An improved RNS generator $2^n \pm k$ based on threshold logic

Hector Pettenghi, Ricardo Chaves and Leonel Sousa
IST/INESC-ID
R. Alves Redol, 9, 1000-029 Lisboa, Portugal
{hector,rjfc,las}@sips.inesc-id.pt

María J. Avedillo
IMSE/CNM, CSIC
Av. Américo Vespucio s/n, 41092 Sevilla, Spain
avedillo@imse.cnm.es

***Abstract*— This paper presents a new scheme for designing residue generators using threshold logic. This approach is based on the periodicity of the series of powers of 2 taken modulo $2^n \pm k$. In addition, a new algorithm is proposed to obtain a new set of partitions which are more advantageous in terms of area and delay for the presented topology. Experimental results in the analized range of *k* and *n* show that new proposed circuits using the novel partitioning are 70% faster and provide area savings of 64%, when compared with similar circuits using the partitioning methods presented to date.**

Keywords- generator modulo A, residue number systems, binary-to residue number systems converter, threshold logic.

I. INTRODUCTION

Residue arithmetic has been used in digital computing systems for many years [1]. The modular characteristic of the Residue Number System (*RNS*) offers the potential for high-speed and parallel arithmetic. *RNS* is a carry-free arithmetic system that improves the computation performance, and exhibits more parallelism and smaller area [1-2]. Arithmetic operations, e.g. addition, substraction, and multiplication can be carried out on residue digits independently and concurrently more efficiently than the conventional two's complement systems [1]. *RNS* has shown significant efficiency in implementing different types of Digital Signal Processing (*DSP*) applications, such as filtering [3], FFT computation [4], fault-tolerant computer systems [5], communication [6], and cryptography [7].

Whatever the application, *RNS* uses a number of residue generators which forms binary-to-residue converters [1,8-9]. However, residue converters can compromise the speed of the whole system. The most well-known *RNS* moduli set is formed by three-moduli *RNS* $\{2^n-k, 2^n, 2^n+k\}$ with $k=1$. This moduli set has attracted attention since it is well suited for effective regular *VLSI* implementations [10] and very efficient combinational converters from/to the binary system exist [11]. Different moduli sets have been proposed for the direct/reverse conversion to increase the parallelism and the dynamic range to $m=4n$, for example $\{2^n \pm 1, 2^n \pm 3\}$ [12], and $\{2^n \pm 1, 2^{2n}\}$ [13]. In addition, another four moduli set can be derived $\{2^n+k_1, 2^n+k_2, 2^n+k_3, 2^n+k_4\}$ with k_1, k_2, k_3 and k_4 positive/negative odd numbers or zero, chosen in such a way that a valid group of co-prime numbers is obtained [8]. The generators $2^n \pm k$ with *k* odd and $k \geq 3$ are more complex than the ones with $k=1$. However, some aplications of this sorts of residue generators to the field of *DSP* have been proposed in [5], [14]. At the present, one of the most efficient generator mod $A = 2^n \pm k$ are implemented using the periodic properties of powers of two modulo *A*, $\left|2^j\right|_A$ [14]. This paper proposes a quite different approach to build any moduli set with an appropiate dinamic range, which is based on the concept of threshold gate (TG) [15], rather than the traditional logic. TG logic allows for the implementation of more complex functions while reducing the logic depth and gate count, when compared with traditional logic gates [16].

In this paper we propose an improved topology for residue generators mod *A*. The implementation of this topology is carried out in threshold logic, which takes advantages of a novel input partitioning also proposed herein. Experimental results show that new residue generators using the novel partitioning can achieve improvements over 60% both area and delay in comparison with other ways of partitioning. This paper is organized as follows. Section II presents the topology based on threshold gates and the parameters which define them. Section III introduces the mathematical groundwork for the periodicity of the series powers of 2 carried out by mod *A*. In addition, a new efficient algorithm to compute the periodicity of any *A* is proposed. Section IV details the proposed general design. Section V presents a complexity analysis, in terms of delay and area, of the obtained circuits which are completed with the experimental results of Section VI. A summary of the work proposed is presented in Section VII.

II. GENERATOR MOD A

An *m*-input Boolean function $f(x_{m-1}, \ldots, x_1, x_0)$, which can be defined by $(m+1)$ real numbers (threshold *T* and weights $w_{m-1}, \ldots, w_1, w_0$, where weight w_i is associated with variable x_i) in such a way that:

$$f(x_{m-1}, \ldots, x_1, x_0) = \begin{cases} 1 \ \textit{iff} \left(\sum_{i=0}^{m-1} w_i x_i \geq T \right) \\ 0 \ \textit{otherwise} \end{cases} \quad (1)$$

where $x_i \in [0,1]$ and the multiplication and summation are arithmetic rather logic operations, is called a threshold function.

In a generator mod A with m input , $\{x_{m-1},...,x_1,x_0\}$ and a outputs represented as a set of residues $\{s_{a-1},...,s_1,s_0\}$, where $a=\lceil \log_2 A \rceil$. The generator converts an integer $X=\sum_{j=0}^{m-1} 2^j x_j$ into $|X|_A=\sum_{j=0}^{a-1} 2^j s_j$, i.e., it computes:

$$|X|_A=\left|\sum_{j=0}^{m-1} 2^j x_j\right|_A \tag{2}$$

The direct division of X by A to find the remainder is an inefficient solution. The technique proposed in [1] give us an improved algorithm based on the expression:

$$|X|_A=\left|\sum_{j=0}^{m-1}\left|2^j\right|_A x_j\right|_A=\sum_{i=0}^{a-1} 2^i s_i \tag{3}$$

If the values of $\left|2^j\right|_A$ are directly available, $|X|_A$ can be computed by merely adding mod A these terms $\left|2^j\right|_A$ for which $x_j=1$. In addition, these functions are periodic functions and so they can be implemented [15] by using separate TG networks.

Figure 1 shows an efficient topology to implement a residue generator mod A in a single TG network [16]. The threshold gate network has m inputs $\{x_{m-1},...,x_1,x_0\}$ with associated weights $\left\{\left|2^{m-1}\right|_A,\ldots,\left|2^1\right|_A,\left|2^0\right|_A\right\}$, represented on the left side of each threshold gate, which provides the sequences of residues. The value of the threshold, T, is showed in the lower right corner of each gate. Lets define the summatory of the sequence of residues as:

$$S=\left|\sum_{j=0}^{m-1}\left|2^j\right|_A\right| \ , \tag{4}$$

which has a size of bits of $t=\lceil \log_2(S+1)\rceil$. The value of $S=Q\cdot A+R$, provides the the quotient Q and the reminder R. The $(r+1)$ bit binary number $[S_{a+r},\ldots,S_{a+1},S_a]$, where

$$r=\lfloor \log_2(Q)\rfloor=\lfloor \log_2(\lfloor S/A\rfloor)\rfloor, \tag{5}$$

represents the quotient Q, and the a-bit number $[S_{a-1},\ldots,S_1,S_0]$ is the binary representation of the remainder R. In terms of delay, the division operation is implemented in

$$L=(\lceil \log_2 A\rceil+\lfloor \log_2(Q)\rfloor) \tag{6}$$

levels by using also L threshold gates. In terms of area, the main contribution of hardware is not given by the weights of the inputs but by the exponential increase of the weights associated to the quotient output, $2^r A$. This topology presents an unfeasible power-delay trade-off for a large values of r. Therefore, a more efficient scheme which reduces the value of the summatory of the sequence of residues is required. The proposal scheme is based on the periodic properties of powers of two mod A.

III. PERIODIC PROPERTIES OF $\left|2^j\right|_A$

The periodicity of the series of $\left|2^j\right|_A$ [14] will be of key importance for designing new generators mod A.

A. Generator Mod A based on the P(A) concept

This sort of generator is based on the adapted definition of period of the odd module *A P(A)* [17] which is described as follows:

- *Definition 1*: The *period of the odd module A P(A)* is the minimum distance between two distinct 1's in the array of residues of powers of 2 taken mod A, i.e *P(A)* = $\min\{j \mid j>0 \text{ and } |2^j|_A=1\}$. *P(A)* is simply called the *period of A* [14].

B. Generator Mod A based on the HP(A) concept

This sort of generator is based on the adapted definition of half-period of the odd module *A HP(A)* [17] which is described as follows:

- *Definition 2*: The *half-period of the odd module A HP(A)* is the minimum distance between a pair of subsequent 1 and $A-1$ in the array of residues of powers of 2 taken mod A. (Note that $A-1$ mod $A\equiv -1$).

Whereas *P(A)* exists for any *A*, *HP(A)* exists for some *A* only. *HP(A)* is called a half-period because if it exists then $P(A)=2\cdot HP(A)$. Several rules between them are derived in [14] given a generalization of the concept widely used in the case of $k=-1$. This concept sets that

$$\left|2^{za}\right|_{2^a+k}=1,\ z \text{ nonnegative integer and } k=-1, \tag{7}$$

and with the concept of *P(A)* is derived that

$$\left|2^{zP(A)+j}\right|_A=\left|2^j\right|_A,\ z \text{ nonnegative integer,} \tag{8}$$

similarly with the concept of *HP(A)* is derived that

$$\left|2^{zHP(A)+j}\right|_A=(-1)^z\cdot\left|2^j\right|_A,\ z \text{ nonnegative integer.} \tag{9}$$

C. Generator mod A based on a minimal summatory of contributions:

This novel generator is based on a new algorithm herein proposed, which explores the possibility of using positive and negative residues in the array of powers of 2 (as in the *HP*(*A*) case). This algorithm reduces the weight of the residue values in order to obtain a minimal weighted contribution for the summation of residues (4), and also for the value of r (5).

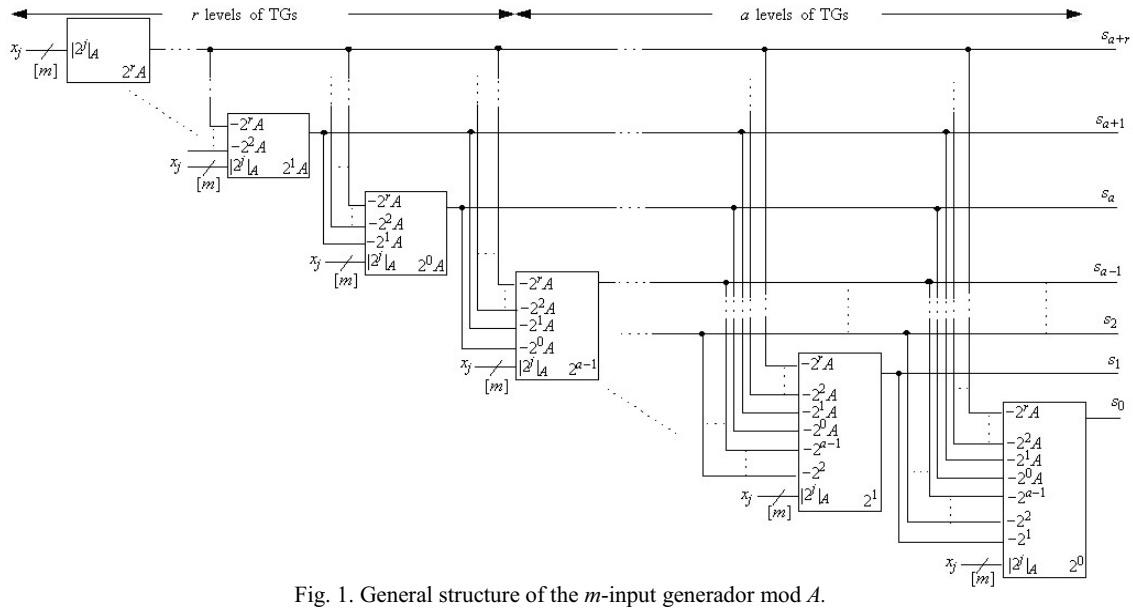

Fig. 1. General structure of the *m*-input generador mod *A*.

The proposed method to obtain the array of residues of powers of 2 is based on the property which establishes that for each x_j, two values of residue exist derived from $\left|2^j\right|_A$. This property is defined as follows,

$$\left|2^j\right|_A = \begin{cases} b_j, & 0 \le b_j < A \\ b_j - A, & -A < b_j - A \le 0 \end{cases} \tag{10}$$

Choosing the smaller value between $\left|b_j\right|$ and $\left|b_j - A\right|$ a minimal summatory of contributions can be obtained. Let us denoted this minimum value of residue contribution by v_j, which is obtained by the operation: $v_j = -\left|b_j - A\right|$ *iff* $\left|b_j\right| > (A-1)/2$, otherwise $v_j = \left|b_j\right|$; resulting v_j in the range $-\lfloor A/2 \rfloor + 1 \le i \le \lfloor A/2 \rfloor - 1$. Taking into account this idea, a novel algorithm is described applying that [14]:

$$\left|2^{j+1}\right|_A = \left|2 \cdot v_j\right|_A \tag{11}$$

Applying (11) recursively we can derived the new algorithm (Algortihm 1). Up to now, to obtain array of residues of powers of 2 mod *A*, the calculation of each $\left|2^j\right|_A$ was required. However, with this new algorithm, arrays of residue presented in [14] can be derived without any residue calculation. Different cases to obtain the array of residues have been denoted as: case 1, case 2, and case 3 if they are *P*(*A*) based, *HP*(*A*) based, or based on a minimal summatory of contributions, respectively. At line 10 is provided the residue sequence for case 3 in $\{v_{m-1},...,v_1,v_0\}$, whereas at line 17 and 26 is provided the residue sequence for cases 1 and 2 respectively in $\{b_{m-1},...,b_1,b_0\}$.

Algorithm 1 to obtain an array of residue values $\{v_{m-1},...,v_1,v_0\}$ with a minimal cost function in $S_{\max}$

```
1: for j = 1 : m
2:   v_1 = 1 & v_{j+1} = 2·v_j ;
3:   if v_{j+1} > ⌈A/2⌉
4:     v_{j+1} = 2·v_j − A ;
5:   elseif v_{j+1} < −⌈A/2⌉                    Case 3
6:     v_{j+1} = 2·v_j + A ;
7:   else
8:     v_{j+1} = 2·v_j ;
9:   end
10: end
11: for j = 1 : m
12:   if v_j < 0
13:     b_j = v_j + A ;
14:   else                                      Case 1
15:     b_j = v_j ;
16:   end
17: end
18: for j = 1 : m
19:   if b_j == A − 1
20:     for i = j : (⌈A/2⌉ − 1) + j
21:       b_j = b_i − A ;
22:     end                                     Case 2
23:   else
24:     b_j = b_j ;
25:   end
26: end
```

As values of $\left\|2^j\right|_A\Big|$ are always less than A, weights associated with primary inputs in the TG network are drastically reduced from 2^j to $\left|2^j\right|_A$, $j = 0, \ldots, m-1$. So, the set of input variables $\{x_{m-1},...,x_1,x_0\}$ can be partitioned into Γ_i sets, where $\Gamma_i = \{x_q \mid \left|2^q\right|_A = i\}$, and $-(A-1) \le i \le A-1$. In addition, the sets are fitted into $0 \le i \le A-1$, $-A+1 < i < A-1$, $-\lfloor A/2 \rfloor + 1 \le i \le \lfloor A/2 \rfloor - 1$ in case 1, 2 and 3 respectively.

In order to illustrate the concepts presented in this section an example for obtaining the three different arrays of residues for a 16-bit residue generator mod 13 is presented as follows. In case 1 the resulting array, with $j = 0, \ldots, 15$, is:

$$\left|2^{j=\{0,1,\ldots,15\}}\right|_{13} = \{1,2,4,8,3,6,12,11,9,5,10,7,1,2,4,8\}$$

In case 2,

$$\left|2^{j=\{0,1,\ldots,15\}}\right|_{13} = \{1,2,4,8,3,6,-1,-2,-4,-8,-3,-6,1,2,4,8\}$$

and in case 3,

$$\left|2^{j=\{0,1,\ldots,15\}}\right|_{13} = \{1,2,4,-5,3,6,-1,-2,-4,5,-3,-6,1,2,4,-5\}$$

In all cases we can order the partitions into 12 sets Γ_i, with the same weight i [14]. For example in case 1, $\Gamma_0 = \{x_0, x_{12}\}$, $\Gamma_1 = \{x_1, x_{13}\}$, ..., $\Gamma_7 = \{x_{11}\}$. Additionally, the weight, i, of one set of partition may be different for each case, for example x_9 is included into the set Γ_5 in case 1, Γ_{-8} in case 2, and Γ_5 in case 3.

IV. Design Procedure

In this section, we present a novel scheme for the m-input generator mod A based on threshold gates for the three different ways of partitioning presented in the previous section. This generator mod A can be designed by using the following procedure:

Step 1: Obtaining the array of residues of powers of 2 for m bits by means of the proposed algorithm.

Step 2: Standardize the range of the ouput of the residue generator to form the Γ_i sets in a positive range $0 \le i \le A-1$ as is in case 1. In cases 2 and 3 the residue generator output without any correction is in a range positive and negative. In [15] is proposed the addition of a correction term to obtain a typical range $0 \le i \le A-1$ in case 2. The correction value *COR* is chosen as the minimum value that satisfies $\left|\sum_{j \in v_j < 0} \overline{b_j} + COR\right|_A = 0$.

From [15] is derived that:

$$S = \sum_{j=0}^{m-1} \left|b_j\right| + COR \tag{12}$$

$$\left|X\right|_A = \left|\sum_{j \in b_j > 0} b_j + \sum_{j \in b_j < 0} \overline{b_j} + COR\right|_A \tag{13}$$

Is important to clarify that the *COR* factor in case 3 is obtained in the same way only replacing b_j by v_j. From now on and since *COR* is a constant, the correction factor is added to the threshold value of each TG of the residue generator.

Step 3: Once the sequence is defined and corrected we make the partition of the set of input bits $\{x_{m-1},...,x_1,x_0\}$ onto Γ_i sets, finishing the preparatory steps.

Step 4: The last step consist in calculating the value of r, which depends of the maximum summatory of all residue contributions $S_{\max} = \sum_{j=0}^{m-1} \left|2^j\right|_A + COR$ as is showed in (5).

In order to better illustrate the design procedure, the design of a 20-bit generator mod 37 is presented. Table I summarizes the parameters *COR*, $S_{\max}$, Q and r needed for the characterization of this case study. Figures 2 depicts the topology for cases 1, 2, and 3. The partitions for each case are derived as follows

Case 1: The number of partitions is $P(37) = 36$. In this case the addition of *COR* is not needed, since there are no partitions with negative weights. The partitioning in this case is $\Gamma_1 = \{x_0\}$, $\Gamma_2 = \{x_1\}$, ..., $\Gamma_{35} = \{x_{19}\}$ so the addition of *COR* to the thresholds, is not needed.

Case 2: In this case $20 > HP(37)$ since $HP(37) = 18$. In this case exists a correction factor $COR = 34$ obtained by $\left|3 + COR\right|_{37} = 0$. In this case 34 is added to all thresholds in each TG. The resulting partitioning are $\Gamma_1 = \{x_0, \overline{x}_{18}\}$, $\Gamma_2 = \{x_1, \overline{x}_{19}\}$, $\Gamma_4 = \{x_2\}$, ..., $\Gamma_{18} = \{x_{17}\}$.

Case 3: A correction factor, $COR = 3$, is also required, obtained by $\left|71 + COR\right|_{37} = 0$, which is added to all thresholds in each TG. The partitioning in this case is $\Gamma_1 = \{x_0, \overline{x}_{18}\}$, $\Gamma_2 = \{x_1, \overline{x}_{19}\}$, $\Gamma_4 = \{x_2\}$, ..., $\Gamma_{18} = \{x_{17}\}$.

Figure 2 depicts the complexity of the topologies with the differents ways of partitioning presented. The benefits exhibited in topology case 3 can be explained by the significant reduction in the value of $S_{\max}$, Q and r presented in Table I.

TABLE I

Parameters for designing residue generators mod 37

	COR	$S_{\max}$	Q	r
Case 1	0	402	10	3
Case 2	34	368	9	3
Case 3	3	177	4	2

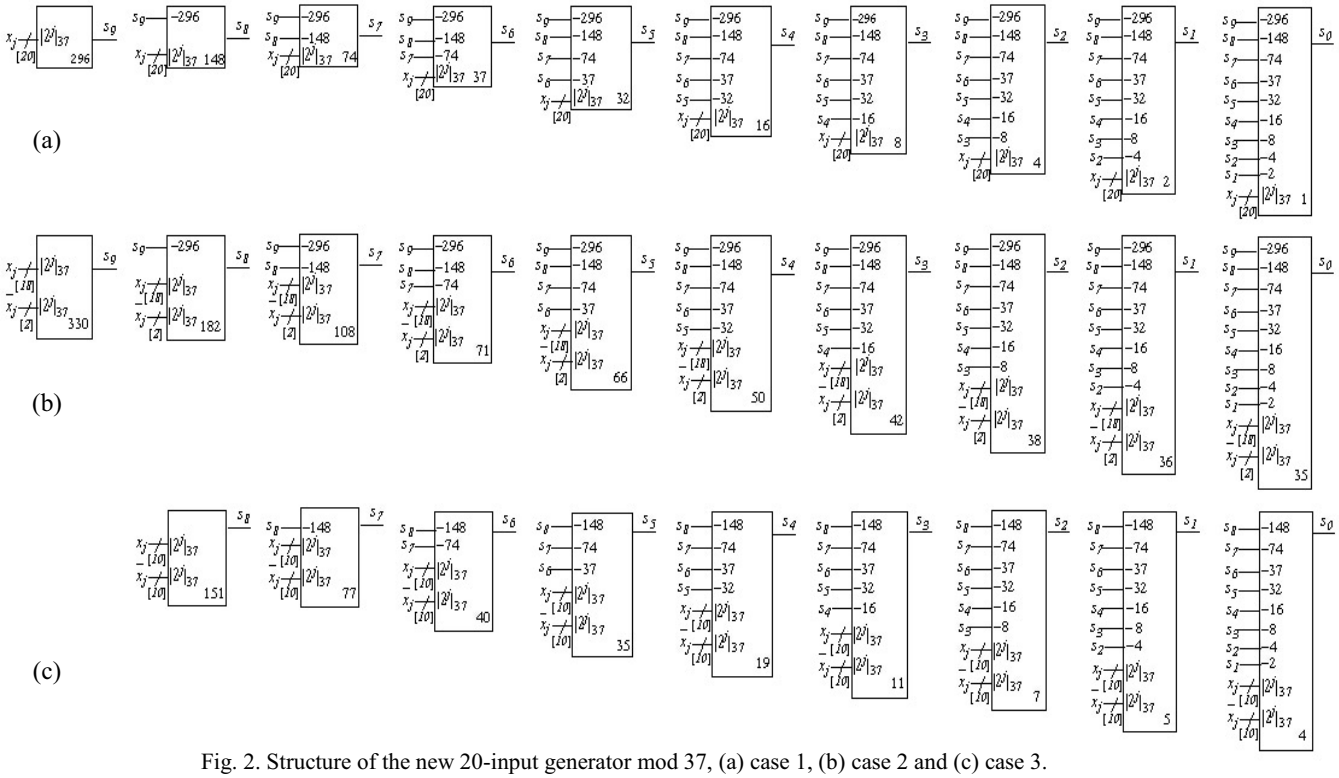

Fig. 2. Structure of the new 20-input generator mod 37, (a) case 1, (b) case 2 and (c) case 3.

V. COMPLEXITY ESTIMATION AND COMPARISON

Improved residue generators mod A can be achieved with the use of the proposed way of partitioning, which reduces the value of the summatory of sequence of residues (4). The novel residue generator mod A based on the proposed algorithm (case 3) is compared with the same topology using the other cases of partitioning [14]. All the designs were designed following the procedure described in Section IV. In order to obtain a reliable comparison of area and delay values, a simplified gate model based on the β-driven approach [18] is used. Fan-out and fan-in delays are not considered.

A. Delay Estimation

To estimate the delay, we need to analyze the depth of the stages which provides the value of the quotient, Q, and the residue, R. The first stage of TGs has a depth of r, whereas the second one has a depth of a, as shown in Figure 1. Thus, the delay estimation is given by:

$$Delay = a + r \tag{14}$$

Figure 3a depicts the delay estimation, in threshold gate levels for the new residue generator. These values reflect the average results for the several k odd numbers $\in [-7,+7]$, for each n. It is noticeable that the topology using case 3 of partitioning is faster than the same topology using the other cases of partitioning, resulting in a delay reduction of at least 6% in the whole range of analyzed n and k values.

B. Area Estimation

Let us consider the transistor-input area with an associated weight equal to 1 as the area unit used for this estimation. In our approach, an input x_j with an associated weight $\left|2^j\right|_A$ imposes an increase of area transistor of $\left|2^j\right|_A$ in comparison with an input with an associated weight equal to 1. Therefore, the measure of the area is carried out as the summation of all weighted contribution for each TG. With this, the overall estimation of area for the topology is:

$$Area = \left(2^0 + 2^1 + \ldots + 2^r\right) \cdot A \cdot a + S \cdot (a + r) + \ldots$$
$$\ldots + \left(2^r \cdot (r-1) + 2^{r-1} \cdot (r-2) + \ldots + 2^2\right) \tag{15}$$

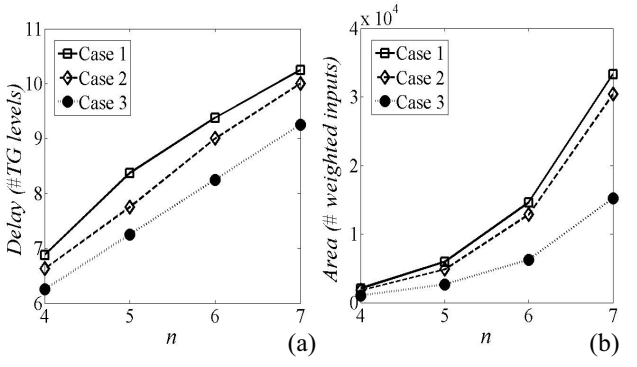

Fig. 3. Average comparisons for $2^n \pm k$, being k odd numbers $\in [-7,+7]$, (a) delay and (b) area.

Figure 3b depicts the average area values, in terms of weighted contributions of inputs for each TG. As above, the average values were obtained for each value of *n* in the range of odd $k \in [-7,+7]$. The obtained estimation suggests a significant area reduction for the topology using the proposed partitioning in comparison with the related art [14]. Thus, area reductions of at least 42%, are expected. The area estimation results indicate an exponential increase with *n,* for all three cases, giving an unfeasible power-delay trade-off for large values of *n.*

VI. Experimental Results

In order to evaluate the performance of the proposed residue generators experimental results were also obtained and compared. The topologies have been validated in CADENCE using 0.13μm Standard technology from UMC. Simulations of gates for the β-driven approach with this technology have proven that the fan-in is limited to small values of weighted inputs. Note that the technology used is not optimized for the design of threshold gates. Nevertheless, experimental results for residue generators with 16 input bits with presented topology have been obtained and compared.

TABLE II

Experimental results for cases 1 and 2 normalized to case 3

	k	*-5*	*-3*	*-1*	*+1*	*+3*	*+5*
Area	**Case 1**	1.62	1.63	1.14	2.51	2.14	1.12
	Case 2	1.23	1.18	1.16	1.00	1.64	1.13
Delay	**Case 1**	1.95	2.32	1.08	1.89	1.72	1.20
	Case 2	1.33	1.18	1.08	1.00	1.70	1.20

Table II shows the obtained experimental results, regarding area and delay of residue generators modulo $2^4 + k$, for case 1 and 2 normalized to the proposed case 3. The proposed topology with the novel method of partitioning exhibits equal or better area-delay trade-off in comparison with the related art presented in [14]. For example, in the particular case of residue generator mod 19 , $(2^4 + 3)$, an are reduction of 1.64 times and a speed up of 1.70 are achieved for topology 3 when compared with topologies 1 and 2, respectively.

VII. Conclusions

Novel methods to design residue generators modulo $2^n \pm k$, for threshold logic are presented for any *n* and *k*. The threshold logic approach takes advatage of the periodic properties of the powers of two modulo $2^n \pm k$. The proposed scheme reduces the hardware cost in comparison with traditional implementations using threshold gates. First, a novel algorithm, which exploits the periodic properties of the series of powers of 2 taken from modulo $2^n \pm k$, provides us with the sequences of residues in a more efficient way when compared with the existing state of the art. With this simpler algorithm, all the design can be derived for any value of *n* and *k*. The novel topology with three different ways to separate the partitions are compared, two from the related art, and the proposed one. The novel idea of separation shows to be advantageous in terms of area and delay, confirmed by both the theoretical analysis and the experimental results. These results suggest that improvements up to 64% in area and 70% in delay can be achieved.

References

[1] N.S. Szabo and R.I. Tanaka, Residue Arithmetic and its Applications to Computer Technology, McGraw-Hill, New York, 1967.

[2] Garner, "The residue number system", *IRE Trans. Electronic Computer*, vol. EC-8, pp.140-147, Jun.1959.

[3] G.C. Cardarilli, et al., "Reducing power dissipation in FIR filters using the residue number system", *IEEE Proc. Midwest Symp. on Circuits and Syst.*, pp. 320-323, 2000.

[4] P. G. Fernandez, et al., "A RNS-Based Matrix-Vector-Multiply FCT Architecture for DCT Computation," *Proc. 43th IEEE Midwest Symposium on Circuits and Systems*, pp. 350-353, 2000.

[5] S. J. Piestrak, "Self-testing checkers for arithmetic codes with any check base A," in *Proc. I991 Pacijic Rim tnt. Symp. Fault-Tolerant Syst.,* Kawasaki, Japan, Sept. 2627, 1991, pp. 162-167.

[6] J. Ramirez, et al., "Fast RNS FPL-Based Communications Receiver Design and Implementation," *Proc. 12th Int. Conf. Field Programmable Logic*, pp. 472-481, 2002.

[7] J. Bajard, and L. Imbert, "A Full RNS Implementation of RSA," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 769-774, Jun 2004.

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

[9] R. M. Capocelli and R. Giancarlo, "Efficient VLSI networks for converting an integer from binary system to residue number system and vice versa," *IEEE Trans. Circuits Syst.,* vol. CAS-35, pp. 1425-1430, Nov. 1988.

[10] R. Zimmermann, "Efficient VLSI implementation of modulo (2n±1) addition and multiplication", *Proc. of the 14th IEEE Symp. on Computer Architecture*, pp. 158-167, 1999

[11] D. Gallaher, F. Petry, and P. Srinivasan, "The digital parallel method for fast RNS to weighted number system conversion for specific moduli (2^k 1; 2^k; 2^k + 1)", *IEEE Trans. Circuits Syst. II*, vol. 44, pp. 53-67, Jan 1997.

[12] P.V. Mohan. "New reverse converters for the moduli set {2^n-3, 2^n-1, 2^n+1,2^n+3}", *Int Conf. on Signal Proc. & Communications*, pp 188-192, 2004.

[13] R. Chaves and L. Sousa. "{2^{n+1},$2^{(n+k)}$,2^{n-1}}:A New RNS Moduli Set Extention", *IEEE Euromicro Symposium on Digital System Design: Architectures, Methods and Tools*, IEEE Computer Society, pp. 210-217, Sep2004.

[14] S.J. Piestrak, "Design of Residue generators and multioperand adders using carry-save adders". *IEEE Trans. on Computers*, Vol. 43, pp. 68-77, Jan. 1994.

[15] S. Cotofana and Vassiliadis, "Periodic symmetric functions, serial addition, and multiplication with neural networks", *IEEE Trans. on Neural Networks,* Vol. 9, 6, Nov. 1998, pp.1118-1128.

[16] J.M. Quintana, M.J. Avedillo, H. Pettenghi, "Design of Residue Generators using Threshold Logic". *Proc. IEEE Midwest Symp. on Circuits and Systems*, 46; vol. 3, pp. 1427-1430

[17] V. Piuri, M. Berzieri, A. Bisaschi, and A. Fabi, "Residue arithmetic for a fault-tolerant multiplier: The choice of the best triple of bases," *Microproc. and Microprogr.,* vol. 20, pp. 15-23, 1988.

[18] V.I. Varshavsky, "β-driven threshold elements", *Proc. GLSVLSI'98*, Lafayette (USA), 1998, pp.52-58.