An Area Optimization Method for Digital Filter Design

Sang-Hun Yoon, Jong-Wha Chong, and Chi-Ho Lin

In this paper, we propose an efficient design method for area optimization in a digital filter. The conventional methods to reduce the number of adders in a filter have the problem of a long critical path delay caused by the deep logic depth of the filter due to adder sharing. Furthermore, there is such a disadvantage that they use the transposed direct form (TDF) filter which needs more registers than those of the direct form (DF) filter. In this paper, we present a hybrid structure of a TDF and DF based on the flattened coefficients method so that it can reduce the number of flip-flops and full-adders without additional critical path delay. We also propose a resource sharing method and sharing-pattern searching algorithm to reduce the number of adders without deepening the logic depth. Simulation results show that the proposed structure can save the number of adders and registers by 22 and 26%, respectively, compared to the best one used in the past.

Keywords: Digital filter, flattened coefficient, adder sharing, architecture.

I. Introduction

The multiplication of a variable by a set of constants is a central operation in video processing, digital television, data transmission, and wireless communications. The area-time optimization of this operation has often been accomplished by using a shift-and-add multiplication algorithm, combined with techniques to reduce the number of nonzero bits in the binary representation of the coefficients. For example, a signed-digit code was efficiently applied in [1], [2].

Generally, a finite impulse response filter is separated into two kinds of structures. One is the direct form (DF) and the other is the transposed direct form (TDF). The DF filter requires the adder to have as many operands as the number of filter taps, and it gives rise to very long propagation delays. On the other hand, a TDF structure forms a short critical path, but it has a drawback in that it needs many registers. A DF structure is suitable for the implementation on a digital signal processor which has a multiplication and accumulation function, while the TDF is mainly applied to an application specific integrated circuit for high speed processing.

This multiplication part of the TDF filter can be simplified as it is a multiplication between the identical input and the fixed coefficients. In order to simplify this part, many researches such as a multiplier block [3], [4] and common sub expression [5], [6] have been pursued. In the field of digital filter design, while there are a lot of research efforts to reduce the number of adders, few efforts have been taken to reduce the number of registers. Only a truncation or round-off has been used, but each of these methods inevitably causes a round-off error.

The purpose of this paper is to design a new digital filter that uses the flattened-coefficients method to keep the critical path delay similar to the TDF filter and reduces the filter area by minimizing the number of both adders and registers without a round-off error.

Manuscript received Feb. 25, 2004.

This work was supported by the research fund of Hanyang University (HY-2003-T).

Sang-Hun Yoon (phone: +82 2 2290 0558, email: shyoon11@ihanyang.ac.kr) and Jong-Wha Chong (email: jchong@hanyang.ac.kr) are with the Department of Electronic Engineering, Hanyang University, Seoul, Korea.

Chi-Ho Lin (email: ich410@semyung.ac.kr) is with the Department of Computer Science, Semyung University, Jecheon, Korea.

The remainder of this paper is organized as follows. Section II explains the flattened coefficients method. Sections III to V describe the finite impulse response filter design method of the proposed structure. Section V shows a pattern searching algorithm. Section VI shows simulation results that compare the number of full-adders and flip-flops with the conventional method. Section VII contains the conclusion of this paper.

II. Flattened-Coefficients Method

In general, the logic depth of the critical path in a filter is determined by the maximum number of non-zero terms (NZTs) of each coefficient. The 'maximum' number can be changed into the 'average' number using the flattened coefficients method [7], which makes it possible to improve the characteristics of a filter while maintaining the critical path delay. The number of flip-flops can be reduced in the middle of the TDF and DF using the flattened coefficients method.

Figure 1 shows an example where the numbers of NZTs are 1, 5, and 2, and the critical path is formed along the arrow. At this moment, if it is necessary to shorten the critical path length, two kinds of methods can be considered. One, described in Fig. 2, makes a pipeline structure by inserting registers into the wanted level, while another, shown in Fig. 3, reconstructs it in order to ignore some adders on the critical path.

However, these methods have some drawbacks such that either additional registers are required or the characteristics of the filter worsen. Also, since these methods basically use the TDF, they demand more flip-flops than in the DF.

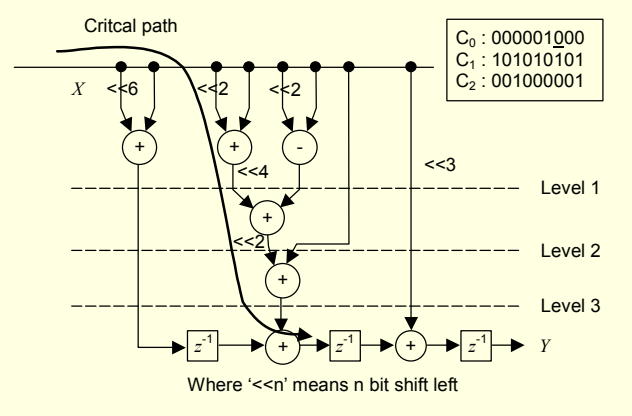

Fig. 1. Critical path of a filter.

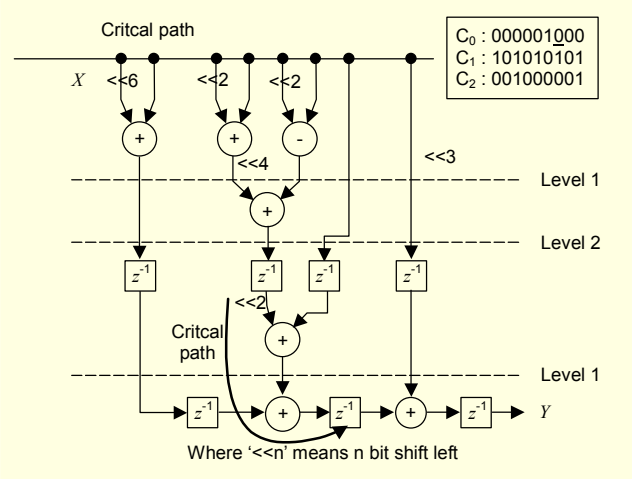

Fig. 2. Pipeline structure of the filter.

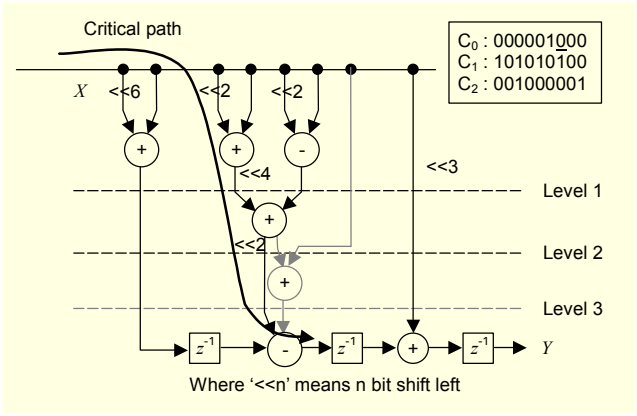

Fig. 3. Deleted NZT structure of the filter.

The flattened-coefficients method can solve the above problems using the following theorems and definitions.

In general, a digital filer can be represented as

$$Y_n = \sum_{i=0}^{L-1} X_{n-1} C_i \,, \tag{1}$$

where Y_n is the *n*-th output, X_n is the *n*-th input and C_i is the *i*-th coefficient. Because the hardware uses only values stored in the registers and inputs received at that time, a relative position in the input sequence is more important than an absolute position. Therefore, the definition below is necessary to implement (1) to hardware.

Definition:

X^0 : input value at time n, which is equal to X_n

X^{-k}: input value delayed k times by register

Y : filter output value at time n, which is equal to Y_n

AZ^{-k}: some value 'A' delayed k times by accumulation register, where k is an integer

By the definition, the following theorems are valid.

Theorem 1.

$$X^{-n} = X^0 Z^{-n}, \quad \text{where n is an integer.}$$

Proof. We proceed by induction on n. If n = 0, then the result

is obvious. In this case, both values in either side of '=' represent the current input which is not delayed by the register at all. Assume then that $X^{-(n-1)} = X^0 Z^{-(n-1)}$. The TDF and DF filter outputs can be represented as (2) and (3), respectively, by definition.

$$\begin{aligned} Y &= (((X^0 C_{n-1} Z^{-1} + X^0 C_{n-2}) Z^{-1} + \cdots) Z^{-1} + X^0 C_1) Z^{-1} + X^0 C_0 \\ &= X^0 C_{n-1} Z^{-(n-1)} + X^0 C_{n-2} Z^{-(n-2)} + \cdots + X^0 C_1 Z^{-1} + X^0 C_0. \end{aligned} \tag{2}$$

$$Y = X^0 C_0 + X^{-1} C_1 + \cdots + X^{-(n-2)} C_{n-2} + X^{-(n-1)} C_{n-1}. \tag{3}$$

From our definition, $X^0, X^{-1}, \ldots, X^{-(n-1)}$ can be derived as

$$\begin{aligned} X^0 &= X^0, \\ X^{-1} &= X^0 Z^{-1}, \\ &\vdots \\ X^{-(n-2)} &= X^0 Z^{-(n-2)}, \\ X^{-(n-1)} &= X^0 Z^{-(n-1)} \end{aligned} \tag{4}$$

If the number of filter taps is increased by 1, (2) and (3) can be modified to (5) and (6).

$$\begin{aligned} Y &= (((X^0 C_n Z^{-1} + X^0 C_{n-1}) Z^{-1} + \cdots) Z^{-1} + X^0 C_1) Z^{-1} + X^0 C_0 \\ &= X^0 C_n Z^{-n} + X^0 C_{n-1} Z^{-(n-1)} + \cdots + X^0 C_1 Z^{-1} + X^0 C_0. \end{aligned} \tag{5}$$

$$Y = X^0 C_0 + X^{-1} C_1 + \cdots + X^{-(n-1)} C_{n-1} + X^{-n} C_n. \tag{6}$$

With (4) through (6), $X^{-n} = X^0 Z^{-n}$ can be derived because filter outputs from the TDF and DF filters are identical. □

Theorem 2.

$X^{-n} = X^{-k} Z^{-(n-k)}$ where n and k are integers and $n \geq k$.

Proof. Theorem 2 is a general form of theorem 1. As $X^{-k} = X^0 Z^{-k}$ is satisfied by theorem 1, theorem 2 can then be represented as $X^{-n} = X^0 Z^{-k} Z^{-(n-k)}$. Since AZ^{-k} is the representation of the value A that is delayed k times through the register for accumulation, $X^0 Z^{-k} Z^{-(n-k)}$ indicates that X^0 is delayed by (n-k) times again after being delayed k times. The fact that $X^0 Z^{-k} Z^{-(n-k)}$ is equal to $X^0 Z^{-n}$ means X^0 is delayed n times through a register. The description above can be arranged as in (7).

$$\begin{aligned} X^{-n} &= X^{-k} Z^{-(n-k)} \\ &= X^0 Z^{-k} Z^{-(n-k)} = X^0 (Z^{-k} Z^{-(n-k)}) \\ &= X^0 Z^{-n}. \end{aligned} \tag{7}$$

□

If theorems 1 and 2 are applied to Fig. 1, (8) can be derived.

$$\begin{aligned} Y &= (2^3) X^0 + (2^8 + 2^6 + 2^4 - 2^2 + 2^0) X^{-1} + (2^6 + 2^0) X^{-2} \ldots \\ &= (2^3 X^0 + 2^8 X^{-1} + 2^6 X^{-1} + 2^4 X^{-1}) \\ &\quad + (-2^2 X^{-1} + 2^0 X^{-1} + 2^6 X^{-2} + 2^0 X^{-2}) \\ &= (2^3 X^0 + 2^8 X^{-1} + 2^6 X^{-1} + 2^4 X^{-1}) \\ &\quad - (2^2 X^0 - 2^0 X^0 - 2^6 X^{-1} - 2^0 X^{-1}) Z^{-1}. \end{aligned} \tag{8}$$

Figure 4 shows the filter implemented by (8). In Fig. 4, the logic depth is changed from level 3 to 2 which means a reduction of the critical path delay. Figure 5 shows a comparison between the conventional TDF filter, seen in Fig. 1, and the modified filter using the flattened coefficients method shown in Fig. 4. As depicted in Fig. 5, the register for storing the intermediate accumulation value, defined by the register of accumulation (RA), is changed to the one for delaying the input value, input register (IR). Also, the accumulation adder (AA) is changed to the adder for the summation of delayed inputs (AI). Since the RA stores an intermediate accumulation value of the multiplication between the coefficients and inputs, it requires longer register bits than the IR does. Furthermore, the AA has more bit lengths than the AI does. This is because the AA adds large-bit multiplication values between the coefficients and inputs, while the AI adds two small-bit inputs. Therefore, if a filter is changed by the flattened coefficients method, hardware complexity can be reduced because the number of both full-adders and flip-flops are decreased.

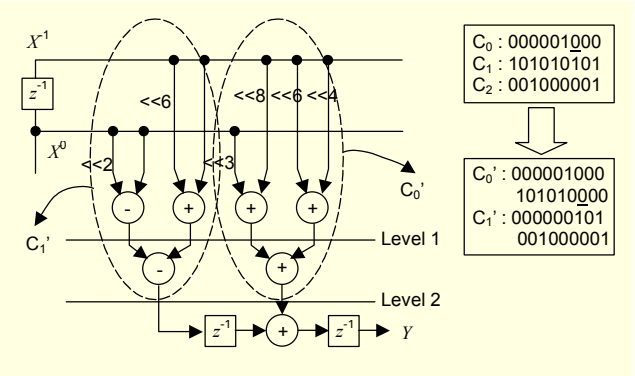

Fig. 4. Modified filter using the flattened coefficients method.

III. Pattern Searching for Adder Sharing

1. Adder Sharing

It is important that the number of adders and registers should be decreased to reduce the hardware area of a filter. In the previous section, the method to reduce the number of full-adders and flip-flops is explained. This section introduces an algorithm to reduce the number of adders by an adder sharing method.

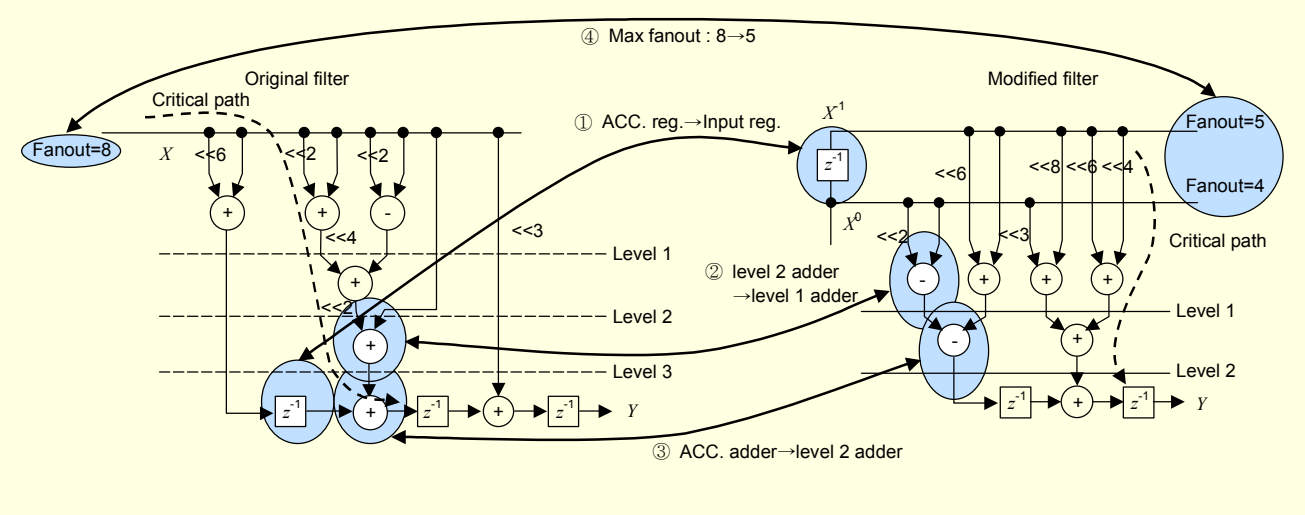

Fig. 5. Comparison between Figs. 1 and 4.

Since the conventional TDF filter has a characteristic where the current input multiplies with each fixed coefficient, the multiplier block method [3], [4] can be used for adder sharing. However such adder-sharing methods can not be applied to the flattened coefficients method because the conventional filter takes the same input for the multiplication with each coefficient while the flattened coefficients method uses differently delayed inputs. Therefore, a new adder sharing structure is necessary for the flattened coefficients method.

For example, we assume that the coefficients are converted into a canonical signed digit form by quantization, as in (9). Equation (10) shows that every four NZTs of the coefficient are recombined and rearranged in order that the deepest logic depth becomes two adder levels. Equation (11) can be derived by factorization.

$$\begin{aligned} C_0 &= -2^3 + 2^1, \\ C_1 &= -2^2 + 2^0, \\ C_2 &= -2^4 + 2^2, \\ C_3 &= -2^3 + 2^1, \end{aligned} \tag{9}$$

$$\begin{aligned} Y &= -2^3 X^0 + 2^1 X^0 + -2^2 X^{-1} + 2^0 X^{-1} \\ &\quad - 2^4 X^{-2} + 2^2 X^{-2} - 2^3 X^{-3} + 2^1 X^{-3} \\ &= (-2^3 X^0 + 2^1 X^0 + -2^2 X^{-1} + 2^0 X^{-1}) \\ &\quad + (-2^4 X^0 + 2^2 X^0 - 2^3 X^{-1} + 2^1 X^{-1}) Z^{-2}. \end{aligned} \tag{10}$$

$$\begin{aligned} Y &= (-2^3 X^0 + 2^1 X^0 - 2^2 X^{-1} + 2^0 X^{-1}) \\ &\quad + (-2^4 X^0 + 2^2 X^0 - 2^3 X^{-1} + 2^1 X^{-1}) Z^{-2} \\ &= (-2^3 X^0 + 2^1 X^0 - 2^2 X^{-1} + 2^0 X^{-1})(1 + 2^1 Z^{-2}) \\ &= (-2^2 (2^1 X^0 + 2^0 X^{-1}) + (2^1 X^0 + 2^0 X^{-1}))(1 + 2^1 Z^{-2}) \\ &= (2^1 X^0 + 2^0 X^{-1})(1 - 2^2)(1 + 2^1 Z^{-2}). \end{aligned} \tag{11}$$

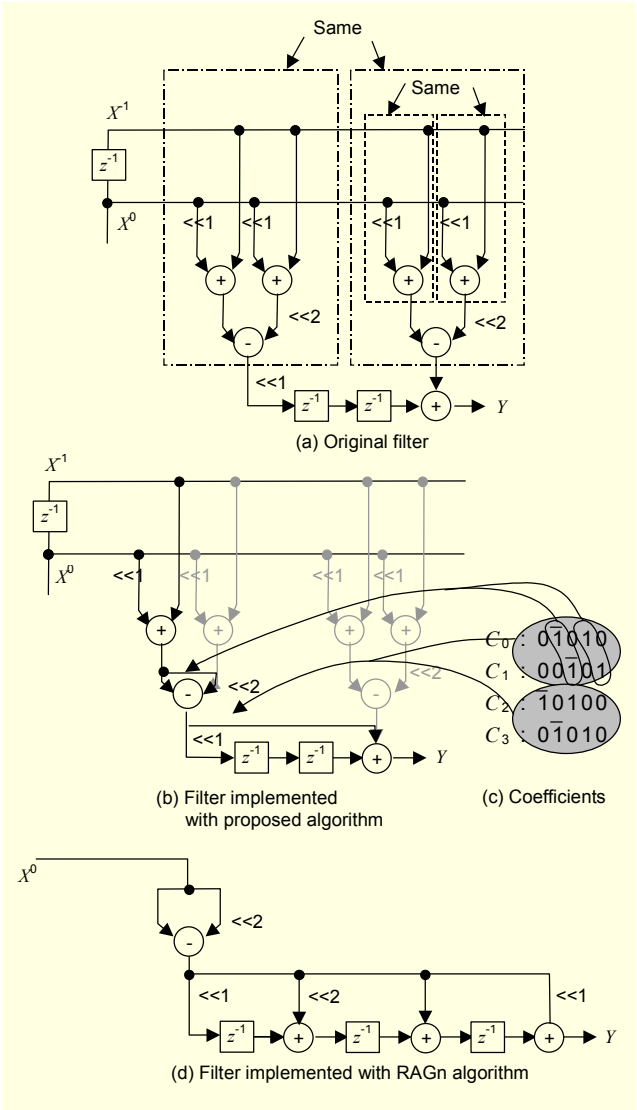

Fig. 6. An example of an adder-shared structure.

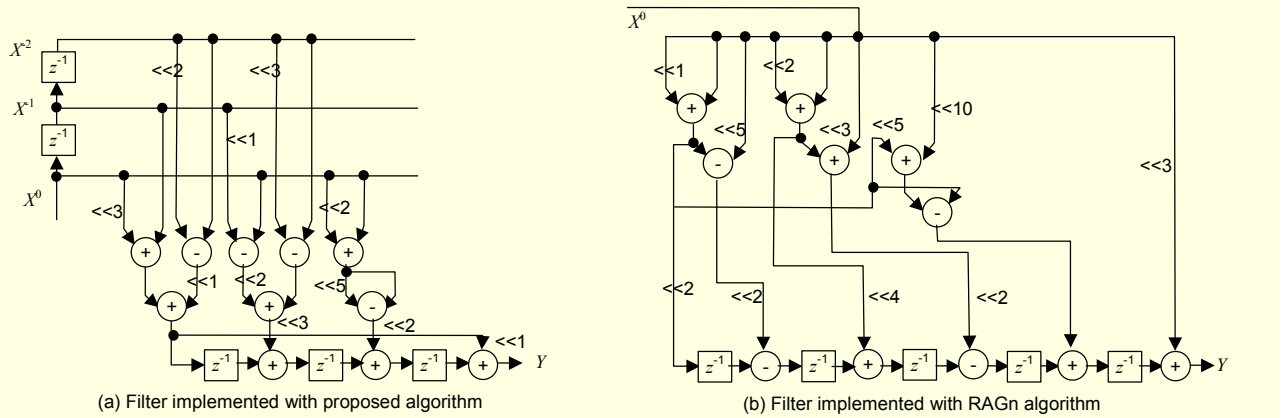

Fig. 7. A second example of an adder-shared structure.

Figure 6(a) can be modified to Fig. 6(b) by (11). Figure 6(c) shows the coefficients represented by CSD format. In Fig. 6(c), the small circles ranging from C_0 to C_1 represent two NZT pairs which can share one adder, as in Fig. 6(b), and the pattern shapes of the two NZT pairs are the same. In the same manner, two big circles on C_0, C_1, C_2 and C_3 show that the shared part of the three adders and pattern shapes of the NZT in each circle are the same. The searching for NZTs that can share adders can be accomplished using the pattern searching of the same shape in the coefficients matrix, as shown in Fig. 6(c) as well as in the factorization of (11). If the RAGn algorithm [3] is used to construct this example, the structure in Fig. 6(d) can be made. In this case, because two accumulation adders are reduced, the proposed adder sharing method can save one more adder than a conventional one, although one more adder for the multiplier block is needed. Adders can be shared in adder levels 1 and 2. We define adder sharing types I, II, III as the cases when adders can be shared in level 1 and level 2 at the same time, only in level 2, and only in level 1, respectively.

For another example, let's assume that the coefficients are calculated as

$$\begin{aligned} C_0 &: 000000001000 && (8_{10}) \\ C_1 &: 010010\bar{1}00\bar{1}01 && (1117_{10}) \\ C_2 &: 000000\bar{1}010\bar{1}0 && (-26_{10}) \\ C_3 &: 000001010000 && (80_{10}) \\ C_4 &: 000001001\bar{1}010 && (58_{10}) \\ C_5 &: 000000010\bar{1}00 && (12_{10}) \end{aligned} \qquad (12)$$

The proposed architecture and the best structure in terms of adder cost and logic depth among RAGn [3], BHM [4] , SLBHM [8], and C1 algorithm [9] for (12) are presented in Figs. 7(a) and 7(b), respectively. This example shows that the proposed algorithm can reduce the logic depth of the filter as well as the number of maximum fanouts.

IV. Positioning of the Shared Adders

In the previous sections, we described a method to share hardware resources. In this subsection, the methods for positioning the multiplication results between the inputs and coefficients are discussed. The positioning is defined as the process of controlling n and k in Theorem 2. It is a matter of course, but it can be overlooked. For example,

$$Y = (aX^{-1} + bX^{-2} + cX^{-3} + dX^{-4})Z^{-5} \qquad (13)$$

$$= (aX^{-2} + bX^{-3} + cX^{-4} + dX^{-5})Z^{-4} \qquad (14)$$

$$= (aX^{-5} + bX^{-6} + cX^{-7} + dX^{-8})Z^{-1} \qquad (15)$$

Here, (13) shows that the inputs delayed one, two, three, and four times are summed up, and the result is then delayed five times again. It provides an equal result with the summation of six, seven, eight, and nine times the delayed values. Also, the results of (14) and (15) are the same as the summation of six, seven, eight, and nine times the delayed values. Figure 8 shows the filter architecture implemented by (14) and (15).

One thing that has to be carefully controlled here is that the distance among the shared adders must be maintained. Figure 9(a) shows the case where the distance among the shared adders is 2. This distance has to be kept even if the accumulation position on the filter is changed as in Fig. 9(b).

The shaded circle in Fig. 10 illustrates an overlapping of the accumulation position as a result of positioning. In this case, the adder with three operands should be used for the accumulation. However, this causes an increase in the critical path delay. Therefore, to solve this problem, one of the overlapped adders can be moved to a collision free location, as shown in Fig. 11. In summary, for implementation of the proposed filter, it is

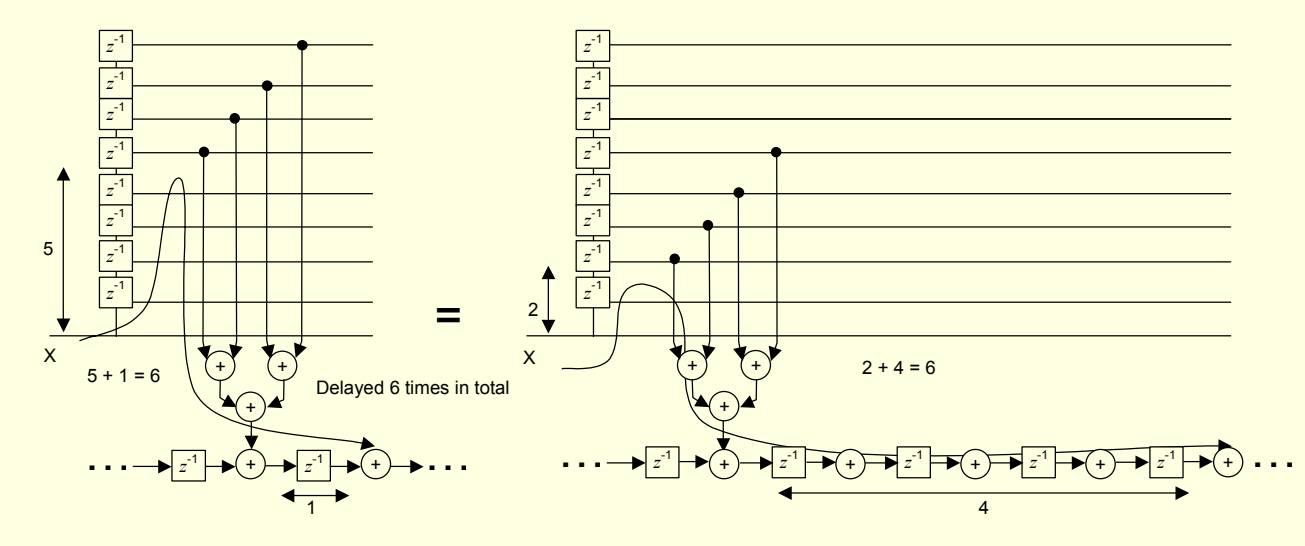

Fig. 8. Filter with the same results.

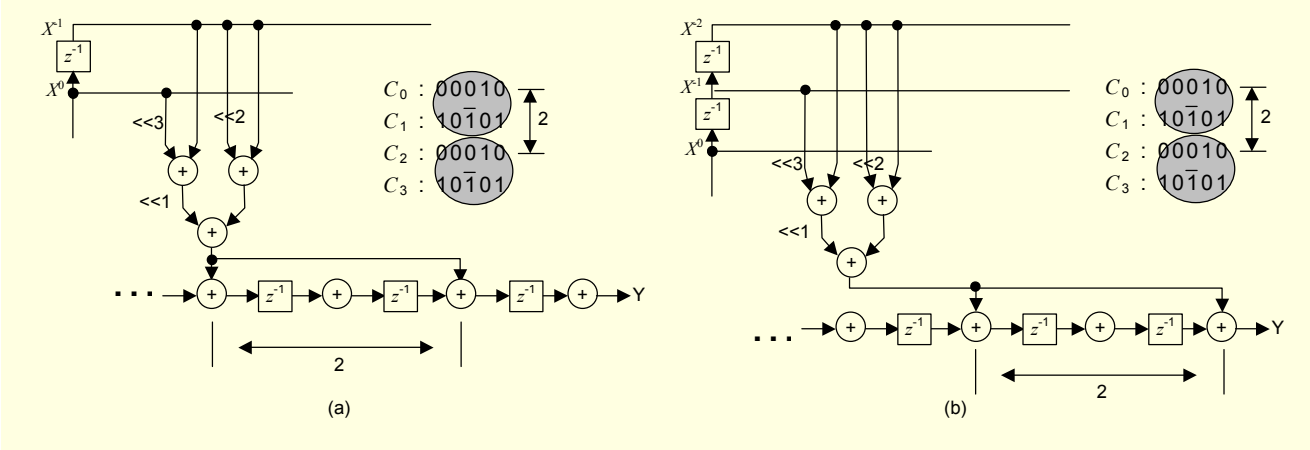

Fig. 9. The number of accumulation registers between shared patterns.

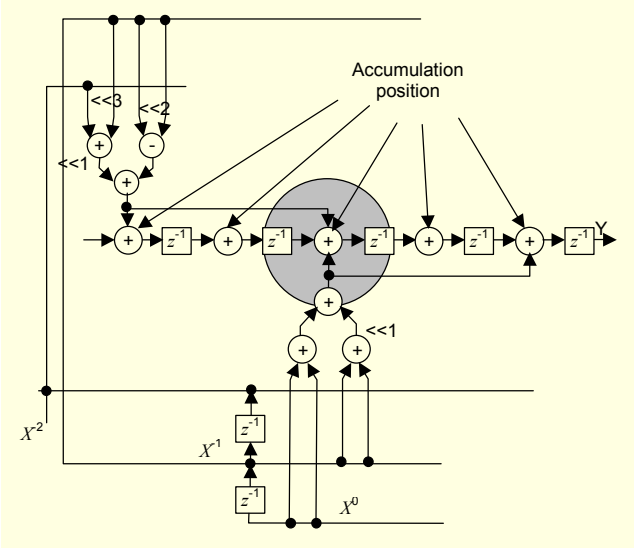

Fig. 10. Overlapped shared adders on accumulation position.

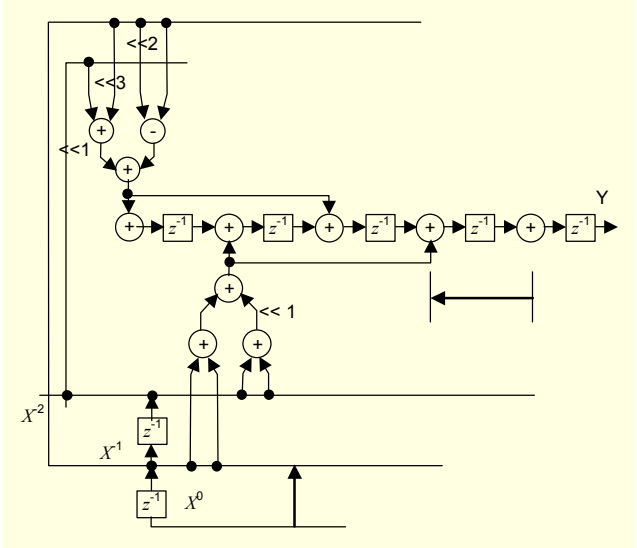

Fig. 11. Repositioned pattern.

necessary to conduct a two-step process. The first step is the pattern searching for adder sharing as described in section III, and the second is the positioning process so as not to overlap the position of the shared adder as illustrated in section IV.

V. Sharing-Pattern Searching Algorithm

Finding optimally shared patterns using the proposed method is confronted by a non-polynomial (NP) problem. Therefore, we have adopted a sub-optimal algorithm for the proposed method which searches patterns using a sharing probability to resolve the NP problem.

Step 1. Load coefficients to Coeff_Orig[][]
Step 2. Change coefficients from binary to canonical signed digit
Step 3. Initialize temporal matrices Coeff_Remained[][] = Coeff_Orig[][]
Step 4. Search all two_term patterns and count the appearance in Coeff_Remained[][] for each patterns.
Step 5. Sort two_term patterns by the number of appearance.
Step 6. For I = 1 : number of different two_term patterns
Search four_term patterns with sorted two_ term patterns and count the appearance in Coeff_Remained[][] for each patterns : UsedForTermNum
Step 7. Delete four_term patterns which have maximum UsedFourTermNum from Coeff_Remained[][]
Step 8. If UsedFourTermNum > 1 goto Step 4 else goto Step 9
Step 9. Search two_term patterns and count the appearance in Coeff_Remained[][] for each patterns : UsedTwoTermNum.
Step 10. Delete two_term patterns which have maximum UsedTwoTermNum from Coeff_Remained[][]
Step 11. If maximum UsedTwoTermNum > 1 goto Step 9 else goto Step 12
Step 12. Construct filter

VI. Experimental Results

1. Design Example

In order to verify the performance of the proposed filter, the specifications in Table 1 are used and its coefficients are generated using MATLAB™. Four methods are implemented and compared; the multiplier block [3], [4], shared common subexpression [5], [6], Step-Limiting RAGn [8], and the proposed structure. Figure 12 and equation (16) show the frequency response of the filter and the filter coefficients, respectively.

Table 1. Specification of the designed filter.

No. of filter taps	48
Edge frequency	0.075, 0.125
Coefficient bits	10 bit
Stopband ripple	-41.9 dB
Total no. of nonzero terms	94
No. of input bits	8 bit
No. of accumulation bits	19 bit

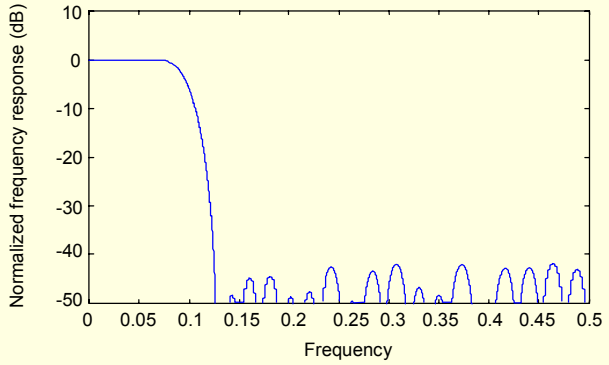

Fig. 12. Frequency response of the designed filter.

$$
\begin{aligned}
C_0 &= 2^{-9} & C_{12} &= 2^{-6} \\
C_1 &= 2^{-8} + 2^{-10} & C_{13} &= 2^{-7} - 2^{-10} \\
C_2 &= 2^{-9} & C_{14} &= -2^{-7} \\
C_3 &= 2^{-9} & C_{15} &= -2^{-5} + 2^{-7} - 2^{-9} \\
C_4 &= -2^{-9} & C_{16} &= -2^{-5} - 2^{-7} + 2^{-9} \\
C_5 &= -2^{-8} - 2^{-10} & C_{17} &= -2^{-5} - 2^{-8} \\
C_6 &= -2^{-7} & C_{18} &= -2^{-6} - 2^{-10} \\
C_7 &= -2^{-7} + 2^{-10} & C_{19} &= 2^{-6} + 2^{-8} + 2^{-10} \\
C_8 &= -2^{-8} + 2^{-10} & C_{20} &= 2^{-4} + 2^{-7} + 2^{-10} \\
C_9 &= 2^{-8} & C_{21} &= 2^{-3} \\
C_{10} &= 2^{-6} - 2^{-8} & C_{22} &= 2^{-2} - 2^{-4} - 2^{-6} - 2^{-10} \\
C_{11} &= 2^{-6} + 2^{-10} & C_{23} &= 2^{-2} - 2^{-4} + 2^{-7} + 2^{-10}
\end{aligned}
\tag{16}
$$

Since searching optimum patterns for adder sharing is a combinatorial NP hard problem, the process of searching for optimum patterns consumes too much computing time. In this paper, we used the heuristic method to get an optimal filter.

2. Design Results

Figure 13 shows the block diagram of the proposed filter

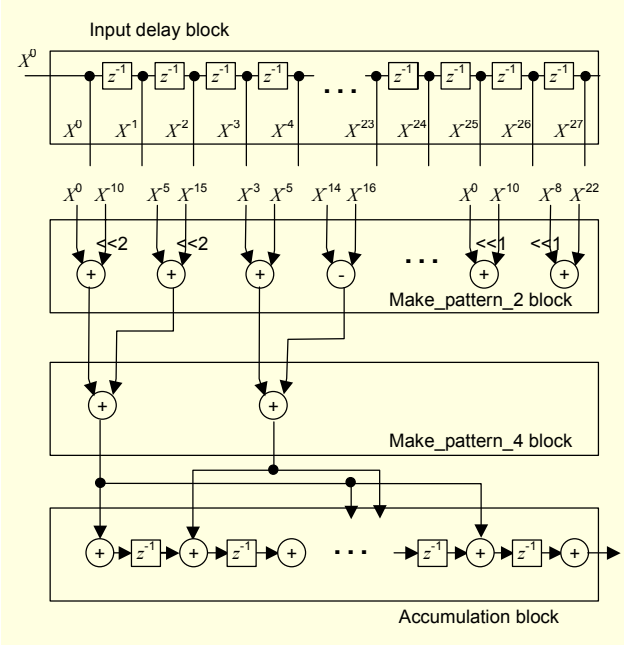

Fig. 13. Block diagram of the designed filter.

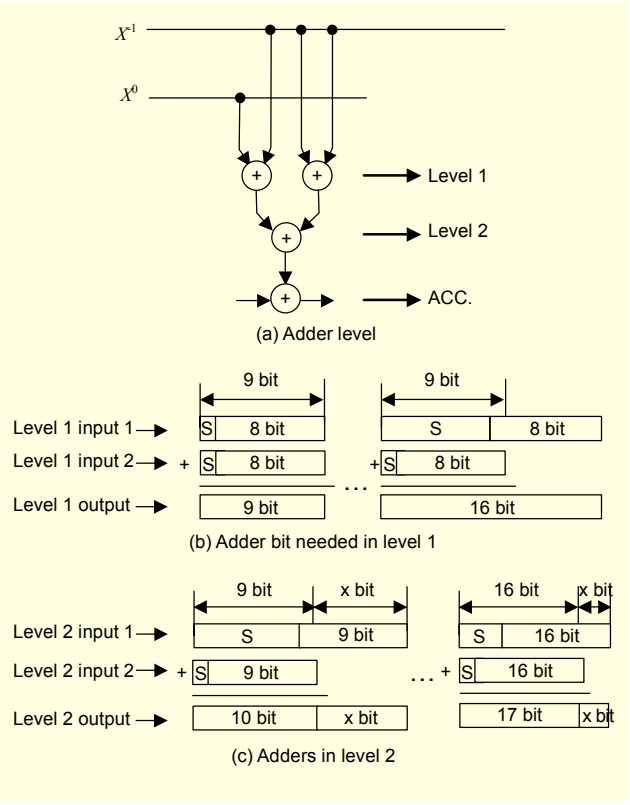

Fig. 14. Adder bits in each level.

which consists of an input delay, make_pattern_2, make_pattern_4, and accumulation block. The input delay and accumulation block are the parts used to delay input X and the accumulation value through the registers, respectively. The make_pattern_2 block adds/subtracts two delayed inputs. The make_pattern_4 block adds/subtracts two outputs from the make_pattern_2 block.

Since we assumed that the number of input bits of the filter designed in this section is 8, a 9-bit adder, illustrated in Figs. 14(a) and 14(b), is necessary to add the two delayed and/or shifted inputs. This results in 9- to 16-bit data. To calculate the level 2 additions in Fig. 14(a), 10- to 17-bit adders are necessary, as shown in Fig. 14(c), which is proved by the simulation of 19-bit operations that are needed in the accumulation of intermediate data. As described above, the adders in each level have a different number of bits. Also, the IR and RA have a different number of flip-flops. Therefore, if the algorithm is compared by only the number of adders or registers, the correct outcome cannot be derived. As a result, this paper compares not only the number of adders and registers but also the total number of bits for the adder and register.

Table 2 shows the results of the filters structured by the

Table 2. Comparison of filter resources.

		RAGn		BHM		SLRAGn		Proposed	
		No.	Total bits	No.	Total bits	No.	Total bits	No.	Total bits
Adder	level 1	5	45	5	45	5	45	21	189
	level 2	6	84	4	56	6	56	12	168
	level 3	0	0	2	28	0	0	0	0
	accumulator	47	893	47	893	47	893	23	437
	total	58	1022	58	1022	58	1022	56	794
Register	input	0	0	0	0	0	0	27	216
	accumulator	48	912	48	912	48	912	24	456
	total	48	912	48	912	48	912	51	672

multiplier block (RAGn) [3], BHM [4], Step-Limiting RAGn (SLRAGn) [8], and the proposed structure. 'No.' and 'total bits' represent the number of adders or registers and the total number of bits in each design, respectively. In level 2 from Table 2, the number of total bits, 84, can be calculated by 14 × 6, since the average number of adder bits in level 2 is 14; 10- to 17-bit adders are used, as shown in Fig. 14(c). By the proposed structure, some adders in an accumulation block are transformed to either level 1 or level 2, and from the RA to the IR. In conclusion, our method can accomplish more than a 26% and 22% reduction in the number of flip-flops and full-adders, respectively, compared to conventional methods.

Table 3 shows the synthesis results to prove Table 2 using the LSI10k library with Synopsys™ DesignAnalyzer. It demonstrates the benefits of the proposed structure including large reductions in the critical path delay.

Table 3. Synthesized results of the filter in Table 1.

	RAGn	SCSE	SLRAGn	Proposed
Combinational logic	7,528	7,868	7,530	6,268
Sequential logic	8,208	8,208	8,208	6,291
Total area	15,736	16,076	15,738	12,559
Critical path delay	50.29	29.59	49.28	29.15

Table 4 shows the design results of another two examples, from which our method turns out to be excellent in both hardware areas and in the critical path delay.

Table 4. Other design examples.

Filter spec.	Algorithm	Tot. adder bits	Tot. reg. bits	Logic depth
T=25 N=9 NZT= 45	RAGn	490	450	2
	SCSE	502	450	2
	SLRAGn	490	450	2
	Proposed	397	318	2
T=59 N=14 NZT = 170	RAGn	1,736	1,416	3
	SCSE	1,760	1,416	2
	SLRAGn	1,788	1,416	2
	proposed	1,644	1,360	2

VII. Conclusion

In this paper, we proposed a new method to reduce both the number of registers and adders which occupy most of the area in a digital filter. The proposed method made it possible to predict the critical path delay and transform a TDF partially into a DF filter. A new adder sharing method has been introduced. As a result, the number of flip-flops and full-adders were decreased. In this paper, the algorithm to obtain the optimal result is not used. Therefore, we cannot assert that the results obtained by the proposed method are always the most suitable ones. If the searching and positioning algorithm that is able to find the optimal result is used, the performance will improve. Researches to find the algorithms for the optimal solution are being conducted.

References

[1] R. Hartley, "Optimization of Canonical Signed Digit Multipliers for Filter Design," *Proc. IEEE Int'l. Symp. Circuits Systems*, Singapore, June 1991, pp. 1992-1995.

[2] M. Potkonjak et al., "Multiple Constant Multiplication: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination," *IEEE Trans. Computer-Aided Design*, vol. 15, no. 2, Feb. 1996, pp. 151-165.

[3] A. Dempster and M.D. Macleod, "Use of Minimum-Adder Multiplier Blocks in FIR Digital Filters," *IEEE Trans. Circuits Syst. II*, vol. 42, Sept. 1995, pp. 569-577.

[4] D.R. Bull and D.H. Horrocks, "Primitive Operator Digital Filter," *Proc. Inst. Ele. Eng. Circuits, Devices and Systems*, vol. 138, pt. G, June 1991, pp. 401-412.

[5] R. Hartley, "Subexpression Sharing in Filters Using Canonical Signed Digit Multipliers," *IEEE Trans. Circuits Syst. II*, vol. 43, Oct. 1996, pp. 677-688.

[6] M. Martinez-Peiro, E.I. 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, Mar. 2002, pp. 196-203.

[7] S.H. Yoon and J.W. Chong, "FIR Digital Filter Implementation Using Flattened Coefficient," *Proc. IEEE Int'l Symp. Circuits and Systems*, Geneva, May 2000, pp. III-363-366.

[8] H.J. Kang and I.C. Park, "FIR Filter Synthesis Algorithms for Minimizing the Delay and the Number of Adders," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 8, Aug. 2001, pp. 770-777.

[9] A. Dempster, S. Demirsoy, and I. Kale, "Designing Multiplier Blocks with Low Logic Depth," *Proc. IEEE Int'l Symp. Circuits and Systems*, Mar. 2002, pp. V-773-776.

Sang-Hun Yoon was born in Seoul, Korea, in 1973. He received the BS and MS degrees in electronic engineering from Hanyang University, Seoul, Korea, in 1996 and 1998. Since 2000, he has been a Lecturer in the Department of Computer Science, Hanyang Woman's College, Seoul, Korea. His current research interests include VLSI circuit design, especially wireless modem design.

Jong-Wha Chong received the BS and MS degrees in electronic engineering from Hanyang University in Seoul, Korea, in 1975 and 1977 and the PhD degree from Waseda University, Japan in 1981 in electronic communication engineering. From 1979 to 1980, he was with NEC Central laboratory. Since 1981, he has been a Professor at Hanyang University. His current research interests are in VLSI design for digital signal and image processing, video compression, high-speed wireless LAN, and digital communication systems.

Chi-Ho Lin received the BS, MS, and PhD degrees in electronics engineering from Hanyang University, Seoul, Korea, in 1985, 1987, and 1996, respectively. Since 1992, he has been with the Department of Computer Science at Semyung University where he is currently an Associate Professor. His research interests are VLSI CAD algorithms, SOC design methodology, VLSI & ASIC designs, and lower power designs. He is currently involved in the administering affairs of IEEK.