Dynamic Delay Variation Behaviour of RNS multiply-add Architectures

Kleanthis Papachatzopoulos, Ioannis Kouretas and Vassilis Paliouras
Electrical and Computer Engineering Dept.
University of Patras
Patras, Greece

***Abstract*—In this paper we investigate the impact of intra- and inter-die variations on the delay sensitivity of certain Residue Number System (RNS) arithmetic circuits in comparison to ordinary binary arithmetic logic. The timing yield of systems that contain multiply-add units (MAC) is of great importance since they dominate important applications such as digital signal processing. Specifically, we employ two different delay models for the estimation of delay distributions of RNS and binary MAC architectures. Our analysis quantitatively proves that RNS MAC architectures that use bases of the form $\{2^n-1, 2^n, 2^n+1\}$ demonstrate better normalized delay variation than binary MAC architectures to characterize both their static timing behaviour and the timing behaviour taking into account the sensitizable paths. Furthermore, it is shown that certain simplified RNS MAC architectures outperform conventional RNS MAC architectures in terms of the $\mu + a \cdot \sigma$ delay variation metric.**

***Index Terms*—Residue arithmetic, variation-tolerant design, RNS, Monte-Carlo.**

I. Introduction

As process technology moves below the 90-nm technology node, integrated circuits are becoming severely prone to process variations, as more and more inaccuracies are introduced in the manufacturing process. Such unwanted variations can significantly affect the timing and power yield of ICs. For this reason, novel methods are sought to handle process-induced variability and to render the circuit architectures tolerant to process variations.

The main technology parameters affected by the introduced variations are gate length, L_{gate}, gate width, W_{gate}, oxide thickness, T_{ox}, threshold voltage, V_{th}, the nature of which can be random or systematic based on the process stages that are involved in their construction [1]. These variations can be further distinguished into *inter-* and *intra-die* variations. Inter-die variations refer to variations that are manifested from die to die and affect device parameters in the same manner across the chip, while intra-die variations affect differently each device.

Traditional Static Timing Analysis used for estimation of timing yield and adjustment of the design in order to satisfy the timing constraints became inefficient due to the fact that corner-based models do not provide the required efficiency and are highly pessimistic. In order to tackle this problem, *Statistical Static Timing Analysis (SSTA)* has been proposed as an alternative, handling delays as probability density functions (PDF). In [2], an efficient statistical timing analysis algorithm is used, considering spatial correlation among devices. A quadratic model of gate and wire delays is used in [3] providing higher accuracy improvement than the linear canonical delay model.

In this paper we investigate RNS as a representation which leads to multiply-add units that behave well in the presence of process inter- and intra process variations. Similar analysis in the past indicate such benefits, yet they rely on models that do not take into account dependencies on input data. More precisely, we use an improved model over [4], so as to take into account the role of data input patterns to the delay variation. The new model facilitates a more realistic timing analysis than the conventional static timing analysis, taking into account the sensitisable critical paths. It is quantitatively shown that RNS MACs outperform binary MACs in terms of *normalized delay variations*, even under the new data-dependent model. Furthermore, specific RNS structures are comparatively studied in terms of the delay variations and are found to provide variation tolerant solutions.

The rest of this paper is organized as follows: Section II describes the RNS basics and the architecture of RNS and binary multiply-add units and Section III describes the employed Delay Model. Section IV presents and discusses the experimental results of SMCA and DMCA and Section V provides the conclusion.

II. RNS basics

A. Review of RNS basics

The RNS transforms an integer X into an N-tuple of residues x_i,

$$X \xrightarrow{\text{RNS}} \{x_0, x_1, \ldots, x_{N-1}\} \tag{1}$$

and $x_i = \langle X \rangle_{m_i}$, where $\langle \cdot \rangle_{m_i}$ denotes the modulo m_i operation and m_i is a member of a set of N integers, $\{m_0, m_1, \ldots, m_{N-1}\}$, which is called *base* of the RNS. The modulo operation $\langle X \rangle_m$ returns the integer remainder of the division $X \bmod m$, i.e., the integer k such that $X = m \cdot l + k$ where l is an integer. The requirement on the members of base for a unique representation of the integer X is that they should be co-prime integers, that is $gcd(m_i, m_j) = 1$, $i \neq j$. Thus, each integer X is uniquely represented using the base $\{m_0, m_1, \ldots, m_{N-1}\}$, $0 \leq X \leq \prod_{i=0}^{N-1} m_i$. The product

$$M = \prod_{i=0}^{N-1} m_i \tag{2}$$

is called *dynamic range* of the RNS.

Each operation of the form $Z = \langle Y \diamond X \rangle_{m_i}$, where $X \xrightarrow{\text{RNS}} \{x_0, x_1, \ldots, x_{N-1}\}$, $Y \xrightarrow{\text{RNS}} \{y_0, y_1, \ldots, y_{N-1}\}$ and $Z \xrightarrow{\text{RNS}} \{z_0, z_1, \ldots, z_{N-1}\}$ and $\diamond$ denoting one of the operations of addition, subtraction or multiplication, can be performed as $z_i = \langle x_i \diamond y_i \rangle_{m_i}$ for $i = 0, 1, \ldots, N-1$. As it can be seen, the above operations are performed in a carry-free manner. This facilitates the result computation of wide bit length numbers, because the computation is partitioned into a set of N parallel computations, as the result of each channel independence on the data of any other channel. An integer expressed in RNS can be converted to binary representation by means of the Chinese Remainder Theorem (CRT).

B. RNS-based MAC Units

This subsection describes the organization of Modulo and binary multiply-add units. These circuits perform the operation $A \cdot B + C$, where A, B, and C are integers. The architecture of a MAC consists of three main parts as depicted in Fig. 1: the first one, an AND

978-1-4799-5341-7/16/$31.00 ©2016 IEEE

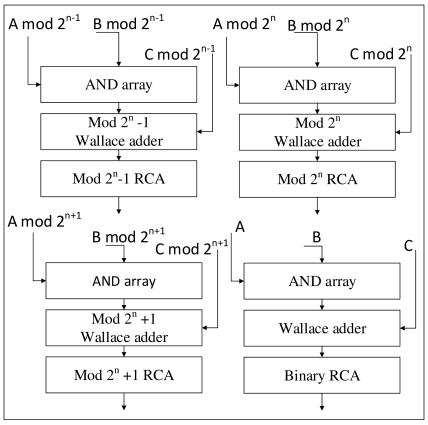

Fig. 1. The organization of RNS and binary MAC.

array, generates the partial products of multiplication, the second one, the Modulo/binary Wallace adder, adds the partial products generated form the AND array with the bits of operand C, and the last one performs the final carry-propagate addition of the two partial results generated from the Modulo/binary Wallace adder using a Ripple-Carry Adder. The partial products summation is performed by a Wallace tree, which reduces the bit products at the earliest opportunity, using FAs and HAs [5]. In case of final modulo m_i addition, with $m_i \neq 2^n, 2^{n-1}$, an additional Subtractor-Multiplexer circuit in order to map the data to the range $[0, m_i - 1]$ is used.

We implemented and compared two different modulo MAC architectures. The first one is a conventional FA-based MAC architecture and the second is a simplified architecture introduced in [6]. The simplified architecture reduces the number of FAs-HAs used for the residue bit product addition of the operands, exploiting certain properties of bit products in modulo architectures with redundant representations. An algorithm introduced in [6] can identify bit products, the sum of which is always less than two, thus allowing the use of OR gates to reduce the adder array.

When a sufficient amount of computations are performed using the same representation, the delay-variation overhead of converters can be compensated using resizing, as they are used only in the beginning and the end of processing.

III. Delay Models

A. Static and Dynamic Monte-Carlo Analysis

In order to estimate the timing behaviour of the circuits described in detail in Section II-B, we perform two types of Monte-Carlo simulations, namely Static Monte-Carlo Analysis (SMCA) and Dynamic Monte-Carlo Analysis (DMCA). For each type of simulation, we adopt a different delay model, each providing results with a different level of accuracy. The SMCA and DMCA delay models are discussed in Section III-B.

For the case of SMCA, Gaussian random variables are used to model the impact of inter- and intra-die variations on gate delays, in order to estimate the maximum output delay of each circuit instance. We consider 1000 instances, i.e., we assume that each circuit is implemented on one of 1000 different dies. This type of analysis estimates the upper bound to the maximum delay at the output of a circuit.

On the other hand, for the case of DMCA, we estimate the maximum output delay using a more detailed model than SMCA, with the objective to take into account data dependencies of the delays, i.e., to capture the impact of the previous output value of a logic component on the output delay when a new input pattern is applied to the input. Specifically, while we model gate delays as Gaussian random variables resembling SMCA, we assume that each circuit is implemented on one of 100 different dies and, for each of these dies, we measure the maximum delay of circuit, by applying an input signal of 100 uniformly distributed random values, limited due to the required computational cost.

TABLE I
Fast control values which define output delay for several gates

Gate	Control Value	Transition
OR	1	$0 \rightarrow 1$
NOR	1	$1 \rightarrow 0$
AND	0	$1 \rightarrow 0$
NAND	0	$0 \rightarrow 1$

B. Employed Delay Models

This subsection describes the employed delay models. Initially, we use the a model introduced by Bowman *et al.* [7]. Let the i^{th} logic element of test circuit be implemented on the j^{th} die. The delay of this element, D_{ij}, is modeled as:

$$D_{ij} = D_{\text{nom}}(i) + D_{\text{inter}}(i,j) + D_{\text{intra}}(i,j), \quad (3)$$

where $D_{\text{nom}}(i)$ represents the nominal delay of i^{th} logic element, $D_{\text{inter}}(i,j)$ and $D_{\text{intra}}(i,j)$ represent the independent inter-die and intra-die delay variations, respectively. Variations $D_{\text{inter}}(i,j)$ and $D_{\text{intra}}(i,j)$ are modeled as Gaussian random variables. Since all logic elements implemented on the j^{th} die have the same inter-die variation, $D_{\text{inter}}(i,j)$, the delay model expressed by (3) is reduced to

$$D_{ij} = D_{\text{nom}}(i) + D_{\text{inter}}(j) + D_{\text{intra}}(i,j). \quad (4)$$

As mentioned before, two types of variation analysis are performed, namely SMCA and DMCA. For SMCA, we estimate the maximum delay at the output of a circuit, assuming random Gaussian variables for the inter- and intra-die variations. Eq. (5) expresses the delay at the output of i logic element:

$$T_i = \max\{T_{i-1}, T_{i-2}, \ldots, T_{i-k}\} + D_{ij}, \quad (5)$$

where D_{ij} is the delay of logic element i expressed by (4), $T_{i-1}, T_{i-2}, \ldots, T_{i-k}$ denote the delays at the output of logic elements $i-1, i-2, \ldots, i-k$, respectively which drive logic element i.

For the DMCA, we further improve the delay model described by (5) to compute the delay imposed on a signal transition propagating through the particular gate. The next delay model is a simplified version of the timing model presented in [8]. If no transition propagates through the gate, the gate does not contribute to the delay. As an illustrative example, assume that the output of an OR gate is 0 and its inputs are both switching to 1, but not simultaneously. The output becomes 1 and the output delay is determined by the path delay of the input that becomes 1 first, since 1 is a control value for an OR gate. In this case, the faster path determines the actual time, when the output of the gate becomes 1. Table I depicts a list of gates and the transitions due to control input values that allow the particular output delay computation. The first column of Table I denotes the gate functionality and the second one the control value that determines the output delay for the transition shown in the third column. Eq. (6) describes mathematically the delay model used for

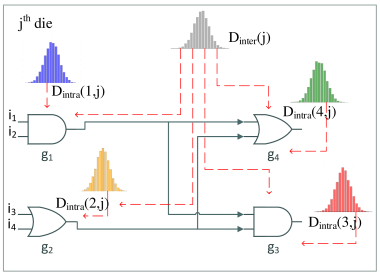

Fig. 2. Delay Model.

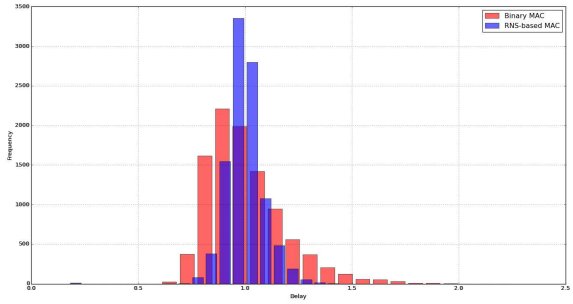

Fig. 3. Normalized Delay Distributions for RNS-based MAC and binary MAC in case of 0.2 inter- and 0.1 intra-die variations.

DMCA:

$$T_i = \begin{cases} \max\{T_{i-1}, T_{i-2}, \ldots, T_{i-k}\} + D_{ij}, & \text{if } \tilde{g}_i = 1, \tilde{h}_i = 0 \\ \min\{T_h, T_{h-1}, \ldots, T_{h-k}\} + D_{ij}, & \text{if } \tilde{g}_i = 1, \tilde{h}_i = 1, \\ 0, & \text{if } \tilde{g}_i = 0 \end{cases} \tag{6}$$

where T_i is the total delay at the output of logic element i, $\tilde{g}_i = 0$ and 1 denotes the lack or the existence of output transition for the logic element i respectively, $\tilde{h}_i$ is 1 when one of the rows of Table I is true, i.e., the transition is due to the control value, $i-1, i-2, \ldots, i-k$ is the set of fan-in elements of logic element i and $h, h-1, \ldots, h-k$ is the subset of fan-in elements of logic element i, the output value of which assume the *control value* of Table I.

As an example, assume the circuit of Fig. 2, composed by four gates, namely g_1, g_2, g_3 and g_4 manufactured on the j^{th} die. Now, assume that the input pins i_1, i_2, i_3, and i_4 are assigned successively the following input vectors 1100 and 1110. Then $\tilde{g_1} = 0$ and $\tilde{g_2} = 1$ and according to DMCA delay model (6), the delays T_1 and T_2 of g_1 and g_2 are evaluated as:

$$T_1 = 0, \tag{7}$$

$$\begin{aligned} T_2 &= \min\{T_{i_3}, T_{i_4}\} + D_{2j} \\ &= \min\{T_{i_3}, T_{i_4}\} + D_{\text{nom}}(2) + D_{\text{inter}}(j) + D_{\text{intra}}(2, j), \end{aligned} \tag{8}$$

where T_{i_3}, T_{i_4} is the delay between successive values of input pins i_3 and i_4. Similarly, the delay of the gates g_3 and g_4 are respectively:

$$\begin{aligned} T_3 &= \max\{T_1, T_2\} + D_{4j} \\ &= \max\{T_1, T_2\} + D_{\text{nom}}(3) + D_{\text{inter}}(j) + D_{\text{intra}}(3, j), \end{aligned} \tag{9}$$

$$T_4 = 0. \tag{10}$$

As for the nominal delays, a full adder (FA) is assumed to have a delay for input to carry output, $t_{\text{FA},s_{in}\rightarrow c_o} = 2t_{\text{gate}}$ and from input to sum output $t_{\text{FA},s_{in}\rightarrow s_o} = 2.5t_{\text{gate}}$, where t_{gate} is the unit gate delay. Additionally, AND-OR and XOR gates have a nominal delay of $t_{\text{AND-OR,XOR}} = t_{\text{gate}}$, while AND and OR gates have a nominal delay of $t_{\text{AND,OR}} = 0.5t_{\text{gate}}$. A half adder (HA) is assumed to have a nominal delay from input to carry output $t_{\text{HA},s_{in}\rightarrow c_o} = t_{\text{gate}}$ and from input to sum output $t_{\text{HA},s_{in}\rightarrow s_o} = 0.5t_{\text{gate}}$. The particular values are obtained based on actual cell delays of a standard-cell library.

IV. Experimental Results and Discussion

This section discusses the impact of inter- and intra-die variation on the delay sensitivity of certain MAC architectures. The following results were obtained under the assumption of Gaussian inter- and intra-die variations. The delay sensitivity of certain RNS and binary structures that demonstrate different mean delay values is quantified using normalized metrics. In particular, let T_{RNS} and T_{Bin} denote the delay of RNS and binary structures, respectively, and m_{RNS} and m_{Bin} denote their mean values. Then $T'_{\text{RNS}} = \frac{T_{\text{RNS}}}{m_{\text{RNS}}}$ and $T'_{\text{Bin}} = \frac{T_{\text{Bin}}}{m_{\text{Bin}}}$ are the corresponding normalized delays. Therefore, the delay variation, $\sigma^2(T')$, is the variance of normalized delays and we refer to this quantity as *normalized delay variation*.

TABLE II
DMCA for RNS-based and binary MAC.

inter-die	intra-die	$\sigma^2_{T'_{Bin}}$	$\sigma^2_{T'_{RNS}}$	Reduction %
0.05	0.05	0.0307	0.0050	83.62
0.05	0.1	0.0289	0.0054	81.44
0.05	0.15	0.0303	0.0053	82.54
0.05	0.2	0.0291	0.0058	80.05
0.05	0.25	0.0291	0.0062	78.69
0.05	0.3	0.0285	0.0066	76.70
0.1	0.05	0.0307	0.0056	81.69
0.1	0.1	0.0301	0.0055	81.64
0.1	0.15	0.0297	0.0056	81.24
0.1	0.2	0.0293	0.0058	80.39
0.1	0.25	0.0285	0.0062	78.35
0.1	0.3	0.0297	0.0070	76.23
0.15	0.05	0.0301	0.0062	79.51
0.15	0.1	0.0309	0.0066	78.79
0.15	0.15	0.0299	0.0063	78.77
0.15	0.2	0.0299	0.0063	79.00
0.15	0.25	0.0292	0.0071	75.72
0.15	0.3	0.0300	0.0071	76.34
0.2	0.05	0.0321	0.0067	79.13
0.2	0.1	0.0327	0.0075	76.93
0.2	0.15	0.0313	0.0078	75.19
0.2	0.2	0.0298	0.0072	75.87
0.2	0.25	0.0305	0.0074	75.79
0.2	0.3	0.0312	0.0077	75.49
0.25	0.05	0.0334	0.0086	74.18
0.25	0.1	0.0333	0.0085	74.49
0.25	0.15	0.0348	0.0081	76.77
0.25	0.2	0.0320	0.0088	72.44
0.25	0.25	0.0323	0.0080	75.27
0.25	0.3	0.0326	0.0090	72.26
0.3	0.05	0.0347	0.0093	73.31
0.3	0.1	0.0355	0.0095	73.33
0.3	0.15	0.0344	0.0100	71.06
0.3	0.2	0.0366	0.0091	75.26
0.3	0.25	0.0353	0.0094	73.43
0.3	0.3	0.0312	0.0102	67.27

A. DMCA and SMCA for Modulo and Binary MAC

Initially, DMCA is performed on RNS $\{2^n-1, 2^n, 2^n+1\}$ MAC, using a non-diminished representation for the modulo-$\left(2^n+1\right)$ channel, and binary MAC with the same dynamic range. Table II presents

TABLE III
PERCENTAGE OF INSTANCES OF RNS-BASED AND BINARY MAC THE DELAY OF WHICH ARE ABOVE $\mu + 2 \cdot \sigma$ IN CASE OF DMCA.

inter-die	intra-die	BIN %	RNS %	Reduction %
0.1	0.1	0.0454	0.0447	1.54
0.1	0.2	0.0451	0.0396	12.19
0.1	0.3	0.0463	0.0312	32.61
0.2	0.1	0.0453	0.0364	19.64
0.2	0.2	0.0458	0.0342	25.32
0.2	0.3	0.0430	0.0324	24.65
0.3	0.1	0.0435	0.0318	26.89
0.3	0.2	0.0456	0.0284	37.71
0.3	0.3	0.0421	0.0308	26.84

TABLE IV
PERCENTAGE OF INSTANCES OF RNS-BASED AND BINARY MAC THE DELAY OF WHICH ARE ABOVE $\mu + 2 \cdot \sigma$ IN CASE OF SMCA.

inter-die	intra-die	BIN %	RNS %	Reduction %
0.1	0.1	0.0240	0.0180	25.00
0.1	0.2	0.0250	0.0240	4.00
0.1	0.3	0.0220	0.0160	27.27
0.2	0.1	0.0220	0.0260	-18.18
0.2	0.2	0.0190	0.0230	-21.05
0.2	0.3	0.0210	0.0210	0.00
0.3	0.1	0.0220	0.0280	-27.27
0.3	0.2	0.0220	0.0260	-18.18
0.3	0.3	0.0290	0.0180	37.93

the normalized delay variation for the case of the $\{2^{10}-1, 2^{10}, 2^{10}+1\}$ RNS base compared to binary MAC with a 30-bit dynamic range. As it can be noticed from Table II, the savings range up to 80%. Additionally, Table III depicts the percentage of instances the maximum delay of which is greater than $\mu_{T'} + 2 \cdot \sigma_{T'}$, where μ is the mean value and σ the standard deviation of T'. The experimental data reveal that the RNS architecture offers significant normalized delay tolerance against binary architectures for the DMCA. Table IV depicts the percentage of instances the maximum delay of which is greater than $\mu_{T'} + 2 \cdot \sigma_{T'}$, but in case of SMCA. In same cases of Table IV binary MAC outperforms RNS MAC and this may happen due to the limited number of instances. Fig. 3 depicts the two normalized delay distributions in case of 0.2 inter die and 0.1 intra die variations.

B. DMCA for OR-based and conventional Modulo MAC

Table V compares the delay variation of the simplified OR-based modulo MAC described in [6] and a conventional modulo MAC (cf. [4]) using the quantity $\mu + a \cdot \sigma$ as a variation metric for certain

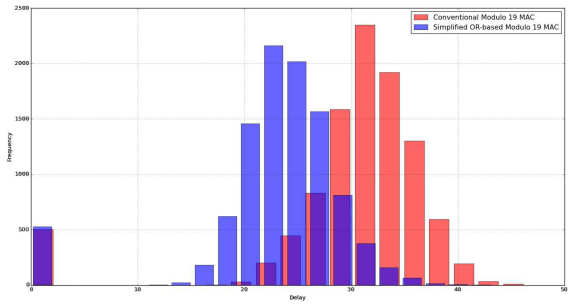

Fig. 4. Delay Distribution of Novel Modulo-19 MAC and conventional Modulo-19 MAC in case of 0.2 inter- and 0.1 intra-die variations.

TABLE V
NUMBER OF INSTANCES OUT OF 10^4 OF MOD-19 MAC AND OR-BASED MOD-19 MAC THE DELAY OF WHICH ARE ABOVE $\mu_{conv} + a \cdot \sigma_{conv}$ DELAY BOUND FOR CERTAIN VALUES OF a IN CASE OF DMCA.

a		0.5		1		1.5	
inter-die	intra-die	Conv	OR	Conv	OR	Conv	OR
0.1	0.1	2567	69	374	6	18	1
0.1	0.2	2662	109	521	7	48	0
0.1	0.3	2720	161	640	26	81	0
0.2	0.1	2640	104	523	9	36	1
0.2	0.2	2687	147	576	25	66	1
0.2	0.3	2792	168	648	21	76	2
0.3	0.1	2675	83	640	7	72	0
0.3	0.2	2715	137	710	20	117	2
0.3	0.3	2847	198	819	25	133	4

values of a. The delay is determined by the $\mu + a \cdot \sigma$ of conventional modulo-19 MAC. More specifically, Table V reports the number of circuit instances that exceed certain timing constraints for modulo-19 MAC, implemented with the previous two architectures, evaluated via DMCA. It is clearly shown that the simplified architecture presents improved delay variation characteristics, since the number of the simplified modulo-19 MAC instances that exceed the $\mu + a \cdot \sigma$ delay bound is substantially less than that for the conventional modulo-19 MAC. Fig. 4 depicts the delay distributions in case of 0.2 inter- and 0.1 intra-die variations.

V. CONCLUSION

In this paper, various RNS structures are compared with the equivalent binary structure or with other RNS structures. The metrics used are normalized delay variation and $\mu + a \cdot \sigma$. It is shown in general that modulo MAC with bases of the form $\{2^n - 1, 2^n, 2^n + 1\}$ perform better with the presence of intra- and inter-die variations.

Further DMCA shows that the simplified OR-based architecture presented in [6] exhibits improved delay variation characteristics than the FA-based architectures, in terms of number of instances that violate a given timing constraint.

REFERENCES

[1] M. Orshansky, S. Nassif, and D. Boning, *Design for manufacturability and statistical design: a constructive approach.* Springer Science & Business Media, 2007.

[2] H. Chang and S. S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,*, vol. 24, no. 9, pp. 1467–1482, 2005.

[3] L. Zhang, W. Chen, Y. Hu, J. Gubner, C. C.-P. Chen *et al.*, "Correlation-preserved statistical timing with a quadratic form of gaussian variables," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2437–2449, 2006.

[4] I. Kouretas and V. Paliouras, "Residue arithmetic for variation-tolerant design of multiply-add units," in *Integrated Circuit and System Design. Power and Timing Modeling, Optimization and Simulation*. Springer, 2010, pp. 26–35.

[5] B. Parhami, *Computer arithmetic: algorithms and hardware designs*. Oxford University Press, Inc., 2009.

[6] G. Dimitrakopoulos and V. Paliouras, "A novel architecture and a systematic graph-based optimization methodology for modulo multiplication," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 2, pp. 354–370, 2004.

[7] K. Bowman, S. G. Duvall, and J. D. Meindl, "Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 183–190, 2002.

[8] R. Garg and S. P. Khatri, "Sensitizable statistical timing analysis," in *Analysis and Design of Resilient VLSI Circuits*. Springer, 2010, pp. 131–151.