Runtime Resonance Noise Reduction with Current Prediction Enabled Frequency Actuator

Yiyu Shi, Jinjun Xiong, Howard Chen, and Lei He

***Abstract*—Power delivery network (PDN) is a distributed resistance–inductance–capacitance *(RLC)* network with its dominant resonance frequency in the low-to-middle frequency range. Though high-performance chips' working frequencies are much higher than this resonance frequency in general, chip runtime loading frequency is not. When a chip executes a chunk of instructions repeatedly, the induced current load may have harmonic components close to this resonance frequency, causing excessive power integrity degradation. Existing PDN design solutions are, however, mainly targeted at reducing high-frequency noise and not effective to suppress such resonance noise. In this work, we propose a novel approach to proactively suppress this type of noise. A method based on the high dimension generalized Markov process is developed to predict current load variation. Based on such prediction, a clock frequency actuator design is proposed to proactively select an optimal clock frequency to suppress the resonance. To the best of our knowledge, this is the first in-depth study on proactively reducing instruction loop induced PDN resonance noise at the runtime.**

***Index Terms*—Frequency actuator, resonance noise, stochastic current modeling.**

I. Introduction

The continuous scaling trends of higher operating frequency, lower power supply voltage, and more functionality for integrated circuits have made it extremely challenging to design a reliable power delivery network (PDN). There are two dominant types of noise present in a PDN: *peak noise* and *resonance noise* [2]. Peak noise usually occurs when the instantaneous switching current load becomes maximum [8] for a short duration with its energy spectrum lying in the high-frequency range [2]. Abundant research has been done to minimize peak noise for PDN design (e.g., [3], [7], [10], [13]).

Resonance noise is a result of the distributed resistance–inductance–capacitance (*RLC*) characteristics of the PDN, which includes parasitic inductance of interconnect and decoupling capacitance. With details to be discussed in Section II, it compromises chip performance, hold-time margins, and gate oxide integrity [1], [12]. Despite the importance of resonance noise for reliable PDN design, resonance noise suppression has not gained enough attention in the electronic design automation (EDA) community. The traditional solutions, such as adding more passive capacitors to diminish the supply's ac impedance, more supply pins, using C4 packages to lower inductance, are not effective to suppress the resonance noise, as both off-chip and even on-board capacitors are self-resonant at frequencies well below $f_{\rm res}$. Recently, Ang *et al.* [1] proposed to dynamically switch on-chip decoupling capacitors to suppress resonance noise for microprocessors' PDN designs, while the authors of [12] provided an on-die resonance-suppression circuit that uses band-limited active damping to reduce resonance noise. However, in addition to the large area overhead and leakage issues due to decaps, all these approaches are retroactive, i.e., they only remedy the noise problem when the noise problem has occurred already. However, this is often too late: Wrong values might be latched at the rising/falling edge of the clock right after the violation occurs, thus causing a full save and restore of the whole architecture state later. To avoid such performance lost, a better way would be proactively suppressing the resonance noise when such issues are predicted to happen soon.

In this paper, we propose a proactive PDN design approach to suppress resonance noise by dynamically adjusting run-time clock frequency with the aid of a frequency actuator, which contains on-chip dynamic power supply current sensors to measure the dynamic current loads and programmable phase-locked loops (PLLs) to change clock frequencies. Compared with baseline design without frequency actuator, experimental results show that the frequency actuator without current prediction reduces maximum noise by 16% and average noise by 30%, while the frequency actuator with current prediction reduces maximum noise by 77% and average noise by 85%.

The remainder of this paper is organized as follows. We motivate the study of this work in Section II, and present the problem formulation and overall design methodology in Section III. We develop the stochastic current prediction and optimal frequency selection algorithm in Section IV. The experimental results are presented in Section V and concluding remarks are given in Section VI.

II. Motivation

We illustrate the concepts of both peak noise and resonance noise by adopting a simple lumped *LC* model of the power delivery system through PCB and package to chip in Fig. 1. L_b and C_b are the lumped inductance and capacitance of the PCB board. L_p and C_p represent those of the package, C_c is the lumped on-chip intrinsic capacitance, and C_d is the decoupling capacitance. Power supply voltage is $V_{\rm dd}$, and the dynamical current load of the PDN is $I(t)$. Voltage fluctuation due to $I(t)$ is given by the voltage droop $V_n(t)$. For the simplicity of presentation, the lumped resistors are not included in this model as they do not directly impact the nature of the analysis to be performed.

Most existing work on PDN designs models the load $I(t)$ as a *single* current spike $I_0(t)$ with a short duration time τ_p, i.e., $I(t) = I_0(t)$ for $0 \le t \le \tau_p$, and otherwise $I(t) = 0$. It is typical that the current spike $I_0(t)$ is modeled as a triangular waveform within $[0, \tau_p]$ [8]. The peak noise resulting from $I_0(t)$ is essentially a high-frequency noise with its frequency in the range of $1/\tau_p$. For 65 nm designs, the duration τ_p is on the order of 100 ps, which produces peak noise on the order of 10 GHz.

But in reality, the current load $I(t)$ possesses a much lower frequency component due to the periodic nature of functional execution. Without loss of generality, we assume $I(t)$ has $I_0(t)$ as its only periodic current component[1] with a period of $T \gg \tau_p$, i.e.,

$$I(t) = \sum_{k=0}^{\infty} I_0(t - kT). \tag{1}$$

[1]The real case can be treated as a superposition of this simple scenario with different combinations of $I_0(t)$ and T.

Manuscript received March 25, 2009; revised August 06, 2009 and October 02, 2009. First published December 11, 2009; current version published February 24, 2011. This paper is supported in part by an NSF CAREER Award and a UC MICRO grant sponsored by Actel and Fujitsu. Some preliminary results of this paper appeared in Proceedings of the Asia and South Pacific Design Automation Conference (ASPDAC'09), pp. 373–378.

Y. Shi and L. He are with Department of Electrical Engineering, University of California, Los Angeles, CA 90095 USA (e-mail: yshi@ee.ucla.edu; lhe@ee.ucla.edu).

J. Xiong and H. Chen are with the IBM Research Division, T. J. Watson Research Center, Yorktown Heights, NY 10598 USA (e-mail: jinjun@us.ibm.com; haowei@us.ibm.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2009.2036266

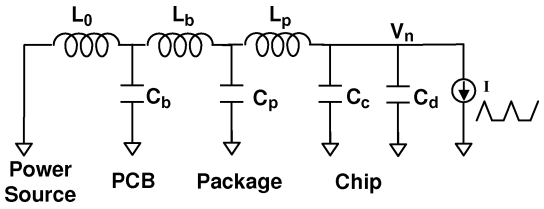

Fig. 1. Circuit representation of a PDN.

The corresponding spectrum can be obtained by performing Fourier transform of (1) as

$$I(j\omega) = \frac{\sqrt{2\pi}}{T} \sum_{k=-\infty}^{\infty} I_0(jk\Omega)\delta(\omega - k\Omega) \tag{2}$$

where $\Omega = 2\pi/T$ and $I_0(j\omega)$ is the Fourier transform of $I_0(t)$. δ is the unit impulse function.

Next we discuss about the computation of the dominant resonance frequency $\omega_{\rm res}$ of the system in Fig. 1. A direct derivation, however, requires the expression of the impedance at V_n, which is a very complicated expression. To avoid this, we derive it under two assumptions which always hold for realistic designs. With those assumptions, the resonance frequency is straightforward to compute.

First, if the resonance frequency $\omega_{\rm res}$ satisfies

$$\omega_{\rm res} L_b \gg \frac{1}{\omega_{\rm res} C_b} \tag{3}$$

then as an approximation we can neglect both L_0 and C_b and connect L_b directly to ground.

Second, if the resonance frequency $\omega_{\rm res}$ also satisfies

$$\omega_{\rm res} L_p \gg \frac{1}{\omega_{\rm res} C_p} \tag{4}$$

then the L_b and C_p can also be neglected and L_p can be directly connected to ground as an approximation.

Note that (3) and (6) actually imply

$$\omega_{\rm res} \gg \frac{1}{\sqrt{L_b C_b}}, \quad \omega_{\rm res} \gg \frac{1}{\sqrt{L_p C_p}}. \tag{5}$$

Under such conditions, the dominant resonance frequency $f_{\rm res}$ of this system is approximately decided by the *LC* tank of C_d, C_c, and L_p, i.e.,

$$\omega_{\rm res} \approx \frac{1}{\sqrt{L_p(C_c + C_d)}}. \tag{6}$$

The resonance frequency for a typical PDN design is on the order of 100 MHz (equivalent to 20 cycles for a 2 GHz CPU), which is far from the frequency range of peak noise, but rather closer to the periodic function execution frequency.

Combine the above discussions, and the voltage response at node V_n can be expressed as

$$V_n(j\omega) = H(j\omega)I(j\omega) \tag{7}$$

$$= H(j\omega)\frac{\sqrt{2\pi}}{T} \sum_{k=-\infty}^{\infty} I_0(jk\Omega)\delta(\omega - k\Omega) \tag{8}$$

where $H(j\omega)$ is the impedance at V_n and (8) is obtained by directly inserting (2) into (7). Since it is a parallel resonance, $H(j\omega)$ approaches infinity at $\omega = \omega_{\rm res}$. Accordingly, when the low frequency components of $I(t)$ are close to the resonance frequency $\omega_{\rm res}$, i.e., when

$$k\Omega = k2\pi/T \approx \omega_{\rm res}, \exists k \in Z \tag{9}$$

$V_n(j\omega)$ will be amplified infinitely from $I_n(j\omega)$ at $\omega = \omega_{\rm res}$. As a result, the voltage fluctuation V_n in time domain will increase significantly.

It is generally believed that inserting decoupling capacitance can reduce noise of a PDN. But this is valid only for high-frequency peak noise reduction as shown below. At high frequency $\omega \gg \omega_{\rm res}$, we have

$$|H(j\omega)| \approx \left|\frac{1}{\omega(C_d + C_c)}\right|. \tag{10}$$

By increasing decoupling capacitance C_d, we can effectively reduce the magnitude of high-frequency response $|H(j\omega)|$, thus reducing peak noise.

In contrast, suppression of resonance noise can be tricky, as it greatly depends on the run-time operation, which affects the low-frequency components of $I(t)$. But it is clear that an effective way to minimize the low-frequency resonance noise would be to change the clock period T directly so that the relation (9) does not hold.

III. Problem Formulation

In order to control the low frequency component of the current load to avoid the resonance frequency, we need to dynamically adjust work load period T. There are two possible ways to apply the adjustment: the first one is to adjust the PLL to change clock frequency; the other one is to adjust the power supply voltage level [11]. Either approach can effectively change the duration of work load, thus achieving different frequency response. As an illustration, we choose to adjust clock frequency directly by employing a programmable PLL design similar to [4], [9].

In the proposed scheme, we employ on-chip current sensors to monitor the peak of the dynamic current load $I(t)$ for each clock domain of interest in the design. Modern VLSI designs usually possess millions of gates, and measuring the current load through each individual gate is impossible. However, the locality of power grid indicates that those current loads flow through nearby power pads, and the total number of power pads is usually limited (10–100). So instead of measuring individual current load, we can measure the "local" sum of current loads through the power pads. If any current load contains frequency component close to the resonance frequency, so will the current through the nearby power pads. Accordingly, it can be detected. An overall schematic of the dynamic power supply current sensor, similar to the one proposed in [5], is shown in Fig. 2. It is composed of a current mirror to sense the current through the power pad, an voltage subtracter to map the current to voltage, and finally a diode and a capacitor to preserve the peak voltage when the clock is high. Accordingly, the sampling from analog-to-digital converter (ADC) can be done at the falling edge of the clock. The capacitor is discharged through the transistor when the clock is low. Note that this is only an illustration of the sensor structure. To make it work, there are many practical issues to be considered, such as the sizing of the transistors in the current mirror to ensure proper scaling and minimum leakage, as well as the value of the capacitor for proper charge and discharge time. However, those considerations are all from the circuit design aspect, and are beyond the scope of this paper.

Based on the sampled $I(t)$ data in history, the *control unit* then predicts the future current load profile, and subsequently determines an optimal clock frequency to be generated by the programmable PLL. This procedure continues such that the low-frequency components of

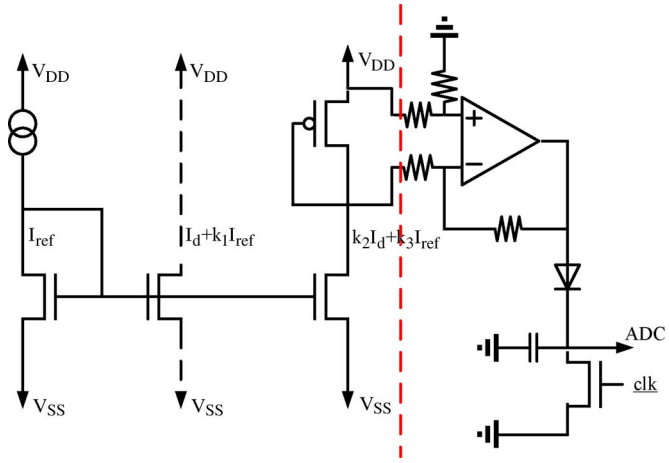

Fig. 2. Schematic of the dynamic current sensor.

the current load $I(t)$ will never get close to the resonance frequency $f_{\rm res}$.

IV. Stochastic Current Prediction and Optimal Frequency Selection

A. Current Prediction Modeling

For a given clock domain of interest, there are n number of current sensors in place to monitor the current load $I(t)$ through the power pads. For the simplicity of presentation and similarity to the study in [8], we represent the current in each clock cycle as a triangular waveform, and each current sensor records either a peak or average current value for this waveform.[2] Such a monitored value for current sensor j at cycle k is denoted as i_k^j; We record all the currents for the same cycle as a vector I_k, i.e.,

$$I_k = \left[i_k^1, i_k^2, \ldots, i_k^n\right]^T \tag{11}$$

where n is the total number of sensors.

To select an optimal frequency, we need to know how the frequency response would change for the incoming work load variations. We propose to use a linear filter as the predictor to predict the future current load. The idea is as follows. Given Q historical current vectors $I_{k-1}, I_{k-2}, \ldots, I_{k-Q}$, the current vector I_k can be predicted as

$$\hat{I}_k = \sum_{i=1}^{Q} \Psi_i I_{k-i} \tag{12}$$

where Ψ_i are n-by-n matrices ($\Psi_i \in R^{n\times n}$) to be determined. For optimal prediction, Q can be set equal to the correlation distance of the current vectors, or can be determined by experiments. As how large Q is directly impacts the efficiency of the prediction, instead of using all historical current vectors, we propose to use the idea of current sampling, i.e., we sample the historical current vectors with spacing L, and obtain M current vectors such that $M \times L = Q$. After sampling, the current vector I_k can be predicted as

$$\hat{I}_k = \sum_{i=1}^{M} \Psi_i I_{k-iL}. \tag{13}$$

[2]Note that the proposed frequency actuator design does not require the triangular waveform assumption.

Our goal is to determine a set of Ψ_i such that (13) is a good predictor for I_k for any randomly selected current vectors in $M \times L$ consecutive clock cycles. This a standard problem in signal processing field, and can be solved by either predetermined linear filter with fixed Ψ_i or adaptive least-mean-square (LMS) filter with changing Ψ_i. For details, please refer to [6].

Accordingly, at any clock cycle k, we can efficiently compute the future L cycles' current vectors $\hat{I}_k, \hat{I}_{k+1}, \ldots, \hat{I}_{k+L-1}$ by using the $M \times L$ history current loads $I_{k-1}, I_{k-2}, \ldots, I_{k-ML}$. Specifically, we have

$$\hat{I}_{k+l} = \sum_{i=1}^{M} \Psi_i I_{k+l-iL}, \quad 0 \le l \le L-1. \tag{14}$$

With the assumptions of triangular waveform and constant rising and falling times, we can directly reconstruct the waveform of any future K ($K \le L$) cycles' current load $u(t)$ from the predicted peak currents as

$$\begin{aligned} u(t) &= \sum_{i=1}^{K} \hat{I}_{k+i} u_\Delta\left(t - (i-1)\times T\right) \\ &= \sum_{i=1}^{K} u_i\left(t - (i-1)\times T\right) \end{aligned} \tag{15}$$

where $u_\Delta(t)$ is the triangular waveform with unit peak current value with the starting time $t = 0$ set at the beginning of the incoming clock cycle k, each $u_i(t) = \hat{I}_{k+i} u_\Delta(t)$ is the triangular waveform with the predicted current value of $\hat{I}_{k+i}$, and T is the clock period.

With (15), the optimal clock frequency can be determined from

$$\arg\min_T \left| \sum_{i=0}^{K} U_i(j\omega_{\rm res}) e^{-ji\omega_{\rm res}T} \right| + \lambda(T - T_{\min}) \tag{16}$$

where the first part is the Fourier transform of (15), and the second part is the penalty for changing clock frequency. Normally, we want the chip to operate at the designated frequency from performance binning, and decreasing the clock frequency can adversely affect the circuit performance such as the delay and the throughput. λ is a positive number reflecting how aggressive the frequency actuator. We find that $\lambda = 4.5$ results in minimum system latency overhead in the presence of resonance noise in the testcases we studied, as will be demonstrated in Section V. $T_{\min}$ is the minimum clock period decided by performance binning.

The problem is an unconstrained nonlinear optimization problem and general optimization techniques such as Newton's method can be applied. Practically, there are only a finite number of discrete clock periods available for any digital-based programmable PLL design, which enables another more efficient way of solving the problem. Denote the finite set of available clock periods as $\{T_1, \ldots, T_m\}$, we can easily find the optimal clock period by evaluating (16) over different T_i and select the optimal one that minimizes the objective function. The efficiency can be further improved by precalculating some of the coefficients in the optimization. For more details, please refer to [6].

The hardware implementation cost for LMS filter can be approximately estimated as follows. the dominant cost is the total number of M matrix-vector multiplication, M vector-vector outer production and M matrix-matrix summation. By using a pipeline technique, only one matrix-vector multiplier, one vector-vector multiplier and one matrix-matrix adder need to be implemented. Given n on-chip sensors, all vectors' length is n and accordingly the total number of floating number multiplication is about $O(n^2)$. If we use S bits to represent any floating number, then approximately $O(S^2)$ gates are necessary to

implement a single multiplication of two floating numbers for combinational logic. Accordingly, $O(n^2S^2)$ gates are necessary to implement the whole algorithm.

For the predetermined linear filter with n sensors and S bits for floating number, the dominant cost is only the matrix-vector multiplication, and accordingly the total gate count is roughly 1/3 of the LMS adaptive filter design. This can be further reduced if we note that the coefficients of the filter are fixed. Each floating number multiplier can be synthesized with only $O(S)$ gates by utilizing LUT's to store those coefficients. Therefore, the total gate count would be on the order of $O(n^2S)$.

V. Performance Estimation

We first verify the accuracy and efficiency of our prediction algorithm with current data from the mobile chip. We apply both the predetermined linear filter and the LMS adaptive filter designs to our frequency actuator and use 32 points ($M = 32$) in history with spacing $L = 400$, and predict the currents in the incoming 400 clock cycles. Experimental results show that adaptive filter has an average prediction error of 1.51%, whereas that of the predetermined linear filter is 13.4%. On the other hand, we observe that the maximum prediction error for adaptive filter can be as large as 311%, indicating the failure of convergence, whereas the predetermined linear filter has a maximum error of 11.6%. To further verify the prediction accuracy, we have also tested the algorithm on a high-performance microprocessor design. The adaptive filter has an average prediction error of 3.76% and a maximum prediction error of 527%. The predetermined linear filter, on the other hand, has an average prediction error of 9.64%, and a maximum prediction error of 17.7%.

In addition, we study how the number of current sensors affects the noise reduction on the mobile chip. Our experiments show that the noise reduction is almost the same when the number of the current sensors is greater than 5% of the total power pad number. In other words, there is no need to place many sensors for the measurement.

Next we conduct experiments for the mobile chip and the microprocessor to illustrate the resonance noise reduction effect. When clock changes, we assume that the spikes in the current profile remain the same, and only the spacing between the spikes scales with the clock rate.[3] From the design, the tracking time for PLL is 75 clock cycles. The clock frequency can be selected from 1.5 to 0.8 GHz with an interval of 0.1 GHz. The retroactive model incrementally reduces the clock frequency by 0.1 GHz until the noise is below the tolerance bound. When the noise is below the tolerance bound, it tries to incrementally increase the clock frequency by 0.1 GHz until the maximum frequency or when noise violation occurs. The proactive model select optimal frequency based on predicted current every 400 clock cycles. We apply simulation with the current profile and the distributed RLC model of the PDN to get the maximum and average voltage droop. Compared with the baseline model without frequency actuator, the retroactive approach can only reduce the max noise by up to 14% and reduce the mean noise by up to 33%. On the other hand, our proactive approach with predetermined linear filter can reduce the max noise by up to 61% and the mean noise by up to 67%, while the proactive approach with the LMS adaptive filter can reduce the max noise by up to 79% and the mean noise by up to 87%.

We also study the impact of λ in (16) on the system latency overhead, and the results are depicted in Fig. 3. From the figure we can see that for both testcases, the latency overhead attains its minimum around $\lambda = 4.5$. This is in accordance with our expectation: A small λ corresponds to an aggressive actuator design, and this can cause unnecessary clock frequency decrease due to prediction error, thus increasing the latency overhead. On the other hand, a large λ corresponds to a conservative actuator design, and this can cause the chip to reboot when the resonance noise occurs and the clock frequency does not change. From the figure, we can also see that when λ is small, the LMS adaptive filter performs worse than the predetermined linear filter in terms of latency overhead. When λ is large, the latency overheads from the two types of filters increase with a similar trend.

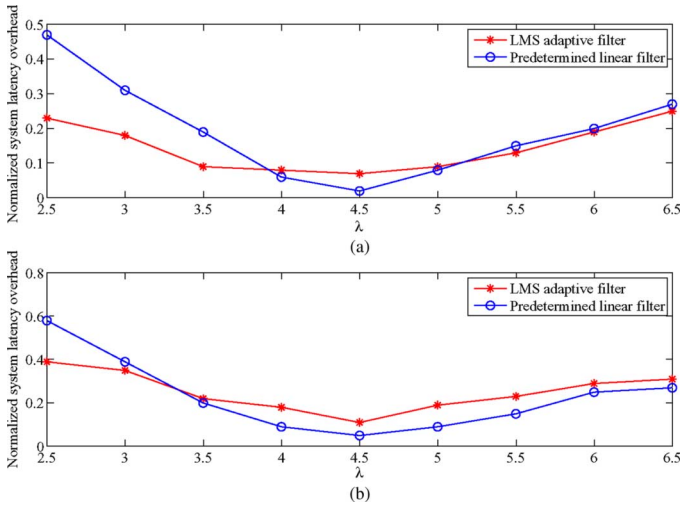

Fig. 3. Normalized system latency overhead w.r.t. the parameter λ for two testcases.

Finally, we implement the LMS adaptive filter and the predetermined linear filter-based designs using digital circuits using Cadence Encounter RTL Compiler. The predetermined linear filter based actuator can cause the gate count to be increased only by 0.02% for the uP design, while the design of the LMS adaptive filter causes the gate count to increase by 0.4%. The area overhead can be reduced significantly if we precalculate and perform table lookup for components in our prediction methods and if we timeshare hardware as the frequency of prediction is much lower than the clock rate. This will be our future work.

VI. Conclusion

Because of the distributed *RLC* characteristics of a PDN, resonance noise at the low-to-middle frequency range significantly affects the reliability of a PDN and chip performance. In contrast to existing retroactive solution that only remedies the noise problem when the noise problem has occurred already, we have proposed a novel design approach to proactively suppress resonance noise. To achieve this goal, we have developed an efficient stochastic current load prediction method based on a generalized Markov process modeling. We have presented a frequency actuator that utilizes both on-chip dynamic current sensors and a programmable PLL for frequency adjustment. Significant resonance noise reduction has been achieved on industrial testcases.

Acknowledgment

The authors would like to thank the anonymous reviewers for their review comments, which helped to improve the quality and presentation of this work significantly.

[3]This is reasonable as the slews of the current loads are solely determined by the sizes of the transistors instead of the clock period.

REFERENCES

[1] M. Ang, R. Salem, and A. Taylor, "An on-chip voltage regulator using switched decoupling capacitors," in *Proc. IEEE Int. Solid State Circuits Conf.*, 2000, pp. 438–439.

[2] K. Bathey and M. Swaminathan, "Resonance analysis and simulation in packages," in *Proc. Elect. Perform. Electron. Packag.*, Oct. 1995, pp. 169–172.

[3] K.-H. Erhard, F. Johannes, and R. Dachauer, "Topology optimization techniques for power/ground networks in VLSI," in *Proc. Euro. Des. Test Conf. (DATE)*, 1992, pp. 362–367.

[4] S. Khadanga, "Synchronous programmable divider design for PLL using 0.18 μm CMOS technology," in *Proc. Int. Workshop Syst.-on-Chip for Real-Time Appl.*, 2003, pp. 281–286.

[5] Y. Lechuga, R. Mozuelos, M. Martinez, and S. Bracho, "Built-in dynamic current sensor for hard-to-detect faults in mixed-signal ICs," in *Proc. Euro. Des. Test Conf. (DATE)*, 2002, pp. 205–211.

[6] Y. Shi, J. Xiong, H. Chen, and L. He, "Stochastic current prediction enabled frequency actuator for runtime resonance noise reduction," in *Proc. Asia South Pacific Des. Autom. Conf. (ASPDAC)*, 2009, pp. 373–378.

[7] Y. Shi, J. Xiong, C. C. Liu, and L. He, "Efficient decoupling capacitance budgeting considering operation and process variations," in *Proc. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, 2008, pp. 1253–1263.

[8] H. Su, S. S. Sapatnekar, and S. R. Nassif, "Optimal decoupling capacitor sizing and placement for standard-cell layout designs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 22, no. 4, pp. 428–436, Apr. 2003.

[9] Y. Sumi, S. Obote, N. Kitai, R. Furuhashi, Y. Matsuda, and Y. Fukui, "PLL frequency synthesizer with an auxiliary programmable divider," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jul. 1999, pp. 532–536.

[10] X. D. Tan and C. J. Shi, "Reliability-constrained area optimization of VLSI power/ground networks via sequence of linear programmings," in *Proc. Des. Autom. Conf. (DAC)*, 1999, pp. 78–83.

[11] A. Waizman and C. Y. Chung, "Resonant free power network design using extended adaptive voltage positioning (EVAP) methodology," *IEEE Trans. Adv. Packag.*, vol. 23, no. 3, pp. 236–244, Aug. 2001.

[12] J. Xu, P. Hazucha, M. Huang, P. Aseron, F. Paillet, G. Schrom, J. Tschanz, C. Zhao, V. De, T. Karnik, and G. Taylor, "On-die supply-resonance suppression using band-limited active damping," in *Proc. IEEE Int. Solid State Circuits Conf.*, 2007, pp. 286–288.

[13] M. Zhao, R. Panda, S. Sundareswaran, S. Yan, and Y. Fu, "A fast on-chip decoupling capacitance budgeting algorithm using macromodeling and linear programming," in *Proc. Des. Autom. Conf. (DAC)*, 2006, pp. 217–222.

High-Speed Algorithms and Architectures for Range Reduction Computation

Francisco J. Jaime, Miguel A. Sánchez, Javier Hormigo, Julio Villalba, and Emilio L. Zapata

***Abstract*—Range reduction is a crucial step for accuracy in trigonometric functions evaluation. This paper shows and compares a set of algorithms for additive range reduction computation and their corresponding application-specific integrated circuit implementations (ensuring an accuracy of one unit in the last place). A word-serial architecture implementation has been used as a reference for clearer comparisons. Besides, a new table-based pipelined architecture for range reduction has also been proposed.**

***Index Terms*—Algorithms, algorithms implemented in hardware, computer arithmetic, cost/performance, elementary function approximation, pipeline.**

I. INTRODUCTION

Range reduction is the first step in the computation of many elementary functions. A poor range reduction in the evaluation of elementary functions can lead to catastrophic accuracy problems when input arguments are large, as reported in [1]. Different software and hardware solutions have been proposed to deal with this problem [2]–[4].

We consider the range reduction achieved via the operator $\bmod^*$ defined as follows:

$$X \bmod^* A = (X - K \cdot A) \tag{1}$$

with X and A being real numbers, $X \geq A$ and K being an integer such as: $K \cdot A \leq X < (K+1)A$.

The design described in [3], based on the double residue modular range reduction (DRMRR), presents a full hardware solution for floating point range reduction. This approach is capable of an accuracy of one unit in the last place (ULP) for any input number according to IEEE 754 simple precision floating point representation.

Let X be a positive floating point number such as

$$X = M \times 2^E = \left(\sum_{i=0}^{n-1} x_i 2^{-n+1+i} \right) \times 2^E \tag{2}$$

where E is the exponent within the floating point representation and M is the input vector X mantissa, which has a length of n bits: $M = (x_{n-1} \cdot x_{n-2} \cdots x_1 x_0)$ (x_{n-1} is considered as the hidden bit within the floating point representation, thus it is treated as a normal bit by the following algorithms).

According to this representation, DRMRR algorithm computes the following equation for every bit:

$$R(i) = \begin{cases} R(i-1) + x_i \cdot m_i^+, & \text{if } R(i-1) < 0 \\ R(i-1) + x_i \cdot m_i^-, & \text{if } R(i-1) \geq 0 \end{cases} \tag{3}$$

where $R(i)$ is the accumulated value through all the additions, $R(-1) = 0$ and (m_i^+, m_i^-) are the elementary residues defined as $m_i^+ = 2^{i-n+1+E} \bmod^* A$ and $m_i^- = (2^{i-n+1+E} \bmod^* A) - A$.

Manuscript received April 15, 2009; revised July 24, 2009 and September 21, 2009. First published November 03, 2009; current version published February 24, 2011. This work was supported in part by the Ministry of Education and Science of Spain under Contract TIN2006-01078, by the FSE (Fondo Social Europeo), and by the Junta of Andalucía under the Project P07-TIC-02630.

The authors are with the Computers Architecture Department, Málaga University, Málaga 29071, Spain.

Digital Object Identifier 10.1109/TVLSI.2009.2033932