Copyright © 1988, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

ENCODING SYMBOLIC INPUTS FOR MULTI-LEVEL LOGIC IMPLEMENTATION

by

Sharad Malik, Robert K. Brayton, and
Alberto Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M88/69

11 November 1988

ENCODING SYMBOLIC INPUTS FOR MULTI-LEVEL LOGIC IMPLEMENTATION

by

Sharad Malik, Robert K. Brayton, and
Alberto Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M88/69

11 November 1988

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

TITLE PAGE

ENCODING SYMBOLIC INPUTS FOR MULTI-LEVEL LOGIC IMPLEMENTATION

by

Sharad Malik, Robert K. Brayton, and
Alberto Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M88/69

11 November 1988

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

Encoding Symbolic Inputs for Multi-Level Logic Implementation

Sharad Malik Robert K. Brayton Alberto Sangiovanni-Vincentelli

Dept. of EECS
University of California, Berkeley

Abstract

This paper presents an approach for symbolic minimization of combinational logic to be implemented in multi-level form. We consider the problem of finding optimal binary encodings for the different values of a symbolic input variable. An optimal encoding is one that leads to a minimal multi-level implementation of the resulting Boolean logic. Our approach is based on finding encodings that result in large common divisors among a set of expressions. We develop a theoretical foundation that permits us to view all possible common algebraic divisors resulting from all possible encodings of the symbolic variable. We determine necessary and sufficient conditions that the encoding must satisfy for a given common divisor to exist in an encoded implementation. Determining the encoding that results in large common divisors is shown to be equivalent to solving a face embedding problem, for which efficient heuristics exist. We increase the power of our approach by extending it to detect non-algebraic (Boolean) divisors. An interesting aspect is that an initial multi-level decomposition is produced as a by-product of the encoding process.

1 Introduction

Symbolic minimization of combinational logic involves optimizing a logic description with symbolic inputs and outputs (e.g. [8]). A symbolic variable v, may take a value from any general set $V = \{v_0, v_1, \ldots, v_{n-1}\}$ as opposed to a Boolean or binary valued variable which is restricted to the set $\{0, 1\}$. We consider the case where the logic description has symbolic inputs. As an example, consider the combinational logic description of the controller of a microprocessor which may have a symbolic input *opcode* that takes values from the set $\{ADD, SUB, JMP, MOV\}$. The symbolic variable may be represented by a multiple-valued (MV) variable (e.g. [10]). The encoding problem is to find a binary encoding for each of the values of the MV variable, that results in a minimal implementation of the resulting Boolean logic. This problem has been studied extensively for implementations in two-level form ([8]). Recently, state assignment techniques (e.g. [6]) using input and output encoding have been developed for multi-level implementations. However, only common divisors consisting of a single cube have been considered so far. We present an approach that finds encodings that result in common divisors with one or several cubes and is therefore potentially more powerful.

Even for the case of Boolean inputs the problem of multi-level logic minimization is NP-hard and all optimization techniques for any reasonable sized circuits are heuristic. Several approaches towards the problem have been used (e.g. [5,1,7]). An important step in the optimization process in most of these techniques is the detection of common sub-expressions. These are then factored out and implemented as intermediate nodes in the network. This leads to significant savings in the logic, since several occurrences of the sub-expression are replaced by a single intermediate variable. The most common technique for extracting common sub-expressions uses kernel intersections ([4]). We restrict ourselves to this stage of the multi-level optimization process and find encodings that lead to large kernel intersections. We develop a theoretical foundation that permits us to view all possible kernel intersections resulting from all possible encodings of the symbolic variables. We determine necessary and sufficient conditions that the encoding must satisfy for a kernel intersection to exist in an encoded implementation. Determining an encoding that results in large kernel intersections is shown to be equivalent to solving a face embedding problem ([9]), for which efficient heuristics exist ([9,11]). We increase the power of our approach by extending it to detect non-algebraic (Boolean) divisors.

The organization of this paper is as follows. Sections 2 and 3 cover the background material on functions of multiple-valued variables and on using kernels for extracting common sub-expressions. Section 4 extends the concept of kernels to expressions of multiple-valued variables. The relationship between kernel intersections in an encoded implementation and kernels of expressions with multiple-valued variables is explored here. The condition under which a kernel intersection can be obtained in an encoded representation is expressed as a set of constraints on the encoding. Section 5 describes the necessary and sufficient conditions for these constraints to be satisfiable and proves that solving the constraints is equivalent to solving a face embedding problem ([9]). Section 6 extends these ideas to selecting encodings that result in non-algebraic factors. Finally, in Section 7 we summarize a global procedure for finding encodings that result in many large common divisors.

2 Function Representation

Since the case of several symbolic variables can be mapped into the case of a single symbolic variable[1], we restrict ourselves to a single symbolic variable throughout the paper. The rest of the variables are binary valued.

The notation presented in this section is the same as that used in two-level multiple-valued minimization [10]. Let the symbolic variable v take values from the set $V = \{v_0, v_1, \ldots, v_{n-1}\}$. v may be represented by a multiple-valued variable, X, restricted to the set $P = \{0, 1, \ldots, n-1\}$, where each symbolic value of v maps onto a unique integer in P. Let $B = \{0, 1\}$. A binary valued function f of a single MV variable X and $m-1$ binary-valued variables is a mapping:

$$\mathbf{f} : P \times B^{m-1} \rightarrow B$$

Each element in the domain of the function is called a **minterm** of the function. Let $S \subseteq P$. X^S represents the Boolean function:

$$X^S = \begin{cases} 1 & \text{if } X \in S \\ 0 & \text{otherwise} \end{cases}$$

X^S is called a **literal** of variable X. If $|S| = 1$ then this literal is a minterm of X. For example, $X^{\{0\}}$ and $X^{\{0,1\}}$ are literals and $X^{\{0\}}$ a minterm of X. If $S = \phi$, then the value of the literal is always 0. If $S = P$ then the value of the literal is always 1. For these two cases, the value of the literal may be used to denote the literal. If $X^{S_1} \Rightarrow X^{S_2}$ then $X^{S_2} \subseteq X^{S_1}$. We note the following:

1. $X^{S_1} \subseteq X^{S_2}$ if and only if $S_1 \subseteq S_2$
2. $X^{S_1} \cup X^{S_2} = X^{S_1 \cup S_2}$
3. $X^{S_1} \cap X^{S_2} = X^{S_1 \cap S_2}$

The literal of a binary-valued variable y is defined as either the variable or its Boolean complement. A **product term** is a Boolean product (AND) of literals. If a product term evaluates to 1 for a given minterm, it is said to contain the minterm. A **cube** is a product term in which all the literals are binary valued. Note this distinction between a product term and a cube; the latter does not involve any MV variables. A **sum-of-products** (SOP) is a Boolean sum (OR) of product terms. For example: $X^{\{0,1\}}y_1y_2$ is a product term, y_1y_2 a cube and $X^{\{0,1\}}y_1y_2 + X^{\{3\}}y_2y_3$ is a SOP. A function f, may be represented by an SOP expression, f.

3 Kernels and Kernel Intersections

We first review the process of common sub-expression extraction when there are no MV variables. Common sub-expressions consisting of multiple cubes can be extracted from Boolean expressions using the algebraic techniques described in [4]. We review some definitions presented there.

Definition 3.1 *A* **kernel** *of an expression g is defined by the following rules:*

1. *A kernel k of an expression g is the quotient of g and a cube c; $k = g/c$.*
2. *A kernel k is* cube-free, *i.e. no cube is an algebraic factor of k.*

A **co-kernel** *associated with a kernel is the cube divisor used in obtaining that kernel.*

[1] For example, if there are two symbolic variables $v1$ and $v2$ taking values from sets $V1$ and $V2$ respectively, then these may be replaced by a single symbolic variable v taking values from the set $V1 \times V2$. This is in fact better than considering $v1$ and $v2$ separately since the encoding for v takes into account the interactions between $v1$ and $v2$.

As an example consider the expression $g = ae + be + z$ and the cube e. The quotient of g and this cube; g/e is $a + b$. No other cube is a factor of $a + b$, hence it is a kernel of g. The cube e is the co-kernel corresponding to this kernel.

A key result concerning kernels is that two expressions may have common sub-expressions of more that one cube if and only if there is a kernel intersection of more than one cube for these expressions [4]. Thus, we can detect all multiple-cube common sub-expressions by finding all multiple-cube kernel intersections. In [2] algorithms for detecting kernel intersections are described by defining them in terms of the rectangular covering problem. We use the rectangular covering approach for developing our ideas in the rest of the paper. This is because of the ease in understanding the concepts involved. However, the ideas hold using any technique of kernel extraction.

As an example of the rectangular covering formulation, consider the expressions g_1 and g_2:

$$\begin{array}{rcccccc} g_1 & = & ak + & & bk + & & c \\ & & 1 & & 2 & & 3 \end{array}$$

$$\begin{array}{rcccccc} g_2 & = & aj + & & bj + & & d \\ & & 4 & & 5 & & 6 \end{array}$$

The integers below each cube are unique identifiers for that cube. Both g_1 and g_2 have two kernels (shown below).

expression	*co − kernel*	*kernel*
g_1	1	$ak + bk + c$
g_1	k	$a + b$
g_2	1	$aj + bj + d$
g_2	j	$a + b$

The corresponding rectangular covering formulation has the representation:

	a	b	ak	bk	aj	bj	c	d
1	0	0	1	2	0	0	3	0
k	1	2	0	0	0	0	0	0
1	0	0	0	0	4	5	0	6
j	4	5	0	0	0	0	0	0

The above representation is a **co-kernel cube matrix** for the set of expressions. A row in the matrix corresponds to a kernel, whose co-kernel is the label for that row. Each column corresponds to a cube which is the label for that column. A non-zero entry in the matrix specifies the integer identifier of the cube (in the original expressions) represented by the entry. A rectangle $\mathcal{R}$ is defined as a set of rows $S_r = \{r_0, r_1, \ldots, r_{m-1}\}$ and a set of columns $S_c = \{c_0, c_1, \ldots, c_{n-1}\}$ such that for each $r_i \in S_r$ and each $c_j \in S_c$, the (r_i, c_j) entry of the co-kernel cube matrix is non-zero. $\mathcal{R}$ is said to cover each such entry. $\mathcal{R}$ is denoted as: $\{R(r_0, r_1, \ldots, r_{m-1}), C(c_0, c_1, \ldots, c_{n-1})\}$.

A rectangular covering of the matrix is defined as a set of rectangles that cover the non-zero integers in the matrix at least once (and do not cover a 0 entry). An integer covered by one rectangle need not be covered again. Alternatively, once an integer is covered, all other occurrences of it are replaced by don't cares (they may or may not be covered by other rectangles). Each rectangle that has more than one row indicates a kernel intersection. A covering for the above co-kernel cube matrix is: $\{R(2,4), C(1,2)\}$, $\{R(1), C(7)\}$, $\{R(3), C(8)\}$

The kernel intersection $a + b$ between the two expressions is indicated by the first rectangle in the cover. The resulting implementation suggested by the covering is:

$$g_1 = kg_3 + c$$

$$g_2 = jg_3 + d$$

$$g_3 = a + b$$

We will use the total number of literals in the factored form of all the Boolean expressions as the metric for circuit size [5]. The above description has two fewer literals than the original description.

4 Kernels and Multiple-Valued Variables

Now consider the case where one of the input variables may be MV. The following example has a single MV variable X with six values and six binary valued variables.

$$f_1 = \underset{1}{X^{\{0,1\}}ak} + \underset{2}{X^{\{2\}}bk} + \underset{3}{c}$$

$$f_2 = \underset{4}{X^{\{3,4\}}aj} + \underset{5}{X^{\{5\}}bj} + \underset{6}{d}$$

The integers below each product-term are unique identifiers for that product-term.

Our aim is to find a binary encoding for the various values of X that results in large kernel intersections and thus smaller implementations.

Definition 4.1 *An* **encoding** $\mathcal{E}$, *of the values of a MV variable X is a mapping of each distinct value of X onto a distinct vertex in some q dimensional Boolean space, B^q. The* **encoded value** *of a minterm X^α, denoted by $enc(X^\alpha)$, is the singleton set containing the vertex in B^q that X^α is mapped onto. The encoded value of a literal of X is the union of the encoded values of its constituent minterms.*

As an example, consider the following encoding, $\mathcal{E}_1$, for the values of X:

$$X^{\{0\}}:000 \quad X^{\{1\}}:100 \quad X^{\{2\}}:001$$
$$X^{\{3\}}:110 \quad X^{\{4\}}:010 \quad X^{\{5\}}:011$$

We will use the variables s_0, s_1 and s_2 for this 3 dimensional Boolean space. Here $enc(X^{\{0\}}) = \{\bar{s}_0\bar{s}_1\bar{s}_2\}$ and $enc(X^{\{0,1\}}) = \{\bar{s}_0\bar{s}_1\bar{s}_2, s_0\bar{s}_1\bar{s}_2\}$. A set of points in B^q may be alternatively represented as a sum-of-products expression equivalent to the vertices in this set. For example $enc(X^{\{0,1\}})$ may be expressed as $\bar{s}_0\bar{s}_1\bar{s}_2 + s_0\bar{s}_1\bar{s}_2$ or equivalently as $\bar{s}_1\bar{s}_2$. Vertices in B^q that are not images of any value of X are **don't care** vertices. For example, $s_0\bar{s}_1s_2$ and $s_0s_1s_2$ are don't care vertices for $\mathcal{E}_1$. They may be included in any encoded value for simplification.

Definition 4.2 *An* **encoded implementation** *of a set of expressions $\{f_1, f_2, \ldots f_n\}$; where each f_i represents a $P \times B^{m-1}$ function; is a set of expressions $\{g_1, g_2, \ldots g_n\}$ where each g_i has been obtained from f_i by replacing each MV literal in f_i by its encoded value.*

Using encoding $\mathcal{E}_1$ with f_1 and f_2 we obtain the following encoded implementation,

$$g_1 = \bar{s}_1\,\bar{s}_2\,ak + \bar{s}_1\,s_2bk + c$$

$$g_2 = s_1\bar{s}_2\,aj + s_1s_2bj + d$$

s_0 does not appear in the expressions above. This is because the don't cares $\{s_0\bar{s}_1\,s_2, s_0s_1s_2\}$ were used to simplify the expressions. Note that $\bar{s}_2\,a + s_2b$ is now a kernel intersection, and can be extracted as an intermediate variable resulting in the following implementation.

$$g_1 = \bar{s}_1\,g_3k + c$$

$$g_2 = s_1g_3j + d$$

$$g_3 = \bar{s}_2a + s_2b$$

In contrast, consider the following encoding, $\mathcal{E}_2$:

$$X^{\{0\}}:001 \quad X^{\{1\}}:110 \quad X^{\{2\}}:100$$
$$X^{\{3\}}:111 \quad X^{\{4\}}:000 \quad X^{\{5\}}:011$$

This encoding results in the following encoded implementation:

$$g_1 = ((\bar{s}_0\,\bar{s}_1\,s_2 + s_0s_1\bar{s}_2\,)\,a + s_0\bar{s}_1\,\bar{s}_2\,b)k + c$$
$$g_2 = ((s_0s_1s_2 + \bar{s}_0\,\bar{s}_1\,\bar{s}_2\,)a + \bar{s}_0\,s_1s_2b)j + d$$

There are no kernel intersections between the two expressions. This implementation has 26 literals, compared to 12 literals with $\mathcal{E}_1$. Thus, we see that the choice of encoding has a profound effect on the resulting kernels and kernel intersections. In the above case the good encoding was intentionally selected so that it resulted in a good kernel intersection. The exact nature of the procedure involved in this selection will be described in the following sections.

The definitions and matrix representations given in Section 3 are for binary valued expressions of binary valued variables. We now extend these to binary valued expressions with MV variables. As in Section 2 we consider only one of the variables to be MV, and the remaining variables to be binary valued.

We modify the definitions for the kernel and co-kernel for a binary valued expression f of MV variables as follows:

Definition 4.3 *A* **kernel** *of an expression f is a product-term free quotient of f and a product term. The* **co-kernel** *is the product-term divisor used in obtaining the kernel.*

The **co-kernel cube matrix** is defined as follows:

Definition 4.4 *Each row represents a kernel (labeled by its co-kernel product-term) and each column a cube (labeled by this cube). Each non-zero entry in the matrix now has two parts. The first part is a positive integer that corresponds to the product-term in the expression (***product-term part***). The second part is the MV literal (***MV part***) which when ANDed with the cube corresponding to its column and the co-kernel corresponding to its row forms this product term.*

For the example above, both f_1 and f_2 have two kernels as shown below:

expression	*co − kernel*	*kernel*
f_1	1	$akX^{\{0,1\}} + bkX^{\{2\}} + c$
f_1	k	$aX^{\{0,1\}} + bX^{\{2\}}$
f_2	1	$ajX^{\{3,4\}} + bjX^{\{5\}} + d$
f_2	j	$aX^{\{3,4\}} + bX^{\{5\}}$

The co-kernel cube matrix for this is given below:

	a	b	ak	bk	aj	bj	c	d
1	0	0	1	2	0	0	3	0
	0	0	$X^{\{0,1\}}$	$X^{\{2\}}$	0	0	1	0
k	1	2	0	0	0	0	0	0
	$X^{\{0,1\}}$	$X^{\{2\}}$	0	0	0	0	0	0
1	0	0	0	0	4	5	0	6
	0	0	0	0	$X^{\{3,4\}}$	$X^{\{5\}}$	0	1
j	4	5	0	0	0	0	0	0
	$X^{\{3,4\}}$	$X^{\{5\}}$	0	0	0	0	0	0

In this matrix the product-term part is given above the MV part for each entry. The adjectives MV and binary will be used with co-kernel cube matrices and rectangles in order to distinguish between the multiple-valued and the binary case. We will demonstrate how this matrix is used to detect all potential kernel intersections. Each potential kernel intersection is then used to specify a set of constraints. An encoding must satisfy these constraints for the potential kernel intersection to be an actual kernel intersection in the encoded implementation.

A rectangle is defined as in the case for all binary variables. Associated with each rectangle $\mathcal{R}$ is a **constraint matrix** $M^{\mathcal{R}}$ whose entries are the MV parts of the entries of $\mathcal{R}$. For example, consider the MV co-kernel cube matrix given above. M_1, given below, is the constraint matrix for the rectangle {R(2,4), C(1,2)}.

$$M_1 = \begin{bmatrix} X^{\{0,1\}} & X^{\{2\}} \\ X^{\{3,4\}} & X^{\{5\}} \end{bmatrix}$$

Definition 4.5 *An encoding $\mathcal{E}$ of the values of X is said to* **satisfy** *M, if there exists for each row i and each column j, in M, sum-of-products representations of logic functions, k_i^r and k_j^c respectively (of the binary encoding variables) such that for all minterms X^a:*

$enc(X^a) \subseteq k_i^r.k_j^c$ if and only if $X^a \subseteq M(i,j)$

M is said to be **satisfied** *by $\mathcal{E}$. If $\mathcal{E}$ exists, then M is said to be* **satisfiable.**

Informally, $k_i^r.k_j^c$ contains precisely the vertices that are encoded values of the minterms in $M(i,j)$ and possibly don't care vertices.

For example consider the matrix M_1. The encoding $\mathcal{E}_1$ satisfies M_1 with:

$$k_1^r = \bar{s}_1 \quad k_2^r = s_1$$
$$k_1^c = \bar{s}_2 \quad k_2^c = s_2$$

Not all matrices are satisfiable. An example of a matrix that cannot be satisfied is (as will be shown in Section 5):

$$M_3 = \begin{bmatrix} X^{\{1\}} & X^{\{2\}} & X^{\{5\}} \\ X^{\{4\}} & X^{\{5\}} & X^{\{1\}} \end{bmatrix}$$

Definition 4.6 *A* **reduced constraint matrix** *M_r of a constraint matrix M has the following properties:*

1. *$M_r(i,j) \neq 0$*
2. *$M_r(i,j) \subseteq M(i,j)$*

Note that this definition includes the original constraint matrix.
An example of a reduced constraint matrix for M_1 is:

$$M_2 = \begin{bmatrix} X^{\{1\}} & X^{\{2\}} \\ X^{\{3,4\}} & X^{\{5\}} \end{bmatrix}$$

Definition 4.7 *An encoding $\mathcal{E}$ is said to* **satisfy** *a rectangle, $\mathcal{R}$ if:*

1. *It satisfies some reduced constraint matrix of that rectangle*
2. *for the encoding, $\sum_i |k_i^r| > 1$; $|k_i^r|$ is the number of cubes in k_i^r. i varies over the rows of the rectangle.*

$\mathcal{R}$ is said to be **satisfied** *by $\mathcal{E}$. If $\mathcal{E}$ exists, then $\mathcal{R}$ is said to be* **satisfiable.**

We will now show how the rectangles in the MV co-kernel cube matrix can be used to detect kernel intersections in any encoded implementation without knowing the encoding.

Theorem 4.1 *There is a one to one correspondence between the set of all satisfiable rectangles of the MV co-kernel cube matrix and the set of all kernel intersections of all encoded implementations.*

Proof

Part 1: Each satisfiable rectangle implies a kernel intersection in some encoded implementation.

Let $\mathcal{R}$ be the satisfiable rectangle. Thus, an encoding can be found that satisfies some reduced constraint matrix M_r corresponding to this rectangle. This encoding ensures that each literal $M_r(i,j)$ can be written as $k_i^r k_j^c$. We obtain a binary rectangle $\mathcal{R}_{enc}$ from $\mathcal{R}$ as follows. For each row of $\mathcal{R}$ there are $|k_i^r|$ rows in $\mathcal{R}_{enc}$. Let Λ_i be the co-kernel corresponding to row i in $\mathcal{R}$. The corresponding co-kernel for the lth of these $|k_i^r|$ rows in $\mathcal{R}_{enc}$ is $\Lambda_i\, k_{il}^r$ (k_{il}^r is the lth cube in k_i^r). Let Γ_j be the cube corresponding to the jth column in $\mathcal{R}$. For each column j in $\mathcal{R}$ we have $|k_j^c|$ columns in $\mathcal{R}_{enc}$. The lth of these corresponds to the cube $\Gamma_j\, k_{jl}^c$. Since $\sum_i |k_i^r| > 1$, $\mathcal{R}_{enc}$ has at least two rows.

With this encoding, $\mathcal{R}_{enc}$ is a rectangle in the co-kernel cube matrix of the encoded implementation. Since $\mathcal{R}_{enc}$ has at least two rows, it corresponds to a kernel intersection.

Part 2: Each kernel intersection in an encoded implementation corresponds to a satisfiable rectangle $\mathcal{R}$ in the MV co-kernel cube matrix. $\mathcal{R}$ is satisfied by the encoding.

The proof follows the reverse path traced out by the proof for Part 1. Any kernel intersection of the encoded implementation implies a rectangle, $\mathcal{R}_{enc}$, of at least two rows, in the binary co-kernel cube matrix. Let λ_i be the

co-kernel corresponding to row i of this rectangle and γ_j be the cube corresponding to column j. Let s_i be the cube in λ_i that has the bits used in the encoding of X. Similarly, let s_j be the cube in γ_j that has the bits used in the encoding of X. We construct a rectangle $\mathcal{R}^-$ in the MV co-kernel cube matrix as follows.

For each row and column of $\mathcal{R}_{enc}$ construct a row and column of $\mathcal{R}^-$. The co-kernel Λ_i, corresponding to row i, and the cube Γ_j, corresponding to column j in $\mathcal{R}^-$ are obtained as follows.

Case 1: $\exists j \mid s_j \neq 1$

$\Gamma_j = \gamma_j / s_j$. $\Lambda_i = \lambda_i / s_i$. The MV part of entry (i, j) in $\mathcal{R}^-$ is assigned the literal corresponding to $s_i\, s_j$.

Case 2: $\forall j,\ s_j = 1$

Here all the bits of the encoding space are in the co-kernel. Let X_{s_i} be the literal of X corresponding to the encoded bits s_i. Then, $\Gamma_j = \gamma_j$ and $\Lambda_i = (\lambda_i / s_i) X_{s_i}$, where / is the algebraic division operator. The MV part of each entry of the ith row in $\mathcal{R}^-$ is assigned the literal 1.

At this point there may exist columns in $\mathcal{R}^-$ that correspond to the same cube and rows that correspond to the same co-kernel. Merge each such set of columns (rows) into a single column (row) corresponding to that cube (co-kernel). The MV part of each entry of this column (row) is the disjunction of the MV parts of all the columns (rows) it replaces. $M^{\mathcal{R}^-}$ is necessarily a reduced constraint matrix of some matrix $M^{\mathcal{R}}$ for some rectangle $\mathcal{R}$ in the MV co-kernel cube matrix. Since $\mathcal{R}_{enc}$ has at least two rows (it was derived from a kernel intersection), $\mathcal{R}$ is a satisfiable rectangle. ■

Theorem 4.1 gives the relationship between potential kernel intersections for the MV variable expressions (rectangles in the MV co-kernel cube matrix) and actual kernel intersections for the encoded implementations (rectangles in the binary co-kernel cube matrix). Note the significance of the result of Theorem 4.1. It says that we can view all possible kernel intersections for all possible encodings by constructing the MV co-kernel matrix and considering all its satisfiable rectangles!

5 Satisfying the Constraints

Theorem 4.1 gives necessary and sufficient conditions, in terms of the MV co-kernel cube matrix, for the existence of kernel intersections in any encoded implementation. We now look at the necessary and sufficient conditions for a constraint matrix to be satisfiable and an encoding process that satisfies it.

Definition 5.1 *An encoding $\mathcal{E}$ of the values of X is said to* **minimally satisfy** *a constraint matrix M, if, for each row i and each column j in M, there exist cubes c_i^r and c_j^c respectively, in some q dimensional Boolean space such that for all minterms X^a:*

$enc(X^a) \subseteq c_i^r . c_j^c$ if and only if $X^a \subseteq M(i, j)$.

M is said to be **minimally satisfied** *by $\mathcal{E}$ and $\mathcal{E}$ is called a* **minimal encoding**. *If such an $\mathcal{E}$ exists, M is said to be* **minimally satisfiable**.

Informally, $c_i^r . c_j^c$ has only vertices corresponding to the minterms in $M(i, j)$ and possibly don't care vertices.

Note that this definition is a special case of Definition 4.5 with the additional constraint that each k_i^r and k_j^c be a single cube.

In Section 4 the encoding $\mathcal{E}_1$ minimally satisfies M_1. We observe that if the constraint matrix is minimally satisfied then both the kernel intersection and/or the set of co-kernels is the smallest (in terms of the number of cubes) of all possible encodings. This is desirable since it leads to a smaller number of cubes in the implemented logic.

We first determine the conditions under which a constraint matrix can be minimally satisfied. Later in this section we extend this to determine conditions for satisfying a matrix with a non-minimal encoding.

Definition 5.2 *An $m \times n$ constraint matrix M satisfies the* **intersection condition** *if:*

$$\forall i,\ \forall j\ M(i,j) = \left(\bigcup_{k=1}^{n} M(i,k)\right) \cap \left(\bigcup_{l=1}^{m} M(l,j)\right)$$

Informally, this implies that if a minterm X^a is present somewhere in row i and also in column j, then $X^a \subseteq M(i,j)$.

As an example, consider matrices M_1, M_2 and M_3 in Section 4. While M_1 and M_2 satisfy the intersection condition, it is not satisfied at $M_3(1,2)$.

Theorem 5.1 *If M is minimally satisfiable then it satisfies the intersection condition.*

Proof Suppose that the intersection condition is not satisfied at $M(i,j)$ and there exists a minimal encoding. Let c_i^r and c_j^c be the cubes associated with row i and column j in this encoding. There exists a minterm X^a such that $X^a \subset (\bigcup_{k=1}^{n} M(i,k)) \cap (\bigcup_{l=1}^{m} M(l,j))$ and $X^a \not\subset M(i,j)$ i.e. X^a occurs somewhere in row i and also in column j but not in $M(i,j)$. Therefore, $\exists k \neq j$, $\exists l \neq i$ s.t. $X^a \subseteq M(i,k)$ and $X^a \subseteq M(l,j)$. Thus, $enc(X^a) \subseteq c_i^r.c_k^c$ and $enc(X^a) \subseteq c_l^r.c_j^c$. Hence, $enc(X^a) \subset c_i^r$ and $enc(X^a) \subset c_j^c$ and therefore $enc(X^a) \subset c_i^r.c_j^c$. But $c_i^r.c_j^c$ has only encoded values of minterms of $M(i,j)$ and don't care vertices. This is a contradiction proving that no minimal encoding could have existed. ■

Theorem 5.1 determines the necessary condition for a constraint matrix to be minimally satisfiable. We will show that this condition is sufficient to ensure the satisfiability by giving a procedure for determining a satisfying encoding.

The procedure is related to the **face embedding problem**. In [9] this problem was defined for the optimal state assignment problem when the target technology is two-level logic. A brief statement of the problem is given below.

Definition 5.3 *Face Embedding Problem: Given a multi-valued variable X, and a set $\mathcal{X} = \{X_1, \ldots, X_n\}$ of literals of X, find an integer q, an encoding $\mathcal{E}$ of the values of X onto B^q, and cubes $c_i \subseteq B^q, 1 \leq i \leq n$ such that:*

$$enc(X^a) \subseteq c_i \quad \Leftrightarrow \quad X^a \subseteq X_i, \quad 1 \leq i \leq n$$

Each literal X_i to be embedded in c_i is called a face constraint.

Informally, each cube c_i contains only the minterms in X_i and possibly don't care vertices.

As an example, consider the set $\mathcal{X} = \{X^{\{0,1,2\}}, X^{\{3,4,5\}}, X^{\{0,1,3,4\}}, X^{\{2,5\}}\}$. The encoding $\mathcal{E}_1$ in Section 4 satisfies the face constraints specified by $\mathcal{X}$. The set of cubes corresponding to the four constraints are $\{\bar{s_1}, s_1, \bar{s_2}, s_2\}$.

Theorem 5.2 specifies how a minimal encoding may be obtained by solving a face embedding problem.

Theorem 5.2 *For an $m \times n$ constraint matrix M that satisfies the intersection condition, an encoding $\mathcal{E}$ minimally satisfies M if and only if it satisfies the following face constraints:*

$\bigcup_{k=1}^{n} M(i,k) \quad 1 \leq i \leq m$

$\bigcup_{l=1}^{m} M(l,j) \quad 1 \leq j \leq n$

Proof

If Part:

Let $\mathcal{E}$ be the encoding that satisfies the face constraints and let c_i^r and c_j^c be the cubes associated with row i and column j respectively. Since $enc(M(i,j)) \subseteq c_i^r$ and $enc(M(i,j)) \subseteq c_j^c$ then $enc(M(i,j)) \subseteq c_i^r.c_j^c$. To show that $c_i^r.c_j^c$ contains only vertices corresponding to $X^a \subseteq M(i,j)$, consider any X^a such that $enc(X^a) \subseteq c_i^r.c_j^c$ Then $enc(X^a) \subseteq c_i^r$ and $enc(X^a) \subseteq c_j^c$ implying that X^a is in row i and column j. Therefore, $X^a \subseteq (\bigcup_{k=1}^{n} M(i,k)) \cap (\bigcup_{l=1}^{m} M(l,j))$ This implies that $X^a \subseteq M(i,j)$ since M satisfies the intersection condition. Thus $\mathcal{E}$ minimally satisfies M.

Only If Part:

Suppose $\mathcal{E}$ minimally satisfies M and $X^a \subseteq M(i,j)$. Then there exist cubes c_i^r and c_j^c such that $enc(X^a) \subseteq c_i^r.c_j^c$. Then, $enc(X^a) \subseteq c_i^r$ and $enc(X^a) \subseteq c_j^c$. Thus, $enc(\bigcup_{k=1}^{n} M(i,k)) \subseteq c_i^r$ and $enc(\bigcup_{l=1}^{m} M(l,j)) \subseteq c_j^c$. To see that c_i^r has vertices corresponding only to minterms in row i consider any X^a in M such that $enc(X^a) \subseteq c_i^r$. X^a must be present in some column of M. Let l be one such column. Therefore, $enc(X^a) \subseteq c_l^c$. Thus $enc(X^a) \subseteq c_i^r.c_l^c$, implying $X^a \subseteq M(i,l)$ (since $\mathcal{E}$ is a minimal encoding). Therefore X^a does lie in row i satisfying its row face constraint. A similar argument holds for the face constraint of column j. Thus, $\mathcal{E}$ satisfies all the face constraints. ■

Theorem 5.2 gives the necessary and sufficient conditions that have to be satisfied by any minimal encoding. In [9] it was shown that the face constraints for any set can be satisfied with at most the number of bits (dimension of the encoding space) equal to the number of distinct values of X. Therefore, we know that if there is no intersection violation for a matrix M, then it is minimally satisfiable.

It may appear that by forcing the encoding to be minimal we are potentially losing some rectangles that may have non-minimal encodings satisfying the constraint matrix. In fact, this is not so as the following theorem states.

Theorem 5.3 *For a constraint matrix M, the following statements are equivalent:*

a) M satisfies the intersection condition

b) M is minimally satisfiable

c) M is satisfiable

Proof

1. $a \Rightarrow b$.

Since any set of face constraints can be satisfied, then by Theorem 5.2 there exists an encoding that minimally satisfies M.

2. $b \Rightarrow c$.

Obvious.

3. $c \Rightarrow a$.

Let $\mathcal{E}$ be a satisfying encoding and k_i^r and k_j^c be the expressions corresponding to row i and column j respectively. For each k_i^r , if $|k_i^r| > 1$ split row i in M into $|k_i^r|$ rows such that the lth of these rows contains all minterms X^a such that $enc(X^a) \subseteq k_{il}^r$. (Recall that k_{il}^r is the lth cube in k_i^r.) Do the same for all columns j for which $|k_j^c| > 1$. For this modified matrix M', obtained by this splitting process, there is a minimal encoding that satisfies it. In this encoding, k_{il}^r is the cube corresponding to the lth sub-row of row i, and k_{jp}^c is the cube corresponding to the pth sub column of column j, in M. Consider any minterm X^a that belongs both to row i and to column j in M. In M', X^a must belong to some row l that was part of row i in M and some column p that was part of column j in M. Since M' satisfies the intersection condition, $X^a \subseteq M'(l,p)$. Since $M'(l,p) \subseteq M(i,j)$, M also satisfies the intersection condition. ■

We note that even though Theorem 5.3 demonstrates the equivalence between satisfiability and minimal satisfiability, it does not follow that Theorem 5.2 is valid with "minimally" eliminated from its statement. There may exist a non-minimal encoding that does not satisfy the face constraints[2].

Since the intersection condition gives the precise relation for a matrix to be satisfiable, it can be used to derive a satisfiable reduced constraint matrix in case the original constraint matrix is not satisfiable. Let $M(i,j)$ be the entry of the constraint matrix M where the intersection condition is not met. Now, $X^\Psi = ((\bigcup_{k=1}^{n} M(i,k)) \cap (\bigcup_{l=1}^{m} M(l,j))) - M(i,j)$ is the literal that has the *offending* minterms. For each minterm, $X^a \subseteq X^\Psi$, we need to drop X^a from either row i or column j. When there are several entries in M where the intersection condition fails, the above reduction needs to be done at each violation. The problem of determining the least set of minterms to be dropped from M, so that the resulting reduced constraint matrix M_r is satisfiable, can be formulated as a covering problem. We do not pursue this further in this paper.

6 Non-Algebraic Factors

The previous sections demonstrated the encoding procedure used to obtain kernel intersections which are algebraic factors. In this section we extend this technique to extract non-algebraic (Boolean) factors.

We illustrate this with the following example. Consider the expressions:

$$f_1 = aX^{\{1\}} + bX^{\{2\}}$$

$$f_2 = aX^{\{1\}} + cX^{\{3\}}$$

$$f_3 = bX^{\{4\}} + cX^{\{3\}}$$

[2]For example, if M has a row with at least two minterms per entry, then this row could be split into two rows and then encode using the face constraints of the split matrix. This encoding can easily violate the face constraints of the original matrix

Each of these has a single kernel corresponding to the co-kernel 1. The co-kernel cube matrix for these expressions is given below. For clarity, only the MV part of the matrix has been shown since the product term part is obvious.

	a	b	c
1	$X^{\{1\}}$	$X^{\{2\}}$	0
1	$X^{\{1\}}$	0	$X^{\{3\}}$
1	0	$X^{\{4\}}$	$X^{\{3\}}$

Consider the rectangle $\mathcal{R}$ ={R(1, 2, 3), C(1, 2, 3)}. This does not correspond to the original definition of a rectangle since it has entries corresponding to the 0 literal of X. However, we will show that satisfying the constraint matrix of this rectangle will lead to Boolean factors in the encoded implementation.

The constraint matrix M for $\mathcal{R}$ is:

$$M = \begin{bmatrix} X^{\{1\}} & X^{\{2\}} & 0 \\ X^{\{1\}} & 0 & X^{\{3\}} \\ 0 & X^{\{4\}} & X^{\{3\}} \end{bmatrix}$$

There are no intersection violations in M. The face constraints for M are satisfied by the following encoding:

$$X^{\{1\}} : 01 \quad X^{\{2\}} : 11$$
$$X^{\{3\}} : 00 \quad X^{\{4\}} : 10$$

If s_0 and s_1 are the bits used in the encoding, then $c_1^r = s_1$ and $c_3^c = \bar{s}_0\bar{s}_1$. Therefore, $c_1^r . c_3^c = s_1.\bar{s}_0\bar{s}_1 = 0$. Also, $M(1,3) = 0$. This is not unexpected. The encoding ensures that $c_i^r . c_j^c$ has only vertices corresponding to minterms in $M(i,j)$ or don't care vertices. Therefore, if $M(i,j) = 0$ then $c_i^r . c_j^c$ can contain only don't care vertices. In this case the entire 2-dimensional encoding space was used up, therefore $c_1^r . c_3^c$ had to be the empty cube.

In general, the 0 entries in M result in null or don't care cubes in the encoded implementation. Therefore, the rectangle in the co-kernel cube matrix is a common non-algebraic factor between the expressions corresponding to its rows.

For the example above, the selected encoding leads to the following encoded implementation:

$$g_1 = s_1\ g_4$$
$$g_2 = \bar{s}_0\ g_4$$
$$g_3 = \bar{s}_1\ g_4$$
$$g_4 = \bar{s}_0\ s_1\ a + s_0\ b + \bar{s}_0\ \bar{s}_1\ c$$

g_4 is a non-algebraic factor of g_1, g_2 and g_3. Rows and columns that contain 0 literals may be added to the rectangles in the rectangular covering process to result in non-algebraic factors with potentially greater literal savings in the encoded implementation. In this example, there are no satisfiable rectangles without using the 0 entries, and hence no kernel intersections no matter what encoding is chosen.

We now generalize this idea. A constraint matrix that has entries corresponding to the zero literal is referred to as a **non-algebraic matrix** and the rectangle it was derived from referred to as a **non-algebraic rectangle**. The following theorem formalizes the concept presented through the example above.

Theorem 6.1 *Each satisfiable non-algebraic rectangle corresponds to a common Boolean factor in some encoded implementation.*

Proof Let M be the constraint matrix (non-algebraic) corresponding to such a rectangle. Since the rectangle is satisfiable, we know that for each (i,j), $k_i^r . k_j^c$ has only vertices corresponding to minterms in $M(i,j)$. Therefore, when $M(i,j) = 0$, $k_i^r . k_j^c$ must be either the null cube or have only don't care vertices. Hence, $k_i^r . k_j^c$ may be included in the factor while retaining the functionality. Thus, the rectangle in the co-kernel cube matrix results in a common Boolean factor. ∎

Note how this theorem extends Theorem 4.1 to consider potential factors that could not be detected by purely algebraic techniques. The definition of the reduced constraint matrix in Section 4 can now be relaxed for the non-algebraic case. The condition $M_r(i,j) \neq 0$ is not required. This leads to additional freedom in dropping the constraints, as discussed in Section 5, in order to make a constraint matrix satisfiable.

7 The Global Encoding Process

In Section 4, the procedure for detecting all potential kernel intersections was given and in Section 5 the encoding process for converting potential kernel intersections to actual kernel intersections was described. This was extended to recognize non-algebraic factors in Section 6. We now tie up these ideas for encoding of the values of X given $\{f_1, f_2 \ldots f_n\}$.

The first step is the extraction of kernels for each f_i. Even though this part was not described in the earlier sections, it can be done using the algorithms in [2] or [4]. The co-kernel cube-matrix is constructed for these kernels. A rectangular covering of this matrix with satisfiable rectangles (possibly non-algebraic), defines the set of potential common factors. We know that by solving the row and column face constraints for each such rectangle, we can convert that potential factor into an actual factor. Given a covering with several such rectangles we can realize all the factors by solving the row and column face constraints of all these rectangles simultaneously [3]. We know that given enough bits this can always be done. These constraints can be satisfied using the techniques described in [9] and [11].

An interesting side-effect of the proposed approach for encoding is that it performs the first stage of the multi-level logic optimization process, viz. the extraction of common sub-expressions. In fact, because of the don't cares that may be used in the encoding process, it is imperative that the encoding process itself do the initial decomposition/factorization. A two-level minimizer such as ESPRESSO ([3]) minimizing each node with the don't cares, followed by kernel extraction may not produce the same result as the decomposition/factorization done by the encoding process. This is because the minimizer works independently of the kernel extraction process and may use the don't cares in such a way that the algebraic factors do not correspond to those determined by the encoding process.

8 Conclusions

In this paper we have proposed an approach for the encoding of input variables for multi-level implementation. We have established a theoretical framework for this encoding process that can result in many large common factors. Necessary and sufficient conditions were determined for any encoding so as to result in a common factor. It was shown that these conditions are satisfied if and only if a set of face constraints is satisfied. The theoretical foundation of this approach is analogous to the theoretical foundation of the symbolic minimization techniques that were successful for two-level implementations [9]. This makes us quite optimistic about the quality of multi-level minimization algorithms based on these ideas. These will be reported in a subsequent paper. We are also investigating combining these techniques with output encoding, leading to state assignment algorithms for multi-level logic implementation.

References

[1] D. Bostick, G. D. Hachtel, R. Jacoby, M. R. Lightner, P. Moceyunas, C. R. Morrison, and D. Ravenscroft. The Boulder Optimal Logic Design system. In *Proceedings of the International Conference on Computer-Aided Design*, 1987.

[2] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. Wang. Multi-level logic optimization and the rectangular covering problem. In *Proceedings of the International Conference on Computer-Aided Design*, 1987.

[3] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers, 1984.

[4] R. K. Brayton and C. McMullen. The decomposition and factorization of Boolean expressions. In *Proceedings of the International Symposium on Circuits and Systems*, 1982.

[3] However, a little optimality may be lost here. It may be possible that the constraints for all the rectangles can be satisfied with fewer bits than needed to satisfy all the face constraints. The face constraints may be relaxed to allow for the following situation to occur. The encoding for a minterm X^a that does not appear anywhere in matrix M may be contained in c_i^r (c_i^c) corresponding to some row (column) i in M as long as it is not contained in any of the c_j^c (c_j^r) for any of the columns (rows) in M.

[5] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang. MIS: A multiple-level logic optimization system. *IEEE Transactions on Computer-Aided Design*, (6):1062–1081, November 1987.

[6] S. Devadas, H.-K. Ma, A. R. Newton, and A. Sangiovanni-Vincentelli. MUSTANG: State assignment of finite state machines targeting multi-level logic implementations. *IEEE Transactions on Computer-Aided Design*, (12):1290–1300, December 1988.

[7] D. Gregory, K. Bartlett, A. DeGeus, and G. Hachtel. SOCRATES: A system for automatically synthesizing and optimizing combinational logic. In *Proceedings of the Design Automation Conference*, 1986.

[8] G. De Micheli. Symbolic design of combinational and sequential logic circuits implemented by two-level logic macros. *IEEE Transactions on Computer-Aided Design*, (4):597–616, October 1986.

[9] G. De Micheli, R. K. Brayton, and A. Sangiovanni-Vincentelli. Optimal state assignment for finite state machines. *IEEE Transactions on Computer-Aided Design*, (3):269–285, July 1985.

[10] R. L. Rudell and A. Sangiovanni-Vincentelli. Multiple-valued minimization for PLA optimization. *IEEE Transactions on Computer-Aided Design*, (5):727–750, September 1987.

[11] T. Villa. *Constrained Encoding in Hypercubes: Algorithms and applications to logical synthesis.* Technical Report UCB/ERL M87/37, Electronics Research Lab., U. C. Berkeley, May 1987.