Modified Booth 1's Complement and Modulo 2^n-1 Multipliers.

C. Efstathiou[1] & H. T. Vergos[2,3]

[1] *Department of Informatics, TEI of Athens, Ag. Spyridonos St., 12210 Egaleo, Athens, Greece.*
[2] *Computer Engineering & Informatics Dept., University of Patras, 26 500 Greece*
[3] *Computer Technology Institute, 3 Kolokotroni Str., 262 21 Patras, Greece*

Abstract

In this paper we derive a novel modified Booth multiplier architecture which is based on 1's complement arithmetic. We also extend our theory to the design of modulo 2^n-1 multipliers. The proposed 1's complement modified Booth multipliers have an execution latency which is approximately the same as that offered by their 2's complement counterparts with a completely regular structure. Therefore, pipelined implementations of them can be derived in a straightforward manner. The proposed modified Booth modulo 2^n-1 multipliers can find great applicability in Residue Number System applications.

1. Introduction

Multipliers are met in almost all modern special and general purpose processors. They can be implemented either as strictly serial or serial-parallel or all parallel [1]. The advances in VLSI implementation technology, as well as the need for higher throughput and shorter latencies have made the parallel multipliers the only attractive alternative for contemporary integrated circuits. In a parallel multiplier all partial products are generated in parallel and then added successively until the final product is derived. The techniques that have been presented for speeding up the multiplication process aim towards two main directions :

1) The reduction of the number of the generated partial products. The Modified Booth algorithm [2, 3] reduces the number of the partial products that need to be added to approximately the half.
2) The speed-up of the summation of the partial products. Carry–Save Adder (CSA) arrays, Wallace trees [4] and Dadda parallel counters [5] are among the most commonly used structures.

However, when the multiplier operands are wide enough (for example in floating-point multiplication) the above techniques may not be adequate for effective reduction of the multiplier's latency.

For spreading and "hiding" the latency of the multiplication over more CPU cycles pipelining is commonly used. However, the intrinsic architecture of the modified Booth algorithm which is based on 2's complement arithmetic is not suitable for pipelined implementations [6]. The main reasoning behind this claim is the 2's complement nature of the algorithm. That is, the generation of the 2's complement of a number A requires its inversion and the addition of 1. The addition of 1 at each stage may make the time to generate the partial products unaffordable. Therefore, a common solution often employed when a multioperand CSA is used, is to pass the addition of these 1s to the final stage adder. The resulting architecture however, is hard to pipeline without significant area and performance penalties. The problem of pipelining a Booth multiplier is examined in [6] but focus is mostly given on the power - performance ratio. When Wallace trees or Dadda counters are used for the addition of the partial products, the resulting multiplier can neither be easily pipelined nor has a regular layout.

Motivated by the above, we present a new modified Booth architecture for 1's complement arithmetic. Our design is based on the following observations :

- In 1's complement arithmetic the addition of 1 is not required to get the complement of a number and
- 1's complement addition can be done as fast as that of 2's complement arithmetic [7, 8].

The proposed 1's complement multipliers have completely regular structure and can be pipelined straightforwardly. Pipelined versions of the proposed 1's complement multipliers can be used as a core for the multiplication of floating-point numbers. In the vast majority of modern computer systems the IEEE standard [9] for the representation of floating-point numbers is used. Since each mantissa in this representation is at least 23 bits wide the execution latency of the multiplier will be long even when the speed up techniques mentioned earlier are used. Therefore, a simple to derive pipelined implementation of the proposed 1's complement Booth multipliers can be used effectively in this case.

Modulo arithmetic is used in several applications of the computer systems. Modulo 2^n-1 multipliers are used today in digital signal processors which are based on the residue number system (RNS) [10-12] and cryptography [13, 14]. For modulo multiplication various ROM-based solutions using look-up tables have been proposed and compared [15]. Modulo multipliers which are based on CSA or adder trees have been presented in [16]. The only attempt for applying a Booth algorithm to modulo 2^n-1 multipliers was presented in [17].

In this paper we give the formal foundation for modified Booth modulo 2^n-1 multipliers as well as indicative results when these are used in RNS arithmetic. In the RNS system multiplication is performed in parallel among three channels that usually are chosen to perform modulo 2^n, 2^n–1 and 2^n+1 multiplication. In such a system, the proposed multipliers are highly applicable.

2. Preliminaries

In an unsigned representation the n-bit binary vector $x_{n-1}x_{n-2} \ldots x_1x_0$ represents the integer value $X = \sum x_i 2^i$

0-7803-6542-9/00/$10.00 © 2000 IEEE

which lies within the range [0, 2^n-1]. In a modulo 2^n-1 number system the same vector represents the value $|X|_{2^n-1} = |\sum x_i 2^i|_{2^n-1}$ which lies within the range [0, 2^n-2]. In the second case, the number zero has two representations, namely the all 0 and the all 1 vector. In 1's complement representation the (n+1)-bit binary vector $x_n x_{n-1} x_{n-2} \ldots x_1 x_0$ represents the value $X = -x_n(2^n-1) + \sum x_i 2^i \in [-(2^n-1), +(2^n-1)]$, whereas in 2's complement representation, the same vector represents the value $X = -x_n 2^n + \sum x_i 2^i \in [-2^n, +(2^n-1)]$.

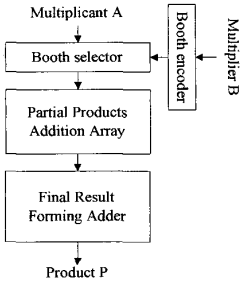

Fig. 1. Modified Booth multiplier's block diagram

Fig. 1 presents a block diagram of any multiplier implementing the modified Booth's algorithm. In the widely used 2-bit recoding form of the algorithm successive triplets of the bits of multiplier B are examined and encoded by the Booth encoder block of Fig. 1 as an element of the set {-2, -1, 0, +1, +2}. The encoded information along with the multiplicand A are then used for forming the partial products by the Booth selector block. These partial products are then reduced to two by the Partial Products Addition Array block. The Final Result Forming Adder adds the two remaining quantities and produces the final result.

3. 1's complement modified Booth multipliers

For introducing our 1's complement modified Booth multipliers we will show that the number of partial products in a 1's complement arithmetic multiplication can be reduced. To this end we utilize the following Lemma (the proofs of the Lemmas are given in [18]):

Lemma 1. Let $A = a_n a_{n-1} a_{n-2} \ldots a_1 a_0$ be a signed binary number in 1's complement representation. Then :

$$A2^i = a_n a_{n-1} a_{n-2} \ldots a_1 a_0 \underbrace{a_n a_n \ldots a_n}_{i \text{ bits}}$$

Consider the signed binary numbers $A = a_n a_{n-1} a_{n-2} \ldots a_1 a_0$ and $B = b_n b_{n-1} b_{n-2} \ldots b_1 b_0$ in 1's complement representation to be multiplied (a_n, b_n denote the sign of the corresponding operand). Let A be the multiplicand and B the multiplier. The latter can be expressed as :

$$\begin{aligned}
B &= -b_n(2^n-1) + b_{n-1}2^{n-1} + \ldots + b_2 2^2 + b_1 2 + b_0 = \\
&= -b_n(2^{n+1}-1) + b_n 2^n + b_{n-1}2^{n-1} + \ldots + b_2 2^2 + b_1 2 + b_0 = \\
&= -b_n 2^{n+1} + b_n 2^n + b_{n-1}2^{n-1} + \ldots + b_1 2 + b_0 + b_n = \\
&= -b_n 2^{n+1} + b_n 2^n + b_{n-1}2^n - b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + \\
&\quad + b_1 2 + b_0 + b_n = \\
&= 2^n(b_n + b_{n-1} - 2b_n) - b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + \ldots + b_0 + b_n = \\
&= 2^n(b_n + b_{n-1} - 2b_n) + (-b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + b_{n-3}2^{n-2}) - \\
&\quad b_{n-3}2^{n-3} + \ldots + b_1 2 + b_0 + b_n = \\
&= 2^n(b_n + b_{n-1} - 2b_n) + 2^{n-2}(b_{n-2} + b_{n-3} - 2b_{n-1}) - b_{n-3}2^{n-3} + \\
&\quad + \ldots + b_1 2 + b_0 + b_n = \\
&= \ldots = \\
&= 2^n(b_n + b_{n-1} - 2b_n) + 2^{n-2}(b_{n-2} + b_{n-3} - 2b_{n-1}) + \ldots + \\
&\quad + (b_0 + b_n - 2b_1)
\end{aligned}$$

That is, $B = \sum_i 2^{2i}\left(b_{2i-1} + b_{2i} - 2b_{2i+1}\right)$, where b_{n+1}, $b_{-1} = b_n$ and $(b_{2i-1} + b_{2i} - 2b_{2i+1})$ can be viewed as an element of the set {-2, -1, 0, +1,+2}.

According to the above, the product AB can be expressed as : $AB = A\sum_i 2^{2i}\left(b_{2i-1} + b_{2i} - 2b_{2i+1}\right) =$

$= \sum_i A\,2^{2i}\left(b_{2i-1} + b_{2i} - 2b_{2i+1}\right)$. That is, $AB = \sum_i PP_i$ (1)

where $PP_i = A2^{2i}(b_{2i-1} + b_{2i} - 2b_{2i+1})$ are the partial products which are derived according to Lemma 1 and summarized in Table I (x' is used to denote the complement of bit x). Relation (1) reveals that a modified Booth 1's complement multiplier can be designed following the block diagram of Fig. 1.

Table I. Recoded partial products

b_{2i+1}	b_{2i}	b_{2i-1}	PP_i	Partial Product (PP_i)
0	0	0	0	000 ... 000 or 111...111
0	0	1	$+A2^{2i}$	$a_n \ldots a_0\ a_n a_n \ldots a_n$ (2i trailing a_n bits)
0	1	0	$+A2^{2i}$	$a_n \ldots a_0\ a_n a_n \ldots a_n$ (2i trailing a_n bits)
0	1	1	$+2A2^{2i}$	$a_n \ldots a_0\ a_n a_n \ldots a_n a_n$ (2i +1 trailing a_n)
1	0	0	$-2A2^{2i}$	$a'_n \ldots a'_0 a'_n a'_n \ldots a'_n a'_n$ (2i +1 trailing a'_n)
1	0	1	$-A2^{2i}$	$a'_n \ldots a'_0 a'_n a'_n \ldots a'_n$ (2i trailing a'_n)
1	1	0	$-A2^{2i}$	$a'_n \ldots a'_0 a'_n a'_n \ldots a'_n$ (2i trailing a'_n)
1	1	1	0	111...111 or 000 ... 000

Area and time efficient implementations of the encoder and the selector circuits can be derived when a 3-bit bus approach is used for their interconnection. Figs 2a and 2b present sample implementations.

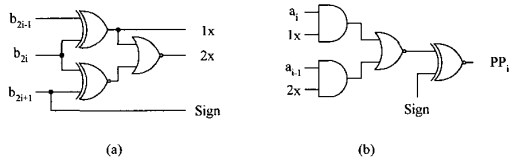

Fig. 2. Encoder (a) and selector (b) blocks.

The addition of the partial products can be done as in the 2's complement multipliers by either a CSA or tree structures. Since 1's complement arithmetic is used here, every carry-out bit must be fed back as a carry-in to the adder array. In the case where the 1's complement multiplier is used as a block of a floating-point multiplier, a final 1's complement adder with single representation of zero [7, 8, 17] must be used. We have to note that very efficient 1's complement adder designs have been proposed in [7, 8], therefore we will not consider this further in this work.

Fig. 3 presents a 8x8 modified Booth 1's complement multiplier with a CSA. The comparison of the proposed structure against a 2's complement multiplier structure reveals that both require

approximately the same area and should offer similar execution times. We will present indicative results confirming this in Section 5. However, as it is obvious from this figure the proposed multiplier has a completely regular structure; therefore pipelined implementations can be derived easily. For example in Fig. 3 if the partial product generation and product reduction has a delay close to that of the final adder a simple 30-bit register before the 1's complement adder can be utilized to spread the overall latency in two consecutive shorter cycles.

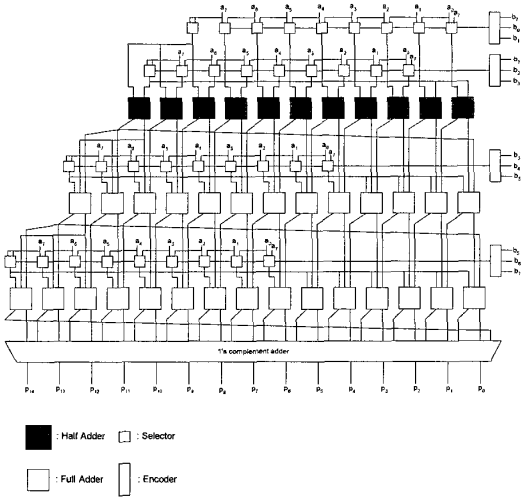

Fig. 3. 1's complement 8 x 8 multiplier.

4. Modulo 2^n-1 Modified Booth multipliers

For high performance modulo multiplication dedicated multipliers are required which can be implemented as combinational or pipelined circuits.

For introducing our modulo 2^n-1 modified Booth multipliers we will firstly show that the number of partial products can be reduced. To this end we utilize the following Lemma :

Lemma 2. Let $X = x_n x_{n-1} x_{n-2} \ldots x_1 x_0$ (x_n is the sign bit) a signed binary number in 1's complement representation. Then :

$$\left|X2^i\right|_{2^n-1} = x_{|n-1+i(n-1)|_n} x_{|n-2+i(n-1)|_n} \ldots x_{|1+i(n-1)|_n} x_{|0+i(n-1)|_n}$$

Since the modified Booth multiplication algorithm is based on signed binary number representations, for applying it to unsigned binary numbers, as for example to the two modulo 2^n-1 numbers A, B, these should first be transformed to positive binary numbers by adding a leading zero, that is $A = 0a_{n-1}a_{n-2} \ldots a_1a_0$, $B = 0b_{n-1}b_{n-2} \ldots b_1b_0$ ($a_n, b_n = 0$).

As shown in the previous section, the multiplier B can be expressed as : $B = \sum_i [2^{2i}(b_{2i-1}+b_{2i}-2b_{2i+1})]$, where $b_{n+1}, b_{-1}, b_n = 0$. The value of the product AB in modulo 2^n-1 can be expressed as :

$$|AB|_{2^n-1} = \left|A\sum_i 2^{2i}(b_{2i-1}+b_{2i}-2b_{2i+1})\right|_{2^n-1} =$$

$$= \left|\sum_i \left|A2^{2i}(b_{2i-1}+b_{2i}-2b_{2i+1})\right|_{2^n-1}\right|_{2^n-1} =$$

$$\left|\sum_i A\left|2^{2i}(b_{2i-1}+b_{2i}-2b_{2i+1})\right|_{2^n-1}\right|_{2^n-1} = \left|\sum_i PP_i\right|_{2^n-1} \text{ where}$$

$PP_i = A\left|2^{2i}(b_{2i-1}+b_{2i}-2b_{2i+1})\right|_{2^n-1}$ are the partial products of the multiplication, which are derived according to Lemma 2 and summarized in Table II.

Table II. Recoded partial products

$b_{2i+1}b_{2i}b_{2i-1}$		Partial Product (PP_i)
0 0 0	0	000 ... 000 or 111...111
0 0 1	$+A2^{2i}$	$a_{\|n-1+2i(n-1)\|_n}\, a_{\|n-2+2i(n-1)\|_n} \ldots a_{\|0+2i(n-1)\|_n}$
0 1 0	$+A2^{2i}$	$a_{\|n-1+2i(n-1)\|_n}\, a_{\|n-2+2i(n-1)\|_n} \ldots a_{\|0+2i(n-1)\|_n}$
0 1 1	$+A2^{2i+1}$	$a_{\|n-1+(2i+1)(n-1)\|_n}\, a_{\|n-2+(2i+1)(n-1)\|_n} \ldots a_{\|0+(2i+1)(n-1)\|_n}$
1 0 0	$-A2^{2i+1}$	$a'_{\|n-1+(2i+1)(n-1)\|_n}\, a'_{\|n-2+(2i+1)(n-1)\|_n} \ldots a'_{\|0+(2i+1)(n-1)\|_n}$
1 0 1	$-A2^{2i}$	$a'_{\|n-1+2i(n-1)\|_n}\, a'_{\|n-2+2i(n-1)\|_n} \ldots a'_{\|0+2i(n-1)\|_n}$
1 1 0	$-A2^{2i}$	$a'_{\|n-1+2i(n-1)\|_n}\, a'_{\|n-2+2i(n-1)\|_n} \ldots a'_{\|0+2i(n-1)\|_n}$
1 1 1	0	111...111 or 000 ... 000

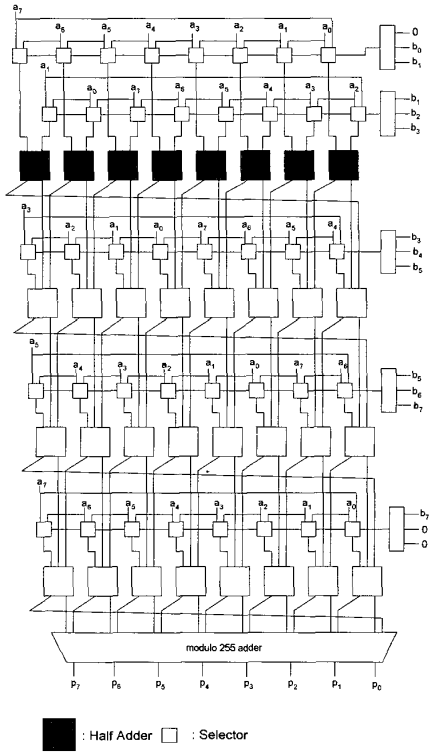

Fig. 4. Modified Booth modulo 255 multiplier.

The above analysis indicates that a modified Booth modulo 2^n-1 multiplier can be designed following also in this case the block diagram of Fig. 1. As a sample we present in Fig. 4 the modulo 255 multiplier.

5. Results & Discussion

In this paper we have presented a modified Booth 1's complement multiplier architecture. We also extended our developed theory for the design of modulo 2^n-1 Booth multipliers.

The structure of the proposed 1's complement modified Booth multipliers resembles that of the 2's complement ones but it is also completely regular. We expect and we will verify by actual implementations that corresponding 1's and 2's complement multipliers will offer the same performance and similar implementation area. The proposed 1's complement multipliers however can be pipelined in a straightforward manner, therefore they may be more applicable in cases of long operands in which spreading the multiplication latency among several cycles is imperative, as for example IEEE floating point mantissas multiplication.

For realistic measures the conventional (integer) and the proposed 1's complement multipliers were described in HDL for operand lengths of 4, 8, 16 and 32 bits. For all the designs a 2-bit recoding scheme and a CSA reduction was adopted. All designs were mapped to the AMS CUB implementation technology (0.6μm, 2-metal layer, 5.0V). During synthesis the netlists were optimized for speed and as a secondary target the tool was instructed to try to recover as much area as possible. Results that completely verify our expectations are given in Table III. All results associated with circuit delay were gathered assuming worst case process parameters and are measured in ns. All area results are in mils^2.

Table III. Area and Delay results of 2's and the proposed 1's complement multipliers

	2's complement		1's complement	
Operands' Length (Bits)	**Area**	**Delay**	**Area**	**Delay**
4	353,2	5,33	370,2	5,40
8	1130,7	10,10	1180,7	10,03
16	3170,5	17,09	3470,5	16,95
32	12254,4	30,64	12642,3	30,42

Table IV. Area and Delay results of the proposed modulo 2^n-1 multipliers

Operands' Length (Bits)	Area	Delay
4	345,5	5,57
8	1046,3	10,47
16	2988,9	17,59
32	11993,1	31,07

The structure of the proposed modulo 2^n-1 multipliers also resembles that of the modulo 2^n multipliers. Therefore we expect that the modulo 2^n-1 multipliers will offer similar performance and implementation area, making them highly applicable in a RNS system. Table IV presents the results that we have obtained for modulo 2^n-1 multipliers with inputs of 4, 8, 16 or 32 bits, following the procedure described earlier.

References

[1] K. Hwang, *Computer Arithmetic : Principles, Architecture and Design*, John Wiley and Sons Publishers, 1979.

[2] A. D. Booth, "A Signed Binary Multiplication Technique", J. Mech. and Applied Mathematics, No. 4, June 1951, pp. 236 – 240.

[3] L. P. Rubinfield, "A Proof of the Modified Booth Algorithm for Multiplication", IEEE Trans. on Computers, vol. C-24, pp. 1014-1015, October 1975.

[4] C. S. Wallace, "A suggestion for a Fast Multiplier", IEEE Trans. on Computers, vol. EC-13, pp. 14-17, February 1964.

[5] P. R. Capello and K. Steiglitz, "A VLSI Layout for a Pipelined Dadda Multiplier", ACM Trans. on Computer Systems 1(2), pp. 157 – 174, May 1983.

[6] A. Wu, et. al., "Modified Booth pipelined multiplication", Electronics Letters, Vol. 34, No. 12, pp. 1179 – 1180, June 1998.

[7] C. Efstathiou et. al., "Area–Time Efficient Modulo 2^n-1 Adder Design", IEEE Trans. on Circuits and Systems–II, Vol. 41 (7), pp. 463–467, July 1994.

[8] L. Kalamboukas, et. al., "High-Speed Regular-Layout Modulo 2^n-1 Adders", IEEE Trans. on Computers, Vol. 49, (4), pp. 673-680, July 2000.

[9] Institute of Electrical and Electronics Engineers. *IEEE Standard for Binary Floating-Point Arithmetic*. IEEE Standard 754-1985. New York: IEEE, 1985.

[10] M. A. Sonderstrand et. al., *Residue Number System Arithmetic : Modern Applications in Digital Signal Processing*, IEEE Press, New York, 1986.

[11] K. M. Elleithy & M. A. Bayoumi, "Fast and Flexible Architectures for RNS arithmetic decoding", IEEE Trans. on Circuits and Systems–II, vol. CAS-39, pp. 226 – 235, April 1992.

[12] M. A. Bayoumi et. al., "A Look-Up Table VLSI Design Methodology for RNS Structures used in DSP Applications", IEEE Trans. on Circuits and Systems, vol. CAS-34, pp. 604-616, June 1987.

[13] R. Zimmermann et. al., "A 177 Mb/s VLSI Implementation of the International Data Encryption Algorithm", IEEE Journal of Solid-State Circuits, vol. 29 (3), pp. 303 – 307, March 1994.

[14] A. Curiger, *VLSI Architectures for Computations in Finite Rings and Fields*, PhD. Thesis, Swiss Federal Institute of Technology (ETH), Zurich 1993.

[15] A. Skavantzos and P. B. Rao, "New multipliers modulo 2^n-1", IEEE Trans. on Computers, Vol. 41, No. 8, pp. 957-961, August 1992.

[16] Z. Wang, G. A. Jullien, W. C. Miller, "An algorithm for multiplication modulo 2^N-1", Proc. of the 39th Midwest Sym. on Circuits and Systems, pp. 1301-4, vol. 3, 1997

[17] R. Zimmermann, "Efficient VLSI Implementation of Modulo (2^n±1) Addition and Multiplication", in Proc. of 14th IEEE Symp. on Comp. Arithmetic, Adelaide, Australia, pp. 158–167, April 1999.

[18] C. Efstathiou and H. T. Vergos, " Modified Booth 1's Complement and Modulo 2^n-1 Multipliers", Computer Technology Institute Technical Report No. 2000/09/03.