A Case for Computer Architecture Performance Metrics that Reflect Process Variability

Bogdan F. Romanescu, Michael E. Bauer, Daniel J. Sorin, and Sule Ozev
{bfr2, meb26, sorin, sule}@ee.duke.edu
Department of Electrical and Computer Engineering
Duke University

I. Introduction

As computer architects, we frequently analyze the performance of systems, and we have developed well-understood metrics for reporting and comparing system performances. The dominant textbook in our field [7] is subtitled "A Quantitative Approach" and it repeatedly emphasizes the need for quantitative performance metrics that accurately reflect actual performance rather than just aspects of performance. Students are taught to report how long it takes a processor to run a benchmark rather than just the processor's clock rate or the instructions per cycle (IPC) it achieves on a benchmark, both of which present incomplete pictures of performance.

Architects now face an issue, increasing *process variability* [5, 10], that requires us to add a new aspect to performance metrics. As transistor and wire dimensions continue to shrink, the variability in these dimensions—across chips and within a given chip—has a greater impact. Process variability complicates system design by introducing uncertainty about how a fabricated processor will perform. Although we design a processor to run at a nominal (mean or expected) clock frequency, the fabricated implementation may stray far from this expected performance. Some amount of process variability has always existed, and we have traditionally coped with it by designing for the mean performance and then "speed binning" the fabricated chips. Comparisons between designs have also been made based on mean performances. For small amounts of variability, considering only mean performance is a suitable approach. However, as variability increases, it might be wiser to design and compare processors based on more than simply the mean.

Consider the example probability distribution functions (PDFs) shown in Figure 1. These Normal (Gaussian) PDFs represent the online transaction processing (OLTP) performances of two hypothetical system designs, P1 and P2, as measured in TPC-C transactions per minute (denoted by tpmC). Both system designs have the same mean performance of 1 million tpmC. Thus, if we only consider mean performance, which is the only metric used currently, the system designs are equivalent. However, inspecting their performance distributions reveals significant differences that could affect their relative *utilities*. Utility can be defined in many ways. One possible definition of design utility, U_{design}, is how much money we can make by selling a large (statistically significant) number of systems with this design. To determine U_{design}, we need to know the utility of each possible fabricated chip as a function of performance, $U_{chip}(p)$, and then we can integrate over all possible chip performances. If we denote the performance PDF as $f(p)$, we have:

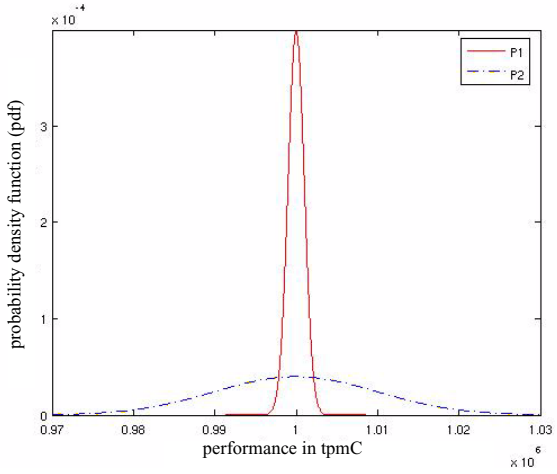


Fig. 1. Two Example Performance Distributions

$$U_{design} = \int_0^\infty U_{chip}(p) \times f(p) dp$$

A simplistic chip utility function would assume that a chip's utility is linearly proportional to its performance, i.e., $U_{chip}(p) = p$. In this particular scenario, we have:

$$U_{design} = \int_0^\infty p \times f(p) dp = E[p]$$

This equation for U_{design} is equivalent to the expected value (mean) of p, $E[p]$. Thus, for this exact utility function,

looking solely at the mean would have been sufficient. However, this equivalence is only true for this simplistic and unlikely chip utility function. This chip utility function is unlikely because, for example, a chip on the very low end of the performance distribution (e.g., at 400K tpmC) might have approximately no value (i.e., nobody would buy it). Similarly, a chip on the very high end of the distribution could command a super-linear premium, as is typical for today's high-end chips. If one shops for a computer and compares the prices of functionally equivalent processors with different clock frequencies, one will observe this same highly non-linear phenomenon.

Returning to our example in Figure 1, we observe that our preference for a particular design depends heavily on the particular chip utility function. For example, if we were to fabricate large volumes of both P1 and P2, we would expect to have more P2 chips that are on the low end of the distribution. If the chip utility function gives relatively little benefit to chips on the low end of the distribution, then P1 will be favorable.

The goal of this paper is to convince architects that they must consider more than mean performance when comparing and designing systems. Using the system utility metric will enable them to tune performance in a way that maximizes utility rather than simply mean performance. In the rest of this paper, we first discuss low-level process variability and how it impacts high-level performance variability, including quantitative experimental data (Section II). We then describe utility functions in more detail and use hypothetical utility functions to illustrate their impact on performance analysis (Section III). We then use commercial pricing data to approximate real-world utility functions for high-end and embedded processors (Section IV). Lastly, we draw conclusions from this analysis (Section V).

II. Process Variability: Causes, Effects, Analysis, and a Case Study

Causes. Process variability arises due to several specific causes, but the over-arching cause is the inability to perform VLSI fabrication with every feature *exactly* as planned. The design might specify that a transistor is 130nm long, but, due to fabrication imperfections, some transistors may be somewhat shorter or longer. Some sources of variability are approximately constant within a given die (or wafer) but vary from die to die (D2D) or wafer to wafer (W2W). The variability within-die (WID) has a systematic component, due to physical phenomena, and it causes spatial correlations between nearby transistors. For example, if the dopant density is a little greater than nominal for a given transistor, it is likely that nearby transistors will also have greater than nominal dopant density. Other sources of WID variability are random and thus cause no spatial correlations.

Effects. For older CMOS technologies, a small amount of variability could be ignored; a variability of 1nm for a 250nm feature had negligible impact. However, when we consider newer technologies and, in particular, future technologies, the same absolute variability of 1nm becomes a considerable fraction of a device's length or width. In addition to affecting a transistor's length and width, process variability also has a non-trivial impact on a transistor's gate oxide thickness and threshold voltage. These four low-level parameters—*L, W,* t_{ox}*, and* V_{t0}—are generally considered to be the most sensitive to process variability [12].

Low-level process variability impacts the behavior of individual transistors and wires, although we focus on transistors in this paper. Due to a longer length, a given transistor may switch more slowly than expected. Due to a wider channel, a transistor might switch more quickly, and it may also present a larger input capacitance to upstream transistors that are driving it. There are a vast number of such effects at the transistor-level, and they manifest themselves as performance variability at the gate-level and at the system-level.

Analysis. Analyzing system performance in the presence of process variability is a difficult challenge. However, within the past few years, there has been a large amount of research to develop CAD tools that can perform what is often referred to as *statistical static timing analysis (SSTA)*. SSTA tools produce statistical performance results, such as the full PDF of a circuit's delay or a circuit's mean delay and the standard deviation of this delay. SSTA tools have been developed in industry (Magma DA's Quartz SSTA, Extreme DA's Extreme XT, and Synopsys's PrimeTime VX) and in academia [3, 4, 6, 11, 13, 14, 15], and the different tools make different tradeoffs in terms of accuracy and analysis run-time.

Because none of the SSTA tools listed above are publicly available yet, our research group has developed its own SSTA tool that computes the mean and standard deviation of the delay through a given path in a circuit. The details of this SPICE-based model are beyond the scope of this paper, but we provide a short high-level overview of its features. The model considers uncorrelated process variability in L, W, t_{ox}, and V_{t0}, for a 130nm process (parameters are in Table I), and it assumes that the low-level variability adheres to a Normal distribution. By analyzing all paths that could possibly be critical in the presence of variability, the tool can analytically compose these results and compute the mean and standard deviation of the entire circuit's delay. We validated the model's accuracy by comparing it to full Monte Carlo analysis on the well-known ISCAS benchmark circuits.

Motivational Case Study. As a simple example, we analyze the integer ALU in the open-source Illinois Verilog Model (IVM), which was developed and generously distributed by Prof. Sanjay Patel's research group at the University of Illinois [8]. The mean delay of this ALU is approximately 2ns. To understand the impact of variability, we look at 6σ/mean,

TABLE I. Process Parameters

parameter	mean	variance
L	160nm	15%
W (PMOS)	550 nm	4.4%
W (NMOS)	250 nm	9.6%
T_{ox}	3.3 nm	10%
V_{t0} ***(PMOS)***	-0.3169 V	10%
V_{t0} ***(NMOS)***	0.365 V	10%

where σ represents the standard deviation. We include the 6σ term because it represents +/- 3σ, and 3σ is often used as a measure of variability. We divide by the mean to normalize the results. For the integer ALU, the normalized variability is approximately 9%. Considering that we assumed relatively optimistic values for low-level variances, this simple example motivates the need to consider process variability now and even more in the future.

III. The Impact of Different Chip Utility Functions

Given that variability is an increasing issue and given that we need performance metrics that incorporate this phenomenon, we now discuss some hypothetical, yet intuitive, chip utility functions, and we explain how designers can use them to focus their efforts towards the most profitable areas for optimization.

In Section I, we presented a simplistic chip utility function: $U_{chip}(p) = p$. This utility function misses two important features. First, it assumes a continuous function, which is unlikely given the current procedure of "speed binning" fabricated chips based on their speeds (clock frequencies). Companies tend to sell chips at quantized performance levels (where performance for identical designs can be measured by clock frequency), such as 3 GHz, 3.3 GHz, etc., rather than in a continuous spectrum. When shopping for a laptop, we are given a small number of frequency options for a given chip. Thus, a chip utility function is unlikely to be continuous. Rather, we might expect it to look something like the utility functions in Figure 2. In the figure, we have performance on the x-axis, rather than clock frequency, so that we can keep the discussion general and enable comparisons of non-identical designs.

The second shortcoming of the simplistic utility function from Section I, as mentioned before, is that it is linear, and a chip's value is unlikely to be strictly linear (e.g., how much would you pay for a 1 MHz Pentium4?). Thus, in Figure 2, we present three hypothetical chip utility functions that explore the impact of non-linearity. Note that the relative heights of the different utility functions do not matter, since

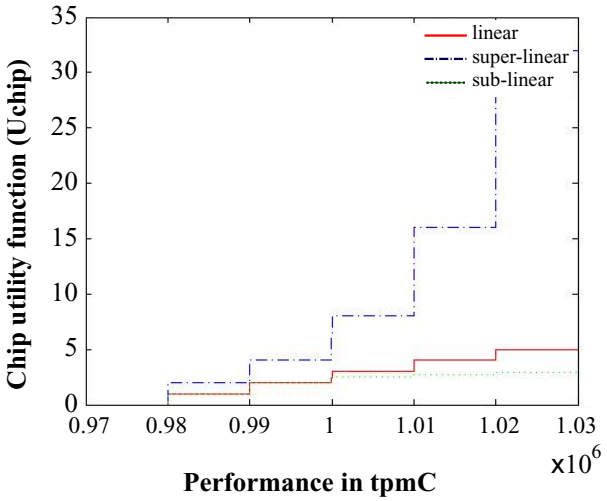
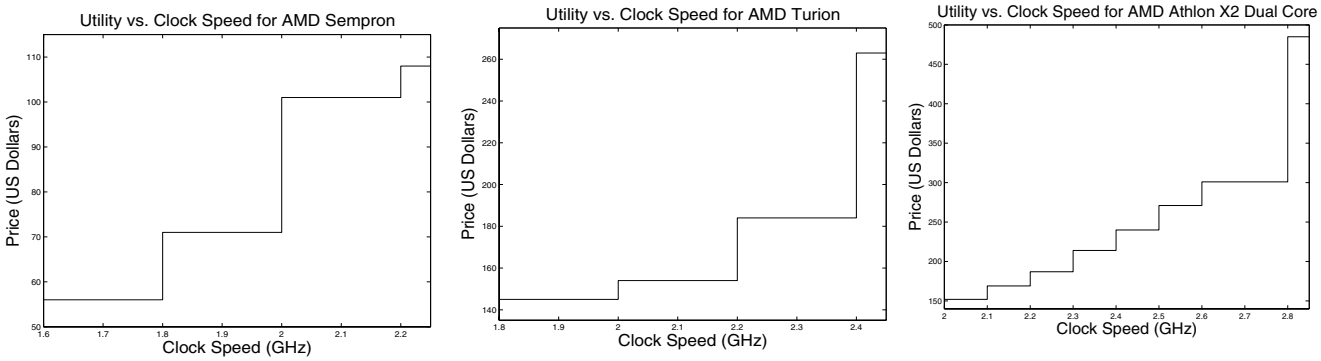


Fig. 2. Three Chip Utility Functions

we are not comparing across utility functions. One of the three curves is linear, in that the utility of each bin is a linear function of performance. The other two curves are super-linear and sub-linear. The super-linear chip utility function is familiar to those of us who shop for high-end laptops and desktops. The sub-linear chip utility function, however, may at first glance seem improbable. However, for chips where performance is not critical, such as embedded processors, there may be little utility beyond a given level of performance (e.g., a coffeemaker does not benefit from a 2 GHz controller). In fact, one might expect such a chip utility function to be a single step (i.e., a chip is either useful or not useful depending on if its performance is greater or less than a specific threshold), although we do not illustrate this more extreme function in the figure. We certainly do not claim that any of these three hypothetical chip utility functions are the exact shapes that would be used by any particular company, but they are intuitive and they help to illustrate possible differences between chip utility functions. In Section IV, we will present approximations of real-world chip utility functions that are based on commercial data.

The importance of the chip utility function is revealed by computing the system design utility functions for both systems P1 and P2 for the three chip utility functions in Figure 2. For the linear chip utility function, P1 and P2 unsurprisingly have almost identical system design utilities. For the super-linear chip utility function, P2 has a 20% advantage over P1. P2's edge is due to the larger utility for its chips that fall in the high end of the distribution. For the sub-linear chip utility function, P1 has a 7% advantage over P2. The moral of the story is that system design utility depends highly on the specific chip utility function and on the performance distribution for a system design.



Sempron Turion Athlon DualCore

Fig. 3. AMD High-End Processor Chip Utility as Function of Performance. Note that axes have different scales.

IV. Approximating Real-World CHIP Utility Functions

In Section III, we described intuitive chip utility functions to highlight potential differences between them. These idealized chip utility functions also simplified the computation of design utility. In this section, we describe approximations of actual chip utility functions for current microprocessors.

To define chip utility functions, we must determine which metrics to use. For a performance metric, we use the chip's clock frequency. Frequency is not an ideal metric, because performance is not a linear function of clock frequency, but it is readily available information. For a power metric, we use the chip's power rating, which is also available from the manufacturer. The most challenging metric to choose is the utility metric. We have chosen to approximate utility by using the manufacturer's list price for the chip. The price of an object is a reasonable approximation of its utility, although we realize that this approximation can potentially be skewed by business issues (e.g., cost, trying to undercut the competition, etc.).

We now present utility functions for various high-end and embedded processors. We collected our data from the Intel and AMD websites [2, 1, 9].

4.1 High-End Processors

For high-end processors, performance is generally the most critical factor, although there has recently been more emphasis on power-efficiency instead of simply raw performance. These high-end processors are used in servers, desktops, and laptops. We separate our analysis by vendor, focusing on AMD and Intel, in order to enable more even comparisons.

AMD. In Figure 3, we plot chip utility as a function of performance for three families of AMD processors: Sempron (256KB L2 cache), Turion (1MB L2 cache), and Athlon DualCore (1MB L2 cache).

The Sempron utility curve flattens out from the 2GHz to the 2.2GHz chip. The Sempron has the smallest L2 cache of the three chip families, and our hypothesis is that the Sempron becomes memory bottlenecked beyond 2GHz. Thus, the 2.2GHz processor's marginal benefit is likely outweighed by its additional power consumption.

The Turion displays an intuitive utility curve that reflects the premium in value for marginally faster processors. With its 1MB L2 cache, it can exploit the greater clock frequencies. This curve is what we expected for high-end processors.

The Athlon utility curve is similar to that of the Turion, although it does not rise as steeply until the chip exceeds 2.6GHz. We had expected the utility curve for the multicore to be steeper than those for the single core processors, because the performance gain is multiplied by the number of cores on the chip.

Intel. Similar to our analysis of the AMD chips, we analyzed current Intel chip utilities. The three chip families we studied are the Pentium4 (1MB L2 cache), Xeon (2MB L2 cache), and CoreDuo Mobile (2MB L2 cache).

The Xeon's utility curve increases steadily until starting to flatten a bit at the high end of performance. Our hypothesis is that the power consumption of server chips, such as this Xeon, is an important factor. A company with a large number of servers may not be willing to pay significantly more money for power and cooling in order to obtain a small benefit in performance. Thus, the highest end chips may not offer much additional utility.

The Pentium4 has a utility curve that is nearly flat on the lowest end, but then increases steadily as a function of clock frequency. This chip is primarily for desktops, and desktop power consumption is not yet a major issue (unless one con-

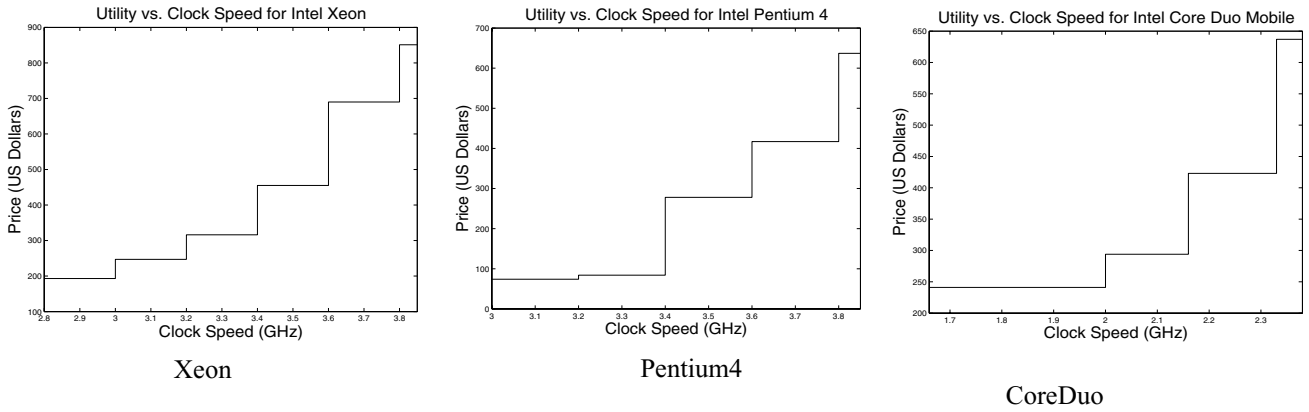


Fig. 4. Intel High-End Processor Chip Utility as Function of Performance. Note that axes have different scales.

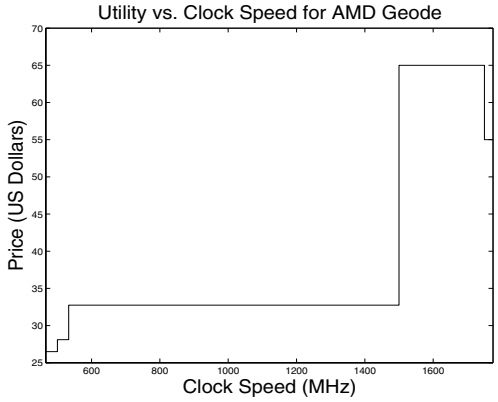


Fig. 5. AMD Geode Utility as Function of Performance

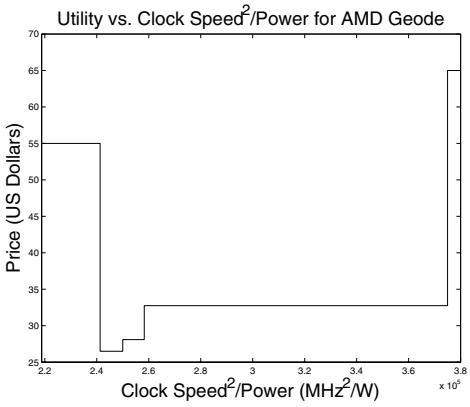


Fig. 6. AMD Geode Utility as Function of Performance^2/Watt

siders the power consumed by multiple desktops, which the average desktop buyer does not).

The CoreDuo shows an intuitive utility curve, similar to that of the AMD Turion. The lower frequencies of multicore chips means that even chips on the high end of their clock frequency ranges will still not consume an exorbitant amount of power.

4.2 Embedded Processors

For embedded processors, chip utility functions reflect the greater value placed on power efficiency. We focus our attention on AMD's Geode embedded processor. In Figure 5, we plot the chip utility function for the Geode as a function of performance.

We observe a strikingly different curve from those that we observed for high-end processors. Most notably, the utility curve reflects three phenomena. First, there is a steep increase in utility as a function of performance on the lowest end of the performance spectrum. The power consumptions of the these three lowest-end chips are all very low and approximately equal, and thus performance matters most.

Second, there is a big gap between the 533MHz chip and the 1.5GHz chip, and the utility of the 1.5GHz chip does not continue on the super-linear trend of the lowest-end chips.

Third, there is a *decrease* in utility moving from the 1.5GHz chip to the 1.75GHz chip. This decrease reflects the poorer power efficiency of the 1.75GHz chip. To further illustrate this effect, Figure 6 plots chip utility as a function of power efficiency, measured as performance^2/power (in units of MHz^2/Watt). We observe that the highest performing chip, the 1.75 GHz chip (at \$55), has the lowest power efficiency. The rest of the datapoints remain in the same order on the x-axis. What we observe is that there is a large utility for high performance (despite poor power efficiency), but even greater utility for the chip with slightly less performance but far better power efficiency.

V. Conclusions

In this paper, we have demonstrated the need for performance metrics that capture the impact of process variability. Considering only mean performance is not sufficient. Instead, we must consider a system's performance distribu-

tion and the utility of chips as a function of performance. By doing so, we can present a complete picture of performance that enables us to fairly compare system designs. We can also focus architects' efforts towards those aspects of a design that affect the performance distribution. For example, if most of the variability is due to the design of the register file and this variability is hurting system design utility, then architects can focus their efforts towards re-designing the register file to mitigate its performance variability.

This paper has focused on performance and power, but virtually all of our discussion also applies to thermal characteristics, reliability, and other system properties that are affected by process variability. For example, we can also determine a design's thermal distribution and we can develop a chip utility function that depends on thermal distribution. This utility function would depend on the cost to cool the chip. Furthermore, we could combine metrics, such as performance and reliability (i.e., performability), and explore their distributions and utility functions.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant CCF-0444516, the National Aeronautics and Space Administration under Grant NNG04GQ06G, a Duke Warren Faculty Scholarship (Sorin), and donations from Intel Corporation. Any opinions, findings and conclusions or recomendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation (NSF) or the National Aeronautics and Space Agency (NASA).

REFERENCES

[1] Advanced Micro Devices. AMD Introduces Line Of Low-Power, High-Performance AMD Geode Embedded X86 Processors. http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_543 85510,00.h% tml, May 2004.

[2] Advanced Micro Devices. AMD Processor Pricing. http://www.amd.com/us-en/Corporate/VirtualPressRoom/0,,51_104_609,00.html?re% dir=CPPR01, Dec. 2006.

[3] A. Agarwal, D. Blaauw, and V. Zolotov. Statistical Timing Analysis for Intra-Die Process Variations with Spatial Correlations. In *Proceedings of IEEE ICCAD*, pages 900–907, Nov. 2003.

[4] C. Amin et al. Statistical Static Timing Analysis: How Simple Can We Get? In *Proceedings of the 42nd Design Automation Conference*, pages 652–657, June 2005.

[5] S. Borkar. Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. *IEEE Micro*, 25(6):10–16, Nov/Dec 2005.

[6] H. Chang and S. S. Sapatnekar. Statistical Timing Analysis Considering Spatial Correlations Using a Single Pert-like Traversal. In *Proceedings of International Conference on Computer Aided Design*, pages 621–625, Nov. 2003.

[7] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann, third edition, 2003.

[8] Illinois Advanced Computing Systems Group. Illinois Verilog Model. http://www.crhc.uiuc.edu/ACS/tools/ivm/about.html.

[9] Intel Corporation. Intel Processor Pricing. http://www.intel.com/intel/finance/pricelist/processor_price_list.pdf?iid=In% vRel+pricelist_pdf, Dec. 2006.

[10] International Technology Roadmap for Semiconductors, 2003.

[11] J. Le, X. Li, and L. T. Pileggi. STAC: Statistical Timing Analysis with Correlation. In *Proceedings of the 41st Design Automation Conference*, pages 343–348, June 2004.

[12] S. Nassif. Design for Variability in DSM Technologies. In *Proceedings of First International Symposium on Quality of Electronic Design*, pages 451–454, Mar. 2000.

[13] C. Visweswariah et al. First-Order Incremental Block-Based Statistical Timing Analysis. In *Proceedings of the 41st Design Automation Conference*, pages 331–336, June 2004.

[14] Y. Zhan, A. J. Strojwas, X. Li, and L. T. Pileggi. Correlation-Aware Statistical Timing Analysis with Non-Gaussian Delay Distributions. In *Proceedings of the 42nd Design Automation Conference*, pages 77–82, June 2005.

[15] L. Zhang et al. Correlation-Preserved Non-Gaussian Statistical Timing Analysis with Quadratic Timing Model. In *Proceedings of the 42nd Design Automation Conference*, pages 83–88, June 2005.