New realization of sum of products and transposed sum of products implementation by partitioned arithmetic

A New Realization of Sum of Products and Transposed Sum of Products Implementation by Partitioned Arithmetic

K. NAKAYAMA

Abstract—A new approach to the implementation of sum of products (SP) and transposed sum of products (TSP) using partitioned arithmetic is proposed. This approach is able to realize SP and TSP with recent advanced semiconductor memory technology without multipliers. It is also shown to offer significant reductions in memory capacity requirement and computational complexity. Particularly, in the case of large size SP and TSP this approach becomes more effective and useful. A comparison between the proposed approach and the conventional methods is discussed. Dynamic range constraints and output roundoff noise analysis are also discussed.

I. Introduction

Sum of products (SP) and transposed sum of products (TSP) are defined by (1) and (2), respectively,

$$Y=\sum_{i=0}^{N-1} A_i X_i \tag{1}$$

$$Y_i=A_i X, \qquad i=0,1,\cdots,N-1 \tag{2}$$

which play a very important role in digital signal processing techniques, such as digital filters and the discrete Fourier transform. Therefore, it is desirable to reduce computational complexity of the SP and TSP and to implement these operations in a compact and low cost package.

Manuscript received February 21, 1980; revised September 24, 1980, January 8, 1981, and October 26, 1981.

The author is with the Transmission Division, Nippon Electric Company, Ltd., Nakahara-ku, Kawasaki-City, 211 Japan.

Recently, many effective approaches have been proposed. Burrus *et al.* proposed number theoretic transforms, by which the fast digital convolution and fast FIR filtering can be performed [1], [2]. Peled and Liu proposed a new realization of digital filters using a table look-up method. This method realizes the SP implementation using read only memories (ROM) effectively [3], [4]. Jenkins proposed residue number systems (RNS) applications to digital filters using ROM and accumulators only. In this approach, the Chinese Remainder Theorem is effectively applied to reduce ROM capacity [5].

This paper proposes a new realization of the SP and TSP, using partitioned arithmetic [6]. In the new SP algorithm either A_i or X_i is partitioned into partial bit streams, and redundancy between all partial bit streams is used to reduce computational complexity. In the TSP algorithm A_i is partitioned into partial bit streams. This approach enables the realizations of the SP and TSP with recent advanced semiconductor memory technology without multipliers. It becomes very attractive for large size SP and TSP, and adaptive filtering. Computational complexity and memory capacity requirement are thus reduced.

II. Sum of Products Algorithm

In order to implement a real time sum of products (SP) in a compact and low cost package, it is very important to replace the multiplications by simpler operations such as additions and shifts, which are realized with small digital hardware. Furthermore, effective use of general purpose digital devices, such as memory devices is desirable. In this section a new SP algorithm satisfying the above requirements in hardware realizations is introduced. The SP of A_i and X_i are defined by (1). The smaller wordlength data is chosen to be partitioned, but this choice is somewhat arbitrary. In this paper, A_i is partitioned into the partial bit streams a_{ij}. When A_i is represented by sign-magnitude, then

$$A_i=\sum_{j=1}^{K} a_{is}2^{M_0-jM}\cdot a_{ij} \tag{3}$$

where

a_{ij}	M bit positive integer,
K	number of partial bit streams a_{ij} that is, $j=1,2,\cdots,K$,
a_{is}	sign (1 or -1) of A_i,
2^{M_0-1}	weight of MSB of A_i,
M	number of bits in a_{ij}.

Combining (1) and (3), Y is expressed as follows:

$$Y=\sum_{i=0}^{N-1}\left\{a_{is}2^{M_0}\sum_{j=1}^{K} a_{ij}2^{-jM}X_i\right\}. \tag{4}$$

Two procedures to calculate (4) without multiplications are described here.

Method I

$\tilde{X}_{ij}^{*}$ is calculated at first according to (5).

Step 1

$$\tilde{X}_{ij}^{*}=a_{is}2^{M_0-jM}\cdot X_i, \qquad i=0,1,\cdots,N-1, \qquad j=1,2,\cdots,K. \tag{5}$$

0098-4094/82/0600-0404$00.75 ©1982 IEEE

Y can be expressed with $\tilde{X}_{ij}^*$ as follows:

$$Y=\sum_{i=0}^{N-1}\sum_{j=1}^{K} a_{ij}\tilde{X}_{ij}^*. \tag{6}$$

Since the wordlengths of a_{ij} are M bits, a_{ij} is an integer from 0 through 2^M-1. Let l be an element of this integer set. Equation (6) is evaluated through the following procedures. Partial sum S_l for these $\tilde{X}_{ij}^*$, whose multiplicand a_{ij} is equal to l, is calculated in the next step.

Step 2

$$S_l=\sum_{(i,j)\in\Omega_l}\tilde{X}_{ij}^*, \qquad l=0,1,\cdots,2^M-1 \tag{7}$$

where a_{ij} is equal to l for $(i,j)\in\Omega_l$.

The $S_l(l=0,1,\cdots,2^M-1)$ can be obtained through about KN additions of $\tilde{X}_{ij}^*$. Since l becomes a multiplicand of S_l, then Y in (6) can be expressed as a sum of products of S_l and l

$$Y=\sum_{l=0}^{2^M-1} S_l\cdot l. \tag{8}$$

Equation (8) can be evaluated through the following sequential accumulation of S_l. Let $\tilde{S}_m$ be defined as follows:

$$\tilde{S}_m=\sum_{l=m}^{2^M-1} S_l, \qquad m=2^M-1,2^M-2,\cdots,1. \tag{9}$$

$\tilde{S}_m$ satisfies the following sequential relation:

Step 3

$$\tilde{S}_m=\tilde{S}_{m+1}+S_m, \qquad m=2^M-1,2^M-2,\cdots,1$$
$$\tilde{S}_m=0, \qquad m=2^M. \tag{10}$$

Therefore, $\tilde{S}_m$ can be obtained through 2^M-1 times additions. From (8) and (9) Y can be obtained as a sum of $\tilde{S}_m$

Step 4

$$Y=\sum_{m=1}^{2^M-1}\tilde{S}_m. \tag{11}$$

Equation (11) requires 2^M-1 additions. Totally, the number of additions in Method I becomes

$$0_1(\text{ADD})=KN+2(2^M-1). \tag{12}$$

The number of shifts required in Step 1 becomes

$$0_1(\text{SFT})=(K+1)N. \tag{13}$$

The procedure to be actually implemented are shown by Steps 1, 2, 3, and 4.

Method II

Equation (4) can also be written as follows:

$$Y=\sum_{j=1}^{K}\sum_{i=0}^{N-1}(a_{ij}X_i^*)2^{-jM} \tag{14}$$

where

Step 1

$$X_i^*=a_{is}X_i 2^{M0}, \qquad i=0,1,\cdots,N-1 \tag{15}$$

which is precalculated. Let the partial sum of $X_i^*(i=0,1,\cdots,N-1)$, whose multiplicand $a_{ij}(i=0,1,\cdots,N-1)$ is equal to l, be

TABLE I

EXAMPLE FOR PROPOSED SP ALGORITHM (METHOD I)

Step 1

A_0	10.101	$\tilde{X}_{01}^*$	$-\hat{2}^1X_0$	$\tilde{X}_{02}^*$	$-\hat{2}^3X_0$
A_1	01.101	$\tilde{X}_{11}^*$	$\hat{2}^1X_1$	$\tilde{X}_{12}^*$	$\hat{2}^3X_1$
A_2	01.010	$\tilde{X}_{21}^*$	$\hat{2}^1X_2$	$\tilde{X}_{22}^*$	$\hat{2}^3X_2$
A_3	10.001	$\tilde{X}_{31}^*$	$-\hat{2}^1X_3$	$\tilde{X}_{32}^*$	$-\hat{2}^3X_3$
A_4	11.001	$\tilde{X}_{41}^*$	$-\hat{2}^1X_4$	$\tilde{X}_{42}^*$	$-\hat{2}^3X_4$
A_5	00.110	$\tilde{X}_{51}^*$	$\hat{2}^1X_5$	$\tilde{X}_{52}^*$	$\hat{2}^3X_5$
A_6	00.011	$\tilde{X}_{61}^*$	$\hat{2}^1X_6$	$\tilde{X}_{62}^*$	$\hat{2}^3X_6$
A_7	01.110	$\tilde{X}_{71}^*$	$\hat{2}^1X_7$	$\tilde{X}_{72}^*$	$\hat{2}^3X_7$

Step 2

S_0	$\tilde{X}_{31}^*+\tilde{X}_{61}^*$ $=-\hat{2}^1X_3+\hat{2}^1X_6$
S_1	$\tilde{X}_{01}^*+\tilde{X}_{02}^*+\tilde{X}_{12}^*+\tilde{X}_{32}^*+\tilde{X}_{42}^*+\tilde{X}_{51}^*$ $=-(\hat{2}^1+\hat{2}^3)X_0+\hat{2}^3X_1-\hat{2}^3X_3-\hat{2}^3X_4+\hat{2}^1X_5$
S_2	$\tilde{X}_{21}^*+\tilde{X}_{22}^*+\tilde{X}_{41}^*+\tilde{X}_{52}^*+\tilde{X}_{72}^*$ $=(\hat{2}^1+\hat{2}^3)X_2-\hat{2}^1X_4+\hat{2}^3X_5+\hat{2}^3X_7$
S_3	$\tilde{X}_{11}^*+\tilde{X}_{62}^*+\tilde{X}_{71}^*$ $=\hat{2}^1X_1+\hat{2}^3X_6+\hat{2}^1X_7$

Step 3

$\tilde{S}_1$	$\tilde{S}_2+S_1=S_3+S_2+S_1$
$\tilde{S}_2$	$\tilde{S}_3+S_2=S_3+S_2$
$\tilde{S}_3$	S_3

Step 4

Y	$\tilde{S}_1+\tilde{S}_2+\tilde{S}_3=S_1+2S_2+3S_3$ $=-(\hat{2}^1+\hat{2}^3)X_0+(3\cdot\hat{2}^1+\hat{2}^3)X_1+(2\cdot\hat{2}^1+2\cdot\hat{2}^3)X_2$ $-\hat{2}^3X_3-(2\cdot\hat{2}^1+\hat{2}^3)X_4+(\hat{2}^1+2\hat{2}^3)X_5+3\cdot\hat{2}^3X_6+(3\cdot\hat{2}^1+2\hat{2}^3)X_7$

S_{jl}, where l is an integer from 0 to 2^M-1.

Step 2

$$S_{jl}=\sum_{i\in\Omega jl} X_i^*, \qquad j=1,2,\cdots,K, \qquad l=0,1,\cdots,2^M-1 \tag{16}$$

where a_{ij} is equal to l for $i\in\Omega_{jl}$. Since the multiplicand of S_{jl} is l, (14) can be rewritten as follows:

$$Y=\sum_{j=1}^{K}\left(\sum_{l=0}^{2^M-1} S_{jl}\cdot l\right)2^{-jM}. \tag{17}$$

By exchanging the order of accumulations

$$Y=\sum_{l=0}^{2^M-1}\left(\sum_{j=1}^{K} S_{jl}2^{-jM}\right)l. \tag{18}$$

The partial sum S_l expressed by (7) can be obtained using S_{jl} as follows:

Step 3

$$S_l=\sum_{j=1}^{K} S_{jl}2^{-jM}, \qquad l=0,1,\cdots,2^M-1. \tag{19}$$

The following procedure to obtain Y using S_l is the same as Method I.

Step 4

$$\tilde{S}_m=\tilde{S}_{m+1}+S_m, \qquad m=2^M-1,2^M-2,\cdots,1 \tag{10}$$
$$\tilde{S}_m=0, \qquad m=2^M.$$

TABLE II
PERFORMANCE PARAMETERS FOR TABLE LOOK-UP METHOD AND PROPOSED METHOD (METHODS I AND II)

	Table look-up method	Method I	Method II
Number of additions	$\frac{N}{P}L_X - 1$	$KN + 2(2^M - 1)$	$KN + 2(2^M - 1)$
Number of shifts	$\frac{N}{P}(L_X - 1)$	$(K + 1)N$	$K \cdot 2^M + N$
Output roundoff noise	$\sigma_0^2 \frac{N}{P}$	$KN\sigma_0^2$	$2^M\sigma_0^2$
Dynamic range	$\sqrt{N}\, 2^{M_0} \max_i \lvert X_i \rvert$	$2^{M_0} \max_i \lvert X_i \rvert$	$2^{M_0}\sqrt{\frac{N}{2^M}} \max_i \lvert X_i \rvert$
Internal data wordlengths	$W_0 + (\frac{1}{2}\log_2 \frac{N}{P}) + (\frac{1}{2}\log_2 N + M_0)$	$W_0 + \frac{1}{2}\log_2 KN + (\frac{1}{2}\log_2 N + M_0)$	$W_0 + \frac{1}{2}M + (\frac{1}{2}\log_2 N + M_0)$
Memory capacity	$NL_X + \frac{N}{P}2^P(L_A + 1 + \log_2 P)$	$N(L_A + 1 + L_X) + L_X 2^M$	$N(L_A + 1 + L_X) + L_X 2^M$

Step 5

$$Y = \sum_{m=1}^{2^M - 1} \tilde{S}_m. \tag{11}$$

Method II is carried out through Steps 1, 2, 3, 4, and 5. The number of additions is the same as that of Method I

$$0_2(\text{ADD}) = KN + 2(2^M - 1). \tag{20}$$

The number of shifts required in Step 3 becomes

$$0_2(\text{SFT}) = K \cdot 2^M + N. \tag{21}$$

In each method no multipliers are required since their operations are replaced by additions and shifts. Memory devices can be effectively applied to obtain the partial sum S_l or S_{jl} using the partial bit stream a_{ij} as an address signal.

An example for the proposed algorithm is illustrated in Table I.

III. Roundoff Noise Analysis and Dynamic Range Constraints

The internal data wordlengths are another important factors termining signal processing speed and hardware size. When 2's-complement representation is employed, overflow has to be prevented at the multiplier input. The internal data wordlengths are determined from two factors, the scaling factor and the output roundoff noise gain. Their theoretical formulas are summarized in Table II and the actual derivations are given in the Appendix.

Roundoff error at a multiplier output can be assumed to be distributed in the region $[-\Delta/2, \Delta/2]$ uniformly, where Δ is 2^{-t} and t indicates the least significant bit. Its variance σ_0^2 is given by $\Delta^2/12$. In the formulas for the internal data wordlengths, W_0 is the basic common wordlength, which is determined by the required noise performance, the second term corresponds to the output roundoff noise, and the third term is determined by the dynamic range constraints.

IV. Hardware Realization

The proposed approach can be effectively realized using random access memories (RAM's). The partial sum or partial sum of products are stored in a RAM. Their address is the partial bit stream a_{ij}. The hardware realization for Method I is described here. Its block diagram is shown in Fig. 1. The implementation is described using the illustration in the following.

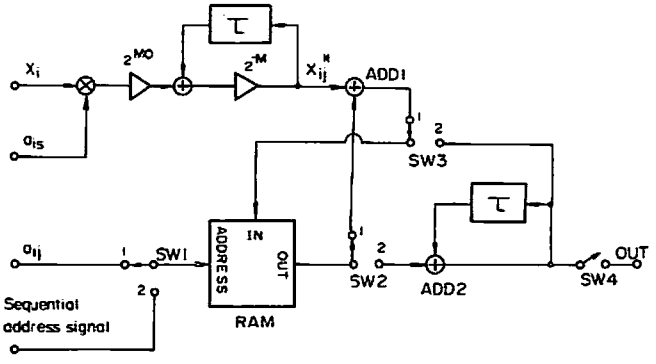

Fig. 1. SP hardware realization using RAM for Method I.

Phase 1: Turn on SW1, SW2, and SW3 to side 1, and turn off SW4. $\tilde{X}_{ij}^*$ is obtained through multiplying X_i by the sign of A_i and shifting by 2^{-jM}. At the same time, the partial sum S_l is read from a RAM using a_{ij} as an address signal. A new S_l is stored again in the RAM at the same address after addition of $\tilde{X}_{ij}^*$ and $S_l(a_{ij} = l)$. This operation is repeated N times $i = 0$ through $N-1$. At this phase S_l by (7) is obtained.

Phase 2: Turn on SW1, SW2, and SW3 to side 2 and turn off SW4. The partial sum S_l is read from the RAM using the sequential address signal $l = 2^M - 1$ through 1, and is accumulated by an ADD2. $\tilde{S}_l(l = 0 \sim 2^M - 1)$ by (10) is obtained as the ADD2 output, and stored again in the RAM at the same address.

Phase 3: Turn on SW1 and SW2 to side 2 and turn off SW3 and SW4. The partial sum $\tilde{S}_l$ is read from the RAM using the sequential address signal $l = 2^M - 1$ through 1, and accumulated by the ADD2. Then Y is obtained as the final output of the ADD2. Y goes through SW4 to the SP output.

Memory Capacity

The number of signals, coefficients, and partial sums stored in memories are N, N, and 2^M, respectively. Their wordlengths are L_X, $L_A + 1$, and L_X, respectively. Therefore, the total required memory size becomes

$$C = N(L_A + 1 + L_X) + L_X 2^M. \tag{22}$$

C is the same for two methods.

V. Numerical Examples for Performance Comparison

In order to compare the proposed methods and a more conventional one, performances of the table look-up method, suggested by Peled and Liu [3], and Methods I and II are listed in Table II. In this table, L_X is the internal data wordlength and P is the number of serial inputs to a ROM.

Computational Complexity

The computational complexity is evaluated by the number of additions and shifts in the numerical examples. Since A_i and X_i are considered as the coefficient and the internal data, respectively, their wordlengths usually must be around 12 bits and 20

TABLE III
NUMBER OF COMPUTATIONS IN THE PROPOSED APPROACH. *NUMBER OF ADDITIONS. **NUMBER OF SHIFTS. THE UPPER AND LOWER ROWS INDICATE METHODS I AND II, RESPECTIVELY. THEY ARE NORMALIZED BY N. DATA WITH UNDERLINE IMPLY MINIMUM NUMBER OF COMPUTATIONS

	L_A = 12 bit							
N \ K	2		3		4		6	
	*	**						
20	8.30	3.00	4.50	4.00	4.70	5.00	6.30	7.00
	8.30	7.40	4.50	3.40	4.70	2.60	6.30	2.20
50	4.52	3.00	3.60	4.00	4.28	5.00	6.12	7.00
	4.52	3.56	3.60	1.96	4.28	1.64	6.12	1.46
100	3.26	3.00	3.30	4.00	4.14	5.00	6.06	7.00
	3.26	2.28	3.30	1.48	4.14	1.32	6.06	1.24
500	2.25	3.00	3.06	4.00	4.03	5.00	6.01	7.00
	2.25	1.26	3.06	1.10	4.03	1.06	6.01	1.05
1000	2.13	3.00	3.03	4.00	4.01	5.00	6.01	7.00
	2.13	1.13	3.03	1.05	4.01	1.03	6.01	1.02

TABLE IV
NUMBER OF COMPUTATIONS IN TABLE LOOK-UP METHOD. *NUMBER OF ADDITIONS. **NUMBER OF SHIFTS. THEY ARE NORMALIZED BY N

L_X \ P	4		6		8	
	*	**				
16	4.00	3.75	2.67	2.50	2.00	1.88
20	5.00	4.75	3.33	3.17	2.50	2.38
24	6.00	5.75	4.00	3.83	3.00	2.88

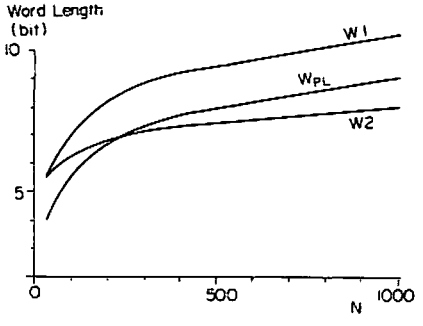

Fig. 2. Internal data wordlengths. W_i and W_{PL} correspond to Method i and table look-up method, respectively.

bits, respectively, in actual applications. The numbers of additions and shifts for each method are shown in Tables III and IV. In these tables L_A is the wordlength of A_i except for the sign bit and K is the number of partial bit streams. The number of additions and shifts are all normalized by the SP size N. In the conventional method with multipliers, the number of additions is L_A+1. Therefore, the proposed approach can reduce significantly the computational complexity, which becomes 25 percent for $N=100$ and $L_A=12$ bits. The table look-up method requires a higher computational complexity, which becomes 153 percent for $P=4$, $L_A=12$ bits, $L_X=20$ bits, and $N=100$ than the proposed method. The condition $P=4$ indicates the required memory capacities in both approaches are the same.

Internal Data Wordlengths

Numerical examples are illustrated in Fig. 2, where L_A, K, and P are taken as 12 bits, 2 and 6, respectively, and W_0 and M_0 are

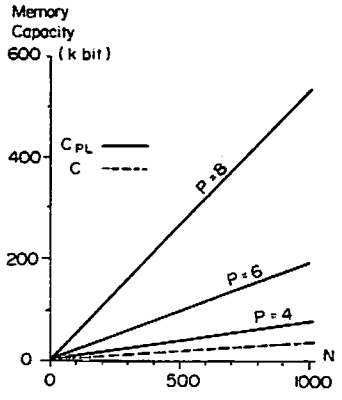

Fig. 3. Memory capacity requirement for proposed method C and table look-up method C_{PL}.

TABLE V
NUMBER OF COMPUTATIONS PROPOSED TSP REALIZATION. *NUMBER OF ADDITIONS. **NUMBER OF SHIFTS. THEY ARE NORMALIZED BY N.

	L_A = 12 bit							
N \ K	2		3		4		6	
	*	**						
20	4.10	2.05	2.70	3.05	3.30	4.05	5.10	6.05
50	2.24	2.02	2.28	3.02	3.12	4.02	5.04	6.02
100	1.62	2.01	2.14	3.01	3.06	4.01	5.02	6.01
500	1.12	2.00	2.03	3.00	3.01	4.00	5.00	6.00
1000	1.06	2.00	2.01	3.00	3.00	4.00	5.00	6.00

taken as zero. The difference between the proposed approach and the table look-up method is small.

Memory Capacity

Numerical example is shown in Fig. 3, where L_A is taken to be 12 bits and L_X be 20 bits. This figure shows that the proposed approach can significantly reduce the memory capacity requirement from the table look-up method.

VI. TRANSPOSED SUM OF PRODUCTS

As mentioned previously, the TSP is defined by (2)

$$Y_i = A_i X, \qquad i=0,1,\cdots,N-1. \tag{2}$$

This operation appears in the transposed transversal filters. In this case, partitioned arithmetic can be also effectively applied. In the TSP algorithm, A_i is to be partitioned and expressed with the partial bit streams a_{ij} as in (3). The proposed TSP operating process is described in the following.

Phase 1: $X^*l, l=0,1,\cdots,2^M-1$ is obtained through 2^M-2 times sequential additions, where $X^*=2^{M_0}X$ which is precalculated

$$X^*l = X^*(l-1)+X^*, \qquad l=1,2,\cdots,2^M-1$$
$$X^*l = 0, \qquad l=0. \tag{23}$$

X^*l is stored in a RAM using l as the write address signal.

Phase 2: X^*l is read from the RAM K times $j=K$ through 1, using a_{ij} $(j=1,2,\cdots,K)$ as the read address signal, and accumulated after shifting by 2^{-M}. XA_i is obtained after multiplied by sign of A_i. Phase 2 is repeated N times $i=0$ through $N-1$, then all $XA_i(i=0,1,\cdots,N-1)$ are obtained. In this approach,

the numbers of additions and shifts become

$$0_{TSP}(ADD) = (2^M - 2) + N(K-1) \tag{24}$$

$$0_{TSP}(SFT) = KN + 1. \tag{25}$$

A numerical example is shown in Table V. The number of additions is reduced to about 14 percent of that of a direct method using multipliers, for $L_A = 12$ bits and $N = 100$. In order to compare the SP with the TSP, additional $N-1$ additions must be required for the TSP, and the normalized number of additions increases about one per filter output sample. The number of additions for the TSP becomes about 80 and 52 percent compared with the SP and the table look-up method, respectively. The memory capacity is the same as that of the SP.

VII. Conclusion

A new approach to implement the sum of products and transposed sum of products is proposed that employs partitioned arithmetic.

Its hardware can be realized in small memory capacity and low computational complexity using a RAM. The proposed approach is superior to the conventional one for large size SP and TSP, and adaptive filtering.

Appendix

The output roundoff noise and the dynamic range constraints in the proposed SP algorithm are analyzed in detail here.

1) Output Roundoff Noise

Method I: Equation (5) shows that KN shifts of 2^{-M} are required. When the shifting by jM bits is performed recursively using only an M bit shifter, the transfer function from the M bit shifter to the SP output is approximately unity.

Method II: Equation (19) shows that S_{jl} is shifted by 2^{-M} and accumulated K times, to obtain S_l. In this case, roundoff error sources are considered to be one for each S_l. The transfer function from each noise source to the SP output is unity.

2) Dynamic Range

Let $D_i^M (i = 1,2)$ be the signal level from the SP input to the shifter input and let $D_i^0 (i = 1,2)$ be the signal level from the input to the output of the SP.

Method I: From (5), D_1^M is

$$D_1^M = 2^{M_0} \max_i |X_i|. \tag{A1}$$

Method II: From (19), D_2^M is

$$D_2^M = \max_{l,k} \left| \sum_{j=1}^{k} S_{jl} \cdot 2^{M_0-(j-1)M} \right|, \qquad k = 1,2,\cdots,K. \tag{A2}$$

Since $2^{-M} \ll 1$, D_2^M is approximately

$$D_2^M = 2^{M_0} \max_{l,k} \left| \sum_{j=1}^{k} S_{jl} \right|, \qquad k = 1,2,\cdots,K. \tag{A3}$$

In order to calculate the distribution of S_{jl}, the following assumptions on the distribution of X_i and a_{ij} are employed.

Assumption 1: Let φ_X be the probability distribution function of X_i,

$$\varphi_X(X) = \frac{\alpha}{\sqrt{2\pi}\,\sigma_X} e^{-X^2/2\sigma_X^2}, \quad |X| \le 3\sigma_X$$
$$= 0, \qquad |X| > 3\sigma_X \tag{A4}$$

$$\int_{-3\sigma_X}^{3\sigma_X} \varphi_X(X) = 1 \tag{A5}$$

where

$$\max_i |X_i| = 3\sigma_X. \tag{A6}$$

Let X_1 and X_2 be the random variables which satisfy (A4) and (A5). It is assumed that $Y = X_1 + X_2$ also satisfies (A4) and (A5). Its variance is $\sigma_{X1}^2 + \sigma_{X2}^2$ and the maximum is $\sqrt[3]{\sigma_{X1}^2 + \sigma_{X2}^2}$.

Assumption 2: a_{ij} are assumed to be distributed uniformly in the region of $[0, 2^M)$, let φ_a be the PDF

$$\varphi_a(a) = 2^{-M}, \qquad 0 \le a < 2^M$$
$$= 0, \qquad a < 0, 2^M \le a. \tag{A7}$$

These assumptions must be modified according to the actual applications, but do not lose generality in the performance comparison for several kinds of approaches. From Assumptions 1 and 2, the distribution of S_{jl} becomes the same distribution as X_i, with mean = 0 and variance = $N\sigma_X^2/2^M$. Then D_2^M is

$$D_2^M = 2^{M_0} \sqrt{\frac{N}{2^M}} \max_i |X_i|. \tag{A8}$$

Next, the dynamic range constraints at the SP output are considered. Y is expressed in (1)

$$Y = \sum_{i=0}^{N-1} A_i X_i. \tag{1}$$

When $A_i X_i$ and $A_j X_j (i \ne j)$ are independent, the variance of Y is $\sum_{i=0}^{N-1} A_i^2 \sigma_X^2$, where σ_X^2 is the variance of X_i. Then, $\max |Y|$ becomes

$$\max |Y| = \left(\sum_{i=0}^{N-1} A_i^2 \right)^{1/2} \cdot \max_i |X_i|. \tag{A9}$$

Since

$$\left(\sum_{i=0}^{N-1} A_i^2 \right)^{1/2} \le \sqrt{N} \max_i |A_i| = \sqrt{N}\, 2^{M_0} \tag{A10}$$

$$\max |Y| \le \sqrt{N}\, 2^{M_0} \max |X_i|. \tag{A11}$$

D_1^0 and D_2^0 are equal to $\max |Y|$

$$D_i^0 = D_2^0 = \max |Y|. \tag{A12}$$

The SP input scaling is determined by the more significant value D_i^M or D_i^0.

Acknowledgment

The author thanks M. Hibino and T. Mizukami for their discussions.

References

[1] R. C. Agarwal and C. S. Burrus, "Number theoretic transforms to implement fast digital convolution," *Proc. IEEE*, pp. 550–560, Apr. 1975.

[2] C. S. Burrus, "Digital filter structures described by distributed arithmetic," *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 674–680, Dec. 1977.

[3] A. Peled and B. Liu, "A new hardware realization of digital filters," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-22, pp. 456–462, Dec. 1974.

[4] B. Liu and A. Peled, "A new hardware realization of high-speed fast Fourier transformers," *IEEE Trans. Acoust. Speech, Signal Processing*, vol. ASSP-23, pp. 543–547, Dec. 1975.

[5] W. K. Jenkins and B. J. Leon, "The use of residue number systems in the design of finite impulse response digital filters, *IEEE Trans. Circuits Syst.*, vol. CAS-24, pp. 191–201, Apr. 1977.

[6] K. Nakayama, "Digital filter," U.S. Patent, 4 255 794, Mar. 10, 1981.