Modulo 2^n+1 Arithmetic Units with Embedded Diminished-to-Normal Conversion

Evangelos Vassalos, Dimitris Bakalis
Electronics Laboratory, Dept. of Physics
University of Patras
Patras, Greece
vassalos@upatras.gr, bakalis@physics.upatras.gr

Haridimos T. Vergos
Dept. of Computer Engineering & Informatics
University of Patras
Patras, Greece
vergos@ceid.upatras.gr

***Abstract*—The diminished-one representation has been proposed for RNS-based systems with moduli of the 2^n+1 forms as an encoding that is more efficient than the normal representation in the arithmetic processing units. However, its use necessitates a two-step reverse conversion, in which a diminished-to-normal conversion is first performed before the final residue-to-binary conversion resulting in performance loss. In this paper we introduce efficient modulo 2^n+1 adders, subtractors and multipliers that accept diminished-one operands at their inputs and derive normal operands at their outputs, that is, we embed the diminished-to-normal conversion within the arithmetic processing. Experimental results show that the proposed one-step approach is more efficient in terms of delay.**

Keywords-residue number system; modulo arithmetic; modulo 2^n+1 arithmetic units; diminished-one representation; normal representation

I. INTRODUCTION

The Residue Number System (RNS) [1] [2] is an alternative number representation commonly adopted for speeding up computations in digital signal processing [3-6], cryptography [7-9] and telecommunication applications [10-12]. A non-positional RNS is defined by a set of L moduli, suppose $\{m_1, \ldots, m_L\}$ that are pair-wise relatively prime. An integer A has a unique representation in the RNS, given by the set $\{a_1, \ldots, a_L\}$ of residues, where $a_i = |A|_{m_i}$ is the modulo m_i residue of A. An operation $\otimes$ over an RNS is defined as $(z_1, \ldots, z_L) = (a_1, \ldots, a_L) \otimes (b_1, \ldots, b_L)$, where $z_i = |a_i \otimes b_i|_{m_i}$. The computation of z_i only depends on a_i, b_i, and m_i implying that all z_is can be computed in parallel, each in a separate arithmetic unit often called a channel. Since all channels operate in parallel and each deals with narrow residues instead of wide numbers, significant speedup over the usual binary representation may be achieved.

An RNS-based system consists of three main blocks. At first, all binary inputs are converted to their corresponding sets of residues with binary-to-residue (forward) converters, according to the specified moduli set. Then, the arithmetic processing is performed in parallel in each modulo channel. The arithmetic computations that are usually performed consist of several additions, subtractions and multiplications, which can be efficiently realized in modulo arithmetic. Finally, the RNS representation of the results is converted back to binary with residue-to-binary (reverse) converters.

RNS channels with moduli of the 2^n, 2^n-1 or 2^n+1 forms have received significant attention. This is because the arithmetic circuits that have been proposed for the $2^n\pm1$ moduli are almost as efficient as the binary ones for the same operand widths [13-27]. Furthermore, efficient converters exist between the residue and the binary representation for various moduli sets [26] [28-38]. In such moduli sets, the 2^n+1 channel has to deal with operands one bit wider than the corresponding 2^n-1 or 2^n channels, leading to a performance bottleneck. To avoid this, and given that in the case of a zero operand the result can be derived straightforwardly, [39] introduced the diminished-one representation. In the diminished-one representation each operand is represented decreased by one compared to its normal representation and hence only n bits are required for its representation. A separate bit is utilized to indicate a zero operand or result. The diminished-one representation has the advantage that it allows to better equalize the delay of the modulo 2^n+1 channel with the delay of the remaining channels of the moduli set. During the last years, efficient architectures for diminished-one modulo 2^n+1 addition, subtraction and multiplication have been presented in the open literature, that are more efficient than those for the normal representation [14] [15] [19] [21] [22] [24] [26] [27]. To this end, the diminished-one representation is preferred for modulo channels of the 2^n+1 forms in RNS-based systems.

Binary-to-residue conversion for the diminished-one representation is equally efficient as the conversion for the normal representation [26] [40]. However, almost all residue-to-binary converters that have been reported until now assume a normal representation for the residues of the 2^n+1 forms. The only exception is the reverse converter that is reported in [26] for the $\{2^n\text{-}1, 2^{n+k}, 2^n\text{+}1\}$ moduli set. Hence, if the diminished-one representation is adopted in order to speed up the arithmetic processing, then a two-step approach is required in the reverse conversion; a diminished-to-normal converter has to be used before the final residue-to-binary conversion. The diminished-to-normal converter can be based on a binary incrementer [15] but its logarithmic delay [41] can cancel all the speedup achieved in the arithmetic processing.

In this paper we explore the embedding of the diminished-to-normal conversion within the arithmetic

978-0-7695-4494-6/11 $26.00 © 2011 IEEE
DOI 10.1109/DSD.2011.66

processing units. To this end, we present novel architectures for designing modulo 2^n+1 arithmetic units (adders, subtractors and multipliers), that assume the diminished-one representation at the inputs and the normal representation at the outputs. The proposed architectures are based on those of the corresponding diminished-one arithmetic units. They can be efficiently utilized in RNS-based systems with moduli of the 2^n+1 forms where the arithmetic processing is based on the diminished-one representation while the reverse conversion assumes the normal representation.

The remaining of the paper is organized as follows. The next two sections present novel modulo 2^n+1 adders and subtractors while section IV presents novel modulo 2^n+1 multipliers. Section V presents experimental results and comparisons. The last section concludes the paper.

II. Modulo 2^n+1 Addition

Suppose that A and B denote two modulo 2^n+1 operands, that is, $0 \le A, B < 2^n + 1$. Let (a_z, A^*) and (b_z, B^*) denote the diminished-one representations of A and B, respectively, where a_z and b_z are the zero indication bits of the two operands that are equal to 1 only when A or B is equal to 0 and $A^*=a_{n-1}\ldots a_0$ and $B^*=b_{n-1}\ldots b_0$ are n-bit wide vectors which are defined according to the following equations [15]:

$$A^* = \begin{cases} A-1, A \neq 0 \\ 0, \quad A = 0 \end{cases}, \qquad B^* = \begin{cases} B-1, B \neq 0 \\ 0, \quad B = 0 \end{cases}$$

Given the diminished-one representations of A and B, the sum of the two operands taken modulo 2^n+1 in normal representation, according to the values of A and B, is summarized in Table I.

Let us consider the following cases:

- $A \neq 0$ and $B \neq 0$ (equivalently $a_z = 0$ and $b_z = 0$)

It holds that

$$\left|A+B\right|_{2^n+1} = \left|A^*+B^*+2\right|_{2^n+1} = \left\|A^*+B^*+1\right|_{2^n+1}+1\Big|_{2^n+1}.$$

It is well known that a diminished-one adder produces at the output the n least significant bits of the sum of its input operands taken modulo 2^n+1 increased by one [21]. The most significant bit of the abovementioned arithmetic operation can be derived by checking whether the two input operands are bitwise complementary [40]. Moreover, an n-bit Carry Save Adder (CSA) with Inverted End-Around Carry (IEAC), composed of n Full Adders (FAs) and an inverter, produces at its output, in carry and sum format, the sum of its input operands taken modulo 2^n+1 increased by one. Hence, the first increment in the modulo addition operation ($\left|A^*+B^*+1\right|_{2^n+1}$) can be realized by a CSA with IEAC with inputs A^*, B^* and 0, while the second can be realized by an enhanced diminished-one adder [40], as shown in Fig. 1 ($r_n\ldots r_0$ denote the n+1 bits of the result). Obviously, the n FAs of the CSA are simplified to Half Adders (HAs).

TABLE I. Modulo 2^n+1 Addition

a_z	b_z	$\left\|A+B\right\|_{2^n+1}$
0	0	$\left\|A^*+B^*+2\right\|_{2^n+1}$
0	1	$\left\|A^*+B^*+1\right\|_{2^n+1}$
1	0	$\left\|A^*+B^*+1\right\|_{2^n+1}$
1	1	$\left\|A^*+B^*\right\|_{2^n+1}$

- $A \neq 0$ and $B = 0$ or $A = 0$ and $B \neq 0$

In this case, only a single increment is required. We can still use the architecture of Fig. 1 as long as we inhibit the increment performed at the IEAC CSA. When A or B or both are equal to 0, the carry output of the most significant HA of the CSA is equal to 0 and thus its inverted value that is equal to 1 is driven to the least significant bit position (Y_0 in Fig. 1) in the enhanced diminished-one adder. We can change this bit value to 0 by a 2-input NOR gate driven by the carry out of the HA and by $a_z \vee b_z$ ($\vee$ denotes a logic OR). The output of the NOR gate is driven to the least significant bit input of the enhanced diminished-one adder.

- $A = 0$ and $B = 0$ (equivalently $a_z = 1$ and $b_z = 1$)

In this case, both increments must be inhibited. The modification described for the previous case inhibits the increment performed by the IEAC CSA in this case too, while the second inhibition can be performed by inverting the half-sum signal (h_0) at the least significant bit position within the enhanced diminished-one adder, when a_z=b_z=1.

The complete architecture for the proposed modulo 2^n+1 adder is shown in Fig. 2. Note that none of the modifications made to the architecture of Fig. 1, in order to cover the last two cases, reside on the critical path of the circuit.

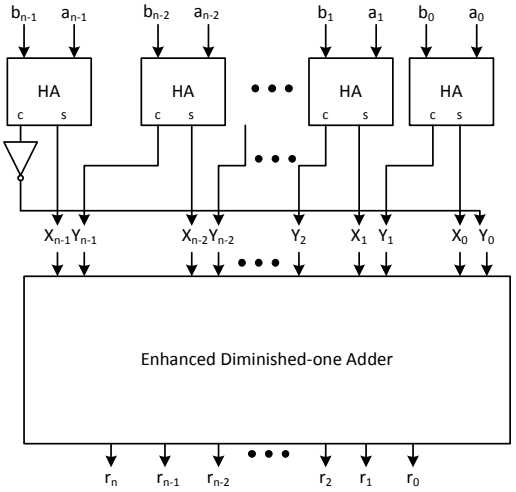

Figure 1. Modulo 2^n+1 addition.

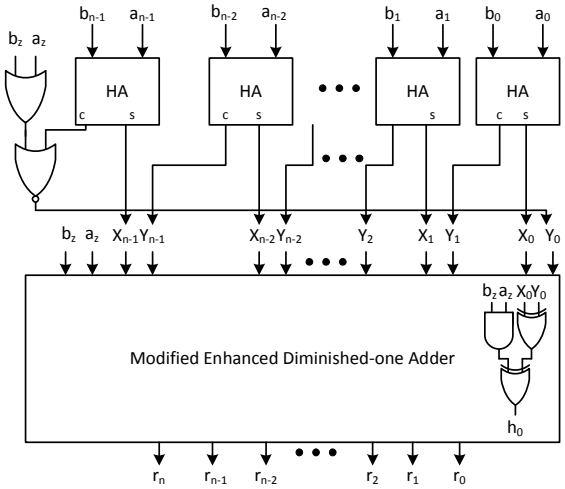

Figure 2. Proposed modulo 2^n+1 adder architecture.

III. MODULO 2^n+1 SUBTRACTION

Let us now consider the modulo 2^n+1 subtraction. Given the diminished-one representations of two modulo 2^n+1 operands A and B, (a_z, A^*) and (b_z, B^*), the difference of A and B taken modulo 2^n+1 in normal representation, $|A-B|_{2^n+1}$, can be derived as follows. We distinguish the following four cases:

- $A \neq 0$ and $B \neq 0$
  It holds that

$$|A-B|_{2^n+1} = |(A^*+1)-(B^*+1)|_{2^n+1} = |A^*-B^*|_{2^n+1}$$
$$= |A^*+(2^n-1)-B^*+2|_{2^n+1} = |A^*+\overline{B^*}+2|_{2^n+1}$$

where $\overline{B^*}$ denotes the bitwise complement of B^*.

- $A \neq 0$ and $B = 0$
  Since B^*=0, $-B^* = +B^*$ and therefore it holds that

$$|A-B|_{2^n+1} = |(A^*+1)-B^*)|_{2^n+1} = |A^*+B^*+1|_{2^n+1}$$

- $A = 0$ and $B \neq 0$
  It holds that

$$|A-B|_{2^n+1} = |A^*-(B^*+1)|_{2^n+1} = |A^*-B^*-1|_{2^n+1}$$
$$= |A^*+(2^n-1)-B^*+1|_{2^n+1} = |A^*+\overline{B^*}+1|_{2^n+1}$$

- $A = 0$ and $B = 0$
  Since B^*=0, $-B^* = +B^*$ and therefore it holds that

$$|A-B|_{2^n+1} = |A^*-B^*|_{2^n+1} = |A^*+B^*|_{2^n+1}$$

TABLE II. MODULO 2^n+1 SUBTRACTION

A_z	B_z	$\lvert A-B\rvert_{2^n+1}$
0	0	$\lvert A^*+\overline{B^*}+2\rvert_{2^n+1}$
0	1	$\lvert A^*+B^*+1\rvert_{2^n+1}$
1	0	$\lvert A^*+\overline{B^*}+1\rvert_{2^n+1}$
1	1	$\lvert A^*+B^*\rvert_{2^n+1}$

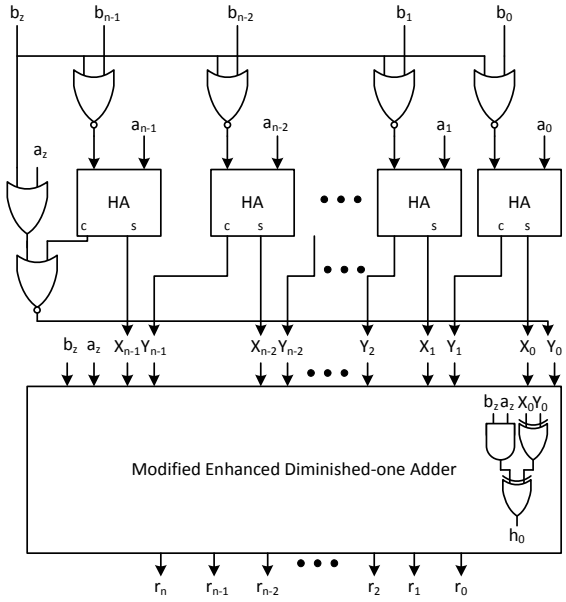

Figure 3. Proposed modulo 2^n+1 subtractor architecture.

All four above cases are summarized in Table II. It is obvious that the modulo 2^n+1 subtraction can be realized by the modulo 2^n+1 adder presented in the previous section, provided that we complement the bits of B^* when $B \neq 0$ and leave the bits of B^* unaltered when $B = 0$. The proposed architecture for modulo 2^n+1 subtraction is shown in Fig. 3.

IV. MODULO 2^n+1 MULTIPLICATION

Almost all architectures that have been presented, during the last few years, for modulo 2^n+1 multiplication [24-27], consist of three stages. At first, all partial products (*PPs*) and correction terms (*CTs*) are formed. These are actually *n*-bit wide vectors. Then, a multi-operand adder (usually an adder tree) composed of CSAs with IEAC is used to compress all *PP*s and *CT*s to two *n*-bit wide vectors that are finally added with a fast two-operand modulo 2^n+1 adder in order to produce the result of the multiplication.

Consider two modulo 2^n+1 operands A and B and their corresponding diminished-one representations (a_z, A^*) and (b_z, B^*). Assume at first that both A and B are not equal to 0. A diminished-one modulo 2^n+1 multiplier will produce at its

output the diminished-one representation of the product of A and B, that is,

$$\left|A\times B-1\right|_{2^n+1}=\left|(A^*+1)\times(B^*+1)-1\right|_{2^n+1}$$
$$=\left|A^*\times B^*+A^*+B^*\right|_{2^n+1}$$

In order to get the product in normal representation we have to compute

$$\left|A\times B\right|_{2^n+1}=\left|(A^*+1)\times(B^*+1)\right|_{2^n+1}$$
$$=\left|A^*\times B^*+A^*+B^*+1\right|_{2^n+1}$$

Hence, we can utilize any diminished-one architecture for modulo 2^n+1 multiplication and we can get the normal representation of the result by: (a) increasing by one the CT of the diminished-one multiplier (which in several cases is constant), and (b) replacing the fast diminished-one two-operand adder that produces the n-bit wide result with an enhanced diminished-one two-operand adder as the one presented in [40] that produces all n+1 bits of the result. The proposed architecture is shown in Fig. 4.

We also have to take into account the cases where A or B (or both) are equal to 0. In these cases, A^* = 0…0 or B^* = 0…0 (or both), but the architecture of Fig. 4 will produce a result which is not equal to 0 since it assumes that a diminished-one value of 0 at its inputs represents the normal value of 1. In order to get the correct result, we can simply force the n+1 outputs in these cases to logic 0 by a series of 2-input AND gates driven by the logic NOR of the a_z and b_z zero indication bits. The complete architecture is then given in Fig. 5.

The proposed architecture can be based on any architecture for diminished-one modulo 2^n+1 multiplication. In the following we present architectures that can be derived based on the diminished-one modulo multipliers presented in [27] and [24], which are considered the most efficient non-recoded and Booth-recoded architectures in the open literature, respectively.

The architecture of [27] derives a partial product matrix with n n-bit wide partial products and an additional n-bit correction term (CT) equal to 0…0($\overline{a_1 b_{n-1}}$)($a_1 b_{n-1}$). The partial products and the correction term are then added with an adder tree composed of CSAs with IEAC and finally a fast diminished-one two-operand adder is used to derive the n-bits of the result. We have to note that zero handling is not taken into consideration in the multipliers of [27]. According to Fig. 5, we can use the architecture of [27] for deriving the result of the multiplication in normal representation as long as we: (a) replace the fast diminished-one adder with the enhanced one [40] in order to get n+1 bits at the output, (b) add the 2-input AND gates and the 2-input NOR gate at the output for handling the zero operands and (c) increase the correction term by one. This is easily achieved by replacing the CT vector of [27] with the correction term 0…01($\overline{a_1 b_{n-1}}$).

Consider now the architecture of [24] which derives Booth-recoded diminished-one multipliers and assume that n is even. The architecture derives n/2+2 partial products, each n-bits wide. n/2 of them are attributed to the Booth-recoding, one is a correction term that is input dependent and the final partial product is a constant correction term with a value equal to 1. Similarly to the previous case, we can use the architecture of [24] for deriving the proposed modulo multiplier. What we have to do is to use the enhanced diminished-one adder and the AND gates at the output while also increasing the CT by one.

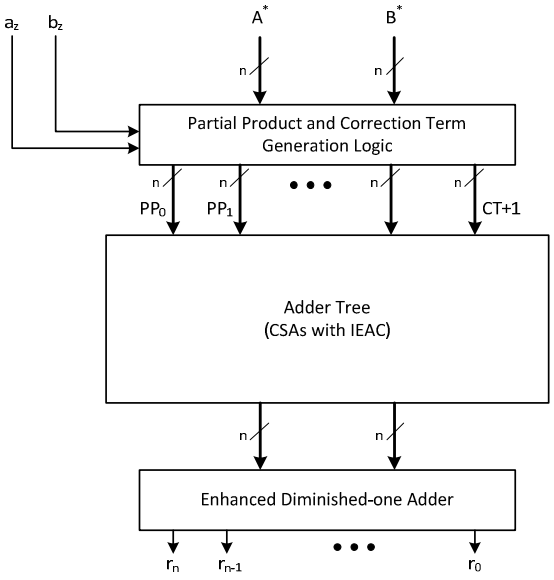

Figure 4. Modulo 2^n+1 multiplication.

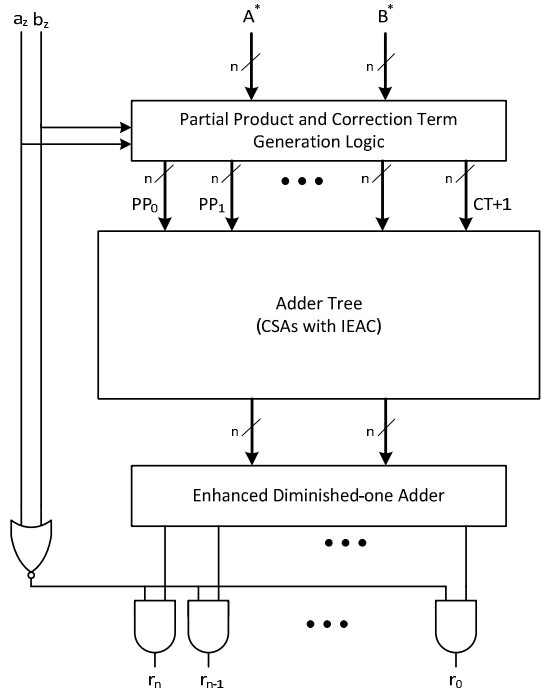

Figure 5. Proposed modulo 2^n+1 multiplier architecture.

TABLE III. UNIT-GATE DELAY AND AREA ESTIMATIONS

	Diminished-one with zero handling	*Diminished-one + Converter*	*Proposed*
		Delay	
Adders	$2\log n + 6$	$3\log n + 9$	$2\log n + 5$
Subtractors	$2\log n + 7$	$3\log n + 10$	$2\log n + 6$
Multipliers	$4D(n/2+2) + 2\log n + 9$	$4D(n/2+2) + 3\log n + 12$	$4D(n/2+2) + 2\log n + 10$
		Area	
Adders	$3n\log n + 3n + 6$	$(7/2)n\log n + 5n + 8$	$9/2n\log n + 7/2n + 9$
Subtractors	$3n\log n + 4n + 6$	$(7/2)n\log n + 6n + 8$	$9/2n\log n + 9/2n + 9$
Multipliers	$8n^2 + (9/2)n\log n + (29/2)n + 6$	$8n^2 + 5n\log n + (39/2)n + 8$	$8n^2 + 9/2n\log n + (37/2)n + 8$

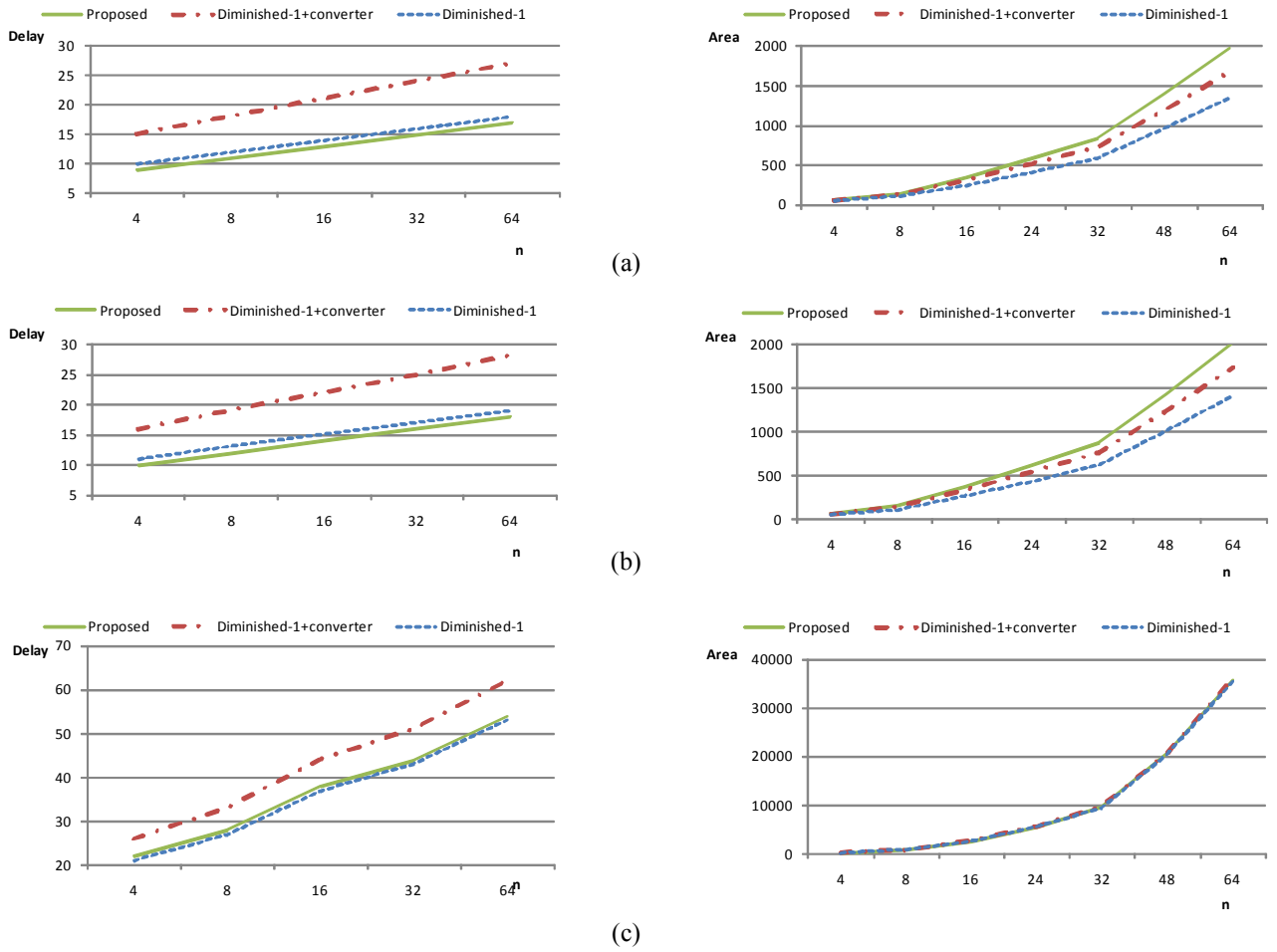

Figure 6. Unit-gate delay and area comparison of (a) adders, (b) subtractors, and (c) multipliers.

V. EVALUATION AND COMPARISONS

In this section we evaluate the proposed arithmetic units and compare them against the corresponding units that use a diminished-one arithmetic unit with zero handling along with a diminished-to-normal converter at the output. For the diminished-one arithmetic units we assume the architectures proposed in [15] [19] and [24], which are considered to be the current state of the art. For the diminished-to-normal converter we assume a controlled binary incrementer with a Sklansky parallel-prefix structure [15].

An area-delay comparison of the various converters can be based on the unit gate model [42]. The unit gate model assumes that each monotonic gate counts as one gate equivalent for both area and delay, while two-input XOR and XNOR gates count for two gate equivalents for both area and delay.

TABLE IV. CMOS VLSI DELAY AND AREA RESULTS

	Diminished-one + Converter		*Proposed*	
n	*Delay (ns)*	*Area (μm^2)*	*Delay (ns)*	*Area (μm^2)*
		Adders		
4	0.809	762	0.555	707
8	1.145	2022	0.693	1827
16	1.568	5302	0.838	4677
32	2.064	13260	1.042	11608
		Subtractors		
4	0.867	793	0.557	746
8	1.222	2085	0.797	1889
16	1.633	6145	0.934	4897
32	2.145	14673	1.125	11869
		Multipliers		
4	1.285	2207	0.967	2093
8	1.962	6232	1.418	5927
16	2.848	19083	1.883	18911
32	3.490	64288	2.346	62648

Table III presents area and delay estimations of the proposed arithmetic units. The proposed adders and multipliers are based on the enhanced diminished-one adders presented in [40] while the proposed multipliers are based on the Booth-recoded multipliers presented in [24]. The *D(k)* function in the multipliers' case denotes the number of levels that are required in a Dadda adder tree for reducing *k* partial products to two. Table III also presents the area and delay estimates of the corresponding diminished-one arithmetic units, according to [15] [19] and [24], as well as the estimates of the above mentioned diminished-one arithmetic units along with the diminished-to-normal converter. For the diminished-to-normal converter, a delay equal to $\log n+3$ equivalent gates and an area equal to $(1/2)n\log n+2n+2$ equivalent gates are assumed.

Fig. 6 graphically compares the unit-gate area and delay of the various arithmetic units for values of *n* up to 64. We observe that the delay of the proposed circuits is significantly smaller than that of the diminished-one circuits with the converter while it is very close to the delay of the diminished-one arithmetic units. Regarding the area, the proposed adders and subtractors are slightly larger than the other two circuits under comparison while all multipliers have approximately the same area.

The unit-gate model estimations for area and delay can only be considered as indicative. To attain realistic results, arithmetic units for 4 values of *n* were described in HDL. After simulating the resulting descriptions, the circuits were synthesized and mapped to a 90 nm CMOS implementation technology [43]. The Synopsys Design Compiler tool [44] in the topographical mode was used for the synthesis and mapping of the circuits. In this mode, for achieving faster timing closure, the tool performs floorplanning in parallel with synthesis and mapping and the design is annotated with wiring lengths and fan-out and parasitic capacitances coming directly from the floorplan of the design and not from a wire load model. We assumed that each circuit's input and output is driven by the output of a D flip flop and drives the input of a D flip flop of the same implementation library, respectively. A typical corner (1.2V, 25^{o}C) was considered. Each circuit was recursively optimized for speed using a bottom-up approach. A final area recovery step was then applied. Table IV presents the attained area and delay results. The results validate that the proposed circuits are significantly faster than those where the diminished-to-binary conversion is performed separately at the output of the diminished-one arithmetic unit.

VI. CONCLUSIONS

Novel modulo 2^n+1 adders, subtractors and multipliers have been presented in this paper. The proposed circuits, apart from their arithmetic operation they also perform representation conversion, since they accept diminished-one operands at their inputs and produce their results in the normal representation, avoiding that way the need for a two-step reverse conversion process. The proposed arithmetic units can be efficiently utilized in RNS-based systems with moduli of the 2^n+1 forms where the arithmetic processing is based on the diminished-one representation while the residue-to-binary conversion assumes the normal representation.

ACKNOWLEDGEMENT

This work was supported by the Caratheodory Programme of the University of Patras (D.178).

REFERENCES

[1] P. V. Ananda Mohan, Residue Number Systems: Algorithms and Architectures, Netherlands: Kluwer Academic Publishers, 2002.

[2] A. Omondi and B. Premkumar, Residue Number Systems: Theory and Implementation, London: Imperial College Press, 2007.

[3] R. Chaves and L. Sousa, "RDSP: A RISC DSP based on Residue Number System," In Proceedings of 6th Euromicro Symposium on Digital System Design, 2003, pp. 128–135, doi:10.1109/DSD.2003.1231911.

[4] P. G. Fernandez and A. Lloris, "RNS-based implementation of 8x8 point 2D-DCT over field-programmable devices," Electronics Letters, vol. 39, no. 1, Jan. 2003, pp. 21–23, doi: 10.1049/el:20030084.

[5] Y. Liu and E. Lai, "Moduli set selection and cost estimation for RNS-based FIR filter and filter bank design," Design Automation for Embedded Systems, vol. 9, no. 2, 2004, pp. 123–139, doi:10.1007/s10617-005-1186-4.

[6] G. Cardarilli, A. Nannarelli and M. Re, "Residue number system for low-power DSP applications," In Proceedings of Asilomar Conference on Signals, Systems and Computers, Apr. 2007, pp. 1412–1416, doi: 10.1109/ACSSC.2007.4487461.

[7] J. -C. Bajard and L. Imbert, "A full RNS implementation of RSA," IEEE Transactions on Computers, vol. 53, no. 6, Jun. 2004, pp. 769–774, doi: 10.1109/TC.2004.2.

[8] J. -C. Bajard, S. Duquesne, M. Ercegovac, and N. Meloni, "Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography," In Proceedings of of SPIE, Aug. 2006, doi:10.1117/12.679541.

[9] D. Schinianakis, A. Fournaris, H. Michail, A. Kakarountas, and T. Stouraitis, "An RNS implementation of an F_p Elliptic Curve Point Multiplier," IEEE Transactions on Circuits and Systems I, vol. 56, no. 6, Oct. 2009, pp. 1202–1213, doi: 10.1109/TCSI.2008.2008507.

[10] U. Meyer-Baese, A. Garcia and F. Taylor, "Implementation of a communications channelizer using FPGAs and RNS arithmetic," VLSI Signal Processing, vol. 28, no. 1–2, May 2001, pp. 115–128, doi: 10.1023/A:1008167323437.

[11] M. Panella, and G. Martinelli, "An RNS architecture for quasi-chaotic oscillators," Journal of VLSI Signal Processing, vol. 33, no. 1–2, Jan. 2003, pp. 199–220, doi: 10.1023/A:1021162422734.

[12] A. S. Madhukumar and F. Chin, "Enhanced architecture for Residue Number System-based CDMA for high-rate data transmission," IEEE Transactions on Wireless Communications, vol. 3, no. 5, Sep. 2004, pp. 1363–1368, doi: 10.1109/TWC.2004.833509.

[13] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, and J. Kalamatianos, "High-Speed Parallel-Prefix Modulo 2^n-1 Adders," IEEE Transactions on Computers, vol. 49, no. 7, Jul. 2000, pp. 673–680, doi: 10.1109/12.863036.

[14] H. T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-One Modulo 2^n+1 Adder Design," IEEE Transactions on Computers, vol. 51, no. 12, Dec. 2002, pp. 1389–1399, doi: 10.1109/TC.2002.1146705.

[15] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Handling Zero in Diminished-one Modulo 2^n+1 Adders," International Journal of Electronics, vol. 90, no. 2, 2003, pp. 133–144, doi: 10.1080/0020720310001482 6.

[16] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Fast Parallel-Prefix Modulo 2^n+1 Adders," IEEE Transactions on Computers, vol. 53, no. 9, Sep. 2004, pp. 1211–1216, doi: 10.1109/TC.2004.60.

[17] R. Patel, M. Benaissa, and S. Boussakta, "Fast Parallel-Prefix Architectures for Modulo 2^n-1 Addition with a Single Representation of Zero," IEEE Transactions on Computers, vol. 56, no. 11, Nov. 2007, pp. 1484–1493, doi: 10.1109/TC.2007.70750.

[18] H. T. Vergos, and C. Efstathiou, "A Unifying Approach for Weighted and Diminished-1 Modulo 2^n+1 Addition," IEEE Transactions on Circuits and Systems II, vol. 55, no. 10, Oct. 2008, pp. 1041–1046, doi: 10.1109/TCSII.2008.2001964.

[19] E. Vassalos, D. Bakalis, and H. T. Vergos, "Novel Modulo 2^n+1 Subtractors," In Proceedings of Conference on Digital Signal Processing, Jul. 2009, doi: 10.1109/ICDSP.2009.5201052z.

[20] C. Efstathiou, "Efficient Modulo 2^n+1 Subtractors for Weighted Operands," In Proceedings of International Conference on Electronic Circuits and Systems, Dec. 2010, doi: 10.1109/ICECS.2010.5724439.

[21] Z. Wang, G. A. Jullien, and W. C. Miller, "An Efficient Tree Architecture for Modulo 2^n+1 Multiplication," Journal of VLSI Signal Processing, vol. 14, no. 3, 1996, pp. 241–248, doi: 10.1007/BF00929618.

[22] Y. Ma, "A simplified architecture for modulo (2^n+1) multiplication," IEEE Transactions on Computers, vol. 47, no. 3, Mar. 1998, pp. 333–337, doi: 10.1109/12.660169.

[23] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Modified Booth Modulo 2^n-1 Multipliers," IEEE Transactions on Computers, vol. 53, no. 3, Mar. 2004, pp. 370–374, doi: 10.1109/TC.2004.1261842.

[24] L. Sousa, and R. Chaves, "A Universal Architecture for Designing Efficient Modulo 2^n+1 Multipliers," IEEE Transactions on Circuits and Systems I, vol. 52, no. 6, Jun. 2005, pp. 1166–1178, doi: 10.1109/TCSI.2005.849143.

[25] H. T. Vergos, and C. Efstathiou, "Design of Efficient Modulo 2^n+1 Multipliers," IET Computers and Digital Techniques, vol. 1, no. 1, Jan. 2007, pp. 49–57, doi: 10.1049/iet-cdt:20060026.

[26] R. Chaves, and L. Sousa, "Improving Residue Number System Multiplication with more Balanced Moduli Sets and Enhanced Modular Arithmetic Structures," IET Computers and Digital Techniques, vol. 1, no. 5, Sep. 2007, pp. 472–280, doi: 10.1049/iet-cdt:20060059.

[27] C. Efstathiou, I. Voyiatzis, and N. Sklavos, "On the Modulo 2^n+1 Multiplication for Diminished-1 Operands," In Proceedings of Conference Signals, Circuits and Systems, Nov. 2008, doi: 10.1109/ICSCS.2008.4746907.

[28] Y. Wang, "Residue-to-binary converters based on new Chinese Remainder Theorems," IEEE Transactions on Circuits and Systems II, vol. 47, no. 3, Mar. 2000, pp. 197–205, doi: 10.1109/82.826745.

[29] Z. Wang, G. A. Jullien, and W. C. Miller, "An improved residue-to-binary converter," IEEE Transactions on Circuits and Systems I, vol. 47, no. 9, 2000, pp. 1437–1440, doi: 10.1109/81.883343.

[30] N. Burgess, "Efficient RNS-to-binary conversion using high-radix SRT division," IEE Proceedings – Computers and Digital Techniques, vol. 148, no. 1, 2001, pp. 49–52, doi: 10.1049/ip-cdt:20010202.

[31] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder based residue to binary number converters for (2^n-1, 2^n, 2^n+1)," IEEE Transactions on Signal Processing, vol. 50, no. 7, Jul. 2002, pp. 1772–1779, doi: 10.1109/TSP.2002.1011216.

[32] B. Cao, C. –H. Chang, and T. Srikanthan, "An efficient reverse converter for the 4-moduli set {2^n-1, 2^n, 2^n+1, 2^{2n}+1} based on the new Chinese Remainder Theorem," IEEE Transactions on Circuits and Systems I, vol. 50, no. 10, Oct. 2003, pp. 1296–1303, doi:10.1109/TCSI.2003.817789.

[33] A. Omondi, "Fast residue-to-binary conversion using base extension and the Chinese Remainder Theorem," Journal of Circuits, Systems and Computers, vol. 16, no. 3, Jun. 2007, pp. 379–388, doi: 10.1142/S021812660700371X.

[34] B. Cao, C. –H. Chang, and T. Srikanthan, "A residue-to-binary converter for a new five-moduli set," IEEE Transactions on Circuits and Systems I, vol. 54, no. 5, May 2007, pp. 1041–1049, doi: 10.1109/TCSI.2007.890623.

[35] P. V. Ananda Mohan, and A. Premkumar, "RNS-to-binary converters for two four-moduli sets {2^n-1, 2^n, 2^n+1, 2^{n+1}-1} and {2^n-1, 2^n, 2^n+1, 2^{n+1}+1}," IEEE Transactions on Circuits and Systems I, vol. 54, no. 6, Jun. 2007, pp. 1245–1254, doi: 10.1109/TCSI.2007.895515.

[36] Y. Kuo, S. Lin, M. Sheu, J. Wu, and P. Wang, "Efficient VLSI design of a reverse RNS converter for new flexible 4-moduli set (2^{p+k}, 2^p+1, 2^p-1, 2^{2p}+1)," In Proceedings of International Symposium on Circuits

and Systems, May 2009, pp. 437–440, doi: 10.1109/ISCAS.2009.5117779.

[37] A. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, and S. Timarchi, "Efficient reverse converter designs for the new 4-moduli sets $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n+1}\text{-}1\}$ and $\{2^n\text{-}1, 2^n\text{+}1, 2^{2n}, 2^{2n}\text{+}1\}$ based on New CRTs," IEEE Transactions on Circuits and Systems I, vol. 57, no. 4, Apr. 2010, pp. 823–835, doi:10.1109/TCSI.2009.2026681.

[38] K. Gbolagade, R. Chaves, L. Sousa, and S. Cotofana, "An improved RNS reverse converter for the $\{2^{2n+1}\text{-}1, 2^n, 2^n\text{-}1\}$ moduli set," In Proceedings of International Symposium on Circuits and Systems, May 2010, pp. 2103–2106, doi:10.1109/ISCAS.2010.5537062.

[39] L. Leibowitz, "A simplified binary arithmetic for the fermat number transform," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 24, no. 5, Oct. 1976, pp. 356–359, doi: 10.1109/TASSP.1976.1162834.

[40] H. T. Vergos, and D. Bakalis, "On Implementing Efficient Modulo 2^n+1 Arithmetic Components," Journal of Circuits, Systems and Computers, vol. 19, no. 5, Aug. 2010, pp. 911–930, doi:10.1142/S0218126610006529.

[41] H. Parandeh-Afshar, A. Afzali-Kusha, and A. Khakifirouz, "A Very Fast and Low Power Pseudo-Incrementer for Address Bus Encoder/Decoder," In Proceedings of International Conference on Microelectronics, Dec. 2006, pp. 91–94.

[42] A. Tyagi, "A reduced-area scheme for carry-select adders," IEEE Transactions on Computers, vol. 42, no. 10, Oct. 1993, pp. 1163–1170, doi: 10.1109/ICCD.1990.130219.

[43] Synopsys Inc, SAED 90nm EDK, 2011, https://www.synopsys.com/apps/protected/university/members.html.

[44] Synopsys Inc, Synopsys Design Compiler User Guide, Version E-2010.12, 2010.