Proceedings of the 6th Nordic Signal Processing Symposium - NORSIG 2004
June 9 - 11, 2004
Espoo, Finland

# Low-Complexity Constant Coefficient Matrix Multiplication Using a Minimum Spanning Tree Approach

*Oscar Gustafsson, Henrik Ohlsson, and Lars Wanhammar*

Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, SWEDEN
Tel: +46-13-28 {4059, 4059, 1344}, Fax: +46-13-13 92 82, E-mail: {oscarg, henriko, larsw}@isy.liu.se

**ABSTRACT**

*In this paper a novel approach for realizing constant coefficient matrix multiplication using few additions and subtractions is proposed. This method is applicable in, e.g., FIR filter banks, transforms, and polyphase form FIR filters for sample rate changes. Examples show that the proposed method yields good results compared to realizing the matrix multiplication by utilizing multiple coefficient multiplication techniques for the rows or columns separately.*

## 1. INTRODUCTION

The problem of multiplying one data with several constant coefficients has been well studied over the years [1]–[8]. By expressing the multiplications using shifts and additions or subtractions, the number of additions required are decreased by using common partial results. This is referred to as multiple constant multiplication (MCM). It should be noted that by transposing a MCM block a sum-of-product is realized as shown in Fig. 1.

It is possible to outline the previously proposed approaches into three different classes. The subexpression sharing algorithms are based on finding recurring pattern in the coefficient representation, and, hence, depends on the representation of the coefficient [2]–[4], [6]. Multiplier blocks uses a graph representation to construct a network of adds and shifts, independent of the representation [1]. Finally, difference methods computes simple differences between the coefficients, and then applying the same method to the differences [5], [7], [8].

In many DSP applications the computations are effectively matrix multiplications on the form

$$\begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_N \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{2,1} & \dots & a_{M,1} \\ a_{1,2} & a_{2,2} & \dots & a_{M,2} \\ \dots & \dots & \dots & \dots \\ a_{1,N} & a_{2,N} & \dots & a_{M,N} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_M \end{bmatrix} = \begin{bmatrix} A_1 \\ A_2 \\ \dots \\ A_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_M \end{bmatrix} \quad (1)$$

where $x_i$ are inputs, $y_i$ outputs, and $a_{i,j}$ constants. A row is denoted $A_i$. This is the case for, e.g., FIR filter banks [9], linear transform, such as DCTs [10], polyphase decomposed FIR filters for sample rate changes [11], and state-space digital filters [10], [12].

It is, off course, possible to apply MCM to each row of the matrix and then transpose the block or each column and add the different row elements together, but the redundancy is not fully utilized.

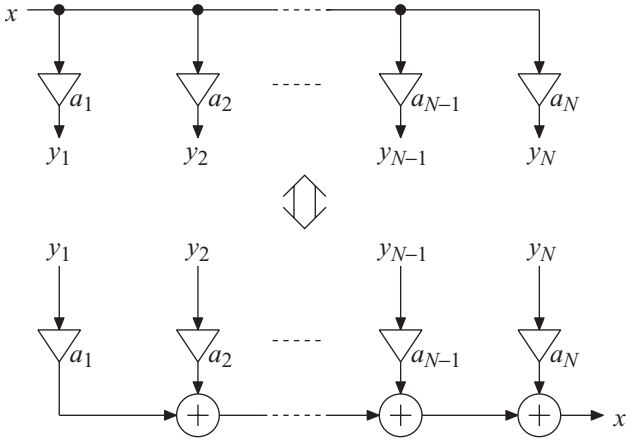


**Fig. 1.** Multiple constant multiplication block and transposed sum-of-products block.

In [13] an approach based on number splitting was proposed. However, this is aimed at working with infinite precision number representation. A two-stage approach based on subexpression sharing has been proposed in [2]. Here, first subexpressions for the columns where identified and then subexpressions for the rows. Recently, a multiplier block algorithm for matrix multiplication was presented [14].

In this work an algorithm for low complexity constant coefficient matrix multiplication based on differences is proposed. It uses a minimum spanning tree (MST) to select the coefficients, which warrants low execution time as an MST can be found in polynomial time [15], [16]. In the next section the graph representation is discussed. This is similar to the one used in [5], [7], [8]. Then the proposed algorithm is introduced, with some discussions of properties. Some experimental results are presented and compared with other algorithms. Finally, some conclusions are drawn.

## 2. GRAPH REPRESENTATION

The use of minimum spanning trees for design of multiple constant multipliers with low arithmetic complexity have previously been proposed for the one input case [5], [7], [8].

### 2.1. Undirected Graph

A row of the matrix multiplication and the relation between the rows can be represented using an undirected graph, where each vertex corresponds to one row of the matrix, $A_i$. The graph is fully connected, i.e., it has edges between all pairs of coefficients. These edges corresponds to the differ-

ence between the two rows and each edge is assigned a cost, $c_{i,j}$, corresponding to the minimal cost for implementing the difference between the rows $A_i$ and $A_j$. Fig. 2 shows an undirected graph for four rows.

### 2.2. Minimum Spanning Tree

An MST of a graph with weighted edges is a set of edges that connects all vertices while minimizing the sum of the edge weights, which in this case corresponds to minimizing the implementation cost of the differences. This is a well studied problem with several polynomial time algorithms available [15], [16]. For a graph, an MST can be found using a greedy algorithm. Hence, a short execution time is required for computing an MST. Note that, in the general case, a graph may have several MSTs.

Using an MST for selecting the differences often yields a solution where one, or several, of the original coefficients are used to form two, or more, differences. It is possible to find a set of differences, yielding a minimal implementation cost, where each coefficient is used only once to compute a difference by finding a Hamiltonian path in the graph instead of an MST [5]. A Hamiltonian path is a path that only visit each vertex once with a minimum cost. Finding such path is an NP-hard problem. However, a Hamiltonian path may only yield a solution as good as an MST, it will never be better.

The depth of an MST, i.e., the path from the root to the leaf with the highest number of vertices between them, gives the critical path of the difference stage. However, finding an MST with a constrained depth is also an NP-hard problem.

## 3. PROPOSED APPROACH

The proposed approach can be divided into four steps. In the discussion we will assume that all elements in the matrix have integer values.

I. Identical rows are identified by first dividing each row by a power of two so that at least one element is odd. Then, the sign of each row is changed so that the first non-zero element in a row is positive. Among the rows that are identical, only one is kept. Furthermore, rows that only contains a single one is removed from the matrix.
II. If there are any rows left, the edge weights, i.e., the cost for the differences, are determined using the cost measure discussed in Section 3.1.
III. The MST is computed using an arbitrary method, e.g., one discussed in [16].
IV. The required differences are now rows in a new matrix. They are realized by applying the algorithm to this matrix. This is further discussed in Section 3.3.

### 3.1. Cost Measure

The cost for realizing a difference is here defined as the number of additions required for realizing a difference using a minimum signed digit representation, e.g., canonic singed digit (CSD) representation [10]. The difference between row $i$ and row $j$ is defined as

$$d_k(i, j) = 2^m a_{k,i} \pm 2^n a_{k,j}, \quad k = 1, 2, \ldots, M \tag{2}$$

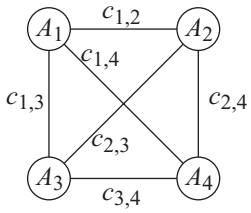


**Fig. 2.** Graph corresponding to four matrix rows.

where $m$, $n$, and the sign is selected to minimize the cost. By allowing shifts and variable sign it is possible to find differences with lower cost. The cost, i.e., the number of additions required to realize the difference is then

$$c_{i,j} = \sum_{k=1}^{M} \#\mathrm{CSD}(d_k(i, j)) - 1 \tag{3}$$

where #CSD($x$) is the number of non-zero bits of the CSD representation of $x$.

The differences corresponds to what must be added to one row to obtain another row. However, at least one row must be realized without the use of another row. This is handled by adding a root vertex† to the graph. This vertex corresponds to all possible rows with one non-zero element equal to one, i.e., [1 0 0 ... 0], [0 1 0 ... 0], etc.

There are other ways to compute the cost to realize a single row, i.e., a difference. For example, one could apply an MCM algorithm to obtain a smaller cost [14]. However, using the proposed cost measure will guarantee convergence, and that the total cost estimated after one stage will never be exceeded.

### 3.2. Convergence

The algorithm will always converge in a maximum of

$$\max_j \left\{ \sum_{i=1}^{M} \#\mathrm{CSD}(a_{i,j}) - 1 \right\} \tag{4}$$

steps, i.e., proportional to the maximum number of non-zero bits for one row. It is easy to realize this by noticing that for the worst case, one stage is removing one non-zero bit from each row. This is what happens when there is only one row in the matrix. As the algorithm stops when there is one non-zero bit left the expression in (4) follows.

The cost obtained from one MST is based on the fact that each difference is realized using separate CSD multiplications and adding up the results for each row separately. However, as the same approach can be used for the differences, we can guarantee that the MST cost is an upper bound on the resulting number of additions for each stage. Selecting other cost measures for the differences does not have this advantage as the realization of the differences then is different from the way the cost is computed.

†. The term *root vertex* is usually used in the case of directed graphs. However, the graph in question can be made a directed graph with each undirected edge corresponding to two directed edges, one in each direction, but because of that, it can be handled as an undirected graph.

### 3.3. Difference Selection

The actual savings in additions are obtained in two different ways. First, by deriving rows from other rows with a lower cost than deriving them from the root node. Second, if any of the required differences are identical, only one of them must be realized. For each edge there may be more than one difference with minimum cost. Hence, the selection of differences is important.

The current implementation of the algorithm can, at each stage, form all possible combinations of differences, and compute the cost of the corresponding MSTs. Then, one difference set with minimum cost is selected. However, this is only possible to do in reasonable time when the number of combinations are reasonably small, maybe up to 1000. Furthermore, selecting the difference set with the lowest cost at one stage is not a guarantee that the total cost after more stages will be minimized. This approach could be extended to a complete tree search, where the algorithm is applied to each possible combination of differences until convergence.

It is also possible to transpose the difference matrix and obtain a realization for the transposed matrix and then transpose the network of additions and subtraction to obtain the required rows. This may be especially useful when there are few and long rows as the algorithm generally works better on "high and narrow" matrices.

After the algorithm has converged it may be possible to find identical differences in different stages. This can be straightforwardly utilized by connecting a computed difference to a later stage, and, hence, save additions.

## 4. RESULTS

To illustrate the results with the proposed algorithm a number of example matrix multiplications are realized.

### 4.1. Example 1

In this example the following linear transformation, designed in [17] as an example of the algorithm in [2], is considered

$$T = \begin{bmatrix} 7 & 8 & 2 & 13 \\ 12 & 11 & 7 & 13 \\ 5 & 8 & 2 & 15 \\ 7 & 11 & 7 & 11 \end{bmatrix} \tag{5}$$

A straightforward implementation using CSD representation requires 29 additions. Using the algorithm in [2] leads to a total of 20 additions. Using RAGn [1] on the columns leads to 21 additions, 9 for the coefficients and 12 for adding the partial results. Applying it on the rows requires 22 additions (10 + 12). The algorithm in [14], here referred to as BHMM, yields a solution with 14 additions.

For the first stage using the proposed approach the MST in Fig. 3(a) is obtained. The total cost after the first stage is 15. The MST corresponds to the differences in (6) where $R$ denotes the root vertex and the other indices corresponds to the row number in (5). These differences yields the MST in Fig. 3(b) and a total cost of 14 (10 for the MST and 4 for the previous stage).

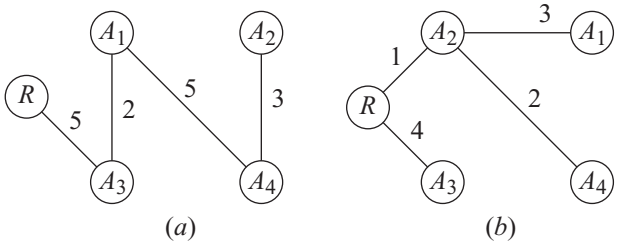


**Fig. 3.** MSTs in Example 1 for (a) stage 1 and (b) stage 2.

$$D_1 = \begin{bmatrix} d(R,3) \\ d(1,3) \\ d(1,4) \\ d(2,4) \end{bmatrix} = \begin{bmatrix} 1 & 8 & 2 & 15 \\ 1 & 0 & 0 & -1 \\ 0 & 3 & 5 & -2 \\ 5 & 0 & 0 & 2 \end{bmatrix} \tag{6}$$

The differences for the MST in stage 2 are

$$D_2 = \begin{bmatrix} d(R,3) \\ d(R,2) \\ d(1,2) \\ d(2,4) \end{bmatrix} = \begin{bmatrix} 0 & 2 & 5 & -2 \\ 1 & 0 & 0 & 0 \\ 0 & 4 & 1 & 8 \\ 7 & 0 & 0 & 0 \end{bmatrix} \tag{7}$$

From this stage all rows are realized separately. This corresponds to that all edges of the MST are between the root vertex and any other vertex. The resulting realization with 14 additions is shown in Fig. 4. The paths corresponding to the MSTs in Fig. 3 are marked in bold.

### 4.2. Example 2

In this example a linear-phase FIR filter for decimation with a factor three is designed. The passband and stopband edges are at $0.3\pi$ and $0.35\pi$, respectively. The filter is designed for a passband ripple of 0.01 and a stopband ripple of 0.001. Using a filter order of 110 this filter is synthesized in MATLAB using *remez.m* and rounding the coefficients to 12 bits precision.

A direct realization using CSD representation requires 218 additions. Applying RAGn [1] to the columns requires 52 + 72 = 124 additions. The current implementation of the BHMM algorithm [14] has problems handling negative numbers, and, thus, can not be used for comparison.

Using the proposed method, 103 additions are required. The algorithm converges in five stages. However, four additions can be removed by identifying that the same differences are required in multiple stages. Hence, 99 additions are required.

### 4.3. Example 3

In this example, general matrix multiplication with $N$x$M$ matrices is considered. The number of rows, $N$, varies between 2 and 10 in steps of 2 and the number of columns, $M$, varies between 2 and 6. For each matrix size 20 random matrices with 6 bits coefficients (max 64) are used. Only positive coefficients are used due to problems with negative numbers in the current implementation of the BHMM algorithm in [14]. The matrix multiplications are realized using the proposed method (MMST), separate columns using RAGn [1], and by using BHMM [14].

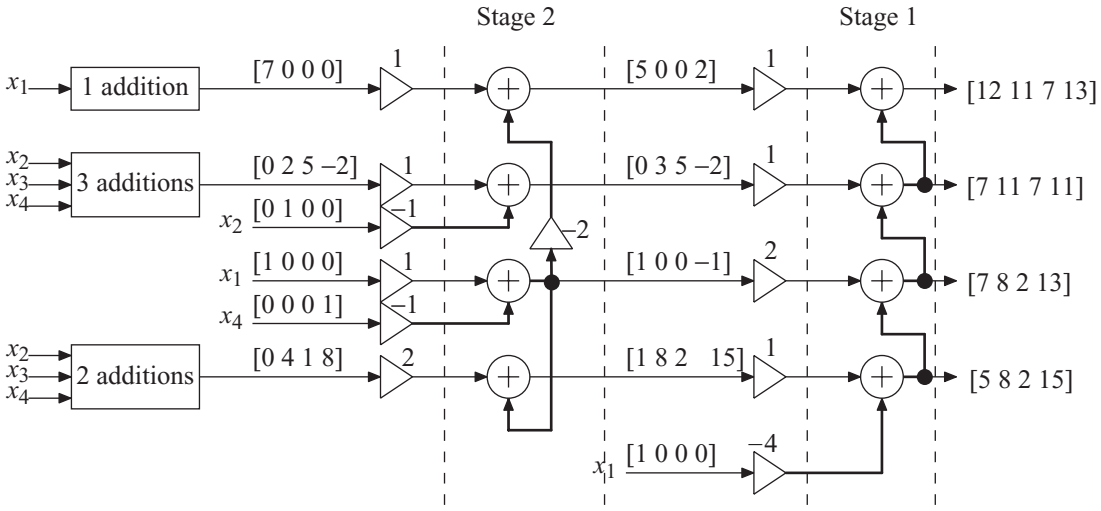


**Fig. 4.** Realization of the matrix multiplication in Example 1.

In Table 1 all 500 cases are considered and the number of cases where one algorithm is better than another is shown. From this we can see that the BHMM algorithm in [14] yields the best results in most cases. The proposed algorithm beats BHMM in approximately 5% of the cases. Comparing with separate generation of columns, the proposed algorithm is better in 28% of the cases. Although, no results are presented in this work, the execution time of the proposed algorithm is consistently significantly lower than that of the algorithm in [14].

**Table 1.** Number of cases where one algorithm is better than the other for the 500 random matrices in Example 3.

| Algorithm 1 | Algorithm 2 | Number of additions | | |
|---|---|---|---|---|
| | | 1 < 2 | 1 = 2 | 1 > 2 |
| MMST | RAGn | 139 | 77 | 284 |
| MMST | BHMM | 26 | 60 | 414 |
| BHMM | RAGn | 264 | 82 | 54 |

## 5. CONCLUSIONS

In this paper an algorithm for constant coefficient matrix multiplication with few additions and subtractions was proposed. It is based on computing a minimum spanning tree (MST) which gives that the algorithm executes in polynomial time. The results show that the algorithm can produce better results than applying standard multiple constant multiplication techniques to the rows or the columns separately. However, a useful technique would be to select the best result among the proposed technique, the BHMM algorithm in [14], and realizing each column separately using the RAGn algorithm in [1].

## 6. REFERENCES


[1] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst.–II*, vol. 42, no. 9, pp. 569–577, Sep. 1995.

[2] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan, "Multiple constant multiplications: efficient and versatile framework and algorithms for exploring common subexpression elimination," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 2, pp. 151–165, Feb. 1996.

[3] R. I. Hartley, "Subexpression sharing in filters using canonic signed digit multipliers," *IEEE Trans. Circuits Syst.–II*, vol. 43, pp. 677–688, Oct. 1996.

[4] R. Pasko, P. Schaumont, V. Derudder, S. Vernalde, and D. Durackova, "A new algorithm for elimination of common subexpressions," *IEEE Trans. Computer-Aided Design Integrated Circuits*, vol. 18, no. 1, pp. 58–68, Jan. 1999.

[5] K. Muhammad and K. Roy, "A graph theoretic approach for synthesizing very low-complexity high-speed digital filters," *IEEE Trans. Computer-Aided Design*, vol. 21, no. 2, Feb 2002.

[6] M. Martínez-Peiró, E. Boemo, and L. Wanhammar, "Design of high speed multiplierless filters using a nonrecursive signed common subexpression algorithm," *IEEE Trans. Circuits Syst.–II*, vol. 49, no. 3, pp. 196–203, Mar. 2002.

[7] O. Gustafsson and L. Wanhammar, "A novel approach to multiple constant multiplication using minimum spanning trees," in *Proc. IEEE Midwest Symp. Circuits Syst.*, Tulsa, OK, Aug. 4–7, 2002, vol. 3, pp. 652–655.

[8] H. Ohlsson, O. Gustafsson, and L. Wanhammar, "Implementation of low-complexity FIR filters using a minimum spanning tree," in *Proc. IEEE Mediterranean Electrotechnical Conf.*, Dubrovnik, Croatia, May 12–15, 2004, in press.

[9] A. G. Dempster and N. P. Murphy, "Efficient interpolators and filter banks using multiplier blocks," *IEEE Trans. Signal Processing*, vol. 48, no. 1, pp. 257–261, Jan. 2000.

[10] L. Wanhammar, *DSP Integrated Circuits*, Academic Press, San Diego, 1999.

[11] O. Gustafsson and A. G. Dempster, "On the use of multiple constant multiplication in polyphase FIR filters and filter banks," under review to *Nordic Signal Processing Symposium*, Espoo, Finland, June 9–11, 2004.

[12] L. Wanhammar and H. Johansson, *Digital Filters*, Linköping University, 2003.

[13] A. Chatterjee, R. K. Roy, and M. A. d'Abreu, "Greedy hardware optimization for linear digital circuits using number splitting and refactorization," *IEEE Trans. VLSI Syst.*, vol. 1, no. 4, pp. 423–431, Dec. 1993.

[14] A. G. Dempster, O. Gustafsson, and J. O. Coleman, "Towards an algorithm for matrix multiplier blocks," in *Proc. European Conf. Circuit Theory Design*, Kraków, Poland, Sept. 1–4, 2003.

[15] E. Lawler, *Combinatorial Optimization: Networks and Matriods*, Dover Publications, 2001.

[16] C. F. Bazlamacci and K. S. Hindi, "Minimum-weight spanning tree algorithms: a survey and empirical study," *Computers & Operations Research*, vol. 28, no. 8, pp. 767–785, 2001.

[17] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, Wiley, 1998.