Efficient Modulo $2^n + 1$ Multi-Operand Adders

H. T. Vergos
Computer Engineering & Informatics Dept.
University of Patras,
26500 Patras, Greece
Email: vergos@ceid.upatras.gr

D. Bakalis
Department of Physics
University of Patras
26500 Patras, Greece
Email: bakalis@physics.upatras.gr

C. Efstathiou
Informatics Dept.,
ATEI of Athens
12210 Athens, Greece
Email: cefsta@teiath.gr

***Abstract*— A new architecture for modulo 2^n+1 multi-operand addition (MOMA) of weighted operands is introduced. It is based on the use of a translator circuit that enables to use n-bit operations for performing the weighted multi-operand addition. Our experimental results indicate that the proposed MOMAs offer significant savings in execution time compared to the previously proposed solutions that either require two parallel additions or a carry-save adder tree with twice the depth of the proposed while they can be implemented in less area in most cases.**

I. Introduction

Arithmetic modulo 2^n+1 is used in pseudorandom number generation, cryptography [1], convolution computations without round-off errors [2] but is most commonly met as a part of a residue number system (RNS) [3], which is an arithmetic system well-suited to applications in which the operations are limited to addition, subtraction and multiplication. The RNS has been adopted in the design of digital signal processors [3], [4], FIR filters [5] and communication components [6].

Three-moduli RNS bases of the $\{2^n-1,\ 2^n,\ 2^n+1\}$ form have received significant attention, mainly due to the existence of very efficient combinational converters from / to the binary system. In these RNSs the execution delay is dictated by the modulo 2^n+1 channel, because this has to handle $(n+1)$-bit wide operands. The diminished-1 representation [7] attacks this problem, by having each operand represented decreased by one compared to its weighted representation and by deriving the results in an alternative manner when any operand or result is zero. Let A and B denote two $(n+1)$-bit operands, such that $0 < A,\ B \leq 2^n$. In the diminished-1 representation, $A^\star$ and $B^\star$ are used to represent A and B, with $A^\star = |A-1|_{2^n+1}$ and $B^\star = |B-1|_{2^n+1}$ respectively. Their diminished-1 sum $S^\star$ is computed as :

$$S^\star = |S-1|_{2^n+1} = |A+B-1|_{2^n+1} = |A^\star+B^\star+1|_{2^n+1} \quad (1)$$

by a diminished-1 (equivalently one's complement or end-around-carry) adder [8]. The need for handling zero operands and results separately, as well as, the need for time and hardware consuming input / output translators from / to the weighted to / from the diminished-1 representation, make the diminished-1 representation attractive only when a large number of calculations takes place before a new conversion. In all other cases the weighted representation is more suitable.

One of the arithmetic components that has been heavily researched in residue arithmetic is the multi-operand modulo adder (MOMA). Hardware support for multi-operand modulo addition is highly appreciated in several multiply-and-add intensive computations, such as digital filtering, convolution estimation and FFT transforms. The first effort for a modulo 2^n+1 MOMA for operands in the weighted representation (hereafter denoted as *weighted MOMA*) appeared in [9], but required several parallel-adders connected in series. The problem of designing MOMAs for generalized moduli was attacked in [10]–[12]. Unfortunately, the architecture proposed for the weighted MOMA in [12], although more efficient than those of [10], [11], still requires two parallel adders connected in series to provide its result. The need for two additions was canceled in [13], at the cost of doubling each carry-save adder (CSA) stage of the tree and requiring four carry-lookahead units at the final stage that operate in parallel. On the other hand, modulo 2^n+1 MOMAs for operands in the diminished-1 representation (hereafter denoted as *diminished MOMA*) have been shown a lot easier to design. In [14] it was shown that a diminished MOMA can be implemented by an inverted end-around carry (EAC) CSA tree and a final diminished-1 adder. In the following a k-operand multi-operand adder modulo 2^n+1, will be denoted as a MOMA$(k, 2^n+1)$.

In this paper, a new architecture is introduced for weighted MOMAs. A translator circuit is proposed that enables to express the modulo 2^n+1 sum of weighted operands as a congruent modulo 2^n+1 sum of n-bit operands. The required translator circuit is shown to be a simplified CSA stage; therefore, it has a very small area and time complexity. After the translation, a diminished MOMA is used with minor changes at the final parallel adder so as to provide a correct $(n+1)$-bit weighted result. Since the proposed architecture uses just one parallel addition and simple CSA stages, it outperforms all previous proposals for weighted MOMAs.

II. New MOMA Architecture Derivation

In this section we propose a new architecture for a weighted MOMA, that consists of a translation stage, an inverted EAC CSA tree and a final diminished-1 adder.

A. The Translator Circuit

Suppose that $A = a_n a_{n-1} a_{n-2} \ldots a_1 a_0$ and $B = b_n b_{n-1} b_{n-2} \ldots b_1 b_0$ denote two operands in weighted representation, with $0 \leq A,\ B \leq 2^n$ and A_{n-1} and B_{n-1} denote the n-bit vectors composed by the least significant bits of A

and B respectively. For the weighted modulo 2^n+1 addition of A, B it then holds that :

$$\begin{aligned}\left|A+B\right|_{2^n+1} &= |(2^n \times a_n + A_{n-1}) + (2^n \times b_n + B_{n-1})|_{2^n+1} \\ &= |2^n \times (a_n + b_n) + A_{n-1} + B_{n-1}|_{2^n+1} \qquad (2)\end{aligned}$$

Let s_n and c_n denote the sum and carry bits of the $(a_n+b_n)\times 2^n$ addition contained in (2) respectively and let $\overline{x}$ denote the complement of x. s_n and c_n are bits with weights 2^n and 2^{n+1} respectively. Using them in (2) produces :

$$\begin{aligned}|A+B|_{2^n+1} &= \left|2^{n+1}c_n + 2^n s_n + A_{n-1} + B_{n-1}\right|_{2^n+1} = \\ &= |A_{n-1} + B_{n-1} - 2c_n - s_n|_{2^n+1}\,. \qquad (3)\end{aligned}$$

Since for $x \in \{0,1\}$ it holds that :

$$|-x|_{2^n+1} = |2^n + 1 - x|_{2^n+1} = |\overline{x} - 1|_{2^n+1} \qquad (4)$$

from (3) we can further derive that :

$$\begin{aligned}\left|A + B\right|_{2^n+1} &= |A_{n-1} + B_{n-1} + 2\overline{c}_n + \overline{s}_n - 3|_{2^n+1} \\ &= |A_{n-1} + B_{n-1} + 2\overline{c}_n + \overline{s}_n + 2^n - 2|_{2^n+1} \\ &= |A_{n-1} + B_{n-1} + (2^n - 4 + 2\overline{c}_n + \overline{s}_n) + 2|_{2^n+1} \\ &= \left||A_{n-1} + B_{n-1} + D + 1|_{2^n+1} + 1\right|_{2^n+1} \qquad (5)\end{aligned}$$

where $D = 2^n - 4 + 2\overline{c}_n + \overline{s}_n$, that is the n-bit vector $111\ldots1\overline{c}_n\overline{s}_n$.

Let $Y^\star = y_{n-1}y_{n-2}\ldots y_0 0$ and $U = u_{n-1}u_{n-2}\ldots u_0$ denote the carry and sum output vectors of the carry-save addition of A_{n-1}, B_{n-1} and D indicated in (5), respectively. It then holds that :

$$\begin{aligned}\left|A+B\right|_{2^n+1} &= \left||A_{n-1} + B_{n-1} + D + 1|_{2^n+1} + 1\right|_{2^n+1} \\ &= \left|\left|\sum_{i=1}^{n}(2^i \times y_{i-1}) + \sum_{i=0}^{n-1}(2^i \times u_i) + 1\right|_{2^n+1} + 1\right|_{2^n+1} \\ &= \left|\left|\sum_{i=1}^{n-1}\Big(2^i \times (y_{i-1} + u_i)\Big) + u_0 - y_{n-1} + 1\right|_{2^n+1} + 1\right|_{2^n+1} \\ &= \left|\left|\sum_{i=1}^{n-1}\Big(2^i \times (y_{i-1} + u_i)\Big) + u_0 + \overline{y}_{n-1}\right|_{2^n+1} + 1\right|_{2^n+1} \\ &= \left||Y + U|_{2^n+1} + 1\right|_{2^n+1} \\ &= |Y + U + 1|_{2^n+1}\,, \qquad (6)\end{aligned}$$

where $Y = y_{n-2}y_{n-3}\ldots y_0\overline{y}_{n-1}$. That is, the sum of the weighted operands A and B modulo 2^n+1 is congruent to the modulo 2^n+1 sum of the n-bit vectors Y and U plus 1. We can therefore use the inverted end-around carry save adder stage of Fig. 1, which accepts the vectors A_{n-1}, B_{n-1} and D and produces Y and U as a translator circuit from the weighted to an n-bit pair representation provided that a correction equal to 1 is also taken into account. The $(n-2)$ leftmost full adders (FAs) of Fig. 1, have an area and time complexity equal to that of a half adder since their input bits coming from operand D are equal to 1. The two rightmost FAs along with the accompanying NAND and XNOR gates can also be simplified (Fig. 2 presents examples of simplified circuits) considering that a_n (b_n) and a_1 or a_0 (and b_1 or b_0) can not be simultaneously at 1.

Generalizing the above, given k $(n+1)$-bit numbers, suppose $X_1, X_2, \ldots, X_k$, with $0 \le X_1, X_2, \ldots, X_k \le 2^n$ we may group them in $\lceil \frac{k}{2} \rceil$ pairs and use $\lceil \frac{k}{2} \rceil$ translator circuits in parallel (if k is odd, then X_k is paired with 0) to derive $\lceil \frac{k}{2} \rceil$ pairs of n-bit vectors $Y_1, Y_2, \ldots, Y_{\lceil \frac{k}{2} \rceil}$ and $U_1, U_2, \ldots, U_{\lceil \frac{k}{2} \rceil}$, hereafter referred to as *diminished vectors*, such that :

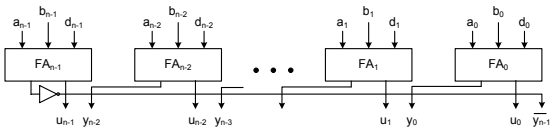

Fig. 1. Translator circuit

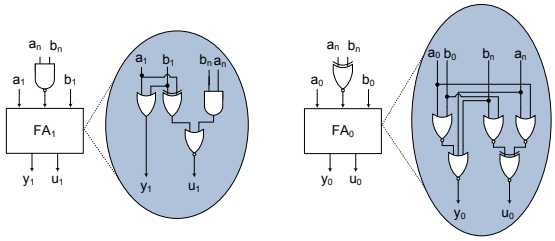

Fig. 2. Simplified circuitry for the two least significant bit positions.

$$\left|X_1 + X_2 + \ldots + X_k\right|_{2^n+1} = \left|\sum_{i=1}^{\lceil \frac{k}{2} \rceil} Y_i + \sum_{i=1}^{\lceil \frac{k}{2} \rceil} U_i + \left\lceil \frac{k}{2} \right\rceil\right|_{2^n+1} \qquad (7)$$

B. The Inverted EAC CSA Tree

In [14], an inverted EAC CSA Dadda tree was proposed for the modulo 2^n+1 reduction of multiple n-bit vectors in two final addends. We also adopt an inverted EAC CSA Dadda tree as our reduction vehicle.

In our case, we need to add in modulo 2^n+1 arithmetic, $2\lceil \frac{k}{2} \rceil + 1$ vectors in total, that is, the diminished vectors and a vector, suppose COR, which represents the total correction. COR includes both the translators correction, $\Big(\left|\lceil \frac{k}{2} \rceil\right|_{2^n+1}$ term of (7)$\Big)$, and the correction introduced by the inverted EAC CSA tree. The use of an inverted EAC CSA tree is based on the observation that the carry output at the most significant bit position, suppose c_n, that has a weight of 2^n, can be complemented and added at the least significant bit position in the next stage, provided that a correction equal to 2^n is taken into account, since it holds that :

$$|c_n 2^n|_{2^n+1} = |-c_n|_{2^n+1} = |2^n + \overline{c_n}|_{2^n+1}\,.$$

Taking into account that each CSA reduces the number of addends by one, for attaining two final addends from the $(2\lceil \frac{k}{2} \rceil + 1)$ vectors of our EAC CSA tree, $(2\lceil \frac{k}{2} \rceil - 1)$ CSAs are required, each one providing an inverted feedback carry. Therefore, the correction required by our EAC CSA tree is $2^n \times \big(2\lceil \frac{k}{2} \rceil - 1\big)$.

Let F and G denote the two final addends produced by the inverted EAC CSA tree. From the above analysis we get that :

$$\left|\sum_{i=1}^{\lceil \frac{k}{2} \rceil} Y_i + \sum_{i=1}^{\lceil \frac{k}{2} \rceil} U_i + COR\right|_{2^n+1} = \left|F + G + 2^n \times \left(2\left\lceil \frac{k}{2} \right\rceil - 1\right)\right|_{2^n+1}$$

or equivalently that :

$$\begin{aligned}&\Big|\sum_{i=1}^{\lceil \frac{k}{2} \rceil} Y_i + \sum_{i=1}^{\lceil \frac{k}{2} \rceil} U_i + \left\lceil \frac{k}{2} \right\rceil\Big|_{2^n+1} = \\ &\qquad = \left|F + G - \left(2\left\lceil \frac{k}{2} \right\rceil - 1\right) - COR + \left\lceil \frac{k}{2} \right\rceil\right|_{2^n+1}\end{aligned}$$

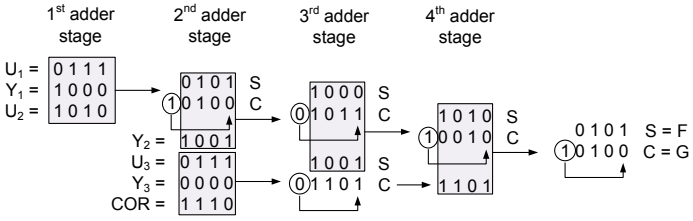

Fig. 3. Inverted EAC CSA tree operation

which according to (7), results into :

$$|X_1 + X_2 + \ldots + X_k|_{2^n+1} = \left|(F + G + 1) - \left\lceil \frac{k}{2} \right\rceil - COR\right|_{2^n+1} \quad (8)$$

C. The Final Stage Adder

If we use as COR the constant $\left(-\left\lceil \frac{k}{2} \right\rceil\right)$, (8) takes the form :

$$|X_1 + X_2 + \ldots + X_k|_{2^n+1} = |F + G + 1|_{2^n+1} \quad (9)$$

that according to (1) reveals that we can derive the n least significant bits of the weighted MOMA$(k, 2^n + 1)$ result, by adding F and G in a diminished-1 parallel adder.

The most significant bit of the weighted MOMA$(k, 2^n + 1)$, should be 1, only when $|X_1 + X_2 + \ldots + X_k|_{2^n+1} = 2^n$, or equivalently when $|F+G+1|_{2^n+1} = 2^n$ or since $0 \leq F,\ G \leq 2^n - 1$ when $F + G = 2^n - 1$, that is, when F and G are bit-wise complementary. This condition can be easily detected as the logical AND of the XOR of the bits of F and G with the same weight. Since in every fast adder architecture there is a preprocessing stage that computes the half-sum term, that is, the XOR of the corresponding input operands bits, the extra hardware required for the most significant bit of the weighted addition is small. Note that this operation will not add any delay on the critical path of the adder. In some adder cases (known as XOR adders) the half sum term is also used as the carry propagate term. The group propagate terms in these adders are the logical AND of the half-sum terms and therefore no extra hardware is required for the derivation of the most significant bit of the weighted MOMA.

D. An Example of the Proposed Weighted MOMA

We exemplify the use of the architecture derived in the previous section, by designing a weighted MOMA(6,17). Let the input operands be denoted by X_1 up to X_6. Let us further consider the input vector for these operands, $X_1 = X_4 = 4_{10}$, $X_2 = 12_{10}$, $X_3 = X_5 = 16_{10}$ and $X_6 = 9_{10}$.

Three translator circuits are used; each accepts a pair of operands and a D vector. For the pairing $(X_1, X_2), (X_3, X_4)$ and (X_5, X_6), the D vectors that are computed are 1111_2, 1110_2 and 1110_2 respectively. The translator circuits then produce the vectors $U_1 = 0111_2$, $Y_1 = 1000_2$, $U_2 = 1010_2$, $Y_2 = 1001_2$, $U_3 = 0111_2$ and $Y_3 = 0000_2$.

The U_i and Y_i vectors along with the correction vector $COR = |-3|_{17} = 14_{10} = 1110_2$ are then used in the inverted EAC CSA tree. We assume that this is designed as a Dadda tree and its operation is indicated in Fig. 3. S and C respectively represent the sum and carry vectors produced by each CSA of the tree. The most significant bit of the carry vector is complemented and used at the least significant bit position in the next addition. These bits are circled in Fig. 3.

The two resulting vectors F and G are then used as inputs to an augmented diminished-1 parallel adder. Since these are not complementary, the most significant bit of the result is 0. The diminished-1 adder is an adder that increments the integer sum of its input vectors when the carry output of their integer addition is 0 and leaves it unchanged otherwise. In our case, since $F = 0101_2$ and $G = 0100_2$, the diminished-1 adder will provide 1010_2 as the n least significant bits of the result.

III. Comparisons

In this section, we compare the proposed MOMA architecture against those proposed in [12], [13]. Both qualitative and quantitative comparison results are presented.

For our qualitative comparisons, we use the simple unit-gate model proposed in [15]. This model assumes that each two-input gate, excluding exclusive-OR, accounts as one equivalent gate for both area and delay. An exclusive-OR gate accounts for two equivalent gates for both area and delay. Finally, a 2 to 1 multiplexer accounts for two equivalent gate delays and has an area complexity of three equivalent gates. Table I summarizes the area and delay estimates that have been derived, using the following assumptions :

i. All binary adders used follow the Kogge-Stone [16] parallel-prefix carry computation architecture.
ii. All diminished adders used follow the parallel-prefix carry computation architecture of [8].
iii. $\theta(a)$ denotes the minimum number of stages in a CSA tree that processes a input operands.

For various test cases, Table II presents the attained unit-gate delay and area estimates in equivalent gates. From the estimates it becomes obvious that the MOMAs designed according to either the proposed architecture or the architecture presented in [12] are far faster and smaller than the corresponding circuits proposed in [13]. This is because the proposal of [13] attacks the design of MOMAs for arbitrary moduli. Therefore, in contrast to the other two architectures, it does not effectively exploit the properties of arithmetic modulo $2^n + 1$. We therefore do not consider this architecture in our quantitative results.

The estimates also indicate that the proposed architecture heavily outperforms the proposal of [12] in operation speed. This is attributed to the fact that the proposed architecture removes the second parallel adder (subtractor) out of the critical path of [12]. The translators added have a delay complexity almost equal to the final selection multiplexors of [12]. On the average of the examined cases, the proposed architecture leads to 32% faster MOMAs. Considering the area complexity, the proposed architecture leads to smaller implementations, when n and k are not both large. For example, the proposed MOMA(4,17) requires 21% less area than the corresponding circuit of [12]. On the other hand, a large value of k implies a large number of translators while a large value of n results in wider translators. When either k or n increases, the area required by the proposed MOMAs also increases and may exceed the savings of the second parallel adder (subtractor) and the multiplexors required by [12].

TABLE I

AREA AND DELAY ESTIMATIONS PROVIDED BY THE UNIT-GATE MODEL

MOMA$(k, 2^n + 1)$		
Architecture	*Delay*	*Area*
[13]	$14\left\lceil\theta(k)+1\right\rceil + 2\log(n+4)$	$35k(n+1) + 3(n+2)\log(n+2) +$ $3(n+3)\log(n+3) + 6(n+4)\log(n+4) - 24n + 52$
[12]	$4\left\lceil\theta(2k+1)-1\right\rceil +$ $2\log n + 2\log(n+1) + 10$	$7k(n+1) + 3n\log n + 3(n+1)\log(n+1) - 2n + 20$
Proposed	$4\theta\left(2\left\lceil\frac{k}{2}\right\rceil+1\right) + 2\log n + 6$	$17\left\lceil\frac{k}{2}\right\rceil n + 6\left\lceil\frac{k}{2}\right\rceil + \frac{9}{2}n\log n - \frac{11}{2}n + 5$

TABLE II

QUALITATIVE COMPARISON RESULTS - EQUIVALENT GATES

MOMA		*[13]*		*[12]*		*Proposed*	
k	n	Delay	Area	Delay	Area	Delay	Area
4	4	48	905	31	211	22	167
8	4	76	1605	39	351	26	315
12	4	90	2305	43	491	30	463
4	8	49	1592	34	414	24	353
8	8	77	2852	42	666	28	637
12	8	91	4112	46	918	32	921
4	16	51	3034	38	864	26	761
8	16	79	5414	46	1340	30	1317
12	16	93	7794	50	1816	34	1873

TABLE III

QUANTITATIVE COMPARISON RESULTS

MOMA		*[12]*		*Proposed*		*Savings (%)*	
k	n	Delay	Area	Delay	Area	Delay	Area
4	4	2.93	10155	2.16	7627	26.3	24.9
8	4	3.58	18856	2.55	15209	28.8	19.3
12	4	4.01	26373	2.86	22974	28.7	12.9
4	8	3.30	19567	2.35	15624	28.8	20.2
8	8	3.94	35647	2.74	30410	30.5	14.7
12	8	4.34	50230	3.08	45998	29.0	8.4
4	16	3.68	39379	2.57	32508	30.2	17.4
8	16	4.36	70444	2.95	62203	32.3	11.7
12	16	4.79	97505	3.26	93321	31.9	4.3

For our quantitative results, an HDL generator was developed for providing HDL descriptions for the MOMAs proposed and those of [12]. After simulating the resulting descriptions, the designs were mapped to a CMOS standard cell library (180nm, 6-metal layer, 1.8 V), assuming typical process parameters. A bottom-up approach was followed during mapping. Once a hierarchy level was mapped and optimized for delay and area, "don't touch" primitives were applied to it, for preserving the architecture in each description as much as possible. The derived results are given in Table III. The delay results are expressed in ns, while those for the implementation area in μm^2. The attained results indicate that on the average of the examined cases, the proposed MOMAs are 30% faster and in parallel require 15% less implementation area.

IV. CONCLUSIONS

In this manuscript we have proposed a new architecture for designing multi-operand modulo 2^n+1 adders that is based on the use of translator circuits that enable to perform weighted operand addition using congruent n-bit additions. The derived architecture leads to the fastest proposed designs, removing a whole parallel adder out of the critical path of the fastest earlier proposal. Our quantitative results indicate that, on the average of the examined cases, savings of 30% in the execution time result with parallel savings in the implementation area, that are significant, when the number of operands is not very large or the operands are not very wide.

REFERENCES

[1] H. Nozaki et al., "Implementation of RSA Algorithm based on RNS Montgomery Multiplication," in *Proc. of the 3^{rd} International Workshop on Cryptographic Hardware and Embedded Systems, Lecture Notes in Computer Science Vol. 2162, Springer-Verlag*, 2001, pp. 364–376.

[2] Y. Ma, "A Simplified Architecture for Modulo (2^n+1) Multiplication," *IEEE Trans. Comput.*, vol. 47, no. 3, pp. 333–337, 1998.

[3] M. A. Soderstrand et al., *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, 1986.

[4] J. Ramirez et al., "RNS-enabled Digital Signal Processor Design," *Electronics Letters*, vol. 38, no. 6, pp. 266–268, 2002.

[5] G. C. Cardarilli, Al. Nannarelli, and M. Re, "Reducing Power Dissipation in FIR Filters using the Residue Number System," in *Proc. of the IEEE 43^{rd} IEEE Midwest Symposium on Circuits and Systems*, 2000, pp. 320–323.

[6] J. Ramirez et al., "Fast RNS FPL-based Communications Receiver Design and Implementation," in *Proc. of the 12^{th} International Conference on Field Programmable Logic, Lecture Notes in Computer Science Vol. 2438, Springer-Verlag*, 2002, pp. 472–481.

[7] L. M. Leibowitz, "A Simplified Binary Arithmetic for the Fermat Number Transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 24, pp. 356–359, 1976.

[8] H. T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-One Modulo 2^n+1 Adder Design," *IEEE Trans. Comput.*, vol. 51, pp. 1389–1399, 2002.

[9] L. Skavantzos, "Design of Multi-operand Carry-Save Adders for Arithmetic Modulo (2^n+1)," *Electronics Letters*, pp. 1152–1153, 1989.

[10] K. E. Elleithy, M. A. Bayoumi, and K. P. Lee, "$\theta(\log n)$ Architectures for rns Arithmetic Decoding," in *Proc. of the 9^{th} IEEE Symposium on Computer Arithmetic*, 1989, pp. 202–209.

[11] C. K. Koc and C. Y. Hung, "Multi-operand Modulo Addition Using Carry-Save Adders," *Electronics Letters*, vol. 26, pp. 361–363, 1990.

[12] S. J. Piestrak, "Design of Residue Generators and Multioperand Modular Adders using Carry-Save Adders," *IEEE Trans. Comput.*, vol. 43, pp. 68–77, 1994.

[13] G. Alia and E. Martineli, "Designing Multioperand Modular Adders," *Electronics Letters*, vol. 32, pp. 22–23, 1996.

[14] C. Efstathiou, H. T. Vergos, G. Dimitrakopoulos, and D. Nikolos, "Efficient Diminished-1 Modulo 2^n+1 Multipliers," *IEEE Trans. Comput.*, vol. 54, no. 4, pp. 491–496, 2005.

[15] A. Tyagi, "A Reduced-Area Scheme for Carry-Select Adders," *IEEE Trans. Comput.*, vol. 42, no. 10, pp. 1163–1170, 1993.

[16] P. M. Kogge and H. S. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Trans. Comput.*, vol. C-22, pp. 786–792, 1973.