2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Efficient Diminished-1 Modulo 2^n+1 Multiplier Architectures

Xiaolan Lv
1.School of Electronic and Information Engineering
South China University of Technology
Guangzhou, China
jingling220@126.com

Ruohe Yao
School of Electronic and Information Engineering
South China University of Technology
Guangzhou, China
phrhyao@scut.edu.cn

***Abstract*—The main components of an artificial neuron are adders and multipliers. In order to implement neural network, large number of adders and multipliers are required. The efficient architectures for diminished-1 modulo 2^n+1 multipliers are described. The results and operands of the new modulo 2^n+1 multipliers use the diminished-1, avoiding *n*+1 bit circuit. And the presented multipliers can handle zero inputs and results. The proposed modulo 2^n+1 multiplier are built using three major functional modules, partial products generation block, partial products reduction block and a final diminished-1 adder block. The final modulo 2^n+1 addition block is built around a sparse carry computation unit for the analytical and experimental results. And this indicates that the significant area and power of the proposed multipliers is superior to the earlier proposals, with a high operation speed.**

Keywords—Residue number system (RNS); Diminished-1 representation; modular multiplier; VLSI

I. INTRODUCTION

This Residue number system (RNS) is an efficient alternative number system which has been an important research field in computer arithmetic for many decades. One of the key advantages of RNS is a carry-free number system, which can represent number in a non-weighted form. High speed and less hardware complexity could be achieved by decomposing a large binary number into a set of smaller residues [1-2]. RNS has drawn widespread attention for design of FIR filters [3], digital signal processors [4] and communication components [5]. The main components of an artificial neuron are adders and multipliers. In order to implement neural network, large number of adders and multipliers are required.

Arithmetic modulo 2^n+1 is most commonly met as a part of a RNS, which plays an important role in many algorithms. Modulo 2^n+1 multiplier is one of the critical components. It is used in pseudorandom number generation, cryptography and convolution computations without round-off errors. Typically, operands and results are presented in weighted representation. However, a number in the range of $[0, 2^n]$ is required (n+1) bits for its representation. In order to overcome the problem of n+1 bits wide circuits, to all channels operate on n-bit ones, Leibowitz [6] introduced the diminished-1 number system. In the diminished-1 number system, each number X is represented by $X=x_zX$-1, and the n-bit diminished-one operand are denoted by $X_{-1}=X-1=x_{n-1}\cdots x_1x_0$. Therefore, the diminished-1 modulo 2^n+1 arithmetic are combinational circuits accepting n-bit operands. But special treatment is required zero handling of diminished-1 number system, which is an attractive problem.

Numerous algorithms and architectures have already been published for modulo 2^n+1 multiplier[7-12]. It is well known that the number of partial products is cut in half by using Booth recoding, which leads to a shallower adder tree. But, this saving may be overwhelmed by complexity of the recoding logic generation. A few of non-Booth encoded modulo 2^n+1 multipliers have recently been investigated in the work of [9-12]. In the work of Ma [7], bit-pair radix-4 Booth recoding technique is used to modulo (2^n+1) multipliers. The multipliers accept operations in diminished-1 representation. The number of the PP was reduced to approximately n/2 at the cost of two additional modulo (2^n+1) adders. Sousa and Chaves proposed the modified radix-4 Booth recoding modulo 2^n+1 multipliers in [8]. The modulo multipliers use diminished-1 representation to handle zero operands. But correction term generator (CTG) is a complex combinational circuit that leads to complexities in the partial product generator. Cruiger et al. [9] proposed multipliers in which one input uses the diminished-1 representation with n-bit wide, whereas the other uses weighted representation. However the usefulness of the proposed multipliers appears to be limited to FIR filters and cryptographic application for some special applications. The modulo 2^n+1 multiplier is proposed in the work of Chen et al [10] with result and one multiplicand use weighted representation, while the other multiplicand uses diminished-1 representation. This is also done for achieving an efficient dedicated design block for which they were originally intended. In the work of Wang et al. [11], the diminished-1 number representation is used with n-bit input operands, and a zero partial-product counting circuit is required. However, zero operands and results were not handed. Efstathiou et al. [12] proposed a diminished-1 multiplier without Booth recoding. The multiplier uses an $n\times(n+3)$ partial-product array along with a CSA tree. These multipliers were analytically and experimentally shown to outperform those of Wang et al. [11] and Ma [7] in terms of delay and power, whereas treatment of zero operands or results was not considered.

This Project was supported by the National Natural Science Foundation of China (*No. 61274085*).

In this manuscript, we propose a novel architecture, for diminished-1 representation of the result and both inputs, which utilizes the observations made in the work of Efstathiou et al. [12]. The two improvements of the proposed architecture lie in:

- Process of zero operands and results was considered.
- A sparse tree based modulo2^n+1 adder was used in the last stage addition, which has less number of carry merge cells and less inter-stage wiring. The use of the sparse tree adder results in a very efficient design of the modulo 2^n+1 multiplier.

The proposed modulo 2^n+1 multiplier architecture is based on CSA tree and sparse tree modulo 2^n+1 adder, and handling of zero inputs and results was considered. The analytical result shows that the proposed multiplier outperforms the solutions described in [3-12].

II. BACKGROUND

In the following the representation of diminished-1 operands proposed in [13], which includes zero indication bit is adopted. According to this representation a number $X\in[0, 2^n]$ is represented as x_zX_{-1}, where x_z is zero indication bit and X_{-1} is magnitude representation. Terms x_z and X_{-1} are defined as follows:

$$x_z=\begin{cases}0 & if\ \ X\neq 0\\ 1 & if\ \ X=0\end{cases}$$
$$X_{-1}=\begin{cases}X-1 & if\ \ X\neq 0\\ 0 & if\ \ X=0\end{cases} \qquad (1)$$

When $X\neq0$, $X_{-1}\in$ [0, 2^n-1] is an n-bit wide number, hence, (n+1)-bit circuits can be avoided in this case. Let $|X|_y$ denote the modulo Y residue of X. However, when $X=0$, $X_{-1}=|0-1|_{2^n+1}=|-1|_{2^n+1}=|2^n+1-1|_{2^n+1}=2^n$ is an (n+1)-bit number, the representation of 0 is treated in a special way.

III. ALGORITHM

In this section, a new architecture for modulo 2^n+1 multiplier is proposed, the result and two inputs uses diminished-1 representation. Suppose that $a_zA_{-1}=a_zA_{n-1}A_{n-2}...A_0$ is the diminished-1 representation of multiplicand A and $b_zB_{-1}=b_zB_{n-1}B_{n-2}\ldots\ B_0$ multiplier B, $q_zQ_{-1}=q_zQ_{n-1}Q_{n-2}...Q_0$ is the diminished-1 representation of $A\times B$ modulo 2^n+1 respectively. a_z, b_z, and q_z are zero indication bits, A_{-1}, B_{-1}, and Q_{-1} are the diminished-1 numbers. According to handle zero inputs and result, the product of a diminished-1 modulo multiplier is derived according to the following cases:

1. When one of the two inputs is zero, the result is zero.

2. When none of the input operands is zero, the result is zero.

3. When none of the input operands is zero, the result is nonzero.

We distinguish the three cases by $a_z\vee b_z=1$; $\overline{a_z\vee b_z}=1$, and $A\times B=2^n+1$; $\overline{a_z\vee b_z}=1$, and $A\times B\neq 2^n+1$. In the following, we use the notations "□","□", and "–" to denote AND, inclusive-OR, and complement operations respectively.

A. When $a_z\vee b_z=1$ (inputs and result are zero)

In this case, a_z=1 or b_z=1, or a_z=1 and b_z=1. The value of Q is equal to 0. This case can be handled by setting the output of each Q_{-1} to 0, and q_z to 1.

B. When $\overline{a_z\vee b_z}=1$, $A\times B=2^n+1$ (inputs are nonzero, result is zero)

In this case, a_z=0 and b_z=0. But $A\times B$=2^n+1, so $|A\times B|_{2^n+1}=0$. The value of Q is equal to 0. This case can be handled by setting the output of each Q_{-1} to 0, and q_z to 1. According to the diminished-1 modulo 2^n+1 arithmetic, operations [6, 11] are defined as following ,

$$\begin{aligned}&|Q_{-1}|_{2^n+1}\\ &=|A\times B-1|_{2^n+1}\\ &=|(A_{-1}+1)(B_{-1}+1)-1|_{2^n+1}\\ &=|A_{-1}\times B_{-1}+A_{-1}+B_{-1}|_{2^n+1}\\ &=|C+S+1|_{2^n+1}\end{aligned} \qquad (2)$$

In (2), C and S denote the carry and sum output vectors of the inverted end around carry save addition (CSA) tree. From $A\times B$=2^n+1, we can further derive C+S=2^n. It then holds C+S=2^n-1, that is, C and S are bit-wise complementary. This condition can be easily detected as the logical AND of the XOR of the bits of C and S vectors with the same weight, and let H denotes the signal. It should be noted that this logic operation will not add any delay on the critical path of the modulo multiplier.

C. When $\overline{a_z\vee b_z}=1$, $A\times B\neq 2^n+1$ (inputs and result are nonzero)

In this case, A_{-1}, $B_{-1}\neq0$, $A_{-1}=|A-1|_{2^n+1}$ and $B_{-1}=|B-1|_{2^n+1}$ are two n-bit numbers, for the diminished-1 modulo 2^n+1 multiplier of A with B, it holds that

$$\begin{aligned}Q_{-1}&=|Q-1|_{2^n+1}=|AB-1|_{2^n+1}\\ &=|(|A_{-1}\times B_{-1}|_{2^n+1}+A_{-1}+B_{-1}|_{2^n+1}\end{aligned} \qquad (3)$$

Taking into account that i+j≤2n−2, the first partial product vector $|A_{-1}\times B_{-1}|_{2^n+1}$ of relation (2) can be written as

$$\left|A_{-1}\times B_{-1}\right|_{2^n+1}=\left|\sum_{i=0}^{n-1}A_i 2^i\sum_{j=0}^{n-1}B_j 2^j\right|_{2^n+1}$$
$$=\left|\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}A_iB_j 2^{i+j}\right|_{2^n+1} \quad (4)$$
$$=\left|\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}A_iB_j\left|2^{i+j}\right|_{2^n+1}\right|_{2^n+1}$$

According to [6, 11], the diminished-1 arithmetic operations are defined as following,

$$\left|2^{ns+i}\right|_{2^n+1}=\begin{cases}\left|2^i\right|_{2^n+1} & n\ \ is\ \ even\\ -\left|2^i\right|_{2^n+1} & n\ \ is\ \ odd\end{cases} \quad (5)$$

And for $x\in\{0,1\}$, it holds

$$\left|-x\right|_{2^n+1}=\left|2^n+1-x\right|_{2^n+1}=\left|2^n+\bar{x}\right|_{2^n+1} \quad (6)$$

Where $\bar{x}$ represents the one's complement of x. From (6), the above relation (4) can be rewritten as sum of partial products

$$\left|A_{-1}\times B_{-1}\right|_{2^n+1}=\left|\sum_{i=0}^{n-1}\sum_{j=0}^{n-1-i}A_iB_j 2^{\left|i+j\right|_n}-\sum_{i=1}^{n-1}\sum_{j=n-i}^{n-1}A_iB_j 2^{\left|i+j\right|_n}\right|_{2^n+1}$$
$$=\left|\sum_{i=0}^{n-1}\sum_{j=0}^{n-1-i}A_iB_j 2^{\left|i+j\right|_n}+\sum_{i=1}^{n-1}\sum_{j=n-i}^{n-1}(2^n+\overline{A_iB_j})2^{\left|i+j\right|_n}\right|_{2^n+1}$$
$$=\left|\begin{array}{l}\sum_{j=0}^{n-1}A_0B_j 2^j\\ +\sum_{i=1}^{n-1}(\sum_{j=0}^{n-1-i}A_iB_j+\sum_{j=n-i}^{n-1}\overline{A_iB_j})2^{\left|i+j\right|_n}+\sum_{i=1}^{n-1}\sum_{j=n-i}^{n-1}2^n 2^{\left|i+j\right|_n}\end{array}\right|_{2^n+1} \quad (7)$$
$$=\left|\sum_{i=0}^{n-1}(P_i+C_i)\right|_{2^n+1}$$

Where P_i denotes the i-th partial product. The correction factor introduced by complementing the A_iB_j terms with weight more than 2^{n-1} and repositioned it to the $(i+j-n)$th column. For each such complementation and repositioning, a corresponding correction factor C_i is taken into account. P_i and C_i can be reformulated as:

$$\begin{cases}P_0=\sum_{j=0}^{n-1}A_0B_j 2^j & i=0\\ P_i=(\sum_{j=0}^{n-1-i}A_iB_j+\sum_{j=n-i}^{n-1}\overline{A_iB_j})2^{\left|i+j\right|_n} & i=1,\cdots,n-1\end{cases} \quad (8)$$

$$C_i=\sum_{j=n-i}^{n-1}\left|2^n 2^{\left|i+j\right|_n}\right|_{2^n+1}=2^n\left(2^i-1\right) \quad (9)$$

According to (8) and (9), the following partial products P_i and C_i correction factors matrix are derived:

$$\begin{array}{lccccccc}
P_0= & A_0B_{n-1} & A_0B_{n-2} & \ldots & A_0B_1 & A_0B_0 & & C_0=0\\
P_1= & A_1B_{n-2} & A_1B_{n-3} & \ldots & A_1B_0 & \overline{A_1B_{n-1}} & & C_1=2^n(2^1-1)\\
P_2= & A_2B_{n-3} & A_2B_{n-4} & \ldots & \overline{A_2B_{n-1}} & \overline{A_2B_{n-2}} & & C_2=2^n(2^2-1)\\
\cdots & & & & & & & \\
P_{n-2}= & A_{n-2}B_1 & A_{n-2}B_0 & \ldots & \overline{A_{n-2}B_3} & \overline{A_{n-2}B_2} & & C_{n-2}=2^n(2^{n-2}-1)\\
P_{n-1}= & A_{n-1}B_0 & \overline{A_{n-1}B_{n-1}} & \ldots & \overline{A_{n-1}B_2} & \overline{A_{n-1}B_1} & & C_{n-1}=2^n(2^{n-1}-1)
\end{array}$$

The total correction, C_t, required for the formation of the above n partial products is equal to

$$C_t=\sum_{j=0}^{n-1}C_i=\sum_{j=1}^{n-1}2^n\left(2^i-1\right)$$
$$=2^n\left(2^i-1-n\right) \quad (10)$$

In order to further reduce the partial products into two summands, we consider that the reduction of the partial products is performed by using modulo carry-save addition. Tree architectures have been introduced by Dadda [14]. And the first speed-up technique for multiplication is to accelerate the addition of the partial products using a carry-save adder tree (Dadda tree) [14], a Carry, Sum vector pair (C, S) is reached by a n+1 stage Carry Save Adder (CSA) array. At the same time, we must consider correction factor introduced during the reduction of the partial products into two summands. Assume that c_n is the carry output at the most significant bit position, each carry bit with weight 2^n, since

$$\left|2^n c_n\right|_{2^n+1}=\left|-c_n\right|_{2^n+1}=\left|2^n+\overline{c_n}\right|_{2^n+1} \quad (11)$$

According to the above formulate, c_n can be complemented and placed at bit position 0 of the next stage, that is, in the next to the least significant bit, and introducing a correction factor 2^n for each such operation. Since n+1 CSA stages are required for the reduction of the n+3 partial products into two summands (a Carry and Sum vector pair (C, S)), the total correction by n+1 CSA stages is

$$C_f=(n+1)2^n \quad (12)$$

The last correction factor for the entire partial product matrix can be calculated from the sum of C_p and C_f as follows:

$$\left|C\right|_{2^n+1}=\left|C_t+C_f\right|_{2^n+1}$$
$$=\left|2^n\left(2^i-1-n\right)+(n+1)2^n\right|_{2^n+1}=1 \quad (13)$$

The constant '1' in equation (13) is the last correction factor, C_{-1}=C-1=0 is its diminished-one number representation, that is, a total correction 0 ... 0000 is introduced. It is important to note that the correction term should have the zero value non-null values, since in this case less than n+1 the computed carries of weight 2^n will be produced if it is ignored during the reduction of the partial products. According to the previous discussion, the product Q_{-1} of relation (1) can be rewritten as:

$$\begin{aligned} Q_{-1} &= \left|Q-1\right|_{2^n+1} = \left|AB-1\right|_{2^n+1} \\ &= \left|P_i + A_{-1} + B_{-1} + C_{-1}\right|_{2^n+1} \end{aligned} \tag{14}$$

As a result, we get a completely rectangular array, and the partial product complete matrix of the n+3 partial product of the proposed diminished-1 modulo 2^n+1 multiplier is given by:

$$\begin{array}{lccccc}
P_0 = & A_0B_{n-1} & A_0B_{n-2} & \ldots & A_0B_1 & A_0B_0 \\
P_1 = & A_1B_{n-2} & A_1B_{n-3} & \ldots & A_1B_0 & \overline{A_1B_{n-1}} \\
P_2 = & A_2B_{n-3} & A_2B_{n-4} & \ldots & \overline{A_2B_{n-1}} & \overline{A_2B_{n-2}} \\
\ldots & & & & & \\
P_{n-2} = & A_{n-2}B_1 & A_{n-2}B_0 & \ldots & \overline{A_{n-2}B_3} & \overline{A_{n-2}B_2} \\
P_{n-1} = & A_{n-1}B_0 & \overline{A_{n-1}B_{n-1}} & \ldots & \overline{A_{n-1}B_2} & \overline{A_{n-1}B_1} \\
P_n = & A_{n-1} & A_{n-2} & \ldots & A_1 & A_0 \\
P_{n+1} = & B_{n-1} & B_{n-2} & \ldots & B_1 & B_0 \\
C_{-1} = & 0 & 0 & \ldots & 0 & 0
\end{array}$$

In order to further reduce inter-stage wiring and large carry merge cell density, a new end-around inverted carry (EAIC) adder based on sparse tree is used for the last stage modulo addition. The modulo addition accepts Sum vector and Carry vector from the previous stage and produces the required product.

Thus, the new diminished-1 modulo 2^n+1 multiplier can not only avoid (n+1)-bit arithmetic circuits in the computation process, but also handle zero inputs and results.

IV. Architecture for Diminished-1 Modulo Multipliers

From the above arithmetic algorithms discussed, the proposed implementation of the modulo 2^n+1 multiplier consists of three modules, that is, partial products generation module, partial products reduction module and the last diminished-1 modulo 2^n+1 adder module. For the sake of presenting the complete implementation of the proposed multiplier, it is briefly described below.

The first module is to generate partial products. Basic logic gates of AND or NAND gates that form a bit of each partial product.

The second module is to reduce the partial products to two last summands. The n+3 partial products can be reduced into two summands by the CSA tree architecture indicated in Fig. 1 for n=8. It is well known that the irregularity of the architecture cause layout difficulties if Dadda tree is used to build a integer multiplier. In the paper, it can be seen from Fig.1 that the FA and HA based adder array for the Dadda tree is extremely regular, and is therefore easier to implement than the Dadda tree binary multiplier, especially when n becomes large. The CSA tree is usually constructed with FA. But in our multipliers, since C_{-1} is the all 0s vector, one stage of the CSA tree that accepts this operand can be further simplified only using a row of half adders (HAs).

The partial products reduction module is composed only by FA and HA blocks. Each such block produces a carry at its most significant bit position, which are complemented and added to the bits of the least significant bit position.

The last module is to add the sum and carry vectors from partial products reduction module to produce the required product. Fig.2 presents the proposed Prop/4 32-bit EAIC adders. The notation Prop/k is used to denote the proposed modulo adders in which every kth (k=2, 4, 8…) carry is computed, instead of calculating the all carry terms for every bit position. For a 32-bit sparse IEAC with sparseness factor equal to 4, that is, k=4, the carries are computed for bit positions −1, 3, 7, 11, 15, 19, 23 and 27. Bit position −1 corresponds to the inverted carry out of the bit position 31. The logic level implementation of the carry select block is shown in Fig. 3. For larger adders, the sparse version of the modulo adders introduced in this paper with less number of cells and less inter-stage wiring, which provid better performance in the proposed multiplier. The implementation of logic-level used in adders that is given in Fig. 4. For computing the most significant bit of Q, equivalently, zero indication bit q_z. Taking into account that q_z should be set to 1 if $a_z \vee b_z = 1$; or $\overline{a_z \vee b_z} = 1$ and $A \times B = 2^n + 1$. We conclude that q_z can be computed straightforwardly as $a_z \square b_z \square H$. Fig. 1 presents an example of the proposed multiplier architecture for the case that n=8. It should be noted that handle 0 operands in diminished-1 representation did not add any delay on the critical path of the proposed multiplier.

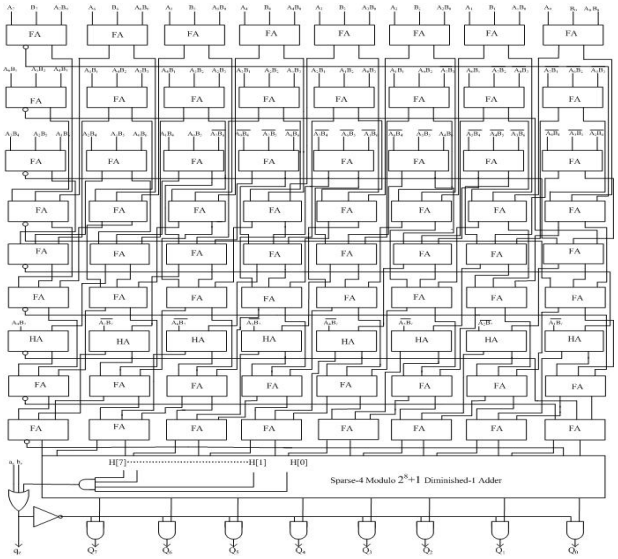

Fig. 1. The proposed modulo 2^8+1 multiplier

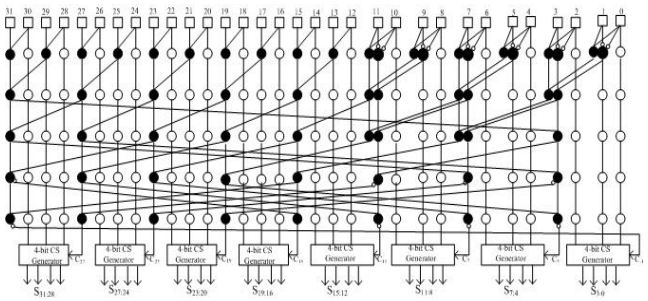

Fig. 2. Prop-*n*/4 32-bit EAIC adders.

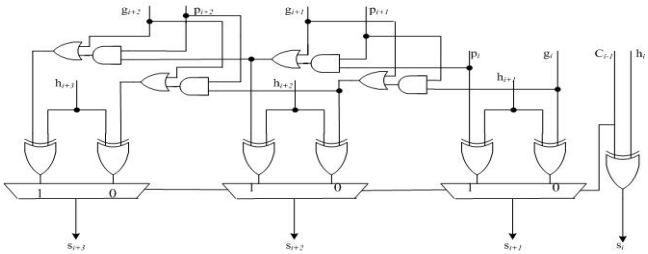

Fig. 3. The logic level implementation of the carry select

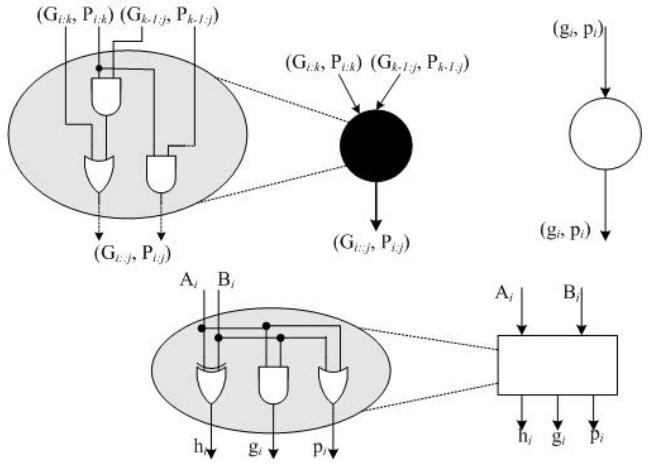

Fig. 4. The logic-level implementation of the basic cells used in adders.

Example 1: When n=8, let A=154, B=199, then $A_{-1}=(99)_{16}$, $B_{-1}=(C6)_{16}$ and $\overline{a_z \vee b_z}=1$.According to Figs. 1,these eleven partial products are diminished-1 added and the result is $Q_{-1}=(3E)_{16}$, and H=0 can be detected by the logical AND of the logical XOR of final S_i and C_i, we can further have $q_z = a_z \vee b_z \vee H = 0$. Fig. 5*a* shows the computation process.

Example 2: When n=9, A=17, B=26, then $A_{-1}=(013)_{16}$, $B_{-1}=(01A)_{16}$ and $\overline{a_z \vee b_z}=1$. These twelve These eleven partial products are diminished-1 added and the result is $Q_{-1}=(000)_{16}$, and H=1 can be detected by the logical AND of the logical XOR of final S_i and C_i. Since $q_z = a_z \vee b_z \vee H = 1$ we have $\overline{q_z}=0$ that will provide the expected diminished-one representation result $Q_{-1}=(00)_{16}$ and $q_z = 1$. Fig. 5*b* shows the computation process.

Example 3:When n=8, A=0, B=199, then $A_{-1}=(100)_{16}$, $B_{-1}=(C6)_{16}$ and $\overline{a_z \vee b_z}=0$. Since $q_z = a_z \vee b_z \vee H = 1$, we have $\overline{q_z}=0$ that will provide the expected diminished-one representation result $Q_{-1}=(00)_{16}$ and $q_z = 1$.

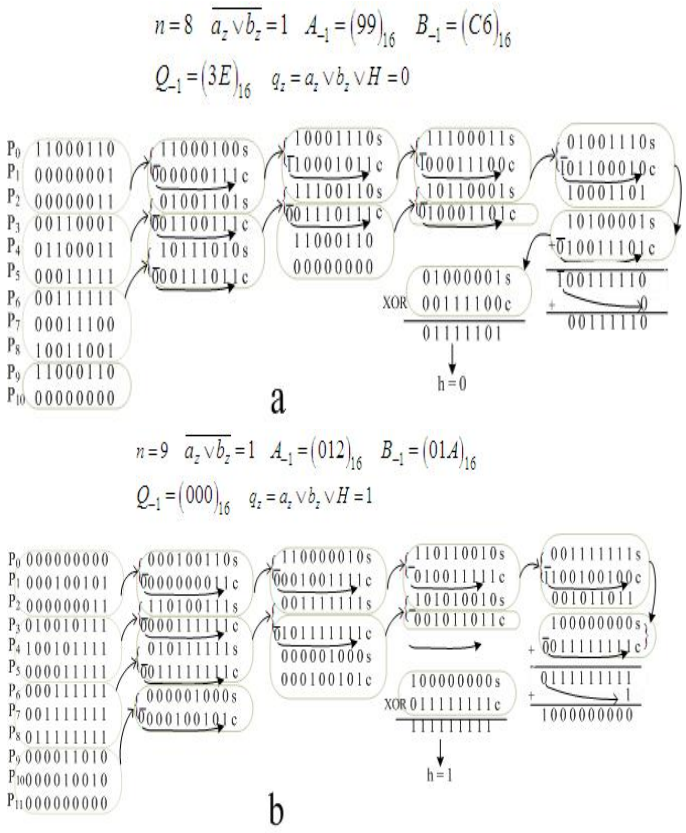

Fig. 5. Examples of diminished-1 modulo 2^n+ 1 multiplication
a :Numeric illustration of Example 1 b : Numeric illustration of Example 2

V. HARDWARE ANALYSIS

In this section, we will analyze the area and time complexity of the proposed multipliers. For our analysis, we adopt the approximations of the unit gate model proposed in the work of Tyagi [15], that is, we consider that all 2-input monotonic gates are computed in 1 gate equivalent for both area and delay, while a 2-input XOR or XNOR gate counts as 2 gate equivalents for both area and delay. Table 1 presents the area and delay of logic-gates and basic-cells in gate equivalents according to this model approximately.

For the proposed modulo multipliers, the delay and area results for any values of n are given in the follow. The area(A) and delay(T) requirements of the new multipliers consist of four parts, which are the partial products generation (PPG), the partial products reduction (PPR) , the last stage addition (FSA) and the handle zero operands(HZO) modules respectively, they can be written as

$$A_{PROPOSED} = A_{PPG} + A_{PPR} + A_{FSA} + A_{HZO} \qquad (15)$$

$$T_{PROPOSED} = T_{PPG} + T_{PPR} + T_{FSA} + T_{HZO} \quad (16)$$

The required $n\times(n+3)$ partial products bits can be generated in parallel uses n^2 AND or NAND gates, so $A_{PPG}=n^2$, $T_{PPG}=1$.

We consider that these patial products are then reduced to two summands by the use of the CSA tree. The depth in FA stages of a Dadda tree is a function, suppose D(k), where D(k) denotes the depth in FAs of a k-operand CSA tree. D(k) is listed in Table 2 for all practical values of k. Therefore, the CSA tree of the proposed multiplier has D(n+3) stages, each of the n columns of the tree. Since a FA is equivalent to a compressor with fator of 3:2, we have to use (n+1) rows of n FAs each for reducing the n+3 partial product matrix to the 2 n-bit vectors that will then be added by the last adder. Furthermore, since $C_{-1}=0$ is the all 0s vector, the row of FAs can be simplified to a row of HA. The total area of partial products reduction is $A_{PPR}=7n^2+3n$, while its execution delay is $T_{PPR}=4D(n+3)$.The area and the delay of the sparse tree based Inverted EAC in the stage [16] are

$$A_{FSA} = 3\left(\left\lceil \frac{n}{4} \right\rceil + 1\right)\left\lceil \log 2^n \right\rceil + 25\left\lceil \frac{n}{2} \right\rceil - 9 \quad (17)$$

$$A_{FSA} = 2\left\lceil \log 2^n \right\rceil + 4 \quad (18)$$

The handle zero operands(HZO) module that handle operands in parallel with the other processing, therefore its delay does not count in, except for a 2-input "AND" gate is added in output of the proposed multipliers. Therefore, the delay of the handle zero operands (HZO) module takes 2 unit-gate equivalents delay. That is $T_{HZO}=1$, $A_{HZO}=n+3$.

Summing the above delays, we conclude that the area and the delay of the proposed multipliers can be

$$\begin{aligned} A_{PROPOSED} &= A_{PPG} + A_{PPR} + A_{FSA} + A_{HZO} \\ &= 8n^2 + 4n + 3\left(\left\lceil \frac{n}{4} \right\rceil + 1\right)\left\lceil \log 2^n \right\rceil + 25\left\lceil \frac{n}{2} \right\rceil - 6 \end{aligned} \quad (19)$$

$$\begin{aligned} T_{PROPOSED} &= T_{PPG} + T_{PPR} + T_{FSA} + T_{HZO} \\ &= 4D(n+3) + 2\left\lceil \log 2^n \right\rceil + 6 \end{aligned} \quad (20)$$

TABLE I. UNIT GATE MODEL

Logic gate/Basic cell	Area	Delay
NOT	0	0
AND, OR, NAND, NOR	1	1
XOR, XNOR	2	2
HA	3	2
FA	7	4

TABLE II. FA STAGES IN K OPERAND TREES

k	4	5…6	7…9	10…13	14…19	20…28	29…42	43…63
D(k)	2	3	4	5	6	7	8	9

VI. CONCLUSION

In this paper, we have described a novel architecture for diminished-1 modulo 2^n+1 multiplier. The new architecture adopts a pure radix-4 Booth recoding and a Wallce tree to reduce the number of the partial products and speed up the computation in the partial products reduction stage. Moreover, sparse tree based Inverted-end around carry adder is used in the last stage addition. The new architecture can handle zero inputs and result. The comparisons with the most existed efficient solutions indicate that the speed of the novel multipliers can be the same or better while being more compact. Besides, the very regular structure is well suited to VLSI implementations and direct application for pipeline architecture.

REFERENCES

[1] P.V.A. Mohan, "Residue number systems: algorithms and Architectures," Norwell, MA: Kluwer, 2002.

[2] K Navi, A.S Molahosseini and M. Esmaeildoust, " How to teach residue number system to computer scientists and engineers ," IEEE Trans. Educ. , 2011, 54(1): 156-163.

[3] R. Conway and J. Nelson, " Improved RNS FIR filter architectures ," IEEE Trans. Circuits Syst. II, 2004, 51(1): 26–28.

[4] J. Ramirez, A. Garcia, S. Lopez-buedo, and A. Lloris, "RNS-enabled digital signal processor design," Electron. Lett., 2002, 38(6), 266-268.

[5] A. S Madhukuma and F. Chin, "Enhanced architecture for residue number system-based CDMA for high-rate data transmission," IEEE Trans. Wirel. Commun., 2004, 3(5), 1363-1368.

[6] L. Leibowitz, "A simplified binary arithmetic for the fermat number transform ," IEEE Trans. Acoust. Speech Signal Process, 1976, ASSP-24, 356–359.

[7] Y. Ma, "A simplified architecture for modulo (2^n+1) multiplication," IEEE Trans. Comput., 1998, 47,(3), pp. 333–337.

[8] L. Sousa and R. Chaves, " A universal architecture for designing efficient modulo 2^n+1 multipliers IEEE Trans， " Circuits Syst. I., 2005,52(6), 1166–1178.

[9] A. Curiger, H. Bonnenberg, and H. Kaeslin, "VLSI architectures for multiplication modulo (2^n+1)," IEEE J. Solid-State Circuits, 1991, 26(7), 990–994.

[10] J. W. Chen and R. H Yao, "Efficient modulo multipliers for diminished-1 representation," IET Circuits, Devices Syst., 2010, 4(4), 291–300.

[11] Z Wang, G. A. Jullien, and W. C. Miller, "An efficient tree architecture for modulo (2^n+1) multiplication," VLSI Signal Process., 1996, 14, 241–248.

[12] C. Efstathiou, H. T. Vergos, G. Dimitrakopoulos, and D. Nikolos, "Efficient diminished-1 modulo 2^n+1 multipliers," IEEE Trans. Comput., 2005, 54(4), 491–496.

[13] C. Efstathiou and I. Voyiatzis, "Handling zero in diminished-1 modulo 2^n+1 subtraction," Signals, Circuits and Systems (SCS), 2009 3rd International Conference on. IEEE, 2009, 1-6.

[14] L. Dadda, "On parallel digital multipliers," Alta Frequenza, 1976, 45, 574–580.

[15] A. Tyagi, "A reduced-area scheme for carry-select adders," IEEE Trans. Comput., 1993, 42(10), 1163–1170.

[16] H.T.Vergos and G. Dimitrakopoulos, " On Modulo 2^n+1 Adder Design ," IEEE Trans. Comput., 2012, 61(2), 173–186.