

# Implications of Device Timing Variability on Full Chip Timing

Murali Annavaram, Ed Grochowski, Paul Reed

Microarchitecture Research Lab, Intel Corporation

2200 Mission College Blvd, Santa Clara, CA 95054

[murali.m.annavaram@intel.com](mailto:murali.m.annavaram@intel.com), [edward.grochowski@intel.com](mailto:edward.grochowski@intel.com), [paul.reed@intel.com](mailto:paul.reed@intel.com)

## Abstract

*As process technologies continue to scale, the magnitude of within-die device parameter variations is expected to increase and may lead to significant timing variability. This paper presents a quantitative evaluation of how low level device timing variations impact the timing at the functional block level. We evaluate two types of timing variations: random and systematic variations. The study introduces random and systematic timing variations to several functional blocks in Intel® Core™ Duo microprocessor design database and measures the resulting timing margins. The primary conclusion of this research is that as a result of combining two probability distributions (the distribution of the random variation and the distribution of path timing margins) functional block timing margins degrade non-linearly with increasing variability.*

## 1. Introduction to Device Variations

As process technologies continue to scale to ever-smaller dimensions, the magnitude of device parameter variations is expected to increase [2][3][5][7][14]. There are two primary sources for these variations. First, the transistor gate length in today's process technology is shorter than the wavelength of the light used in etching the devices. The wavelength of the light used in etching a 65-nm transistor is 193-nm. Unfortunately, the gap between the transistor width and the wavelength of light is expected to grow with each process technology, since the wavelength has not been reducing at the same rate as the silicon process. This gap results in several within die and die-to-die fluctuations.

Second, as the process technology continues to scale the transistor area is halved with every generation. Hence, number of dopant atoms within the channel also decrease exponentially. For instance, Borkar [3] showed that the number of dopant atoms in a 65nm transistor channel are just around 100. Borkar showed that even small fluctuations in the number of dopant atoms result in significant change in the transistor's electrical characteristics.

Device variations may be broadly classified as die-to-die variations, random (uncorrelated) within-die variations, and systematic (correlated) within-die variations [5]. These variations lead to several problems in chip design. For instance, increased susceptibility to transient errors, more rapid wear-out, and timing variability are some of the manifestations of device variability. This paper focuses on timing variability which

occurs when transistors with the same physical dimensions within a die have different delay characteristics. In particular, we study how random and systematic within-die variations impact the overall chip timing.

A major contribution of this paper is that we use Intel® Core™ Duo microprocessor design database and introduce timing variations to basic cells (such as NAND gates) within three synthesized functional blocks (FUBs). We measure how cell level variations affect the FUB timing margins. The primary conclusion of this research is that as a result of combining two probability distributions (the distribution of the random variation and the distribution of path timing margins) functional block timing margins degrade non-linearly with increasing variability. Our results also show that variability affects larger FUBs more than smaller FUBs. Finally, we show that systematic variations impact FUB timing even more than random variations. It is important to note that the focus of this paper is to present quantitative evaluation of timing variability on an industrial CPU design database in order to provide valuable data that can be used by researchers in further studies. This paper, in particular, does not present any specific solutions to the variability problem as this work is still on-going.

The rest of this paper is organized as follows. Section 2 presents our methodology of applying timing variability to the standard cells in three FUBs. Section 3 presents how the FUB timing changes when varying the underlying cell timings. Section 4 presents the methodology for generating systematic random numbers and how we applied systematic variations at the FUB level. Section 5 compares results from systematic and random variations. Section 6 presents a brief survey of the related work and some potential solutions to timing variability and we conclude in Section 7.

## 2. FUB Timing Variability Methodology

This section presents the details of the timing variability study done on three FUBs (PMSYN, MOSS and DCCTLS) from the Intel® Core™ Duo microprocessor design database. We chose these three FUBs for our study because they are synthesized blocks, containing only standard cells whose behavior is well-characterized. The netlist sizes of the three FUBs are shown in Table 1. DCCTLS is the largest FUB with 17562 standard cells and 18159 circuit paths through the FUB. As a first step we took the netlist representation of each of the three FUBs and generated the timing information using

an internally-developed static timing analysis tool. The tool analyzes the netlist, determines the expected delay for each standard cell, generates an expected interconnection delay based on the wire lengths, and computes the delays for each circuit path within the block.

The delay computed by the tool for each path in a FUB is compared against the target clock period to compute a timing margin of the FUB. If the total delay along a path is less than the target clock period, then the path has positive margin and is considered non-critical. On the other hand, if the path delay is more than the target clock period, then that path has a negative margin and is considered critical. Before we started on the timing variability study, we generated the margins for the three FUBs assuming no device variability. These margins are used as the baseline for comparison in our study. Note that a well-crafted design has most paths with margins just above zero.

	Cell	Nodes
DCCTL5	17562	18159
MOSS	3561	3884
PMSYN	3039	2637

**Table 1 Cell and Node sizes of 3 Synthesizable FUBs**

## 2.1 Cell Size Considerations for Variability

By default the timing tool looks up the cell characterization library (given as input to the tool) to determine the delay of each standard cell. Cell delay is typically a function of the process technology and the cell size. In our design library, each cell has 8 sizing options. The smallest cell uses the smallest transistors available in a given process while the biggest cell uses transistors that are 8 times wider. For a given process, bigger cells have shorter delay but consume more area and power. Typically designers use bigger cells in critical paths to meet the timing requirements and smaller cells in non-critical paths to improve the transistor density.

Early on in the study we realized that the three FUBs in our study use standard cells of varying sizes to meet the timing and area requirements. One indirect consequence of using bigger cells is that these cells are less susceptible to random dopant variations since they tend to use transistors with wide channel widths. Based on discussions with our designers and experiments conducted internally, we estimated that cell variation is inversely proportional to the square root of the cell size. For instance, doubling the cell size reduces the variability by 30%.

## 2.2 Generating Timing Overrides

Once the baseline timing margins with no variability are established, we used our timing analysis tool's override commands to study the effects of device variability. The override commands provide the ability to override the default cell timing. The designer can specify a

multiplier and an adder coefficient for any cell in a FUB netlist. The tool then multiplies the default delay obtained from the library with the multiplier coefficient and then adds the adder coefficient to generate the new delay. For instance, if the default delay of a NAND cell is 10 picoseconds and the override command specifies a 0.9 multiplying and 0.3 adder coefficients the tool computes the new cell delay as 9.3 picoseconds ( $10 \cdot 0.9 + 0.3$ ). In this paper, the adder coefficient is always set to zero and only the multiplier coefficient is used to study the effects of timing variations. Since the study is restricted to timing within a FUB it is reasonable to expect that wire lengths used in these FUBs are quite small and as such the effect of variation on interconnection delays is negligible. Hence interconnection delay variability is not modeled in this study.

In the first set of experiments, we study the effects of random timing variations on FUB timing margins. Systematic variations are discussed later in Section 4. Since determining the exact probability distribution for variations is difficult to obtain, we assume the variability of the smallest cells in a given process varies according to the Gaussian distribution. Similarly, it is difficult to quantify the extent of device variability that may be expected in future process generations. Hence, we study a range of values. Our chosen random distributions assume that device timing can vary by 0%, 10%, 20%, 30%, 40%, and 50%. This ratio is simply the sigma/mean of the Gaussian random numbers. Note that it is possible for a cell to be faster or slower than its pre-specified delay since the Gaussian distributed random multipliers can be either less than 1 or greater than 1.

For each FUB and for each variability value, we generated as many random numbers as there are cells in that FUB. For each variability value, the Gaussian distributed random numbers can be used directly as multiplying coefficients for the smallest cells since we assume the variation of smallest cells follow the Gaussian distribution. As discussed in Section 2.1, for bigger cells in the FUB we scaled the random number by square root of the cell size and then multiplied the scaled random number with the default cell delay. For instance, consider two NAND gates, one is the smallest NAND cell with delay  $t_{small}$  and the other is 2 times bigger with delay  $t_{large}$ . If the Gaussian random timing multipliers associated with the cells are  $R_{small}$  and  $R_{large}$  then the effective delays with variation are  $t_{small} * R_{small}$ , and  $t_{large} * 1/\sqrt{2} * R_{large}$ , respectively. For each variability value, we reran the timing tool with the recalculated cell delays and computed the timing margin.

In order to study the statistical distribution of circuit timing variations, for each variability value we generated 40 timing override command files per FUB. Each file contains one set of override commands for all the cells in a FUB and is used as input to one run of the timing analysis tool. Thus for each FUB and for each variability value we

ran the timing analysis tool 40 times with a different set of override commands as inputs. We automated the task of generating the override command files, running the tool, and extracting timing margins. Note that we limited the number of statistical runs to 40 in order to speedup the simulations. We have used much larger number of runs for one of the FUBs (PMSYN) in early studies and the results are similar to the 40 run results presented here.

### 3. Random Timing Variation Results

In this section we present three sets of data for each of the three FUBs. First, we show the distribution of margins without variability. Then we show how the worst case negative margins are impacted over the 40 runs. Finally, we analyze why the negative margins increase non-linearly with variability.

#### 3.1 Baseline Setup Time Margins

In this section we first present the setup time margins for the baseline design of the three FUBs assuming no timing variability for the devices. Figure 1 shows PMSYN's setup timing margins for the base case; only 17 paths have negative margin and the worst negative margin is only -5% of the target clock period. Figure 2 and Figure 3 show the same data for MOSS and DCCTLS, respectively. As is expected of any good industrial design, there is a steep hill near the zero setup margins for all three FUBs indicating that a vast majority of the paths are optimized such that the delay through the paths is just below one clock cycle. A steep hill near zero timing margin is an optimal design choice for power, performance, and area tradeoffs. Even in a large FUB such as DCCTLS, only 33 paths have negative margin and the worst negative margin is -8.6% of the clock period. The design database used in this study is a few steppings behind the production database. These graphs, however, indicate that a high degree of tuning has already taken place during the design process, with perhaps a small amount of timing improvement remaining for future steppings.

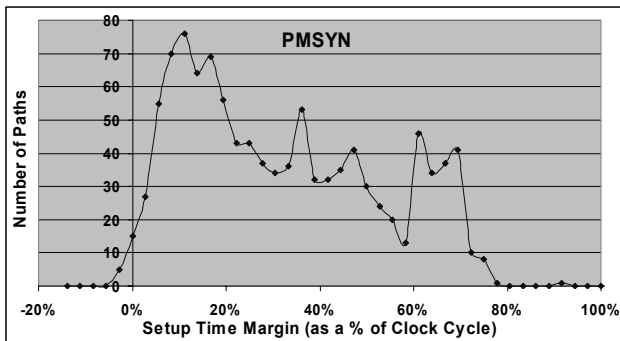


Figure 1 PMSYN Setup Margins without Variability

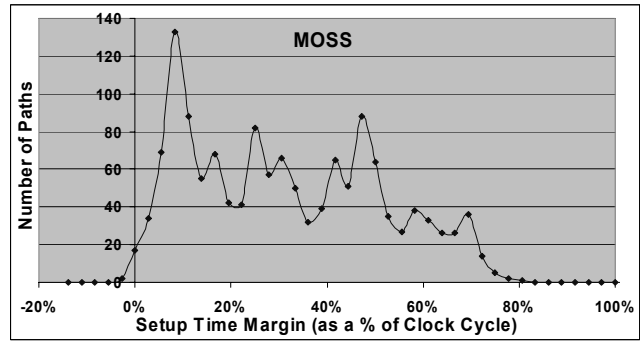


Figure 2 MOSS Setup Margins without Variability

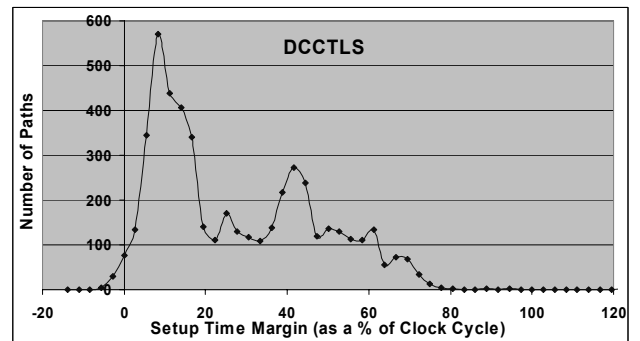


Figure 3 DCCTLS Setup Margins without Variability

#### 3.2 Setup Time Margins With Variability

In this section we will quantify the effects of device timing variations on the FUB setup timing margins. The Y-axis in Figure 4 shows how PMSYN's worst negative setup margin is impacted when the device timing varies between 0% and 50%. Each line in the figure shows the worst negative margin sorted across the 40 runs for a given variability value. For instance, if the device timing varies by 10%, the worst negative path margin increases from -5% to -7.2% of the clock period for the slowest FUB in the group of 40. On the right-hand side of Figure 4, one can see that a few FUBs in the group of 40 actually become faster as a result of adding 10% variability. Since we are modeling uncorrelated random variations, it is possible for some cells to get faster than the baseline and hence some critical paths may be eliminated. With 20% device variability the negative margin increases to -12% for the slowest FUB in the group of 40, and with 30% variability the negative margin becomes -20% of the clock period. With these higher amounts of variability, no FUBs run faster than the baseline. Figure 5 and Figure 6 show how the negative margin increases with increasing variability for MOSS and DCCTLS, respectively.

Figure 7 shows the effects of increasing timing variability on the worst negative margins, averaged over the 40 runs. There is a distinct knee in the curve for small variability values and the negative margin increases non-linearly with variability. This implies that as device variability increases in the future, the full-chip timing

margins can be expected to degrade dramatically. Figure 7 also demonstrates that variability affects larger FUBs (DCCTLs) more than smaller FUBs. As the number of paths that must run synchronously increases as is the case with larger FUBs, the FUB negative margin also increases. With increasing number of paths, there is greater probability of having outliers (slow paths). Since the chip can clock only as fast as its slowest path, larger FUBs are more likely to create lower frequency chips.

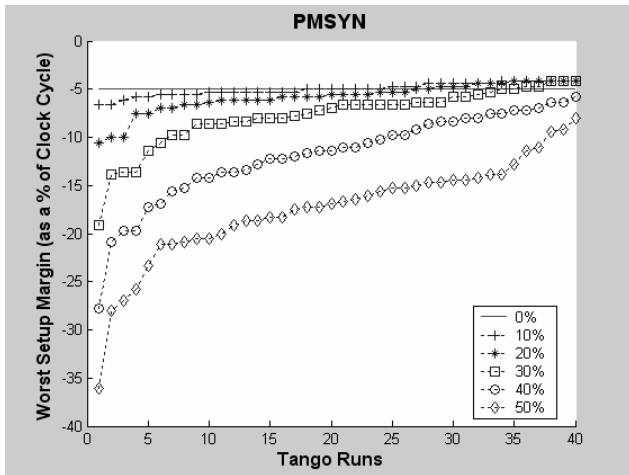


Figure 4 PMSYN Worst Setup Margin with Variability

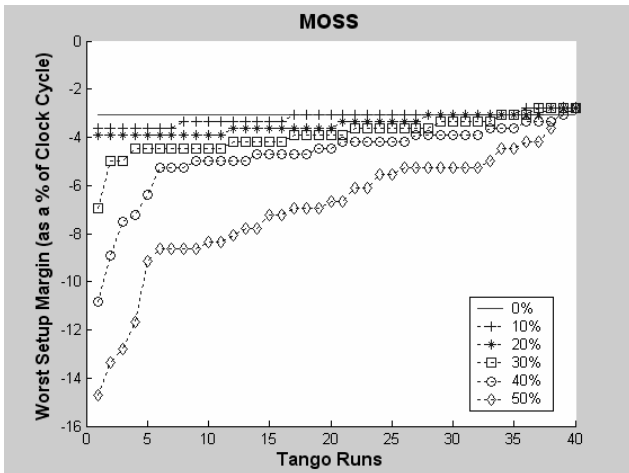


Figure 5 MOSS Worst Setup Margin with Variability

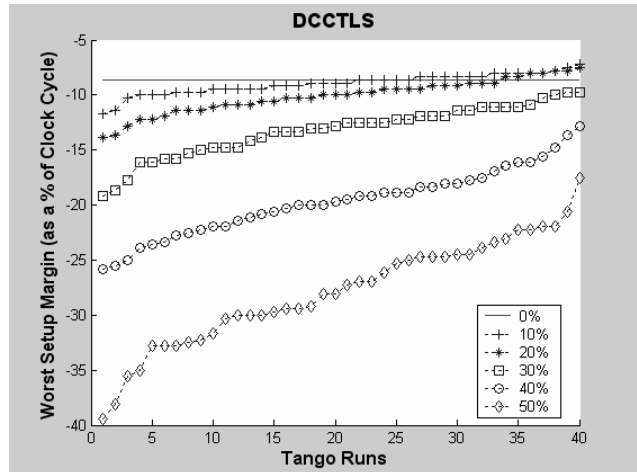


Figure 6 DCCTLs Worst Setup Margin with Variability

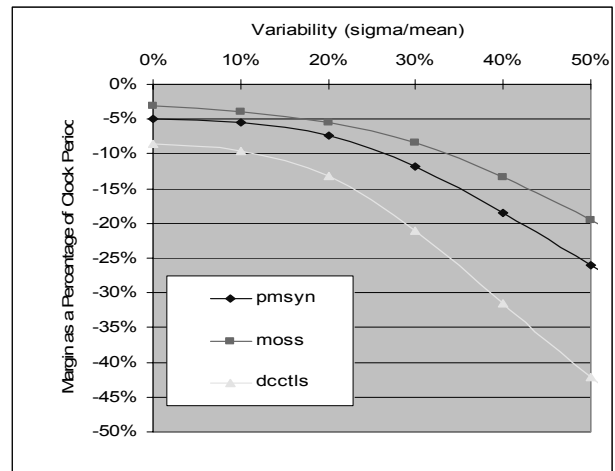


Figure 7 Effects of Timing Variability on Margin

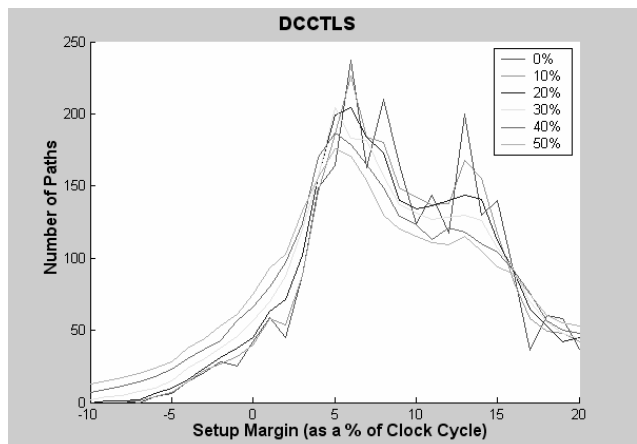


Figure 8 DCCTLs Path Distribution with Variability

### 3.3 Analysis of Non-linear Timing Degradation

The non-linear increase in delay with increasing variability may be explained as follows. As shown in Section 3.1 any realistic chip design contains paths with a

distribution of timing margins. With small amounts of variability, the FUB timing margins continue to be dominated by one or two slowest paths, which explains why some FUBs in the group of 40 become faster with small variability. However, as variability is increased, increased numbers of paths have the potential to affect the FUB-level timing margins. The distribution of path margins is determined by the combination of the natural distribution of path margins without variability, and the distribution of the random timing variations.

Figure 8 plots the timing margins for all the paths within DCCTLS with variability. Paths that do not meet the timing goal have negative margin and are on the left side of the graph. With 0% variability, the design contains only a small number of paths that do not meet speed. The 0% variability line becomes very steep as it passes through zero margin towards small positive margins. The steepness of the graph reveals that the designers have done a good job in optimizing the design (i.e. the delays of all critical paths are approximately the same). Some paths are not timing-critical and have positive margins.

With increasing timing variability, the graph becomes less steep as it crosses through zero timing margin. Since the area under the curve is the same in all cases, the introduction of random timing variations has converted paths that formerly had small positive margins into paths that now have small negative margins. Random timing variations have the effect of “smoothing out” any abrupt features in the functional block’s natural distribution of margins. In particular, the steep hill around zero margin so carefully constructed by the chip designers becomes much more gradual. This causes the overall FUB timing to degrade in a non-linear manner.

The combination of the path timing margin distributions and the random timing variation distributions may be computed mathematically using a convolution. A convolution of the DCCTLS margin in Figure 3 with a Gaussian distribution, yields similar results as the timing analysis tool in Figure 8.

#### 4. Impact of Systematic Timing Variability

In this section, we study the impact of systematic timing variability. Random timing variability assumes cell variations are independent of each other. Systematic timing variability, on the other hand, implies that cells that are closely located in space are likely to have similar variations. Since systematic variations depend on cell location, it is necessary to know the FUB layout so as to identify the (X, Y) co-ordinates of each cell within a FUB. The synthesis tool used in this research creates a layout file from which we can extract the co-ordinates of each cell in the FUB.

Table 2 shows the bounding co-ordinates for the three FUBs. These co-ordinates are in units of 1/1000 of a

micron. We divided each FUB into a grid of 1 micron X 1 micron and assumed that all cells that fall within a grid have the same variability. Furthermore, cells in the “nearby” grids have correlated variability as described in the next section.

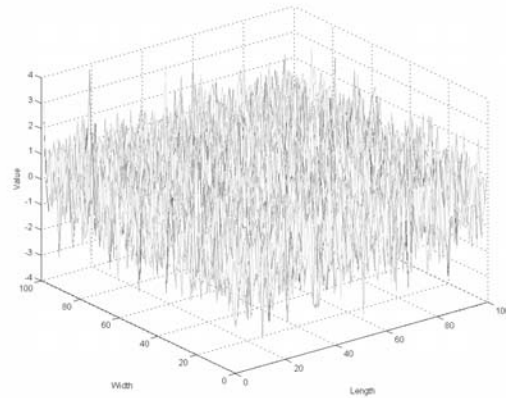
	XMIN	XMAX	YMIN	YMAX
PMSYN	1155	171825	1100	175475
MOSS	1815	175500	1315	259615
DCCTLS	1155	368060	1315	471325

**Table 2 FUB dimensions used in systematic variability**

#### 4.1 Generating Correlated Random Numbers

The next step in our analysis is to generate correlated random numbers. For this purpose, we first generated a matrix of uncorrelated random numbers. Then we averaged each number with several neighbors to generate correlated random numbers. Generating correlated random numbers introduces new degrees-of-freedom in choosing the amount and shape of correlation between neighboring cells. For our experiments, we characterized the distance (=1/frequency) over which the magnitude of the variability has halved by running a 2D FFT on the matrix of correlated random numbers. Figures 9-11 shows a 100x100 matrix with uncorrelated random numbers, and correlated random numbers in which the magnitude of the variability has halved at a distance (D) of 4, 12, and 25 units, respectively.

We superimposed the fub layout on the correlated random number matrix to generate the timing overrides. For instance, if a cell “i” has (Xi, Yi) co-ordinates then we use Matrix [Xi, Yi] as the override value in our static timing tool.



**Figure 9 Random Number Matrix**

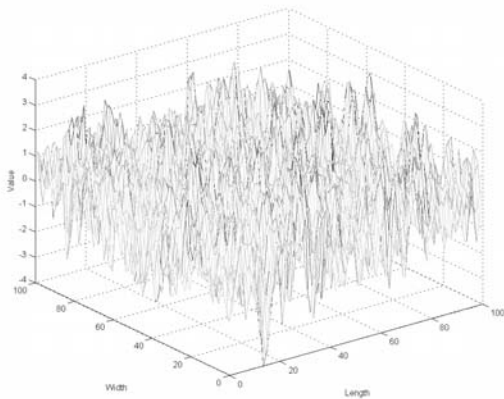


Figure 10 Correlated Random Numbers (D=4)

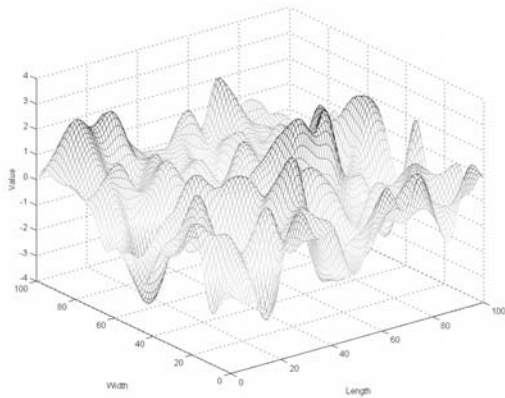


Figure 11 Correlated Random Numbers (D=12)

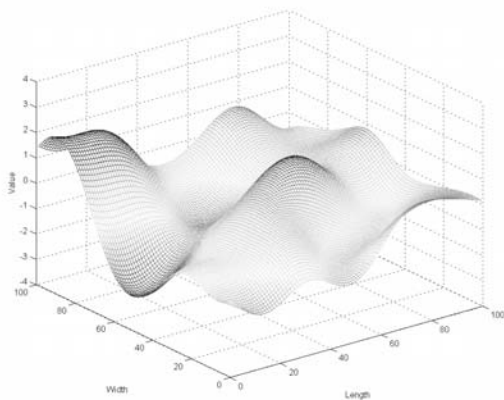


Figure 12 Correlated Random Numbers (D=25)

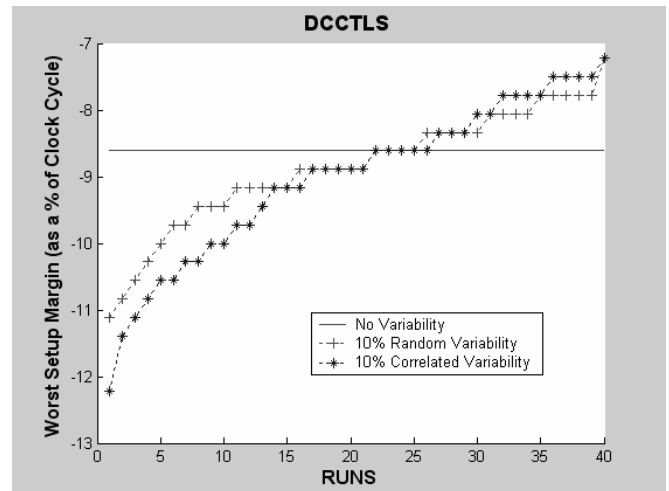


Figure 13: Comparing Systematic and Random Variability

## 5. Systematic Timing Variation Results

The graph in Figure 13 compares the worst case setup margin of DCCTLS for 10% random variability and 10% systematic variability. The most important observation from this graph is that negative margin paths get even worse with systematic variability. Further analysis of the layout files showed that cells that are in the critical path are also spatially co-located in order to reduce any potential interconnection delays. Hence, in systematic variation when one cell on the critical path is negatively impacted then all other cells on the critical path are likely to be impacted in a similar fashion. On the other hand, in uncorrelated random variations, different cells within a critical path can potentially have opposing timing effects thereby reducing the worst setup margin. Systematic variation leads to more severe timing effects than random variation because with systematic variation, multiple gates along a critical path are systematically affected in the same manner, whereas with random variation, cancellation may occur as certain gates within a path become faster whereas other gates within the same path become slower.

## 6. Related Work

It is only recently that researchers have begun to address the challenges posed by device parameter variations to future high performance microprocessor designs. Borkar et al. [2] noted that technology scaling beyond 90nm leads to larger device parameter variations, which are changing the microprocessor design problem from deterministic to probabilistic. Bowman et al. [4][5][6] used statistical analysis to study the effects of device parameter variations. A model for the maximum

clock frequency of a microprocessor (FMAX) is derived and compared against wafer sort data. The results are bounded for two extremes of within-die variations: completely systematic variations, and completely random variations. The author concludes that essentially a generation of performance gain may be lost in the 50nm generation due to systematic within-die fluctuations. Harris and Naffziger [10] characterize the effects of process variation on clock skew.

Solutions to the timing variability problem may be grouped into two categories: near-term and long-term. One near-term solution is the use of selective device sizing. Since device variability is more problematic at small feature sizes, designers can selectively employ bigger devices along critical timing paths. Wider transistors are less susceptible to random timing variations in addition to having increased drive strength. A preliminary analysis of the critical paths in DCCTLS showed that only 200 nodes out of a total of 18000 nodes contribute to the top 10 worst negative margin paths. Judiciously increasing the device sizing of these 200 nodes can reduce the variability in these speed paths. The addition of timing variability as a design constraint by sizing tools is an extension of today's device sizing algorithms which focus on delay and power constraints.

While some near-term solutions are already being used in today's designs, long-term solutions are also being investigated. Perhaps the ultimate solution to timing variability is the use of asynchronous design. Asynchronous systems [11] do not require a global clock as logic blocks are activated only after the arrival of their input data. The substitution of handshaking logic for the global clock enables asynchronous designs to cope with any amount of timing variation. While a synchronous system will fail if random variations cause the timing margins to degrade to negative values, a fully-handshaked asynchronous system will continue to function correctly (albeit at reduced performance). However, the asynchronous design paradigm poses significant challenges to both verification and test.

Globally-Asynchronous, Locally-Synchronous (GALS) [12] is a compromise between synchronous and fully asynchronous designs. In GALS design, a processor is partitioned into several components each with its own clock. A synchronization mechanism is used when a signal travels between clock boundaries. Drophso et al. [8] proposed dividing a CPU into four clock domains where the clock frequency can be varied in each domain independent of the others. While the main focus of GALS is to avoid the problem of distributing a global clock over a large die and to increase a microprocessor's power efficiency, an extreme version of GALS can potentially reduce a microprocessor's susceptibility to random timing variations by reducing the number of paths that need to run synchronously.

Other approaches have been proposed to addressing reliability and variability without deviating too far from conventional synchronous design. Austin [1] proposed DIVA, an on-line checking mechanism inserted into the retirement stage of a microprocessor. The checker allows instructions to commit to the architected state only if there are no errors in the computation. In the case of an error, the checker repairs the CPU state and restarts the pipeline. Such errors may arise from multiple sources; for example, soft errors, design errata, and timing failures.

Ernst et al. [9] proposed Razor which tunes the supply voltage of a circuit by monitoring the error rate during circuit operation, thereby eliminating the need for voltage margins and to improve power efficiency. The combinatorial logic block is speculatively evaluated and they use a Razor flip-flop that double-samples pipeline stage values, once with a fast clock and again with a time-borrowing delayed clock. If the value changes during the double sampling a recovery is performed. Uht et al. [16] proposed TIMERTOL design methodology which has one combinatorial logic block that is overclocked and multiple safely clocked blocks of the same logic. Using multiple check logic blocks can check all overclocked computation with hardware blocks that are safely clocked. In the context of our work, overclocking refers to situations in which a path fails to run at the target frequency due to random timing variations in device parameters.

In addition to increasing the magnitude of random timing variations, shrinking device sizes are also expected to lead to more rapid wear-out and increased susceptibility to soft errors. Mukherjee et al. [13] proposed redundant multi-threading to cope with failures due to soft errors. Srinivasan et al. [15] proposed dynamic reliability management, which dynamically varies processor's voltage and frequency in accordance with the workload demands, which improve reliability at the expense of throttling performance.

## 7. Conclusions and Future Work

In this paper we presented a quantitative evaluation of how device parameter variations impact full-chip timing. A major contribution of this paper is that we studied the effects of timing variations on FUBs from the Intel® Core™ Duo microprocessor design database. Our first set of experiments studied the effects of uncorrelated random variations where cell delays vary independent of each other. Our results showed that as timing variability is increased from 0% to 50%, timing margins degrades slowly at first, and then more rapidly. This is due to the effects of combining two probability distributions: the natural distribution of timing margins with the distribution due to random variations. The net effect of random timing variations is to make the steep timing wall built by the designers much more gradual. Furthermore, results from this work show that device variability affects larger FUBs

more than smaller FUBs; the number of independent paths is proportional to the FUB size and as the path count increases the probability of a slow path due to device variability also increases. High performance microprocessor design trends require designers to craft steep timing walls for extracting the highest performance and to use large FUBs to handle the increasing design complexity. Both these trends, unfortunately, are likely to exacerbate the impacts of timing variability on future high performance chip designs.

We also studied the effects of systematic variations. We show that negative margin paths get even worse with systematic variability as multiple gates along a critical path are systematically affected in the same manner.

Our preliminary work focused on quantifying the impact of timing variations within FUBs. There is significant amount of future work lies ahead. As an immediate follow-on, while this paper assumed Gaussian distributed random variability, it is not clear what the probability distribution of variability in future process generations will be. How do the conclusions in this paper change with different probability distribution functions? The results in this paper show that the overall impact on chip timing varies vastly depending on the extent of device variability. Since it is not clear what the extent of variability in future process will be, it is expedient to explore multiple solutions each targeted for a given variability level. This work ignored how process variability in long wires impacts chip timing. We believe that it is important to model the interconnection delay variability on chip timing when considering inter-FUB connectivity.

More recently chip industry shifted its focus from high performance uniprocessors to chip multiprocessors (CMPs) with several simpler cores. The reduction in core complexity will likely reduce the length as well as the number of critical paths. Simpler cores are less susceptible to variability. For example, in a single core design, the worst case path in the core limits the maximum clock frequency (FMAX). For a multi-core design, each core could theoretically run at an individual core FMAX. Since each core could run a separate FMAX value, the overall chip clock frequency would improve relative to a single-core design, which would mitigate the impact of variations on mean FMAX. However, with increasing core count in CMPs the uncore logic that enables communication between the cores will grow non-linearly. One interesting research question is how does device variability impact the uncore logic of CMPs?

Through this work we show that a significant change in the design philosophy is necessary to address problems associated with device variations.

## 8. Acknowledgements

This work benefited greatly from the valuable insights provided by Shekhar Borkar, Keith Bowman, Shih-Lien Lu and Chris Wilkerson at Intel Corporation. We would like to thank Chip Laub for providing us access to the design database and setting up with tools to do the timing analysis. We would like to thank John Shen for providing guidance throughout this research.

## 9. References

- [1] T. Austin. DIVA: A Reliable Substrate for Deep Submicron Microarchitecture Design. In *Proceedings of the 32nd International Symposium on Microarchitecture*, pages 196-207, November 1999.
- [2] S. Borkar, T. Karnik., S. Narendra, J. Tschanz, A. Keshavarzi and V. De. Parameter variations and impact on circuits and microarchitecture. In *proceedings of Design Automation Conference*, Pages 338 – 342, June 2003.
- [3] S. Borkar. Designing reliable systems from unreliable components: the challenges of transistor variability and degradation. In *IEEE MICRO*, 25(6):10-16, November 2005.
- [4] K.A Bowman, S.G Duvall and J.D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration, In *IEEE Journal of Solid-State Circuits*, 37(2):183-190 Feb 2002.
- [5] K. A. Bowman, X. Tang, J.C. Eble, and J.D. Meindl. Impact of extrinsic and intrinsic parameter fluctuations on CMOS circuit performance. In *IEEE Journal of Solid-State Circuits*, 35(8):1186-1193, Aug. 2000.
- [6] K. A. Bowman, S.B. Samaan, and N.Z. Hakim. Maximum Clock Frequency Distribution Model with Practical VLSI Design Considerations. In *Proceedings, 2004 International Conference on Integrated Circuit Design and Technology*, pages 183-191, 2004.
- [7] J. Cortadella, A. Kondratyev, L. Lavagno, C.P. Sotiriou. Coping with the Variability of Combinational Logic Delays. In *Proceedings of the 22nd International Conference on Computer Design*, pages 505-508, October 2004.
- [8] S. Dropsho, G. Semeraro, D.H. Albonese, G. Magklis, and M.L. Scott. Dynamically Trading Frequency for Complexity in a GALS Microprocessor. In *Proceedings of the 37th International Symposium on Microarchitecture*, pages 157-168, December 2004.
- [9] D Ernst, N.S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and Trevor Mudge. Razor: A Low-Power Pipeline Based on Circuit-Level Timing Speculation. In *Proceedings*

- of the 36th International Symposium on Microarchitecture, pages 7-18, December 2003.
- [10] D. Harris and S. Naffziger. Statistical Clock Skew Modeling With Data Delay Variations. In *IEEE Transactions on VLSI Systems*, 9(6):888-898, December 2001.
- [11] S. Hauck, Asynchronous Design Methodologies: An Overview. In *Proceedings of the IEEE*, 83 (1), 69-93, January 1995.
- [12] A. Iyer and D. Marculescu. Power and Performance Evaluation of Globally Asynchronous Locally Synchronous Processors. In *Proceedings 29th International Symposium on Computer Architecture*, pages 158-170, May 2002.
- [13] S. S. Mukherjee, M. Kontz, and S. K. Reinhardt. Detailed design and evaluation of redundant multi-threading alternatives. In *Proceedings, 29th Annual International Symposium on Computer Architecture*, pages 99-110, May 2002.
- [14] M. Orshansky and K. Keutzer. A General Probabilistic Framework for Worst Case Timing Analysis. In *Proceedings 39th Design Automation Conference*, pages 556-561, June 2002.
- [15] J. Srinivasan, S.V. Adve, P. Bose and J.A. Rivers. The Case for Lifetime Reliability-Aware Microprocessors. In *Proceedings 31st International Symposium on Computer Architecture*, pages 276-287, June 2004.
- [16] A. Uht. Achieving Typical Delays in Synchronous Systems via Timing Error Tolerant. University of Rhode Island TR-032000-0100, March 2000.