

# Modeling and Architectural Simulations of the Statistical Static Timing Analysis of the Non-Gaussian Variation Sources for VLSI Circuits

Abu M. Baker\* and Yingtao Jiang\*

\* Department of the Electrical and Computer Engineering  
University of Nevada, Las Vegas  
Las Vegas, NV 89154

**Abstract-** As CMOS technology scales down, process variation introduces significant uncertainty in power and performance to VLSI circuits and significantly affects their reliability. Although Static-Timing Analysis (STA) it is an excellent tool, but current trends in process scaling have imposed significant difficulties to STA. As one of the promising solutions, Statistical static timing analysis (SSTA) has become the frontier research topic in recent years in combating such variation effects. This paper will be focusing on two aspects of SSTA and its applications in VLSI designs: (1) Statistical timing modeling and analysis; and (2) Architectural implementations of the atomic operations (max and add). Experimental results have shown that our approach can provide 282 times speedup when compared to a conventional CPU implementation.

**Index Terms-** Statistical static timing analysis, CMOS, VLSI, non-Gaussian, process variations

## I. INTRODUCTION

Static-timing analysis (STA) has been one of the most ubiquitous and popular analysis engines in the design of digital circuits for the last 30 years. However, in recent years, the increased loss of predictability in semiconductor devices has raised concern over the ability of STA to effectively model statistical variations. This has resulted in all-encompassing research [1], [2] in the so-called statistical STA (SSTA), which marks a significant departure from the traditional STA framework. The fundamental paleness of STA is that while global shifts in the process (referred to as die-to-die variations) can be approximated by creating multiple corner files, there is no statistically rigorous method for modeling variations across a die (referred to as within-die variations). However, with process scaling progressing well into the nanometer regime, process variations have become significantly more pronounced and within-die variations have become a non-negligible component of the total variation. It is shown that the incapability of STA to model within-die variation can result in either an over- or underestimate of the circuit delay, depending on the circuit topology [3]. Hence, STA's desirable property of being conservative may no longer hold for certain circuit topologies while, at the same time, STA may be overly pessimistic for other circuit topologies. This accuracy problem of STA can be even more pronounced in advanced processes. Consequently, the need for an effective modeling of process variations in timing analysis has led to extensive research in statistical STA.

SSTA algorithms can be broadly categorized into path-based and block-based. The path based SSTA seeks to estimate timing statistically on selected critical paths. However, the task of selecting a subset of paths whose time constraints are statistically critical has a worst-case computation complexity that grows exponentially with respect to the circuit size. Hence the path based SSTA is not easily scalable to handle realistic circuits. On the other hand, the block based SSTA champions the notion of progressive computation. Specifically, by treating every gate/wire as a timing block, the SSTA is performed block by block in the forward direction in the circuit timing graph without looking back to the path history. As such, the computation complexity of block based SSTA would grow linearly with respect to the circuit size. However, to realize the full benefit of block based SSTA, we have to address a challenging issue that timing variables in a circuit could be correlated due to either global variations( [4], [5], [6]) or path reconvergence ( [7], [8]). Global correlation refers to the statistical correlation among timing variables in the circuit due to global variations such as inter or intra-die spatial correlations, same gate type correlations, temperature or supply voltage fluctuations, etc. Path correlation, on the other hand, is caused by the phenomenon of path reconvergence, that is, timing variables in the circuit can share a common subset of gate/wire blocks along their path histories. Several solutions have been proposed to deal with either of these two types of correlations. In [4], [5], [6], the dependence on global variations is explicitly represented using a canonical timing model. However, these approaches did not take into account the path correlations. In [8], a method based on common block detection is introduced to deal with the path correlations. However, this method does not address the issue of dependence on global variations. To the best of our knowledge, there is no existing method that has dealt with both types of correlations simultaneously. We present a novel block based SSTA modeling in this paper that is designed to consider both global correlations and path correlations:

- We develop a model encompassed with numerical computations and tightness probabilities to conditionally approximate the MAX/MIN operator by a linear mixing operator.

- We extend the commonly used canonical timing model to be able to represent all possible correlations, including the path correlations, between timing variables in the circuit.

The remainder of this paper is organized as follows. SSTA problem formulation has been described in Section II. Section III details the solution approaches. Section IV details our architectural simulation and also present results from experiments which were conducted in order to benchmark our approach. We conclude in Section V.

## II. SSTA PROBLEM FORMULATION

In this section, we will formally define the problem to be solved.

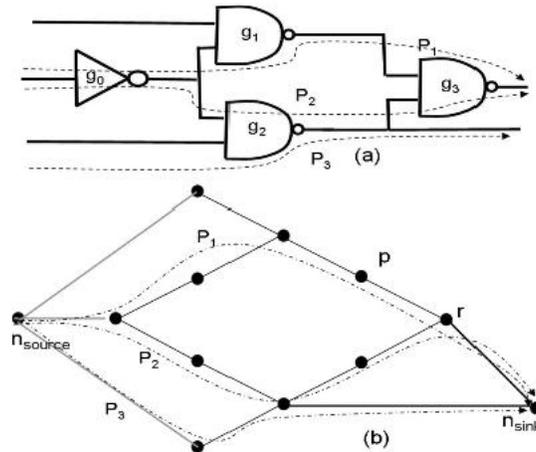


Fig. 1. Combinatorial circuit and its corresponding DAG [9].

Definition: A combinational circuit can be described using a Directed Acyclic Graph (DAG)  $G$  given as  $G = \{N, E, n_s, n_f\}$ , where  $N$  is the set of nodes corresponding to the input/output pins of the devices in the circuit,  $E$  is the set of edges connecting these nodes, each with weight  $d_i$ , and  $n_s, n_f$  are respectively source and sink of the graph. Figure 1(a) shows a digital circuit and its corresponding DAG is shown in Figure 1(b).

### Problem Formulation:

Let  $p_i$  be a path of ordered edges from a source to a sink in  $G$ . Let  $D_i = \sum d_{ij}$  be the path length of  $p_i$ . Then  $D_{max} = \max(D_1, \dots, D_i, \dots, D_n)$  is referred as the SSTA problem of the circuit.

There are two main challenges in SSTA. The Topological Correlation which emerges from reconvergent paths, these are the ones which originate from a common node and then converge again at another node (reconvergent node). Such correlation complicates the maximum operation as it now has to be computed over correlated RVs. In a circuit example shown in Figure 2, one can see that the two red paths reconverge at the rightmost gate ( $g_3$ ).

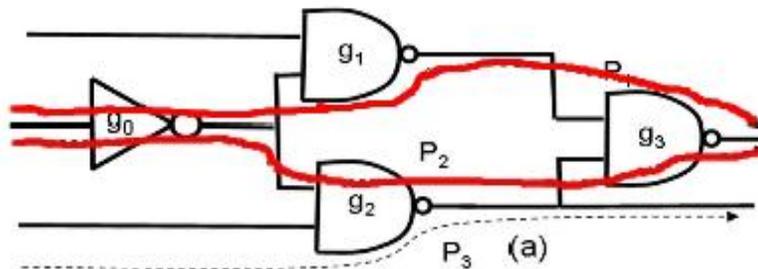


Fig. 2. Topological Correlation [9].

The second challenge is the Spatial Correlation. It arises due to device proximity on the die and gives raise to the problems of modeling delay and arrival time so that correlations are included, as well as preserving and propagating these correlations. Figure 3 shows such two paths correlated by two closely placed gates ( $g_1$  and  $g_2$ ).

### III. SOLUTION APPROACHES

The most general and brute force method of solving the above mentioned problem is to use numerical integration [7]. Although exact and applicable, this method is highly computationally expensive and thus, undesired. This leads to another approach, namely, the use of Monte Carlo methods [8]. The exact structure of these methods varies with the problem at hand. However, in general they all follow a common pattern: perform a statistical sampling of the sample space, perform deterministic computation for each sample, and aggregate the results into one final. In order to decrease the error, a lot of samples need to be taken, which, on the other hand, increases the computation effort. Therefore, probabilistic analysis methods are highly desired. Two such exist, one is the *Path-based* approach and the other is the *Block-based* approach.

The Path-based approach constructs a set of nodes that are likely to form the critical paths. The delay for each of these paths is then computed and a statistical maximum is performed over these results to yield the worst case delay.

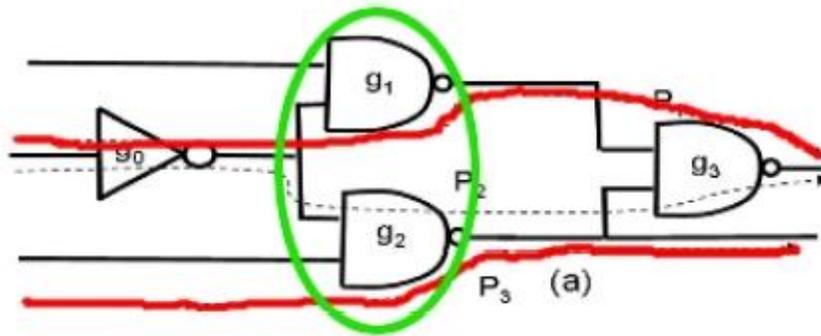


Fig. 3: Spatial Correlation [9]

However, there are several problems associated with this approach. Sometimes it is hard to construct a set of likely critical paths. Therefore, the worst case scenario can be unintentionally omitted. This significantly increases the number of computations needed. Therefore, it is desired to use the Block-based approach. There instead of constructing critical paths the whole graph is traversed node by node. For all fan-in edges to a node the associated delay is added to the arrival time at the source node (the node upstream of the current one). The final arrival time at the node is computed using a maximum operation over the previous results. This approach has the advantage of propagating only two times, the rise and the fall time.

#### A. Distribution Propagation Approaches

Analytical handling of distributions would be a good and computationally inexpensive approach. However, due to the nonlinearities and nonnormalities that are to occur in the dependencies and distributions used, it becomes a task close to impossible. There exist ways of handling this problem analytically, but assumptions are inevitable part of them. Therefore, another way is to discretize the distributions and normalize them so that the discrete values sum up to 1. In this way new set of probability mass functions is constructed, which closely approximates the real densities.

Now summation is an easy task to do. The result of such an operation is just a sum of shifted and scaled values of the delay. The shifts and the magnitude of the scaling is determined by the distribution of the arrival time.

$$z = x + y \tag{1}$$

$$f_z(t) = f_x(1)f_y(t - 1) + f_x(2)f_y(t - 2) + \dots + f_x(n)f_y(t - n) \tag{2}$$

$$f_z(t) = \sum_{i=-\infty}^{\infty} f_x(i)f_y(t - i) = f_x(t) * f_y(t) \tag{3}$$

Where  $x, y$  are the delays of two devices which are connected in series.  $z$  is the resulting delay.  $f_x, f_y, f_z$  are the delay functions of the device  $x, y$  and convolution of  $x$  &  $y$  function respectively.

Performing discrete time convolution is enough to compute the resulting delay from two devices in series. In order to compute the maximum delay between two paths ( $x$  and  $y$ ) two cases have to be considered. Either one of the path  $y$  has a particular delay and path  $x$  has a delay less than or equal to the one of  $x$  or vice versa (equation 5). In order to obtain a density function this must be computed for all possible values of the delay  $t$ .

$$z = \max(x, y) \tag{4}$$

$$f_z(t) = F_x(\tau < t)f_y(t) + F_y(\tau < t)f_x(t) \tag{5}$$

*B. Propagation of Delay Dependences*

Expressing the delay of each device by a linear combination of independent random variables leads to the creation of the canonical form.

$$d_a = \mu_a + \sum_i^n a_i z_i + a_{n+1} R \tag{6}$$

where  $\mu_a$  is the mean delay,  $z_i$  represents the  $n$  independent RVs used to express the spatially correlated device-parameter variations,  $R$  represents the residual independent variation, and coefficients  $a_i$ 's represent the sensitivity of delay to each of the RVs.

It will be convenient to express both the sum and the maximum of such canonical forms in a canonical form. This will preserve the same approach throughout the computation of the delay for the whole circuit. Expressing the sum ( $C$ ) of two canonical delays ( $A$  and  $B$ ) is almost a straightforward task. The only unintuitive part is the coefficient of residual independent variation  $c_{n+1}$ . As the two coefficients, of which it is composed, correspond to independent (orthogonal) RVs, the new coefficient must be equal to the combined magnitude of the two.

$$C = A + B \tag{7}$$

$$\mu_c = \mu_a + \mu_b \tag{8}$$

$$c_i = a_i + b_i \text{ for } 1 \leq i \leq n \tag{9}$$

$$c_{n+1} = \sqrt{a_{n+1}^2 + b_{n+1}^2} \tag{10}$$

Computation of the maximum is a significantly more involved. As the maximum operation is nonlinear, but the canonical form is, only an approximation of the maximum can be computed. The following is an algorithm proposed for solving this problem [10].

1) *Compute variances and covariance of A and B*

First of all the variance and covariance of the canonical forms  $A$  and  $B$  need to be calculated.

$$\sigma_a^2 = \sum_i^n a_i^2 \quad \sigma_b^2 = \sum_i^n b_i^2 \quad r = \sum_i^n a_i b_i \tag{11}$$

2) *Compute tightness probability  $T_A = P(A > B)$  (the probability that arrival time A is larger than B) as presented in [11]*

$$T_A = \Phi\left(\frac{\mu_a - \mu_b}{\theta}\right) \tag{12}$$

$$\Phi(x) = \int_{-\infty}^x \phi(x) dx \tag{13}$$

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) \tag{14}$$

$$\theta = \sqrt{\sigma_a^2 + \sigma_b^2 - 2r} \tag{15}$$

3) *Compute mean and variance of  $C = \text{maximum}(A, B)$*

The new mean and variance of the new canonical form  $C = \text{maximum}(A, B)$  have to be expressed.

$$\mu_c = \mu_a T_A + \mu_b (1 - T_A) + \theta \phi\left(\frac{\mu_a - \mu_b}{\theta}\right) \tag{16}$$

$$\sigma_c^2 = (\mu_a + \sigma_a^2) T_A + (\mu_b + \sigma_b^2) (1 - T_A) + (\mu_a + \mu_b) \theta \phi\left(\frac{\mu_a - \mu_b}{\theta}\right) - \mu_c^2 \tag{17}$$

4) *Compute sensitivity coefficient  $c_i$  using the tightness probability*

Then the weighting coefficients for the maximum.

$$c_i = a_i T_A + b_i (1 - T_A) \text{ for } 1 \leq i \leq n \tag{18}$$

5) *Compute sensitivity coefficient  $c_{n+1}$  of canonical form  $c_{approx}$  to make the variance of  $c_{approx}$  equal to the variance of  $C = \text{maximum}(A, B)$ .*

It was shown in [12] that a valid coefficient  $c_{n+1}$  always exists as the residue  $(\sigma_c^2 - \sum_i^n c_i^2)$  is always greater than or equal to zero.

Unfortunately, this approach only computes an estimate, which by no means guarantees conservative results. Therefore, it is not suitable as it might underestimate the delay on some occasions. Another way of coping with the problem is the use of the following relation.

$$\max(\sum_i^n a_i, \sum_i^n b_i) \leq \sum_i^n \max(a_i, b_i) \tag{19}$$

Consider the very simple canonical form for two delays  $d_a = \mu_a + a\Delta$  and  $d_b = \mu_b + b\Delta X$ , where  $\mu_a$  and  $\mu_b$  are the mean delays of  $d_a$  and  $d_b$  respectively, and  $a$  and  $b$  are their sensitivities to the common RV  $\Delta X$ . In [13] an example of  $d_a$  and  $d_b$  is shown as a function of  $\Delta X$ . The maximum of  $d_a$  and  $d_b$  is the upper envelope of these two intersecting lines, which is a nonlinear function and cannot be expressed exactly by the canonical form. Hence, to represent this maximum, a linear function of  $\Delta X$  must be constructed that approximates this nonlinear function.

Note that  $c_{approx}$  will at times underestimate and at times overestimate the actual result. On the other hand, the method proposed in [5] constructs a bound  $d_{c_{bound}} = \mu_{c_{bound}} + c_{bound}\Delta X$ , where  $\mu_{c_{bound}} = \max(\mu_a + \mu_b)$  and  $c_{bound} = \max(a, b)$ . As can be seen, the error of  $c_{approx}$  will be smaller than that of  $c_{bound}$ , where as  $c_{bound}$ , will be guaranteed conservative.

This result guarantees that if the higher of the coefficients corresponding to a particular independent RV is selected, then the result will be conservative. Therefore, a bounding canonical  $C_{bound}$  form of the delay can be constructed by selecting the higher mean and the largest coefficients.

$$\mu_c = \max(\mu_a, \mu_b) \tag{20}$$

$$C_{bound_i} = \max(a_i, b_i) \tag{21}$$

### C. Nonlinear and Nonnormal Approaches

Because of the existence of nonnormal distributions and nonlinear dependencies, special canonical forms have been developed to cope with these challenges [11]. All of these are handled by numerical computations and tightness probabilities. In order to include the effect of nonlinear dependencies additional term is included in the form.

$$d_a = \mu_a + \sum_i^n a_i z_i + \sum_{i=1}^n \sum_{j=1}^n b_{ij} z_i z_j + a_{n+1} R \tag{22}$$

Where  $z_1$  to  $z_n$  represent sources of normal variations, and  $z_{n+1}$  to  $z_{n+m}$  are RVs with nonnormal variations.

For the nonnormal distributions the same approach is used. The delay terms for both the normally distributed contributions and the nonnormal ones.

$$d_a = \mu_a + \sum_i^n a_i z_i + \sum_j^m a_{n+j} z_{n+j} + a_{n+m+1} R \tag{23}$$

Equations 22 and 23 can be aggregated in the following common form.

$$d_a = \mu_a + \sum_i^n a_i z_i + f(z_{n+1}, \dots, z_{n+m}) + a_{n+1} R \tag{24}$$

Where  $f$  represents the nonlinear function and is described as a table for computational purposes, and RVs  $z_{n+1}$  to  $z_{n+m}$  represent sources of normal variations with nonlinear dependences or nonnormal variations.

## IV. ARCHITECTURAL SIMULATIONS

The vector-thread (VT) architectural paradigm describes a class of architectures that unify the vector and multithreaded execution models. In other words, VT architectures compactly encode large amounts of structured parallelism in a form that lets simple microarchitectures attain high-performance at low power by avoiding complex control and datapath structures, and by reducing activity on large wires. Moreover, VT exploits fine-grained parallelism locality more effectively than traditional superscalar, VLIW, or multithreaded architectures. In this paper, we feed our statistical model to the specified Scaled Vector Thread Architecture as well as Graphic Processing Unit (GPU) GeForce 8800 GTX architecture with some parameters shown in Table 1 which includes a MIPS-RISC control processor or **Single instruction, multiple data** (SIMD) processor, 32 Kbytes of cache, and a four-lane vector-thread unit or multiprocessor that can execute 16 operations per cycle and support up to 128 simultaneously active virtual processor threads.

Table 1. Processor parameters

Clock Rate	Vector Thread Units/ # of Multiprocessors	# of Clusters/# of Processors	# of Registers (per cluster)/# of Registers (per processor)
400 MHz	4	16	8192

Our SSTA delay model has been implemented in C/C++ and tested by benchmark circuits. It is noted that before testing all benchmark circuits are re-mapped into a library which has gates of *not*, *nand2*, *nand3*, *nor2*, *nor3* and *xor/xnor*. Table 2 summarizes the performance comparison and runtime estimations. We ran 60 large IWLS, ITC and ISCAS benchmark designs to compute the per-

circuit speed of our tightness probability based SSTA engine implemented on vector thread architecture. This tightness probability based analysis was performed with 4 vector thread units. Columns 1 list the name of the circuit. Columns 2, 3 and 4 list the number of primary inputs, primary outputs and gates in the circuit. Columns 5 and 7 list the GPU and CPU runtime, respectively. The time taken to transfer data between the CPU and GPU was accounted for in the GPU runtimes listed. In particular, the data transferred from the CPU to the GPU is the arrival times at each primary input, and the  $\mu$  and  $\sigma$  information for all pin-to-output delays of all gates. Column 8 reports the speedup obtained by using a single GPU card. Our results indicate that our approach can obtain an average speed up of about 282 times as compared to a serial CPU implementation and is faster than GeForce 8800 GTX.

Table 2: SSTA results using tightness probability

Circuit	# Inputs	#Outputs	#Gates	Single GPU runtimes (s)	Scaled VT Processor runtimes (s)	CPU runtimes	Speedup For Single GPU	Speedup For Scaled VT
b14	276	299	9496	4.734	4.201	1303.63	275.394x	310.314x
b15_1	483	518	13781	6.952	6.521	1891.884	272.116	290.121x
b17	1450	1511	41174	20.736	19.311	5652.45	272.589x	292.706x
b18	3305	3293	6599	6.326	5.977	905.924	143.197	151.568x
b21	521	512	20977	10.311	10.101	2879.765	279.298x	285.097x
b22_1	734	725	25253	12.519	12.210	3466.783	276.913x	283.929x
s832	23	24	587	0.298	0.248	80.585	270.376x	324.939x
s8381	66	33	562	0.295	0.278	77.153	261.341x	277.528x
s1238	32	32	857	0.432	0.419	117.651	272.248x	280.789x
s1196	32	32	762	0.388	0.359	104.609	269.796x	291.389x
s1423	91	79	949	0.521	0.497	130.281	249.858x	262.134x
s1494	14	25	1033	0.508	0.489	141.812	279.414x	290.004x
s1488	14	25	1016	0.5	0.481	139.479	279.187x	289.977x
s5378	199	213	2033	1.16	0.979	279.094	240.58x	285.080x
s92341	247	250	3642	1.949	1.766	499.981	256.57x	283.114x
s13207	700	790	5849	3.512	3.271	802.963	228.633x	245.479x
s15850	611	684	6421	3.675	3.347	881.488	239.855x	263.366x
s35932	1763	2048	19898	11.318	11.008	2731.638	241.349x	248.150x
s38584	1464	1730	21051	11.544	11.104	2889.924	250.335x	260.259x
s38417	1664	1742	18451	10.341	9.97	2532.991	244.958x	254.061x
C1355	41	32	715	0.366	0.309	98.157	268.363x	317.660x
C1908	33	25	902	0.446	0.393	123.828	277.46x	315.083x
C2670	233	140	1411	0.797	0.689	193.705	242.906x	281.139x
C3540	50	22	1755	0.842	0.803	240.93	286.1x	300.037x
C432	36	7	317	0.155	0.139	43.518	280.605x	313.079x
C499	41	32	675	0.347	0.317	92.665	267x	292.318x
C5315	178	123	2867	1.461	1.379	393.588	269.323x	285.415x
C6288	32	32	2494	1.197	1.139	342.381	285.927x	300.597x
C7552	207	108	3835	1.899	1.810	526.477	277.214x	290.871x
C880	60	26	486	0.253	0.224	66.719	263.923x	297.852x
Avg							258.994x	282.1352x

## V. CONCLUSION

In this paper, we have presented the implementation of tightness probability based SSTA on Vector Thread Architecture as well as a GPU GeForce 8800 GTX architecture. Tightness probability based SSTA is computationally expensive, but crucial in design timing closure since it enables an accurate analysis of the delay variations. Our implementation computes multiple timing analysis evaluations for a single gate in parallel. Threads which execute in parallel do not have data or control dependencies on each other. All threads execute identical instructions, but on different data. Our results indicate that our approach can provide 282 times speedup when compared to a conventional CPU implementation.

#### REFERENCES

- [1] T. Kirkpatrick and N. Clark, "PERT as an aid to logic design," *IBM J.Res. Develop.*, vol. 10, no. 2, pp. 135–141, Mar. 1966.
- [2] H. Jyu, S. Malik, S. Devdas, and K. Keutzer, "Statistical timing analysis of combinational logic circuits," *IEEE Trans. Very Large Scale Integr.(VLSI) Syst.*, vol. 1, no. 2, pp. 126–137, Jun. 1993.
- [3] R. Brashear, N. Menezes, C. Oh, L. Pillage, and M. Mercer, "Predicting circuit performance using circuit-level statistical timing analysis," in *Proc. DATE*, Mar. 1994, pp. 332–337.
- [4] C. Visweswariah, K. Ravindran, and K. Kalafala, "First-order parameterized block-based statistical timing analysis," *TAU'04*, Feb 2004.
- [5] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," *ComputerAided Design, 2003 International Conference on. ICCAD-2003*, pp. 900 – 907, Nov 2003.
- [6] H. Chang and S. S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," *ICCAD'03*, pp. 621–625, Nov 2003.
- [7] A. Agarwal, V. Zolotov, and D. Blaauw, "Statistical timing analysis using bounds and selective enumeration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1243 –1260, Sept 2003.
- [8] A. Devgan and C. Kashyap, "Block-based static timing analysis with uncertainty," *ICCAD'03*, pp. 607–614, Nov 2003.
- [9] D. Blaauw, K. Chopra, A. Srivastava, and L. Scheffer, "Statistical timing analysis: From basic principles to state of the art.," *IEEE Transactions on ComputerAided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 589-607, April 2008.
- [10] H. Chang and S. Sapatnekar, "Statistical timing analysis considering spatial correlations using a single pert-like traversal," in *Proc. ICCADO3*, 2003.
- [11] Charles E. Clark. The greatest of a finite set of random variables. *Operations Research*, 1961.
- [12] D. Sinha, N. Shenoy, and H. Zhou, "Statistical gate sizing for timing yield optimization," in *Proc. ICCAD*, 2005, pp. 1037–1041.
- [13] H. Chang, V. Zolotov, S. Narayan, C. Visweswariah, Parameterized block-based statistical timing analysis with non-Gaussian parameters, nonlinear delay functions, in: Proceedings of the Design Automation Conference, June 2005, pp. 71–76.

#### AUTHORS

**First Author** – Abu M Baker, Ph.D. Candidate at the University of Nevada, Currently working at Microsoft Corporation and [bakera2@unlv.nevada.edu](mailto:bakera2@unlv.nevada.edu).

**Second Author** – Dr. Yingtao Jiang, Associate Professor in the Department of Electrical and Computer Engineering at the University of Nevada, Las Vegas, [yingtao.jiang@unlv.edu](mailto:yingtao.jiang@unlv.edu)

**Correspondence Author** – Abu M Baker, [bakera2@unlv.nevada.edu](mailto:bakera2@unlv.nevada.edu), 15975 NE 84<sup>th</sup> Way, Redmond, WA-98052, +1-281-222-8936.