Minimizing the adder cost in multiple constant multipliers

S. Rahimian Omam[1a)], S. M. Fakhraie[1], and O. Shoaei[1]
[1] IC Design Center, University of Tehran,
16 Azar Street, Enghelab Ave., Tehran, Iran
a) s.rahimian@ece.ut.ac.ir

Abstract: The hardware complexity of digital filters is mainly dominated by the coefficient multipliers. Implementing fixed-point coefficient multiplication as a network of adders, subtractors, and shifters, yields lower power consumption. In such filters the number of adders (and subtractors) determines the implementation cost. The reason is that shifts are implemented as hard-wired inter-block connections and are considered "free". In transposed implementation of an FIR filter, each input is multiplied by several coefficients. Considering all the coefficients as a multiplier block and omitting the redundancies by sharing the common fundamentals among different coefficients, yields great reduction in the number of arithmetic operations. This paper presents a graph based algorithm to reduce the computational complexity of multiple constant multiplications. Simulation results show that using the proposed method results good improvement in adder cost of multiplier blocks.

Keywords: digital filter, adder cost, graph representation
Classification: Integrated circuits

References

[1] A. G. Dempster and M. D. Macleod, "Using all signed-digit representations to design single integer multipliers using subexpression elimination," *ISCAS 2004*, vol. 3, pp. III–165–168.
[2] D. R. Bull and D. H. Horrocks, "Primitive operator digital filter," *IEEE Proc. G.*, vol. 138, no. 3, pp. 401–412, June 1991.
[3] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc. Circuits Devices Syst.*, vol. 141, no. 5, pp. 407–413, Oct. 1994.
[4] A. G. Dempster and M. D. Macleod, "Use of minimum-adder multiplier blocks in FIR digital filters," *IEEE Trans. Circuits Syst.*, vol. 42, no. 9, pp. 569–577, Sept. 1995.

DOI: 10.1587/elex.3.340
Received May 21, 2006
Accepted June 15, 2006
Published July 25, 2006

1 Introduction

Using an efficient number system to represent fixed-point coefficients can make a significant reduction in computational complexity of a digital filter.

CSD (canonic signed digit) representation requires the fewest adders in comparison to all other alternatives (since the hardware cost is similar for adders and subtractors, both of them are referred as adders). On average CSD representation uses 33% fewer nonzero digits than binary representation [1].

Graph representation can be used to further reduce hardware complexity. The concept of graph representation of multiplier block was first introduced by Bull and Horrocks in [2]. In this representation, each graph vertex shows an adder with two inputs and each edge represents a multiplication by a power of 2 which can be implemented as a binary shift.

The algorithm presented in [2] minimizes the number of adders in a multiplier block referred in this work as 'BH' algorithm. This algorithm has been modified in [3] and is denoted in this work as 'BHM'. The algorithm in [3] uses more graph topologies than CSD representation and despite the improvement, it is not optimum. To illustrate the point, consider the integer 75 as a coefficient. The CDS representation and the algorithm in [3] each produces a cost of 3 as shown in Fig. 1 (a) and Fig. 1 (b) respectively, whereas the optimum cost is 2 as shown in Fig. 1 (c).

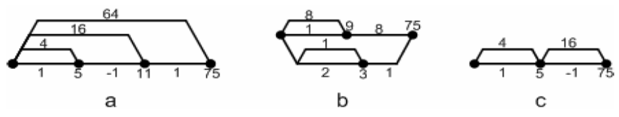

Fig. 1. Different representations of 75 as a coefficient. a) CSD, b) BHM, and c) the optimum representation.

BHM contributes to an adder cost saving of 26.6% compared to CSD representation for 32 bit coefficients [3].

Another graph-based method named MAG (minimum adder graph) is represented in [3] to minimize the adder cost of a single coefficient. The key to reducing the cost below that of CSD is the use of more diverse graph topologies than CSD. To find the minimum adder cost for a particular integer coefficient, all possible graph topologies have been considered in [3]. They propose some rules for graph generation by using graph theory and generate all graph topologies for different numbers of adders used to implement nonzero bits of different coefficients. It is also proven in [3] that it is sufficient to implement only odd integers and use shifts to produce even integers. Consequently, the algorithm considers only odd fundamentals (the graphs with odd vertex values). Using these graph topologies, a look-up table is produced containing all coefficients which can be implemented with 1, 2, ..., N adders. To implement a coefficient, one can search the look-up-table to find the odd fundamentals (i.e. the odd number produced by successively dividing by two)

© IEICE 2006
DOI: 10.1587/elex.3.340
Received May 21, 2006
Accepted June 15, 2006
Published July 25, 2006

of the coefficients with minimum adder cost and implement each coefficient as proposed by the chosen graph structure. Ref. [4] nominates RAG (reduced adder graph) algorithm, using the look-up table in [3] to minimize the adder cost of a multiplier block. This algorithm consists of two parts, optimum search and heuristic result completion. Our proposed algorithm utilizes the optimum search part of RAG and proposes a new method for the heuristic part.

In the transposed realization of FIR filters, the input is first multiplied by all coefficients and then the results go through the registers. This allows us to better consider all of the coefficient multipliers with the same input as a single multiplier block and further optimize it. The power consumption of a transposed FIR filter can be reduced by considering multiplier block and employing fewer adders to implement the multipliers.

2 Graph generation

In graph representation, the input vertex shows the odd values that don't need any adder to implement (a cost 0 value). In MAG method, the input vertex indicates to 1 for all graph topologies. It means that the input value of all graphs is equal to the input sequence. Since 1 is the unique odd amount with the adder cost of 0, there is no other way in single multiplier implementation. But it is not the same in multiplier block implementation. After implying a coefficient, the coefficient and its intermediate nodes can be used as the input vertex of the next graph, because their adder costs have been calculated for pervious coefficients and now they are available to synthesize a new fundamental without any extra cost. Thus, the graph topologies may have unequal input values. The input vertexes can be equal to 1 or any prior implemented value (coefficients or their intermediate nodes).

There are some rules illustrated in [3] to generate the graph topologies. We add an extra rule which states that "the output degree of input vertexes must be equal to 1 to avoid redundant topologies". Using these rules and considering different values for input vertex, all possible graph topologies are generated. In Fig. 2, the graphs for adder cost of 1 to 4 are presented. These graphs are the base of our proposed algorithm. When the inputs are equal, the first structure in each adder cost shows the CSD representation. These structures cover all topologies shown in [3] and consist of more general cases.

There are some theories in [3] about using the graph topologies which result in the following features. It is proved that following these features can lead us to achieve the optimum structure faster.

(I) Only odd fundamentals are examined (i.e. only OF graphs are considered).

(II) The cost of an even number is the same as that of its odd fundamental.

(III) When examining higher cost graphs, only sums of two odds and sums of an odd and an even are considered.

(IV) Negative numbers are not examined explicitly, because the cost of a negative integer is the same as its positive counterpart; there needs to be one

© IEICE 2006
DOI: 10.1587/elex.3.340
Received May 21, 2006
Accepted June 15, 2006
Published July 25, 2006

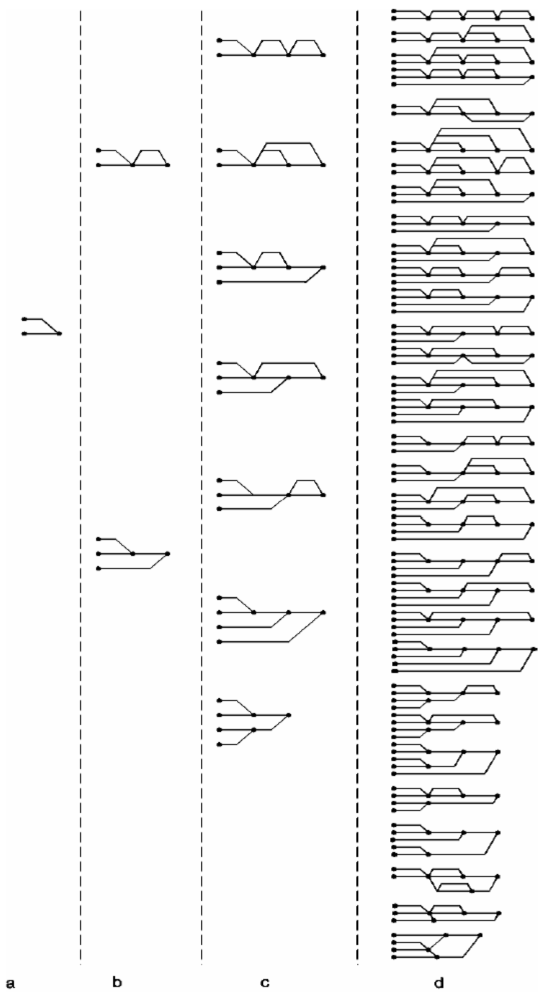

Fig. 2. Possible graph topologies with different inputs for cost 1 to 4.

negated edge in each path from the initial vertex to the terminating vertex to change the sign of the output. These features are used in our proposed algorithm. According to the above features, for all even coefficients and even intermediate nodes, their odd fundamentals are substituted for the original values.

3 Proposed algorithm

This algorithm is based on using the implemented integers as the input vertex of a graph to implement a new coefficient. It checks different combinations of realized nodes to generate a new coefficient with lower adder cost. At the beginning, the only input value is one but after synthesizing the first coefficient, this coefficient and its intermediate nodes can be used as the input vertex too. At first it tries to implement the coefficients with minimum adders (1 adder) and then checks the pair-wise combination of synthesized nodes to find a new coefficient. If there is any coefficient that can't be synthesized during these steps, it tries to implement one of the unsynthesized coefficients with 2 adders (using Fig. 1(b) topologies). In this step it tries to combine two or three synthesized value to implement a new coefficient. After finding

© IEICE 2006
DOI: 10.1587/elex.3.340
Received May 21, 2006
Accepted June 15, 2006
Published July 25, 2006

the first coefficient synthesizable with Fig. 1 (b) topologies, it goes to the beginning and checks the pair-wise combination of the new nodes and other implemented nodes. The algorithm repeats these steps till no other coefficient is synthesized. Then it uses Fig. 1 (c) topologies which use three adders and the algorithm continues.

In RAG algorithm the pair-wise combination of synthesized fundamentals are checked, but because they don't have all topologies for more complex combinations (e.g. combination of three fundamentals), they use the above mentioned look-up-table to implement the next coefficient. It can ignore some redundancies and requires more adders.

For fixed-point filter coefficients with least significant bits of 2^{-B}, scaling by 2^B produces integer coefficients.

The algorithm can be expressed as follows:

1. Change the negative coefficients to positive. Reduce all the coefficients to odd fundamentals. Store the results in the 'incomplete set' (the set of coefficients not yet synthesized). In this level all the coefficients with adder cost equal to 0 (powers of 2), reduce to 1. Remove 1 from the incomplete set and add it to S which indicates the set of synthesized values.

2. Using Fig. 2 (a), synthesize all the coefficients which can be implemented by one adder. Use the members of S as the input vertexes. According to Fig. 2 the cost 1 integers can be generated as

$$c = s(i) \pm 2^k \cdot s(j) \qquad \forall s(i), s(j) \in S, \quad i, j, k \in N \tag{1}$$

   When $s(i) = s(j) = 1$ it is reduced to cost 1 structure in [3].

   Remove the implemented coefficient from incomplete-set and put it in S (if even, add the odd fundamental). Repeat this step until no more fundamentals are added to S.

3. Using the members of S as the input values, search among the Fig. 1 (b) topologies to find a synthesizable coefficient. For example, the first topology shown in Fig. 2 (b), can generate the following numbers.

$$c = \left[s(i) \pm 2^k \cdot s(j)\right] \cdot \left(1 \pm 2^l\right) \qquad \forall s(i), s(j) \in S, \quad i, j, k, l \in N \tag{2}$$

   Generate the numbers using these topologies till finding one of the incomplete set numbers.

   Remove the coefficient from the 'incomplete set' and add the coefficient and its intermediate node to S (if each of them are even, add the odd fundamental instead). After implementing each coefficient in this step, go to step 2. If no coefficient can be implemented using these topologies, go to step 4.

© IEICE 2006
DOI: 10.1587/elex.3.340
Received May 21, 2006
Accepted June 15, 2006
Published July 25, 2006

4. Repeat the step 3 for Fig. 2 (c).
5. Repeat the step 4 for Fig. 2 (d).

Theoretically, there are some coefficient sets that cannot be synthesized during this algorithm and they need five adders or more to be implemented. However, this case is less probable in this work than in RAG algorithm. In our simulations, the maximum adder cost for 22 bit coefficients is 3. RAG checks the combinations shown in Fig. 2 (a) and some special cases of Fig. 2(b). Because of more combinations covered by the proposed algorithm in comparison to RAG, it gives better results for a multiplier block. Furthermore, RAG needs the look-up table in [3] to implement the coefficients but the proposed method doesn't use it. This algorithm is considered as a modification of RAG and is called MRAG (modified RAG)

4 Case study

As a case study, we have used two FIR filters. The first is a sharp half-band FIR filter with order of 147 which has 37 different 22-bit coefficients. The second filter is an FIR filter with order of 25 and 13 different 20-bit coefficients.

These filters are realized in transposed structure. The multiplier blocks are realized using CSD, RAG, and our proposed method (MRAG). Coefficients of these filters and Simulation results are shown in Table I.

Table I. Number of adders used to implement the multiplier blocks and filter coefficients

		Filter #1	Filter #2
Order		147	25
No. of coefficients		37	13
Word length		22	20
Cost	CSD	146	52
	RAG	77	30
	MRAG	58	24

Filter #1 coeffs			Filter #2 coeffs
F(1)=29	F(25)=4704	F(49)=59685	F(1)=83
F(3)=-53	F(27)=-6119	F(51)=-71089	F(2)=149
F(5)=101	F(29)=7860	F(53)=84611	F(3)=579
F(7)=-176	F(31)=-9982	F(55)=-100805	F(4)=1223
F(9)=288	F(33)=12545	F(57)=120482	F(5)=-2074
F(11)=-450	F(35)=-15617	F(59)=-144886	F(6)=-5412
F(13)=678	F(37)=19276	F(61)=176031	F(7)=4910
F(15)=-989	F(39)=-23609	F(63)=-217404	F(8)=17303
F(17)=1405	F(41)=28718	F(65)=275612	F(9)=-8105
F(19)=-1953	F(43)=-34719	F(67)=-364847	F(10)=-47858
F(21)=2662	F(45)=41757	F(69)=522055	F(11)=5854
F(23)=-3567	F(47)=-50004	F(71)=-882824	F(12)=165964
		F(73)=2667757	F(13)=259980
F(n)=F(148-n)	F(2n)=0		F(n)=F(26-n)

The first half-band filter shows 60% and 24.7% improvement in the adder cost in comparison to CSD and RAG, respectively. For the second filter, the improvements are 53.8% and 20% in comparison to CSD and RAG, respectively.

5 Conclusion

This paper generates graph topologies with different inputs. A new algorithm is proposed which checks all possible combinations of realized values to find

© IEICE 2006
DOI: 10.1587/elex.3.340
Received May 21, 2006
Accepted June 15, 2006
Published July 25, 2006

a new coefficient. This algorithm tries to omit the redundancies to reduce the adder cost of the multiplier block.

In our simulations, the maximum adder cost for each coefficient is three and approximately most coefficients are implemented using one or two adders. It is commonly known that RAG has the best performance among all other methods proposed before this work. It is shown that the proposed method (MRAG) has a lower adder cost in comparison to RAG especially in high order filters and large fixed-point coefficients. Another interesting feature of our proposed method is that it does not need a look-up table, as generating the previously-used look-up tables for adder costs of more than 3 is difficult because of long simulation time.

© IEICE 2006
DOI: 10.1587/elex.3.340
Received May 21, 2006
Accepted June 15, 2006
Published July 25, 2006