MRPF: An Architectural Transformation for Synthesis of High-Performance and Low-Power Digital Filters

Hunsoo Choo, †Khurram Muhammad, Kaushik Roy

Electrical & Computer Engineering Department
Purdue University
West Lafayette, IN 47907
chooh,kaushik@ecn.purdue.edu

†Texas Instruments
Dallas, TX 75243
khurram@msp.sc.ti.com

Abstract

We present a graph theoretical methodology that reduces the implementation complexity of a vector multiplied by a scalar. The proposed approach is called MRP (minimally redundant parallel) optimization and is presented in FIR filtering framework to obtain a low-complexity multiplierless implementation. The key idea is to expand the design space using shift inclusive differential coefficients together with computation reordering using a graph theoretic approach to obtain maximal computation sharing. The transformed architecture of a filter is obtained by solving a set cover problem of the graph. A simple algorithm based on a greedy approach is presented. The proposed approach is merged with common sub-expression elimination. The simulation results show that 70% *and* 16% *improvement in terms of computational complexity over simple implementation (transposed direct form) and common sub-expression, respectively, when using carry lookahead adder synthesized from synopsys designware library in .25* μ *technology.*

1 Introduction

Digital filtering is the most frequently used computation in *digital signal processing* (DSP) systems. In many applications such as high speed and low power communication transceivers, fixed coefficient *finite impulse response* (FIR) filters are routinely employed as custom designed digital blocks. With the continual need for improving the data rates at reduced power dissipation, it is preferred for filters to be implemented in application specific integrated circuits rather than DSP core. In this paper, we examine the subject from the viewpoint of reducing the *computational redundancy* that is defined as the *excess computation over the minimum number of bit operations needed for a given sequence of operations* [5].

Multiplierless FIR filter implementation has been a topic of special interest in DSP system design area. Simple implementations using signed-powers-of-two (SPT) and canonical signed digit (CSD) number representations are reported by many researchers [11]. Many approaches, which try to find another quantization reducing the implementation cost in the vicinity of optimal solution, can not guarantee to satisfy the desired frequency response. Some recently proposed scheme such as [10] reduces the dynamic range of computation for lower power, but is not effective when there is weak correlation between coefficients. Some other approaches such as *common sub-expression elimination* (CSE) focus on the realization of low-complexity *multiple constant multiplications* [3, 8, 9]. However, because the structure from CSE approach is highly irregular, it is extremely difficult and expensive to pipeline the design to increase the operating speed. Also, in deep sub-micron technologies, the effectiveness of these algorithms is reduced due to the expensive interconnect cost.

In this paper, we use an graph based approach exploiting the *arithmetic equivalence to identify non-redundant computations* [4, 5, 6]. The proposed approach is merged with the philosophy of CSE to further improve the results. *Shift inclusive differential coefficient* (SIDC) [4] is used to maximize complexity reduction. The optimization problem is formulated as a graph where filter parameters are appropriately defined as vertices and edges. Low-complexity implementation of a fully parallel filter is obtained by using well-known graph problems. An algorithm based on greedy approach is presented to solve the optimization problem. This can be combined with all the previously cited complexity reduction schemes in a judicious manner. Even though this paper demonstrates the proposed approach for the case of FIR filter implementation, it can be directly applied to any applications which can be expressed as a *vector scaling operation* [5] like transposed direct form IIR filters.

1530-1591/03 $17.00 © 2003 IEEE

2. Differential coefficients and graph

An M-tap *linear time-invariant* FIR filtering operation is described as $y(n) = \sum_{i=0}^{M-1} c_i x(n-i) = \sum_{i=0}^{M-1} P_i^{(n)}$ where c_i is the ith coefficient and $x(n-i)$ is the data sample at the time instance $n-i$. $P_i^{(n)}$ is the partial product $c_i\ x(n-i)$ at the time instance n. Transposed direct form (TDF) of FIR filter recasts the *inner product* based filter equation as a product of a scalar (the input data) with a vector (the filter coefficients). This allows resource sharing which is the basis for CSE [3, 9, 8]. Using *Differential coefficients*, $P_i^{(n)} = (c_i - c_j)x(n-i) + P_j^{(n-i+j)}$ [5]. The goal is to maximally simplify the computation of $(c_i - c_j)x(n)$ by identifying a $P_j^{(n-i+j)}$ which must be computed prior to $c_i x(n-i)$.

Let $G = \{V, E\}$ be a graph representation for the optimization problem. Then, a vertex $v_i (\in V)$ represents a coefficient c_i and the edge weight $e_{i,j} (\in E)$ indicates the cost of resources to multiply a data sample with the differential coefficient $c_j - c_i$. Depending on the number representation of the coefficients and implementation of multiplication, $e_{i,j}$ can have a different value. If array multiplier is used for multiplication, $e_{i,j}$ represents the number of adder-arrays that is equal to the number of non-zero bits in the binary representation of a differential coefficient.

Because we target implementation of the TDF of FIR filter, $x(n)$ is the common input to all taps of FIR filter. The simple graph consisting of two vertices (v_i and v_j) and one edge $e_{i,j}$ represents $x(n) \times c_j$ can be obtained by $x(n) \times c_i + x(n) \times e_{i,j}$. Hence, any sub-graph covering all the vertices in the graph representation of a filter leads to an unique multiplierless implementation of an FIR filter. Especially, a minimum spanning tree is a preferable solution leading to an implementation of a filter with small delay [5].

3 The MRP Approach

3.1 SID coefficients

Assuming a multiplication $c_i \cdot x(n)$ has been computed already, any product of $2^L \cdot c_i \cdot x(n)$ $(L > 0)$ is easily obtained by L-bit shift of $c_i \cdot x(n)$ to the left. Because shift operation can be simply implemented by hard wiring, the computational cost of shift operation is negligible. Combining these two approaches of shifting and differential coefficients, $P_j^{(n)} = 2^L c_i x(n) \mp (c_j \pm 2^L c_i)x(n)$. This allows us more flexibility in obtaining $P_j^{(n)}$ from $P_i^{(n)}$. In the sequel, coefficient $c_j - 2^L c_i$ will be referred to as a *shift inclusive differential* (SID) coefficient, and the operator '–' may represent either an *add* or a *subtract* operation.

Complexity reduction of a filter is achieved by identifying c_i which has to be computed prior to c_j making SIDC a simple number including fewer non-zero bits for $j = 0, 1, \cdots, M-1$. If we represent this problem as a graph, then there can exist several edges between two vertices and c_j can be visited from c_i using any one of the edges. The least expensive way to visit c_j from c_i will be the simplest implementation of $(c_j - 2^L c_i)x(n)$.

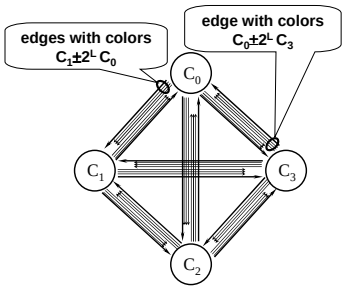


Figure 1. Graph representation using SIDC

Figure 1 shows the directed graph for a 4-tap filter. There are $2(W+1)$ edges directed from c_i to c_j for all i, j each of which represents the cost (edge weight) of SID coefficients $c_j - 2^L c_i$ for a particular value of $L (0 \le L \le W)$. Negative value of L is not considered, because it has a potential of re-quantization unless the number of bits are extended beyond the *least significant bit* (LSB) increasing the actual wordlength of intermediate computations. Hence, there are a maximum of $2(W+1)(M-1)M$ distinct colored edges in the graph.

Our goal is to identify the order of computations which reduces the overall effort required to compute the filter output. The resulting implementation is named as a *minimally redundant parallel implementation* (MRPI) and a filter that is implemented in MRPI is called a MRP filter (MRPF).

3.2 Problem definition

We want to compute filtering operation $\mathbf{C} \cdot x(n)$ where $\mathbf{C} = [c_0\ c_1 \cdots\ c_{M-1}]$ for an M-tap filter and target dedicated fully parallel implementations for high-speed applications. Hence, the goal is to find a sub-graph which covers all vertices minimizing the cost of implementation. Especially, a minimum spanning tree [2] is preferred solution due to smaller tree depth that is directly related to the filter delay.

For convenience of explanation, *a color ξ of an edge is defined as the value of $c_j - 2^L c_i$ for a particular value of L*. In a graph representation of M-tap filter, there are M vertices each of which has maximum cardinality of $2(W+1)(M-1)$ when $0 \le L \le W$. The solution requiring the smallest amount of resources to implement the product $\mathbf{C} \cdot x(n)$ consists of a set of colors whose edges visit all vertices at least once, such that the total cost of implementation of these colors is minimum. Suppose an edge color k is selected, then so are all edges with the same color in the graph. That is, the computation result of $k \cdot x(n)$ can

be reused everywhere requiring the color k. Moreover, all the shifted values of $k \cdot x(n)$ are also available without any additional computational cost. Hence, we define a class of such *k* and its shifts as a *color class*. The smallest element of a color class is referred to as a *primary color* and other elements are called *secondary colors*.

In addition, a *color set of a color* is defined as a set of vertices which can be visited by a primary color and its secondary colors. From color sets of colors, we can compute two properties of a color; *frequency* and *cost*. The size of color set is store in $frequency$ indicating the number of computation re-use requiring only one addition. *cost* of a color indicates the number of resources required for a product of $x(n)$ by the value of the color. Then, our goal is to find the lowest cost set of colors such that the edges in these sets cover all of the vertices in the graph. This is a well known NP-complete problem called *weighted minimum set cover* (WMSC) and it can be solved using a good heuristic approach [2, 4].

3.3 Decision strategy

In order to select the best color for a solution set, we introduce a *benefit function*, ***f***, which reflects the relative amount of complexity reduction by a color. The benefit function is defined as

$$\boldsymbol{f} = \beta \cdot frequency - (1-\beta) \cdot cost \qquad (0 \leq \beta \leq 1) \quad (1)$$

A color with smaller cost is preferred while higher value of $frequency$ is preferred. β can then be used to favor more vertex coverage versus lower cost of implementing the color. Hence, ***f*** reflects the amount of computations which can be saved. That is, the larger ***f*** is, the more preferred the color is. In addition, the value of β can be used to consider the effect of interconnect cost of the technology. In deep sub-micron technologies, it may be cheaper to compute more than to share more because of the drive requirement caused by computation re-use. This particular issue is modeled in the proposed approach to demonstrate that the decisions for choosing the final solution can be changed based on information about the technology — although we do not propose a solution to choose the actual value of β for a given technology. $\beta = 0.5$ indicates that interconnect cost is negligible and the benefit function equally weights the drive requirements stemming from high frequency of a color with the cost of implementing the color. $0 \leq \beta < 0.5$ gives more weight to interconnection cost while $0.5 < \beta \leq 1.0$ emphasizes choosing colors with less cost and high frequency.

3.4 Greedy approach

Now, we present the first part of the proposed algorithm which is referred to as *stage A* [4].

Step 1 *As a first step, the coefficients of the filter are normalized by the largest coefficient.*

Step 2 *We can divide coefficients into two sets: primary coefficients and secondary coefficients as we did for colors. All secondary coefficients are removed and only the primary coefficients with the smallest value are kept, since no computation is required to implement partial products involving the secondary coefficients.*

Step 3 *Build a graph (vertices and edges) with primary coefficients and remove all secondary colors from graph.*

Step 4 *Construct color sets and compute the cost, the frequency and the benefit function (equation (1)) for all colors.*

Step 5 *Solve the weighted minimum set cover problem.*

1. *Initialize minimum set cover (MSC)-Solution set to empty set*
2. ***while** (Color Sets are not empty) **do***
   - *(a) Select a color which has the largest value of benefit function and add the color to MSC-Solution set*
   - *(b) Remove all vertices covered by the selected color from Color Sets*
   - *(c) Update frequencies of all colors and recalculate the benefit function of each color*

Step 6 *Check if there are vertices having the same value with any color belonging to MSC-Solution set. If there is, all incoming edges to the vertex is eliminated, because no predecessor to the vertex is necessary.*

The algorithm which is described above decides a *MSC-Solution set* and generates spanning graph consisting of one or several disconnected graphs. For a fast and low-complexity implementation, spanning tree with minimum tree height should be identified from each disconnected graph. The *all pairs shortest path algorithm* [2] can be used to select a root vertex of a tree. The *all pairs shortest path* algorithm computes a matrix including shortest distances of all pairs of vertices. If two vertices belong to the same connected graph, the corresponding element of distance matrix includes the minimum distance between these two vertices. Otherwise, distance matrix has ∞ as an element. Hence, application of *all pairs shortest path algorithm* to the graphs defined by the stage A generates a sparse matrix shown in Figure 3(a). Each sub-matrix M_l corresponds to the distance matrix of each connected sub-graph.

Considering a sub-matrix M_l, each row, r_t, contains the shortest distance to the vertices belonging to the same connected graph. Let m_t be the maximum value of a row, r_t. Then, the value of m_t implies the maximum depth of leaf vertex (tree height) when vertex v_t is selected as a root. Hence, by comparing the values of m_t, root vertex can be determined by selecting a vertex having the smallest value of m_t. Once roots of trees are defined, the structures of spanning trees are easily determined by selecting minimum cost edges which connects predecessors and successors among WMSC-Solution color set from root to leaf vertices.

3.5 Example Implementation of MRPF

We present an example filter implementation by the proposed algorithm. Let us consider an asymmetric 8-tap FIR filter with coefficients $\mathbf{C} = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7\} = \{7, 66, 17, 9, 27, 41, 56, 11\}$. Because all the coefficients are primary, all coefficients are further processed. Figure 2 is the complete graph of the example filter showing some colors. From the graph, color set is constructed computing the frequency and cost of each color.

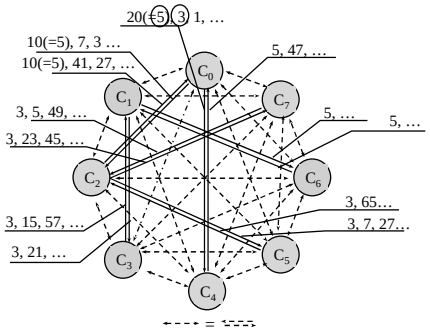
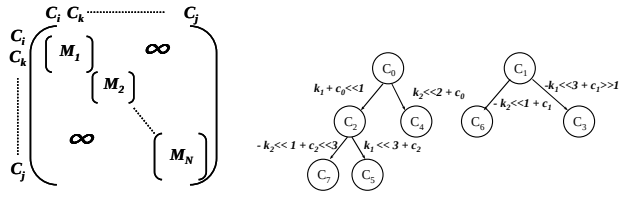


Figure 2. Complete directed graph

In this example, color 3 and 5 are the best candidates for solution set and it can be verified that all vertices of the graph are covered by colors 3 and 5 (Figure 2). The resulting sub-graph (two disconnected graphs) by the solution set, $\{3, 5\}$, is shown as solid arrows in Figure 2. Suppose c_0 and c_1 are determined as roots by applying all pairs shortest path algorithm to the sub-graph. Then, the resulting spanning trees can be easily determined by selecting a minimum cost edge which belongs to the solution set, $\mathbf{K} = \{k_1, k_2\} = \{3, 5\}$. Figure 3(b) shows the resulting spanning trees with tree height of two. For convenience of explanation, a *SEED* set is defined as an union of the solution color set and all roots of spanning trees. In case of the example filter, $SEED = \{c_0, c_1, k_1, k_2\} = \{7, 66, 3, 5\}$. The resulting implementation is shown in Figure 4.

The structure of the filter is composed of two parts. One is the *SEED multiplication network* and the other is *overhead add network*. SEED multiplication network performs



(a) Sparse matrix for MRPI sub-graph (b) Spanning tree of 8-tap example filter

Figure 3. Sparse matrix and Spanning trees

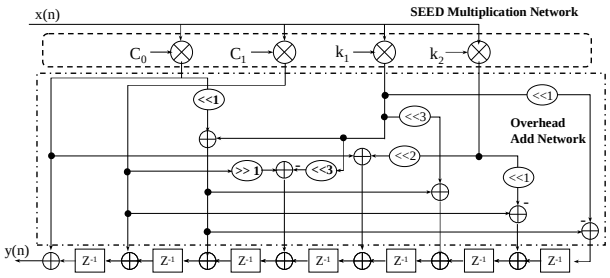


Figure 4. Filter structure of example filter

simple vector scaling operations which compute the multiplications of input data (vector) and each element (scalar) of SEED. Overhead network finishes the desired multiplications (multiplication of input data and coefficients) by shifting and adding the values computed in SEED multiplication network. To compute a vertex (coefficient) that does not belong to SEED, the predecessor of each vertex should be computed in advance. The computation order of vertices is already reflected in the structure of spanning trees. Hence, the overhead add network has to show the exact image of the spanning trees in Fig 3(b). The resulting implementation exploits full parallelism for higher performance and maximal computation sharing for low power dissipation.

4 MRPI as an Architectural Transformation

In previous section, we showed how the computational complexity can be reduced by manipulating arithmetic relation between given coefficients. The proposed technique extracts the most basic computations necessary from which the remaining computations can be obtained using a simple overhead add operation. This is as if the elements of SEED form a basis upon which the remaining computations can be found using very simple overhead add operations. It is interesting to note that the SEED multiplication network performs a vector scaling operation! Hence, we can recursively use the MRPI approach on the SEED network to break it into another SEED and overhead network. Similarly, any other complexity reduction approach can be applied to SEED multiplication network. For example, computation sharing multiplication algorithm [1] or *common sub-expression elimination* (CSE) [3, 9] can be candidates for such reduction. In this paper, we use CSE proposed by [3] to provide a logical optimization in preference to the arithmetic approach of MRPI to reduce the complexity of SEED network to reap the benefits of both approaches.

The block diagram of resulting filter structure by combining CSE and MRPI is shown in Figure 5. Application of CSE to SEED multiplication network results in two sub-blocks. One sub-block computes the multiplications between input data and sub-expressions and the other sub-block finishes multiplications required by SEED multiplication network whose size is a fraction of the size of the

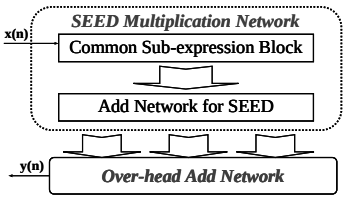


Figure 5. Filter structure of MRPI+CSE

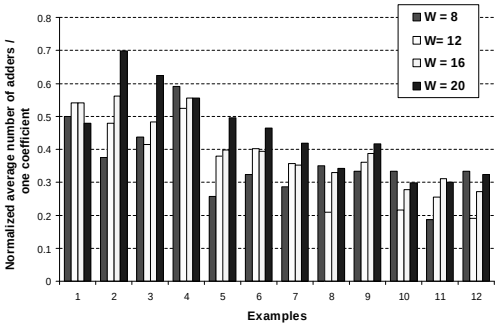


Figure 6. Uniformly scaling: MRPF vs Simple(SPT)

original filter (Table 1) in general.

We finally conclude this section by pointing out very important advantages of the MRPI to the brute-force CSE approach. First, much more control on the selection of the final solution set: the value of β can be used for skewing the solution and roughly modeling the effect of interconnects. Second, better architecture for pipelining: pipelining is desired to be used to increase the speed of operation. Unlike CSE (section 1), the proposed MRP approach nicely breaks the solution into several networks and provides a natural place to pipeline the filter. By slight modifications to the selection algorithm, we can change the size of SEED network to reduce the register overhead. The recursive application of MRP to SEED network extends the level of pipelining by providing an additional point where the registers can be inserted to further speed up the filter.

5 Numerical Results

Several example filters are configured to have symmetric property (Table 1), and the necessary resources for implementation are computed. We assume all filters are implemented in folded and transposed direct form [7]. The overhead add operations due to folding-over of symmetric filter is not considered because the overhead is identical for all cased irrespective of the type of implementation. Three different number representations are considered. Filters based on CSE approach [3] use CSD number representation. For MRPI, sign-magnitude (SM) and SPT number representation are considered. In addition, two different scaling techniques are considered; *uniformly scaled coefficients* and *maximally scaled coefficients* [5].

Figure 6 shows the comparison results of uniformly scaling case using SPT. Each data is normalized by the corresponding result of simple implementation. Even though the results using SM are not shown here, we observe that the efficiency of the proposed approach does not depend on the number representation of coefficients. It means that from any initial solution, MRP optimization technique can provide significant reduction of complexity. Especially for the filters with larger than 200 filter taps, only 0.3 adder is required for multiplication per tap when wordlength of coefficients is $16bit$. It means that add operations to compute FIR filter output dominate the power consumption rather than the multiplication. On average, 60% of reduction in complexity can be obtained from simple implementation when using uniformly scaled coefficients. Figure 7 shows the comparison results of maximally scaling case using SPT coefficients. Maximal scaling scheme increases the complexity significantly. Especially, wordlength of coefficients has strong relation to the implementation complexity. In case of 8 and 12 bit wordlength, 60% average reduction in complexity is obtained from simple implementation. However, for large wordlength like 16 or 20 bit, complexity reduction ratio is 40% when using SPT number representation on average.

Finally, Figure 8 shows the comparisons of CSE and combination of MRPI and CSE. When compared with simple implementation using SPT for coefficients, combination of logical (CSE) and arithmetic (MRP) optimization techniques maximizes computation sharing resulting in 66% and 74% average reduction in complexity for uniformly and maximally scaling, respectively. To show the efficiency of MRPI as an architectural transformation, all the simulation results are normalized with the corresponding results of CSE. The proposed approach shows 17% and 15% complexity reduction for uniformly scaling and maximally scaling respectively.

We clearly demonstrated that proper computation reordering methodology exploiting arithmetic equivalence maximizes the potential of computation sharing and is a ef-

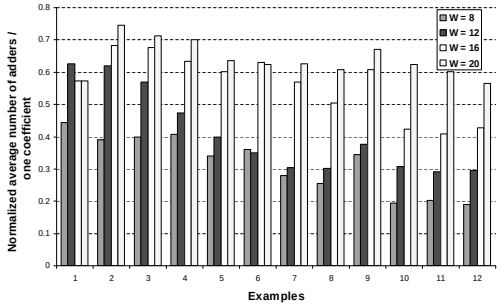


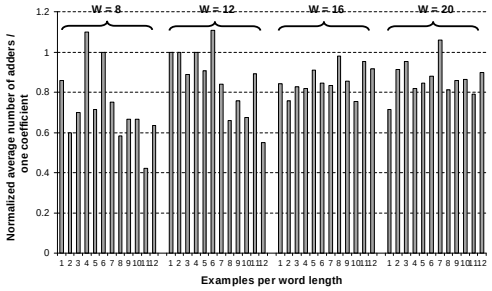
Figure 7. Maximal Scaling: MRPF vs Simple (SPT)

Example number	1 BW LP	2 PM LP	3 LS LP	4 BW LP	5 PM BS	6 LS BS	7 PM BS	8 PM LP	9 LS BS	10 LS LP	11 PM BP	12 LS BP
f_{p1}	0.15	0.15	0.15	0.5	0.2	0.2	0.3	0.4	0.3	0.4	0.3	0.3
f_{s1}	0.25	0.25	0.25	0.53	0.24	0.24	0.32	0.4175	0.32	0.4175	0.315	0.315
f_{p2}					0.42	0.42	0.7		0.7		0.705	0.705
f_{s2}					0.38	0.38	0.68		0.68		0.69	0.69
R_p(dB)	2	2	2	3	3	3	2	3	2	3	1	1
R_s(dB)	60	60	60	65	50	50	50	70	50	70	70	70
Filter order	20	35	60	80	98	126	164	214	232	260	301	350
Size of SEED with SPT (roots,solution set)	9 (3,6)	18 (6,12)	30 (9,21)	26 (9,17)	41 (17,26)	53 (26,27)	63 (29,34)	62 (29,33)	46 (22,24)	40 (24,26)	75 (27,48)	80 (35,45)
Size of SEED with SM (roots,solution set)	9 (3,9)	17 (6,11)	30 (7,23)	27 (10,17)	46 (18,28)	46 (14,32)	44 (9,35)	63 (32,31)	47 (21,26)	77 (21,56)	79 (29,50)	61 (25,36)

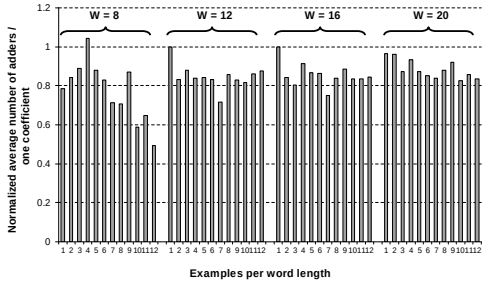
• BW: Butterworth, PM: Parks−Mclellan, LS: Least square, LP: Low Pass, BP: Band Pass, BS: Band Stop(Notch)

• Size of SEED indicates the number of elements belonging to SEED using 16 bit maximally scaled coefficients under depth constraint of 3

Table 1. Filter specs of example filters and the size of SEED set after MRP Transformation



(a) Uniformly scaling



(b) Maximally scaling

Figure 8. MRPF+CSE vs CSE(SPT)

fective approach in reducing the computational complexity for high-performance and low-power design.

6 Conclusion

We presented a complexity reduction technique which can be used to obtain multiplierless implementations of FIR digital filters. The main idea is to *reorder* computations and to maximize computation sharing between different multipliers. The reordering problem is mapped to graph and the multiplierless solution is obtained by solving a set cover problem on the graph. A simple algorithm based on a greedy approach was presented. The proposed schemes can be used together with other existing complexity reduction approaches to further simplify the design. When combined with CSE, we demonstrated a 70% and 16% reduction in complexity from simple implementation and CSE on average, respectively.

References

[1] H. Choo, K. Muhammad, and K. Roy. Decision feedback equalizer with two's complement computation sharing multiplication. In *IEEE int. Conf. Acoustics, Speech and Signal Processing*, volume 2, pages 1245–1248, 2001.

[2] T. H. C. et. al. *Introduction to algorithms*. The MIT Press and McGraw-Hill Book Company, 1998.

[3] R. I. Hartley. Subexpression sharing in filters using canonic signed digital multipliers. *IEEE Trans. Circuits and Systems II*, 43(10):677–688, 1996.

[4] K. Muhammad and K. Roy. Minimally redundant parallel implementation of digital filters and vector scaling. In *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, volume 6, pages 3295–3298, 2000.

[5] K. Muhammad and K. Roy. A graph theoretic approach for synthesizing very low-complexity high-speed digital filters. *IEEE trans. Computer-Aided Design of Integrated Circuits and Systems*, 21(2), Feb 2002.

[6] K. Muhammad. *Algorithmic and Architectural Techniques for Low Power Digital Signal Processing*. PhD thesis, Purdue University, 1999.

[7] K. K. Parhi. *VLSI digital signal processing systems: Design and Implementation*. Wiley-Interscience, 1999.

[8] I.-C. Park and H.-J. Kang. Digital filter synthesis based on minimal signed digit representation. In *Proc. Design Automation Conference*, pages 468–473, 2001.

[9] M. Potkonjak, M. B. Srivastava, and A. P. Chandrakasan. Multiple constant multiplications: Efficient and versatile framework and algorithms for exploring common subexpression elimination. *IEEE Tran. Computer-Aided Design of Integrated Circuits and Systmes*, 15(2):151-165, Feb 1996.

[10] S. Ramprasad and I. N. H. N. R. Shanbhag. Decorrelating (decor) transformations for low-power digital filters. *IEEE Trans. Circuits and Systems-II*, 46(6):776–788, Jun 1999.

[11] H. Samueli. An improved search algorithm for the design of multiplierless fir filters with powers-of-two coefficients. *IEEE Trans. Circuits and Systems*, 36(7):1044–1047, Jul 1989.