다중 상수 곱셈을 위한 하드 와이어드 분산 연산

김대원*, 최준림**
한국전자통신연구원*, 경북대학교**

Hardwired Distributed Arithmetic for Multiple Constant Multiplications and Its Applications for Transformation

DaeWon Kim*, JunRim Choi**
ETRI*, KNU*
E-mail : *won22@etri.re.kr, jrchoi@ee.knu.ac.kr

Abstract

We propose the hardwired distributed arithmetic which is applied to multiple constant multiplications and the fixed data path in the inner product of fixed coefficient as a result of variable radix-2 multi-bit coding. Variable radix-2 multi-bit coding is to reduce the partial product in constant multiplication and minimize the number of addition and shifts.

At results, this procedure reduces the number of partial products that the required multiplication timing is shortened, whereas the area reduced relative to the DA architecture. Also, this architecture shows the best performance for DCT/IDCT and DWT architecture in the point of area reduction up to 20% from reducing the partial products up to 40% maximally.

Introduction

Most calculations in Many applications in DSP, telecommunications, graphics and control systems involve multiplications with one variable and several constants. These multiplications have made it difficult for hardware architecture designers as far as hardware cost. Many people have tried to improve the process by exploring throughput, area, gate counts and power. The greatest advantage when designing algorithms lies in the designer's ability to recognize and manipulate regularity, symmetry and other structural properties of computation. The recent developments of synthesis tools and compilers have solved these problems to a certain degree. But two ongoing efforts are being explored with regards to optimization techniques. [1][2][3] One is the optimization of isolated multiplication with constants, and the other is minimization of the number of shifts, with little attention paid to the number of additions. [4] Therefore, it is important not to ignore the reduction of partial products in multiplication before designing multiplier. First, we introduce an algorithm which we have modified from Booth's algorithm and which we called the variable radix-2 multi-bit coding in order to reduce the partial product in constant multiplication and minimize the number of addition and shifts. Variable radix-2 multi-bit coding[5] is for single constants multiplication and then, we can expand the formulation in inner products, vector analysis or dot production generation which called hardwired distributed arithmetic (HDA). HDA means the fixed data path in the inner product of fixed coefficient as a result of variable radix-2 multi-bit coding. HDA is formulated for the multiple constant multiplications. Therefore, HDA can be applied to a lot of transformations and the multiplications of filter coefficients. Among these applications, I chose the two transformation techniques which are often used in image compression: discrete cosine transform (DCT) and discrete wavelet transform (DWT).

A. Variable Radix-2 multi-bit coding (2^k SD representation):

Let the multiplicand X and the multiplier Y be n+1 bit two's complementary format binary integers.

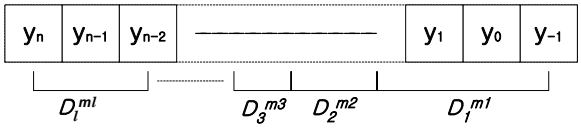

Fig. 1. n+1 bit two's complementary format of Y
(l : total number of SD, ml : number of recoded bit per SD D_l^{ml} : l th 2k SD with ml bit recoding)

Fig.1 shows n+1 bit position represented by 2's complement. By definition $y_{-1}=0$, which represents the appended zero bit to the right of y_0. From the general multi-bit recoding algorithm, if D_l^{ml} noted as SD is the radix of 2^k, the multiplicand X is only shifted by k-bit. Therefore, we can find the general form for radix two multi-bit coding to avoid D_l^{ml}, which is not the radix of 2^k as follow.

$$Y = \sum_{i=1}^{l} D_i^{mi} \cdot 2^{Li+1}$$

$$D_i^{mi} = y_{Li} + \sum_{j=0}^{mi-3} y_{j+Li+1} \cdot 2^j - y_{L(i+1)} \cdot 2^{mi-2}$$

$$(where, \quad mi \geq 3)$$

$$Li = \sum_{j=1}^{i-1} m_j - i$$

$$(where, \quad if \quad i=1, \quad then \quad \sum_{j=1}^{i-1} m_j = 0) \qquad (1)$$

From (1), Li is new variable, which shows the encoded bit position. mi which is lager than 3, means this equation is

applied with lager size of variable multi-bit encoding.
Thus, we introduced the general 2^k SD representation with variable radix two multi-bit recoding.

B. Proof of variable radix-2 multi-bit coding.

The proof consists of substituting the value of D_i^{mi} from (1) into Y and showing that the result is the same as the two's complement binary value of *Y*. After substituting the value of Di^{mi} and rearranging it, we obtain as follow (2)

$$Y = \sum_{i=1}^{l} (y_{Li} + \sum_{j=0}^{mi-3} y_{j+Li+1} \cdot 2^j - y_{L(i+1)} \cdot 2^{mi-2}) \cdot 2^{Li+1} \quad (2)$$

By using $2^{i+1} - 2^i = 2^i$, if we remove the bracket then, index of y starts from –1 to n, then total number of bits is n+2. Therefore, if we apply

$m1 + m2 + m3 + \cdots\cdots\cdots + m(l-1) + ml - l = n$ and

y_{-1} =0, the equation will shows below (3)

$$\begin{aligned} Y &= -y_{m1+m2+\cdots+m(l-1)+ml-(l+1)} \cdot 2^{m1+m2+m3+\cdots+m(l-1)+ml-l-1} \\ &\quad + y_0 \cdot 2^0 + y_1 \cdot 2^1 + \cdots + y_{m1-2} \cdot 2^{m1-2} + \cdots + y_{m1+m2-3} \cdot 2^{m2-2} + \cdots\cdots \\ &\quad + y_{m1+m2+\cdots+m(l-1)+ml-(l+2)} \cdot 2^{m1+m2+m3+\cdots+m(l-1)+ml-l-2} \\ &= -y_{n-1} \cdot 2^{n-1} + y_0 \cdot 2^0 + y_1 \cdot 2^1 + \cdots + y_{n-2} \cdot 2^{n-2} \\ &= -y_{n-1} \cdot 2^{n-1} + \sum_{i=0}^{n-2} y_i \cdot 2^i \end{aligned} \quad (3)$$

The last equation in Y means two's complement binary form. Because, from our representation, we induced general two's complement binary format.

C. Fast Encoding for deciding l, ml: In Variable Radix-2 multibit coding.

The key point is how to find the l, ml in eq.(1). Because k-bit encoding is included from modified booth encoding to k-1 bit encoding, we induced from k bits modified booth encoding.

① Find the maximum group that has only one 1,0 in each bit position.
② Find the maximum group that has all 1's or 0's except MSB.
③ If MSB is 0, it's the positive encoding, if not, negative encoding.

Therefore, the procedure of radix-2 multi-bit coding is summarized as fig 2

D. Multiple Constant Multiplication and Hardwired Distributed Arithmetic (HDA)

In this chapter, we apply variable radix-2 multi-bit coding to multiple constant multiplications. Distributed Arithmetic (DA) is basically a bit-serial computational operation that forms inner products of a pair of vectors in single direct step. Now, we call variable radix-2 multi-bit coding applied to multiple constant multiplications as Hardwired Distributed Arithmetic (HDA). HDA means the fixed data path in the inner product of fixed coefficient. In this chapter, we compare the HDA with the direct DA. As an example of direct DA inner product generation consider the calculation of the following sum of products:

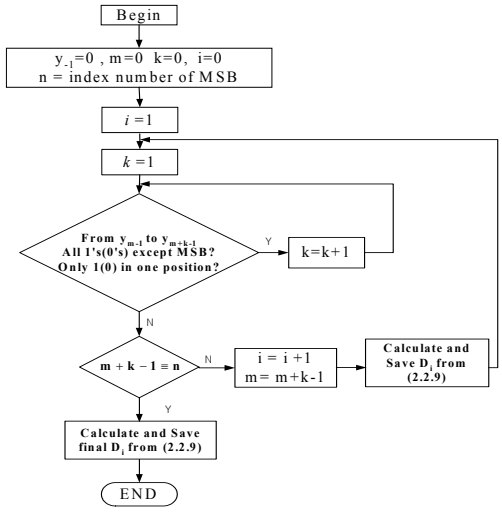

Fig 2. The flow chart of scanning the 2^k SD and finding Di

$$Y = \sum_{n=1}^{N} A_n X_n \quad (4)$$

In (4), A_n are fixed coefficients and the X_n are the input data words. Fixed coefficient A_n can be substituted by the expression of variable radix-2 multibit coding and replaced in Y as (5).

$$A_n = \sum_{i=1}^{\ln} D_i^{mi} \cdot 2^{Li+1} \quad (5)$$

$$Y = \sum_{n=1}^{N} \left(\sum_{i=1}^{\ln} D_i^{mi} \cdot 2^{Li+1} \right) \cdot X_n \quad (6)$$

$$Y = \sum_{n=1}^{N} \left(\sum_{i=1}^{\ln} (D_i^{mi} \cdot 2^{Li+1} \cdot X_n) \right) \quad (7)$$

Eq. (7) is similarly formed like DA. Because *ln* is not fixed, we can't calculate N summation first like direct DA. In this equation, *ln* decides the number of partial product in a coefficient. Total *l*xN products are generated. The purpose of encoding is finding the minimized number of partial product and implementing high speed. We apply variable radix-2 multi-bit coding and the result is simplified because all sign digits are simply represented by 2^k.

$$Y = \sum_{n=1}^{N} \left(\sum_{i=1}^{\ln} (2^{ai} \cdot 2^{Li+1} \cdot X_n) \right) = \sum_{n=1}^{N} \left(\sum_{i=1}^{\ln} (2^{ai+Li+1} \cdot X_n) \right) \quad (8)$$

From (8), we should calculate $2^{ai+Li+1} \cdot X_n$ first in every *ln*'s when *ln* is fixed. This calculation is only shifting *Xn* by $2^{ai+Li+1}$ and accumulating the shifted *Xn*'s in *ln* times. When Xn's are accumulated, we don't always use accumulators but just the fixed data path because all coefficients are fixed. The HDA means replacing accumulator with only data path like wallace adder tree because we know all bit position to be added.

From now on, we investigate a general 2^k SD representation form with variable radix-2 multi-bit coding and sign extension. If the multiplier or multiplicand is a constant coefficient, this method has an advantage because we get partial products from 2^k SD with variable radix-2 multi-bit coding and replace them by hardwiring. Therefore, this algorithm is most powerful for multiplication with constant coefficients such as FIR or IIR filter, DCT/IDCT and wavelet format. Table 1 shows the comparison of other architectures in general characteristic. Among many hardware implementations of vector matrix, DCT and wavelet coding is very useful for the image compression in JPEG, MPEG. In the next session, we will apply this algorithm to DCT/IDCT and wavelet transform.

Table 1. The characteristic comparison of the three schemes

	ROM-based DA[6]	HDA	Adder-based DA[7]
The number of partial products	2^n	$\sum_{i=1}^{\ln} Li$	2^n
$N \times N$ Required ROM	$N \times 2^N$	0	0
Additional requirement	N serial accumulator	Parallel addition (depends on pipelining)	Max N serial accumulator
Speed dependency	n cycle Accumulator	Adder	Max n cycle Accumulator

E. Hardware Architecture for Computational Unit (CU) in DCT and DWT with HDA

We employed Chen's algorithm [8] to ensure sufficient accuracy with small amount of hardware in fixed-point data form.. Fig. 3 show basic CU (computational unit) for DCT/IDCT matrix using carry save adders.

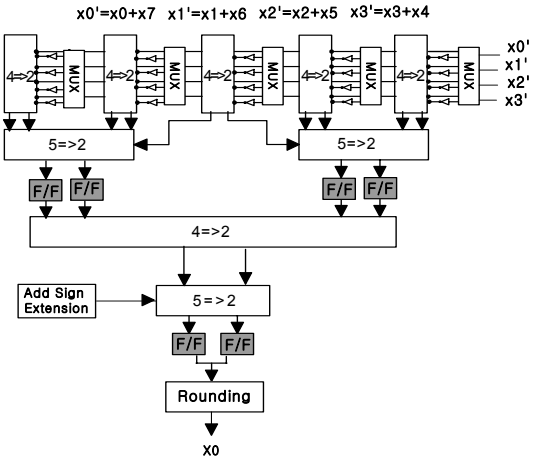

Fig. 3. DCT X0 matrix summation network in CU

Fig. 3 is the example of the calculation for DCT even matrix. This computational unit consists of three blocks: 4:2 compressor, 5:2 compressor and sign extension adder. 4:2 compressor generates sum and carry from four 16-bit inputs. The input control block calculates 4 sums and subtractions of each input pair ($x_0 \pm x_7$, $x_1 \pm x_6$, $x_2 \pm x_5$, $x_3 \pm x_4$) and these 4 results become the inputs to the CU for DCT. In IDCT, all architecture are similar to this.

In DWT design, the computational unit is needed for filter coefficient multiplication. Architecture to generate 2 coefficients in DWT is shown in Fig 4.

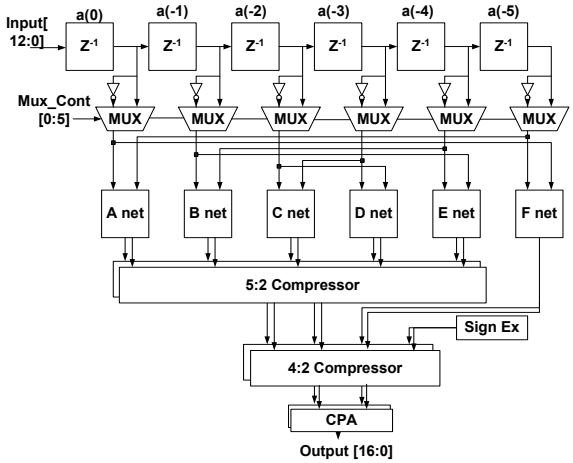

Fig. 4. Architecture for the generation of 2 DWT coefficients in Daubechies N=6

In Fig 4, Input data are 14 bit data which is 8 bit image data multiplied data by 2^4 because the filter coefficients are 14-bit 2's complement binary format. This is important to satisfy the accuracy for precision in filter coefficient. In general, dynamic range of DWT coefficients is greater than that of input data. Input data is 8 bit image and intermediate coefficient is taken as 16 bit range after multiplying filter coefficients. We generate 2 DWT coefficients per 1 computational unit as shown in Fig 4. This is composed of three parts: input delay and control, filter summation network and carry save adder tree for partial products. In input delay and control, inputs are delayed for 6-tap filter multiplication and to make the simple hardware architecture for the filter bank, control block generates the summation network input that is multiplied by -1 in advance, as following the sign of filter coefficient and we make adder network with N=6 Daubachies's symmetric coefficients characteristic in low and high pass filter using the fixed data path applied HDA. These summation networks ignore multiplying the minus sign of filter coefficients. An *A* network calculates $a(0)*h_h(0)$ and $a(0)*h_l(5)$. $h_h(5)$and $h_l(0)$ are same as A.[9] Therefore, we simultaneously make two coefficients with parallel network. From Fig 4.3, A network is composed of the architecture for 5 partial product summation. In DCT/IDCT, we apply 5:2 compressor to 5 partial products and also, we can modified the architecture. *A* net architecture is like Fig. 5.

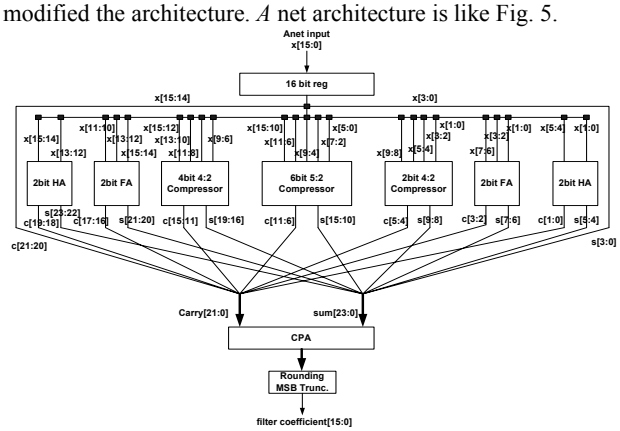

Fig. 5. *A* Network architecture

F. The reduced partial products

. In the case of DCT/IDCT, the modified Booth algorithm generates 7 partial products of the 12-bit input data and one sign extension to be added for multiplication. Then we need to add 32 partial products for each row of the matrix multiplier. If we process 8 pixels at a time, we will have to add 256 partial products for even and odd matrix multiplications. If DA method is used for the same conditions, we need to add 16 partial products for each row, thus there are 128 partial products for even and odd matrix multiplications. But the required hardware for the 24 ROMs and the ROM access time become an overhead that the DA method is not the most suitable for high speed processing. Table 3 compares the hardware requirement to generate the same throughput for each method in DCT/IDCT and DWT per one coefficient multiplication in 16 bit number in maximum reduced case.
From Table 2, maximum 40% reduction of partial product is performed by variable radix-2 multi-bit coding.

Table 2. The comparison of partial products

Direct Multiplication	Distributed Arithmetic	V. Radix-2 Multi-bit Coding		
		DCT	IDCT	DWT
256	128+24ROMs	120	144	30(N=6)
				21(N=4)

G. The comparison of Hardware Cost.

We verified two type of architecture for the comparison of area and gate counts. For the comparison of hardware cost in DCT/IDCT, we implemented two architectures in DCT/IDCT using each 0.35um CMOS technology. Those results are shown in Table 3. Even if these architectures depend on DA algorithm, the computational unit is common in all algorithms. Therefore, we can just compare the one element computational unit for example, computing X0. Table 3 shows the computational unit for our design gives 21% in area saving.

Table 3. The implemented results for computational unit in ROM-based DA and HDA in DCT X0 (Using 0.35um 1poly 4metal process)

	area(mm^2)	gate counts
ROM-based DA	8936	7075
HDA	8575	6812

In our DWT design, we can eliminate this ROM table. Adder-based DA can be applied in filter bank. In three stage DWT design, to perform one multiplication with filter coefficients, maximum N clock delays are needed. Therefore, all results are similar in DCT/IDCT. From N=4, we compare the number of the addition and multiplication. Table 4 is calculated that all the architectures are designed with 16 bit coefficients. In these results, we can save the maximum 256 partial products in multiplier. And we design DWT cu which has 3452 gate counts in 0.35um CMOS technology.

Table 4. Comparison of hardware cost. (N=4)

	Multiplier	Adder	Scheduling
Folded[10]	16	12	Complex
Digit[11]	14	12	Simple
Systolic[9]	24	24	Simple
Sheu[12]	16	12	Simple
Kim[13]	14	12	Simple
Proposed	0	21	Simple

Conclusion

In this paper, HDA algorithm using variable radix-2 multi-bit coding algorithm is presented and verified in image compression. We conclude that this algorithm is the most powerful for the matrix multiplication with constant coefficients. Also, in area, it makes better efficiency in the comparison of other DA algorithms. We designed DCT/IDCT and DWT using HDA resulting from variable radix-2 multi-bit coding, and orthogonal transpose memory. The gate counts of the implemented CU are 6.8K in 0.35μm CMOS technology and are reduced by 20% comparing with those of ROM-based DA in DCT/IDCT. Also, we designed DWT with same manner in DCT/IDCT and we lessened the number of partial products and substituted multiplier with the minimized data path. As a results, DWT CU has 100% utilization and simple architecture comparing with ROM-DA and Adder-based DA. The multiple constants multiplication with the large number of computation in many applications should be optimized and HDA also can be a great solution for optimal ASIC design to minimize the chip size.

References

[1] J. O. Penhollow, “Study of arithmetic recording with applications multiplication and division,” Ph.D. dissertation, Univ. Illinois, Urbana, Sept. 1962.

[2] J.E Robertson, “Theory of computer arithmetic employed in the design of the computer at the university of Illinois,” Digital Computer Lab., Uni. Illinois, Urbana, June. 1960.

[3] H. Samueli, “An improved search algorithm for optimization for the FIR coefficients represented by a canonic signed-digit code,” *IEEE Trans. Circuits Syst.*, vol.34 pp.1192-1202, 1987.

[4] Miodrag Potkonjak, Mani B. Sirvastava and Anantha P. Chandrakasan, “Multiple constant multiplications: Efficient and Versatile framework and algorithms for Exploring common subexpression elimination,” *IEEE Transactions on Computer-Aided Design of Integrated Circuit and Systems*, vol. 15, No. 2, pp 151-164, Feb. 1996.

[5] Dae Won Kim, Jun Rim Choi, “Variable radix-2 multibit coding for 400Mpixel/s DCT/IDCT of HDTV video decoder,” *Integration the VLSI Journal*, Elsevier, vol. 35 pp.47-67, 2003

[6] Stanley A white, “ Applications of Distributed Arithmetic to Digital Signal Processing: A Tutorial Review’” *IEEE ASSP Magazine, July, 1989*

[7] Y. J . Kim, Y. J . Jang, H. S. Lee, “Adder - based Distributed Arithmetic DWT Proces sor Design.” *Proc of The 28th KISS Spring Conference*, Vol. 28, no. 1, pp16- 18, Apil 2001.

[8] M.T. Sun, T.C. Chen, and A.M. Gottlieb, “VLSI implementaion of a 16x16 discret cosine transform,” *IEEE Trans. Circuits Syst.*, vol. 35, no. 4, pp. 610-617, Apr. 1989.

[9] Vishiwanata, M., Owens, R.M., and Irwin, M.J., “VLSI architectures for the discrete wavelet transform,” *IEEE Trans. CAS-II*, vol.42, no.5, pp.305-316, 1995.

[10] Knowles, G., “VLSI architecture for the discrete wavelet transform,” *Elctron. Lett*., vol.26, pp. 1184-1185, 1990.

[11] Pahri, K.K., and Nishitani, T., “VLSI architecture for the discrete wavelet transform,” *IEEE Trans. VLSI Systems*, vol.1, no.2, pp.191-202, 1993.

[12] Chung, S.F., Sheu, M.H., and Shieh, M.D., “A pipelined VLSI architecture with module structure design for the discrete wavelet transform,” *IEEE Int. Symp. Circuits & systems*, vol.4, pp.325-355, 1996.

[13] Seung-Kwon Peak and Lee-Sup Kim, “2D DWT VLSI architecture for wavelet image processing,” *Elctron. Lett*., vol.34, pp.537-538, 1998.