A Universal Architecture for Designing Efficient Modulo $2^n + 1$ Multipliers

Leonel Sousa, *Senior Member, IEEE,* and Ricardo Chaves

***Abstract*—This paper proposes a simple and universal architecture for designing efficient modified Booth multipliers modulo $(2^n + 1)$. The proposed architecture is comprehensive, providing modulo $(2^n + 1)$ multipliers with similar performance and cost both for the ordinary and for the diminished-1 number representations. The performance and the efficiency of the proposed multipliers are evaluated and compared with the earlier fastest modulo $(2^n + 1)$ multipliers, based on a simple gate-count and gate-delay model and on experimental results obtained from CMOS implementations. These results show that the proposed approach leads on average to approximately 10% faster multipliers than the fastest known structures for the diminished-1 representation based on the modified Booth recoding. Moreover, they also show that the proposed architecture is the only one taking advantage of this recoding to obtain faster multipliers with a significant reduction in hardware. With the used figures of merit, the proposed diminished-1 multipliers are on average 10% and 25% more efficient than the known most efficient modulo $(2^n + 1)$ multipliers for Booth recoded and nonrecoded multipliers, respectively.**

***Index Terms*—Arithmetic units, digital signal processing, multipliers, residue number systems (RNS).**

I. Introduction

MODULO $(2^n + 1)$ multiplication has been widely used in the Fermat number transform and in residue number system (RNS) implementations [1], which are frequently applied, for example, in signal processing [2]–[4]. Although simple ROM-based structures can be used for modular multiplication, they are only efficient for a small n, while implementations based on arithmetic components are more suitable for medium and large values of n.

In the last few years, several algorithms and architectures were proposed for designing modulo $(2^n + 1)$ multipliers [5]–[10]. Some of those multipliers [7], [8], [10] were specifically designed for the diminished-1 number representation proposed by Leibowitz [11] for Fermat number arithmetic.

Curiger *et al.* [5] investigated several methods for modulo $(2^n + 1)$ multiplication concluding that, for n large and for full-custom circuit design, the method based on (2^n) additions and a bit-pair recoding scheme with output correction is better than the method that applies a $(n + 1) \times (n + 1)$-bit binary multiplier and a final modulo $(2^n + 1)$ subtractor. However, an $(n + \lfloor \log_2 n/r \rfloor)$-bit adder is required for residue reduction of the carry save adder (CSA) array output (for r-bit recoding). Later, the method based on the binary multiplier was modified in order to separately process the most significant bit (MSB) of the operands [3], [6].

During the last few years, several architectures have been proposed to design fast modulo $(2^n + 1)$ multipliers for large n [7]–[12]. Four main stages can be identified in all those architectures: 1) partial products and correction terms (CTs) are generated; 2) partial products are added by using CSA structures, for example, adder trees (Wallace trees) [13]; 3) the CTs are applied, namely by adding them to the previous sum, and finally in stage 4) carry and sum bit-vectors are combined by using a fast carry propagate modulo $(2^n + 1)$ adder.

The architecture for modulo $(2^n + 1)$ multiplication proposed in [7], based on a Wallace tree, is specific for the diminished-1 number representation. It requires the precomputation of a data-dependent CT before applying the Wallace tree, which introduces an additional delay corresponding to some adders to implement an $(n - 1)$-bit counter. In [12], it is shown that part of this delay can be hidden by taking away this circuit from the critical path.

The architecture in [8] applies bit-pair recoding to reduce the number of partial products [13], but it also only accepts numbers in diminished-1 representation. The number of partial products was reduced to approximately $n/2$, but two additional modulo $(2^n + 1)$ CSAs for modulo reduction are required. An improved architecture of this type has been proposed in [10], also for diminished-1 representation. For the first time, a modulo $(2^n + 1)$ architecture that applies bit-pair recoding and Wallace tree adders has been proposed in [9]. However, the main focus of this work is on modulo multiplication for the international data encryption algorithm (IDEA) block cypher. The architecture can be used for the modulo $(2^n + 1)$ multiplication of operands in the ordinary representation, but requires a dedicated circuit which introduces an additional delay to accommodate the 2^n input value. Further adders have to be used to adapt this architecture to support input operands in the diminished-1 representation, resulting in a circuit area and delay increase.

In this paper, a new and efficient architecture is proposed for designing efficient modulo $(2^n + 1)$ multipliers. This architecture is also based on bit-pair recoding and takes full advantage of partial product entry tables with predefined values to distribute the correction process through the various addition phases, in stages 1) and 2). Moreover, stage 3) referred above corresponds at most to a single addition of a CT derived by a set of simple logic equations. We also prove that this method is valid for both

Manuscript received December 22, 2003; revised June 25, 2004. This work was supported in part by the Portuguese Foundation for Science and Technology (FCT) through Program FEDER. This paper was recommended by Associate Editor S.-G. Chen.

The authors are with the Electrical and Computer Engineering Department, Instituto Superior Técnico (IST) and Instituto de Engenharia de Sistemas e Computadores (INESC-ID), 1000-029 Lisbon, Portugal (e-mail: Leonel.Sousa@inesc-id.pt; Ricardo.Chaves@inesc-id.pt).

Digital Object Identifier 10.1109/TCSI.2005.849143

normal and diminished-1 representations, leading to a universal architecture suitable for designing efficient very large-scale integration (VLSI) modulo $(2^n + 1)$ multipliers.

Experimental results show that the proposed multiplier structure is the fastest among all known modulo (2^n+1) multipliers and the first one to take full advantage of Booth recoding to speedup modulo $(2^n + 1)$ multiplication, as well as reducing the amount of required hardware and power consumption.

This paper is organized as follows. In Section II, modulo $(2^n + 1)$ multiplication with bit-pair recoding is briefly presented. An algorithm for modulo (2^n+1) multiplication is proposed in Section III and an architecture for its implementation is presented in Section IV. Based on experimental results presented in Section V, the proposed multipliers are evaluated and compared with other well known fast modulo $(2^n + 1)$ multipliers. Section VI concludes the paper.

II. Modulo $(2^n + 1)$ Multiplication With Modified Booth Recoding

The binary representation of integers modulo $(2^n + 1)$ requires $(n + 1)$-bits. A single bit can be used to identify one of the possible $2^n + 1$ values, while the other n-bits are necessary to represent the remaining 2^n different values. In the ordinary binary representation, the MSB only takes the value "1" for the number 2^n. On the other hand, in the diminished-1 representation, proposed to simplify modulo 2^n+1 basic arithmetic operations [11], the zero is uniquely identified by the MSB ("1"). Another representation proposed for data encryption in [14] uses n-bit numbers in ordinary representation, except for the value 2^n that is represented by the number 0 (the value 0 is not used).

The reduction modulo $(2^n + 1)$ of a number A with at most $2n$ bits can be computed with a single subtraction

$$\langle A\rangle_{2^n+1} = \langle A \bmod 2^n - A \operatorname{div} 2^n\rangle_{2^n+1}. \tag{1}$$

For ordinary representation, the modulo (2^n+1) product (P) of a multiplicand Y by a multiplier X can be formulated as

$$P = \langle\langle y \times x\rangle_{2^n+1} - y_n \times x - x_n \times y + x_n \wedge y_n\rangle_{2^n+1} \tag{2}$$

where z_n represents the nth bit and z the remaining n least significant bits of the number Z. By considering P, X, Y as diminished-1 representations

$$\begin{aligned} P+1 &= \langle (X+1)\times(Y+1)\rangle_{2^n+1} \\ &\Rightarrow P = \langle \langle y\times x\rangle_{2^n+1} + \bar{y}_n \times x + \bar{x}_n \times y \\ &\quad + (x_n \vee y_n)\times 2^n\rangle_{2^n+1}. \end{aligned} \tag{3}$$

Modified booth recoding can be applied to speed up the computation of the partial product terms in (2) and (3) ($x_{n+1} = x_n = x_{-1} = 0$)

$$\langle y\times x\rangle_{2^n+1} = \left\langle \sum_{i=0}^{\lfloor \frac{n}{2}\rfloor} (x_{2i-1} + x_{2i} - 2x_{2i+1}) 2^{2i} \times y \right\rangle_{2^n+1}. \tag{4}$$

The two types of partial products found in (4) can be generated modulo $(2^n + 1)$ by using (5) for $2^k \times y$

$$\begin{aligned} &\langle 2^k \times y\rangle_{2^n+1} \\ &= \left\langle \sum_{i=0}^{n-k-1} y_i 2^{i+k} - \sum_{i=n-k}^{n-1} y_i 2^{k-n+i} + 2^n + 1 \right\rangle_{2^n+1} \\ &= \left\langle \sum_{i=0}^{n-k-1} y_i 2^{i+k} + \sum_{i=n-k}^{n-1} \bar{y}_i 2^{k-n+i} \right. \\ &\quad \left. + \sum_{i=0}^{n-k-1} 2^{k+i} + 2 \right\rangle_{2^n+1} \\ &= \left\langle \sum_{i=0}^{n-k-1} y_i 2^{i+k} + \sum_{i=n-k}^{n-1} \bar{y}_i 2^{k-n+i} + 2^n - 2^k + 2 \right\rangle_{2^n+1} \\ &= \left\langle y_{\langle n-k-1:0\rangle} \# \bar{y}_{\langle n-1:n-k\rangle} - 2^k + 1 \right\rangle_{2^n+1} \end{aligned} \tag{5}$$

and, derived in a similar way, by using (6) for $-2^k \times y$

$$\langle -2^k \times y\rangle_{2^n+1} = \left\langle \bar{y}_{\langle n-k-1:0\rangle} \# y_{\langle n-1:n-k\rangle} + 2^k + 1 \right\rangle_{2^n+1} \tag{6}$$

where $y_{\langle w:v\rangle}$ represents bits of y originally located in positions from v (less significant) to w (more significant) and the symbol # is used to concatenate bits.

This section is focused on the computation of the common term $\langle x \times y\rangle_{2^n+1}$ in (2) and (3), by adding modulo (2^n+1) the $\lfloor (n/2)\rfloor + 1$ partial products in (4). From now on, it is assumed that the simplest modulo $(2^n + 1)$ adders are used for that purpose, the ones for diminished-1 representation, which naturally add the constant 1 ($A + B + 1 = A + B + \bar{c}_{\text{out}}$) [7].

The only term dependent on y of the partial products $(p_{k=2i})$ is generated by a cyclic left-shift of k bits of y and by complementing the k least significant bits (5), or the $n - k$ most significant bits (6). The multiplication algorithm proposed in this paper manipulates both the p_k terms and the terms that only depend on k in (5) and (6), designated by *CTs* ($\text{ct}_k : 2\times 0 \le k \le 2\times\lfloor (n/2)\rfloor$), in order to obtain a simple expression for the modulo $(2^n + 1)$ addition of these terms. These terms are presented in Table I, where $p_k(000\,111)$ is obtained by considering $y = 0$ [9]. The final CT can be formulated as

$$\text{CT} = \left\langle \sum_{k=2\times(i=0)}^{2\times(i=\lfloor \frac{n}{2}\rfloor)} \text{ct}_k \right\rangle_{2^n+1} \tag{7}$$

$$\begin{aligned} \text{ct}_k = &-2^k \times (x_{k-1}\bar{x}_k\bar{x}_{k+1} \vee \bar{x}_{k-1}x_k\bar{x}_{k+1} \vee \bar{x}_{k-1}\bar{x}_k\bar{x}_{k+1}) \\ &- 2^{k+1} \times (x_{k-1}x_k\bar{x}_{k+1}) + (x_{k-1}\bar{x}_k x_{k+1} \\ &\vee \bar{x}_{k-1}x_k x_{k+1} \vee x_{k-1}x_k x_{k+1}) \times 2^k \\ &+ (\bar{x}_{k-1}\bar{x}_k x_{k+1}) \times (2^{k+1}) + (1). \end{aligned} \tag{8}$$

By performing the diminished-1 addition of the CTs, considering a common term $(-2^k - 2^{k+1})$ and by applying the distributive property, CT can be formulated as

$$\begin{aligned} \text{CT} = \Bigg\langle &\sum_{k=2\times(i=1)}^{2\times(i=\lfloor \frac{n}{2}\rfloor)} [(x_{k-1}\bar{x}_k\bar{x}_{k+1} \\ &\vee \bar{x}_{k-1}x_k\bar{x}_{k+1} \vee \bar{x}_{k-1}\bar{x}_k\bar{x}_{k+1}) \times 2^{k+1} \\ &+ (x_{k-1}x_k\bar{x}_{k+1} \vee \bar{x}_{k-1}\bar{x}_k x_{k+1}) \times 2^k \\ &+ x_{k+1} \times 2^{k+2} - (2^k + 2^{k+1})] + \text{ct}_{2\times 0} \Bigg\rangle_{2^n+1}. \end{aligned} \tag{9}$$

TABLE I
GENERAL PRODUCT TERM (p_{2i}) AND CORRECTION TERM (ct_{2i})

x_{2i+1}	x_{2i}	x_{2i-1}	$p_{2i}(x_{2i+1}x_{2i}x_{2i-1})$	ct_{2i}
0	0	0	$'0\cdots01\cdots1' \equiv 2^{2i}-1$	-2^{2i}
0	0	1	$y_{n-2i-1:0}\#\bar{y}_{n-1:n-2i}$	-2^{2i}
0	1	0	$y_{n-2i-1:0}\#\bar{y}_{n-1:n-2i}$	-2^{2i}
0	1	1	$y_{n-2i-2:0}\#\bar{y}_{n-1:n-2i-1}$	-2^{2i+1}
1	0	0	$\bar{y}_{n-2i-2:0}\#y_{n-1:n-2i-1}$	2^{2i+1}
1	0	1	$\bar{y}_{n-2i-1:0}\#y_{n-1:n-2i}$	2^{2i}
1	1	0	$\bar{y}_{n-2i-1:0}\#y_{n-1:n-2i}$	2^{2i}
1	1	1	$'1\cdots10\cdots0' \equiv 2^n-2^{2i}$	2^{2i}

It is easy to prove the correctness of the following equations, where $n_0 = 1$ for n odd and $n_0 = 0$ otherwise

$$\left\langle -\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor}(2^{2i}+2^{2i+1}) \right\rangle_{2^n+1} = \left\langle -\sum_{i=2}^{n+\bar{n}_0} 2^i \right\rangle_{2^n+1} = 6+2\bar{n}_0$$

$$\sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} x_{2i+1}\times 2^{2i+2} = \sum_{i=2}^{\lfloor \frac{n}{2} \rfloor+1} x_{2i-1}\times 2^{2i} = \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor}(x_{2i-1}\times 2^{2i}) - x_1\times 2^2. \tag{10}$$

Therefore, (9) is equivalent to (11) shown at the bottom of the page.

The $\mathrm{ct}_{2\times 0}$ term was individually considered in order to simplify the presentation of the proposed algorithm for modulo (2^n+1) multiplication. As it will be proved, an efficient algorithm and a universal architecture for computing (2) and (3) can be obtained by manipulating the expressions for p_k and CT (11).

III. ALGORITHM FOR MODULO (2^n+1) MULTIPLICATION

The algorithm for modulo (2^n+1) multiplication uses both (2) and (3). So far, the expressions for p_k and CT to compute the common term $\langle x\times y\rangle_{2^n+1}$ have been obtained. All the other terms in (2) and (3) have to be computed and added to this common term.

A. Modulo (2^n+1) Multiplication for Diminished-1 Representation

For diminished-1 multiplication, modulo (2^n+1) addition of y and x has to be performed whenever x_n and y_n are equal to zero (see (3)). The y term can be added to p_0 whenever $x_n = 0$, which leads to the replacement of the four rows in Table I ($i=0$) by the equivalent rows in Table II and the following expression for the ϕ term in (11):

$$\begin{aligned}\phi &= -\bar{x}_1\bar{x}_0 - 2\bar{x}_1x_0 + x_1\bar{x}_0 - x_1x_0 - 4x_1 + 6 + 2\bar{n}_0\\ &= 2\bar{x}_1\bar{x}_0 + \bar{x}_1x_0 - 2x_1x_0 + 3 + 2\bar{n}_0.\end{aligned} \tag{12}$$

TABLE II
PRODUCT ($p_{2\times 0}$) AND CORRECTION ($\mathrm{ct}_{2\times 0}$) TERMS FOR DIMINISHED-1 REPRESENTATION

x_n	x_1	x_0	$p_{2\times 0}(x_nx_1x_0)$	$ct_{2\times 0}$
0	0	0	$y_{n-1:0}$	-1
0	0	1	$y_{n-2:0}\#\bar{y}_{n-1}$	-2
0	1	0	$\bar{y}_{n-1:0}$	$+1$
0	1	1	$'y_n\cdots y_n' \equiv 2^n-1\vert_{y_n=1} \vee 0\vert_{y_n=0}$	-1
1	0	0	$'0\cdots0'$	-1

TABLE III
PRODUCT (p_{2i}) AND CORRECTION (c_{2i}) TERMS ($i \geq 1$) FOR DIMINISHED-1 REPRESENTATION

x_{2i+1}	x_{2i}	x_{2i-1}	$p_{2i}(x_{2i+1}x_{2i}x_{2i-1})$	ct_{2i}
0	0	0	$'0\cdots01\cdots1' \equiv 2^{2i}-1$	-2^{2i}
0	0	1	$y_{n-2i-1:0}\#\bar{y}_{n-1:n-2i}$	-2^{2i}
0	1	0	$y_{n-2i-1:0}\#\bar{y}_{n-1:n-2i}$	-2^{2i}
0	1	1	$y_{n-2i-2:0}\#\bar{y}_{n-1:n-2i-1}$	-2^{2i+1}
1	0	0	$\bar{y}_{n-2i-2:0}\#y_{n-1:n-2i-1}$	2^{2i+1}
1	0	1	$\bar{y}_{n-2i-1:0}\#y_{n-1:n-2i}$	2^{2i}
1	1	0	$\bar{y}_{n-2i-1:0}\#y_{n-1:n-2i}$	2^{2i}
1	1	1	$'y_n\cdots y_n1\bar{y}_n\cdots\bar{y}_n' \equiv 2^n-2^{2i}\vert_{y_n=1} \vee 2^{2i+1}-1\vert_{y_n=0}$	2^{2i}

In Table II, the x_n bit takes the place of the x_{-1} bit, which is always zero, and $p_{2\times 0}$ is assigned the value zero when $x_n = 1$. To obtain a simple logic equation for CT, the term $-2\times x_1x_0$ has to be eliminated in (12). For $y_n = 1$, the value $-2\times x_1x_0$ can be directly added modulo (2^n+1) to the entry of p_0 corresponding to $x_1 = x_0 = 1$ by changing its value from '0' to "2^n-1" (see Table II). For $y_n = 0$, it can be taken advantage of the fact that $2\times x_1 + x_0$ (see (3)) must be added to ϕ. Thus, p_0 remains with the value "0" but ϕ' takes the place of ϕ in (11) for both values of y_n

$$\begin{aligned}\phi' &= 2\bar{x}_1\bar{x}_0 + \bar{x}_1x_0 - 2x_1x_0\bar{y}_n + 2x_1\bar{y}_n + x_0\bar{y}_n + 3 + 2\bar{n}_0\\ &= 2^1\times(\bar{y}_n\bar{x}_1 \vee \bar{y}_n\bar{x}_0 \vee \bar{x}_1\bar{x}_0)\\ &\quad + 2^0\times(\bar{y}_nx_1x_0 \vee y_n\bar{x}_1x_0) + 3 + 2\bar{n}_0.\end{aligned} \tag{13}$$

To add the remaining bits of x when $y_n = 0$, the terms $x_{2i}\times 2^{2i}$ and $x_{2i+1}\times 2^{2i+1}$ must be added to $\psi(k)$ in (11) for all i in $1 \leq i \leq \lfloor n/2 \rfloor (k = 2\times i)$

$$\begin{aligned}&\psi(k) + x_k\times 2^k + x_{k+1}\times 2^{k+1}\\ &\quad = (\bar{x}_kx_{k+1} \vee x_{k-1}\bar{x}_k \vee x_k)\times 2^k\\ &\qquad + (\bar{x}_{k-1} \vee \bar{x}_k \vee \bar{x}_{k+1})\times 2^{k+1}\\ &\qquad + x_{k-1}x_kx_{k+1}(2^{k+1}+2^k).\end{aligned} \tag{14}$$

Once again, to avoid undesirable arithmetic operations, the term $(2^{2i}+2^{2i+1})$ in (14) can be added to the corresponding partial product for $y_n = 0$, leading to

$$p_{2i}(111) = 2^n - 2^{2i} + 2^{2i} + 2^{2i+1} = 2^{2i+1} - 1.$$

$$\mathrm{CT} = \left\langle \sum_{k=2\times(i=\lfloor \frac{n}{2} \rfloor)}^{2\times(i=\lfloor \frac{n}{2} \rfloor)} \overbrace{[(\bar{x}_kx_{k+1} \vee x_{k-1}x_{k+1} \vee x_{k-1}\bar{x}_k)\times 2^k + \bar{x}_{k+1}\times 2^{k+1}]}^{\psi(k)} + \overbrace{\mathrm{ct}_{2\times 0} - 4\times x_1 + 6 + 2\times \bar{n}_0}^{\phi} \right\rangle_{2^n+1} \tag{11}$$

TABLE IV
MODIFIED PRODUCT ($p_{2\times0}$) AND CORRECTION ($\mathrm{ct}_{2\times0}$) TERMS FOR DIMINISHED-1 REPRESENTATION

$x_n \vee x_{n-1}$	x_1	x_0	$p_{2\times0}$	$ct_{2\times0}$
$0\vee0$	0	0	$y_{n-1:0}$	-1
$0\vee0$	0	1	$y_{n-2:0}\#\bar{y}_{n-1}$	-2
$0\vee0$	1	0	$\bar{y}_{n-1:0}$	$+1$
$0\vee0$	1	1	$'y_n\cdots y_n'\equiv$ $\equiv 2^n-1\vert_{y_n=1}\vee 0\vert_{y_n=0}$	-1
$1\vee0$	0	0	$'0\cdots0'$	-1
$0\vee1$	0	0	$'0\cdots0'$	-1
$0\vee1$	0	1	$y_{n-1:0}$	-1
$0\vee1$	1	0	$\bar{y}_{n-2:0}\#y_{n-1}$	$+2$
$0\vee1$	1	1	$\bar{y}_{n-1:0}$	$+1$

So $\psi(k)$ in (11) is replaced by $\psi'(k)$ in (15) and $p_{2\times i}$ is generated according to Table III

$$\psi'(k) = (\bar{x}_k x_{k+1} \vee x_{k-1}\bar{x}_k \vee y_n x_{k-1} x_{k+1} \vee \bar{y}_n x_k) \times 2^k + (\bar{x}_{k+1} \vee \bar{y}_n \bar{x}_k \vee \bar{y}_n \bar{x}_{k-1}) \times 2^{k+1}. \quad (15)$$

By isolating and simplifying the terms of $\psi'(k)$, for $k = 2\times(i = \lfloor n/2 \rfloor)$ (11), as well as the terms of ϕ' (13), simple logic equations are obtained to compute CT

$$\begin{aligned}\mathrm{CT}\vert_{\mathrm{n\,odd}} &= \left\langle \left[\sum_{k=2\times(i=1)}^{2\times(i=\frac{n-3}{2})} \psi'(k)\right] \right. \\ &\quad + 2^{n-1}\times(x_{n-2}\bar{x}_{n-1} \vee \bar{y}_n x_{n-1}) \\ &\quad + 2^1 \times (\bar{y}_n \bar{x}_1 \vee \bar{y}_n \bar{x}_0 \vee \bar{x}_1 \bar{x}_0) \\ &\quad \left. + 2^0 \times (\bar{y}_n x_1 x_0 \vee y_n \bar{x}_1 x_0) + 2 \right\rangle_{2^n+1} \end{aligned} \quad (16)$$

$$\begin{aligned}\mathrm{CT}\vert_{\mathrm{n\,even}} &= \left\langle \left[\sum_{k=2\times(i=1)}^{2\times(i=\frac{n-2}{2})} \psi'(k)\right] \right. \\ &\quad + 2^1 \times (\bar{y}_n \bar{x}_{n-1} \vee \bar{x}_1 \bar{x}_0 \vee \bar{y}_n \bar{x}_1 \vee \bar{y}_n \bar{x}_0 \vee \bar{x}_{n-1}\bar{x}_1) \\ &\quad + 2^0 \times (\bar{x}_{n-1}\bar{x}_0 \vee \bar{y}_n \bar{x}_{n-1}\bar{x}_1 \\ &\quad \left. \vee\, \bar{y}_n x_{n-1} x_1 x_0 \vee y_n x_{n-1}\bar{x}_1 x_0 \vee y_n \bar{x}_{n-1} x_1) + 2 \right\rangle_{2^n+1}. \end{aligned} \quad (17)$$

In both (16) and (17), the nth bit of the result takes the value $x_n \vee y_n$.

1) Improving the Algorithm for n Even: Only two of the eight entries in Table III are used for n even and $i = n/2 (x_{n+1} = x_n = 0)$, and from those two entries just the term where $x_{n-1} = 1$ is not constant—it takes the value $\langle 2^n \times y = -y\rangle_{2^n+1}$. Since Table II only has five filled entries, these two tables can be merged together into a single one (Table IV), by adjusting the CT.

Hence, Table IV has been obtained from Table II by adding the term $(-y)$ for $x_{n-1} = 1$. This new table can be considered to have only eight entries, by addressing it with $x'_n = x_n \vee x_{n-1}$. However, the CT has to be adjusted, in order to: 1) accommodate the new $\mathrm{ct}_{2\times0}$ for $x_{n-1} = 1$; 2) subtract $\mathrm{ct}_n(001) + 1$ from the final CT, since this term was already part of $\mathrm{ct}_{2\times0}$; 3) include the remaining original $p_n(000)$ term.

TABLE V
PRODUCT ($p_{2\times0}$) AND CORRECTION ($\mathrm{ct}_{2\times0}$) TERMS FOR ORDINARY REPRESENTATION

x_n	x_1	x_0	$p_{2\times0}(x_n x_1 x_0)$	$ct_{2\times0}$
0	0	0	$'0\cdots0'$	-1
0	0	1	$y_{n-1:0}$	-1
0	1	0	$\bar{y}_{n-2:0}\#y_{n-1}$	$+2$
0	1	1	$\bar{y}_{n-1:0}$	$+1$
1	0	0	$\bar{y}_{n-1:0}$	$+1$

By isolating the terms of $\psi'(k)$, for $k = 2\times(i = (n/2))$ (15), and by adding the terms previously referred in 2) and 3) one obtains

$$\langle x_{n-1}\times 2^n + 2^{n+1} + \bar{x}_{n-1}\times(2^n-1) + x_{n-1}\times(2^n-1)\rangle_{2^n+1} = \langle -3 - 2 + \bar{x}_{n-1}\rangle_{2^n+1}. \quad (18)$$

By adding ϕ' (13) with expression (18), for $x_{n-1} = 0$

$$\phi'\vert_{x_{n-1}=0} = 2^1 \times [\bar{x}_{n-1}(\bar{y}_n \vee \bar{x}_1\bar{x}_0 \vee \bar{y}_n\bar{x}_1 \vee \bar{y}_n\bar{x}_0 \vee \bar{x}_1)] + 2^0 \times [\bar{x}_{n-1}(\bar{x}_0 \vee \bar{y}_n\bar{x}_1 \vee y_n x_1)] + 2 - 2. \quad (19)$$

The new ϕ' corresponding to the $\mathrm{ct}_{2\times0}$ terms for $x_{n-1} = 1$ is derived by applying the new values in Table IV to ϕ [see (11)] and by adding the term $2\times x_1\bar{y}_n + x_0\bar{y}_n$ [see derivation of (13)]

$$\phi'\vert_{x_{n-1}=1} = 2^1 \times (x_{n-1}\bar{x}_1 \vee x_{n-1}\bar{y}_n x_1) + 2^0 \times (x_{n-1}x_1\bar{x}_0 \vee x_{n-1}\bar{y}_n x_0). \quad (20)$$

By adding expressions (19) and (20), the following equation is obtained to compute CT:

$$\mathrm{CT}\vert_{\mathrm{n\,even}} = \left\langle \left[\sum_{k=2\times(i=1)}^{2\times(i=\frac{n-2}{2})} \psi'(k)\right] + 2^1 \times (\bar{x}_1 \vee \bar{y}_n\bar{x}_{n-1} \vee \bar{y}_n x_1) + 2^0 \times (\bar{x}_{n-1}\bar{x}_0 \vee x_1\bar{x}_0 \vee \bar{y}_n\bar{x}_{n-1}\bar{x}_1 \vee y_n\bar{x}_{n-1}x_1 \vee \bar{y}_n x_{n-1}x_0) \right\rangle_{2^n+1} \quad (21)$$

and the nth bit of the result takes the value $x_n \vee y_n$.

B. Modulo (2^n+1) Multiplication for Ordinary Representation

For ordinary multiplication, modulo (2^n+1) addition of $(-y)$ and $(-x)$ has to be performed whenever the values of x_n and y_n are nonzero [see (2)]. By following the same approach used for the diminished-1 representation, the $(-y)$ term can be added to $p_{2\times0}$ whenever $x_n = 1$. The x_n bit takes the place of the x_{-1} bit, which is always zero, leading to the last row in Table V and to the following expression for ϕ in (11):

$$\begin{aligned}\phi &= -\bar{x}_n\bar{x}_1 + 2x_1\bar{x}_0 + x_1x_0 + x_n\bar{x}_1\bar{x}_0 - 4x_1 + 6 + 2\bar{n}_0 \\ &= 4x_n + 2\bar{x}_n\bar{x}_1 + x_1\bar{x}_0 + 3 + 2\bar{n}_0. \end{aligned} \quad (22)$$

Unfortunately, by adding $(-x)$ to CT when $y_n = 1$ does not lead to any simple logic equation. This term can be added by filling the unused half part of the last product terms of Table I for $p_{2\times\lfloor n/2\rfloor}$ with the value $\langle -x\rangle_{2^n+1} = \langle \bar{x}_{n-1:0} + 2\rangle_{2^n+1}$ and by replacing the $x_{2\times\lfloor n/2\rfloor+1}$ input bit by y_n. However, this

TABLE VI
PRODUCT TERM ($p_{2\times\lfloor n/2\rfloor}$) FOR ORDINARY REPRESENTATION

y_n	$x_{2\times\lfloor n/2\rfloor}$	$x_{2\times\lfloor n/2\rfloor-1}$	$p_{2\times\lfloor n/2\rfloor}$	$prev_{2\times\lfloor n/2\rfloor}$
0	0	0	$'1\cdots1'\equiv$ $\equiv 2^{2\times\lfloor n/2\rfloor}-1$	$--$
0	0	1	$y_{n-2\times\lfloor n/2\rfloor-1:0}\#$ $\#\bar{y}_{n-1:n-2\times\lfloor n/2\rfloor}$	$--$
0	1	0	$y_{n-2\times\lfloor n/2\rfloor-1:0}\#$ $\bar{y}_{n-1:n-2\times\lfloor n/2\rfloor}$	$--$
0	1	1	$\bar{y}_{n-1:0}$	$--$
1	0	0	$\bar{x}_{n-1:0}$	$[(2^{n-1}-1)n_0$
1	0	1	$\bar{x}_{n-1:0}$	$\vee$
1	1	0	$\bar{x}_{n-1:0}$	$(2^n-1)\bar{n}_0]+2$
1	1	1	$\bar{x}_{n-1:0}$	$[2^n-1]+2$

is a disadvantage regarding the diminished-1 multiplication as it prevents further improvement of the algorithm [Section III-A1)].

The product term $p_{2\times\lfloor n/2\rfloor}$ with the already mentioned modifications is presented in Table VI. Before introducing y_n in Table VI, the previous $p_{2\times\lfloor n/2\rfloor}$ term $(\mathrm{prev}_{2\times\lfloor n/2\rfloor})$ must be added to the CT, together with the constant term 2 from $\langle\bar{x}_{n-1:0}+2\rangle_{2^n+1}$. Adding these terms and x_ny_n [see (2)], (11) and (22) lead to

$$\begin{aligned}\mathrm{CT}=\Bigg\langle&\left[\sum_{k=2\times(i=1)}^{2\times(i=\lfloor\frac{n}{2}\rfloor)}\psi(k)\right]\\&+(2^{n-1}+1)\times[y_nn_0(\bar{x}_{n-2}\vee\bar{x}_{n-1})]\\&+4x_n+2\bar{x}_n\bar{x}_1+x_1\bar{x}_0+x_ny_n+3+2\bar{n}_0\Bigg\rangle_{2^n+1}\end{aligned}\tag{23}$$

Although the terms $(4\times x_n)$ and $\psi(2)$ (see (11)) apparently overlap, fortunately they are never simultaneously different from zero $(x_n=1\Rightarrow x=0)$.

By isolating the terms of $\psi(k)$ for $k=2\times(i=\lfloor n/2\rfloor)$ in (23) and by operating logic simplifications, we obtain (24) and (25) to compute CT

$$\begin{aligned}\mathrm{CT}|_{\mathrm{n\,odd}}=\Bigg\langle&\left[\sum_{k=2\times(i=1)}^{2\times(i=\frac{n-3}{2})}\psi(k)\right]\\&+2^{n-1}\times(x_{n-2}\bar{x}_{n-1}\bar{y}_n\vee y_n\bar{x}_{n-2})+2^2\times x_n\\&+2^1\times(\bar{x}_n\bar{x}_1\vee x_ny_n\vee y_n\bar{x}_{n-2}x_1\bar{x}_0)\\&+2^0\times(x_{n-2}x_1\bar{x}_0\vee\bar{y}_nx_1\bar{x}_0\vee y_n\bar{x}_n\bar{x}_{n-2}\bar{x}_1\\&\vee y_n\bar{x}_{n-2}x_0)+2\Bigg\rangle_{2^n+1}\end{aligned}\tag{24}$$

$$\begin{aligned}\mathrm{CT}|_{\mathrm{n\,even}}=\Bigg\langle&\left[\sum_{k=2\times(i=1)}^{2\times(i=\frac{n-2}{2})}\psi(k)\right]+2^2\times x_n\\&+2^1\times(\bar{x}_n\bar{x}_1\vee y_nx_n\vee\bar{x}_{n-1}x_1\bar{x}_0)\\&+2^0\times(x_{n-1}x_1\bar{x}_0\vee\bar{x}_n\bar{x}_{n-1}\bar{x}_1\\&\vee\bar{y}_n\bar{x}_{n-1}\bar{x}_1\vee\bar{x}_{n-1}x_0)+2\Bigg\rangle_{2^n+1}.\end{aligned}\tag{25}$$

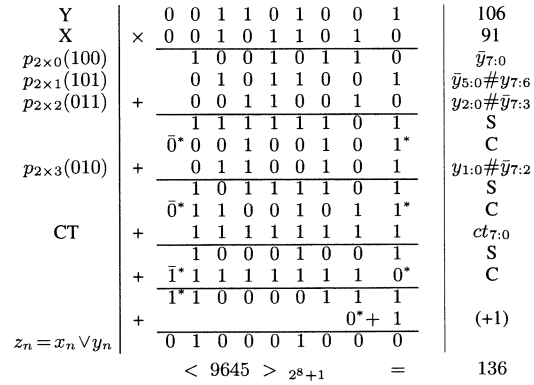

Fig. 1. Example of a diminished-1 modulo $(\mathbf{2^8+1})$ multiplication using the proposed algorithm.

C. Algorithm Formulation

According to the previous sections, the proposed modulo (2^n+1) multiplication algorithm can be formulated through (26) by using Tables VII and VIII for diminished-1 and the ordinary representations, respectively. The $*$ symbol in the definition of ct_k and CT in Table VII means that p_n can assume the value of CT and no further CT has to be applied, as it is represented in (26). The 1's in (26) are automatically added since diminished-1 $(\mathbf{2^n+1})$ units are used. However, due to the fact that the number of additions is one less than the number of terms in the sum, the constant terms "1" and "2" have to be added for the diminished-1 with n even and for all other cases, respectively

$$P=\left\langle\left(\sum_{i=0}^{\lfloor\frac{n}{2}\rfloor-D_E}(p_{2i}+1)+\mathrm{CT}_{xy}+1\right)+1-D_E\right\rangle_{2^n+1}$$
$$D_E=\begin{cases}1, & \text{for diminished-1 and } n \text{ even}\\ 0, & \text{otherwise.}\end{cases}\tag{26}$$

D. An Example

Fig. 1 shows an example of modulo (2^8+1) diminished-1 multiplication using the proposed algorithm. The multiplicand and the multiplier are represented in diminished-1 notation, assuming the values 106 and 91, respectively. The generation of the partial products and of the CTs is done according to Table VII. The carry save addition of these terms leads to the final pair of sum and carry bit vectors (S, C) which are combined in a modulo (2^8+1) carry propagate addition. The result is $(\langle106\times91\rangle_{2^8+1}=137-1)$.

Fig. 2 shows a similar example using the same bit values but assuming the ordinary representation, which means that the operand values will be 105 and 90. The generation of the partial products and of the CTs is done according to Table VIII and the result takes the value $\langle105\times90\rangle_{2^8+1}=198$.

IV. ARCHITECTURE FOR MODULO (2^n+1) MULTIPLICATION

The proposed architecture for modulo (2^n+1) multiplication is presented in Fig. 3. The sum of the generated partial products is obtained by using a Wallace tree of modulo CSAs (MCSA). The Wallace tree accommodates $\lfloor n/2\rfloor+1$ and $\lfloor n/2\rfloor+2$ operands (including the CT) for the diminished-1 representation

TABLE VII
PRODUCT TERM (p) AND CORRECTION TERM (CT) FOR DIMINISHED-1 REPRESENTATION

k	n even		n odd	
	ct_k^*	p_k table		ct_k
0	$\bar{x}_{n-1}\bar{x}_0 \vee x_1\bar{x}_0 \vee \bar{y}_n\bar{x}_{n-1}\bar{x}_1 \vee y_n\bar{x}_{n-1}x_1 \vee \bar{y}_n x_{n-1}x_0$	IV	II	$\bar{y}_n x_1 x_0 \vee y_n\bar{x}_1 x_0$
1	$\bar{x}_1 \vee \bar{y}_n x_1 \vee \bar{y}_n\bar{x}_{n-1}$	–	–	$\bar{y}_n\bar{x}_1 \vee \bar{y}_n x_1\bar{x}_0 \vee y_n\bar{x}_1\bar{x}_0$
$2i\ (1 \le i < \lfloor \frac{n}{2} \rfloor)$	$\psi_A \equiv \bar{x}_{2i}x_{2i+1} \vee x_{2i-1}\bar{x}_{2i} \vee x_{2i}\bar{y}_n \vee x_{2i-1}x_{2i+1}y_n$	III	III	ψ_A
$2i+1\ (1 \le i < \lfloor \frac{n}{2} \rfloor)$	$\psi_B \equiv \bar{x}_{2i-1}\bar{y}_n \vee \bar{x}_{2i}\bar{y}_n \vee \bar{x}_{2i+1}$	–	–	ψ_B
$2 \times \lfloor \frac{n}{2} \rfloor$	–	CT*	III	$x_{n-2}\bar{x}_{n-1} \vee \bar{y}_n x_{n-1}$
n	$z_n = x_n \vee y_n$			

TABLE VIII
PRODUCT TERM (p) AND CORRECTION TERM (CT) FOR ORDINARY REPRESENTATION

k	p_k	ct_k	
		n even	n odd
0	table V	$x_{n-1}x_1\bar{x}_0 \vee \bar{x}_n\bar{x}_{n-1}\bar{x}_1 \vee$ $\vee \bar{y}_n\bar{x}_{n-1}\bar{x}_1 \vee \bar{x}_{n-1}x_0$	$x_{n-2}x_1\bar{x}_0 \vee \bar{y}_n x_1\bar{x}_0 \vee$ $\vee y_n\bar{x}_n\bar{x}_{n-2}\bar{x}_1 \vee y_n\bar{x}_{n-2}x_0$
1	–	$\bar{x}_n\bar{x}_1 \vee y_n x_n \vee \bar{x}_{n-1}x_1\bar{x}_0$	$\bar{x}_n\bar{x}_1 \vee x_n y_n \vee y_n\bar{x}_{n-2}x_1\bar{x}_0$
2	table I	$x_n \vee \bar{x}_2x_3 \vee x_1x_3 \vee x_1\bar{x}_2$	
$2i+1\ (1 \le i < \lfloor \frac{n}{2} \rfloor)$	–	$\bar{x}_{2i+1}$	
$2i\ (2 \le i < \lfloor \frac{n}{2} \rfloor)$	table I	$\bar{x}_{2i}x_{2i+1} \vee x_{2i-1}x_{2i+1} \vee x_{2i-1}\bar{x}_{2i}$	
$2 \times \lfloor \frac{n}{2} \rfloor$	table VI	–	$x_{n-2}\bar{x}_{n-1}\bar{y}_n \vee y_n\bar{x}_{n-2}$

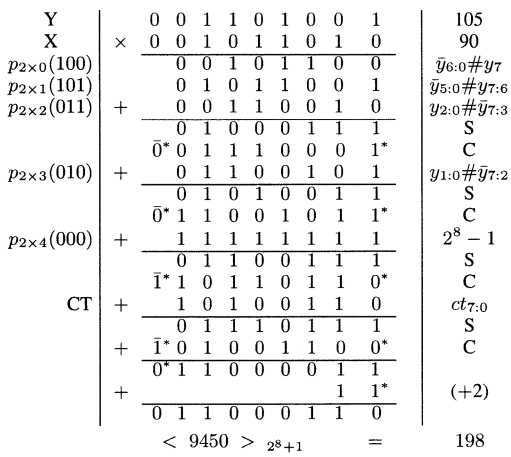

Fig. 2. Example of an ordinary modulo $(\mathbf{2^8 + 1})$ multiplication using the proposed algorithm.

with n even and for all other cases, respectively. An additional MCSA structure, for adding the CT, is represented apart just for simplifying the presentation and the explanation. A modulo carry propagate adder (MCPA) structure is also required to calculate the final result from the sum and carry bit vectors. The nth bit of the product is generated by the MCPA structure for the ordinary representation, while it only depends on x_n and y_n for the diminished-1 representation.

The performance of the proposed multipliers and the comparison with other structures is strongly dependent on the target technology used for the implementation. Therefore, to obtain a fair assessment of these modulo $(2^n + 1)$ multiplier architectures independently of the target technology this evaluation was carried out by first using a neutral and theoretical criteria. Consequently, it was decided to adopt the model proposed by Tyagi, which relates the required circuit area (A) with the propagation time (T) of each logic cell [15]. Each two-input monotonic cell counts as one gate (area and delay), a XOR cell counts as two

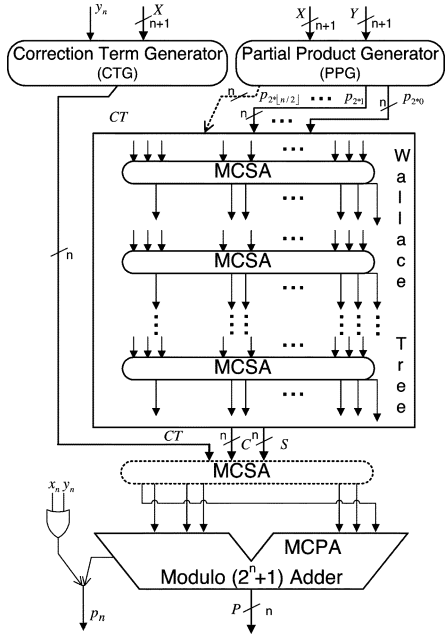

Fig. 3. Block diagram of the proposed architecture for modulo $(2^n + 1)$ multiplication.

gates both to the area and to the delay and an m-bit logic cell, derived from elementary logic cells, counts as $m - 1$ gates to the area and $\lceil \log_2 m \rceil$ gates to the delay.

For the proposed modulo $(2^n + 1)$ multipliers, the main components that contribute to the delay and area are identified in Fig. 3

$$T = T_{\mathrm{PPG}} + T_{\mathrm{CSA}} + T_{\mathrm{COR}} + T_{\mathrm{CPA}}$$
$$A = A_{\mathrm{PPG}} + A_{\mathrm{CSA}} + A_{\mathrm{COR}} + A_{\mathrm{CPA}} \quad (27)$$

where PPG, CSA, COR, and CPA denote the partial-product generator, the CSAs, the correction unit and the final CPA with modulo reduction.

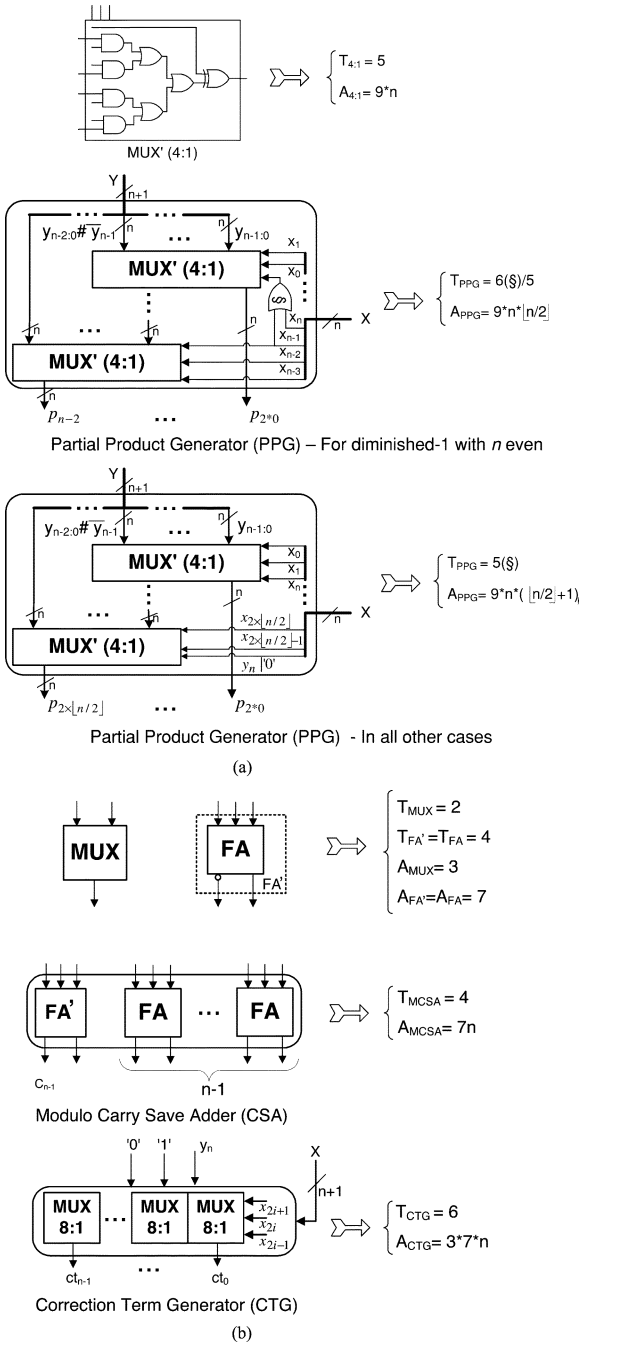

Fig. 4. Components of the architecture proposed for modulo $(2^n + 1)$ multiplication. (a) Partial product generator. (b) Modulo $(2^n + 1)$ CS adders and CT generator.

TABLE IX
NUMBER OF STAGES OF A WALLACE TREE AS A FUNCTION OF NUMBER OF OPERANDS (K)

K	3	4	5 − 6	7 − 9	10−13	14−19	20−28	29−42
$d(K)$	1	2	3	4	5	6	7	8

The final reduction stage is formed by a MCPA structure with a single additional logic gate for computing the nth bit for the diminished-1 representation. A Sklansky parallel-prefix structure with a fast output incrementer [16] can be used to implement a

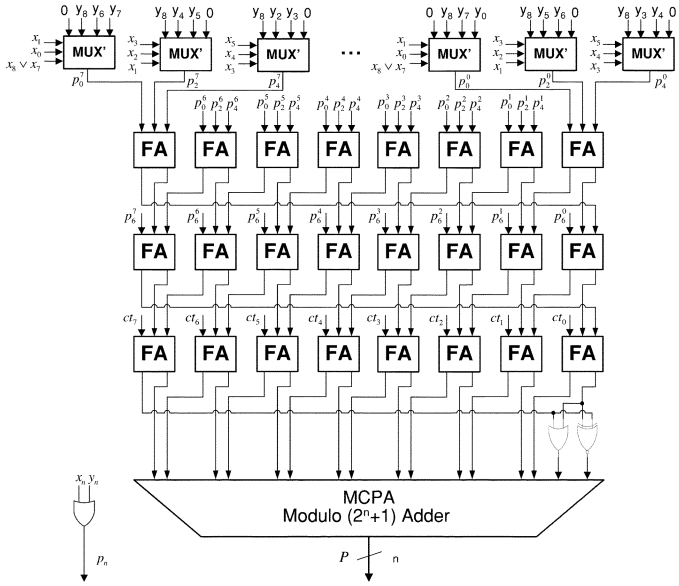

Fig. 5. Architecture for modulo $(\mathbf{2^8 + 1})$ multiplication.

fast n-bit modulo $(2^n + 1)$ CPA for diminished-1 representation, with the delay and the circuit area

$$T_{\mathrm{CPA}} = 2\lceil \log_2 n \rceil + 6$$
$$A_{\mathrm{CPA}} = \frac{3}{2} n \lceil \log_2 n \rceil + 8n. \tag{28}$$

Fig. 4 provides more detailed information about the remaining components of the modulo $(2^n + 1)$ multipliers and their characteristics according to the Tyagi model. The Wallace-tree adder is formed by a set of $K = \lfloor n/2 \rfloor - 1$ modulo CSAs, to compress the column size corresponding to each bit position from K to 2. The number of stages $d(K)$ for adding the K numbers is given in [7] and can also be found in Table IX.

The delay of a 1-bit full-adder is considered to be 4 gates while its area corresponds to seven gates. Therefore, the delay and area of a Wallace-tree that accommodates the n-bit terms should take the values in (29)

$$T_{\mathrm{CSA}} = 4 \times d(K) \quad A_{\mathrm{CSA}} = 7 \times (K - 2) \times n. \tag{29}$$

In general, radix-4 recoding can be implemented by using 8:1 n-bit multiplexers. However, by analysing the Booth Tables I to VI and considering that a controlled inverter is used, this selection only applies to three or four terms. Therefore, a significant reduction in circuit area can be achieved by using a (4:1) n-bit multiplexer plus n XOR gates that will invert the partial product value whenever it is required. The delay and circuit area for such multiplexer (MUX') is given in Fig. 4(a), where the single 3 to 8 decoder common to all n bits is neglected. A (8:1) 1-bit multiplexer is used for generating each of the n-bits of the CT [see Fig. 4(b)]. The T_{COR} corresponds only to the time required to introduce the CT in the final result, since this term is generated in parallel with the other processing.

The total delay and area values of the modulo $(2^n + 1)$ multiplier structure proposed in this paper can be found in the first row of Tables X and XI, for even values of n. In practice, almost all multipliers have an even number of bits, but similar results for odd values of n can also be easily derived from the

TABLE X
COMPARISON OF DELAYS FOR MODULO $(2^n + 1)$ MULTIPLIERS (n EVEN)

$T_{CPA} = 2 \times \lceil \log_2 n \rceil + 6$		
	$\{T_{PPG}\} + \{T_{CSA}\} + \{T_{COR}\}$	
Multiplier	Diminished-1	Ordinary
Proposed	$\{6\} + \{4 \times d(\lfloor n/2 \rfloor + 1)\} + \{0\}$	$\{5\} + \{4 \times d(\lfloor n/2 \rfloor + 1)\} + \{4\}$
Zimmermann [9]	$\{5\} + \{4 \times d(\lfloor n/2 \rfloor + 1)\} + \{4 + 4 + 1\}$ †	$\{5\} + \{4 \times d(\lfloor n/2 \rfloor + 1)\} + \{4 + 2\}$
Ma [8]	$\{6\} + \{4 \times d(\lfloor n/2 \rfloor + 1)\} + \{4 + 4\}$	$--$
Chaves [12] (Wang [7] improved) ‡	$\{MAX[(2 + 4 \times d(n)), 4 \times d(n-2) + 2 \times \lceil \log_2(\log_2(n-2)) \rceil + 4] + 4\} + \{0\}$	$\{1\} + \{4 \times d(n) + 4\} + \{0\}$

†-adapted architecture for diminished-1 representation; ‡-without Booth recoding

TABLE XI
COMPARISON OF CIRCUIT AREAS FOR MODULO $(2^n + 1)$ MULTIPLIERS (n EVEN)

$A_{CPA} = \frac{3}{2} n \lceil \log_2 n \rceil + 8 \times n$		
	$A_{PPG} + A_{CSA} + A_{COR}$	
Multiplier	Diminished-1	Ordinary
Proposed	$\{9 \times \frac{n^2}{2}\} + \{7 \times (\frac{n}{2} - 1) \times n\} + \{21 \times n\}$	$\{9 \times (\frac{n}{2} + 1) \times n\} + \{7 \times (\frac{n}{2} - 1) \times n\} + + \{(21 + 7) \times n\}$
Zimmermann [9]	$\{9 \times (\frac{n}{2} + 1) \times n\} + \{7 \times (\frac{n}{2} - 1) \times n\} + + \{45 \times n\}$ †	$\{9 \times (\frac{n}{2} + 1) \times n\} + \{7 \times (\frac{n}{2} - 1) \times n\} + \{28 \times n + 3 \times n\}$
Ma [8]	$\{9 \times \frac{n^2}{2}\} + \{7 \times (\frac{n}{2} - 1) \times n\} + 14 \times n$	$--$
Chaves [12] (Wang [7] improved) ‡	$\{(n+2) \times n\} + \{7 \times (n-2) \times n\} + \frac{3}{2} \times n \lceil \log_2(\log_2 n) \rceil + 5 \times \log_2 n + 7 \times n$	$n^2 + 7 \times (n+1) \times n$

†-adapted architecture for diminished-1 representation; ‡-without Booth recoding

TABLE XII
EXPERIMENTAL RESULTS FOR THE ORDINARY REPRESENTATION

	Proposed	Zimmer. [9]	Chaves [12] ‡
n	Delay (ns)		
6	7.01	7.29	6.46
8	9.19	9.46	9.02
10	9.98	10.08	9.94
12	9.98	10.54	10.74
14	9.98	11.28	11.1
16	11.47	11.65	11.1
18	11.57	12.45	12.34
20	12.84	12.88	12.34
22	12.84	12.88	12.34
n	Area (μm^2)		
6	10136	10531	9702
8	16976	17211	16941
10	24735	25430	26325
12	34158	34834	37471
14	44722	46063	50645
16	58025	59478	66273
18	71620	73325	83996
20	85463	88880	102485
22	104161	106113	124328

‡-without Booth recoding

TABLE XIII
EXPERIMENTAL RESULTS FOR THE DIMINISHED-1 REPRESENTATION

	Proposed	Zimmer. [9]†	Ma [8]	Wang [7]‡
n	Delay (ns)			
6	6.77	7.3	7.86	6.6
8	8.39	9.34	8.62	8.53
10	9.76	10.76	10.25	9.46
12	9.76	10.92	10.84	9.46
14	9.76	11.33	11.17	10.44
16	10.85	12.04	11.72	11.08
18	11.97	12.99	13.04	12.16
20	12.1	12.99	13.04	12.16
22	12.1	12.99	13.04	12.16
n	Area (μm^2)			
6	9541	11625	8230	10580
8	15418	18714	14027	19298
10	23695	26870	21634	28034
12	32504	36699	30097	39675
14	43213	48090	40289	53791
16	55983	61427	52677	70489
18	69413	75429	65608	88478
20	85137	91685	80871	107299
22	100662	108841	96927	129601

†-adapted architecture for diminished-1 representation
‡-without Booth recoding

previous equations and tables. In the next section, these values will be compared with the delay and area of the most efficient modulo $(2^n + 1)$ multipliers known to date [7]–[9]. Moreover, since there is not any theoretical model accurate enough to represent real synthesized multipliers, the results obtained with this model will be validated with the help of a real implementation.

A. Example of an Architecture for Modulo $(2^8 + 1)$ Multiplication

Fig. 5 shows an implementation of the proposed architecture for the case of the modulo $(2^8 + 1)$ multiplication in the diminished-1 representation. The inputs (operands) enter from the top of the structure and the partial products are generated with the MUX' component represented in Fig. 4(a). During the computation from the top to the bottom, those products are added in the CSA tree and in a final modulo $(2^8 + 1)$ CPA (MPCA). The two logic gates at the top of the MPCA increment the result [(26) with DE = 1]. It can be seen from Fig. 5 that the proposed architecture is extremely regular, which is a clear advantage for VLSI implementations.

V. EXPERIMENTAL RESULTS

The proposed multiplier architectures were specified in the hardware description language (VHDL) and synthesized using

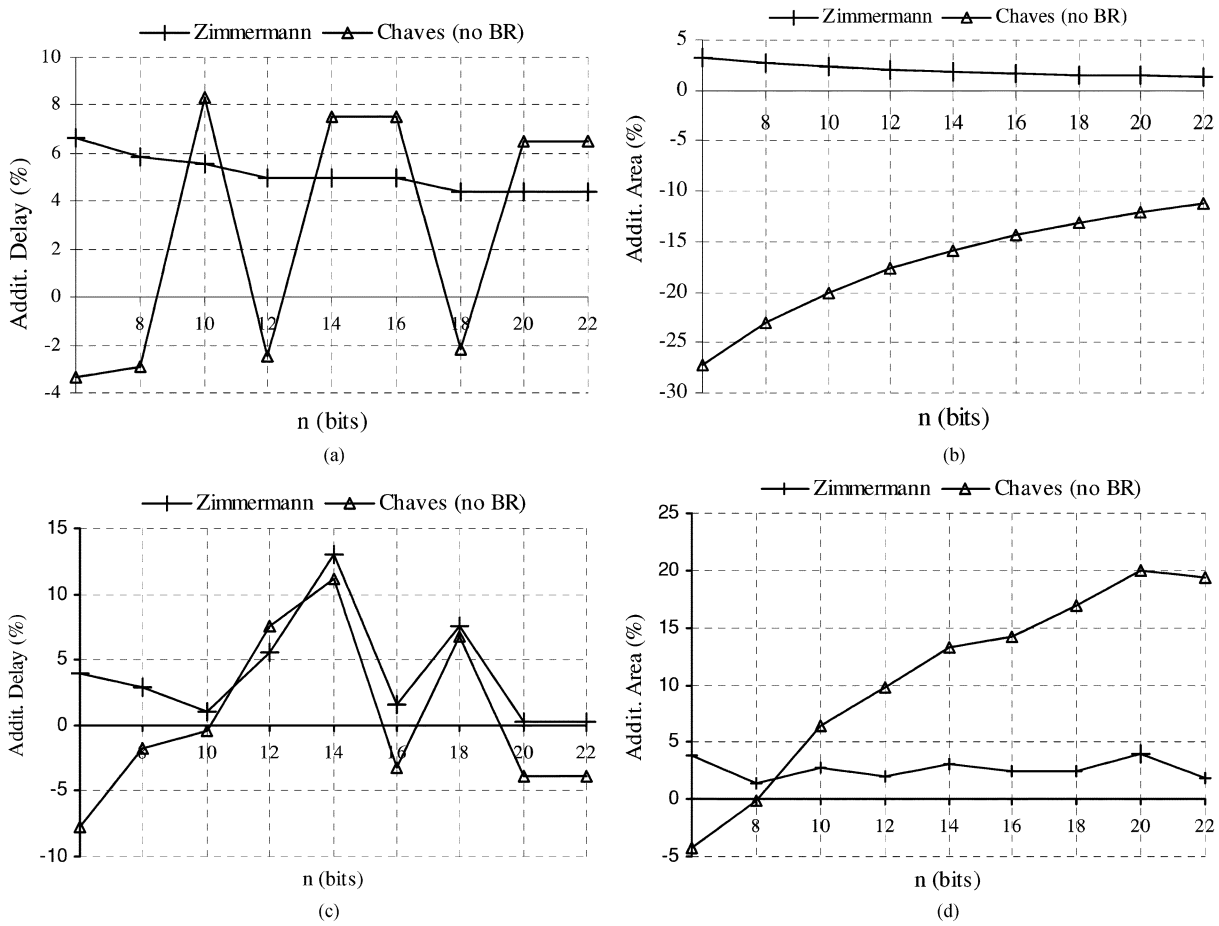

Fig. 6. Relative additional delay and area of multipliers for the ordinary representation, using the proposed structure as reference. (a) (b) Tyagi model. (c) (d) Experimental results.

Synopsys synthesis tools and the Diplomat-25 standard cell library, based on the UMC 0.25-μm CMOS technology process from Virtual Silicon Technology Inc. [17]. Their overall functionality was then thoroughly tested under typical conditions (i.e., typical process, 25 °C and 2.5 V). Some experimental results were also obtained using the recent UMC 0.13-μm CMOS technology process [18] with the purpose of evaluating the impact of interconnect on deeper submicrometer technologies. Unfortunately, the available models for this technology are not so accurate, for example they disregard the net area.

Although all these multipliers have been designed and optimized for a single specific representation, the multiplier presented in Zimmermann [9] was adapted also to the diminished-1 representation by adding the two additional terms and by separately treating the special case of $X, Y = 0$ [see (3)]. To investigate the effectiveness of the use of modified Booth recoding on speeding up modulo $(2^n + 1)$ multiplication, the multiplier described in [7] for the diminished-1 representation was improved by computing the data-dependent CT in parallel with the addition of the partial products on the Wallace-tree and by adding it in a CSA just before the final CPA [12].

Tables X and XI present the modulo $(2^n + 1)$ multipliers' delay and area computed with the Tyagi model. For the sake of simplicity, the addition of a constant CT, required by all multipliers, is not considered in the computation of delays and areas. The experimental results, presented in Tables XII and XIII, are used to validate those estimated values with the theoretical model.

A. Results for the Ordinary Multipliers

The modulo $(2^n + 1)$ multiplier architecture in [9] for the ordinary representation that applies bit-pair recoding and Wallace tree adders does not directly accommodate the 2^n input value in the CT, requiring an additional circuit. This circuit introduces an additional multiplexer in the critical path, which corresponds to an additional delay of 2 gates (see Table X) and $3 \times n$ gates (see Table XI). The experimental results show that the proposed modulo $(2^n + 1)$ multipliers are always faster and occupy less circuit area than the multipliers in [9]. These results are also shown in Fig. 6, where the proposed multiplier was used as reference. There are certain values of n for which the proposed multiplier is about 10% faster (e.g., for $n = 14$ and $n = 18$). On average, the area is about 3% smaller.

Tables X–XII also present the theoretical and the experimental values obtained with the other modulo $(2^n + 1)$ multiplier architectures, for ordinary representation but without Booth recoding [12]. In about half of the considered values of n, the theoretical estimated delays are smaller for those multipliers than for the proposed multiplier. The charts with the obtained experimental values confirm those results, in spite of curves not being completely coincident. The charts in Fig. 6 show a

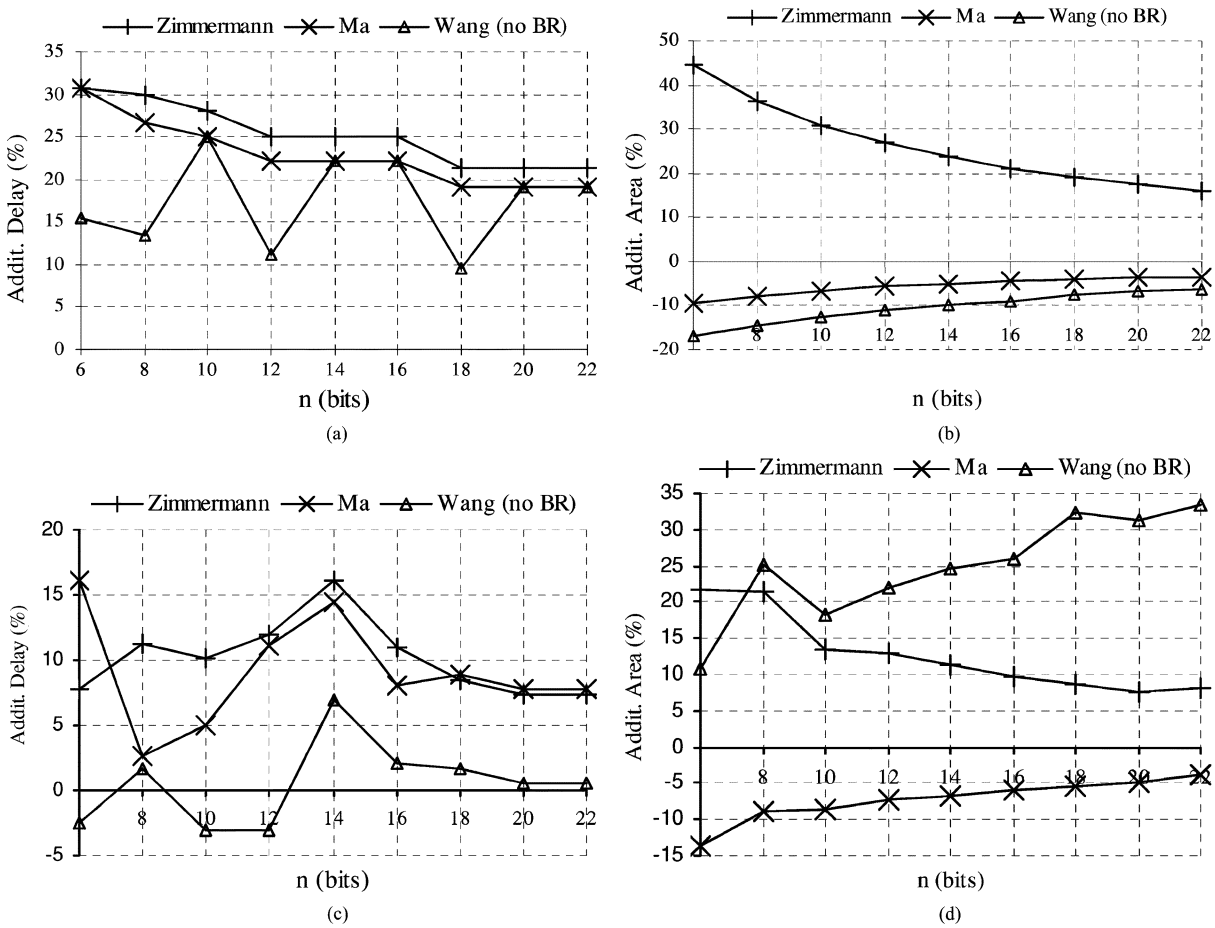

Fig. 7. Relative additional delay and area of multipliers for the diminished-1 representation, using the proposed structure as reference. (a) (b)Tyagi model. (c) (d) Experimental results.

disparity between the theoretical and the experimental results in what concerns the relative additional area. This disparity is mainly due to some inaccuracies in the model of growth of the Wallace tree, since the interconnection area, not considered by the Tyagi model, can be quite significant. In particular, this disparity is greater for multiplication architectures without Booth recoding, such as Chaves [12] and Wang [7], that make use of larger MCSA matrices.

B. Results for the Diminished-1 Multipliers

Table XIII presents the experimental results for the diminished-1 multipliers, in contrast with the theoretical values presented in Tables X and XI. In general, the adaptation of the architecture with Booth recoding proposed in [9] to the diminished-1 representation leads to a multiplier architecture that exhibits the worst efficiency among all considered multipliers. An additional delay of three gates is imposed and $14 \times n$ more gates are required when compared to the ordinary original multiplier. Moreover, Table X also shows that these multipliers impose the longest delay and require the largest area among all studied multipliers with the modified Booth recoding. The usage of modified Booth recoding in [8] implies the addition of a correction value (only dependent on n) and two additional CSA stages in front of the CSA tree (for modulo reduction). Theoretically, its delay is approximately the same as of the Zimmermann multiplier structure, but eight more gates exist in the critical path when compared with the corresponding multiplier structure proposed in this paper. Based on the Tyagi model, the multiplier in [8] exhibits the smallest area, with $7 \times n$ less gates than the proposed multiplier and with $40 \times n$ less gates than the other multipliers.

The plots of the experimental results in Fig. 7 follow the values predicted by the theoretical model. On average, the measured delay is about 10% larger for the multipliers in [8] and [9] than for the modulo $(2^n + 1)$ multiplier proposed in this paper. However, experimental results show that the average circuit area is 7% smaller for the multiplier in [8] than for the proposed one.

Let us analyze the results obtained for the improved modulo $(2^n + 1)$ multipliers without Booth recoding. Both the delay values computed with the Tyagi model and those experimentally obtained evidence that the improved version of the multiplier presented in [7], that does not apply recoding, is preferable to the ones proposed by Zimmermann and Ma that apply this technique. On average, the former is about 10% faster than the latter. For the diminished-1 representation, the architecture proposed in this paper is the only one that really takes advantage of Booth recoding to speedup modulo $(2^n + 1)$ multiplication: it is faster both on average terms and for 7 of the 10 possible values of n.

TABLE XIV
EXPERIMENTAL RESULTS FOR POWER CONSUMPTION

	Power dissipation (mW) with an operating voltage of 2.5 V						
	Diminished-1				Ordinary		
n	Proposed	Zimmer. [9]†	Ma [8]	Wang [7]‡	Proposed	Zimmer. [9]	Chaves [12] ‡
6	47	55	41	52	57	52	58
8	78	90	71	89	92	84	95
10	119	133	105	140	135	126	139
12	164	182	146	189	187	173	203
14	221	238	199	259	244	229	261
16	282	308	256	339	313	295	337
18	357	387	321	421	390	366	431
20	431	470	393	519	475	443	529
22	516	555	471	624	561	531	618

†-adapted architecture for diminished-1 representation; ‡-without Booth recoding

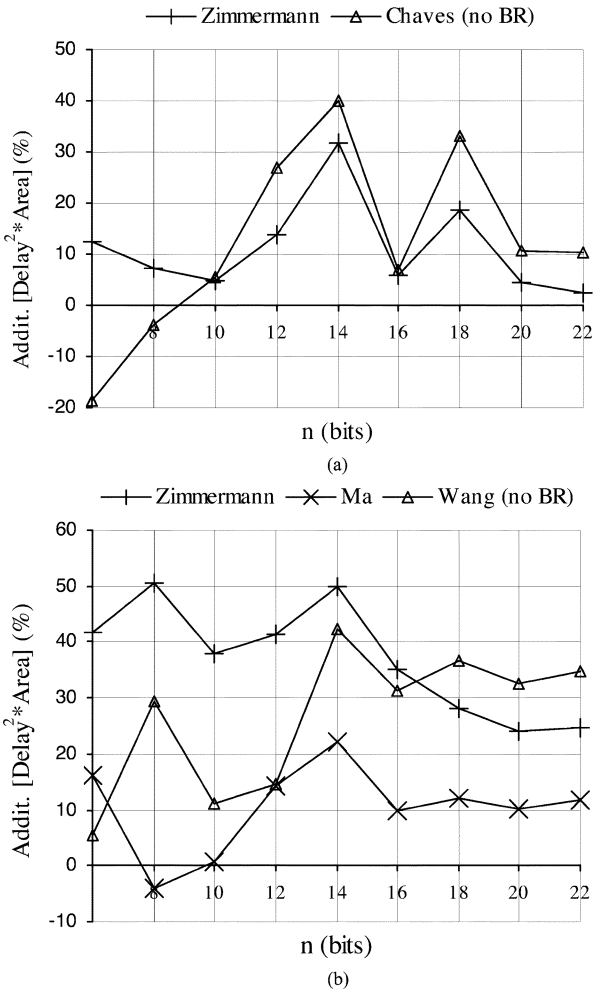

Fig. 8. Experimental results of the product $T^2 \times A$ for the considered multipliers, using the proposed structure as reference. (a) Ordinary representation. (b) Diminished-1 representation.

C. Efficiency and Power Consumption

The charts in Fig. 8 compare the relative efficiency of the multipliers according to the figure of merit given by the product of the square of the delay by the area $(T^2 \times A)$. For the ordinary representation, it is shown that the proposed modulo $(2^n + 1)$ multiplier is more efficient than all the other multipliers except for small values of n, where the modified Booth recoding does not provide an increase in the efficiency (for $n \leq 8$). The efficiency of the proposed multiplier can be up to 30% and 40% greater than the Zimmermann structure and the Chaves multiplier $(n = 14)$, respectively; these figures become 11.3% and 12.4% when average values are considered. The chart in Fig. 8(b) shows that the modulo $(2^n + 1)$ multiplier proposed in this paper is much more efficient than any of the already known multipliers using diminished-1 representation. On average, the proposed multiplier is 10% more efficient than the multiplier in [8], 25% more efficient than the improved version of the multiplier in [7], and 37% more efficient than the multiplier in [9].

Beyond performance and cost, power consumption becomes a measure of growing importance to evaluate the efficiency of electronic systems. The power consumption of all the considered multipliers was estimated with post-synthesis information using the Power Compiler software tool [19]. Table XIV presents the obtained values for the power consumption with an operating voltage of 2.5 V. The charts in Fig. 9 plot the relative power consumption of the different multipliers, by taking the proposed ones as reference. A first and important conclusion that can be drawn from these results is that Booth recoding considerably reduces the power consumption of the multipliers, by reducing the switching activity. Among the considered multipliers, the proposed ones are only surpassed by the one from Zimmermann for the ordinary representation (about 7%) and by the mutiplier proposed by Ma [8] in the class of diminished-1 multipliers (about 10%). It must be noticed that those values have been obtained for the maximum allowed operating frequency of each multiplier, which is greater for the case of the proposed multipliers.

Some of the considered multipliers, namely for diminished-1 representation, were also synthetized with the UMC 0.13 μm CMOS technology process [18], in order to evaluate the interest of the proposed multipliers for deep submicrometer technology implementations. The UMC L130E SG-HS 1P8M process is a single poly, 8-layer metal process with an operating voltage of 1.2 V, while the UMC 0.25 μ 1P5M process is a single poly, 5-layer metal process with an operating voltage of 2.5 V. Experimental results show that switching from 0.25- to 0.13-μm process leads to a reduction of the delay by a factor which ranges between 2 and 3 and of the area by a factor greater than 5. The chart in Fig. 10(b) shows that the relative values for the additional circuit area are similar to those obtained for the 0.25 μm technology. Nevertheless, the plot of the relative delay in both

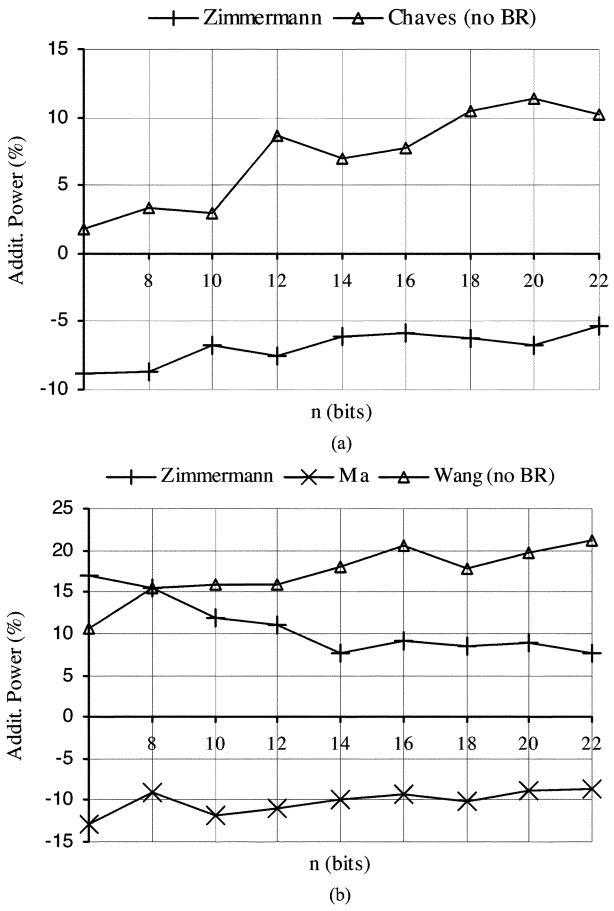

Fig. 9. Experimental results for power consumption, using the proposed multiplier as reference. (a) Ordinary representation. (b) Diminished-1 representation.

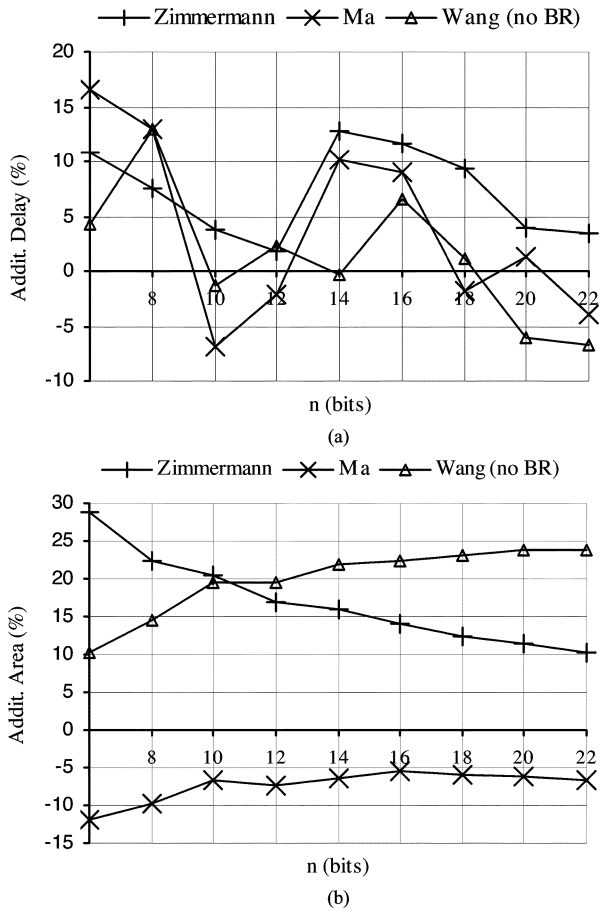

Fig. 10. Relative additional delay and area of multipliers for the diminished-1 representation and 0.13 μm CMOS technology, using the proposed multiplier as reference. (a) Additional delay. (b) Additional area.

technologies, depicted in Figs. 7(c) and 10(a), are not coincident. The effect of interconnect is larger for the 0.13-μm technology and the curve of the delay becomes less regular. However, on average the proposed multiplier continues to be the fastest multiplier: about 5% faster than the Ma and Wang multipliers and about 7% faster than the Zimmermann multiplier. It is also interesting to note that the proposed multipliers are even faster for this technology regarding to the Wang multipliers, increasing about 10 times the difference in delay, from 0.5% to 5%.

VI. Conclusion

This paper proposes a single architecture to design modulo (2^n+1) multipliers with similar efficiency for both the ordinary and the diminished-1 representation. This architecture takes advantage of the modified Booth recoding scheme to simplify and to speedup the correction and the reduction procedures inherent to this type of nonbinary multipliers. A theoretical model was applied to assess the efficiency of the proposed modulo (2^n+1) multipliers and to compare it with the already known modulo (2^n+1) multiplier structures independently of the target technology. The corresponding circuits have been synthetized for a representative set of modulo (2^n+1) multipliers, by using a 0.25 μm StdCell library. The experimental results were used to validate the estimated values with the theoretical model.

The presented results show that the modulo (2^n+1) multipliers based on the proposed architecture are, in general, faster and more efficient than all other known multipliers. Assuming that the product of the area and the square of the delay is a good figure of merit to evaluate the efficiency, it can be concluded that the average efficiency of the proposed multipliers is 11.3% and 12.4% greater than the efficiency of the other two multipliers considered for the ordinary representation. In what concerns the diminished-1 representation, the efficiency of the proposed multipliers is 10%, 25%, and 37% better than the efficiency of the other three multipliers used for comparison. On the other hand, the power consumption estimated for the proposed multiplier is similar to the estimated for the other multiplier structures based on Booth recoding, but is significant less than the power consumption of the multipliers that do no apply recoding. It is also possible to conclude from the obtained results that the architecture proposed in this paper is the first one that really takes advantage of the modified Booth recoding to achieve faster and less expensive modulo (2^n+1) multipliers.

References

[1] P. Mohan, *Residue Number Systems: Algorithms and Architectures*. Norwell, MA: Kluwer, 2002.

[2] M. Menaissa, A. Bouridane, S. Dlay, and A. Holt, "Diminished-1 multiplier for a fast convolver and correlator using the Fermat number transform," in *Proc. Inst. Elect. Eng.—G*, vol. 135, Oct. 1988, pp. 187–193.

[3] A. Ashurand, M. Ibrahim, and A. Aggoun, "Novel RNS structures for the moduli set $(2^n-1, 2^n, 2^n+1)$ and their application to digital filter implementation," *J. Signal Process.*, vol. 46, pp. 331–343, 1995.
[4] R. Chaves and L. Sousa, "RDSP: A RISC DSP based on residue number system," in *Proc. Euro. Symp. Digital System Design: Architectures, Methods, and Tools*, Antalya, Turkey, Sep. 2003, pp. 128–135.
[5] A. Curiger, H. Bonnenberg, and H. Kaeslin, "Regular VLSI architectures for multiplication modulo (2^n+1)," *IEEE J. Solid-State Circuits*, vol. 26, no. 7, pp. 990–994, Jul. 1991.
[6] A. Hiassat, "New memoryless $\mathrm{mod}(2^n \pm 1)$ residue multiplier," *Electron. Lett.*, vol. 28, no. 3, pp. 314–315, Jan. 1992.
[7] Z. Wang, G. A. Jullien, and W. C. Miller, "An efficient tree architecture for modulo (2^n+1) multiplication," *J. VLSI Signal Process. Syst.*, vol. 14, no. 3, pp. 241–248, Dec. 1996.
[8] Y. Ma, "A simplified architecture for modulo $\mathrm{mod}(2^n+1)$ multiplication," *IEEE Trans. Comp.*, vol. 47, no. 3, pp. 333–337, Mar. 1998.
[9] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proc. 14th IEEE Symp. Computer Arithmetic*, Adelaide, Australia, Apr. 1999.
[10] L. Sousa, "Algorithm for modulo (2^n+1) multiplication," *Electron. Lett.*, vol. 39, no. 9, pp. 752–754, May 2003.
[11] L. Leibowitz, "A simplified binary arithmetic for the Fermat number transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-24, no. 5, pp. 356–359, Oct. 1976.
[12] R. Chaves and L. Sousa, "Improved 2^n+1 multipliers,", INESC-ID, Tech. Rep. RT/14/2003, Jul. 2003.
[13] I. Koren, *Computer Arithmetic Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[14] X. Lai and J. Massey, "A proposal for a new block encryption standard," in *Advances in Cryptology-EUROCRYPT 90*, I. B. Damgård, Ed. New York: Springer-Verlag, 1991, vol. 473, pp. 389–404.
[15] A. Tyagi, "A reduced-area scheme for carry-select adders," *IEEE Trans. Comp.*, vol. 42, no. 10, pp. 1163–1170, Oct. 1993.
[16] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. dissertation, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1998.
[17] VST, Virtual Silicon Technology Inc., "Diplomat-25 standard cell library-0.25 μm UMC process," VST, Virtual Silicon Technology Inc., Sunnyvale, CA, Tech. Rep., Dec. 1999.
[18] P. Malisse, "UMC L130E FSG-HS 1P8M logic process with VST libraries," IMEC, Leuven, Belgium, Tech. Rep., Dec. 2003.
[19] J. Flynn and B. Waldo, "Power management in complex SoC Design," Synopsys, Mountain View, CA, Tech. Rep., Apr. 2004.

Leonel Augusto Sousa (M'01–SM'03) received the Ph.D. degree in electrical and computer engineering from the Instituto Superior Técnico (IST), Technical University of Lisbon, Lisbon, Portugal, in 1996.

He is currently a Professor in the Electrical and Computer Engineering Department at IST, and a Senior Researcher at Instituto de Engenharia de Sistemas e Computadores—R&D (INESC-ID). His research interests include computer architectures, parallel and distributed computing, very large-scale integration architectures, and multimedia processor systems. He has contributed more than 67 papers to journals and international conferences.

Dr. Sousa is a member of HiPEAC European Network of Excellence and of ACM.

Ricardo Chaves received the graduate and M.Sc. degrees in electronics and computer science, from the Instituto Superior Télcnico (IST), Technical University of Lisbon, Lisbon, Portugal, in 2001 and 2003, respectively. He is working toward the Ph.D. degree in electrical engineering at the Instituto de Engenharia de Sistemas e Computadores—R&D (INESC-ID), a research institute associated with IST.

He is also a Researcher at INESC-ID. His research interests include architecture and design of very large-scale integration circuits.