A Common Subexpression Sharing Approach for Multiplierless Synthesis of Multiple Constant Multiplications

Yuen-Hong Alvin Ho, Chi-Un Lei and Ngai Wong *

Abstract—**In the context of multiple constant multiplications (MCM) design, we propose a novel common-subexpression-elimination (CSE) algorithm that models synthesis of coefficients into an estimated cost function. Although the proposed algorithm generally does not guarantee an optimum solution, it is capable of finding the minimum/minima of the function in practically sized problems. In our design examples that have known optimal solutions, syntheses of coefficients using the proposed method match the optimal results in a defined search space. We also discover the relationhsip and propose an improvement search space for optimization that combine all minimal-signed-digit (MSD) representations as well as the shifted sum (difference) of coefficients to explore the hidden relationship. In some cases, the proposed feasible solution space further reduces the number of adders/subtractors in the synthesis of MCM from all MSD representations.**

Keywords: ***common subexpression sharing, multiple constant multiplications, genetic algorithm***

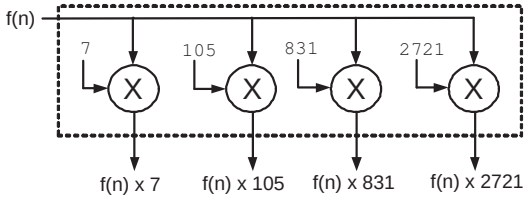

Figure 1: A multiplier-based MCM example

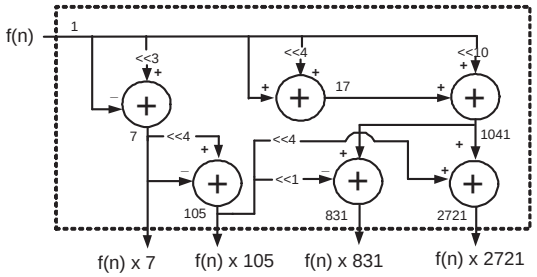

Figure 2: A multiplierless-based MCM example

1 Introduction

Multiple constant multiplications (MCM) are typical operations in digital signal processing (DSP) as well as in the design of finite-impulse-response (FIR) filters, as shown in Fig.1. Multiplierless MCM is the focus of a lot of research on high-speed and low-power devices. In multiplierless MCM, multipliers are replaced by simpler components such as adders, and hard-wired shifts, as shown in Fig.2. (The meaning of "adders" in our paper includes subtractors as well because their hardware cost is similar.)

A commonly used method for minimizing the number of adders in a MCM block is common-subexpression-elimination (CSE) [1–7]. Conventional CSE algorithm uses different signed-digit representations and looks for common subexpressions to decompose coefficients. A proposed algorithm in [1] finds common subexpressions from the CSD representation. Adder reduction of subexpressions is estimated using an exhaustive search. Flores *et al.* have proposed an optimum solution technique [7] that uses a SAT-based 0-1 integer linear programming (ILP) solver. Their results show that the larger search space of all minimal-signed-digit (MSD) representations has improvement over the CSD representation in most cases.

Feasible solution space other than the search space of signed-digit representations may also be used to synthesize MCM. For example, [6] proposes an augmented differential coefficient approach that expands the design space by employing both differences and sums of filter coefficients. And [4] proposes an algorithm that searches for pairs of available subexpressions to synthesize coefficients. The algorithm extracts all subexpressions of coefficients from CSD patterns and also treats coefficients themselves as CSD subexpressions of each other.

In this paper, we propose a CSE algorithm, which determines the syntheses of coefficients by using an estimated cost function. The function accepts syntheses of coef-

*Y-H. Alvin Ho and C-U. Lei and N. Wong are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong. Phone: ++852 +2859 1914 Fax: ++852 +2559 8738 Email: {alvin, culei, nwong}@eee.hku.hk

ficients as inputs and returns an integer that estimates the adder cost. Genetic algorithm [8][9] is used to minimize the cost function when the search space is large. It is found that the solutions of the proposed CSE algorithm match the optimum solutions in [7]. We also propose a feasible solution space that uses shifted sum (difference) of coefficients. The expansion of the feasible solution space is applied into our algorithm and has shown improvements over all MSD representations.

Multiplierless MCM can be synthesized without subexpression sharing. An example is the simple CSD method in [1][2], in which the adder cost to synthesize a coefficient is (#nzbit of the coefficient -1) where #nzbit denotes the number of non-zero bits in the CSD or MSD representations. Because the simple CSD method synthesize a multiplierless MCM without sharing any subexpression, the result of the simple CSD method is used as the upper bound of adder cost in the synthesis of multiplierless MCM. On the other hand, the minimum required adders to synthesize a multiplierless MCM from a set of coefficients are not well determined. Because at least one adder is required to synthesize a coefficient, a well known lower bound of adder cost is the number of coefficients in the MCM.

This paper is organized as follows. In Section 2, we introduce the notations in the paper. In Section 3, different feasible solution spaces are shown in a lookup table of coefficient decompositions. The way to extract decompositions from the CSD or MSD representations as well as the proposed expansion of feasible solutions is described. In Section 4, an estimated cost function that uses lookup table of decompositions is proposed. In Section 5, we introduce a CSE algorithm that synthesizes a multiplierless MCM using the estimated cost function. In Section 6, a worked-out example is given to illustrate the proposed algorithm numerically. In Section 7, our results are compared to the representative results in [4] and [7].

2 Notation

The following notations are used in this paper.

- "C" denotes coefficient.
- "#nzbit" denotes the number of non-zero bits in the CSD or MSD representations.
- "S" denotes subexpression.
- "$Cset$" is defined as the set of coefficients that is waiting to be decomposed.
- "N" is the number of coefficients in $Cset$.
- "n" is an integer where $1 \leq n \leq N$.
- "C_n" is the n^{th} coefficient in $Cset$.

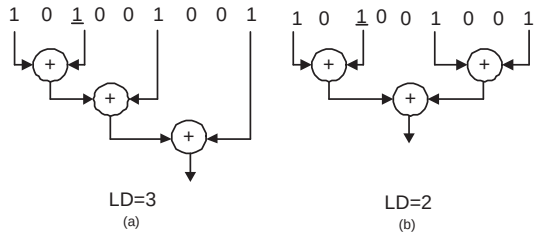

Figure 3: Example of synthesing a coefficient 201: (a) Logic depth (LD) = 3. (b) Logic depth = 2.

- "LT_n" is the decomposition table of C_n.
- "L_n" is the number of decompositions in LT_n.
- "x_n" is an integer variable that is in the range of $1 \leq x_n \leq L_n$. It is also used as an index corresponding to a decomposition in LT_n.
- "$Dset$" is the set of coefficients that have determined synthesis.
- "RT" is the result table that stores the synthesis of multiplierless MCM.
- "LDB" is the pre-defined logic depth bound that limits the maximum logic depth of the MCM. An logic depth example is shown in Fig. 3.

3 Feasible Solution Space

3.1 Introduction

The adder cost minimization problem in the synthesis of a set of coefficient of size N is NP-complete [10]. Therefore, the search space should be bounded to reduce the searching time. In the proposed CSE method, the negative coefficients are converted at the output of MCM block. Only the absolute values of negative coefficients are considered. Similarly, even coefficients are continuously divided by 2 until they become odd. Original coefficients are implemented from left-shifting the corresponding odd coefficients. Also, duplicated coefficients are synthesized once only because multiple outputs can be wired from a single coefficient. The modified coefficients have reduced the search space. As a result, the coefficients in the synthesis of MCM are unique, positive, and odd.

Equation (1) defines a decomposition of a coefficient C_n:

$$C_n = \pm S_i \times 2^p \pm S_j \times 2^q \tag{1}$$

where $S_i(S_j)$ is a subexpression of C_n that must be positive and odd; p and q are left shifts of S_i and S_j for p and q bits respectively; $S_i(S_j) \neq C_n$.

There are infinite decompositions that satisfy equation (1) in a single coefficient. Some constraints are required to limit the search space. In our CSE algorithm, each coefficient C_n generates a lookup table LT_n that stores the set of feasible decompositions of C_n. Different decompositions and values of S_i, S_j, p, and q of coefficient 831 are shown as an illustrative example.

3.2 Table Generation

Step 1) For each coefficient C_n, a lookup table LT_n that stores decompositions of C_n is generated based on all MSD representations of C_n.

During the decomposition extraction from a MSD representation, the decomposition that has the same pair of subexpressions is inserted into LT_n once only. This is because the synthesis of different decompositions of a coefficient is equivalent to each other if they have the same pair of subexpressions.

An example of decomposition extractions is shown in Table 1. $C_2 = 831$ has 3 different MSD representations and they are $10\underline{1}01000000\underline{1}$, $100\underline{11}000000\underline{1}$, and $01101000000\underline{1}$, where $\underline{1}$ represent negative one digit.

The first MSD of $C_2 = 831$ is in fact also the CSD of C_2. Decompositions from $x_2 = 1$ to 7 are generated from the CSD representation. They are extracted from the combinational patterns of the coefficient's signed-digit representation. For examples, the decomposition at $x_2 = 5$ ($\boxed{1}0\boxed{\underline{1}}01000000\underline{1}$) is extracted as $10\underline{1} << 8 + 1000000\underline{1}$ ($831 = 3 \times 2^8 + 63$). Similarly, the decomposition at $x_2 = 6$ ($\boxed{1}0\underline{1}0\boxed{1}000000\underline{1}$) is extracted as $10001 << 6 + \underline{1}00000000\underline{1}$ ($831 = 17 \times 2^6 - 257$).

The rest of MSD representations are also used to expand the number of decompositions. The second and the third MSD representations are $100\underline{11}000000\underline{1}$ and $01101000000\underline{1}$. The additional decompositions are indicated from $x_2 = 8$ to 11 and from 12 to 15 respectively.

CSD and MSD representations have the same minimum number of non-zero bits. The decompositions of all MSD representations expand the solution space over the CSD representation. The table size of all MSD representations is about twice the table size of the CSD representation in Table 1. The increased search space provides more possible decompositions that may lead to less adder cost in the synthesis of multiplierless MCM.

Since small coefficients often occur in the MCM of FIR filters, the decomposition process in this step is often repeated. To reduce the unnecessary computation, decompositions are pre-generated and stored into the hard disk for coefficients up to 13-bits (i.e. 8192). Decompositions of coefficient that are not pre-generated, they are generated during execution and save for future used. The size of our saved data is about 10MB in Matlab 7.

Step 2) Insert decompositions that satisfy equation (2) into LT_n for all n.

Equation (2) is defined as:

$$C_n = \pm S_k \times 2^p \pm S_l \times 2^q \tag{2}$$

where equation (2) is a more constrained version of equation (1). In equation (2), $S_k(S_l)$ is a subexpression of C_n that must be positive and odd; p and q are left shifts of S_k and S_l for p and q bits respectively; $S_k(S_l) \neq C_n$; S_k must be a coefficient in $Cset \cup Dset \cup \{1\}$; S_l must either be a coefficient in $Cset \cup Dset \cup \{1\}$ or #nzbit of C_n − #nzbit of $S_l \geqslant 1$.

Since C_n, S_k, and S_l are restricted to be odd, one of the $S_k \times 2^p$ or $S_l \times 2^q$ must be odd and the other must be even. Equation (2) is then manipulated as follows:

$$C_n = \pm S_k \pm S_l \times 2^q \text{ for } p = 0 \text{ and } q \geqslant 1 \tag{3}$$

or

$$C_n = \pm S_k \times 2^p \pm S_l \text{ for } p \geqslant 1 \text{ and } q = 0 \tag{4}$$

Also, since both C_n and S_k are in $Cset \cup Dset \cup \{1\}$; C_n, S_k, S_l are all positive, equation (3) is manipulated as follows:

$$S_l = (C_n + S_k) \times 2^{-q} \quad \text{or} \tag{5}$$
$$S_l = (C_n - S_k) \times 2^{-q} \quad \text{or} \tag{6}$$
$$S_l = (-C_n + S_k) \times 2^{-q} \tag{7}$$

Equation (4) is manipulated as follows:

$$S_l = +C_n + S_k \times 2^p \text{ or} \tag{8}$$
$$S_l = +C_n - S_k \times 2^p \text{ or} \tag{9}$$
$$S_l = -C_n + S_k \times 2^p \tag{10}$$

Decompositions that satisfy equation (2) are determined based on the shifted sum (difference) of coefficients (C_n and S_k). Exhaustive search are used to search for the feasible decompositions in equation (5) to (10). Decompositions from $x_2 = 16$ to 22 in Table 1 shows the result of 831 when $Cset = \{2721, 831, 105, 7\}$.

Let MCL be the maximum wordlength of coefficients in the CSD representation. For an estimation of complexity, each coefficient has an upper bound of comparisons of $O(N \times MCL)$. Therefore, the cumulated complexity for N coefficients in $Cset$ is $O(MCL \times N^2)$.

The proposed expansion increases the feasible solution space and as a result, better solution may be found. If S_l is in the set $Cset \cup Dset \cup \{1\}$, C_n may be synthesized with 1 additional adder. Otherwise, the upper bound of C_n synthesis equals to #nzbit of S_l.

Similar to the decompositions that are generated from the CSD and the MSD representations, decompositions

of the shifted sum (difference) of coefficients have the same adder cost upper bound. Because S_l is constrained to have at least one lesser #nzbit than C_n in equation (2), the synthesis cost of C_n in any decomposition does not exceed the upper bound (the simple CSD method). Decompositions of the shifted sum (difference) of coefficients introduce decompositions that cannot be found from the CSD and the MSD representations. Section 6 shows a case of MCM that leads to a lower adder cost from the proposed decompositions numerically.

Step 3) Remove invalid decompositions from LT_n based on RT and LDB.

A decomposition in LT_n is invalid if it immediately causes the critical path of MCM exceed LDB when it is synthesized. The critical path of MCM is the largest logic depth (LD) of the synthesized coefficients in RT. The synthesized coefficient in RT has a logic depth equals to $\max(\{LD \text{ of } S\}) + 1$ where LD denotes logic depth and S corresponds to the pair of subexpressions of the decomposition in RT. Subexpressions that have not been synthesized in RT is assumed to have the lowest possible logic depth calculated as $\lceil \log_2(\#\text{nzbit of } S) \rceil$. A decomposition is also invalid if it causes a feedback in the synthesis of MCM after it is inserted into RT.

The decomposition at $x_2 = 22$ in Table 1 is $111 \times 2^5 - 2721$. When $LDB = 3$, the decomposition immediately causes the critical path of MCM exceed LDB. It is found to be invalid as shown below:

$$\begin{aligned} LD \text{ of } 111 &= \lceil \log_2(\#\text{nzbit of } 111) \rceil &= 2 \\ LD \text{ of } 2721 &= \lceil \log_2(\#\text{nzbit of } 2721) \rceil &= 3 \\ & \max(\{2,3\}) + 1 &= 4 > LDB = 3 \end{aligned}$$

Table 1: Complete table of $C_2 = 831$

x_2	Decomposition	Search Space	
1	$1 \times 2^{10} - 193$	CSD / MSD	$\boxed{1}\,0\underline{1}01000000\underline{1}$
2	$-1 \times 2^8 + 1087$	CSD / MSD	$10\,\boxed{\underline{1}}\,01000000\underline{1}$
3	$1 \times 2^6 + 767$	CSD / MSD	$10\underline{1}0\,\boxed{1}\,000000\underline{1}$
4	$-1 + 13 \times 2^6$	CSD / MSD	$10\underline{1}0100000\,\boxed{\underline{1}}$
5	$3 \times 2^8 + 63$	CSD / MSD	$\boxed{1}\,0\,\boxed{\underline{1}}\,01000000\underline{1}$
6	$17 \times 2^6 - 257$	CSD / MSD	$\boxed{1}\,0\underline{1}0\,\boxed{1}\,000000\underline{1}$
7	$1023 - 3 \times 2^6$	CSD / MSD	$\boxed{1}\,0\underline{1}0100000\,\boxed{\underline{1}}$
8	$-1 \times 2^7 + 959$	MSD	$100\,\boxed{\underline{1}}\,\underline{1}000000\underline{1}$
9	$-1 \times 2^6 + 895$	MSD	$100\underline{1}\,\boxed{\underline{1}}\,000000\underline{1}$
10	$7 \times 2^7 - 65$	MSD	$\boxed{1}\,00\,\boxed{\underline{1}}\,\underline{1}000000\underline{1}$
11	$15 \times 2^6 - 129$	MSD	$\boxed{1}\,001\,\boxed{\underline{1}}\,000000\underline{1}$
12	$1 \times 2^9 + 319$	MSD	$0\,\boxed{1}\,101000000\underline{1}$
13	$1 \times 2^8 + 575$	MSD	$01\,\boxed{1}\,01000000\underline{1}$
14	$9 \times 2^6 + 255$	MSD	$0\,\boxed{1}\,10\,\boxed{1}\,000000\underline{1}$
15	$511 + 5 \times 2^6$	MSD	$0\,\boxed{1}\,10100000\,\boxed{\underline{1}}$
16	$383 + 7 \times 2^6$	Proposed	
17	$1055 - 7 \times 2^5$	Proposed	
18	$1279 - 7 \times 2^6$	Proposed	
19	$-961 + 7 \times 2^8$	Proposed	
20	$1041 - 105 \times 2^1$	Proposed	
21	$-9 + 105 \times 2^3$	Proposed	
22	$111 \times 2^5 - 2721$	Proposed	Invalid

4 Proposed Estimated Cost Function

We propose a way to estimate the adder cost of MCM using all lookup tables from LT_1 to LT_N. The value of x_n corresponds to a decomposition in LT_n as an example in Table 1. We define a MCM estimated cost function as:

$$\begin{aligned} f(x_1, ..., x_N) = & \#\text{ValidDecomposition} \\ & + \sum_{i}^{All} (\#\text{nzbit in } NS_i - 1) \end{aligned} \tag{11}$$

where f accepts N input variables from x_1 to x_N and returns an integer number. Before the evaluation, NS is an empty set, and #ValidDecomposition is equal to the number of decompositions in RT.

During the evaluation, decompositions are temporarily inserted into RT in a sequential order, from x_1 to x_N. If the decomposition at x_n is invalid, C_n is inserted into NS. Otherwise, the decomposition at x_n is temporarily inserted into RT; #ValidDecomposition is increased by 1; for the subexpressions at x_n that are not in $Cset \cup Dset \cup 1$, they are inserted into NS.

At the end of all insertions, the value of f is calculated as shown in equation (11). Decompositions that are temporarily inserted are removed after the evaluation. The adder cost of NS is calculated based on the simple CSD method. As a result, the estimated cost function is an upper bound of the adder cost when the MCM is synthesized according to x_1 to x_N.

In a typical MCM design, f may have 10 input variables and each variable may range from 1 to 10. The search space of f is 10^{10}. The minimization complexity of f that uses brute force approach is $O(M^N)$ where M is the table size and N is the number of variables.

It is assumed that the decompositions in a table are randomly ordered. However, useful information can still be obtained from the relationship between different input variables. Genetic algorithm is used into this problem because it can give a local optima solution efficiently even the search space is complicated. In genetic algorithm, minimization of f is started from a population of randomly chosen input variables. Genetic algorithm searches for a pattern of inputs that is closed to the minimum. The pseudo-code of genetic algorithm is shown below:

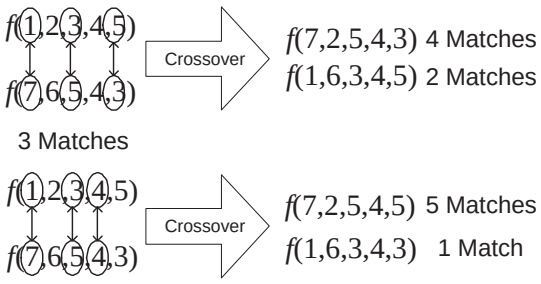

Figure 4: A crossover operation when $f(7,2,5,4,5)$ is the minimum of f

1. Choose initial population
2. Evaluate the fitness of each individual in the population
3. Select best-ranking individuals to reproduce
4. Breed new generation through crossover and mutation (genetic operations) and give birth to offspring
5. Evaluate the individual fitnesses of the offspring
6. Replace worst ranked part of population with offspring
7. Repeat step 3 to step 6 until <terminating condition>

A new population can obtained from randomly exchanging (crossover) variables between two input patterns (chromosomes). An example of a crossover operation is shown in Fig. 4. $f(7,2,5,4,5)$ is assumed to be at the minimum of f; $f(1,2,3,4,5)$ and $f(7,6,5,4,3)$ are one of the pairs of chromosomes in a population. The circled input variables are randomly chosen and exchanged. The number of input vairables that matches the input pattern of $f(7,2,5,4,5)$ are shown in Fig. 4 to indicate the fitness.

In our CSE algorithm, the minimum of f is unknown. The fitness of a chromosome is determined by the estimated cost f. A good cost function is important since the fitness formulation is related to the speed and accuracy. After each crossover operation, the newly created chromosomes are inserted into the population and sorted according to their adder cost. The population has a pre-defined size. Therefore, chromosomes with a large adder cost are dropped before the next crossover operation. Crossover operations are performed iteratively to minimize f. Although genetic algorithm generally does not guarantee an optimum solution, examples have shown that the final solution is well adapted into our CSE algorithm.

Table 2: Subexpression table of $C_1 = 2721$ and $C_2 = 831$

x_1	1	2	.....	24	.....	32	33
	1	1	.....	1041	.....	4384	111
$C_1 = 2721$	673	2209		105		831	831
x_2	1	2	3	.....	19	20	21
	1	1	1	.....	961	1041	9
$C_2 = 831$	193	1087	767		7	105	105

5 Proposed Algorithm

We are proposing an algorithm to maximize the subexpression sharing using equation (11). A simplified flowchart of the algorithm is shown in Fig. 5. The description of the algorithm is as follows:

1. Reducing all multiplication values of MCM to a set of coefficients that are unique, positive, and odd as illustrated in Section 3. Put them into $Cset$.
2. Order the coefficients in $Cset$ by their #nzbit in descending order. If there are multiple coefficients that have the same #nzbit, order them by their integer values in descending order. Create or update all LT_n according to the procedures in Section 4
3. Search for valid decompositions that may synthesize a coefficient with 1 adder in a sequential order, from LT_1 to LT_N. If a coefficient may be synthesized with 1 adder, synthesize the coefficient and update RT; move the coefficients from $Cset$ to $Dset$. If one or more decompositions may synthesize a coefficient with 1 adder, choose a decomposition that would result in the lowest logic depth. If a coefficient is synthesized, go to 7). Otherwise go to 4).
4. Minimize equation (11) and save the sets of decompositions that have the minimum cost.
5. If multiple minima are found in equation (11), pick the set of decompositions according to the following priority: i) the lowest average logic depth of the temporarily synthesized coefficients in $Cset$, ii) the lowest average logic depth of NS, iii) the smallest average value of NS.
6. Synthesize the coefficient(s) that have the most #nzbit into RT. Move the coefficient(s) from $Cset$ to $Dset$. If there are new subexpressions in RT that are not in the set of $Cset \cup Dset \cup \{1\}$, copy them into $Cset$.
7. If $Cset$ is not empty, go back to step 2). Otherwise, the RT would contain the full synthesis of MCM.

Analysis of the algorithm and an illustrative example is shown in Section 6.

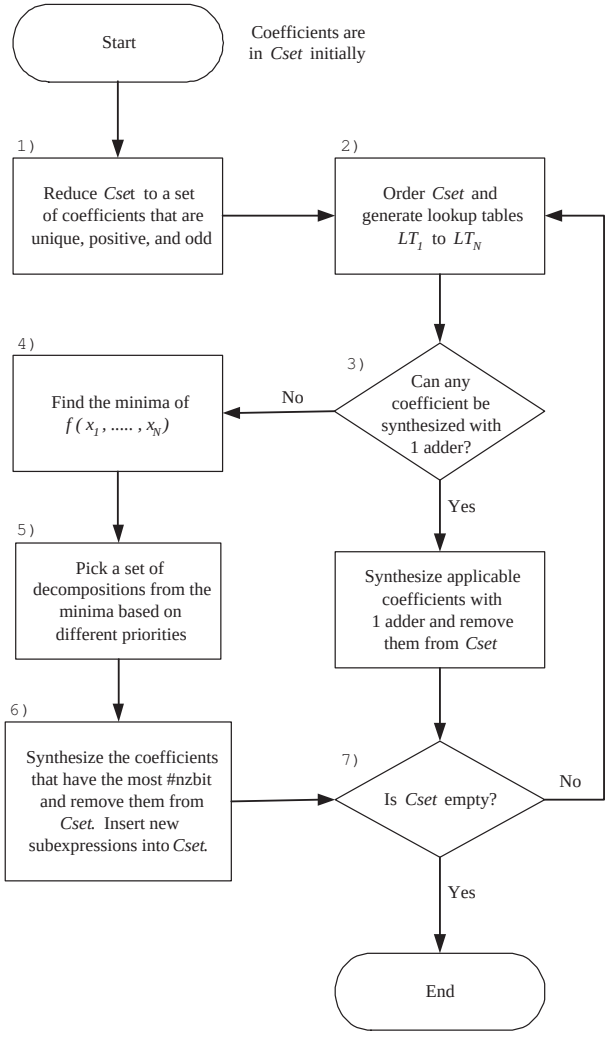

Figure 5: A simplified flowchart of our proposed CSE algorithm

6 Example

As an example to illustrate our algorithm, we set $Cset = \{105, 831, 2721, 896\}$ initially. $LDB = 3$. $N = 4$. $Dset = \{\}$. $RT = \{\}$

In 1), coefficients are reduced to unique, odd, and positive in $Cset$. The resultant $Cset$ is $\{105, 831, 2721, 7\}$ since the actual multiplication of 896 can be implemented at the output of coefficient 7 and left-shift it for 7 bits ($896 = 7 \times 2^7$).

In 2), $Cset$ is ordered and labeled as $\{C_1 = 2721, C_2 = 831, C_3 = 105, C_4 = 7\}$. LT_1 to LT_4 is generated accordingly. The complete table of LT_2 is shown in Table 1. Coefficients that have a large #nzbit generally need more adders. They are also likely to be in the critical path of the MCM. As a result, the order of coefficients in $Cset$ is important because coefficients at the beginning of $Cset$ have higher priority to synthesize. C_1 always has the highest priority and, in this case, C_4 has the lowest. Decompositions of coefficients that synthesize first generally have a larger table size because they have fewer constraints from the synthesized coefficients in RT. A larger table provides a larger search space that may lead to a smaller adder cost. Notes that the decompositions at $x_2 = 22$ is invalid because the decomposition causes the MCM greater than the predefined $LDB = 3$. It is removed from LT_2 before proceeding to the next step. As an intermediate result, $L_1 = 33$; $L_2 = 21$; $L_3 = 19$; and $L_4 = 1$.

In 3), coefficients that require only 1 adder are synthesized. This is because the decompositions are likely to appear in the minima of f. Synthesizing coefficients in this step reduces the search space of f. In this example, $C_3 = 105$ and $C_4 = 7$ may be synthesized with 1 adder. As a result, 105 is first implemented with 1 adder as $7 \times 2^4 - 7$. 7 is then implemented with 1 adder as $1 \times 2^3 - 1$. The intermediate result is $Cset = \{C_1 = 2721, C_2 = 831\}$. $Dset = \{105, 7\}$. $RT = \{105 = 7 \times 2^4 - 7, 7 = 1 \times 2^3 - 1\}$. $N = 2$.

In 4), equation (11) is generated in the form of $f(x_1, x_2)$. A simplified table that shows pairs of subexpressions for C_1 and C_2 is shown in Table 2. To illustrate the estimated cost function f, the set of NS for $f(1, 3)$ is $\{673, 767\}$. $f(1, 3) = 4 + (3 + 2) = 9$. The set of NS for $f(24, 20)$ is $\{1041\}$. $f(24, 20) = 4 + (2) = 6$.

In 5), the algorithm picks the set of coefficients decompositions that is further away from the LDB. In general, coefficients and subexpressions having a lower average logic depth may leave more flexibility for coefficients. It reduces the probability of decompositions that violate the LDB constraint. In our example, only 1 solution is found to have the minimum of 6 and it is at $f(24, 20)$. The algorithm goes to 6) directly.

In 6), coefficients with the most #nzbit are synthesized into RT while coefficients with less #nzbit remain in $Cset$. This is because the sizes of the lookup tables of the remaining coefficients may be increased from adding and subtracting the newly inserted coefficients. As a result, the algorithm in the next iteration may find more common-subexpressions with each other. In our example, the coefficient that has the most #nzbit is $C_1 = 2721$ and it has 5 non-zero bits. 2721 is therefore moved from $Cset$ to $Dset$. The new subexpressions are inserted into $Cset$. As an intermediate result, $Cset = \{831, 1041\}$. $Dset = \{105, 7, 2721\}$. $RT = \{105 = 7 \times 2^4 - 7, 7 = 1 \times 2^4 - 1, 2721 = 1041 + 105 \times 2^4\}$. $N = 2$

In 7), since $Cset$ is not empty, go back to 2) and the algorithm enters the second iteration. In 2), the lookup table for 831 is updated and another lookup table is generated for 1041. In 3), 831 is synthesized as $1041 - 105 \times 2^1$. Notes that 1041 is a subexpression that appears in the proposed expansion of search space at $x_2 = 20$ in Table 1. In 6), 1041 is synthesized as $1 \times 2^{10} + 17$. In the third iteration of 3), 17 is synthesized as $1 \times 2^4 + 1$. The final synthesis of MCM is $RT = \{105 = 7 \times 2^4 - 7, 7 = 1 \times 2^4 - 1, 2721 = 1041 + 105 \times 2^4, 831 = 1041 - 105 \times 2^1, 1041 = 1 \times 2^{10} + 17, 17 = 1 \times 2^4 + 1\}$.

7 Results

We have implemented the algorithm in [4] and extracted the data results from [7]. The results are shown in Table 4. The specifications of our filter examples are shown in Table 3.

Filter 1 consists of the coefficients described in Section 6. Filters 2 and 3 are the L1 filter and the example 2 filter in [4]. Filters 4 to 8 are the filters 4 to 8 in [7].

In Table 3, "length" stands for the filter length or the number of multiplications of the MCM. "Width" is the maximum binary bit size of coefficients. "#nzbit" is the maximum number of non-zeros terms in CSD or MSD representations. "#coeff" is the number of coefficients in $Cset$ after the process 1) in Section 5. #coeff is also the lower bound of the adder cost. "Simple CSD" is the adder cost of the simple CSD method. It is also the upper bound of the adder cost.

In Table 4, "LO" is the number of logic operations which equals the total number of adders and subtractors. "LD" is the logic depth. "cputime" is the processing time of our proposed algorithm in seconds. "LO*" is the result of the proposed algorithm using the decompositions from all MSD representations alone.

LDB is set to be $\lceil \log_2(\#\text{nzbit in Table 3}) \rceil$ in our proposed algorithm and in [4] because the increase of LDB from its minimum depth would increase the delay significantly. Take filter 3 as an example, a logic depth increase from 3 to 4 would increase the critical path by 33% but the size of the MCM is only reduced by 2%.

Genetic algorithm is used to minimize the estimated cost function. The population size is set to 300 and the genetic algorithm is terminated when no improvement is found in 5 consecutive iterations. A brute force search is performed when the size of the problem is less than 3000 because the computation time is comparable or less than that of using genetic algorithm.

The speed of our algorithm is acceptable in practically sized problems. Our program is coded in Matlab 7 and executed on a 2.6 GHz PC with 512MB memory. We have implemented the algorithm in [4] for comparison. The result of [7] are extracted directly from their paper. Although the speed of [7] cannot be directly compared, their results are included as a reference. The processing time of the proposed method is longer than those of [7] and [4] in filters 4 to 7. This is expected because the speed of convergence is slow in genetic algorithm and it also requires a longer setup time. For a more complicated problem in filter 8, the processing time of the proposed algorithm is 1280 seconds while it is terminated after 3600 seconds in [7] and it is about 7200 seconds in [4].

Although genetic algorithm generally does not guarantee an optimum solution, the final solution is well adapted into our CSE algorithm. For our synthesis examples of multiplierless MCM, it is shown that in filters 4 to 7, LO* is the optimal solution in the MSD solution space. Our results match the solution in [7] that searches for an optimal solution using the SAT-based 0-1 ILP. LO* in filter 8 outperforms the result in [7] because their program is terminated after 1 hour and as a result is not optimal.

The proposed expansion of search space is also exploited. The expansed search space further reduces the size of the MCM in some cases. Adder reductions are seen in filters 4, 7, and 8 where filters 4 and 7 are optimal in the MSD solution space. All logic depth is restricted to be 3. It saves about 58% adders than simple CSD and saves about 6% adders than MSD. It is also shown in filters 1 to 3 that the proposed algorithm produces a smaller LO than [4]. Although the relationship between the coefficients in terms of subexpressions is situtation dependent [7], the proposal search space exploits hidden and significant relationship between coefficients. It saves more adders for cases of larger filter becuase of more hidden relationship is observed. It shows that actual area is saved using the proposed design.

8 Conclusion

We have proposed a novel CSE algorithm that models synthesis of coefficients into an estimated cost function. Genetic algorithm has been shown to be capable of minimizing the cost function and give a satifactory (nearly

Table 3: Characteristics of the MCM

Filter	length	width	#nzbit	(Lower) #coef	(Upper) CSD
1	4	12	5	4	11
2	25	13	6	13	44
3	121	15	7	51	139
4	81	12	5	28	68
5	121	12	5	34	76
6	60	14	6	20	52
7	60	14	5	29	77
8	60	14	5	28	84

Table 4: Result Comparasion with LD=3

	Proposed			[4]	[7]
Filter	LO*	LO	cputime	LO	LO
1	7	6	30.078	8	NA
2	21	19	1272.515	21	NA
3	56	53	827.560	54	NA
4	29	28	38.825	28	29
5	34	34	62.875	34	34
6	22	22	119.250	22	22
7	34	30	35.750	30	34
8	34	32	1280.625	32	35

optimal) design in a reasonable time within a finite search space. Numerical examples have verified that the proposed algorithm generates coefficients that match the known optimal solutions. We have also proposed an expansion of differential coefficient solution space that generally further reduces the number of adders in the synthesis of MCM from all MSD representations, which reduces power consumption and silicon area in circuit. Futhermore,an improved version is proposed recently [11], which is to calculate the number of half adder and full adder for a more accurate (minimium area) synthesis. We are applying this novel concept and will be published in the future.

Acknowledgment

The authors would like to thank Prof. P. Flores for sharing the filter coefficients. This work was supported in part by the Hong Kong Research Grants Council and the University Research Committee of The University of Hong Kong.

References

[1] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst. II*, vol. 43, no. 10, pp. 677–688, Oct. 1996.

[2] I.-C. Park and H.-J. Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 12, pp. 1525–1529, Dec. 2002.

[3] A. G. Dempster and M. D. Macleod, "Digital filter design using subexpression eliminiation and all signed-digit representations," in *Proc. IEEE Int. Symp. on Circuits and Systems*, vol. 3, May23–26, 2004, pp. III 169–III 172.

[4] C.-Y. Yao, H.-H. Chen, T.-F. Lin, C.-J. Chien, and C.-T. Hsu, "A novel common-subexpression-elimination method for synthesizing fixed-point FIR filters," *IEEE Trans. Circuits Syst. I*, vol. 51, no. 11, pp. 2215–2221, Nov. 2004.

[5] M. D. Macleod and A. G. Dempster, "Multiplierless FIR filter design algorithms," *IEEE Signal Processing Lett.*, vol. 12, no. 3, pp. 186–189, Mar. 2005.

[6] Y. Wang and K. Roy, "CSDC: A new complexity reduction technique for multiplierless implementation of digital FIR filters," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 9, pp. 1845–1853, Sept. 2005.

[7] P. Flores, J. Monteiro, and E. Costa, "An exact algorithm for the maximal sharing of partial terms in multiple constant multiplications," in *Proc. IEEE Intl. Conf. Computer-Aided Design*, San Jose, CA, Nov. 6–10, 2005, pp. 13–16.

[8] P. Mazumder and E. M. Rudnick, *Genetic algorithms for VLSI design, layout & test automation.* Upper Saddle River, NJ: Prentice-Hall, 1999.

[9] M. Gen and R. Cheng, *Genetic algorithms and engineering design*, H. R. Parsaei, Ed. New York, NY: John Wiley & Sons, 1997.

[10] D. R. Bull and D. H. Horrocks, "Primitive operator digital filters," *Proc. Inst. Elec. Eng. G: Circuits, Devices, Syst.*, vol. 138, no. 3, pp. 401–412, 1991.

[11] L. Aksoy, P. Flores, J. Monteiro, and E. Costa, "Optimization of area in digital fir filters using gate-level metrics," in *Proc. IEEE Design Automation Conference (DAC)*, San Diego, CA, June 4–8, 2007.