Efficient Sum-To-One Subsets Algorithm for Logic Optimization

Kuang-Chien Chen
Fujitsu America Inc., San Jose

Masahiro Fujita
Fujitsu Labs. Ltd., Japan

Abstract
Given a set of functions \( G = \{ g_1, g_2, \ldots, g_k \} \), where \( \sum_{i=1}^{k} g_i = 1 \) and each \( g_i \), \( i = 1, \ldots, k \), is a function of \( X = \{ x_1, x_2, \ldots, x_n \} \). The sum-to-one subsets problem is to find all minimal subsets of the functions in \( G \) which add up to 1. A related problem, i.e., finding a minimum-cost sum-to-one subset based on a given cost function, is an important problem that often occurs in logic optimization algorithms. In this paper, we present efficient procedures for solving both sum-to-one subsets and minimum-cost sum-to-one subset problems. We shall apply these techniques to multi-level network optimization algorithms, and will demonstrate that both the efficiency and quality of these algorithms are greatly improved. The application of these techniques to multi-node minimization using Boolean relations will also be discussed.

1. Introduction
Algorithms for optimizing multi-level combinational networks have been greatly improved in recent years [1][3][7][8][12][13]. In [7], an optimization algorithm RENO (Resynthesis for Network Optimization) was proposed in which a given network is minimized by optimally resynthesizing each gate in the network.

As will be discussed, the resynthesis problem in RENO can be transformed into a minimum-cost sum-to-one subset problem. Therefore, the efficiency and quality of the RENO algorithm depend largely on how well this problem can be solved. In the next section we will formally define the minimum-cost sum-to-one subset problem and review the existing approaches for solving it. Then, we discuss an innovative technique for solving the sum-to-one subsets problem by using Ordered Binary Decision Diagrams [6]. An algorithm for solving the minimum-cost sum-to-one subset problem is then developed by incorporating this technique. We have applied these techniques to RENO, and will demonstrate that they can greatly enhance RENO's capabilities in both area and delay minimization. Comparisons with algorithms in SIS-1.0 will be presented. We shall also discuss the application of these techniques to solve the problem of multi-node minimization using Boolean relations [4].

2. The Minimum-Cost Sum-To-One Subset Problem
We give basic definitions and explain the problem to be solved more precisely.

Definition 1: Given a set of functions \( G = \{ g_1, g_2, \ldots, g_k \} \), where \( \sum_{i=1}^{k} g_i = 1 \) and each \( g_i \), \( i = 1, \ldots, k \), is a function of \( X = \{ x_1, x_2, \ldots, x_n \} \). The sum-to-one subsets problem is to find all minimal (i.e., irredundant) subsets of the functions \( g_1, g_2, \ldots, g_k \) which add up to 1 (or tautology). Suppose each \( g_i \), \( i = 1, \ldots, k \), has a given cost. The minimum-cost sum-to-one subset problem is to find a sum-to-one subset from \( G \) which has a minimum cost. Henceforth, we shall refer to the sum-to-one subsets problem as STSOs, and to the minimum-cost sum-to-one subset problem as MC-STSOs.

STSOs and MC-STSOs problems can arise in different settings. But here we shall mainly consider solving the STSOs and MC-STSOs problems that arise in optimizing a multi-level combinational network \( N \). \( N \) has \( m \) primary outputs, \( m \) primary inputs \( X = \{ x_1, x_2, \ldots, x_n \} \), and the function realized by a gate \( v_i \) in \( N \) with respect to the primary inputs is denoted by \( f(v_i) \).

An ordered binary decision diagram (OBDD) is a rooted directed graph representation of a Boolean function \( f(x_1, x_2, \ldots, x_n) \) with respect to a given ordering of the variables. Without loss of generality, suppose the ordering of the variables is \( x_1, x_2, \ldots, x_n \), then each variable \( x_i \) is associated with a unique index \( i, 1 \leq i \leq n \).

Each non-terminal vertex \( w \) in an OBDD is associated with a variable \( x_i \in \{ x_1, x_2, \ldots, x_n \} \) and is labeled with the index of that variable. A non-terminal vertex \( w \) has two children vertices \( low(w) \) and \( high(w) \). The edge between \( w \) and \( low(w) \) is labeled with the negative literal \( \overline{x_i} \) and the edge between \( w \) and \( high(w) \) is labeled with the positive literal \( x_i \). Each OBDD also has two terminal vertices associated with constant 0 and 1 respectively. A 1-path in an OBDD is a sequence of connected vertices, \( \{ w_1, w_2, \ldots, w_t \} \), where \( w_1 \) is the root of the OBDD, \( w_2, \ldots, w_{t-1} \) are non-terminal vertices and \( w_t \) is the constant 1-terminal vertex. A positive term associated with a 1-path consists of all the positive literals appearing on the edges of the path. The depth of a 1-path is the largest index of the non-terminal vertices in the path.

When resynthesizing a gate \( v_i \) in the RENO algorithm, we first choose a set of candidate cubes \( c_1, c_2, \ldots, c_t \) whose functions with respect to the primary inputs of the network are \( f(c_1), f(c_2), \ldots, f(c_t) \) respectively, and a function \( h \) to be resynthesized using those candidate cubes (e.g., \( h \) can simply be \( f(v_i) \)). Each function \( f(c_i) \) has an associated cost (e.g., for area minimization the cost can be the number of literals in the corresponding candidate cube \( c_i \) ) and the goal is to choose a minimum-cost subset of functions from \( f(c_1), f(c_2), \ldots, f(c_t) \) which can realize \( h \) (a basic assumption here is \( h \subseteq \sum_{i=1}^{t} f(c_i) \), which is always satisfied by the candidate cubes chosen in RENO [7]).

Theorem 1: Let \( G = \{ t_1, t_2, \ldots, t_i \} \), and \( h \) be Boolean functions with respect to variables \( X = \{ x_1, x_2, \ldots, x_n \} \). Then

\[ h \subseteq \sum_{i=1}^{k} t_i \quad \text{if and only if} \quad \sum_{i=1}^{k} g_i = 1, \]

\[ g_i = t_i \] is the generalized cofactor of \( t_i \) w.r.t. \( h \) [13].

Base Theorem 1, the gate resynthesis problem in RENO can be formulated as a MC-STSO problem:

1. Since \( h \subseteq \sum_{i=1}^{k} f(c_i) \), we have \( \sum_{i=1}^{k} f(c_i) h = 1 \).
2. Since $\sum f(c_i) = 1$, we can find all the STOSs of the cofactors $f(c_1), f(c_2), ... , f(c_n)$. Each STOS represents a subset of the $f(c_i)$'s which can realize $h$, and thus represents a subset of the $c_i$'s which can be used for reсинthesizing the gate $G_i$. Therefore, a solution of the optimal gate reсинthesization problem corresponds to a STOS with the minimum cost.

3. Solving the MC-STOS Problem

A naive way to solve the MC-STOS problem is to generate all the STOSs, and then choose a subset with a minimum cost. But in practice, all the STOSs are seldom generated. Instead, since $\sum f(c_i) = 1$, a tautology-checking algorithm [7][9] has been used to derive a covering table in which a set of columns that cover all the rows corresponds to a STOS of $f(c_i)$, $f(c_2), ... , f(c_n)$. A MC-STOS can thus be found by finding a minimum-cost column cover of the covering table. In RENO, in order to handle large networks, functions are represented in OBDDs and an OBDD-based tautology-checking algorithm has been developed [7] which is briefly discussed in the next section.

3.1 OBDD-based Tautology-Checking

Let the set of functions in a given MC-STOS problem be $G = \{g_1, g_2, ..., g_k\}$, where $\sum g_i = 1$ and $g_i$'s are represented in OBDDs. The solution can be obtained by the OBDD-based tautology-checking procedure shown in Figure 1[7].

Example 1: Suppose given 6 functions:

$g_1 = x_1^2 + x_2^2 \quad g_2 = x_2^2 \quad g_3 = x_1^2 + x_2^2 \quad g_4 = x_1^2 \quad g_5 = x_1^2 x_2 \quad g_6 = x_1 x_2$

By applying the tautology-checking procedure, we get an expression which represents a covering table:

$(g_1 + g_3 + g_5)(g_1 + g_3 + g_6)(g_1 + g_2 + g_5)(g_1 + g_5)(g_1 + g_2 + g_3 + g_4)$

which is simplified to $(g_1 + g_5)(g_1 + g_3 + g_4)(g_1 + g_5)(g_1 + g_3 + g_4)(g_1 + g_5)(g_1 + g_3 + g_4)(g_1 + g_3 + g_4)(g_1 + g_3 + g_4)$.

Experiments show that the efficiency of Procedure OBDD_tautology_checking is mainly determined by the number of variables that the functions $g_i$'s depend on (i.e., $x_1, x_2, ..., x_n$). Since tautology-checking applies Shannon expansions repeatedly, we need to partition using all the $n$ variables in the worst case. In the next section, we shall discuss how to improve the efficiency of this process.

3.2 An OBDD-based STOSs Algorithm

We discuss a technique for solving the STOSs problem which will lead to a significant speedup of the OBDD-based tautology-checking procedure. This following lemma is a generalization of Lemma 5.10.1 in [5].

Lemma 1: Let $G = \{g_1, g_2, ..., g_k\}$ be a set of functions of variables $X = \{x_1, ..., x_n\}$, and let $A_1, A_2, ..., A_k$ be Boolean variables (they will be referred to as A-variables). Let $S = \{g_1, g_2, ..., g_k\} (1 \leq s \leq k)$ be a subset of $G$. Then the following conditions are equivalent:

(i) $g_1 + g_2 + ... + g_s = 1$
(ii) $A_1 A_2 ... A_s$ implies $A_1 g_1 + A_2 g_2 + ... + A_s g_s$.

So the problem of finding STOSs (not necessarily minimal) of a set of functions $G$ is reduced to that of finding the terms which consist of only positive literals of the A-variables and imply the summation $A_1 g_1 + A_2 g_2 + ... + A_s g_s$. In the following, we shall refer to such terms as A-terms. Lemma 1 leads to a method for deriving all the minimal STOSs.

Theorem 3: Let $G = \{g_1, g_2, ..., g_k\}$ be a set of functions of variables $X = \{x_1, ..., x_n\}$, and let $A_1, A_2, ..., A_k$ be Boolean variables. Let $S = \{g_1, g_2, ..., g_k\} (1 \leq s \leq k)$ be a subset of $G$. Then $S$ is a minimal sum-to-one subset of $G$ if and only if the product term $A_1 A_2 ... A_s$ is a prime implicant of $A_1 g_1 + A_2 g_2 + ... + A_s g_s$ (such terms shall be referred to as prime A-terms).

Example 2: Consider the 6 functions used in Example 1. The following summation expression is obtained:

$A_1 x_1^2 + A_2 x_2 + A_2 g_2 + A_1 g_3 + A_2 g_4 + A_2 g_5 = A_1 x_1^2 + A_2 x_2^2 + A_3 g_3 + A_4 g_4 + A_5 g_2 + A_6 g_6 + A_7 g_7 + A_1 x_1^2 + A_2 x_2^2 + A_3 g_3 + A_4 g_4 + A_5 g_2 + A_6 g_6 + A_7 g_7 (\ast)$

There are totally 31 prime implicants for the expression above, but only 3 of them consists of only the A-variables: $A_1 A_2 A_3, A_1 A_2 A_6, A_1 A_5 A_3$. Therefore, the STOSs corresponding to the prime A-terms are $(g_1, g_2, g_3), (g_1, g_2, g_3, g_4), (g_2, g_3, g_4, g_5)$.

In the process of reсинthesizing a gate in RENO, usually we can find many candidate cubes. Therefore, if we apply Theorem 3, there are many functions in $G$ and we need to use many A-variables and will be time-consuming to derive all the prime implicants for the summation expression. Also, how to order all the X- and A-variables is a problem. These problems, however, can be alleviated by observing that we don't need to derive all the prime implicants for the expression. What we need are only those prime A-terms. Also, all the A-variables are positive unate in the expression (i.e.,
only $A_i$'s, but not $\overline{A_i}$'s appear). Therefore, all the prime $A$-terms consist of only positive literals. Base on those observations, the procedure shown in Figure 2 enables us to efficiently derive all the prime $A$-terms without generating all prime implicates for the summation expression.

Theorem 4: Procedure OBDD_STOSs derives all the prime $A$-terms corresponding to the set of all minimal sum-to-one subsets.

Example 3: Consider the functions in Example 2 again. A partial OBDD for the expression (*) is shown in Figure 3 where the indices for the $A$-variables are from 1 to 6 (a circle in Figure 3 represents a non-terminal vertex, and sharing among vertices are not shown). On the 1-paths, edges labeled with positive literals are shown in solid bold lines and edges labeled with negative literals are shown in dashed bold lines (labels are not shown). If we write down the positive terms for the 1-paths, we have:

\[
\begin{align*}
\{ & A_1A_6, A_1A_4A_6, A_2A_3A_6, A_2A_3A_5, A_3A_1A_6, \\
& A_3A_4A_6, A_4A_3A_6, A_2A_2A_6, A_4A_5A_6 \}.
\end{align*}
\]

After removing redundant $A$-terms, the set of prime $A$-terms is:

\[
\{ A_2A_6, A_4A_6, A_4A_5, A_3A_4 \}.
\]

Experiments show that Procedure OBDD_STOSs is very efficient even when those $g_i$'s are functions of a large number of variables. However, since the $A$-variables are ordered before the $X$-variables, the first $k$ levels in the OBDD for the summation are close to a complete binary expansion tree as we can see in Figure 3. If there are many functions in $G$, the OBDD will be large and it may become infeasible to enumerate all the $A$-terms from the 1-paths.

3.3 A New Algorithm for Solving the Minimum-Cost STOS Problem

So far, we have discussed two methods for solving the MC-STOS problem:

1. Use OBDD-based tautology-checking procedure.
2. Use OBDD-based sum-to-one subsets procedure.

Method (1) is fast even when there are many candidate functions and method (2) is fast even for a large number of variables. To utilize the advantages of both methods,

```
OBDD_STOSs(G) {
    /\n    G = \{g_1, g_2, ..., g_s\}: functions of X = (x_1, x_2, ..., x_n).
    This procedure derives all minimal STOSs of G.
    /\n    1. create k new variables $A_1, A_2, ..., A_k$;
    2. order $A$-variables before $X$-variables;
    /\n    build the OBDD for summation /
    3. $B = A_1g_1 + A_2g_2 + ... + A_sg_s$;
    /\n    enumerate positive terms of depths \leq k /
    4. STOSs = positive_terms_of_1_path(B, k);
    /\n    irredundant STOSs contains all prime $A$-terms /
    5. STOSs = irredundant(STOSs);
    return STOSs;
}
```

Figure 2. A STOSs algorithm based on OBDDs.

5. Multi-Node Minimization Using Boolean Relations

Boolean relation, as discussed in [4], represents a relation that exists among interacting circuit modules. When the concept of Boolean relation is applied to multiple gates in a multi-level network, it represents a larger degree of freedom than don't-cares. Synthesis methods for Boolean relations based on multi-level representations have been proposed [11], but so far there is no effective way to directly minimize multiple gates using Boolean relations. In this section, we shall address this problem and propose an approach based on the techniques discussed earlier.

Without loss of generality, let $V = \{v_1, v_2, ..., v_t\}$ be a set of gates in $N$. We consider the minimization of gates in $V$ using their Boolean relations.

Definition 2: The smoothing operator, $S_{v_i}f$ is defined as:

\[
S_{v_i}f = f_{v_i} + f_{\overline{v_i}}
\]

where $f_{v_i}$ and $f_{\overline{v_i}}$ are the cofactors of $f$ with respect to variables $v_i$ and $\overline{v_i}$, respectively. Let $x = (x_1, ..., x_n)$,
then

\[ S_0 f = S_{1} f \cdots S_{k} f \]

Definition 3: Let \( z_i = h_i(X) \) \((1 \leq i \leq m)\) be the function realized at the primary output \( z_i \) of network \( N \) with respect to the primary inputs \( X \). The specification of \( N \), denoted by \( S(Z, X) \), is represented by the following characteristic function:

\[ S(Z, X) = \prod_{i=1}^{m} (z_i = h_i(X)) \]

The function realized at \( z_i \) can also be expressed as a function of the primary inputs \( X \) and the intermediate variables corresponding to gates in \( V \), i.e., \( z_i = g(X, V) \). Then, the specification of \( N \) can be written as:

\[ S'(Z, X, V) = \prod_{i=1}^{m} (z_i = g(X, V)) \]

Therefore, the Boolean relation for the gates in \( V \) is:

\[ R(X, V) = S_2 (S(Z, X) S'(Z, X, V)) \]

A realization of the gates in \( V \) is legal if the functions of those gates satisfy the condition \( R(X, V) = 1 \), and our goal is to find a legal realization with a minimum cost.

In [5], specifications in the form of \( R(X, V) = 1 \) were discussed extensively. It has been shown that if for each \( X \in \{0,1\}^n \), \( R(X, V) \) reduces to zero or a single term of the \( V \)-variables, then we will be able to derive the lower and upper bounds for the functions realized by gates in \( V \) independently (such specifications are called tabular specifications, and they represent specifications which can be specified using don't-cares).

In such cases, we can synthesize all the gates in \( V \) based on their bounds simultaneously (for example, using the concurrent resynthesis algorithm discussed in [8]). However, if for some \( X \in \{0,1\}^n \), \( R(X, V) \) reduces to several terms of the \( V \)-variables, then the specification is a Boolean relation, and more powerful methods should be used.

Let \( C = \{ c_1, c_2, \ldots, c_k \} \) be a set of candidate cubes (they could be existing cubes in the network \( N \) or cubes generated by procedures such as OBDD-based STOS algorithm). Then, we can express the realization of the gates in \( V \) using these candidate cubes by a general form:

\[ v_1 = A_{1,1} c_1 + A_{1,2} c_2 + \cdots + A_{1,k} c_k \]
\[ v_2 = A_{2,1} c_1 + A_{2,2} c_2 + \cdots + A_{2,k} c_k \]
\[ \vdots \]
\[ v_l = A_{l,1} c_1 + A_{l,2} c_2 + \cdots + A_{l,k} c_k \]

where \( A_{i,j} \), \( 1 \leq i \leq t \) and \( 1 \leq j \leq k \) are all Boolean variables. By representing these functions of the gates and cubes in terms of primary inputs, we have the following expressions:

\[ f(v_l) = A_{1,1} f(c_1) + A_{1,2} f(c_2) + \cdots + A_{1,k} f(c_k) \]
\[ f(v_2) = A_{2,1} f(c_1) + A_{2,2} f(c_2) + \cdots + A_{2,k} f(c_k) \]
\[ \vdots \]
\[ f(v_l) = A_{l,1} f(c_1) + A_{l,2} f(c_2) + \cdots + A_{l,k} f(c_k) \]

Substituting these expressions into \( R(X, V) \), we obtain an expression in terms of the \( A_{i,j} \)'s and \( X \):

\[ R(X, A_{i,j}) = R(x_1, \ldots, x_n, A_{1,1}, \ldots, A_{1,k}, A_{2,1}, \ldots, A_{2,k}, \ldots, A_{l,1}, \ldots, A_{l,k}) \]

Theorem 5: A prime implicant of the expression \( R(X, A_{i,j}) \) which only consists of literals of the \( A_{i,j} \)'s corresponds to a set of legal realizations of the gates in \( V \) that satisfies the relation \( R(X, V) \). We shall also call these prime implicants prime A-terms, but notice that they may contain both complemented and uncomplemented literals of the \( A_{i,j} \)'s.

Therefore, the problem of minimizing a set of gates in \( V \) using their relation \( R(X, V) \) is reduced to the problem of finding a prime A-term of the expression \( R(X, A_{i,j}) \) whose corresponding set of legal realizations contain a legal realization that has a minimum cost.

Example 4: For the network in Figure 5(a), if we only use network don't-cares to minimize it by RENO or the full simplify in SIS-1.0, we get a network of 3 gates and 6 literals. However, if we choose \( V = \{ v_1, v_2 \} \) and minimize the network using Boolean relations, then we have:

\[ S(Z, X) = x_1 x_2 x_3 + x_1 x_2 z + x_2 x_3 x_4 + x_2 x_3 x_5 z \]
\[ S'(Z, X, V) = x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_4 z + x_1 x_2 x_3 x_4 z' + x_1 x_2 x_3 x_4 z' + x_1 x_2 x_3 x_4 z' + x_1 x_2 x_3 x_4 z' \]

Figure 5(a): The original network. Figure 5(b): Minimized network.
Note that \( R(X, V) \) is not a tabular specification, and it shows the condition that must be satisfied by any legal realization of \( v_1 \) and \( v_2 \). Suppose the candidate cubes 
\[ c_1 = x_1, \quad c_2 = x_2, \quad c_3 = x_1 \oplus x_2, \quad \text{and} \quad c_4 = x_1 \oplus x_2 \]
are derived, then we can express the functions of \( v_1 \) and \( v_2 \) as:
\[
\begin{align*}
 f(v_1) &= A_1, x_1 + A_2, x_2 + A_3, (x_1 + x_2) + A_4, (x_1 \oplus x_2) \\
 f(v_2) &= A_3, x_1 + A_2, x_2 + A_3, (x_1 + x_2) + A_4, (x_1 \oplus x_2)
\end{align*}
\]

Substituting these two expressions into the relation \( R(X, V) \), a relation \( R(x_1, x_2, A_i,j) \), \( 1 \leq i \leq 2 \) and \( 1 \leq j \leq 4 \), is obtained. The prime \( A \)-terms of \( R(x_1, x_2, A_i,j) \) are
\[
\begin{align*}
 A_1, x_1, x_2 &+ A_3, x_1 + A_4, x_1 \oplus x_2, A_2, x_1 \oplus x_2, A_3, x_1 + A_4, x_1 \oplus x_2, A_4, x_1 \oplus x_2 \\
 A_2, x_1, x_2 &+ A_3, x_1 + A_4, x_1 \oplus x_2, A_1, x_1 \oplus x_2, A_3, x_1 + A_4, x_1 \oplus x_2, A_4, x_1 \oplus x_2 \\
 A_3, x_1, x_2 &+ A_2, x_1 \oplus x_2, A_1, x_1, x_2 + A_4, x_1 \oplus x_2, A_4, x_1 \oplus x_2, A_2, x_1 \oplus x_2 \\
 A_4, x_1, x_2 &+ A_2, x_1 \oplus x_2, A_3, x_1 + A_4, x_1 \oplus x_2, A_1, x_1 \oplus x_2, A_3, x_1 + A_4, x_1 \oplus x_2
\end{align*}
\]

Each of the prime \( A \)-term corresponds to a set of legal realizations of \( v_1 \) and \( v_2 \). For example, the prime \( A \)-term
\[
A_1, x_1, x_2 + A_4, x_1 \oplus x_2
\]

specifies that if \( v_1 \) only contains \( c_1 \) (since \( A_1, x_1, x_2 \)) and \( c_4 \) should be 1 and as long as \( v_2 \) contains \( c_1 \) (since \( A_4, x_1 \oplus x_2 \)) should be 1, then such realizations will be legal, no matter whether \( v_2 \) contains \( c_1, c_2 \), \( c_3 \), or not.

In the case of area minimization where usually the number of literals is minimized, the cost of a legal realization is the total costs of the cubes being used. Therefore, although each prime \( A \)-term corresponds to a set of legal realizations, each variable \( A_i,j \) can be assigned the cube (i.e., \( c_j \)), and the cost of each prime \( A \)-term can be defined as the sum of the costs of the \( A_i,j \) variables which are 1's. Then, our goal is to find a prime \( A \)-term which has the minimum cost.

Example 5: Consider the network shown in Figure 5(a), where the relation \( R(x_1, x_2, A_i,j) \), \( 1 \leq i \leq 2 \) and \( 1 \leq j \leq 4 \), has been derived in Example 4. Because \( R(x_1, x_2, A_i,j) = 1 \) if and only if \( R(0, 0, A_i,j) = 1 \), \( R(0, 1, A_i,j) = 1 \), \( R(1, 0, A_i,j) = 1 \), and \( R(1, 1, A_i,j) = 1 \) are all true, the following set of clauses can be obtained by tautology checking on \( x_1 \) and \( x_2 \):
\[
\begin{align*}
(A_1, x_1, x_2, A_2, x_1 \oplus x_2, A_3, x_1 + A_4, x_1 \oplus x_2) \\
(A_2, x_1, x_2, A_1, x_1 \oplus x_2, A_3, x_1 + A_4, x_1 \oplus x_2) \\
(A_3, x_1, x_2, A_1, x_1, x_2 + A_4, x_1 \oplus x_2) \\
(A_4, x_1, x_2, A_2, x_1 \oplus x_2, A_3, x_1 + A_4, x_1 \oplus x_2)
\end{align*}
\]

These clauses form a binate-covering problem and they can be solved by existing procedures. A satisfying assignment which has a minimum cost is \( A_1, x_1 = A_2, x_2 = 1 \) and all other \( A_i,j \)'s are 0's. It corresponds to the legal realization \( v_1 = x_1 \) and \( v_2 = x_2 \), shown in Figure 5(b). If we check the results of Example 4, this minimum-cost legal realization is in the set of legal realizations associated with the prime \( A \)-term
\[
A_1, x_1, x_2 + A_2, x_1 \oplus x_2 + A_3, x_1 + A_4, x_1 \oplus x_2
\]

Another minimum-cost legal realization can be obtained by assigning 1 to \( A_1, x_1 \) and \( A_2, x_2 \) and 0 to all other \( A_i,j \)'s. This solution is in the set of legal realizations associated with the prime \( A \)-term
\[
A_1, x_1, x_2 + A_2, x_1 \oplus x_2 + A_3, x_1 + A_4, x_1 \oplus x_2
\]

6. Experimental Results

The techniques discussed in Sections 2-4 have been implemented in the RENO algorithm and the following experiments were conducted:

(1) Comparison with the original RENO algorithm: some results are shown in Table 1. The initial networks are synthesized by SIS-1.0 using script.rugged and then mapped to MCNC standard-cell library using area-mode mapping command map -m 0. Then, the original RENO and improved RENO algorithms are applied to reduce the number of literals. When resynthesizing a gate, its maximum set of permissible functions [12] is used if it can be derived within a time limit. Otherwise, the original function realized by the gate is used. The area-minimized networks are then mapped to library again using map -m 0. As can be seen, due to the use of OBDD-based STOS algorithm, the processing speed of RENO has been greatly improved while achieving comparable or better minimization quality. In cases such as C880, for which the old RENO can finish without being stopped at time limit, about 20 times speedup has been obtained. The initial networks generated by SIS-1.0 using script.rugged and area-mode mapping command are usually high-quality networks of minimal area. Yet on average RENO can still obtain 8.1% area reduction. In cases like r481, significant area reduction of 85% has been obtained. This shows that in order to obtain the best results, the RENO algorithm should be used even when technology-independent minimizations have been done using algorithms such as full simplify.

(2) Application of RENO for delay minimization: the results are shown in Table 2. The initial networks are synthesized by SIS-1.0 using script.rugged and then mapped to the MCNC standard-cell library using the speed-mode mapping command map -m 1. Then, RENO is applied to reduce the maximum depth of the mapped networks, and those level-reduced networks are mapped to library again using map -m 1. On the average, delay of the networks is reduced by 18% and area is also reduced by 2%. In the best case (e.g., act.), 63% delay reduction has been obtained. The results of applying speed_up in SIS-1.0 are also reported in Table 2. In this experiment, the initial networks are processed by speed_up - m mapped, and then the mapping command map - m 1 is applied. On the average, speed_up reduces the delay of the networks by 10.4% but area is increased by 13.4% (however, speed_up is in general more efficient).

7. Conclusions

We discussed techniques for solving the sum-to-one subsets problem using properties of OBDDs and unate functions. By combining these techniques with tautology-checking, an efficient algorithm for solving the MC-STOS problem has been developed, and it greatly improved the processing speed of the RENO algorithm. The ability to generate many candidate cubes efficiently has also enhanced RENO's ability for area and delay minimization. An algorithm for area and delay minimization using Boolean relations is also proposed. We expect that the techniques discussed in this paper will be useful in solving other minimization problems.

References


Table 1: Comparison of area minimization by old and new RENO algorithms

<table>
<thead>
<tr>
<th>Function</th>
<th>SIS-1.0</th>
<th>Old RENO</th>
<th>New RENO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area</td>
<td>Time</td>
<td>Area</td>
</tr>
<tr>
<td>9symm</td>
<td>2952</td>
<td>-</td>
<td>2976</td>
</tr>
<tr>
<td>C1355</td>
<td>6960</td>
<td>timeout</td>
<td>-</td>
</tr>
<tr>
<td>C1908</td>
<td>7632</td>
<td>7992</td>
<td>7162.3</td>
</tr>
<tr>
<td>C432</td>
<td>3064</td>
<td>3024</td>
<td>2020.6</td>
</tr>
<tr>
<td>C5315</td>
<td>24616</td>
<td>timeout</td>
<td>-</td>
</tr>
<tr>
<td>C800</td>
<td>6256</td>
<td>6184</td>
<td>15944.5</td>
</tr>
<tr>
<td>alud</td>
<td>9298</td>
<td>9236</td>
<td>1754.6</td>
</tr>
<tr>
<td>apex1</td>
<td>18272</td>
<td>memory</td>
<td>-</td>
</tr>
<tr>
<td>apex6</td>
<td>11816</td>
<td>11600</td>
<td>1814.4</td>
</tr>
<tr>
<td>apex7</td>
<td>3720</td>
<td>3664</td>
<td>1154.5</td>
</tr>
<tr>
<td>fl1</td>
<td>1448</td>
<td>1042</td>
<td>3.3</td>
</tr>
<tr>
<td>fr2</td>
<td>2238</td>
<td>2144</td>
<td>1001.1</td>
</tr>
<tr>
<td>ll1</td>
<td>13112</td>
<td>12926</td>
<td>2601.8</td>
</tr>
<tr>
<td>pcel</td>
<td>1152</td>
<td>1136</td>
<td>6.8</td>
</tr>
<tr>
<td>rot</td>
<td>10872</td>
<td>timeout</td>
<td>-</td>
</tr>
<tr>
<td>set</td>
<td>1296</td>
<td>1272</td>
<td>4.5</td>
</tr>
<tr>
<td>tr3</td>
<td>11250</td>
<td>10320</td>
<td>212.9</td>
</tr>
<tr>
<td>term1</td>
<td>2656</td>
<td>2592</td>
<td>62.1</td>
</tr>
<tr>
<td>z4ml</td>
<td>680</td>
<td>520</td>
<td>1.1</td>
</tr>
</tbody>
</table>

Table 2: Comparison of delay minimization by speed_up and RENO

<table>
<thead>
<tr>
<th>Functions</th>
<th>Initial</th>
<th>speed_up</th>
<th>New RENO</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Area</td>
<td>Delay</td>
<td>Area</td>
</tr>
<tr>
<td>9symm</td>
<td>3048</td>
<td>18.4</td>
<td>3824</td>
</tr>
<tr>
<td>C1355</td>
<td>7600</td>
<td>20.0</td>
<td>6300</td>
</tr>
<tr>
<td>C1908</td>
<td>7984</td>
<td>35.0</td>
<td>10272</td>
</tr>
<tr>
<td>C432</td>
<td>3224</td>
<td>34.4</td>
<td>3632</td>
</tr>
<tr>
<td>C5315</td>
<td>24912</td>
<td>34.0</td>
<td>29304</td>
</tr>
<tr>
<td>C800</td>
<td>6568</td>
<td>41.8</td>
<td>10576</td>
</tr>
<tr>
<td>alud</td>
<td>6266</td>
<td>34.2</td>
<td>7488</td>
</tr>
<tr>
<td>apex1</td>
<td>18744</td>
<td>29.8</td>
<td>19928</td>
</tr>
<tr>
<td>apex6</td>
<td>12128</td>
<td>24.6</td>
<td>14242</td>
</tr>
<tr>
<td>apex7</td>
<td>3864</td>
<td>19.0</td>
<td>4520</td>
</tr>
<tr>
<td>fl1</td>
<td>1512</td>
<td>29.4</td>
<td>2760</td>
</tr>
<tr>
<td>fr2</td>
<td>2408</td>
<td>11.8</td>
<td>2768</td>
</tr>
<tr>
<td>fr3</td>
<td>13736</td>
<td>31.2</td>
<td>14624</td>
</tr>
<tr>
<td>ll1</td>
<td>1472</td>
<td>21.0</td>
<td>2392</td>
</tr>
<tr>
<td>pcel</td>
<td>1224</td>
<td>11.6</td>
<td>1488</td>
</tr>
<tr>
<td>rot</td>
<td>11416</td>
<td>26.6</td>
<td>15564</td>
</tr>
<tr>
<td>set</td>
<td>1432</td>
<td>26.4</td>
<td>2784</td>
</tr>
<tr>
<td>t8f1</td>
<td>12192</td>
<td>21.6</td>
<td>11736</td>
</tr>
<tr>
<td>term1</td>
<td>2792</td>
<td>13.4</td>
<td>3632</td>
</tr>
<tr>
<td>z4ml</td>
<td>736</td>
<td>13.0</td>
<td>1016</td>
</tr>
</tbody>
</table>