>_{x_1} <0110>_{x_2}$ and $B' = <0022>_{x_1} <0120>_{x_2}$. Since they can be merged, the consensus term is $C = <1122>_{x_1} <0120>_{x_2}$.

Example 8. Consider the product terms $A = <3300>_{x_1} <3210>_{x_2}$ and $B = <0023>_{x_1} <0123>_{x_2}$ shown in Figure 4. We have $A' = <2200>_{x_1} <0210>_{x_2}$ and $B' = <0022>_{x_1} <0120>_{x_2}$. Since they cannot be merged, we find $B'' = <0011>_{x_1} <0110>_{x_2}$. The consensus term is $C = <2211>_{x_1} <0210>_{x_2}$.

Case 2: We now consider product terms A and B at distance 0. The consensus term C of A and B includes all minterms that are included in A and in B. We define A' as follows;

$$a'_{jk} = \begin{cases} a_{jk} & \text{if } b_{jk} \ne 0 \\ 0 & \text{otherwise} \end{cases},$$

for $0 \le k \le r-1$, $1 \le j \le n$. Similarly, we define B'. Normalize A' and B'. If A' and B' are mergable, then the consensus $C = A' + B'$. Otherwise, let B'' be the following product term;

$$b''_{jk} = \mathrm{MIN}(b'_{jk}, q), \text{ for } 0 \le k \le r-1 \text{ and } 1 \le j \le n,$$

where q is the minimum non-zero value of all b'_{jk}. The consensus $C = A' + B''$.

Example 9. Consider the product terms $A = <0210>_{x_1} <1210>_{x_2}$ and $B = <0133>_{x_1} <0130>_{x_2}$. $A' = <0210>_{x_1} <0210>_{x_2}$ and $B' = <0130>_{x_1} <0130>_{x_2}$. Since they cannot be merged we find $B'' = <0110>_{x_1} <0110>_{x_2}$. The consensus term is $C = <0320>_{x_1} <0320>_{x_2}$.

Example 10. Consider the product terms $A = <0210>_{x_1} <1210>_{x_2}$ and $B = <0313>_{x_1} <0130>_{x_2}$. $A' = <0210>_{x_1} <0210>_{x_2}$ and $B' = <0310>_{x_1} <0130>_{x_2}$. A' and B' can be merged and the consensus term is $C = <0320>_{x_1} <0330>_{x_2}$.

3.3 The Sharp Operation

The sharp operation (denoted by #) has been used in binary minimization algorithms [HON74]. In binary logic, the sharp operation is defined as $A \# B = A\overline{B}$, where A and B are product terms. Note that $A\overline{B}$ may not be realizable as a single product term. The sharp operation satisfies $A = A \# B + AB$. In an analogous way, we define the sharp operation of two product terms A and B at distance 0 such that $A + B = A \# B + C + B \# A$, where C is the consensus of A and B. If the distance between A and B is greater than zero, then $A\#B = A$. $A\#B$ is a sum of products expression that may not be unique. We illustrate the sharp operation with the following two examples.

Example 11. Given the product terms $A = <0330>_{x_1} <3210>_{x_2}$ and $B = <0222>_{x_1} <0020>_{x_2}$. $A\#B = <0330>_{x_1} <3200>_{x_2}$. In this case, there is a unique solution.

Example 12. Given the product terms $A = <0330>_{x_1} <3213>_{x_2}$ and $B = <0022>_{x_1} <0022>_{x_2}$. Two possible solutions of $A\#B$ are:

1) $<0330>_{x_1} <3200>_{x_2} + <0300>_{x_1} <0013>_{x_2}$
2) $<0300>_{x_1} <3213>_{x_2} + <0030>_{x_1} <3200>_{x_2}$

The consensus of A and B is $C = <0030>_{x_1} <0033>_{x_2}$ and $B\#A = <0002>_{x_1} <0022>_{x_2}$. We verify that the following holds: $A + B = A\#B + C + B\#A$.

3.4 The Reshape Operation

The reshape operations of two product terms A and B is defined as $A\#C + C + B\#C$, where C is the consensus of A and B. Note that when the distance between A and B is greater than 1 the reshape of A and B is $A + B$.

4 Comparison with Window Literals

There is a significant increase in the complexity of the analysis of sum-of-product expressions when window literals are replaced by universal literals. However, there also is a significant decrease in the number of product terms. In a PLA, the space allotted to literals is large enough to accommodate the largest, and we can view the cost of a literal as a constant. This statement is true for both window and universal literals. We expect a universal literal PLA to have a higher cost than a window PLA, because of greater complexity.

In random logic, we can optimize space by accommodating different literal costs. According to Lei and Vranesic [LEI91] the cost of a universal literal (implemented in current mode CMOS) ranges from 1 to 25 and the MIN gate has a cost of 5. The following example illustrates how a function can be expressed with two product terms but have different costs.

Example 13. The function shown in Figure 5 can be expressed as a sum of two product terms with universal literals. However, there is no unique representation. Below are 3 expressions with their corresponding costs.

	expression	cost
1)	$<0300>_{x_1} <3222>_{x_2} + <0023>_{x_1} <0013>_{x_2}$	40
2)	$<0300>_{x_1} <3211>_{x_2} + <0123>_{x_1} <0013>_{x_2}$	38
3)	$<0300>_{x_1} <3210>_{x_2} + <0223>_{x_1} <0013>_{x_2}$	46

This function requires at least five product terms when window literals are used. Using window literals the function can be expressed as $<0300>_{x_1} <3000>_{x_2} + <0200>_{x_1} <0222>_{x_2} + <0011>_{x_1} <0011>_{x_2} + <0011>_{x_1} <0001>_{x_2} + <0001>_{x_1} <0001>_{x_2}$. According to the cost estimates of Lei and Vranesic [LEI91] the implementation of this expression would cost 101.

$x_2 \backslash x_1$	0	1	2	3
0		3		
1		2		
2		2	1	1
3		2	2	3

Figure 5. A 2 variable function.

Any minimization procedure must take into account that normalized product terms may not be cost effective. For example the product term $<0111>_{x_1} <0100>_{x_2}$ has a cost of 20. The equivalent unnormalized product term $<0123>_{x_1} <0100>_{x_2}$ has a cost of 16. Therefore, a good minimization procedure cannot be restricted to consider only normalized product terms.

In this paper, we have shown two operations, merge and reshape. Merge produces one product term from two, while reshape produces two or more product terms from two. Analogous operations have been successfully used in a minimization algorithm with window literals [DUE92a].

References

[BES86] P. W. Besslich, "Heuristic minimization of MVL functions: a direct cover approach," *IEEE Transaction on Computers*, vol. C-35, Feb. 1986, pp. 134-144.

[DUE87] G. W. Dueck and D. M. Miller, "A direct cover MVL minimization using the truncated sum," *Proceedings of the 18th International Symposium on Multiple-Valued Logic*, May 1987, pp. 221-226.

[DUE92a] G. W. Dueck, R. C. Earle, P. Tirumalai, and J. T. Butler, "Multiple-valued programmable logic minimization by simulated annealing," *Proceedings of the 22nd International Symposium on Multiple-Valued Logic*, May 1992, pp. 66-74.

[DUE92b] G. W. Dueck, "Direct cover MVL minimization with cost-tables," *Proceedings of the 22nd International Symposium on Multiple-Valued Logic*, May 1992, pp. 58-65.

[HON74] S. J. Hong, R. G. Cain, and D. L. Ostapko, "MINI: A heuristic approach for logic minimization," IBM Journal of Research and Development, September 1974, pp. 443 - 458.

[KER82] H. G. Kerkhoff and H. A. J. Robroek, "The logic design of multiple-valued logic functions using charge-coupled devices," *Proceedings of the 12th Symposium on Multiple-Valued Logic*, May 1982, pp. 34-44.

[LEE83] J. K. Lee and J. T. Butler, "Tabular methods for the design of CCD multiple-valued circuits," *Proceedings of the 13th Symposium on Multiple-Valued Logic*, May 1983, pp. 162-170.

[LEI91] K. Lei and Z. G. Vranesic, "On the synthesis of 4-valued current mode CMOS circuits," *Proceedings of the 21st International Symposium on Multiple-Valued Logic*, May 1991, pp. 147-155.

[LEI92] K. Lei and Z. G. Vranesic, "Towards the realization of 4-valued CMOS circuits," *Proceedings of the 22nd International Symposium on Multiple-Valued Logic*, May 1992, pp. 104-110.

[POM81] G. Pomper and J. R. Armstrong, "Representation of multivalued functions using the direct cover method," *IEEE Transaction on Computers*, vol. C-30, Sep. 1981, pp. 674-679

[YUR90] J. M. Yurchak and J. T. Butler, "HAMLET—an expression compiler/optimizer for the implementation of heuristics to minimize multiple-valued programmable logic arrays," *Proceedings of the 21st International Symposium on Multiple-Valued Logic*, May 1991, pp. 147-155.