Digital Filter Synthesis Considering Multiple Adder Graphs for a Coefficient

Jeong-Ho Han, and In-Cheol Park
School of EECS, Korea Advanced Institute of Science and Technology, Daejeon, Korea
jhhan.kr@gmail.com, icpark@ee.kaist.ac.kr

***Abstract*—In this paper, a new FIR digital filter synthesis algorithm is proposed to consider multiple adder graphs for a coefficient. The proposed algorithm selects an adder graph that can be maximally sharable with the remaining coefficients, while previous dependence-graph algorithms consider only one adder graph when implementing a coefficient. In addition, we propose an addition reordering technique to reduce the computational overhead of finding multiple adder graphs. By using the proposed technique, multiple adder graphs are efficiently generated from a seed adder graph obtained by using previous dependence-graph algorithms. Experimental results show that the proposed algorithm reduces the hardware cost of FIR filters by 23% and 3.4% on average compared to the Hartely and RAGn-hybrid algorithms.**

I. Introduction

In many digital signal processing systems, finite impulse response (FIR) digital filters are frequently used because of their stability and linear phase property. Though the FIR filters based on digital processing cores can take an advantage of flexibility, they are not suitable for recent applications demanding real-time performance and low power consumption. For these applications, dedicated digital FIR filter blocks are used to meet the constraints of performance and power consumption. Since the FIR filter consists of many constant multiplications, earlier works have decomposed the multiplications into simple operations such as addition, subtraction and shift and have tried to share the partial sums as many as possible in order to reduce the hardware complexity. As shown in Fig. 1, all coefficients are considered as a whole to design all the constant multiplications into a hardware block called a multiplier block. There are two metrics that are important in the design and comparison of digital FIR filters. The first one is the adder cost, which is the number of adders required to implement a given set of filter coefficients, and the second is the adder step that is the number of adders passing through the critical path. Many algorithms have been proposed to reduce the adder cost and adder step, which can be categorized into two groups, dependence-graph and common subexpression elimination (CSE) algorithms.

The objective of CSE algorithms is to find bit patterns that are common in more-than-one coefficients to reduce the total adder cost by sharing the common bit patterns among the coefficients. To reduce the number of nonzero bits or to maximize the sharing of common bit patterns, the coefficients are converted to a specific number representation such as the signed digit (SD) number system [3] or canonic signed digit (CSD) number system [4]-[7], or minimal signed digit (MSD) number system [8] before searching common bit patterns.

The goal of dependence-graph algorithms is to find an adder graph consisting of the minimal number of partial sum nodes. The total adder cost can be reduced by sharing the previous partial sums generated for a subset of constant multiplications in deriving additional constant multiplications. The relationship among partial sums can be graphically represented as shown in Fig. 2, where a node is associated with the value of the corresponding partial sum. The number on each edge means the shift amount to be applied to the input partial sum, and the minus symbol indicates that the input partial sum is negated before adding it at the output node. For the sake of simplicity, the shift amount of zero is not explicitly shown on the edge. Note that a partial sum can be connected to multiple nodes, meaning that the partial sum is shared for generating the nodes. A new partial sum can be obtained by applying addition/ subtraction/ shift operations to the previous partial sums. The first dependence-graph algorithm introduced by Bull and Horrocks in [1], hereinafter called the BH algorithm, synthesizes filter coefficients one by one in a numerical order. For a selected coefficient, the BH algorithm generates an adder graph by inserting new partial sums required to minimize the error to the coefficient. A modified version, called the BHM algorithm [2], improves three limitations of the BH algorithm and reduces the adder cost by 10%. The n-dimensional reduced adder graph (RAGn) algorithm [2] introduces the concept of adder distance. The adder distance of a coefficient is the number of adders additionally required to achieve the coefficient value from a given adder graph. In general, the RAGn shows better performance than other dependence-graph and CSE algorithms. However, a pre-calculated lookup table is used to find the adder distance in the RAGn, and it induces a limitation on the range of coefficient values. To overcome this limitation, the RAGn-hybrid algorithm [2] replaces the heuristic part requiring the pre-calculated lookup table with the BHM algorithm. Recently, a graph-theoretical approach called SIDC [10] is proposed to consider the differences among the coefficients. The differences are first synthesized and then the coefficients are synthesized based on the differences, but partial sums are not shared between the two steps. For the applications in which the logic delay is more crucial than the logic complexity, the adder step [9] and the wire delay [11] are taken into account as additional metrics.

The previous dependence-graph algorithms consider one coefficient at a time and do not take into account the effect on the rest coefficients when synthesizing the coefficient. Though the adder graph generated in such a way is the best

978-1-4244-2658-4/08/$25.00 ©2008 IEEE

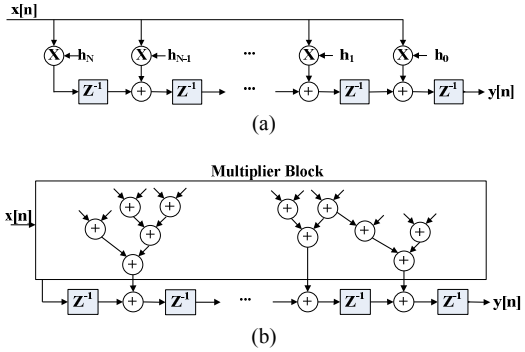

Fig. 1. FIR filter structure. (a) Transposed form. (b) FIR filter using a dedicated multiplier block.

solution for the coefficient, it may not be the best if the remaining coefficients are considered. In this paper, we propose a new dependence-graph algorithm to minimize the adder cost by considering the effect on the remaining coefficients. Considering multiple adder graphs for a coefficient, the proposed algorithm selects an adder graph which can be shared maximally with the remaining coefficients. To reduce the computation overhead, a simple addition reordering technique is applied to generate multiple candidate graphs from a seed adder graph.

II. TERMINOLOGIES AND NOTATIONS

This section introduces some notations that will be used to describe the proposed algorithm.

- ***C***: A set of coefficients to be synthesized. Each coefficient is uniquely represented by an odd positive number, and $|\boldsymbol{C}|$ represents the number of coefficients in ***C***.
- ***S***: A set of partial sums which are generated so far. Starting from {1}, the set grows as coefficients are synthesized one by one. A partial sum that is in the current partial sum set is called a cost-0 partial sum as it can be directly used to synthesize other coefficients.
- ***G***: A set of adder graphs generated for a coefficient. This set is generated by applying the proposed addition reordering to a seed adder tree.
- ***T***: A table of partial sums. Initially this table contains the information on the shift amounts and signs of the cost-0 partial sums involved in a seed adder graph. $|\boldsymbol{T}|$ represents the number of entries in ***T***, t_i denotes the i-th entry, and p_i denotes the partial sum value in t_i.

III. PROPOSED ALGORITHM

Since synthesizing an adder graph for a set of filter coefficients is NP-complete [1], previous dependence-graph algorithms such as the BHM and RAGn rely on heuristic approaches. Considering one coefficient at a time, the heuristic algorithms minimize the additional adder cost required to incorporate the coefficient into the current partial sum set. The adder graph generated for a coefficient may not be the best if we consider the whole coefficient set, because the remaining coefficients are not considered in generating the adder graph. If we consider multiple candidate adder graphs and then select an adder graph that can be shared maximally with the remaining coefficients, the total adder cost can be reduced further.

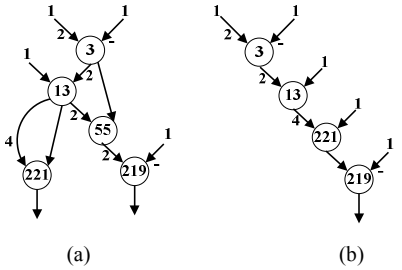

Fig. 2. Graph representations of a filter coefficient set {3, 13, 219, 221} generated by (a) the BHM algorithm, and (b) the RAGn algorithm.

Fig. 3 illustrates an example of the adder graph generation process for a coefficient set {3, 53, 585}. There are two coefficients, 53 and 585, to be synthesized, and the initial condition is presented in Fig. 3-(a) where three adder graphs are possible for 53. The previous dependence-graph algorithm generates an adder graph by inserting new partial sums which minimize the difference to a coefficient and does not consider the remaining coefficients when synthesizing the coefficient as shown in Fig 3-(b). In the example, 53 is synthesized before 583, and only a graph (graph_1) is taken into account for 53. The partial sums generated for 53 cannot be shareable with 585 and the total adder cost becomes 6. Fig. 3-(c) shows the proposed adder graph generation process that considers the remaining coefficients. In this case, the three graphs are considered for 53. To select a proper graph from the multiple graphs, the proposed algorithm generates a temporary partial sum set for each graph by tentatively inserting the partial sums contained in the graph into the current partial sum set, and then synthesizes the remaining coefficients by applying the conventional dependence-graph algorithm such as the BHM to the temporary set. After examining the total adder cost of each graph, the proposed algorithm selects a proper graph (graph_2 in this case) that minimizes the total adder cost. The partial sums of graph_2 are inserted to the current partial sum set. In this case, 585 can be synthesized by using one additional adder and the total adder cost becomes 4. As observed in this example, the total adder cost can be reduced if we consider the sharing effect on the remaining coefficients when synthesizing a coefficient.

A. Generating multiple adder graphs

The computation complexity to find all the possible adder graphs is high even for a coefficient. To reduce the computation overhead, an efficient method is proposed in this subsection to generate multiple adder graphs for a coefficient. In the proposed method, a seed adder graph is first generated for a coefficient to be synthesized by applying one of the previous dependence-graph algorithms, and then multiple adder graphs are generated by the proposed addition reordering technique that changes the order of additions in the seed adder graph. Fig.4 illustrates that two additional adder graphs, (b)

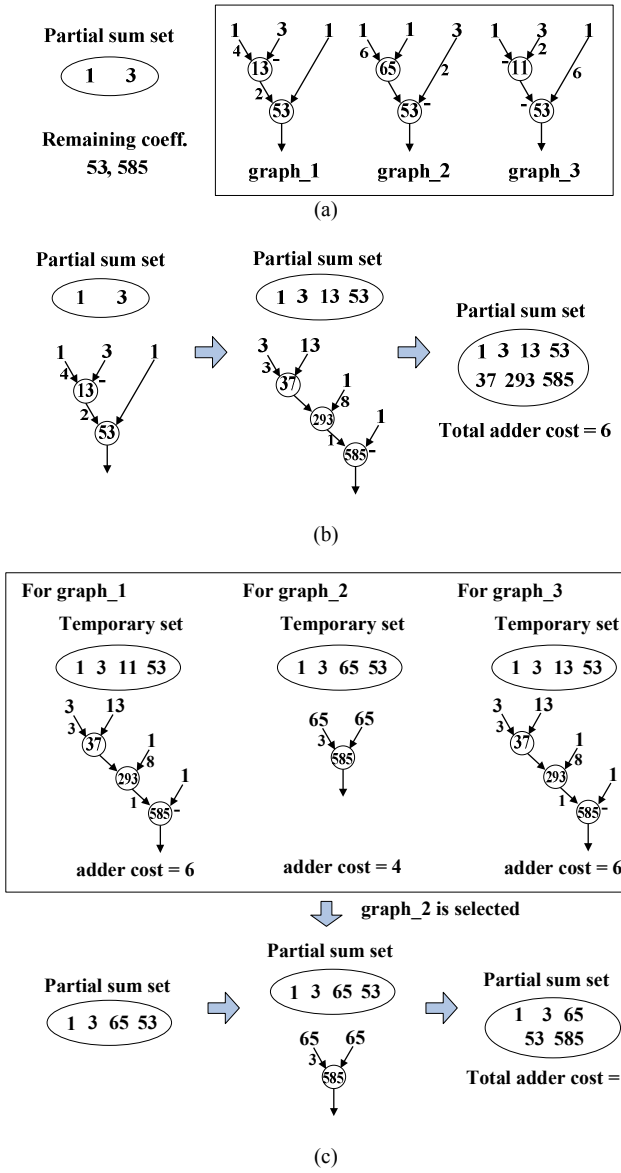

Fig. 3. Synthesis procedure for a coefficient set {3, 53, 585}. (a) Initial condition and three possible adder graphs for 53. (b) The previous dependence-graph algorithms (BHM and RAGn-hybrid). (c) The proposed algorithm that considers three adder graphs for 53.

and (c), are generated from a seed adder graph (a) generated by the BHM algorithm. The proposed addition reordering is different from the computation reordering [10] used to express the filter with the differences of coefficients. In the sense that the differences are computed earlier, the computation is reordered in [10].

Before applying the addition reordering, the seed adder graph is expanded into cost-0 partial sums that are already in the current partial sum set. Fig. 5 illustrates how to expand the

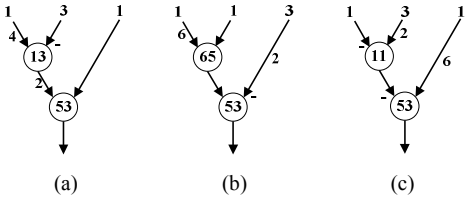

Fig. 4. Two adder graphs (b) and (c) generated by applying the addition reordering to (a).

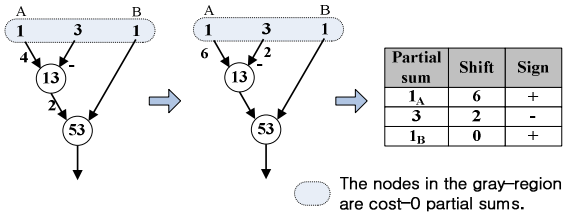

Fig. 5. Expanding a seed adder graph and making a table of cost-0 partial sums. From the root node of 53, the shift amount and the sign of a node are propagated to the upper nodes.

seed adder graph. Note that the seed graph can be constructed by using a cost-0 partial sum more than one times. Let us assume that the seed graph is a tree. In other words, the number of edges coming out from a node is always one in the seed adder graph. If there is a node violating this assumption, the seed graph can be transformed into a tree by duplicating such a node. In this case, we can move the shift amount and the sign of an outgoing edge to the input edges, as there is only one outgoing edge in a tree. Starting from the root node of 53, the shift amount and the sign of the output edge are propagated to each input edge, i.e., the shift amount and the sign are added to and multiplied by those of each input edge, respectively. This propagation is recursively applied until it reaches to the cost-0 partial sum nodes. Through the adder graph expanding procedure, we make a partial sum table $\boldsymbol{T}$ that stores the shift amount and the sign of each cost-0 partial sum node contained in the seed adder tree.

To generate an adder graph, we select two partial sums, t_i, and t_j, from $\boldsymbol{T}$, and make a new entry t_{new} by adding the shifted values of p_i, and p_j. Then, we update $\boldsymbol{T}$ by removing t_i and t_j and inserting t_{new}. To maximize the sharing of partial sums, the value of p_{new} is restricted to a positive odd number by adjusting its shift amount and sign value when inserting the new partial sum into $\boldsymbol{T}$. This procedure is recursively applied until only a partial sum remains in the table. Following the table update sequence, we generate an adder graph that results in the target coefficient. Fig. 6 illustrates an example, where the first and third entries are selected first to make a new partial sum of 65 by performing (1<<6) + (1<<0) and then the two entries in the updated table are combined to derive a new partial sum of 53 by performing (65<<0) – (3 <<2).

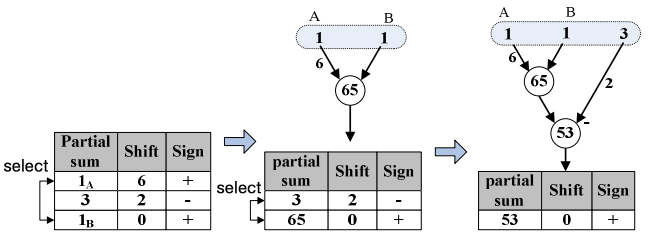

Fig. 6. Generating an adder graph from the partial sum table.

Multiple adder graphs can be obtained by changing the combination of two partial sums to be selected from $\boldsymbol{T}$. The total number of adder graphs, N, that can be generated from a seed adder graph is expressed in Eq. (1), where n is $|\boldsymbol{T}|$. As indicated in Eq. (1), the number of adder graphs grows exponentially as n increases. For example, the number of possible adder graphs is 3 for n=3, 18 for n=4, and 180 for n=5. To reduce the computation overhead, multiple adder graphs are considered only when n is less than 5 in this paper. This restriction will be justified later in Section IV.

$$N = \prod_{i=0}^{n-3} {}_{n-i}C_2 \ , \ (n \geq 3) \\ = \frac{n!}{2!(n-2)!} \times \frac{(n-1)!}{2!(n-3)!} \times \cdots \times \frac{3!}{2!1!} = \frac{n! \times (n-1)!}{2^{n-1}} \quad (1)$$

The procedure to generate multiple adder graphs is summarized below, and a pseudo code of the major function called *Build_graphs* is shown in Fig. 7, where *new_psum* (t_i, t_j) generates a new entry in $\boldsymbol{T}$ by adding the shifted values of partial sums in t_i and t_j, *insert_node* (p_i, g) enlarges the adder tree g by inserting p_i, and *refine_graph*(g) combines a number of nodes that have the same partial sums into a node so as to convert g to an adder graph. The new partial sum is restricted to an odd positive number by adjusting the shift amount and the sign value. In the proposed algorithm, a set of adder graphs $\boldsymbol{G}$ is used to store adder graphs found by applying the proposed addition reordering to a seed adder graph.

Step 1) Initialize $\boldsymbol{G}$ to ø.
Step 2) Transform the seed adder graph g to an adder tree.
Step 3) Make a partial sum table $\boldsymbol{T}$ from the adder tree by using the expanding procedure.
Step 4) If $|\boldsymbol{T}| > 4$ or $|\boldsymbol{T}| = 2$, then $\boldsymbol{G} = \{g\}$ and finish.
Step 5) To generate multiple adder graphs, call function *Build_graphs* ($\boldsymbol{T}$, g_{empty}), where g_{empty} is an empty tree.

B. Description of the proposed algorithm

The proposed algorithm is divided into two parts. The first part is to select a coefficient to be synthesized from $\boldsymbol{C}$, the set of coefficients not synthesized yet. We select a coefficient that is associated with the lowest adder distance. In the second part, multiple adder graphs are generated for the selected coefficient by using the proposed addition reordering and an adder graph is selected that seems to be maximally shared with the remaining coefficients. Then the partial sums in the selected adder graph are inserted into the partial sum set. The proposed algorithm is as follows:

```
Build_graphs (T, g) {
 If (|T| == 2) {
  t_new = new_psum (t1, t2).  (where, t1, t2 ∈ T)
  g_new = insert_node (p_new, g)
  g_cand = refine_graph (g_new)
  G = G ∪ {g_cand} }
 else {
  for (i=1 .. |T|-1) {
   for (j=i+1 .. |T|) {
    t_new = new_psum (t_i, t_j).  (where, t_i, t_j ∈ T)
    g_new = insert_node (p_new, g)
    T_tmp = T − t_i − t_j + t_new
    Build_graphs (T_tmp , g_new) } } } }
```

Fig. 7. Generating multiple adder graphs based on the addition reordering.

Step 1) Prepare for starting.
1-1) Make a coefficient set $\boldsymbol{C}$ from the given filter coefficients.
1-2) Sort the coefficients in $\boldsymbol{C}$ in the increasing order of the number of non-zero bits in the CSD representation.
Step 2) Select a coefficient to be synthesized.
2-1) For each $c \in \boldsymbol{C}$, find an adder graph by using the BHM algorithm.
2-2) Select a coefficient c_{sel} associated with the minimum adder distance, and remove c_{sel} from $\boldsymbol{C}$.
Step 3) For c_{sel}, make a graph set $\boldsymbol{G_{cand}}$ that contains multiple adder graphs generated by the addition reordering.
Step 4) Select a proper adder graph.
4-1) For each $g_i \in \boldsymbol{G_{cand}}$,
4-1-1) Make a temporary partial sum set $\boldsymbol{S_i}$ by merging $\boldsymbol{S}$ and the partial sums contained in g_i.
4-1-2) Synthesize all the remaining coefficients in $\boldsymbol{C}$ by using the BHM algorithm with regarding $\boldsymbol{S_i}$ as the current partial sum set.
4-2) Select an adder graph g_j that minimizes the total adder cost.
Step 5) Update $\boldsymbol{S}$ by merging $\boldsymbol{S_j}$ to the current $\boldsymbol{S}$.
Step 6) Remove the synthesized coefficients from $\boldsymbol{C}$, that is, $\boldsymbol{C} = \boldsymbol{C} - (\boldsymbol{C} \cap \boldsymbol{S})$.
Step 7) If $|\boldsymbol{C}| > 0$, go to Step 2.

If the logic delay of synthesized filter is a major concern, the BHM algorithm employed in Step 2 and 4 can be replaced with another algorithm that can control the adder step, such as SLBHM [9]. The step-limited version of the proposed algorithm is as follows:

Step 2) Select a coefficient to be synthesized.
2-1) For each $c \in \boldsymbol{C}$, find an adder graph by using the SLBHM algorithm.
2-2) Select a coefficient c_{sel} associated with the minimum adder distance, and remove c_{sel} from $\boldsymbol{C}$.
Step 3) Generate multiple adder graphs

3-1) For c_{sel}, make a graph set $\boldsymbol{G_{cand}}$ that contains multiple adder graphs generated by the addition reordering.

3-2) For each $g_i \in \boldsymbol{G_{cand}}$, check the adder step of g_i and remove it from $\boldsymbol{G_{cand}}$ if it has longer adder step than the adder step constraints.

Step 4) Select a proper adder graph.

4-1) For each $g_i \in \boldsymbol{G_{cand}}$,

4-1-1) Make a temporary partial sum set $\boldsymbol{S_i}$ by merging $\boldsymbol{S}$ and the partial sums contained in g_i.

4-1-2) Synthesize all the remaining coefficients in $\boldsymbol{C}$ by using the SLBHM algorithm with regarding $\boldsymbol{S_i}$ as the current partial sum set.

4-2) Select an adder graph g_j that minimizes the total adder cost.

IV. Experimental Results

To compare with previous dependence-graph algorithms such as Hartely[4], BHM[2] and RAGn-hybrid[2], the proposed algorithm is applied to seven FIR filters. The RAGn-hybrid is used in the comparison, because the RAGn has limitation on the coefficient range. Table 1 shows the filter specifications generated by the Remez algorithm in MATLAB, where the last row represents the word-length of coefficients represented in a binary number system. Synthesis results are summarized in Table 2, where we can see that the proposed algorithm outperforms other algorithms in terms of the adder cost, while maintaining the adder step similar to that of the RAGn-hybrid. Compared to Hartely, the proposed algorithm reduces the adder cost by 23% on the average, while the others reduce by from 18% to 20%. As indicated in [10], the SIDC algorithm improves by 11% compared to Hartely.

The operating frequency of the synthesized FIR filter is important in some applications demanding high performance. In these applications, it is needed to restrict the adder step. Such a filter synthesis algorithm called the step-limited BHM has been presented in [9]. Table 3 shows the synthesis results obtained with an adder step constraint of 4. The step-limited version of the proposed algorithm shows better performance than the Hartely and the SLBHM, but the adder cost improvement is reduced compared to the case that does not consider the adder step constraint. To see the tendency of adder cost reduction, more experiments have been performed with changing the adder step constraint. The reduction degrades as the adder step constraint becomes tight, because the number of adder graphs satisfying the adder step constraint reduces.

In the proposed algorithm, multiple adder graphs are considered only when the number of partial sums is less than 5. To justify the restriction, Table 4 shows the distribution of the number of partial sums obtained by expanding seed adder graphs. The percentage that the number of partial sums is greater than 4 is less than 0.62%, meaning that the restriction has a negligible effect on the final synthesis results. We synthesized a number of filters without resorting to the restriction, but there were almost no improvements compared to the results obtained with the restriction. If the restriction is not taken, the computation overhead of the proposed algorithm can grow exponentially as the number of coefficients increases. The restriction plays an important role in reducing the computation overhead dramatically.

Though we restrict the maximum number of adder graphs for a coefficient, the total number of adder graphs considered during synthesis can grow exponentially, if all coefficients are synthesized by considering multiple adder graphs. However, practically, the computation overhead does not grow exponentially as the number of coefficient increases. Fig. 8 shows the average computation times for various filter orders, which reveals that the practical computation time of the proposed algorithm is linearly proportional to the filter order, not exponentially. In practice, many coefficients are synthesized with only one adder, after a coefficient is synthesized with considering multiple adder graphs. The number of coefficients synthesized with only an adder graph grows as the number of filter order increases.

For rigorous comparison, additional experiments were performed for 417 FIR filters on a computer system equipped with a 3GHz Pentium processor and 1GB memory. The filter order ranges from 60 to 300, and the wordlength of coefficients is at least 16 bits. Compared to the BHM, the proposed algorithm improves 321 cases, while the RAGn-hybrid improves 226 cases. Note that there are no degrading cases. On the average, the proposed algorithm reduces the adder cost by

Table 1. Test Filter Specifications

Filter	1	2	3	4	5	6	7
Passband	0.1	0.1	0.1	0.1	0.1	0.18	0.18
Stopband	0.14	0.14	0.14	0.14	0.14	0.19	0.19
# tap	60	120	180	120	180	120	120
Width	16	18	18	20	20	18	20

Table 2. Synthesis Results for the Test Filters

Filter	Hartley[5]		BHM[2]		RAGn-hybrid[2]		Proposed	
	Adder cost	Adder step	Adder cost	Adder step	Adder cost	Adder step	Adder cost	Adder step
1	49	4	42	6	42	6	37	7
2	82	4	64	7	63	9	63	7
3	92	4	76	8	74	8	74	8
4	107	4	81	8	81	8	77	9
5	114	4	95	9	93	10	91	9
6	86	4	68	7	65	9	64	10
7	106	4	88	10	88	10	84	10

Table 3. Synthesis Results for the Test Filters With an Adder Step Constraint of 4

Filter	Hartley[5]		SLBHM[15]		Step-limited Proposed	
	Adder cost	Adder step	Adder cost	Adder step	Adder cost	Adder step
1	49	4	45	4	41	4
2	82	4	69	4	67	4
3	92	4	81	4	80	4
4	107	4	100	4	93	4
5	114	4	108	4	104	4
6	86	4	76	4	72	4
7	106	4	100	4	97	4

Table 4. Distribution of the Number of Partial Sums

Number of partial sums	2	3	4	5	6	7	8
Occurrence (%)	84.2	13.49	1.69	0.37	0.11	0.1	0.03

Table 5. Synthesis Results for 417 Filters Compared to BHM

	RAGn-hybrid[2]	Proposed
Number of improved cases	226	321
Number of worse cases	0	0
Maximum number of reduced adders (%)	12 (13.89%)	14 (15.91%)
Average number of reduced adders (%)	1.43 (1.67%)	3.03 (3.38%)

3.38% including not-improved cases, while the RAGn-hybrid reduces by 1.67%. For some filters, the proposed algorithm reduces the adder cost by 15.91%.

V. Conclusion

We have presented a new filter synthesis algorithm that minimizes the adder cost. The proposed algorithm considers multiple adder graphs for a coefficient to be synthesized in order to take into count the effect on the remaining coefficients, whereas previous dependence-graph algorithms consider only one adder graph when implementing a coefficient. An adder graph that can be shared maximally with the remaining coefficients is selected among multiple adder graphs. Starting from a seed adder graph, we can derive multiple adder graphs by applying the proposed addition reordering technique. To reduce the computational complexity of considering multiple adder graphs, the proposed method is applied when the number of partial sums of the seed adder graph is less than 5, as the other cases are very rare. Experimental results show that the proposed algorithm outperforms the previous dependence-graph algorithms such as the BHM and RAGn-hybrid.

Acknowledgment

This research was supported by Korean Intellectual Property Office.

References

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

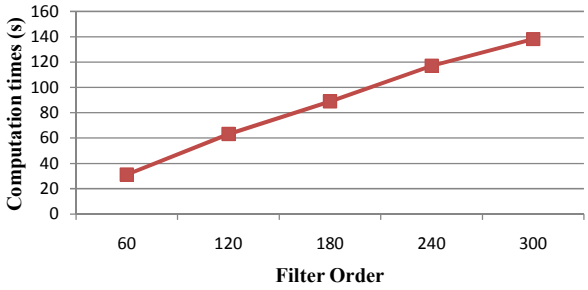

Fig. 8. Average computation times for 417 filters

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

[3] Y. C. Lim, J. B. Evans and B. Liu, "Decomposition of binary integers into signed power-of-two terms, " *IEEE Trans. Circuits Syst.*, vol. 38, pp. 667-672, June 1991.

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

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

[6] H. Samueli, "An improved search algorithm for the design of multiplier-less FIR filters with powers-or-two coefficients," *IEEE Trans. Circuits Syst.*, vol. CAS-36, pp. 1044-1047, July 1989.

[7] A. P. Vinod and E.M.-K. Lai, "On the Implementation of Efficient Channel Filters for Wideband Receivers by Optimizing Common Subexpression Elimination Methods," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 24, no. 2, Feb. 2005.

[8] In-Cheol Park and Hyeong-Ju Kang, "Digital filter synthesis based on an algorithm to generate all minimal signed digit representations," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 12, no. 12, pp. 1525-1529, Dec. 2002.

[9] Hyeong-Ju Kang and In-Cheol Park, "FIR Filter Synthesis Algorithms for Minimizing the Delay and the Number of Adders," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 8, pp.770-777, Aug. 2001.

[10] Hunsoo Choo, Khurram Muhammad and Kaushik Roy, "Complexity Reduction of Digital Filters Using Shift Inclusive Differential Coefficients," *IEEE Trans. Signal Proces*., vol. 52, no. 6, pp.1760-1772, June 2004.

[11] Dongku Kang, Hunsoo Choo, Khurram Muhammad and Kaushik Roy, "Layout-Driven Architecture Synthesis for High-Speed Digital Filters," *IEEE Trans. Veri Large Scale Integration Syst.,* vol. 14, no. 2, Feb. 2006.

[12] Chia-Yu Yao, Hsin-Horng Chen, Tsuan-Fan Lin, Chiang-Ju Chien and Chun-Te Hsu, "A Novel Common-Subexpression-Elimination Method for Synthesizing Fixed-Point FIR Filters," *IEEE Trans. Circuits Syst. II*, vol.51, Nov. 2004.