Multilevel Logic Minimization Using K-map XOR Patterns

Richard F. Tinder

Abstract—Entered variable XOR patterns are used in compressed Karnaugh maps to achieve gate-level minimum functions not possible with standard SOP and POS forms. Procedures are given for minimum cover extraction and verification. The design of a simple ALU illustrates multilevel-multiple output optimization.

I. INTRODUCTION

COMBINING entered variable (EV) subfunctions and XOR patterns in a Karnaugh map (K-map) extraction process [3] is a special and powerful form of K-map multilevel function minimization. Used with two-level logic forms (AND and OR functions) this multilevel minimization approach leads to XOR/SOP, EQV/POS and hybrid forms that often represent a substantial reduction in the hardware not possible otherwise. XOR/SOP and EQV/POS forms are those connecting p-terms with XOR operators or s-terms with EQV operators, respectively. Hybrid forms are those containing a mixture of these.

Conventional (1’s and 0’s) K-map methods involving XOR patterns have been used to obtain coefficient values for Reed-Muller XOR expansions [4]. Such methods have been shown to provide minimum fixed-polarity solutions to such expansions. Sasao [2] has shown that the use of conventional K-maps together with Boolean manipulation can be used to yield a XOR/SOP minimum for some functions. Except for the work cited in [3] no other published work is known to exist on the subject of logic minimization using EV K-map XOR patterns.

In the past the XOR gate (or EQV gate) has been viewed as a two-level device, meaning two units of path delay. Thus, its use with other gates, including other XOR or EQV gates, resulted in multiple units of path delay, hence the term multilevel. Also, XOR logic was known to be expensive, hardware-wise. But the emergence of CMOS IC technology has moved the XOR and EQV gates close to single level gates with respect to compactness and speed [3]. Furthermore, the arrival of FPGA’s and FPLD’s has made XOR/SOP and EQV/POS based logic synthesis practical [2].

In this article it will be shown how simple “pencil-and-paper” methods can be used to extract gate-minimum multilevel logic designs not yet possible by other method including the use of CAD techniques. The methods described in this article make possible multilevel IC designs that occupy much less real estate than would be possible for an equivalent two-level design, and often with little or no sacrifice in speed—an advantage for VLSI design.

Inherent in the K-map extraction methods and the verification of results that follow are the well known SOP and POS defining relations for the XOR and EQV functions [3]

\[
\begin{align*}
a \oplus b &= \bar{a} \cdot b + a \cdot \bar{b} & \text{SOP} \quad (1) \\
a \odot b &= (\bar{a} + b)(a + \bar{b}) & \text{POS} \quad (2) \\
a \oplus b &= \bar{a} \cdot \bar{b} + a \cdot b & \text{SOP} \quad (3) \\
a \odot b &= (\bar{a} + \bar{b})(a + b) & \text{POS}. \quad (4)
\end{align*}
\]

Here, (1) and (2) are logic duals, as are (3) and (4). Also important to verification of the K-map extraction results are two corollaries or lemmas of XOR algebra that express the interchangability of the operators [1]–[3].

Corollary I allows that if any two p-terms, \( \alpha \) and \( \beta \), of a function never take logic 1 simultaneously (hence are mutually disjoint), then

\[
\alpha \cdot \beta = 0 \quad \text{and} \quad \alpha \oplus \beta = \alpha + \beta. \quad (5)
\]

Corollary II by duality, permits that if any two s-terms, \( \alpha \) and \( \beta \), of a function never take logic 0 simultaneously, then

\[
\alpha + \beta = 1 \quad \text{and} \quad \alpha \odot \beta = \alpha \cdot \beta. \quad (6)
\]

Also used in verification of multilevel XOR forms are the distributivity laws in XOR algebra [3] given by

\[
\begin{align*}
(a \cdot b) \oplus (a \cdot c) &= a \cdot (b \oplus c) & \text{(7)} \\
(a + b) \odot (a + c) &= a + (b \odot c) & \text{(8)}
\end{align*}
\]

which are seen to be logically dual relations. Equation (7) can be called the factoring law of XOR algebra.

Where appropriate to do so, reference will be made to expressions (1)–(8) in discussions that follow. Reference will also be made to MINTERM code (logic 0 for a complemented variable and logic 1 for an uncomplemented variable), and to MAXTERM code which is the dual of MINTERM code. For example, expression (1) and (3) are used with MINTERM code while expressions (2) and (4) are used with MAXTERM code.

II. XOR TYPE PATTERNS AND EXTRATION OF GATE-MINIMUM COVER

There are four types of XOR patterns that can be easily identified in EV K-maps. They are

1) Diagonal patterns
2) Adjacent patterns
3) Offset patterns
4) Associative patterns.

Manuscript received October 16, 1993; revised June 29, 1994.

The author is with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752 USA.

IEEE Log Number 9415004.
TINDER: MULTILEVEL LOGIC MINIMIZATION USING K-MAP XOR PATTERNS

Of these, only the offset pattern requires third and higher order K-maps for its appearance. K-maps used in the following discussions are all MINTERM code based, but are used to extract gate-minimum functions in both MINTERM code and MAXTERM code.

Simple examples of the first three patterns are shown in Fig. 1(a) where a six variable function has been compressed into a third-order EV K-map. Empty cells 0 and 2 in Fig. 1(a) are to be disregarded so as to focus attention on the patterns: The diagonal pattern formed by cells 1 and 4 is read in MINTERM code as \( B \cdot X \cdot (A + C) \) or in MAXTERM code as \( B + X + A \cdot C \). Notice that the diagonal pattern lies in the \( B \) domain (\( B \) domain in MAXTERM code) "for all that is \( X \)," and that defining relations (1) and (2) are used for cells 1 and 4 to give \( A \cdot C \) and \( A \cdot C \), respectively, for MINTERM code and MAXTERM code. The adjacent pattern is formed by cells 5 and 7 and is read \( B \cdot C \cdot (A \cdot Z) \) in MINTERM code as \( B + C + A \cdot C \cdot Z \) in MAXTERM code. Here, the adjacent pattern lies at the intersection of domains \( B \) and \( C \) in MINTERM code (\( B + C \) in MAXTERM code), and defining relations (3) and (2) are used to obtain the MINTERM and MAXTERM code extraction, respectively. The offset pattern is formed by cells 5 and 6 and is read in MINTERM code as \( A \cdot Y \cdot (B \cdot C) \) and in MAXTERM code as \( A + Y + B \cdot C \). In this case, the offset pattern lies in the \( A \) domain (\( A \) in MAXTERM code) "for all that is \( Y \)," and the defining relations (1) and (2) are used for cells 5 and 6 to obtain \( B \cdot C \) and \( B \cdot C \), respectively.

Each of the three XOR type patterns extracted from Fig. 1(a) has a gate/input tally of 2/5 (excluding inverters). The gate/input tally is a measure of logic circuit cost (in hardware and real estate). It is defined as the ratio of the total number of gates to the total number of gate inputs including internal gate inputs and, unless stated otherwise, will exclude inverters and their inputs. Then, by comparison, the two-level logic gate/input tally for each is 3/8. Throughout this discussion it is assumed that any entered variable, for example \( X, Y \) or \( Z \), may represent a multivariable term of any complexity.

Associative patterns may combine with any or all of the other three patterns thereby forming compound patterns. For this reason associative patterns require special consideration and will be dealt with in the next section.

### A. Extraction of Gate-Minimum XOR Pattern

#### Cover From EV K-maps

Before illustrating the extraction process by example, it will be instructive to outline the extraction procedure. In this procedure reference will be made to MINTERM and MAXTERM codes. Since all \( K \)-maps are MINTERM code based, extraction of EQV/POS cover from them requires that the \( K \)-map domains be complemented, but not the entered variables [3]. Extraction of XOR/SOP cover follows conventional procedure. The following six step extraction procedure applies generally to all four types of XOR patterns.

#### Extraction Procedure

**Step I:** Identify the type of EV XOR pattern that exists in the \( K \)-map. A diagonal pattern requires identical \( K \)-map cell entries in diagonally located cells. An adjacent pattern requires complementary \( K \)-map cell entries in logically adjacent cells. An offset pattern requires identical cell entries in cells whose coordinates differ by two bits (a Hamming distance of two). Associative patterns require terms associated by an XOR or EQV connective in at least one cell.

**Step II:** Identify any \( K \)-map domains in which the XOR pattern exists. In reading the XOR pattern in MAXTERM code the domains are complemented whereas in MINTERM code they are not.

**Step III:** Extract the XOR pattern of type (1), (2) or (3) that exists by using the defining SOP or POS relations for XOR and EQV given by (1)–(4). Associative patterns are extracted in a manner similar to the extraction of EV \( s \)- and \( p \)-terms [3]. Thus, associative patterns with XOR connectives are extracted in MINTERM code while those with EQV connectives are extracted in MAXTERM code. Compound associative patterns involve some combination of associative pattern with one or more of the other three patterns. They may also include the intersection (ANDing) of patterns or the union (ORing) of patterns. In all cases involving an associative pattern, the associating connective must be preserved in the resulting expression.

**Step IV:** Extract any remaining two-level SOP or POS cover that may exist.

**Step V:** Combine into SOP or POS form the results of Steps (I) through (IV). The resulting expression may be altered as follows: Single complementation of an XOR associated term complements the XOR or EQV connective while double complementation of the associated terms retains the original connective [3].

**Step VI:** If necessary, test the validity of the extraction process. This can be done by introducing the \( K \)-map cell coordinates into the resulting expression. Generation of each cell subfunction of the \( K \)-map validates the extraction procedure.

**Examples:** The simplest associative patterns are formed between XOR-associated or EQV-associated variables and like variables in adjacent cells. Three examples are presented in Fig. 1(b)–(d), all representing second-order \( K \)-map compressions (two EV's). For the first-order EV \( K \)-map, shown in Fig. 1(b), the function \( E \) is read in MINTERM code as

\[
E_{XOR/SOP} = (A \cdot X) \oplus Y = (A + X) \odot Y
\]
and is seen to be a two-level function. Here, according to Step III of the extraction procedure, the associative XOR pattern is extracted in MINTERM code in SOP form with \( X \) located in the \( \tilde{A} \) domain, hence \( \tilde{A} \cdot \tilde{X} \). The \( E_{\text{XOR/SOP}} \) form can be converted to the \( E_{\text{EQV/POS}} \) form by single complementation in the \( \tilde{K} \)-map.

The function \( F \) in the second-order \( K \)-map of Fig. 1(c) is read in MAXTERM code, according to Step III and is given by

\[
F_{\text{EQV/POS}} = [(B + \tilde{Y}) \cdot X] \cdot A
\]

which is a three-level function. In this case the EQV connection associates the \( \tilde{Y} \) in cells 0 and 2 (hence \( B + \tilde{Y} \) in MAXTERM code) with the \( X \) in all four cells. The remaining POS cover in cell 0 is extracted with the don’t care (\( \phi \)) in cell 1 by ANDing the previous result with \( A \) as required by Step IV in the extraction procedure.

The function \( G \) in the third-order EV \( K \)-map, shown in Fig 1(d), is also read in MAXTERM code. Here, the EQV connection associates the \( X \)'s in cells 0, 1, 4 and 5 (thus \( B + X \) in MAXTERM code) with the \( \tilde{Y} \)'s in cells 5 and 7 (hence \( \tilde{A} + \tilde{C} + \tilde{Y} \)) giving the result

\[
G_{\text{EQV/POS}} = [(B + X) \cdot (\tilde{A} + \tilde{C} + \tilde{Y})] (\tilde{A} + C + X)
\]

which is also a three-level function. The term \( (\tilde{A} + C + X) \) removes the remaining POS cover in cells 4 and 6, as required by Step IV.

The two-level minimum results for \( E_{\text{SOP}}, F_{\text{POS}} \) and \( G_{\text{POS}} \) are

\[
E_{\text{SOP}} = XY + \tilde{A} \tilde{X} \tilde{Y} + AY
\]

\[
F_{\text{POS}} = (X + Y)(B + \tilde{X} + \tilde{Y})(B + X)A
\]

\[
G_{\text{POS}} = (B + X + Y)(\tilde{A} + \tilde{C} + \tilde{X} + \tilde{Y})(\tilde{A} + C + X) \cdot (A + B + X)(\tilde{A} + B + X + \tilde{Y}).
\]

The use of associative patterns often leads to significant reduction in hardware compared to the two-level SOP and POS forms. For example, function \( E_{\text{XOR/SOP}} \) has a minimum gate/input tally of 2/4 compared to 4/10 for \( E_{\text{SOP}} \), the two-level SOP minimum cover. The gate/input tally for \( F_{\text{EQV/POS}} \) is 3/6 compared to 4/11 for the \( F_{\text{POS}} \) cover, and function \( G_{\text{EQV/POS}} \) has a minimum gate/input tally of 12/12 compared to 6/22 for \( G_{\text{POS}} \), the two-level POS minimum result, all excluding inverters.

XOR patterns may be combined very effectively to yield gate-minimum results. Shown in Fig. 2(a) is a second-order compression where diagonal, adjacent and offset patterns are associated in MINTERM code by the XOR operator in cell 1. Here, (1) is applied to the diagonal pattern (cells 1 and 4) in the \( B \) domain for all that is \( X \) to yield \( BX (A \oplus C) \). This pattern is then associated with the intersection (ANDing) of the adjacent pattern \( (A \oplus Y) \) and the offset pattern \( (B \oplus C) \) in cells 1, 2, 5 and 6 to give the gate-minimum, three-level result

\[
H_{\text{XOR/SOP}} = \lbrack BX (A \oplus C) \rbrack \oplus \lbrack (A \oplus Y) (B \oplus C) \rbrack
\]

with a gate/input tally of 6/13. Equation (3) is used for the adjacent pattern \( \tilde{Y} \) in the \( \tilde{A} \) domain and \( Y \) in the \( A \) domain

\[
I_{\text{SOP}} = (A + B + X + \tilde{Y})(A + B + \tilde{X} + Y) \cdot (A + B + X + Z)(\tilde{A} + B + \tilde{X} + \tilde{Z}) \cdot (A + B + Y + \tilde{Z})(\tilde{A} + B + \tilde{Y} + Z)
\]

which has a gate/input tally of 7/31.

Compound (interconnected) associative patterns are also possible and may lead to gate-minimum functions, although often of a higher level (hence slower) than those where there is no interconnection between associative patterns. Two examples are given in Fig. 2(b) and (c), both third-order compressions (hence three EV's). Function I is extracted in MAXTERM code yielding the four-level, gate-minimum result

\[
I_{\text{EQV/POS}} = [B + (A \oplus X)] \cdot [Y + (A \oplus B)] \cdot [\tilde{A} + Z]
\]

while (1) is applied to the offset pattern \( \tilde{Y} \) and \( Y \) in the \( BC \) domain, and \( \tilde{Y} \) and \( Y \) in the \( BC \) domain. This is compared to the two-level result

\[
H_{\text{SOP}} = \tilde{A} BCX Y + \tilde{A} BCX + \tilde{A} BCY + BCXY + \tilde{A} BCY + \tilde{A} BCY
\]

which has a gate/input tally of 7/14. Extraction involves the association of an adjacent pattern and a diagonal pattern with the three EQV connectives. The adjacent pattern in domain \( B \) (cells 0 and 2) requires the use of (4) to give \( [B + (A \oplus X)] \). This is associated with the diagonal pattern in cells 0 and 3 by using (4) for all that is \( Y \) to give \( Y + (A \oplus B) \), but is also associated with the cell 3 connective in domain \( \tilde{A} \) for all that is \( Z \). Notice that the terms in square brackets are commutative. For comparison purposes the two-level POS result for function I is given by

\[
I_{\text{POS}} = (A + B + X + \tilde{Y})(A + B + \tilde{X} + Y) \cdot (A + B + X + Z)(\tilde{A} + B + \tilde{X} + \tilde{Z}) \cdot (A + B + Y + \tilde{Z})(\tilde{A} + B + \tilde{Y} + Z)
\]

and has a gate/input tally of 7/30.

The function J in Fig. 2(c) is extracted in MINTERM code giving the four-level, gate-minimum result

\[
J_{\text{XOR/SOP}} = [D(B \oplus Z)(A \oplus C)] \oplus [B(A \oplus X)(C \oplus D)] \oplus [C(D \oplus Y)(A \oplus B)]
\]

with a gate/input tally of 11/25. This function is extracted as three sets of two intersecting patterns all associated by the
TINDER: MULTILEVEL LOGIC MINIMIZATION USING K-MAP XOR PATTERNS

<table>
<thead>
<tr>
<th>Function</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multilevel</td>
<td>2/4</td>
<td>4/7</td>
<td>5/13</td>
<td>7/14</td>
<td>8/15</td>
<td>12/36</td>
</tr>
<tr>
<td>Two-level</td>
<td>7/13</td>
<td>7/14</td>
<td>11/21</td>
<td>12/36</td>
<td>12/35</td>
<td>20/81</td>
</tr>
</tbody>
</table>

three XOR connectives. The “Z” set consists of adjacent and diagonal patterns given by (3) as \((B \otimes Z)\) and \((A \otimes C)\), respectively, which intersect (AND) in the \(D\) domain. The “X” set consists of adjacent and offset patterns given by \((1)\) as \((A \oplus X)\) and \((C \oplus D)\), respectively, which intersect in the \(B\) domain. Finally, the “Y” set also consists of adjacent and offset patterns such that \((3)\) yields \((D \otimes Y)\) and \((A \otimes B)\), respectively, which intersect in the \(C\) domain. As in the previous example, the terms in square brackets are commutative. In comparison the two-level SOP minimum for function \(J\) is given by

\[
J_{SOP} = ABCDXYZ + ABCDXZ + ABZCYD + ACZDX + BCDXYZ + ABCDZ
\]

and has a gate/input tally of 13/74.

Both four-level functions, \(I_{EQQV/POS}\) and \(J_{XOR/SOP}\) are easily verified by introducing in turn the cell coordinates for each into the expression. Generation of the subfunctions in each cell validates the extraction process.

The gate/input tallies for all five functions represented previously were given exclusive of inverters. When account is taken of the inverters required for inputs assumed to arrive active high, the gate/input tally differentials between the multilevel results and the two-level results increases significantly. These gate/input tallies are compared in Table I.

<table>
<thead>
<tr>
<th>Function</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multilevel</td>
<td>2/4</td>
<td>4/7</td>
<td>5/13</td>
<td>7/14</td>
<td>8/15</td>
<td>12/36</td>
</tr>
<tr>
<td>Two-level</td>
<td>7/13</td>
<td>7/14</td>
<td>11/21</td>
<td>12/36</td>
<td>12/35</td>
<td>20/81</td>
</tr>
</tbody>
</table>

There are still other factors that may increase significantly the gate/input tally differential between multilevel and standard two-level SOP and POS minimum forms. These include gate fan-in restrictions, static hazard cover and multiple output optimization considerations. For example, if active high inputs are assumed for function \(J\), six \(p\)-terms must be added to the two-level SOP result to cover eleven possible static hazards, and given likely fan-in restrictions (there are now 18 \(p\)-terms for the two-level result), the differential in gate/input tallies between the \(J_{SOP}\) and \(J_{XOR/SOP}\) forms is expected to widen considerably. Work is in progress on the analysis and elimination of static hazards in multilevel XOR forms and will be presented in a future paper. For logic families such as CMOS, propagation delay is increased significantly with increasing numbers of gate inputs, and it is here where the multilevel XOR forms often have a distinct advantage over their two-level counterparts. For example, the largest number of inputs to any gate in the implementation of function \(J_{XOR/SOP}\) is three whereas for the two-level function \(J_{SOP}\) it is 12 without hazard cover. An example of how multiple output optimization considerations may further increase the gate/input tally differential between the multilevel and two-level approaches to design is given in Section II-D.

B. Algebraic Verification of Optimal XOR Function Extraction From K-maps

Verification of the multilevel XOR forms begins by direct K-map extraction of the function in SOP or POS form by using MINTERM code for XOR connectives and MAXTERM code for EQV connectives. It then proceeds by applying corollary I (5) or corollary II (6) together with distributivity, commutivity and a defining relation given by (1)-(3) or (4).

As an example, consider the function \(H\) in Fig. 2(a) which is extracted in MINTERM code. Verification of this function is accomplished in six steps

\[
H = ABC(X \oplus \bar{Y}) + AB\bar{C}X + \bar{A}\bar{B}C
\]

Step 1 From K-map

\[
= [ABC(X \oplus \bar{Y})] \oplus [A\bar{B}C] \oplus [A\bar{B}C]
\]

Step 2 By (5)

\[
= (ABCX) \oplus (AB\bar{C}Y) \oplus (A\bar{B}C)
\]

Step 3 By (7)

\[
= [\bar{B}X(A \oplus (C))] \oplus [\bar{B}C((A \bar{Y}) \oplus (AY))]
\]

Step 4 By (7)

\[
= [\bar{B}(A \oplus C)] \oplus [B\bar{C}(A \oplus Y)]
\]

Step 5 By (5), (7), (1) and (3)

\[
= [\bar{B}(A \oplus C)] \oplus [(A \oplus Y)(B \oplus C)]
\]

Step 6 By (7) and (1).

Notice that in going from step 3 to step 4 the commutative law of XOR algebra \(3\) is used.

As a second example, consider function \(I\) in Fig. 2(b) which has been extracted in MAXTERM code. Verification of this function is also accomplished in six steps

\[
I = (A + B + X \otimes Y)(A + B + X \otimes Z)
\]

Step 1 From K-map

\[
= (A + B + X \otimes Y) \oplus (A + B + X \otimes Z)
\]

Step 2 By (6)

\[
= (A + B + X) \oplus (A + B + Y) \oplus (A + B + \bar{X})
\]

Step 3 By (8)

\[
= [B + (A + X) \oplus (A + \bar{X})] \oplus [Y + (A + B) \oplus (A + \bar{B})]
\]

Step 4 By (8)

\[
= [B + (A + X)(A + \bar{X})] \oplus [Y + (A + B)(A + \bar{B})]
\]

Step 5 By (6)

\[
= [B + (A \oplus X) \oplus Y + (A \oplus B)] \oplus [A + Z]
\]

Step 6 By (4).

In going from step 3 to step 4 commutivity was applied before application of (8). Also, in step 4, \(B \oplus \bar{B} = 0\).
C. K-map Plotting and Entered Variable XOR Patterns

At the onset let it be understood that one does not usually hunt for applications of the XOR pattern minimization methods described here. It is possible to do this, as the example in this section illustrates, but it is more likely that such methods would be applied to EV XOR patterns that occur naturally in the design of a variety of combinational logic devices. Examples of these include a 2×2 bit “fast” multiplier, comparator design, gray-to-binary code conversion XS3-to-NBCD code conversion, ALU design of the type illustrated herein, binary-to-2’s complement conversion, and NBCD to 84-2-1 code conversion, to name but a few [3]. EV XOR patterns may also occur quite naturally in the design of some state machines as, for example, the design of a three-bit up/down Gray code counter [3].

EV K-map ploting for the purpose of extracting a gate-minimum cover by using XOR patterns is not an exact science, and it is often difficult to find the optimum K-map compression involving specific EV’s, hence specific K-map axis variables. However, for some functions it is possible to plot the map properly directly from the canonical form, as illustrated by the example that follows. It is also possible in a combinational logic design to optimally arrange the entries in an EV truth table to best suit the formation of XOR patterns, but this is more the exception than the rule. The design of an ALU in the next section is such an example.

For some relatively simple functions, the K-map plotting process can be deduced directly from the canonical expression. Consider the simple function given in canonical code form

\[ f(W, X, Y, Z) = \Sigma m(1, 2, 3, 6, 7, 8, 12, 13). \]

Shown in Fig. 3 are the conventional (1’s and 0’s) K-map and the second-order compression (two EV’s) K-map derived directly from the conventional K-map. The two-level SOP minimum and the multilevel XOR/SOP gate-minimum forms are

\[ f_{SOP} = \overline{W} \overline{X} Z + W X \overline{Y} + W \overline{Y} \overline{Z} + \overline{W} Y \]

and

\[ f_{XOR/SOP} = [(Y \oplus W)] \oplus (\overline{X} \overline{Y} Z) \]

which have gate/input tallies of 5/15 and 3/7, respectively. The second-order K-map in Fig. 3(b) is deduced directly from the K-map in Fig. 3(a) by observing that \( W \oplus X \) exists in the \( YZ = 01 \) column, with \( W \) and \( \overline{W} \) located in adjacent \( YZ \) columns. Thus, by taking \( Y \) and \( Z \) as the axis variables and \( W \) and \( X \) as the EV’s for the compressed K-map the XOR patterns appear allowing one to easily extract gate-minimum results. Only in one other compressed K-map are the gate-minimum XOR patterns obvious, and that is shown in Fig. 3(c). In all four other compressed K-map possibilities, those having axes \( W/X, X/Z, W/Z \) and \( W/Y \), the XOR patterns shown in Fig. 3(b) and (c) disappear making a gate-minimum extraction without extensive Boolean manipulation very difficult if not impossible. Notice that the compressed K-map in Fig. 3(c) is not easily obtained from the that of Fig. 3(a).

For complex functions involving five or more variables, the process of generating a gate-minimum result by using XOR EV patterns becomes increasingly more a matter of trial and error as the number of variables increases. Again, the application of the EV XOR pattern approach to design is left more to the natural occurrence of such patterns than it is to the hunt-and-choose method. However, if it is suspected that XOR patterns occur naturally in a given function and if one is familiar with conventional (1’s and 0’s) K-map methods for five or more variables, it is possible to deduce a compressed K-map that will yield XOR/SOP or EQV/POS forms, but which may not represent gate-minimum functions.

D. A Simple Design Example

As an example of multilevel optimization, a simple 1-bit slice arithmetic logic unit (ALU) is designed by using the multilevel XOR approach. The operation table for the ALU is shown in Fig. 4(a) where the eight operations are optimally placed. Here, \( A \) and \( B \) are the data inputs, \( Ci \) and \( Co \) are the carryin and carryout, \( F \) represents the resultant function (arithmetic or logic), and \( M, S_1 \) and \( S_0 \) are the operation controls, all applying to each stage of the ALU. Subtraction is carried out in 2’s complement and the LSB \( Ci \) is the carryin to the least significant bit stage of the ALU.

The function \( F \) and carryout \( Co \) are compressed into the second-order EV K-maps shown in Fig. 4(b) and (c), each representing a fourth-order compression (hence, four EV’s). Both are extracted in MINTERM code giving the optimum four-level results

\[ F_{XOR/SOP} = (A \oplus S_0) \oplus (M Ci) \oplus (S_1 B) + MS_1 B \]

\[ Co_{XOR/SOP} = Ci(A \oplus S_0) \oplus (S_1 B) + (S_1 B)(A \oplus S_0) \]

with a combined gate/input tally of 10/21. Here, terms \( A \oplus S_0 \), \( S_1 B \) and \( (A \oplus S_0) \oplus (S_1 B) \) are seen to be shared between functions \( F \) and \( Co \), a form of multioutput optimization. In
TINDER: MULTILEVEL LOGIC MINIMIZATION USING K-MAP XOR PATTERNS

Fig. 4. Design of a simple ALU. (a) Operation table. (b) Compressed EV K-map and optimum multilevel cover for carryout Co.

Future work should include computer algorithms that will automate the multilevel minimization process described in this work. Given the complexity of many of the patterns, it will not be easy to accomplish this. Furthermore, such algorithms must be combined with suitable two-level minimization algorithms since XOR multilevel results often involve a two-level SOP or POS component.

REFERENCES


III. CONCLUSION AND FUTURE WORK

Use of entered variable XOR patterns in compressed Karnaugh maps is shown to yield gate-minimum results not possible by conventional mapping methods. The XOR pattern method is shown to be especially advantageous in the design of ALU’s. This form of multilevel minimization usually leads to hardware savings in excess of 50 percent. Furthermore, multilevel XOR functions are more easily tested than their two-level SOP or POS counterparts since a single input change in a multilevel XOR function forces a function change. If fan-in restrictions are placed on the two-level result, speed may even favor the XOR method, particularly for CMOS implementations of the functions.

Richard F. Tinder received the B.S. degree in chemical engineering in 1955, and the M.S. and Ph.D. degrees in materials science in 1956 and 1962, all from the University of California (Berkeley).

Since then he has been on the faculty of Washington State University, Pullman, WA, where he is presently Professor of Electrical Engineering and Computer Science. His interests are broadly based, and he has published numerous articles in the areas of matter tensor properties, surface physics, shock dynamics, crystalline microplasticity and digital logic design. His current interests include asynchronous programmable sequencers, asynchronous internally clocked systems, arbiters and metastability, asynchronous sequential ALU’s and multilevel logic design using exclusive-OR forms. He is the author of DIGITAL ENGINEERING DESIGN: A Modern Approach (Prentice Hall, 1991) and is holder of a 1991 patent, "Microprogrammable Asynchronous Controllers for Digital Electronic Systems." In completely unrelated work, he has worked on safety enclosures and systems for high speed vehicles and holds a recent major patent in this area.