INTERNATIONAL JOURNAL OF RESEARCH
In Advanced Engineering Technologies

Volume 4, Issue 2 JULY 2015

Optimized Modified Booth Recorder for Efficient Design of the Operator

[1] **DEEPTHI MALLELA**,M.TECH.,VLSI
[2] **VEERABABU A**,ASSISTANT PROFESSOR,DEPT.OF ECE
[3] SRINIVASULU K,HOD,DEPT. OF ECE
[1,2,3]NARASIMHA REDDY ENGINEERING COLLEGE

ABSTRACT:

The most effective way to increase the speed of a multiplier is to reduce the number of the partial products because multiplication precedes a series of additions for the partial products. To reduce the number of calculation steps for the partial products, MBA algorithm has been applied mostly where CSA has taken the role of increasing the speed to add the partial products. To increase the speed of the MBA algorithm, many parallel multiplication architectures have been researched. A modified booth multiplier has been designed which provides a flexible arithmetic capacity and a tradeoff between output precision and power consumption due to using of SPST architecture. Moreover, the ineffective circuitry can be efficiently deactivated, thereby reducing power consumption and increasing speed of operation. The experimental results have shown that the proposed multiplier outperforms the conventional multiplier in terms of power and speed of operation. In this paper we used Xilinx-ISE tool for logical verification, and further synthesizing it on Xilinx -ISE tool using target technology and performing placing & routing operation for system verification.

Keywords: Computer arithmetic, multiplication by constants, common sub expressions sharing, Add-Multiply operation, arithmetic circuits, Modified Booth recoding, VLSI design.

I. INTRODUCTION

In this paper, we study the various parallel MAC architectures and then implement a design of parallel MAC based on some booth encodings such as radix-2 booth encoder and some final adders such as CLA, Kogge stone adder and then compare their performance characteristics. In general, a multiplier uses Booth algorithm and an array of full adders, this multiplier mainly consists of three parts Wallace tree, to add partial products, booth encoder and final adder. A Digital multiplier is the fundamental component in general purpose microprocessor and in DSP [1]. Most of the DSP methods use discrete cosine transformations in discrete wavelet transformations. Compared with many other arithmetic operations multiplication is time consuming and power hungry. Thus enhancing the performance and reducing the power dissipation are the most important design challenges for all applications in which multiplier unit dominate the system performance and power dissipation. The one most effective way to increase the speed of a multiplier is to reduce the number of the partial products. Although the number of partial products can be reduced with a higher radix booth encoder, but the number of hard multiples that are expensive to generate also increases simultaneously. To increase the speed and performance, many parallel MAC architectures have been proposed. Parallelism in obtaining partial products is the most common technique used in this architecture. There are two common approaches that make use of parallelism to enhance the multiplication performance. The first one is reducing the number of partial product rows and second one is the carry-save-tree technique to reduce multiple partial product rows as two "carry-save" redundant forms. An architecture was proposed in [2] to provide the tact to merge the final adder block to the accumulator register in the MAC operator to provide the possibility of using two separate N/2-bit adders instead of one N-bit adder to accumulate the N–bit MAC results. The most advanced types of MAC has been proposed by Elguibaly in which accumulation has been

combined with the carry save adder (CSA) tree that compresses partial products and thus reduces the critical path. While it has better performance as compared to the previous MAC architectures. Later on a new architecture for a high-speed MAC is proposed by Seo and Kim. The difference between the two is that the latest one carries out the accumulation by feeding back the final CSA output rather than the final adder results.

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, Subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers, the operand usually contains the same number of bits. When the operands are in terpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them.

II. DIFFERENT MULTIPLIERS

Binary Multiplication

In the binary number system the digits, called bits, are limited to the set. The result of multiplying any binary number by a single binary bit is either 0, or the original number. This makes forming the intermediate partial-products simple and efficient. Summing these partial-products is the time consuming task for binary multipliers. One logical approach is to form the partial-products one at a time and sum them as they are generated. Often implemented by software on processors that do not have a hardware multiplier, this technique works fine, but is slow because at least one machine cycle is required to sum each additional partial-product. For applications where this approach does not provide enough performance, multipliers can be implemented directly in hardware.

Hardware Multipliers

Direct hardware implementations of shift and add multipliers can increase performance over software synthesis, but are still quite slow. The reason is that as each additional partial -product is summed a carry must be propagated from the least significant bit (LSB) to the most significant bit (MSB). This carry propagation is time consuming, and must be repeated for each partial product to be summed. One method to increase multiplier performance is by using encoding techniques to reduce the the number of partial products to be summed. Just such a technique was first proposed by Booth [BOO 511. The original Booth‟s algorithm ships over contiguous strings of l‟s by using the property that: 2” + 2(n-1) + 2(n-2) + . . . + 2hm) = 2(n+l) - 2(n-m). Although Booth’s algorithm produces at most N/2 encoded partial products from an N bit operand, the number of partial products produced varies. This has caused designers to use modified versions of Booth‟s algorithm for hardware multipliers. Modified 2-bit Booth encoding halves the number of partial products to be summed.Since the resulting encoded partial-products can then be summed using any suitable method, modified 2 bit Booth encoding is used on most modern floating-point chips LU 881, MCA 861. A few designers have even turned to modified 3 bit Booth encoding, which reduces the number of par tial products to be summed by a factor of three IBEN 891. The problem with 3 bit encoding is that the carry-propagate addition required to form the 3X multiples often overshadows the potential gains of 3 bit Booth encoding. To achieve even higher performance advanced hardware multiplier architectures search for faster and more efficient methods for summing the partial-products. Most increase performance by eliminating the time consuming carry propagate additions. To accomplish this, they sum the partial -products in a redundant number representation. The advantage of a redundant representation is that two numbers, or partial

-products, can be added together without propagating a carry across the entire width of the number. Many redundant number representations are possible. One commonly used representation is known as carry-save form. In this redundant representation two bits, known as the carry and sum, are used to represent each bit position. When two numbers in carry -save form are added together any carries that result are never propagated more than one bit position. This makes adding two numbers in carry-save form much faster than adding two normal binary numbers where a carry may propagate. One common method that has been developed for summing rows of partial products using a carry-save representation is the array multiplier.

High-Speed Booth Encoded Parallel Multiplier Design:

Fast multipliers are essential parts of digital signal processing systems. The speed of multiply operation is of great importance in digital signal processing as well as in the general purpose processors today, especially since the media processing took off. In the past multiplication was generally implemented via a sequence of addition, subtraction, and shift operations. Multiplication can be considered as a series of repeated additions. The number to be added is the multiplicand, the number of times that it is added is the multiplier, and the result is the product. Each step of addition generates a partial product. In most computers , the operand usually contains the same number of bits. When the operands are interpreted as integers, the product is generally twice the length of operands in order to preserve the information content. This repeated addition method that is suggested by the arithmetic definition is slow that it is almost always replaced by an algorithm that makes use of positional representation. It is possible to decompose multipliers into two parts. The first part is dedicated to the generation of partial products, and the second one collects and adds them. The basic multiplication principle is two fold i.e. evaluation of partial products and accumulation of the shifted partial products. It is performed by the successive additions of the columns of the shifted partial product matrix. The „multiplier" is successfully shifted and gates the appropriate bit of the „multiplicand". The delayed, gated instance of the multiplicand must all be in the same column of the shifted partial product matrix. They are then added to form the product bit for the particular form. Multiplication is therefore a multi operand operation. To extend the multiplication to both signed and unsigned.

III. Motivation and fused AM Implementation

A. Motivation

In this paper, we focus on AM units which implement the operation Z=X. (A+B). The conventional design of the AM operator (Fig. 1(a)) requires that its inputs A and B are first driven to an adder and then the input X and the sum Y=A+B are driven to a multiplier in order to get Z. The drawback of using an adder is that it inserts a significant delay in the critical path of the AM. As there are carry signals to be propagated inside the adder, the critical path depends on the bit-width of the inputs. In order to decrease this delay, a Carry-Look-Ahead (CLA) adder can be used which, however, increases the area occupation and power dissipation. An optimized design of the AM operator is based on the fusion of the adder and the MB encoding unit into a single data path block (Fig. 1(b)) by direct recoding of the sum Y=A+B to its MB representation. The fused Add-Multiply (FAM) component contains only one adder at the end (final adder of the parallel multiplier). As a result, significant area savings are observed and the critical path delay of the recoding process is reduced and decoupled from the bit-width of its inputs. In this work, we present a new technique for direct recoding of two numbers in the MB representation of their sum.

B. Review of the Modified Booth Form

Modified Booth (MB) is a prevalent form used in multiplication [15], [20], [24]. It is a redundant signed-digit radix-4 encoding technique. Its main advantage is that it reduces by half the number of partial products in

multiplication comparing to any other radix-2 representation.

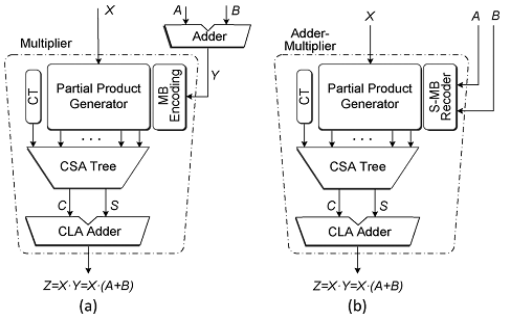


Fig. 1. AM operator based on the (a) conventional design and (b) fused designwith direct recoding of the sum of A and B in its MB representation. The multiplier is a basic parallel multiplier based on the MB algorithm. The terms CT,CSA Tree and CLA Adder are referred to the Correction Term, the Carry-Save Adder Tree and thefinal Carry-Look-Ahead Adder of the multiplier.

Let us consider the multiplication of 2's complement numbers X and Y with each number consisting of $n = 2k$ bits. .The multiplicand Y can be represented in MB form as:

$$Y = \langle y_{n-1} y_{n-2} \cdots y_1 y_0 \rangle_{2's} = y_{2k-1} \cdot 2^{2k-1} + \sum_{i=0}^{2k-2} y_i \cdot 2^i - \langle y_{k-1}^{MB} y_{k-2}^{MB} \cdots y_1^{MB} y_0^{MB} \rangle_{MB} - \sum_{j=0}^{k-1} y_j^{MB} \cdot 2^{2j} \quad (1)$$

$$y_j^{MB} = -2y_{2j+1} + y_{2j} - y_{2j-1}. \quad (2)$$

C. FAM Implementation:

In the FAM design presented in Fig. 1(b), the multiplier is a parallel one based on the MB algorithm. Let us consider the product X.Y. The term $Y = \langle y_{n-1} y_{n-2} \cdots y_1 y_0 \rangle_{2's}$ is encoded based on the MB algorithm (Section II.B) and multiplied with $X = \langle x_{n-1} x_{n-2} \cdots x_1 x_0 \rangle_{2's}$.Both X and Y consist of n=2k bits and are in 2's complement form. Equation (4) describes the generation of the k partial products:

$$PP_j = X \cdot y_j^{MB} = \overline{p}_{j,n} 2^n + \sum_{i=0}^{n-1} p_{j,i} \cdot 2^i.$$

The generation of the i-th bit of the $P_{j,i}$ partial product PP_j is based on the next logical expression while Fig. 3 illustrates its implementation at gate level

$$p_{j,i} = ((x_i \ominus s_j) \wedge one_j) \vee ((x_{i-1} \ominus s_j) \wedge two_j).$$

TABLE I
MODIFIED BOOTH ENCODING TABLE.

Binary			y_j^{MB}	MB Encoding			Input Carry
y_{2j+1}	y_{2j}	y_{2j-1}		sign=s_j	×1=one_j	×2=two_j	$c_{in,j}$
0	0	0	**0**	0	0	0	0
0	0	1	**+1**	0	1	0	0
0	1	0	**+1**	0	1	0	0
0	1	1	**+2**	0	0	1	0
1	0	0	**-2**	1	0	1	1
1	0	1	**-1**	1	1	0	1
1	1	0	**-1**	1	1	0	1
1	1	1	**0**	1	0	0	0

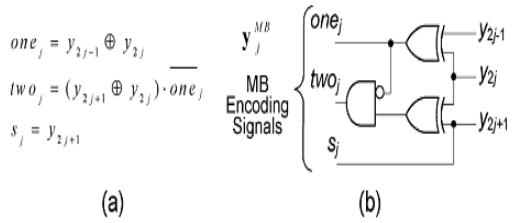


Fig. 2. (a) Boolean equations and (b) gate-level schematic for the implementation of the MB encoding signals.

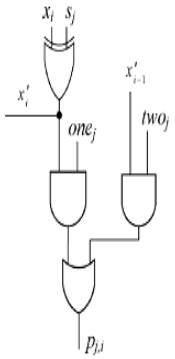


Fig. 3. Generation of the i-th bit $p_{i,j}$ of the partial product PP_j for the conventional MB multiplier.

For the computation of the least and the most significant bits of the partial product we consider and respectively. Note that in case that , the number of the resulting partial products is and the most significant MB digit is formed based on sign extension of the initial 2's complement number.

After the partial products are generated, they are added, properly weighted, through a Wallace Carry-Save Adder (CSA) tree along with the Correction Term (CT) which is given by the following equations:

$$Z = X \cdot Y = CT + \sum_{j=0}^{k-1} PP_j \cdot 2^{2j}$$

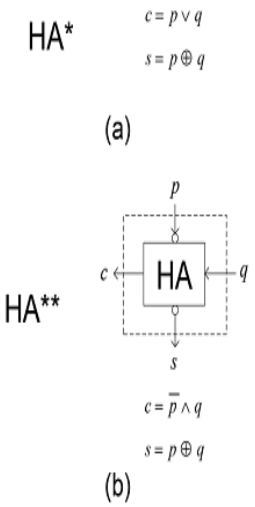


Fig. 4. Boolean equations and schematics for signed (a) HA* and (b) HA**

$$CT = CT(low) + CT(high) =$$
$$= \sum_{j=0}^{k-1} c_{in,j} \cdot 2^{2j} + 2^n \left(1 + \sum_{j=0}^{k-1} 2^{2j+1} \right) \quad (1$$

where $c_{in,j} = (one_j \vee two_j) \wedge s_j$ (see Table I).

Finally, the carry-save output of the Wallace CSA tree is leaded to a fast Carry Look Ahead (CLA) adder to form the final result Z = X . Y as shown in Fig.1 (b).

IV. SUM TO MODIFIED BOOTHR ECODING TECHNIQUE(S-MB)

A. Defining Signed-Bit Full Adders and Half Adders for Structured Signed Arithmetic

In S-MB recoding technique, we recode the sum of two consecutive bits of the input A(a2j,a2j+1) with two consecutive bits of the input B(b2j,b2j+1 into one MB digit Y_J^{MB}. As

we observe from (2), three bits are included in forming a MB digit. The most significant of them is negatively weighted while the two least significant of them have positive weight. Consequently, in order to transform the two aforementioned pairs of bits in MB form we need to use signed-bit arithmetic. For this purpose, we develop a set of bit-level signed Half Adders (HA) and Full Adders (FA) considering their inputs and outputs to be signed.

More specifically, in this work, we use two types of signed HAs which are referred as HA* and HA**. Tables II - IV are their truth tables and in Fig. 4 we present their corresponding Boolean equations. Considering that p,q are the binary inputs C and S , are the outputs (carry and sum respectively) of a HA* which implements the relation 2.C-S = p + q. where the sum is considered negatively signed (Table II, Fig. 4(a)), the output takes one of the values {0,+1,+2} In Table III, we also describe the dual implementation of HA* where we inversed the signs of all inputs and outputs and, consequently, changed the output values to {-2,-1,0}. Table IV and Fig. 4(b) show theoperation and schematic of HA** which implements the relation2.C-S = -p+q and manipulates a negative (p)and a positive (q) input resulting in the output value{-1,0,+1}.

Also, we design two types of signed FAs which are presented in Table V and VI and Fig. 5. The schematics drawn in Fig. 5(a) and (b) show the relation of FA* and FA** with the conventional FA.

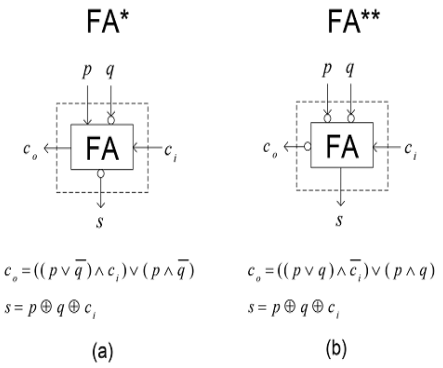


Fig. 5. Boolean equations and schematics for signed (a) FA* and (b) FA**.

B. Proposed S-MB Recoding Techniques

We use both conventional and signed HAs and FAs of Section III.A in order to design and explore three new alternative schemes of theS-MBrecoding technique. Each of the three schemes can be easily applied in either signed (2's complement representation) or unsigned numbers which consist of odd or even number of bits

In all schemes we consider that both inputs A and B are in 2's complement form and consist of 2k bits in case of even or 2k+1 bits in case of odd bit-width. Targeting to transform the sum of A and B (Y = A + B) in its MB representation, we consider the bits a2j,a2j+1 and b2j,b2j+1, as the inputs of the j recoding cell in order to get at its output the three bits that we need to form the MB digit y_j^{MB} according to (2).

1) S-MB1 Recoding Scheme: The first scheme of the proposed recoding technique is referred asS-MB1 and is illustrated in detail in Fig. 6 for both even (Fig. 6(a)) and odd (Fig. 6(b)) bit-width of input numbers. As can be seen in Fig. 6, the sum of A and B is given by the next relation:

$$Y = A + B = \mathbf{y}_k \cdot 2^{2k} + \sum_{j=0}^{k-1} \mathbf{y}_j^{MB} \cdot 2^{2j}$$

$$\text{where} \quad \mathbf{y}_j^{MB} = -2s_{2j+1} + s_{2j} + c_{2j}. \qquad (8)$$

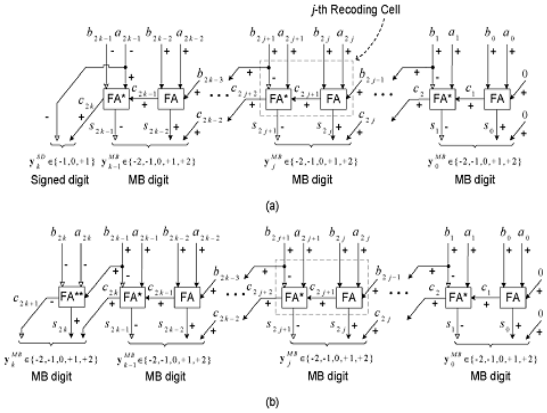


Fig. 6. *S-MB*1 recoding scheme for (a) even and (b) odd number of bits.

TABLE II
HA* BASIC OPERATION.

Inputs		Output Value[1]	Outputs	
p (+)	q (+)		c (+)	s (-)
0	0	0	0	0
0	1	+1	1	1
1	0	+1	1	1
1	1	+2	1	0

[1] $Output\,Value = 2 \cdot c - s = p + q$

TABLE III
HA* DUAL OPERATION.

Inputs		Output Value[2]	Outputs	
p (-)	q (-)		c (-)	s (+)
0	0	0	0	0
0	1	-1	1	1
1	0	-1	1	1
1	1	-2	1	0

[2] $Output\,Value = -2 \cdot c + s = -p - q$

TABLE IV
HA** OPERATION.

Inputs		Output Value[3]	Outputs	
p (-)	q (+)		c (+)	s (-)
0	0	0	0	0
0	1	+1	1	1
1	0	-1	0	1
1	1	0	0	0

[3] $Output\,Value = 2 \cdot c - s = -p + q$

TABLE V
FA* OPERATION.

Inputs			Output Value[1]	Outputs	
p (+)	q (-)	c_i (+)		c_o (+)	s (-)
0	0	0	0	0	0
0	0	1	+1	1	1
0	1	0	-1	0	1
0	1	1	0	0	0
1	0	0	+1	1	1
1	0	1	+2	1	0
1	1	0	0	0	0
1	1	1	+1	1	1

[1] $Output\,Value = 2 \cdot c_o - s = p - q + c_i$

TABLE VI
FA** OPERATION.

Inputs			Output Value[2]	Outputs	
p (-)	q (-)	c_i (+)		c_o (-)	s (+)
0	0	0	0	0	0
0	0	1	+1	0	1
0	1	0	-1	1	1
0	1	1	0	0	0
1	0	0	-1	1	1
1	0	1	0	0	0
1	1	0	-2	1	0
1	1	1	-1	1	1

[2] $Output\,Value = -2 \cdot c_o + s = -p - q + c_i$

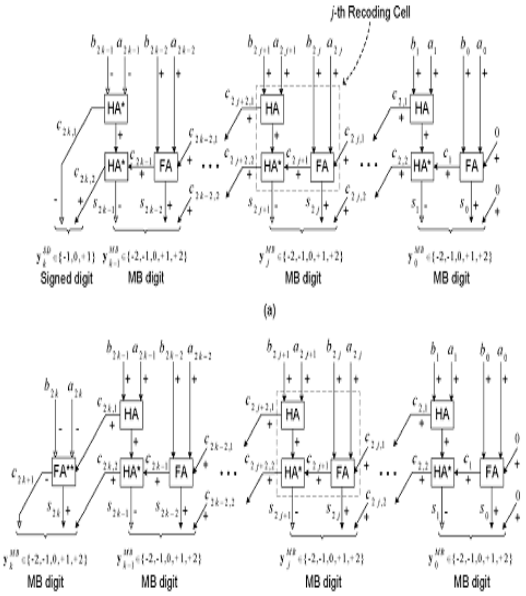


Fig. 7. S-MB2 recoding scheme for (a) even and (b) odd number of bits.

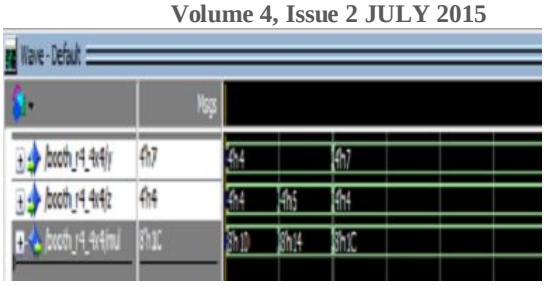
V. SIMULATION RESULTS

The simulation of the program is done using ModelSim tool and Xilinx ISE Design Suite 14.2. The results for the multiplication of 4x4 and 8x8 using Modified Booth Multiplier is shown in this section.The simulation results of 4x4 bit Modified Booth Multiplier in terms of number of occupied slices, number of 4 inputs LUT and IOBs are shown in Table III. The simulation results of 8x8 bit Modified Booth Multiplier in terms of number of occupied slices, number of 4 inputs LUT and IOBs are shown in Table IV.

A. Using ModelSim

1) 4x4 bit

Figure 8. 4x4 bit Modified Booth Multiplication using ModelSim



2) 8x8 bit

Figure 9. 8x8 bit Modified Booth Multiplication using ModelSim

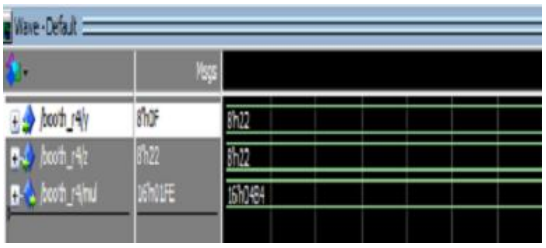


TABLE III. XILINX RESULTS FOR 4X4 BIT MODIFIED BOOTH MULTIPLIER

PARAMETER	Used	Available	Utilization
Number of 4 input LUTs	25	138240	1%
Number of occupied Slices	9	34560	1%
Number of bonded IOBs	16	800	2%

TABLE IV. XILINX RESULTS FOR 8X8 BIT MODIFIED BOOTH MULTIPLIER

PARAMETER	Used	Available	Utilization
Number of 4 input LUTs	153	63400	1%
Number of occupied Slices	43	15850	1%
Number of bonded IOBs	32	210	15%

VI. CONCLUSION

This paper focuses on optimizing the design of the Fused-Add Multiply (FAM) operator. We propose a structured technique for the direct recoding of the sum of two numbers to its MB form. We explore three alternative designs of the proposed S-MB recoder and compare them to the existing ones [12], [13] and [23].The proposed recoding schemes, when they are incorporated inFAM designs, yield considerable performance improvements incomparison with the most efficient recoding schemes found in literature.

REFERENCES

[1] D.J. Magenheimer, L. Peters, K.W. Pettis, and D. Zuras, "IntegerMultiplication and Division on the HP Precision Architecture,"IEEE Trans. Computers,vol. 37, no. 8, pp. 980-990, Aug. 1988.

[2] A.D. Booth, "A Signed Binary Multiplication Technique,"Quarterly J. Mechanical Applications of Math.,vol. IV, no. 2, pp. 236-240,1951.

[3] R. Bernstein, "Multiplication by Integer Constants,"Software—Practice and Experience,vol. 16, no. 7, pp. 641-652, July 1986.

[4] N. Boullis and A. Tisserand, "Some Optimizations of Hardware Multiplication by Constant Matrices,"Proc. 16th IEEE Symp.Computer Arithmetic (ARITH 16),J. -C. Bajard and M. Schulte, eds.,pp. 20-27, June 2003.

[5] M. Potkonjak, M.B. Srivastava, and A.P. Chandrakasan, "Multiple Constant Multiplications: Efficient and Versatile Framework and Algorithms for Exploring Common Subexpression Elimination," IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,vol. 15, no. 2, pp. 151-165, Feb. 1996.

[6] M.D. Ercegovac and T. Lang,Digital Arithmetic.Morgan Kaufmann, 2003.

[7] M.J. Flynn and S.F. Oberman,Advanced Computer Arithmetic Design.Wiley-Interscience, 2001.

[8] R.I. Hartley, "Subexpression Sharing in Filters Using Canonic Signed Digit Multipliers," IEEE Trans. Circuits and Systems II:Analog and Digital Signal Processing,vol. 43, no. 10, pp. 677-688,Oct. 1996.

[9] K.D. Chapman, "Fast Integer Multipliers Fit in FPGAs,"EDN Magazine,May 1994.

[10] S. Yu and E.E. Swartzlander, "DCT Implementation with Distributed Arithmetic,"IEEE Trans. Computers, vol. 50, no. 9, pp. 985-991, Sept. 2001.

[11] P. Boonyanant and S. Tantaratana, "FIR Filters with Punctured Radix-8 Symmetric Coefficients: Design and Multiplier-Free Realizations,"Circuits Systems Signal Processing,vol. 21, no. 4, pp. 345-367, 2002.