A Design Flow for the Precise Identification of the Worst-Case Voltage Drop in Power Grid Analyses

D.P. Karampatzakis, M.K. Tsiampas, N.E. Evmorfopoulos, G.I. Stamoulis
University of Thessaly
Department of Computer and Communication Engineering
Volos, 38221, Greece
ofni@uth.gr, mitsiaba@uth.gr, nestevmo@uth.gr, georges@uth.gr

***Abstract*—Modern IC designs contain hundreds of millions of transistors and new implementations of multi core chips take place in commercial products. Identifying worst-case voltage drop conditions in every hierarchical module supplied by the power grid is a crucial reliability problem in modern IC design. In this paper we focused our efforts on a complete design flow based on innovative results from recent research work. This approach demonstrates a new implementation of construction of the current space which is performed via plain simulation and statistical extrapolation using results from extreme value theory. Experimental results verify the potential of the estimation engine within an industrial EDA flow for performing power grid verification using a custom hierarchical design.**

I. INTRODUCTION

Implementing modern IC designs became a complex project and design flows based on modern EDA tools trying to solve many of new challenges and to overcome new technologies problems. The reduction of the voltage level supplied on the active blocks or modules by the lines of the power distribution network (voltage-drop or IR-drop) constitutes one of the biggest reliability problems in modern nanometer-scale VLSI circuits as it adversely affects circuit speed and noise margins [1]-[2].

Upcoming generations of ICs are going to experience substantial voltage drops (due to increased currents and parasitic), which, combined with the reduced supply levels and increased drop-to-supply ratios, will make the situation extremely harsh. To overcome these problems designers are focusing on checking if the designed power grid is *robust*, i.e. if it constantly maintains a safe voltage level at all active modules under all possible loading conditions.

Voltage drop is a dynamic phenomenon, its value at any time instant being dependent on the transient current waveforms drawn over time by the active modules acting as current sinks. Thus, most existing methodologies complement dynamic analyses with some kind of static (DC) analysis in which one vector (or maybe a collection of vectors) of static values for the sink currents is used to represent the transient current waveforms over all input patterns [1]-[2]. Apart from its comprehensive coverage quality, the static analysis is also simpler to implement in terms of modeling and solution of the power grid, since only the resistive model of the latter needs to be extracted and utilized for the calculation of voltage drop. Two obvious choices for the static currents are the average and maximum values of the transient waveforms over all input patterns. Average current values are relatively easy to estimate but may clearly overlook the worst case and miss conditions for which the grid is essentially non-robust. On the other hand maximum currents do represent an upper bound on voltage drop at all sinks, and hence can guarantee grid robustness under all possible input conditions. Estimating maximum sink currents over all input patterns is a challenging problem by itself, and has been addressed in the past by some researchers [3]-[4], although more recent techniques for overall maximum power/current estimation of the circuit based on simulation and statistical extrapolation [5]-[8] are more appropriate and accurate for the estimation of maximum current at each sink.

In this paper we implement a complete design flow in order to verify the voltage drop on a power grid of a modern digital design. In order to encapsulate our novel methodology for deriving a collection of static current vectors that provide a realistic worst-case voltage drop (not just a loose upper bound) at each sink over all input patterns, and also calculate this worst-case voltage drop [13], we implement a new version of IXTREME engine. The methodology is based on constructing an accurate profile of the space (or locus) of simultaneous sink current variations and locating the specific points that yield the worst-case voltage drop at each sink. The power of our verification engine is supported with automated simulation engines for design simulation and circuit analyses.

The rest of the paper is organized as follows. The next section provides the necessary background for conditions on sink currents that lead to the worst-case voltage drops and demonstrates the methodology. Section III shows the main parts of the design flow. Section IV presents our experimental results comprising of test grids for a custom hierarchical design, and finally section V gives the overall concluding remarks.

II. PROBLEM FORMULATION AND A NOVEL METHODOLOGY

The model that we will assume for the power grid is the resistive linear model with time-varying current sources in place of sinks, since we are seeking static current vectors for DC analysis. Instead, however, of using one single DC vector (such as the vector of overall average or maximum currents) to represent all transient waveforms for all possible input patterns, we will adopt a *multi-cycle* DC current scheme [1] in which only the cycle-accurate current waveform for each sink (corresponding to one specific input pattern) is substituted by a constant DC value (Fig. 1). Based on this scheme we will subsequently attempt to find those cycle-DC current vectors that provide the worst-case voltage drop at any sink for the given power grid. This particular scheme is able to account for the diverse current workloads which arise for dissimilar input patterns (due to the different number of devices being switched), and can even faithfully reproduce the transient behavior of the grid on condition that there exists sufficient decoupling capacitance to smooth out large dynamic peaks of voltage drop during the period of a cycle (especially right after a clock edge

This work is supported by GSRT- Greek Ministry of Development

978-0-7695-3323-0/08 $25.00 © 2008 IEEE
DOI 10.1109/PCI.2008.25

when multiple devices switch simultaneously). Of course the latter may not be entirely true in modern ICs with operating frequencies above 1GHz, and that is why sample dynamic analyses may still have to be run. However those analyses can never cover all input pattern combinations and thus do not overshadow the value of a static analysis, especially one that is based on the considerably more accurate multi-cycle DC current scheme.

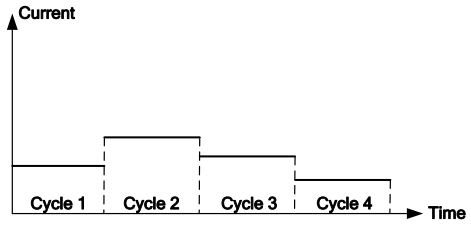

Figure 1. Multi-cycle DC current scheme.

We assume that the power grid (or the portion of the power grid that needs verification) has been extracted as a linear network of resistive branches that meet at $q+p$ nodes, of which p nodes are connected to the external power supply via power pads (which are either located at the periphery of the grid in the case of a wire-bond package, or are scattered across the entire grid area in a C4 methodology), and the remaining q nodes are divided to n sink nodes (with current sources to an external ground node) and $q-n$ internal nodes (for which usually $q >> n$). According to the *modified nodal analysis* (MNA) [12], the $q \times 1$ vector of instantaneous voltages $\underline{U}(t)$ (relative to ground) at all internal and sink nodes is determined by the instantaneous currents drawn by the sinks and the specific network structure, and is given by the following matrix equation:

$$\underline{G} \cdot \underline{U}(t) = -\underline{I}(t) + \underline{G} \cdot \underline{V}_{dd} \quad (1)$$

where $\underline{G}$ is the $q \times q$ *conductance* matrix of the network (formed by the conductance of the network branches), $\underline{I}(t)$ is a $q \times 1$ vector of current excitations at the nodes (with positive currents to the ground at sink nodes and zero entries at all internal nodes), and $\underline{V}_{dd}$ is another $q \times 1$ vector with all entries equal to the supply voltage V_{dd}. By defining $\underline{V}(t) = \underline{V}_{dd} - \underline{U}(t)$ as the voltage drop at all nodes we can rewrite the above network equation in a form which can be solved directly for the voltage drop values:

$$\underline{G} \cdot \underline{V}(t) = \underline{I}(t) \quad (2)$$

In order to obtain appropriate expressions for the voltage drops at the n sink nodes which are of interest, we must enumerate them first in the matrix $\underline{G}$ and vectors $\underline{I}(t)$ and $\underline{V}(t)$, and then we can write eq. (2) as follows:

$$\left[\begin{array}{c|c} \underline{G}_{11} & \underline{G}_{12} \\ \hline \underline{G}_{21} & \underline{G}_{22} \end{array}\right] \cdot \left[\begin{array}{c} \underline{V}_s(t) \\ \hline \underline{V}_i(t) \end{array}\right] = \left[\begin{array}{c} \underline{I}_s(t) \\ \hline \underline{0} \end{array}\right] \quad (3)$$

where $\underline{G}_{11}$, $\underline{G}_{12}$, $\underline{G}_{21}$, and $\underline{G}_{22}$ are submatrices of $\underline{G}$ with sizes $n \times n$, $n \times (q-n)$, $(q-n) \times n$, and $(q-n) \times (q-n)$ respectively, while $\underline{V}_s(t)$ and $\underline{I}_s(t)$ are vectors of size $n \times 1$ (representing the voltage drops and current excitations at the sink nodes respectively) and $\underline{V}_i(t)$ is a vector of size $(q-n) \times 1$ (representing the voltage drops at the internal nodes). The latter equation can be solved with respect to $\underline{V}_s(t)$ and after some calculations gives:

$$\underline{V}_s(t) = (\underline{G}_{11} - \underline{G}_{12}\underline{G}_{22}^{-1}\underline{G}_{21})^{-1} \cdot \underline{I}_s(t) = \underline{R} \cdot \underline{I}_s(t) \quad (4)$$

where the matrix $\underline{R} = (\underline{G}_{11} - \underline{G}_{12}\underline{G}_{22}^{-1}\underline{G}_{21})^{-1}$ is of size $n \times n$ and consists only of non-negative values since $\underline{G}$ is an M-matrix. The process of power grid verification typically involves checking that the voltage drop at all sink nodes does not exceed a safety threshold voltage V_0 (e.g. $V_0 = 0.1\,V_{dd}$) at all time instants *t*, i.e. $\underline{V}_s(t) < V_0$, $\forall t \in \Re$. Since the latter is equivalent to $\max_{t \in R} \underline{V}_s(t) < V_0$ (where the "max" operator is interpreted *component-wise* in vector $\underline{V}_s(t)$), we need to find the maximum voltage drop $\max_{t \in R} V_k(t)$ at each sink $1 \le k \le n$. Each component $V_k(t)$ in vector $\underline{V}_s(t)$ would be as follows:

$$V_k(t) = r_{k1} I_1(t) + r_{k2} I_2(t) + \cdots + r_{kn} I_n(t) = \underline{r}_k^T \cdot \underline{I}_s(t) \quad (5)$$

Thus the verification problem is concerned about maximizing the function $V_k(t)$ given by the above equation. Note that the standard practice of taking maximum current values at each sink effectively executes:

$$\max_{t \in \Re} V_k(t) = \max_{t \in \Re} \left[r_{k1} I_1(t) + r_{k2} I_2(t) + \cdots + r_{kn} I_n(t) \right]$$
$$\le r_{k1} \max_{t \in \Re} I_1(t) + r_{k2} \max_{t \in \Re} I_2(t) + \cdots + r_{kn} \max_{t \in \Re} I_n(t) = \underline{r}_k^T \cdot \max_{t \in \Re} \underline{I}_s(t)$$

which obviously leads to a very conservative upper bound.

In our recent work [13] we develop a novel methodology for deriving a collection of static current vectors that provide a realistic worst-case voltage drop (not just a loose upper bound) at each sink over all input patterns, and also calculate this worst-case voltage drop. The methodology is based on constructing an accurate profile of the space (or locus) of simultaneous sink current variations and locating the specific points that yield the worst-case voltage drop at each sink. Of course in order to find the space of current variations in every detail one has to simulate the circuit for all input patterns which brings about the same obstacle. However, a scaled image of the current space can be obtained via a sample of vectors of sink currents, which can be subsequently extrapolated to the entire space by using results from *extreme value theory* (EVT) [9]. Some past approaches [10]-[11] tried to abstract the current space by prompting the user to enter constraints that express relationships between sink currents.

The problem is that the user cannot determine with much accuracy relationships between different sinks and formulate them as constraints, and even if this could be done, it is not possible to capture all different relationships and interdependencies in the form of a small number of constraints. A significant part of the aforementioned constraints will surely be missed and the calculated worst-case voltage drops (which only reflect the constraints externally provided) will fall short of the observed reality. On the other hand, it is certainly much more accurate and easier for the designer to simulate the design

for a set of input patterns, and construct the current space exclusively out of the simulation results. In addition, constraints in the previous approaches were in the form of vague upper bounds and thus do not generate the actual current space, but only a superset of it which is still much pessimistic. Our approach *accurately constructs* the current space (or, specifically, its portion containing the worst-case points for voltage drop) without enclosing it into bounds and accounting for *all* potential correlations between sink currents.

Our novel methodology follows a process of creation of the sample space (by circuit simulation), univariate EVT estimation in each of the coordinate axes, and shifting of the maximal points of the sample space (to meet the maximal points of the entire space) is *independent* of the supplying power grid and needs to be carried out only once. The main steps in this process are summarized hereafter along with some brief remarks on their implementation and computational complexity:

- *Generate a total of* $m = 5000$ *random pairs of binary input vectors* $\{v_p, v_n\}$ *for the circuit under consideration*. This step can be performed by any standard random number generator producing uniform numbers (a testbench generator). The selection $m = 5000$ for the number of input pairs is discussed below.
- *Simulate the circuit under all generated pairs and record peak current* $\max_{t\in[0,T]} I_k(t)$ *in each sink within a clock cycle*. The computational time required to complete this step is entirely up to the simulator program employed, since there are many different simulators with speeds that range considerably depending on the detail of the analysis and their algorithmic efficiency.
- *Determine the maxima* $\max_{1\le i\le m} I_{k,i}$ *of all univariate samples* $\bar{I}_k$ *and in conjunction with the estimates* $\hat{\omega}_k$ *(results from EVT analyses) compute the n-dimensional difference vector*.
- *Locate the maximal points of the sample space* $\bar{I}_s$. As already mentioned, this step has complexity of $O\left(m(\log_2 m)^{n-2}\right)$ comparisons. The resulting number m_c of maximal points is typically much smaller than the $m = 5000$ points of the sample space.
- *Shift the maximal points of the sample space* $\bar{I}_s$ *by the computed difference vector*. This step is performed by plain component-wise addition of the difference vector to the m_c maximal points of $\bar{I}_s$, and is a trivial one.

The output of all the above steps is a set of shifted sample maximal points for a particular circuit which approximate the position of the maximal front of its entire current space. Then the DC verification of any grid supplying the circuit is performed by the following steps:

- *Apply the* m_c *shifted maximal points as static current vectors to perform an equal number of DC analyses for the given power grid*. This step relies exclusively on a linear network solver, and its execution time is determined by the capability of the solver to carry out m_c DC analyses for the given grid. In our approach we provide also the choice of a convex hull analyses and it is possible to produce a smaller set of maximal points and to reduce the number of DC analyses.
- *For each sink determine the maximum value among the computed DC voltage drops*. The resulting value for each sink finally constitutes the worst-case voltage drop over all input pairs.

The methodology presented above is implemented in our new IXTREME2 estimation engine. The power of this engine is combined with industrial EDA tools and is a part of a complete design flow which is presented in the next section.

III. A DESIGN FLOW FOR POWER GRID VERIFICATION

Encapsulating the new methodology in an industrial design flow is a challenging work and is a result of a combination of different EDA tools. The design flow that we propose is based in common elements of a flow and follows all the types of a typical digital design flow. It is based on industrial tools of the companies, SYNOPSYS and CADENCE, as well as on the programming languages TCL, C and C++. The Fig.2 shows the three main stages of the proposed flow, the design capture – the fast spice simulation and the verification.

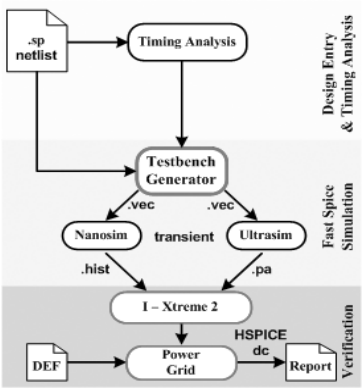

Figure 2. A general overview of the proposed verification Power Grid flow.

In the next paragraphs we present more analytically the three main stages in order to expose all the implementation parts of the novel methodology. The first stage is the design entry and timing constraints generation. For the implementation of this stage, we have developed one procedure and we incorporated one industrial tool. The procedure is the uth_parser.tcl program, which is written in TCL scripting language and its purpose is to create two files. The first file is a "spice" file (.sp, .spi) which contains the circuit we work up, the second file is a configuration file for the timing analysis tool. The two files yielded from the process, constitute the input files of the Pathmill™ static timing analysis tool, which is used in order to execute the timing analysis of the circuit. Both the report file (pathmill.out) of the Pathmill™ and the design-file are being used by the second stage of the design flow.

The second stage of the design flow, executes the transient analysis of the circuit. In order to implement this stage of our

design flow, we incorporated in the flow, two industrial tools and implemented two procedures. In our attempt to make the flow more independent, we used industrial tools for transient analysis, from two different companies. We incorporated Ultrasim™, which is Cadence's transient simulator and Nanosim™ which is the equivalent tool of Synopsys. The first procedure, is the get_time.tcl program, which is also a program written in TCL scripting language. The get_time program, since timing analysis terminated with no errors, reads the report file of Pathmill™, finds the maximum path delay of the circuit passes it to the second process. The second process is the program spice_verilog.tcl, written in TCL scripting language. Purpose of this program is the generation of the required input files, with the appropriate format, for the two transient simulators. For the accomplishment of its purpose the program takes eight inputs and generates four different outputs. Two of the outputs are the configuration files (.cfg) for the two different simulators. The third output is a file with pseudo-random input vectors (.vec) for the circuit. The program may generate vector-files having the vectors either in binary format or in hexadecimal format, which is a more compressed representation of a bit-vector and leads us to smaller sizes of the generated files. The last output file, is the initial circuit design, in SPICE format, with the appropriate cards added by the process, depending on the simulator the user will choose to execute the transient simulation.

The first input of the program is a parameter which determines the simulator will be used for the current transient simulation. Two of the input files provide the configuration data (.cfg) to the program in order to generate the final configuration files for the two simulators. The forth input of the program is a file (.txt) containing the parameters needed for the generation of the vector-files. The fifth input of the program is an integer, determining the total number of the pseudo-randomly generated vectors for the current simulation. The sixth parameter of the program is a file either in Verilog or in SPICE format, containing the uppermost inputs of the design. The seventh input of the program is the output file, of the previously mentioned program get_time.tcl in the same stage of the flow, the file time.uth. The last input parameter of the program is the "spice" file with the design of the circuit. Since the input files for the two simulators have been created with no errors the user chooses either Ultrasim™ or Nanosim™ and passes the write input files to the simulator and executes the simulation. Each simulator generates a report file (".hist" for the Nanosim™ and ".pa" for the Ultrasim™) with the results of the simulation. The data from the report file that concern us are the maximum and the average currents the circuit consumes. These report files are passed to the next stage of the design flow.

The third stage of the design flow includes the mathematical procedures of our design flow. In this stage of our flow we have incorporated two mathematical procedure and two industrial tools. We have also implemented the new version of the statistical estimation engine of the flow and one procedure for processing the layout of the power grid of the circuit. The procedure IEXTREME2 is the engine of the flow. Its two inputs are the report file of the simulator and a parameter that determines which type of currents (maximum or average) the engine will process. The IXTREME2 engine processes the output currents of the simulator based on the EVT theory described above. The output of the engine is a file (Estimated_currents.txt) with the estimated currents according to the EVT. These currents are passed to the second mathematical procedure, of the flow as an input. The second procedure is a program written in C programming language and processes the estimated currents based on Convex Hull theory (make the appropriate calls to the qhull Linux based program). The output of the process is a file with the initial population of input currents decreased.

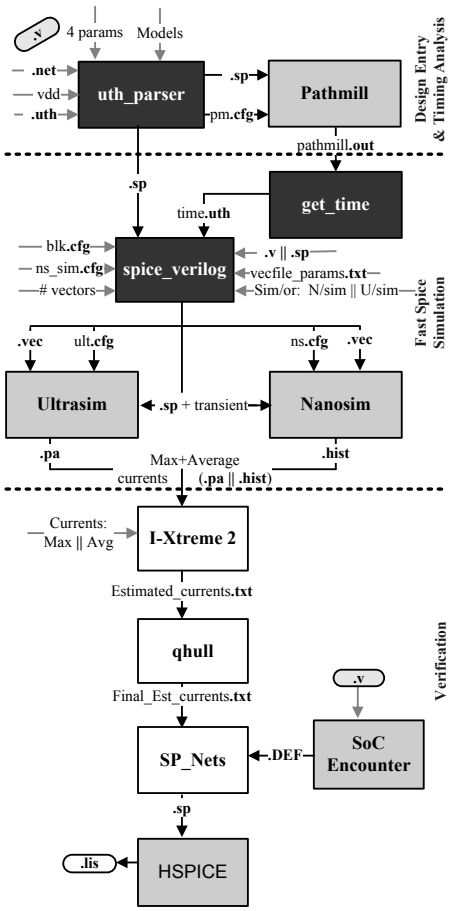

Figure 3. The complete flow for Power Grid verification.

At the same time the two mathematical procedures process the currents, the industrial tool SoCEncounter reads the design and generates the layout of the circuit's power grid. The output of the SoCEncounter and the output of the qhull passed as inputs of the procedure SP_Nets. The SP_Nets procedure is a program written in C++ programming language. The program extracts the resistant from the conductors of the power grid and generates a SPICE file which describes the grid with resistances and models the blocks of the circuit with current sources. The program creates a template with the sets of currents the sources draw. The output file of the previously described process passes as an input to the industrial tool HSPICE™. The tool simulates the circuit that models the power grid and generates a report file (.lis) with the voltage drop on the power grid at the top of every block.

The proposed design flow is a Linux based solution and supports all the proper industrial formats and standards. The fine combination of the EDA tools, mathematical engines and the IXTREME2 engine implements a useful design flow for power grid verification. Finally, the design flow exposes the power of the IXTREME2 featuring the benefits of a novel methodology which will be a demand in the near future for designing robust nanometer-scale ICs.

IV. EXPERIMENTAL RESULTS

The circuits selected for the experimental validation of the design flow is a hierarchical design implementation of a SAD block, part of the H.264 video hardware implementation [14]. The design is implemented in 90nm process technology and the cells are taken from the 90nm GSCLib3.0 Generic Standard Cell Library. The power grid was implemented in process with 9 copper metal layers, sheet resistance R_{sh}= 0.02 ohm/square and supply voltage V_{dd}=1.2V. The design consists from 10 sub-blocks, each occupying area of 50.000 um^2. The estimation of maximum currents at each block was performed first and qhull procedure follows (The presented circuit has 30 x 10 current sets and the data table is not fit on the page to be presented). For the analyses of the power grid we placed the voltage pins using a random process (C4 technology) and we present results for different power grid designs (different number of nodes – 3x3, 4x4, 5x5, 10x10).

Table 2. Voltage drops at each block for different power grid designs.

Design	Pin Voltage (volts)	Grid Dimension			
		3x3	4x4	5x5	10x10
Blocks (B_i)	B_1	1.04	1.14	946.5m	1.080
	B_2	1.03	1.01	1.13	1.091
	B_3	995.7m	1.07	916.8m	1.14
	B_4	1.17	1.07	1.13	1.09
	B_5	1.10	1.15	1.01	1.15
	B_6	1.09	1.10	1.16	1.17
	B_7	1.2	1.14	918.8m	1.09
	B_8	995.7m	1.13	924.2m	1.14
	B_9	1.20	984.1m	850.3m	1.08
	B_{10}	1.09	1.01	1.12	1.08

V. CONCLUSION

A design flow for power grid verification has been developed, which relies on accurate construction of the current space by simulation and statistical extrapolation using results from extreme value theory. The flow is based on industrial EDA tools and exposes the accuracy and flexibility of IXTREME2 Engine. The flow has interfaces using industrial design exchange and report formats. The proposed flow may be used in conjunction with a power grid routing tool in order to prevent grid overdesign and point towards efficient use of routing resources, which will constitute an essential design need for the nanometer-scale generation of VLSI circuits.

REFERENCES

[1] A. Dharchoudhury, R. Panda, D. Blaauw, R. Vaidyanathan, B. Tutuianu, and D. Bearden, "Design and analysis of power distribution networks in PowerPC microprocessors", *ACM/IEEE Design Automation Conf.*, 1998.

[2] G. Steele, D. Overhauser, S. Rochel, and S. Hussain, "Full-chip verification methods for DSM power distribution systems", *ACM/IEEE Design Automation Conf.*, 1998.

[3] S. Chowdhury and J. Barkatullah, "Estimation of maximum currents in MOS IC logic circuits", *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 642-654, 1990.

[4] H. Kriplani, F. Najm, and I. Hajj, "Pattern independent maximum current estimation in power and ground buses of CMOS VLSI circuits: algorithms, signal correlations and their resolution", *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 998-1012, 1995.

[5] A. Hill, C. Teng, and S. Kang, "Simulation-based maximum power estimation", *IEEE Int. Symp. Circuits and Systems*, 1996.

[6] C. Ding, Q. Wu, C. Hsieh, and M. Pedram, "Statistical estimation of the cumulative distribution function for power dissipation in VLSI circuits", *ACM/IEEE Design Automation Conf.*, 1997.

[7] Q. Wu, Q. Qiu, and M. Pedram, "Estimation of peak power dissipation in VLSI circuits using the limiting distributions of extreme order statistics", *IEEE Trans. Computer-Aided Design*, vol. 20, pp. 942-956, 2001.

[8] N. Evmorfopoulos, G. Stamoulis, and J. Avaritsiotis, "A Monte Carlo approach for maximum power estimation based on extreme value theory", *IEEE Trans. Computer-Aided Design*, vol. 21, pp. 415-432, 2002.

[9] J. Galambos, *The Asymptotic Theory of Extreme Order Statistics, 2nd ed.*, Krieger, 1987

[10] D. Kouroussis and F. Najm, "A static pattern-independent technique for power grid voltage integrity verification", *ACM/IEEE Design Automation Conf.*, 2003.

[11] H. Qian, S. Nassif, and S. Sapatnekar, "Early-stage power grid analysis for uncertain working modes", *ACM/IEEE Int. Symp. Physical Design*, 2004.

[12] L. Pillage, R. Rohrer, and C. Visweswariah, *Electronic Circuit and System Simulation Methods*, McGraw-Hill, 1995.

[13] N. Evmorfopoulos, D. Karampatzakis and G. Stamoulis, "Precise Identification of the Worst-Case Voltage Drop Conditions in Power Grid Verification". *Proc. Int'l Conf. Computer Aided Design*, Nov. 2004, pp. 479-484.

[14] M.G. Koziri, G.I. Stamoulis, I.X. Katsavounidis, "A Low-Power VLSI architecture for Intra and Inter prediction in H.264", *IEEE PRIME Conf.*, June 2006, pp. 109 – 112.