Optimal Common Sub-Expression Elimination Algorithm of Multiple Constant Multiplications with a Logic Depth Constraint

Yuen-Hong Alvin HO, Member, Chi-Un LEI, Student Member, Hing-Kit KWAN†, and Ngai WONG, Nonmembers

SUMMARY In the context of multiple constant multiplication (MCM) design, we propose a novel common sub-expression elimination (CSE) algorithm that models the optimal synthesis of coefficients into a 0-1 mixed-integer linear programming (MILP) problem with a user-defined generic logic depth constraint. We also propose an efficient solution space, which combines all minimal signed digit (MSD) representations and the shifted sum (difference) of coefficients. In the examples we demonstrate, the combination of the proposed algorithm and solution space gives a better solution comparing to existing algorithms.

key words: common sub-expression sharing, multiple constant multiplications, mixed-integer linear programming

1. Introduction

Multiple constant multiplication (MCM) constitutes a typical fixed-point arithmetic operation in digital signal processing. It is the focus of various research on high-speed and low-power devices in communication systems [1] and digital signal processing (DSP) systems. Such multiplierless MCMs are utilized, for example, in the design of finite-impulse-response (FIR) filters [2]. In multiplierless MCM, multipliers are replaced by simpler components such as adders and hard-wired shifts (“adders” in our paper include also subtractors as their hardware costs are similar). By using the negative digits (subtractors in circuits) in their signed-digit representations, coefficients may be synthesized with fewer adders, therefore the area and power consumption of the circuit can be reduced. An example of a multiplier-based and a multiplierless-based MCM implementations are shown in Figs. 1 and 2, respectively, wherein 4 multiplications can be replaced by 6 adders and 6 hard-wired shifts after resource sharing.

Given a set of integer coefficients, the common sub-expression elimination (CSE) (e.g., [2]–[9]) is further used for minimizing the number of adders in a MCM block. The CSE algorithms search for common sub-expressions from different signed-digit representations, and re-use the searched common sub-expressions for later synthesis to reduce the total number of additions.

For example, when searching for the synthesis of 673, the binary representation of 673 may be decomposed as $1 \times 2^3 + 161 \left( \begin{array}{l} 1 \end{array} \right)_{10}01000001 = 1 \ll 9 + 10100001$, $5 \times 2^7 + 33 \left( \begin{array}{l} 1 \end{array} \right)_{10}0100001 = 101 \ll 7 + 100001$, or $17 \times 2^5 + 129 \left( \begin{array}{l} 1 \end{array} \right)_{10}0100001 = 10001 \ll 5 + 1000001$. The digits surrounded by boxes are the portion of sub-expression obtained by hard-wired shifts. One of such decompositions will then be chosen at the end of the algorithm. Also, 673 can be used as a sub-expression for other coefficients.

Besides CSE, the graph-based method [10], [11] works on another approach, which is to investigate the relationship between coefficients (not sub-expressions) directly. Therefore, they are not restricted to explicit sub-expression patterns and may give better solutions. However, their search space for high-order filters will become extremely large and is usually solved by heuristic algorithms, therefore optimal solution is not guaranteed. Furthermore, their solutions generally have high logic depth, which is impractical in implementations [2].

In general, a larger search space allows a better chance of finding more common digit patterns between coefficients, and may lead to a better solution otherwise unobtainable in a smaller sub-space. However, as a nature of NP-hard problems [12], the complexity grows exponentially as the size of the search space increases. Analytical lower bound of the solution is given [13], but most CSE algorithms (e.g., [2]–[7]) are heuristic in nature, providing no indication on how

Fig. 1 A multiplier-based MCM example.

Fig. 2 A multiplierless-based MCM example. (≪: binary shift)
far from the optimum their solution is [6]. They are also limited to particular search spaces (signed-digit representations) such as binary [6], canonical signed digit (CSD) [3], [6], minimal signed digit ( MSD) [4],[6], and all signed digit [5]. Only the technique in [6] guarantees the optimal solution.

Specifically, Flores et al. [6] model the CSE problem as a Boolean network that covers all possible syntheses of coefficients which can be used in the MCM. The network is then converted into a 0-1 pure integer linear programming (PILP) problem which is solved with a satisfiability (SAT)-based solver. The model of [6], however, requires as many binary variables as the total number of gates in the circuit plus the primary inputs. This leads to a long computation time for the optimal solutions. Moreover, [6] compares the search spaces of binary, CSD, and MSD representations only. And it does not include a logic depth constraint to ensure the circuit speed. The maximum logic depth constraint in turn provides trade-off between circuit speed and area, which is important for high-speed DSP system design.

Many solvers use branch-and-bound (BB) algorithms to solve PILP problems (e.g., [14]). In short, a BB algorithm first relaxes a PILP problem into a linear programming (LP) problem: if \( C_n \) is a binary variable, it is relaxed to \( 0 \leq C_n \leq 1 \) and then solved with LP. If the \( C_n \) thus solved is fractional, the BB algorithm subdivides \( C_n \) into two sub-problems with \( C_n = 0 \) and \( C_n = 1 \). This process goes on iteratively until all integrality requirements are met. Optimality of the solution is guaranteed by combining the BB framework as well as the simplex algorithm in each node of the branching tree. As the number of integer variables is increased, the solution time of PILP may increase exponentially.

In this paper and its preliminary version [15], the minimization of the number of required adders to synthesize a MCM block with logic depth constraints as a \( 0 - 1 \) mixed-integer linear programming (MILP) problem. (In a MILP problem, only certain variables are integers while the rest are allowed to be continuous.) Compared to a PILP model, the problem contains fewer (binary and total) variables and constraints, therefore it can be solved more efficiently, e.g., using a MILP solver [14].

We propose a new search space that combines all MSD representations and the shifted sum (difference) of output coefficients. Each coefficient is decomposed according to a set of constraints in a defined search space and stored into a lookup table. Synthesis of a coefficient can then be retrieved instantly. The proposed search space is compared against the well known search space of MSD representations using our global optimization approach with logic depth constraint.

The paper is organized as follows. In Sect. 2, different feasible solution spaces are explained. Section 3 presents the formulation of the 0-1 MILP problem. Section 4 compares the results of various search spaces and contrasts them against some representative results in [6]. Section 5 draws the conclusion.

2. Feasible Solution Spaces

To begin with, some notational conventions are specified. “\( n \)” is the integer value of the coefficient. “\( \# \text{nzbit} \)” denotes the number of non-zero bits in the CSD or MSD representations. “\( Oset \)” is the set of coefficients that connect to the output of the MCM block. “\( LT_n \)” is the lookup table of the coefficient n. “\( L_n \)” is the total number of rows in \( LT_n \). “\( Cset \)” is the set of coefficients that may be used to synthesize coefficients in \( Oset \). “\( D_{\text{max}} \)” is a constant representing the maximum logic depth.

A generic sub-expression sharing example is shown in Fig. 3, instead of using 4 adders for coefficients 259 and 447, 1 adder can be saved if sub-expression 65 is shared between both coefficients. An arbitrary positive integer coefficient n can be decomposed as:

\[
n = \pm S_1 \times 2^p \pm S_2 \times 2^q,
\]

where \( n, S_1 \), and \( S_2 \) are positive and odd integers, \( S_1 \neq n \) and \( S_2 \neq n \). \( S_1 \) and \( S_2 \) are sub-expressions of n; \( p \) and \( q \) are the numbers of left shifted bits of \( S_1 \) and \( S_2 \), respectively. Since there are infinite decompositions for a coefficient satisfying (1) (e.g., \( 3 \) may be cast as \( 2^2 - 1, 2^{10} - 1021, 2^{20} - 1048573, \cdots \)), the feasible solution space is generally fixed at a finite set in the CSE algorithms. Traditionally, search space of binary representation, namely, 0 and 1, are used. CSE algorithm based on CSD with a signed digital representation \((-1, 0 \text{ and } 1\) is proposed later [16]. The CSD is unique and has two properties: 1) The number of non-zero digits (i.e., \(-1 \text{ and } 1\) is minimal; 2) Two non-zero digits are not adjacent. CSE algorithm based on MSD is proposed in [4], which neglects the second property. Therefore, it allows more possible combinations for a coefficient and gives a further reduction in sharing sub-expressions with other coefficients. An example of the binary, the CSD, and the MSD representations of 23 are shown below (where \( \underline{1} \) represents the \(-1 \) digit):

1. Binary: 101111 (\#nzbit = 4);
2. CSD: 101001 (\#nzbit = 3);
3. MSD: 1010011 (\#nzbit = 3), 1100111 (\#nzbit = 3).

The following sub-sections describe two search spaces with different mathematical constraints imposed on (1) for optimal synthesis. The MCM coefficients of \( Oset = \{673, 383, 449, 33\} \) are used for illustration.
2.1 Search Space of MSD Representations

Search spaces from the MSD representations are generated based on all digit patterns in the MSD representations described in [6]. Table 1 shows the lookup table for the coefficients in Oset. Decompositions from rows 1 to 7, 1 to 5, and 1 to 3 in the third, fifth, and seventh columns show the search spaces of MSD representations in LT\textsubscript{675}, LT\textsubscript{383}, and LT\textsubscript{449}, respectively. These “Search Space” columns describe different digit patterns extracted from the MSD representations. Since the #nzbit of 33 is two, it can be synthesized with one adder with the logic depth of 1. It is considered to be the best synthesis of the coefficient and therefore LT\textsubscript{33} is fixed at one row without any further expansion of the search space.

2.2 Expanded Search Space Based on Shifted Sum or Difference of Coefficients (SSD)

An expanded solution space of the MSD representations, called the shifted sum or difference of output coefficients (SSD), is proposed to exploit new relationship between sub-expressions and output coefficients. The following constraints are applied to (1) in order to limit the search space to a finite set:

1. $S_2$ must be a coefficient in Oset;
2. #nzbit of $n$ - #nzbit of $S_1$ ≥ 1;
3. #nzbit of $n$ - #nzbit of $S_2$ ≥ 1.

Since (1) restricts $n$, $S_1$, and $S_2$ to be odd, one of $S_1 \times 2^p$ or $S_2 \times 2^q$ must be odd and the other even. Subsequently, (1) can be re-expressed as

$$S_1 = (n + S_2) \times 2^p; \text{ or }$$
$$S_1 = (n - S_2) \times 2^p; \text{ or }$$
$$S_1 = (n + S_2) \times 2^q; \text{ or }$$
$$S_1 = n + S_2 \times 2^q; \text{ or }$$
$$S_1 = n - S_2 \times 2^q; \text{ or }$$
$$S_1 = -n + S_2 \times 2^q.$$ (2)

Since both $n$ and $S_2$ are in Oset, decompositions that satisfy (1) are determined based on the shifted sum (difference) of coefficients ($n$ and $S_2$). Due to the additional constraints imposed on (1), the number of feasible solutions in (2) is finite. Linear search is then used to search for all feasible decompositions from (2).

The search space is expanded on top of the search space of MSD representations. As a result, the search space of the MSD representations is a subset of our proposed SSD search space.

Coefficients in Oset are first copied to Cset. A set of decomposition tables using one of the search spaces described in Sect. 2 is computed (e.g., Table 1). Sub-expressions in the lookup tables are copied into Cset if they are not already there. If a lookup table LT\textsubscript{n} has not been created for the coefficient $n$, then $n$ is decomposed using the MSD representations. This process continues recursively until all coefficients have been copied into Cset and LT\textsubscript{n} is created for all $n \in Cset$. Considering the search space of SSD in Sect. 2.2, when Oset = [673, 383, 449, 33], the resultant Cset is {3, 5, 7, 17, 21, 31, 33, 63, 127, 129, 145, 161, 225, 255, 257, 383, 449, 511, 513, 554, 641, 673}. LT\textsubscript{3}, LT\textsubscript{5}, LT\textsubscript{7}, \cdots LT\textsubscript{673} are formed for all $n \in Cset$.

Note that decompositions with the same pair of sub-expressions are inserted into the lookup table once only. This is because the adder cost with the same pair of coefficients is the same. Decompositions from rows 1 to 12, 1 to 5, and 1 to 3 in Table 1 show the proposed search space for coefficients 673, 383, and 449, respectively. The “Search Space” columns show different digit patterns extracted from the MSD representations as well as the value of $S_2$ when evaluating the corresponding decomposition.

3. Problem Formulation

The multiplierless synthesis problem is formulated into a 0-1 MILP formulation with a minimum adder cost objective function and a group of constraints showing the relationship between sub-expressions. The synthesis problem is then extended into a 0-1 MILP formulation with an additional group of constraints showing the logic depth (essentially a delay time) constraint.
3.1 Minimal Adder Cost Objective Function

To minimize the adder cost (therefore, area and power), a binary variable \( C_n \) is defined for each coefficient \( n \in Cset \). As a result, the total number of binary variables equals the number of coefficients in \( Cset \). \( C_n = 1 \) indicates the coefficient \( n \) is synthesized. Otherwise, \( C_n = 0 \) and the coefficient \( n \) would not be synthesized in the MCM block. Since an adder is needed for every synthesized coefficients, the objective function minimizing the number of adders for the proposed 0-1 MILP model reads:

\[
\min \sum_{n \in Cset} C_n \quad \text{s.t.} \begin{cases} C_n \in \{0, 1\} & \forall n \in Cset \\ C_m = 1 & \forall m \in Oset \end{cases} . \tag{3}
\]

Here \( C_m = 1, \forall m \in Oset \) forces the 0-1 MILP solver to synthesize all coefficients in \( Oset \) for the output coefficients.

3.2 Decomposition Expression Constraints

When the \( \#nzbit = 2 \) for a coefficient \( n \) (e.g., coefficients 3, 5, 7, 17, 31, 33, 63, 127, 129, 255, 257, 511, 513), it can be synthesized with one adder directly from the summation of two different shifted versions of MCM block input. When the \( \#nzbit > 2 \) for a coefficient \( n \), the coefficient must be synthesized from one of the decompositions in \( LT_n \). These decomposition expressions have to be included as constraints in the formulation. For instance, (4) shows the constraints for synthesizing the coefficient 673 in the SSD search space:

\[
C_{673} = C_{161} + C_{545} + C_{641} + C_{21} + \min \{C_3, C_{33}\} + \min \{C_{17}, C_{129}\} + \min \{C_{53}, C_{513}\} + \min \{C_{145}, C_{383}\} + \min \{C_{333}, C_{383}\} + \min \{C_{7}, C_{449}\} + \min \{C_{225}, C_{449}\} + \min \{C_{33}, C_{145}\} . \tag{4}
\]

There are 12 terms in (4) and each corresponds to a possible decomposition in \( LT_{673} \). For example, the first term “\( C_{161} \)” indicates the coefficient 673 may be synthesized with one adder if the coefficient 161 exists in the MCM block (as shown in \( LT_{673} \) row 1). Similarly, the second term “\( C_{545} \)” indicates the coefficient 673 may be synthesized with one adder if the coefficient 45 exists in the MCM block (as shown in \( LT_{673} \) row 2). The fifth term in (4) is a minimization term: \( \min \{C_3, C_{33}\} = 1 \) if both \( C_3 \) and \( C_{33} \) exist; otherwise, \( \min \{C_3, C_{33}\} = 0 \). It indicates that the coefficient 673 may be synthesized with one adder if both of the coefficients 5 and 33 exist in the MCM block (as shown in \( LT_{673} \) row 5). We note that each of the minimization terms, \( \min \{C_a, C_b\} \), is defined as a continuous variable for efficient computation. When both \( C_a \) and \( C_b \) are 1, \( \min \{C_a, C_b\} \) will be enforced to 1. Otherwise, \( \min \{C_a, C_b\} = 0 \). This relation is enforced by four linear inequality constraints:

\[
\begin{align*}
\min \{C_a, C_b\} & \leq C_a, \min \{C_a, C_b\} & \leq C_b, \min \{C_a, C_b\} & \geq 0 \\
\min \{C_a, C_b\} & \geq C_a + C_b - 1 . \end{align*}
\]

The involvement of continuous variables converts the problem from a 0-1 PILP to a 0-1 MILP. Similar relationships apply to the 6th to 12th term in (4) from all different decompositions in \( LT_{673} \). Constraints similar to (4) are required for all lookup tables \( LT_n \) where \( L_n > 1 \) (e.g., coefficients 21, 145, 161, 225, 383, 449, 545, 641, 673 in \( Cset \)).

Comparing to 0-1 PILP model in [6], our proposed algorithm will identify coefficients which can be synthesized with one adder only directly, that is, the situation when \( S_1 = 1 \) and \( S_2 = 1 \) in (1) before minimization, which avoids unnecessary computations. Furthermore, our 0-1 MILP model formulates differently from 0-1 PILP model, which has a simpler formulation in synthesis for coefficients with sub-expression of one non-zero bit only; that is, the situation when \( S_1 = 1 \) or \( S_2 = 1 \) in (1) (e.g., \( C_{161} \) for \( C_{673} \) in (4)). This results in fewer variables in the computations.

3.3 Maximum Logic Depth (\( D_{\text{max}} \)) Constraints

Both area and delay are considered in the high-speed circuit design. Delay is related to the circuit implementation, so a generic definition, logic depth, is used. In MCM design, same coefficients can be synthesized with different logic depths (shown in Fig. 4, \( D \) represents \( \frac{1}{2} \)), so an optional parameter \( D_{\text{max}} \) is proposed to restrict the maximum logic depth of the MCM block as a delay constraint. If \( D_{\text{max}} \) is absent, the 0-1 MILP solver minimizes the problem without the logic depth restriction. If \( D_{\text{max}} \) is provided, constraints described in this section are applied on top of the 0-1 MILP problem in Sects. 3.1 and 3.2.

To form the set of constraints, a continuous variable \( D_n \) is defined for each coefficient \( n \in Cset \). \( D_n \) is the logic depth of \( n \) if \( n \) is synthesized in the MCM block, i.e., when \( C_n = 1 \). Otherwise, when \( C_n = 0 \), \( D_n \) is set to the default value of \( D_{\text{max}} \) to disable the constraints. This is implemented by the two constraints below. When \( D_{\text{max}} \geq 2 \) and \#nzbit of \( n = 2 \), the following constraint is applied:

\[
D_n = D_{\text{max}}(1 - C_n) + C_n . \tag{5}
\]

If \( C_n = 1 \), (5) forces \( D_n \) to 1. Otherwise, \( D_n = D_{\text{max}} \).

When \( D_{\text{max}} \geq 2 \) and \#nzbit of \( n > 2 \), the following constraints are applied:

\[
D_{\text{max}}(1 - C_n) + 2C_n \leq D_n \leq D_{\text{max}} . \tag{6}
\]

If \( C_n = 1 \), (6) renders \( 2 \leq D_n \leq D_{\text{max}} \). Otherwise, \( D_n = D_{\text{max}} \).

Fig. 4 Example of synthesizing a coefficient 201: (a) Logic depth = 3. (b) Logic depth = 2.
Assuming the pair of sub-expressions $S_1$ and $S_2$ are used to synthesize the coefficient $n$, $D_n$ must be equal to $\text{Max}[D_{5}, D_{3}] + 1$ because an adder is used to synthesize $n$ at the output of $S_1$ and $S_2$. In other words, since $n$ is synthesized from the decomposition in $\text{LT}_n$, the coefficients that are used to synthesize $n$ must have their logic depths less than $D_n$. For coefficient 673, the following shows the required constraints produced from $\text{LT}_{673}$:

$$C_{673} \leq (D_{\text{max}} - D_{161}) + (D_{\text{max}} - D_{545})$$
$$+ (D_{\text{max}} - D_{641}) + (D_{\text{max}} - D_{21})$$
$$+ (D_{\text{max}} - \text{Max}[D_{5}, D_{33}]) + (D_{\text{max}} - \text{Max}[D_{17}, D_{129}])$$
$$+ (D_{\text{max}} - \text{Max}[D_{5}, D_{33}]) + (D_{\text{max}} - \text{Max}[D_{33}, D_{45}])$$
$$+ (D_{\text{max}} - \text{Max}[D_{145}, D_{383}]) + (D_{\text{max}} - \text{Max}[D_{35}, D_{383}])$$
$$+ (D_{\text{max}} - \text{Max}[D_{7}, D_{44}]) + (D_{\text{max}} - \text{Max}[D_{225}, D_{449}]).$$

(7)

There are 12 bracketed terms in (7) for the 12 possible decompositions in Table 1. If $C_{673}$ is 1, at least one of the 12 terms has to be greater than or equal to 1. For example, if the first term $(D_{\text{max}} - D_{161})$ is greater or equal to 1, it indicates the coefficient 673 may be synthesized with the logic depth of $D_{161} + 1$ (from $\text{LT}_{673}$ row 1) and $D_{\text{max}} + 1 \leq D_{\text{max}}$. The fifth term in (7) is a maximization term, $\text{Max}[D_{5}, D_{33}]$. It indicates that the coefficient 673 may be synthesized with the logic depth of $\text{Max}[D_{5}, D_{33}] + 1$ (from row 5 in $\text{LT}_{673}$) and $\text{Max}[D_{5}, D_{33}] + 1 \leq D_{\text{max}}$. Similar relationships apply to (7) from each decomposition in $\text{LT}_{673}$. Constraints similar to (7) are required for all lookup tables $\text{LT}_n$ of which there are more than one possible decomposition in it (e.g., coefficients 21, 145, 161, 225, 383, 449, 545, 641, 673 in $\text{Cset}$).

Minimum logic depth is also desired on the top of the minimum adder cost, when $D_{\text{max}} = 3$ and #nzbit of $n > 2$, the following shows the required constraints for $\text{LT}_{673}$:

$$D_{673} \leq D_{\text{max}}(2 - C_{673}) - \text{Min}[C_{5}, C_{33}],$$

(8)

$$D_{673} \leq D_{\text{max}}(2 - C_{673}) - \text{Min}[C_{17}, C_{129}],$$

(9)

$$D_{673} \leq D_{\text{max}}(2 - C_{673}) - \text{Min}[C_{5}, C_{513}],$$

(10)

$$D_{673} \geq D_{\text{max}}(1 - \text{Min}[C_{5}, C_{33}] - \text{Min}[C_{17}, C_{129}])$$
$$- \text{Min}[C_{5}, C_{513}].$$

(11)

Equations (8) to (10) are produced from all possible decompositions in $\text{LT}_{673}$ that may result in $D_{673} = 2$. Note that the #nzbit of (5, 17, 33, 129, 513) is 2. Equations (8) to (11) force $D_{673} = 2$ if the coefficient 673 can be synthesized from coefficients that have their logic depth equal to 1 (combination of 5 and 33, 17 and 129, 5 and 513). Otherwise, (6) and (11) force $D_n = D_{\text{max}}$. Constraints similar to (6), (8), (9) and (10) are required for all coefficients that have their #nzbit greater than 2 (e.g., coefficients 21, 145, 161, 225, 383, 449, 545, 641, 673 in $\text{Cset}$).

Finally, the relationship between sub-expressions and the logic depth constraints are converted into variables and constraints of a 0-1 MILP problem, which is then solved by a generic MILP solver. In our proposed optimization problem, the binary variables represent all feasible sub-expression terms, which depends on the selected search space in Sect. 2; The continuous variables specify the minimization terms in Sect. 3.2 and the delay for each sub-expression in Sect. 3.3; The constraints specify the output coefficients, the decomposition relationship in Sect. 3.2 and the maximum logic depth constraints in Sect. 3.3.

### Table 2: Upper bounds on the problem size with different search spaces.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>68</td>
<td>50</td>
<td>592</td>
<td>94</td>
<td>792</td>
<td>592</td>
<td>94</td>
<td>1678</td>
<td>9655</td>
</tr>
<tr>
<td>8</td>
<td>138</td>
<td>346</td>
<td>2784</td>
<td>276</td>
<td>3533</td>
<td>30550</td>
<td>276</td>
<td>7342</td>
<td>48057</td>
</tr>
<tr>
<td>9</td>
<td>276</td>
<td>867</td>
<td>8576</td>
<td>470</td>
<td>17052</td>
<td>135638</td>
<td>470</td>
<td>34574</td>
<td>207051</td>
</tr>
</tbody>
</table>

3.4 Problem Complexity Analysis

In general, for a standard 0-1 PILP (and 0-1 MILP), its computation complexity has an exponential time complexity with respect to the number of variables, which is the relationship between sub-expressions in the minimization problem. The sub-expressions relationships depend generally on the wordlength, the used search space and the number of coefficients, but also varies with examples since the implicit pattern relationship between sub-expressions is not known in advance. Table 2 shows the maximum complexity of the problem with MSD, SSD and SSD with logic depth constraint ($D_{\text{max}} = 3$) under different wordlengths, which happens when all output coefficients are synthesized. To begin the analysis, some notational conventions are specified. In Table 2, “#bvar” and “#cvar” are the numbers of binary and continuous variables, respectively, and “#const.” denotes the number of constraints. SSD search space requires more constraints than MSD search space because SSD explores more sub-expression relationship than MSD. There is an affordable upper bound on the problem size for practical examples and can be solved by MILP solver, shown by the numerical examples in Sect. 4. We have also included a study involving 100 example filters of random coefficients in in Sect. 4 describing the computation speed with respect to different wordlengths, while computation speed is measured roughly by CPU runtimes.

### 4. Results

We first demonstrate our result using a simple MCM synthesis. For simplicity, the search spaces are pre-generated and stored in the hard disk for coefficients up to 13-bits (i.e., 8192). The size of our saved data measures about 10 MB.
in Matlab 7. Figure 2 shows the synthesis of the required Oset = \{33, 383, 449, 673\} using the SSD search space and the proposed algorithm. Six adders are needed in the synthesized design. Note that the coefficient 673 is synthesized as 673 = 33 \times 2^5 - 383 (row 9 in Lt_{673}), which is not in the search space of the MSD representations. If the search space of the MSD representations is used, the MCM block requires seven adders. Consequently, 14% adder reduction is achieved by the SSD search space.

More fixed-point filter examples are used to compare the results arising from different search spaces. Our experiments are run on a 1.5 GHz computer with 768 MB memory. Some notational conventions are specified: In Table 3, "\#tap" stands for the filter length or the number of multiplications of the MCM block and "width" is the wordlength of coefficients. The subscripts "MSD" and "SSD" indicate the variables. The subscripts "MSD" and "SSD" indicate the specifications and synthesized results of Filter 1 to Filter 8 obtained from [6] and summarized in Table 3.

The complexity performance of the proposed optimization framework is also investigated and compared to [6] in Table 4. The 0-1 PILP models of [6] are constructed and solved with [14] in the same experimental settings. The total solution time of synthesizing all eight filters is 34.66 seconds in our scheme while it takes 246.90 seconds for that in [6], or in other words, a 5.1X average speedup. Instead of modeling the MCM problems into a Boolean network in [6], the problems are directly formulated as 0-1 MILP problems in our proposed method. As a result, the redundancy variables in [6] are greatly reduced. All of the filter examples show that the proposed models have fewer variables and constraints than those in [6]: on average, our proposed models have 82% fewer variables, and 68% fewer constraints. Take Filter 8 as an example, 243 binary variables and 419 continuous variables are defined in our proposed model, while 3568 binary variables are needed in [6]. Our reduced model therefore results in a significant improvement of solution time. In particular, the same problem is solved at 17X faster than that in [6]. It shows that the proposed algorithm is a good MCM synthesizer for large search spaces.

The trade-off between adder cost (circuit area) and maximum logic depths (delay time constraint) is also investigated. When the maximum logic depth is set to 3 (D_{max} = 3), the adder costs in Filters 3, 6, and 8 are increased slightly comparing to the unconstrained case, but the logic depth is reduced by one-third, which allows faster operation. The most complicated problem in our design examples is Filter 8 in the search space of SSD when D_{max} = 3. The problem contains 433 binary variables, 2607 continuous variables, and 16122 linear constraints. The resultant adders cost and logic depth are 31 and 3, respectively. The solution
time of the 0-1 MILP problem is 92.13 seconds.

The computation is also analyzed. Our 0-1 MILP model with different search spaces and the 0-1 PILP model with MSD search space is compared. Figure 5 and Fig. 6 show the averages of cputimes, logic depths, and adder costs for 100 random Oset. Oset are fixed at 10 coefficients in Fig. 5 while the wordlength of Fig. 6 are fixed at 10 bits. 

From figures, the 0-1 MILP model shows a lower computation time than 0-1 PILP model. Also, the figures show the restriction of the logic depth in the 0-1 MILP model with the expense of computation time and adder cost. Furthermore, the computation time of both 0-1 PILP model and 0-1 MILP model increase exponentially with respect to the wordlength.

A further comparison between our proposed algorithm, and a broader range of algorithms like, 0-1 PILP [6], other CSE-based algorithms [2], [8] and graph-based algorithms [10], [11] and the analytical lower bound [13] are listed in Table 5. A 24-th order linear-phase lowpass filter with 12-bit wordlength [10] is used. Our proposed algorithm outperforms all CSE algorithms, and gives the optimal sharing and the lowest adder cost at a low logic depth. It is better than [6] due to the effective expanded SSD search space. On the other hand, graph-based methods produce solutions with lower adder costs than our proposed algorithm, since they exploit coefficients relationship. However, graph-based algorithms give design of high logic depths and they cannot handle high-order filter synthesis.

5. Conclusion

We have proposed a novel CSE algorithm that models the synthesis of MCM in fixed-point arithmetic into a 0-1 MILP problem. The formulated problem is solved efficiently with logic depth consideration. In our practical-size experiments, the proposed method solves the problems up to 17X faster than that in [6]. Our CSE algorithm produces the global optimum in the SSD search space which is an extended search space of MSD. We note that better solutions may still exist outside the SSD search space. In our on-going work, we would develop CSE algorithms that cover the entire search space of all-digit representations [5].

Acknowledgment

The authors would like to thank Prof. P. Flores for sharing the filter coefficients.

References


<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Adder cost</th>
<th>Logic depth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analytical lower bound [13]</td>
<td>14</td>
<td>3</td>
</tr>
<tr>
<td>Paako [8]</td>
<td>23</td>
<td>4</td>
</tr>
<tr>
<td>CI [10]</td>
<td>19</td>
<td>4</td>
</tr>
<tr>
<td>Yao [2]</td>
<td>21/18</td>
<td>3/4</td>
</tr>
<tr>
<td>0-1 PILP (MSD) [6]</td>
<td>20</td>
<td>3</td>
</tr>
<tr>
<td>0-1 MILP (SSD)</td>
<td>18</td>
<td>3</td>
</tr>
</tbody>
</table>

Fig. 5 Comparison between 0-1 PILP model and 0-1 MILP model under MSD, SSD and SSD* representations with respect to the size of wordlength.

Fig. 6 Comparison between 0-1 PILP model and 0-1 MILP model under MSD, SSD and SSD* representations with respect to the number of random coefficients.


