Iterative Radix-8 Multiplier Structure Based on a Novel Real-time CSD Recoding

Yunhua Wang[1], Linda S. DeBrunner[2], Joseph P. Havlicek[1], and Dayong Zhou[1]

[1]School of Electrical & Computer Engineering;
University of Oklahoma,
Norman, OK 73019, USA
{xiao9, joebob, dayong}@ou.edu

[2]Department of Electrical & Computer Engineering;
Florida State University
Tallahassee, FL 32310 USA
linda.debrunner@eng.fsu.edu

***Abstract* − Real-time implementation of many digital signal processing (DSP) algorithms and multimedia applications is performance limited by the available speed, energy efficiency, and area requirement of multiplication. This is exacerbated in handheld multimedia devices due to the small size and limited battery lifetimes. In our previous work, we introduce a novel canonical signed digit (CSD) iterative multiplier structure in which the conversion from 2's complement to CSD representation is implicitly implemented in real-time. In this work, we further improve the iterative multiplier performance by introducing explicit radix-8 hardware support in which the multiplier is shifted by one octal digit in each iteration as opposed to only one or two bits. Thus, this new structure further reduces the power consumption while simultaneously increasing the computational bandwidth significantly with only a small sacrifice in area consumption. This new design also uses a bypass technique to further reduce the need for devices such as carry save adder (CSA) arrays and adder trees for partial product reduction operations. Therefore, the new structure introduced here greatly improves the multiplier throughput and energy efficiency. Moreover, the number of iterations required to complete a fixed length multiply is data dependent as a result of a novel variable shifting technique; hence there is no energy and time overhead expended for unnecessary iterations as observed in multipliers where the number of iterations is fixed. Our results show that this new iterative structure delivers significant performance improvements with respect to speed, area, and power consumption relative to previous iterative multiplier designs.**

I. INTRODUCTION

Hardware implementation of digital signal processing (DSP) algorithms and multimedia applications in technologies such as field programmable gate arrays (FPGAs) and digital signal processors requires a large number of multiplications. Often, the overall performance of the design is limited by constraints on the speed, energy consumption, and area requirements of the available multiplier design options.

This is particularly true for applications centered around modern handheld multimedia devices, where physical size, chip real-estate, power, and battery life are all at a premium. Consequently, intense recent research has been focused on the development of efficient, advanced multiplier techniques to support these demanding applications [1]-[8].

Multiplication involves two basic operations: generation of partial products and accumulation of partial products. Hence, all techniques for speeding up multiplication can be categorized into to two main groups: those that seek to reduce the number of nonzero partial products and those that seek to accelerate the accumulation of partial products. The three main classes of multipliers include parallel multipliers, array multipliers and iterative multipliers [9]. Parallel multipliers generate partial products in parallel and accumulate them using a fast multi-operand adder. With this type of multiplier design, the execution speed is increased (relative to a typical iterative multiplier) at the expense of the increased area that is required for the generation of multiple partial products in parallel. Further speed up can be achieved by using an array multiplier where an array of identical cells generates new partial products and accumulates them simultaneously, such that separate circuits are not required for generation and accumulation. Array multipliers are used widely when high-speed multiplication is required. However, in addition to requiring large area, array multipliers usually do not seek to optimize energy efficiency though exploitation of the specific, data dependent patterns of digits that occur in the multiplier and multiplicand; typical array multipliers are inherently energy inefficient in this regard. These two considerations limit many practical applications of both parallel and array multipliers.

In contrast, typical iterative multipliers utilize a few hardware functional units repeatedly to generate partial products sequentially and add each newly generated product to those previously accumulated. The main characteristics of iterative multipliers are small area consumption; reduced pin count and wire length, and high clock rate. Moreover, by executing a number of iterations that are data dependent, the energy efficiency can be greatly improved relative to array and parallel multipliers. Here, *energy efficiency* refers to the energy required per operation, *e.g.*, nano-Joules/op [2]. Therefore, the choice between implementing a parallel or array multiplier as opposed to an iterative multiplier in any given design is generally a trade-off of computational speed against area requirement and energy efficiency. In this paper,

978-1-4244-2110-7/08/$25.00 ©2007 IEEE

we introduce a new iterative multiplier design that provides simple structure, high throughput, and high energy efficiency, making it particularly suitable for deployment in low power and low area applications.

In an iterative multiplier, the partial products are generated and added together sequentially or iteratively to obtain the final product using shift/add functional blocks [9]. The number of shift/add operations that occur in a given multiply is directly related to the power consumption of the circuit. Thus, the number of iterations required in a multiply directly impacts both the throughput and the energy efficiency of the multiplier. A variety of techniques have been developed to reduce the number of iterations, thereby increasing the efficiency of the shift/add operations [2], [3], [10]; detailed descriptions of several of these are given in Section II below.

Typically, the shift/add functional blocks implemented in an iterative multiplier shift the operand by a fixed number of bits and a fixed number of iterations are required to produce the final product. With this straightforward approach, the control and energy overhead required per machine cycle are constant throughout the entire multiplication operation. The new iterative multiplier structure proposed in this paper employs a variable number of iterations to convert one operand (the multiplier) from 2's complement to a Canonical Signed Digit (CSD) representation in real-time by using a new CSD recoding method where the number of iterations is data dependent. This novel real-time CSD recoding is very simple to implement and requires only a few combinational logic gates. In fact, the actual CSD representation of the multiplier operand is never explicitly present in the hardware; rather, only the control signals that are required for accumulation of the partial products based on the value of the multiplicand are generated. By exploiting the inherent properties of the CSD number system, the proposed multiplier increases the number of zero partial products to approximately 66.7%. Each time a zero partial product occurs, the corresponding add (accumulate) operation is automatically bypassed. Consequently, the number of partial product reduction operations, as are often implemented with carry save adder (CSA) arrays and/or adder trees, is greatly reduced. This in turn dramatically reduces the power consumption of the overall multiplier circuit. Compared to a typical iterative multiplier where the number of iterations is fixed, this approach saves energy and time by bypassing iterations that are not required for the generation of the product.

In this paper, we introduce a new radix-8 iterative multiplier structure that provides simple structure, high throughput, and high energy efficiency, making it particularly suitable for deployment in low power and low area applications.

As in Booth recoding, the CSD representation [11] is a radix-2 number system using the digit set $\{\bar{1}, 0, 1\}$ with the "canonical" property that no two consecutive bits in the CSD number are nonzero. This representation replaces the additions arising from a string of ones in a binary number with a single subtraction, so that the "shift-and-add" algorithm becomes "shift-and-add/subtract". CSD representations have proven to be useful in implementing multipliers with reduced complexity, because the cost of multiplication is a direct function of the number of nonzero bits in the multiplier. Under the assumption that all realizable n-bit operand values are equally likely to occur, the probability that a CSD digit b_j is nonzero is given by [6]

$$P(|b_j|=1)=1/3+(1/9n)[1-(-1/2)^n]. \qquad (1)$$

From (1), we can see that for an n-bit 2's complement multiplier the number of non-zero bits in its CSD representation never exceeds $n/2$ and can be reduced to $n/3$ on average, as the wordlength of the multiplier grows. Therefore, if we can incorporate the CSD number representation into our multiplier, we can significantly reduce the number of non-zero partial products, which in turn increases the multiplier throughput and energy efficiency.

The new real-time CSD recoding multiplier technique and structure are presented in Section II. Performance comparisons of this new multiplier against other existing iterative multipliers are given in Section III, while conclusions appear in section IV.

II. Real-time CSD Multiplier Structure

The conversion of a 2's complement binary number to CSD representation can be implemented in hardware using look-up tables [4], a canonical recoding algorithm [12], or complicated digital circuits [7], [13], but these all are costly in terms of area and power consumption. As a result, many current applications that utilize CSD representations avoid the issue of real-time 2's complement to CSD conversion by limiting CSD optimization to a fixed set of operand values that can be converted *a priori*. This allows the CSD representation to be calculated offline. These approaches can be further improved by multiple constant multiplier (MCM) techniques [14]-[16] or similar techniques. Fixed number CSD representation techniques have been applied to efficiently implement the multiplications for fixed-coefficient digital filters, but these techniques are not applicable to adaptive filters [17] and other inner-product computations in which the multipliers are not know *a priori*.

In this section, we will introduce our iterative multiplier structure, which is based on a novel real-time CSD encoding technique [18]. Instead of converting a binary number into its CSD representation, the CSD recoder in our design only generates the corresponding control signals. Controlled by these signals, the multiplier operation is based on the CSD logic. For better understanding of our method, we first present a new way to convert a 2's complement binary number to its CSD representation.

To reduce the multiple additions arising from a string of ones, we use the simple concept that $x = 2x - x$ to convert x to

another form we refer to as the difference form signed (DFS) number [13]. In the DFS representation, a number may contain instances of the digit pairs "$\bar{1}1$" and "$1\bar{1}$," but sequences of two consecutive ones or two consecutive negative one digits cannot occur. DFS conversion is illustrated in the following example:

x	1 0 1 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1	
$2x = x << 1$	1 0 1 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1 0	
$-x$	1 1 0 1 1 1 1 1 0 1 0 0 1 0 1 1 0 0 1	sign extension
x_{DFS}	$0\bar{1}1000001\bar{1}\bar{1}01\bar{1}10\bar{1}01\bar{1}$	

We now summarize some of the key properties of the DFS number representation [18].

Property 1: No two consecutive nonzero bits in the difference form of x have the same sign.

Property 2: To convert a 2's complement number x to the CSD representation, we only need to replace occurrences of the bit pair "$\bar{1}1$" with "$0\bar{1}$" and/or the bit pair "$1\bar{1}$" with "01" in the DFS representation of x starting from the least significant bit (LSB).

The DFS number is not encoded directly in the hardware circuit, since each DFS number needs twice as much memory space compared to a binary number. However, it serves as a tool to understand our real-time CSD recoding.

Therefore, as a DFS number x_{DFS} is scanned in 3-bit segments from right to left (least to most significant), there are several possible situations:

1) Whenever there are 3 zeros or the number of nonzero digits is one and that nonzero digit is not located at the beginning position (such as "001", "010", "$0\bar{1}0$" or "$00\bar{1}$"), we leave them unchanged.
2) Whenever there are 3-bit segments which contain a pair of "$\bar{1}1$" and "$1\bar{1}$" and with a '0' bit (such as "$01\bar{1}$", "$\bar{1}10$", "$1\bar{1}0$"or $0\bar{1}1$), we convert the bits based on property 2;
3) Whenever the 3-bit segments start with a nonzero bit that is followed by a '0' bit (such as "$10\bar{1}$", "100", "$\bar{1}00$ or "$\bar{1}01$"), we leave the least significant two bits unchanged and continue scanning the remaining digits in 3-bit segments.
4) Whenever there are 3-bit segments with all three digits are nonzero (such as "$1\bar{1}1$", "$\bar{1}1\bar{1}$"), we convert the least significant two bits based on property 2 and leave them unchanged, then continue scanning the remaining digits in 3-bit segments.

In this way, we obtain the CSD representation of the number x_{DFS}. The following example illustrates this algorithm for conversion of a DFS number to its CSD representation:

x_{DFS}	$0\bar{1}1\,00\,00\bar{1}\,1\bar{1}\,01\bar{1}\,10\bar{1}\,01\bar{1}$
x_{CSD}	$00\bar{1}0000\bar{1}\,01\,010\bar{1}0\bar{1}001$

In our proposed multiplier structure, we do not convert a number explicitly into the DFS or CSD representations. Instead, based on the relationship between the two's complement number and its DFS representation, as well as Properties 1 and 2, we obtain the digit-set relationships between a two's complement number, its DFS representation and its CSD representation, which provides us with the corresponding signals that are needed to control the accumulation of partial products in the multiplier. These relationships are shown in Table I, where c_1, c_2, c_3 and c_4 are control signals based on CSD number conversion. Signal c_1 is used to control the add or subtract operation, i.e. addition is performed if $c_1 = 0$ and subtraction is performed if $c_1 = 1$, Signal c_2 is used to control the number of bits that are shifted in each iteration, i.e. $c_2 = 1$ indicates a right shift by 2 bits and $c_2 = 0$ enables right shifting by 3 bits. Signal c_3 is the bypass control signal, where $c_3 = 1$ enables the bypass operation. Finally, c_4 indicates that a nonzero multiple is 2. These signals (defined in Table I) are given by (2) and may be efficiently implemented in hardware using the circuits as shown in Fig. 1.

$$\begin{aligned} c_1 &= b_{i+1}(b_i \oplus b_{i-1}) + b_{i+2}(b_{i+1} + b_i b_{i-1}) \\ c_2 &= b_{i+2} \oplus (b_{i+1} b_{i-1}) + b_{i+2} \oplus (b_{i+1} b_i) \\ c_3 &= \overline{b_{i+1}} \oplus (b_i b_{i-1}) \\ c_4 &= b_{i+1} \oplus (b_i b_{i-1}) \end{aligned} \tag{2}$$

TABLE I
RECODING SCHEME OF CSD ALGORITHM

2's Complement				DFS			CSD			Control Signals*			
b_{i+2}	b_{i+1}	b_i	b_{i-1}	b'_{i+2}	b'_{i+1}	b'_i	b''_{i+2}	b''_{i+1}	b''_i	c_1	c_2	c_3	c_4
0	0	0	0	0	0	0	0	0	0	×	0	1	0
0	0	0	1	0	0	1	0	0	1	0	0	0	0
0	0	1	0	0	1	$\bar{1}$	0	0	1	0	0	0	0
0	0	1	1	0	1	0	0	1	0	0	0	0	1
0	1	0	0	1	$\bar{1}$	0	0	1	0	0	0	0	1
0	1	0	1	1	$\bar{1}$	1	1	0	$\bar{1}$	1	1	0	0
0	1	1	0	1	0	$\bar{1}$	1	0	$\bar{1}$	1	1	0	0
0	1	1	1	1	0	0	#	0	0	×	1	1	0
1	0	0	0	$\bar{1}$	0	0	#	0	0	×	1	1	0
1	0	0	1	$\bar{1}$	0	1	$\bar{1}$	0	1	0	1	0	0
1	0	1	0	$\bar{1}$	1	$\bar{1}$	$\bar{1}$	0	1	0	1	0	0
1	0	1	1	$\bar{1}$	1	0	0	$\bar{1}$	0	1	0	0	1
1	1	0	0	0	$\bar{1}$	0	0	$\bar{1}$	0	1	0	0	1
1	1	0	1	0	$\bar{1}$	1	0	0	$\bar{1}$	1	0	0	0
1	1	1	0	0	0	$\bar{1}$	0	0	$\bar{1}$	1	0	0	0
1	1	1	1	0	0	0	0	0	0	×	0	1	0

"×" means don't care. "#" in CSD column means no CSD bit is generated and wait till next bits come.

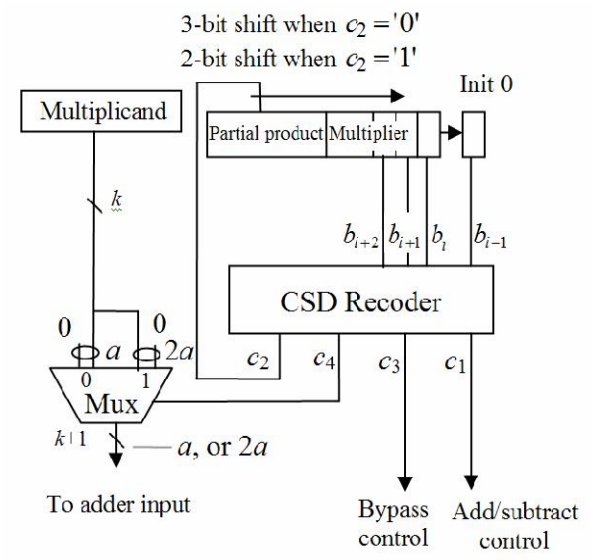

Fig.1 Real-time CSD multiplication based on our novel CSD recoder

III. Comparison with Booth Recoding and Other CSD Recoding Techniques

Compared to existing techniques of varying complexity, the new approach proposed here successfully reduces the implementation overhead by avoiding the need to explicitly represent the CSD number in hardware altogether; only the control signals derived from the DFS number concept are explicitly present in the hardware.

Iterative multipliers with radix-4 and radix-8 based on the Modified Booth's Recoding (MBR) algorithm are most commonly used in modern hardware design due to the low area requirement, low energy consumption and high throughput [9],[11]. In radix-4 MBR, sequential 3-bit segments of a 2's complement number are converted into the digit set $\{\pm 2, \pm 1, 0\}$. For an n by n bit multiplication, this technique reduces the number of partial products by 50% as compared to straightforward 2's complement based multiplication, where, moreover, approximately 25% of the partial products are zeros. In practice, a radix-4 MBR based multiplier generates on average $0.375n$ non-zero partial products. However, after the partial products are generated, they are typically all passed on to the accumulator, including those partial products that are zero. In this way, the number of arithmetic operations in the carry-save structure is not reduced. So, the zero partial products are not fully exploited to improve the multiplier performances.

Compared with a radix-4 Modified Booth's Recoding (MBR) based iterative multiplier which reduces the number of iterations to half, our proposed structure is much faster since the number of iterations has been reduced to 39.58%. Also, instead of computing the five required multiples of the multiplicand a $(0, \pm a, \pm 2a)$ required for radix-4 Booth's recoding, only $\pm a$ and $\pm 2a$ are required for our multiplier. Thus, the complexity of structure is competitive with radix-4 MBR.

Compared with a radix-8 MBR based iterative multiplier, our proposed structure is much simpler and has competitive speed. Instead of computing the nine multiples $(0, \pm a, \pm 2a, \pm 3a, \pm 4a)$ of the multiplicand a that are required for radix-8 Booth's recoding, only $\pm a$ and $\pm 2a$ are required for our multiplier. Consequently, some complex multiplexers are avoided. Furthermore, unlike the multiplier based on radix-4 and radix-8 MBR, once the zero bits in our CSD number are detected, there is no accumulation required – only shifting is required and there is no carry propagation whatsoever. Most of the time, the accumulation process is bypassed. Therefore, our algorithm reduces the latency of the operation, as well as the power consumption of the circuit.

On average, the radix-8 real-time CSD recoding multiplier proposed in this paper eliminates more than 60% of the partial product generation operations that would be required for straightforward two's complement multiplication. The quantitative performance comparisons with other multipliers in terms of the number of partial products, required multiples, non-zero multiples generated and bypassed null partial products are listed in Table II. These data indicate that the proposed multiplier structure is capable of delivering superior performance in terms of low area requirement, high energy efficiency and high throughput. Moreover, it may be conveniently implemented using either synchronized circuits or asynchronous circuits [2].

TABLE II
Complexity Comparison on Average Percentage of Data in the Traditional Multiplier, Radix-4 Booth's Recoding Multiplier, Radix-8 Booth's Recoding Multiplier and Proposed CSD Recoding Multiplier

	Total partial products	Required multiples of multiplicand a	Nonzero multiples generated	Bypassed null partial products
2's complement multiplier	100%	$0, a$	50%	0%
Radix-4 Booth's recoding	50%	$0, \pm a, \pm 2a$	37.5%	0%
Radix-8 Booth's recoding	33.3%	$0, \pm a, \pm 2a, \pm 3a, \pm 4a$	29.17%	0%
Proposed CSD recoding	39.58%	$\pm a, \pm 2a$	39.58%	60.42%

IV. Conclusions

We have presented an efficient high radix iterative multiplier structure based on a novel real-time CSD recoding circuit. Because of the iterative multiplier nature, the proposed design requires lower area compared with array

multipliers. Furthermore, the real-time CSD conversion ensures that the proposed design has the simplest structure among all radix-8 multipliers. Also, this multiplier has the minimum number of nonzero partial products based on the CSD number property. The number of add/subtract operations is further reduced through the use of bypass techniques. Thus, the complexity of the hardware implementation is dramatically reduced as compared to conventional methods, including modified Booth recoding and competing CSD recoding techniques. This approach achieves an overall speed-up as well as reduced power consumption which is particularly critical in mobile multimedia applications.

Finally, unlike other CSD number based multipliers, the structure proposed here uses real time CSD recoding, and does not require a fixed value for the multiplier input to be known *a priori*; as a result, the proposed multiplier can be used for the efficient implementation of digital filters with non-fixed filter coefficients, such as adaptive filters [17].

REFERENCES

[1] J. Kang and J. Gaudiot, "A simple high-speed multiplier design," *IEEE Trans. Comput.*, vol. 55, No. 10, pp. 1253-1258, Oct. 2006.

[2] A. Efthymiou, W. Suntiamorntut, J. Garside, and L.E.M. Brackenbury, "An asynchronous, iterative implementation of the original Booth multiplication algorithm," in *Proc. 10th IEEE Int'l. Symp. Asynchronous Circuits, and Syst.*, Crete, Greece, Apr. 19-23, 2004, pp. 207-215.

[3] J. Hensley, A. Lastra, and M. Singh, "An area- and energy-efficient asynchronous Booth multiplier for mobile devices," in *Proc. IEEE Int'l. Conf. Comput. Design*, San Jose, CA, Oct. 11-13, 2004, pp. 18-25.

[4] M.A. Soderstrand, "CSD multipliers for FPGA DSP applications," in *Proc. IEEE Int'l. Symp. Circuits, Syst.*, vol. 5, Bangkok, Thailand, May 25-28, 2003, pp. V-469 – V-472.

[5] C.-L. Chen, K.-Y. Khoo, and A.N. Willson, Jr., "A simplified signed powers-of-two conversion for multiplierless adaptive filters," in *Proc. IEEE Int'l. Symp. Circuits, Sys.*, vol. 2, Atlanta, GA, May 12-15, 1996, pp. 364-367.

[6] G.K. Ma and F.J. Taylor, "Multiplier policies for digital signal processing," *IEEE ASSP Mag.*, vol. 7, no. 1, pp. 6-20, Jan. 1990.

[7] G.A. Ruiz and M.A. Manzano, "Self-timed multiplier based on canonical signed-digit recoding," *IEE Proc.: Circuits, Devices, Syst.*, vol. 148, no. 5, pp. 235-241, Oct. 2001.

[8] S.-M. Kim, J.-G. Chung, and K.K. Parhi, "Design of low error CSD fixed-width multiplier," in *Proc. 2002 IEEE Int'l. Symp. Circuits, Syst.*, vol. 1, Scottsdale, AZ, May 26-29, 2002, pp. I-69 – I-72.

[9] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Designs*, Oxford Press, London, 1999.

[10] A.D. Booth, "A signed binary multiplication technique," *Quartly J. Mechanics, Appl. Math.*, vol. 4, no. 2, pp. 236-240, Jun. 1951.

[11] P. Pirsch, *Architectures for Digital Signal Processing*, John Wiley & Sons, West Sussex, UK, 1998.

[12] I. Koren, *Computer Arithmetic Algorithms*, 2nd ed., Prentice Hall, 2001.

[13] Y. Wang, L.S. DeBrunner, D. Zhou, and V.E. DeBrunner, "A novel hardware implementation method for adaptive filter coefficients," in *Proc. IEEE Int'l. Conf. Acoust., Speech, Signal Proc.*, Honolulu, Hawaii, Apr. 15-20, 2007, *to appear*.

[14] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc.: Circuits, Devices, Syst.*, vol. 141, no. 5, pp. 407-413, Oct. 1994.

[15] Y. Voronenko and M. Püschel, "Multiplierless multiple constant multiplication," *ACM Transactions on Algorithms,* vol. 3, iss. 2, article no. 11, May 2007.

[16] Oscar Gustafsson, "A Difference Based Adder Graph Heuristic for Multiple Constant Multiplication Problems," *Int'l Symp. Circuits and Syst.,* New Orleans, LA, pp. 1097-1100, May 2007.

[17] S. Haykin, *Adaptive Filter Theory*, 4th ed., Prentice Hall, Upper Saddle River, NJ, 2001.

[18] Y. Wang, L.S. DeBrunner, D. Zhou, and V.E. DeBrunner, "A Multiplier Structure Based on a Novel Real-Time CSD Recoding," in Proc. *IEEE Int'l. Symp. Circuits, Syst.,* New Orleans, LA, May 2007.

[19] M.D. Ercegovac and T. Lang, *Digital Arithmetic*, Morgan Kaufmann, Los Altos, CA, 2003.