Efficient Arithmetic Sum-of-Product (SOP) based Multiple Constant Multiplication (MCM) for FFT

Vinay Karkala*, Joseph Wanstrath‡, Travis Lacour*, Sunil P. Khatri*
* Department of ECE, Texas A&M University, College Station TX 77843
‡ Department of ECE, Rose-Hulman Institute of Technology, Indiana

***Abstract*—In this paper, we present an arithmetic sum-of-products (SOP) based realization of the general Multiple Constant Multiplication (MCM) algorithm. We also propose an enhanced SOP based algorithm, which uses Partial *Max-SAT* (PMSAT) to further optimize the SOP. The enhanced algorithm attempts to reduce the number of rows (partial products) of the SOP, by i) shifting coefficients to realize other coefficients when possible, ii) exploring multiple implementations of each coefficient using a Minimal Signed Digit (MSD) format and iii) exploiting the mutual exclusiveness within certain groups of partial products. Hardware implementations of the Fast Fourier Transform (FFT) algorithm require the incoming data to be multiplied by one of several constant coefficients. We test/validate it for FFT, which is an important problem. We compare our SOP-based architectures with the best existing implementation of MCM for FFT (which utilizes a cascade of adders), and show that our approaches show a significant improvement in area and delay. Our architecture was synthesized using 65nm technology libraries.**

I. Introduction

A Discrete Fourier Transform (DFT) decomposes a sequence of values into their different frequency components. The Fast Fourier Transform (FFT) is an efficient algorithm to compute DFT. FFTs are of great importance to a wide variety of applications, from digital signal processing and solving partial differential equations to algorithms for quick multiplication of large integers. Hence, efficient (in terms of area and delay) hardware implementations of FFT are of significant interest. In this paper we present Sum of Product (SOP) based algorithms that can be used to realize the constant coefficient multiplications in a hardware FFT engine.

The N-point Fast Fourier Transform is calculated using the following formula, where each of the input samples $x(n)$ is multiplied by one of a set of $2N$ constant coefficients at any given time.

$$X(k) = \sum_{n=0}^{N-1} x(n)\{cos(\frac{2.\Pi.n.k}{N}) - jsin(\frac{2.\Pi.n.k}{N})\} \quad (1)$$

Since the number of coefficients is relatively small, the multiplication operation can be optimized in a hardware implementation of an N-point FFT. Approaches to optimize such an operation are referred to as Multiple Constant Multiplication (MCM) techniques in the literature. In many FFT implementations, folding is used (i.e. only one multiplication in 1 is performed at a time, with MAC accumulating the result). Our approach is compatible with such a folded implementation.

The algorithms that we propose are based on the arithmetic Sum of Product (SOP) based realization of MCM. An arithmetic SOP is an expression of the form

$$F = \sum_{i=1}^{P} (\prod_{j=1}^{Q_j} a_{ij}) \quad (2)$$

The SOP is extremely flexible since it can be used to efficiently implement multi operand adders, MACs, multipliers and several other arithmetic blocks that are used in computer arithmetic and DSP. Hence, there have been several proposed implementations of SOP blocks to realize a variety of arithmetic circuits.

Boolean Satisfiability (SAT) [1], [2], [3] is the problem of finding a satisfying assignment (if such an assignment exists) of the variables of a Boolean formula expressed in Conjunctive Normal Form (CNF). There can be multiple satisfying assignments which satisfy a given CNF formula, and SAT returns one such assignment. A Boolean formula expressed in Conjunctive Normal Form is a conjunction (AND) of various *clauses*. Each clause consists of literals (a literal is a variable or its complement) which are ORed together.

An example Boolean formula expressed in CNF form is shown below:

$$R = (\overline{x}+a)(\overline{x}+b)(x+\overline{a}+\overline{b}) \quad (3)$$

A satisfying assignment for the above formula is a=1, b=1 and x=1, since this assignment satisfies all clauses in R.

A variant of SAT is *Max-SAT*, where the objective is to find an assignment which satisfies the maximum number of clauses of a given CNF. Another variant is *partial Max-SAT*, where a particular set H of clauses (hard clauses) must be satisfied, and the remaining clauses S (soft clauses) must be maximally satisfied (i.e the maximum number of these clauses must be satisfied).

The key contributions of this paper are:

- To the best of the authors' knowledge, this is the first approach to use SOPs for the MCM problem in the context of FFT. Previous approaches have used adder cascades for the FFT MCM problem.
- We also propose an enhanced algorithm which makes use of Partial Max-SAT to reduce the number of rows (partial products) generated by optimally exploring alternate representations of each constant coefficient of the MCM using a MSD representation. The number of choices for each coefficient increases linearly with an increase in precision, or the number of coefficients (i.e. the size of the FFT).
- The enhanced algorithm further reduces the number of partial products, by finding mutually exclusive columns (using again an exact partial Max-SAT based formulation) and replacing them by a single partial product.
- The area and delay of our SOP based FFT MCM approach is significantly improved over previous approaches (which were based on a cascade of adders).

The MCM problem for FFT was synthesized using our SOP based approaches, and the area and delay values were calculated using a 65nm technology. For the enhanced approach we use a partial Max-SAT solver called IncWMaxSatz [4].

II. Previous Work

The Fast Fourier Transform [5] is a well known and extensively studied problem, and many hardware realizations for FFT were proposed [6], [7]. In an FFT, all the input samples are multiplied by one of several constant coefficients, and hence the FFT is an instance of the Multiple Constant Multiplication (MCM) problem. In the approaches reported in the literature, constant multiplications are carried out using shift and add operations corresponding to nonzero bit positions in the constant coefficients. Though the arithmetic shift operations are inexpensive, the adder operations in such architectures require multiple cascaded adders which results in a significant increase in area and delay [8].

In order to reduce the area and delay, there have been many implementations proposed for the general MCM problem. There are a wide range of approaches for the general MCM problem. These use various algorithms such as difference based adder graphs [9], carry-save arithmetic [10], minimum spanning trees [11], hyper graphs [12], breadth first search [13], satisfiability [14], [15] in order to minimize the number of adders required. In contrast, our approach is the first technique (to the best of our knowledge), to utilize an SOP based approach for the MCM problem (in the FFT context), thereby achieving *significant* area and delay savings over the previous cascaded adder based solutions, since the SOP based approach uses a single adder.

The efficiency of an algorithm proposed for Multiple Constant Multiplication also depends on the number representation system used. There have been a variety of number representation schemes reported such as 2's complement, Canonical Signed Digit (CSD), Minimal Signed Digit (MSD) representations.

The approach of [10] addresses the realization of constant multiplication using a minimum number of carry save adders. The authors realized constant multiplication for all FFT coefficients, with an input

978-1-4244-8194-1/10/$26.00 ©2010 IEEE

word length up to 19 bits. In contrast, our implementation uses a SOP architecture, with a single (hybrid) adder. Our approach can efficiently realize the MCM circuit for coefficients of arbitrary bit precision and FFT size in an automated fashion, with significantly lower area and delay.

In [16] an architecture was proposed for 32 point FFTs. Based on trigonometric identities, a small number of constant multiplications are identified, which can be combined to generate the remaining coefficients. The method proposed in [16] uses minimum adder multipliers and was observed to be better compared to the Booth and MCM multipliers. An architecture for a 16 point FFT is also shown in the paper. This architecture was originally proposed in [17] where a CSD representation was used. [16] used the same architecture with minimum adder multipliers, which resulted in a reduction in the number of adders. In this paper we compare the area and delay of our approach with [16] and observe a significant improvement in area as well as delay. This improvement increases as the size of FFT is increased.

A previous MCM approach based on Boolean satisfiability [14], [15] optimizes the *number of cascaded adders* for an FIR filter. The main disadvantage of this formulation is that the number of clauses and variables increases rapidly with an increase in taps of the FIR filter. In our formulation, however, the number of clauses increases linearly with increase in either FFT size or number of bits of precision. Also, our approach uses an SOP based architecture, requiring a single hybrid adder and thereby drastically reducing the area and delay of the final circuit. Our SAT formulation minimizes the number of partial products in the SOP which is different from the SAT formulation of [14], [15] which minimizes the number of adders in cascade.

In [18], [19] the authors proposed heuristic algorithms to minimize the number of adders using common sub-expression elimination, which is based on an MSD representation for FIR filters. In contrast to the above approaches, this paper proposes algorithms based on an SOP architecture, which optimizes the area and delay of the MCM problem in the FFT application. Our enhanced SOP approaches provide an exact formulation to minimize the number of partial products of the SOP.

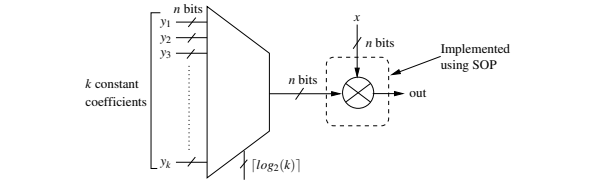
This paper does not claim to be the first to solve the MCM problem or use SOP. However, it is the first (to our knowledge) to use a SOP for the FFT MCM problem.

III. Our Approach

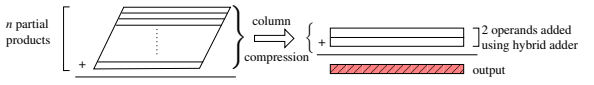
We first describe our SOP based MCM approach for FFT in Section III-A. Next, we describe our enhanced partial Max-SAT based approach to optimize the SOP based MCM approach (in terms of delay and area) in Section III-B.

A. SOP based Constant Multiplication for FFT

In an FFT, the input samples are multiplied by a fixed set of constant coefficients. Our SOP based MCM architecture is shown in the Figure 1 a). Because we do not need all the multiplications to be performed at the same time to calculate the FFT, our architecture has a multiplexer, which is used to select the required coefficient being multiplied. It uses a single SOP based multiplier (shown in Figure 1 b)) which generates all partial product terms required for the coefficient in use. We compress the set of partial products into 2 operands from LSB to MSB by using an appropriate number of full adders (or 3:2 column compressors). Each column is compressed by selecting the three earliest arriving signals and compressing them till no more than 2 bits are left in a column. In each compression one of the outputs (sum) remains in the same column while the other output (carry) is fed to the next higher column. Finally when each of the columns has no more than 2 bits, we use a single hybrid adder to generate the final result.



(a) MCM architecture



(b) SOP based MCM

Fig. 1. Our SOP based MCM architecture for FFTs

Because of the structure of the partial product terms, a large number of compressions have to be performed in the middle columns. This results in a larger delay in generating the final compressed outputs for the middle columns relative to the LSB and MSB columns.

After column compression, two operands are obtained, which need to be added. We perform the addition of these operands using a hybrid adder. The hybrid adder consists of 3 sub-adders - RCA (for lower order bits), KSA (for middle bits), CSA (for higher order bits).

The algorithm used [20] in determining the number of bits of each of the sub-adders of our hybrid adder is as follows:

- *RCA:* The width of the RCA is determined by comparing the delay of the output carry of the i^{th}-bit of the RCA with the delay of the two operands produced by column compression (for the $(i+1)^{th}$-bit position). If output carry of the i^{th}-bit of the RCA arrives earlier than the two operands in $(i+1)^{th}$-bit position, and the output carry of $(i+1)^{th}$-bit of the RCA arrives later than the two operands in the $(i+2)^{th}$-bit position, the width of the RCA can be fixed as i. But this might result in a local minimum, since the arrival times of the two operands from the compression step do not increase monotonically. Hence we perform hill climbing, where we analyze p (p = 3 in our case) additional bits. In other words if the output carry of $(i+1)$, $(i+2)$, ... , $(i+p)$ bit RCAs are all slower than the two operands in the $(i+2)$, $(i+3)$, ..., $(i+p+1)$ bit positions, we set the RCA width to i. If not, we continue to increment i. We found that larger values of p did not improve the results.
- *KSA and CSA:* After determining the number of bits of the RCA, the remaining bits need to be split between the KSA and the CSA sub-adders. To determine this split we use a brute force calculation where the critical delay is determined for each partition of the remaining bits, and the partition resulting in the minimum delay is selected. Since the number of remaining bits is not large, this step is quite fast (it completes within 1*sec* for all all our experiments).

B. Enhanced SOP based MCM for FFT

In this section, we present our enhancement to the SOP based MCM approach described above. In the enhancement, we incorporate three additional optimizations each of these enhancements are discussed in detail below.

1) Simplifying the SOP using MSD Numeral System (E1): It is possible to optimize the number of partial products by expressing coefficients in an alternate numeral system (which is non-canonical). A *canonical numeral system* has exactly one representation for each coefficient. In particular we explore the use of an MSD representation of each constant coefficient, such that the representation of all k constant coefficients requires the fewest partial products. Note that this representation subsumes the CSD representation which was used in previous approaches [17].

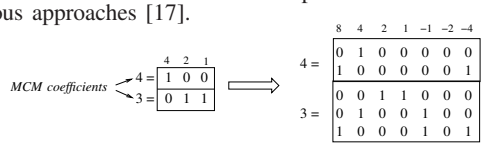


Fig. 2. Enhanced SOP based MCM

Suppose we want to implement an MCM where the coefficients are { 4, 3 } (as shown in Figure 2). If these coefficients are represented in the unsigned binary numeral system with 3 bits, they can be expressed as { 100, 011 }. This would require 3 partial products in the SOP, since the numeral system is canonical (i.e. there is exactly one way to express each coefficient).

On the other hand, if the same coefficients are expressed in a numeral system where the bit positions have the weights {2^3, 2^2, 2^1, 2^0, -2^0, -2^1, -2^2 } (as shown on the right side of Figure 2), then we note that the coefficient '4' can be expressed in 2 ways, while the coefficient '3' can be expressed in 3 ways. Now we need to select exactly one representation for each coefficient. If we represent the coefficient 4 as 0100000, and the coefficient 3 as 0100100, we note that we need only 2 partial products in the MCM. This (compared to 3 partial products for the binary numeral system) allows us to reduce the delay, area and power of the resulting SOP based MCM realization. Of course, the presence of any negative weighted columns in the result would require us to use the 2's complement of the input

data x during partial product generation. The delay and area overhead due to this is accounted for in our results.

Let variables v_k, v_{k-1}, $\cdots$ v_0, v_{-0}, v_{-1}, $\cdots$, $v_{-(k-1)}$ be the variables corresponding to the columns with weights 2^k, 2^{k-1}, $\cdots$ 2^0, -2^0, -2^1, $\cdots$, -2^{k-1} respectively, in the MSD numeral system. For each coefficient c_i, we construct a set S_i of all possible ways of expressing c_i in the redundant numeral system. Now our goal is to pick a particular representation for each c_i from S_i such that the total number of powers of 2 required to represent all coefficients is minimized.

We solve this exactly using a Partial Max-SAT based approach. Again, we use the example of Figure 2 to illustrate this. Recall that a partial Max-SAT problem consists of hard clauses (which must be satisfied) and soft clauses (which must be maximally satisfied).

1) We first write hard clauses to indicate all the ways that a coefficient can be expressed. For each coefficient c_i, let s_i^j be a Boolean variable representing the j^{th} element of S_i. Assume that $c_1 = 4$ and $c_2 = 3$ in Figure 2. Then, we write s_i^j in terms of v's as:
$s_1^1 = v_2 \equiv (\overline{s_1^1} + v_2)(s_1^1 + \overline{v_2})$
$s_1^2 = v_3 \cdot v_{-2} \equiv (s_1^2 + \overline{v_3} + \overline{v_{-2}})(\overline{s_1^2} + v_3)(\overline{s_1^2} + v_{-2})$
Note that the expressions to the right of the '$\equiv$' sign are the SAT clauses for the corresponding expressions on the left of the '$\equiv$' sign.
Similarly, we can write SAT clauses for the other coefficient.
2) We next write hard SAT clauses to indicate that for each coefficient c_i, at least one of its representations in S_i must be selected:
$(s_1^1 + s_1^2)(s_2^1 + s_2^2 + s_2^3)$
3) We write hard clauses to indicate that for each c_i, we need to select exactly one of its representations from S_i:
$(\overline{s_1^1} + \overline{s_1^2})(\overline{s_2^1} + \overline{s_2^2})(\overline{s_2^1} + \overline{s_2^3})(\overline{s_2^2} + \overline{s_2^3})$
4) Finally we write a set of *soft* clauses to express our goal (of minimizing the number of v variables that should be used). This minimizes the number of partial products in the SOP based MCM realization, since each v variable corresponds to a partial product.
$(\overline{v_3})(\overline{v_2})(\overline{v_1})(\overline{v_0})(\overline{v_{-0}})(\overline{v_{-1}})(\overline{v_{-2}})$

The final partial Max-SAT instance consists of the conjunction of the clauses of each of the 4 items above. The partial Max-SAT solver will maximize the number of the above (soft) clauses satisfied, while satisfying all the hard clauses. Maximizing the number of soft clauses above is equivalent to minimizing the number of v variables that are set to '1' (i.e minimizing the number of partial products required in the SOP), thereby achieving our goal.

Suppose that there is a solution that utilizes p partial products, (using the unsigned binary numeral system). In our partial Max-SAT formulation, since one of the choices to express each c_i is its representation in the unsigned binary numeral system, therefore, the above partial Max-SAT formulation will never return a result with more than p of the variables set to a '1'. It is possible that the partial Max-SAT algorithm will return a solution with exactly p variables set, some of which correspond to negative weights. In such a case, we simply ignore the result, and represent each coefficient in the unsigned binary numeral system.

2) Incorporating Shifted Coefficients (E2): After generating the set S_i of ways to express coefficient c_i for each i, we consider the set $S^* = \cup_i S_i$. Now for each $s_i^j, s_k^l \in S^*$, we check if s_i^j can be obtained by a (WLOG left) shift of s_k^l. If so, we remove s_i^j from consideration. If s_i^j can be obtained by a (WLOG left) shift of s_k^l, then $\forall\ p\ s_i^p$ can be obtained by a left shift of some s_k^q. In other words, we do not need to generate any hard clauses for variables s_i^p, $\forall\ p$. This check is repeated until there are no $s_i^j, s_k^l \in S^*$ such that s_i^j can be obtained by a left shift of s_k^l.

Theorem 3.1: The above partial Max-SAT based approach ($E1$) which incorporates shifting ($E2$) results in a solution with the fewest partial products without considering the mutual exclusivity of these partial products.

3) Optimization by Finding Sets of Mutually Exclusive Partial Products (E3): If S is a set of constant coefficients represented in the form of a matrix, there could exist columns i and j in this matrix such that they both do not have $'1'$ entries in any row, (i.e. the i^{th} and j^{th} partial products are mutually exclusive). In such a scenario only one partial product among these is required during column compression, thereby reducing the number of partial products that have to be compressed. In our implementation, we apply this optimization to the solution obtained from the previous two optimizations (although it could be applied before $E1$ and $E2$ as well).

The solution obtained after the previous two optimizations can be mapped to an undirected graph $G(V,E)$, where each column i (a partial product) is a vertex $v_i \in V$, and there exists an undirected edge $e_{ij} \in E$ from v_i to v_j if $S_{ki} \cdot S_{kj} = 0$, for each row k of S. Note that S is the matrix representation of the constant coefficients induced after the optimizations in the previous two sections.

Consider a set of constant coefficients $S = \{1, 6, 9, 10\}$. These coefficients can be represented in the form of a matrix as shown in the Figure 3. The corresponding graph $G(V,E)$ is as shown on the right of Figure 3. Notice that the graph has 4 nodes corresponding to the 4 columns of the matrix, and three edges showing mutual exclusiveness between the corresponding columns. There are three ways in which the columns can be partitioned, such that any partition with two or more columns is such that these columns induce a subgraph of $G(V,E)$ that is a clique. These 3 solutions are $\{\{1\}\{2\}\{3\}\{4\}\}$ $\{\{1\}\{2,4\}\{3\}\}$, and $\{\{1,2\}\{3,4\}\}$. The partial products of each partition can be replaced by one partial product, this partial product is populated with the bits of the multiplicand if the multiplier (constant coefficient) has a '1' in the corresponding column. Hence we require 4, 3 and 2 partial products respectively for the three possible partitioning solutions. Therefore we choose the third partitioning solution since it results in the least number of partial products.

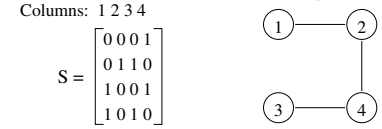


Fig. 3. $G(V,E)$ expressing mutually exclusive columns of S

In other words, the problem statement is to find a minimum cardinality partitioning solution of the columns such that the columns in each partition induce a clique in $G(V,E)$. We find this optimal partition solution using a partial Max-SAT based approach as described below (for the above example):

1) Let w_i be a variable representing a column i, and w_{ij} be a variable representing the pair of mutually exclusive columns i and j. In other words, columns i and j induce a subgraph of $G(V,E)$ which is a clique. In general, we can have variables which represent more than two columns as well. We write the first set of hard clauses which indicate that each column is covered by some partial product.
$(w_1 + w_{12})(w_2 + w_{12} + w_{24})(w_3 + w_{34})(w_4 + w_{24} + w_{34})$
2) We then write the second set of hard clauses which indicate that each column is covered by exactly one partial product.
$(\overline{w_1} + \overline{w_{12}})(\overline{w_2} + \overline{w_{12}})(\overline{w_2} + \overline{w_{24}})(\overline{w_{12}} + \overline{w_{24}})(\overline{w_3} + \overline{w_{34}})(\overline{w_4} + \overline{w_{24}})(\overline{w_4} + \overline{w_{34}})(\overline{w_{24}} + \overline{w_{34}})$
3) Finally we write the set of soft clauses which express our goal of minimizing the cardinality of the partitioning solution
$(\overline{w_1})(\overline{w_2})(\overline{w_3})(\overline{w_4})(\overline{w_{12}})(\overline{w_{24}})(\overline{w_{34}})$
The partial Max-SAT will maximize the number of soft clauses while satisfying all the hard clauses. Hence it minimizes the cardinality of the partitioning solution (i.e. the number of partial products).

Theorem 3.2: The minimum cardinality partitioning of the columns yields an exact minimum number of partial products for S, assuming that the columns of each partition induce a subgraph of $G(V,E)$ that is a clique.

Note that Theorem 3.1 yields an exact minimum number of partial products (without accounting for mutual exclusivity). Theorem 3.2 yields an exact minimum number of partial products of S, by considering mutual exclusivity. In our approach, the order in which we apply the enhancements is $E2 \rightarrow E1 \rightarrow E3$. This may not yield the globally minimum number of partial products.

IV. Experiments

We implemented all our algorithms (the SOP based MCM algorithm and the enhancements $E1$, $E2$ and $E3$) in **perl**. Given the FFT size and the number of bits of precision n desired, our perl script generates the complete SOP based MCM net-list in a 65nm PTM [21] technology. We designed and used a library of 15 gates, which had been pre-characterized for delay and area (using Spice3). The delay of our design is obtained using a capacitive load dependent gate delay model. If our MCM solution uses bits with negative weights, the delay incurred in computing the 2's complement of the multiplicand x is

	Area Ratio ($E1+E2$ SOP/Reg SOP)						Delay Ratio ($E1+E2$ SOP/Reg SOP)					
DFT Size	16	32	64	128	256	512	16	32	64	128	256	512
Prec(n)												
16	0.94	1.00	1.00	1.00	1.00	1.00	1.35	1.00	1.00	1.00	1.00	1.00
20	0.96	1.00	1.00	1.00	1.00	1.00	1.34	1.00	1.00	1.00	1.00	1.00
24	0.93	1.00	1.00	1.00	1.00	1.00	1.31	1.00	1.00	1.00	1.00	1.00
28	0.93	1.00	1.00	1.00	1.00	1.00	1.28	1.00	1.00	1.00	1.00	1.00
32	0.90	1.00	1.00	1.00	1.00	1.00	1.22	1.00	1.00	1.00	1.00	1.00
36	0.83	0.92	0.92	0.92	0.92	0.92	1.13	1.17	1.17	1.92	1.17	1.17
40	0.76	0.84	0.84	0.84	0.84	0.84	1.05	1.07	1.07	1.07	1.07	1.08
44	0.73	0.77	0.77	0.77	0.77	0.77	1.03	0.99	0.99	0.99	0.99	0.99
48	0.67	0.70	0.71	0.71	0.71	0.71	0.96	0.91	0.91	0.91	0.91	0.91

TABLE I
RATIO OF AREAS AND DELAYS OF $E1+E2$ TO REGULAR SOP

	Area Ratio ($E1+E2+E3$ SOP/Reg SOP)						Delay Ratio ($E1+E2+E3$ SOP/Reg SOP)					
DFT Size	16	32	64	128	256	512	16	32	64	128	256	512
Prec(n)												
16	0.78	0.77	0.97	1.00	1.00	1.00	1.16	0.81	0.98	1.00	1.00	1.00
20	0.78	0.76	0.95	1.00	1.00	1.00	1.24	0.82	0.96	1.00	1.00	1.00
24	0.66	0.75	0.97	1.00	1.00	1.00	1.14	0.77	0.98	1.00	1.00	1.00
28	0.73	0.81	0.96	1.00	1.00	1.00	1.14	0.79	0.96	1.00	1.00	1.00
32	0.72	0.73	0.95	1.00	1.00	1.00	1.04	0.75	0.98	1.00	1.00	1.00
36	0.64	0.59	0.87	0.92	0.92	0.92	0.97	0.83	1.06	1.92	1.17	1.17
40	0.57	0.63	0.76	0.84	0.84	0.84	0.88	0.87	1.01	1.07	1.07	1.08
44	0.57	0.52	0.66	0.77	0.77	0.77	0.89	0.77	0.93	0.99	0.99	0.99
48	0.48	0.53	0.65	0.71	0.71	0.71	0.82	0.84	0.82	0.91	0.91	0.91

TABLE II
RATIO OF AREAS AND DELAYS OF $E1+E2+E3$ TO REGULAR SOP

accounted for. The adder used for this purpose is the hybrid adder (the n most significant bits of hybrid adder are used). The active area of our design is also computed.

We compare our approach with that of [16] which is one of the best cascade-of-adder based on MCM approaches. We implemented the 16 point and 32 point FFT MCM realizations reported in [16]. Note that [16] reports realizations only for 16 and 32 point FFT. We estimated the delay of [16] by synthesizing the adders required by their design, and finding the topologically longest delay path from any input to any output of their MCM. Delays of MUXes and control structures were ignored. In our implementation of [16] the area was computed as the active areas of all the adders required by their design. Again, the area of control logic was ignored. In this way, our area and delay estimates for [16] are optimistic. On the other hand, the delay and area estimates for our approaches include the delay and area of any required control logic as well.

The results of area and delay comparison are shown only for 32 point FFT in 4, and 5 due to lack of space. The $x-axis$ in these figures is the bit precision (n). Note that our approach is extremely general, and has been applied to arbitrary sized FFTs, using the same architecture (unlike [16]).

All our SOP based MCMs are superior to that of [16] in terms of both area and delay for 32 point FFTs. Note that enhancements $E1+E2$ as well as $E1+E2+E3$ result in strictly less area than the regular SOP based MCM. However, the enhancements sometimes exhibit increased delay. This is because i) the enhanced approach needs to compute the 2's complement of the input data, thereby adding to the delay of the design and ii) with fewer partial products the maximum arrival time of the adder operands sometimes increases. This artifact is observed for both the 16 and 32 point FFTs. Also this causes the delay of the enhanced SOPs to be somewhat unpredictable as the bit precision is increased. In general, one should use the best design among the regular SOP, and the enhanced SOPs.

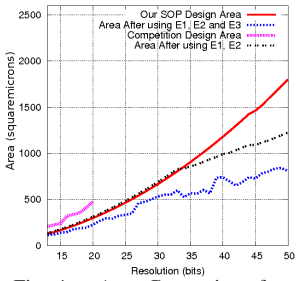


Fig. 4. Area Comparison for 32 point FFT

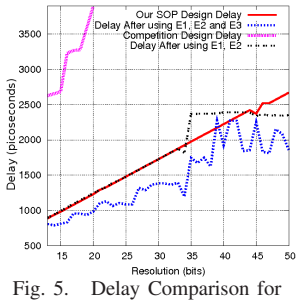


Fig. 5. Delay Comparison for 32 point FFT

We also computed the spurious free dynamic range (SFDR) for all FFTs (from 16 point to 512 point) as a function of bit precision. SFDR is calculated by computing FFT of a sinusoid and then taking the difference of the magnitude of the signal tone and the next highest spurious tone in dB. We observed that the SFDR saturates for bit widths around 50.

Table I indicates that for higher bit precision values, up to 30% gain in area can be obtained by the using $E1+E2$ alone. Similarly Columns 8 to 13 show that up to 10% delay gains can be obtained by the $E1+E2$ SOP approach. Note that due to lack of space, we do not provide the number of bits of each sub-adder in the results. On an average, for higher bit precision ($\geq$ 40 bits) the ratio of the number of bits for RCA:KSA:CSA is 28:40:21 (averaged over all FFT sizes). Note also that for all the cases, the maximum runtime to determine the widths of each sub-adder was less than a second.

Table II shows that for higher bit precisions, an area improvement of more than 50% and a delay improvement of more than 15% is observed. It can also be noted that for higher point FFTs, $E3$ does not have any affect on the results due to absence of mutually exclusive partial products. Note that all the SAT instances have runtimes below 10sec.

V. CONCLUSIONS

Multiple Constant Multiplication is a common problem that occurs in the hardware realization of arithmetic and DSP operations. In this paper, we study this problem in the context of FFT. We develop 2 algorithms – a Sum of Products (SOP) based MCM, and an enhanced SOP based MCM. Both these algorithms have the desirable feature that they only require one hybrid adder to compute the MCM, thereby significantly reducing both delay and area compared to the previous approaches (which use adder cascades). The enhanced approach attempts to reduce the number of partial products in three ways i) by shifting coefficients to realize other coefficients, ii) by expressing the coefficients in a redundant numeral system, iii) by exploiting the mutual exclusiveness within certain groups of partial products. For higher FFT sizes and larger bit precisions, the enhanced approach works much better than the regular SOP based MCM approach.

REFERENCES

[1] "http://www.cs.chalmers.se/cs/research/formalmethods/minisat/main.html," the MiniSAT Page.
[2] M. Silva and J. Sakallah, "GRASP-a new search algorithm for satisfiability," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, November 1996, pp. 220–7.
[3] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient SAT solver," in *Proceedings of the Design Automation Conference*, July 2001.
[4] H. Lin, K. Su, C. Li, and J. Argelich, "IncWMaxSatz," *Max-SAT Evaluation*, 2008.
[5] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing*. Prentice Hall, Third Edition.
[6] M. Koushik, G. Eckhard, and J. Ulrich, "A 64-point fourier transform chip for high-speed wireless lan application using ofdm," *IEEE Journal of Solid-State Circuits*, pp. 484–493, 2004.
[7] A. Vacher and M. Benkhebbab, "A vlsi implementation of parallel fast fourier transform," *European Design and Test Conference*, pp. 250–255, 1994.
[8] B. Elgharbawy, Fayed, "A data merging technique for high-speed low-power multiply accumulate units," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 145 – 148, 2004.
[9] O. Gustaffson, "A difference based adder graph heuristic for multiple constant multiplication problems," *ISCAS*, pp. 1097 – 1100, 2007.
[10] O. Gustaffson and L. Wanhammer, "Low-complexity constant multiplication using carry-save arithmetic for high-speed digital filters," *ISPA*, pp. 212 – 217, 2007.
[11] O. Gustaffson, H. Ohisson, and L. Wanhammer, "Improved multiple constant multiplication using a minimum spanning tree," *Signals, Systems and Computers*, pp. 63 – 66, 2004.
[12] O. Gustaffson, "Towards optimal multiple constant multiplication: A hypergraph approach," *Signals, Systems and Computers*, pp. 1805 – 1809, 2008.
[13] L. Aksoy, O. Gunes, and P. Flores, "An exact breadth-first search algorithm for the multiple constant multiplication problem," *NORCHIP*, pp. 41–46, 2008.
[14] P. Flores, J. Monteiro, and E. Costa, "An exact algorithm for the maximal sharing of partial terms in multiple constant multiplication," *ICCAD*, pp. 13–16, 2005.
[15] L. Aksoy, P. Flores, and J. Monteiro, "Exact and approximate algorithms for the optimization of area and delay in multiple constant multiplication," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, pp. 1013–1026, 2008.
[16] F. Qureshi and O. Gustaffson, "Low-complexity reconfigurable complex constant multiplication for FFTs," *ISCAS*, pp. 1137 – 1140, 2009.
[17] J. E. Oh and M. S. Lim, "New radix-2 to the 4th power pipeline FFT processor," *IEICE Transaction*, pp. 1740–1764, 2005.
[18] I.-C. Park and H.-J. Kang, "Digital filter synthesis based on minimal signed digit representation," *DAC*, pp. 468–473, 2001.
[19] M. Potkonjak, M. B. Srivastava, and C. A. P., "Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 151–165, 1996.
[20] S. Das, "Design automation techniques for datapath circuits," Ph.D. dissertation, University of Colorado, 2007.
[21] PTM, http://www.eas.asu.edu/∼ptm.