Statistical Static Timing Analysis Flow for Transistor Level Macros in a Microprocessor

Vivek S Nandakumar¹, David Newmark², Yaping Zhan², Malgorzata Marek-Sadowska¹
¹Univ. of California, Santa Barbara, CA USA
²Advanced Micro Devices Inc, Austin, TX USA
vivek@ece.ucsb.edu, mms@ece.ucsb.edu

Abstract
Process variations are of great concern in modern technologies. Early prediction of their effects on the circuit performance and parametric yield is extremely useful. In today’s microprocessors, custom designed transistor level macros and memory array macros, like caches, occupy a significant fraction of the total core area. While block-based statistical static timing analysis (SSTA) techniques are fast and can be used for analyzing cell based designs, they cannot be used for transistor level macros. Currently, such macros are either abstracted with statistical timing models which are less accurate or are analyzed using statistical Monte-Carlo circuit simulations which are time consuming. In this paper, we develop a fast and accurate flow that can be used to perform SSTA on large transistor and memory array macros. The delay distributions of paths obtained using our flow for a large, industrial, 45nm, transistor level macro have error of less than 6% compared to those obtained after rigorous Monte-Carlo SPICE simulations. The resulting flow enables full-chip SSTA, provides visibility into the macro even at the chip level, and eliminates the need to abstract the macros with statistical timing models.

Keywords
Statistical Static Timing Analysis (SSTA), transistor level macros, Monte-Carlo simulations.

1. Introduction
Technology scaling has brought the rapid increase in process variability. Its effects on device performance have compelled the industry to transition to statistical techniques for timing sign-off. Traditional corner case analysis (CCA) over constrains the design and often sets stringent, unrealistic timing specifications. Moreover, for technology nodes smaller than 65nm, these overestimated timing bounds compensate the performance improvement due to device scaling. Statistical static timing analysis (SSTA) has gained widespread acceptance for standard cell based designs, as it removes a significant portion of pessimism introduced by conventional approaches like CCA while accounting for global (inter-chip) and local (intra-chip) process variations. The application of both path based and block based SSTA have been shown to be advantageous [1] [2] [7] for cell based ASICs for which reusable timing models could be easily characterized. However, performing SSTA for transistor level macros is non-trivial due to the difficulty in generating appropriate timing models. For example, cache blocks in microprocessors are not made of standard cells. A news article [15] reports that more than 50% of a multi-core processor and more than 30% of each core are occupied by cache arrays and custom, transistor level blocks both of which are not standard-cell based. For custom macros, designers have significantly more transistor level options to improve performance with less overhead than in the case of gate level circuits. Moreover, such macros occur in portions of the processor which are extremely timing critical where variations could adversely affect the final performance.

Recently, two papers [4] [12] addressed the problem of performing statistical timing for macros. In [4], the authors propose a methodology to generate statistical models for large IP macros that can be used in SSTA flows allowing fast analysis. While this method is shown to be accurate, it works only for macros with gates as basic units and cannot be easily adapted for transistor level macros. Additionally, that method requires a prior knowledge of the long and short paths that could become critical. It is applicable for IP macros with fixed gate and routing implementations which is not the case of custom macros. In [12], a method of variation aware transistor level timing analysis for macros is described. The authors build statistical models for macros at a chip level of hierarchy. These abstract models are obtained by using two techniques referred to as pruning and compression. Their compactness is achieved at the expense of the timing and variability information losses. In addition, the device sensitivities to variation are modeled as constants instead of functions of input slews. These approaches introduce some inaccuracy in predicting chip level performance degradation due to variations. To overcome these problems and to perform accurate variation analysis of transistor level macros, designers currently use rigorous, but time consuming Monte-Carlo SPICE (MCS) simulations of selected paths. The simulation run times are of the order of hours/path. It is impractical to perform such MCS simulations on all paths in the macros and is therefore required that the designers have a prior knowledge of the top paths that could potentially become critical.

It hence becomes necessary to have a fast statistical timing analysis flow for transistor level macros that can compute the delay distributions due to process variations of all paths in the macros with accuracy close to MCS simulations. In this paper, we propose a flow that finds a solution to this problem. Our method first groups the macro transistors into logic gates called xcells by applying the Xblock technique described in [6]. Xblock does not approximate any transistor or wire information while grouping, which is vital in preventing any accuracy loss. For all extracted xcells we build timing library considering both
inter-chip and intra-chip process variations using a SPICE circuit simulator. The library is later used by an industrial-standard timing engine to perform block based SSTA of the macro. We also provide a methodology to use SSTA for memory array (cache) composed of SRAM bit-cells arranged in columns.

Using our flow, we obtain statistical distribution for endpoint delays of all paths in a large, industrial macro with accuracy of at least ~95% compared to MCS simulations with almost no runtime overhead. The only computational effort is spent on characterizing the extracted xcells. Our work also provides a comparison between the statistical timing analysis results and MCS simulation on transistor level industrial designs. We not only verify our statistical timing flow, but more importantly, we also validate the proposed macro characterization technique. We eliminate the necessity of using the macro abstraction with statistical timing models, and we provide a flow with visibility into the macro even while performing SSTA at the chip level. One of the important features of this flow is that the characterized variation aware library for transistor level macros can be readily used by an existing commercial timing engine.

This paper is organized as follows: Section 2 provides a discussion of the process variations considered in this paper. Section 3 discusses the proposed flow. Section 4 analyzes the statistical delay distributions obtained using our flow and Section 5 concludes the paper.

2. Global and local process variations

In this paper, we consider the manufacturing process induced variations [13] of parameters that affect circuit performance. Threshold-voltage ($V_{th}$), effective channel length ($L_{eff}$), oxide thickness ($T_{ox}$), mobility ($\mu$), and dopant concentration ($C$) are some of the variation parameters that significantly affect performance. Their variations result in designs with a wide spread of critical path delay distributions that may degrade the timing yield, i.e. decrease the fraction of manufactured chips that meet the timing constraints. For analysis purposes, parameter variations are usually classified into two categories: the inter-chip or global and the intra-chip or local variations. In case of globally varying parameters, their values are the same for all devices on the chip. In other words, the devices within a chip are considered to be in close proximity, so each parameter would have the same amount of variation. Conversely, each device within the same chip takes a different random value for locally varying parameters.

Variation parameters may depend on each other. For instance an increase in $T_{ox}$ also increases $V_{th}$. We use principal component analysis to convert the dependant variation parameters into independent principal components (PCs). In general, the delay of a path $D$ due to variation is given by [14]

$$D = D_0 + \sum_{i=1}^{n} \sigma_{g_{i}} \cdot R(G_{i}) + \sum_{i=1}^{n} \sum_{k=1}^{p} \sigma_{l_{ik}} \cdot R(L_{ik})$$  \hspace{1cm} (1)$$

where,

- $D$ – path delay,
- $D_0$ – nominal delay (without variation),
- $\sigma_{g_{i}}$ – standard deviation of the delay distribution due to the global random variable $R(G_i)$; $i$ varies from 1 to $n$ - number of principal components,
- $\sigma_{l_{ik}}$ – standard deviation of the delay distribution due to the local random variable $R(L_{ik})$; $k$ varies from 1 to $p$ - number of transistors.

In equation (1) the local delay component is dependent on the number of transistors. The fact that, for global variations all transistors within a macro are completely correlated and for local variations all transistors within a macro are completely uncorrelated (statistically independent) helps re-write equation (1) as follows [14]:

$$D = D_0 + \sum_{i=1}^{n} \sigma_{g_{i}} \cdot R(G_{i}) + \sum_{i=1}^{n} \sigma_{l_{i}} \cdot R(L_{i})$$  \hspace{1cm} (2)$$

In equation (2), the number of local random variables $R(L)$ is reduced from $np$ to just $n$ showing that $R(L)$ does not depend on the number of transistors in the macro. This is a useful result because in a macro, the number of transistors $p$ could be in millions.

For equation (2) to hold, it is necessary that all the devices in a macro are uncorrelated. However, this might not be the case in reality as the transistors, especially closely placed, might exhibit certain level of spatial correlation which is layout dependent. Our flow can be modified to handle spatial correlations with the same $O(n)$ complexity by using a layout gridding approach [3] [4].

3. Our approach

The proposed SSTA flow developed for transistor macros is shown in Figure 1. It consists of two major steps.

1. Transistor level macro is converted to gate level blocks called xcells using the Xblock procedure [6].
2. Variation aware library is characterized for these xcells using the variation aware SPICE models.

SSTA engine determines delay distributions for all paths in the macro using the variation libraries. The validation step compares the SSTA results with MCS results. The timing yield step estimates the required arrival time based on the most critical path due to variation and reverse PCA step provides information on the variation sensitivities of each path that can be used for design optimization.

3.1. Convert a macro to the xcells

The conversion of the macro’s transistor level netlist into a netlist of xcells is performed by an internal tool Xblock [6]. Xblock was developed to facilitate hierarchical, transistor level static timing analysis using industrial block-based timing analyzers. It takes as input a transistor level GDSII layout of a macro and obtains a logic (verilog format)
and parasitic netlists (spef format) as outputs that can be used by a static timing engine. The logic netlist consists of xcells each of which contains transistors that are source/drain connected to its output node. Xblock also automatically generates control files for all the inferred xcells to drive the characterization engine for both setup and hold analysis. For certain special xcells (like bit-column) the control file is manually generated to handle complex constraints (like bitline pre-charge in a memory cell). An average xcell other than the bit-column typically consists of 10-15 transistors.

Xblock currently facilitates fast and accurate timing analysis for large industrial macros including memories through a block-based STA engine, providing visibility within the macro while performing chip level STA. Our proposed flow extends the usage of Xblock to generate xcells from transistor level macros that are suitable for SSTA library characterization.

### 3.2. Variation aware device models

In order to characterize variation libraries we first need SPICE device models that are variation aware. Transistor models corresponding to the typical (TT) corner case and the $3\sigma$ variation ranges of different parameters are provided by the foundry. The variation parameters (like $V_{th}$, $L_{eff}$, $T_{ox}$, $\mu$) are dependent on each other. We perform Principal Component Analysis [9] [11] and convert these parameters into uncorrelated Principal Component (PCs). The foundry provides a correlation matrix $C_x$ that specifies the correlations between various interdependent input variables $X_m$. Here, $X_1 = V_{th}$, $X_2 = T_{ox}$, $X_3 = \mu$, … $X_m = L_{eff}$. A linear eigen value decomposition produces a diagonal eigen value matrix $\lambda$ and eigen vector matrix $P$ that satisfy the equation:

$$ P \ C_x \ P^T = \lambda $$

(3)

$C_x$ is an $m \times m$, symmetric correlation matrix given by

$$ C_x = \begin{bmatrix} 1 & \ldots & C_{x1, x_m} \\ \vdots & \ddots & \vdots \\ C_{x_m, x_1} & \ldots & 1 \end{bmatrix} $$

The eigen value matrix $\lambda$ is an $n \times n$ symmetric matrix with all other than the diagonal elements equal to 0.

$$ \lambda = \begin{bmatrix} \lambda_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \lambda_n \end{bmatrix} $$

Each column in the $m \times n$ eigen vector matrix $P$ is a principal component vector $P_{Gi} = [P_{G1}, P_{G2}, P_{G3}, \ldots P_{Gn}]^T$. Apart from the principal components being uncorrelated, PCA reduces [5] the number of dependant input variables $X_i$.
(i = 1 to m) to a much smaller number of principal components $PC_j$ (j = 1 to n). This significantly reduces the number of times each xcell has to be simulated while creating the variation library. The linear relation between the correlated input variables $X_i$ and the uncorrelated principal components $PC_j$ is given by

$$X_i = \sum_{j=1}^{n} P_{ij} R(\lambda_j)$$

where,

$$R(\lambda_j) = N(\mu = 0, \sigma^2 = 1) * \lambda_j$$

$N(\mu = 0, \sigma^2 = 1)$ represents a normal probability distribution with zero mean and variance = 1.

The local variation parameters of the transistors within a macro can be spatially correlated. Our method can be modified to handle such a case by using a correlation matrix $C_{xx}$ instead of the $C_x$ and applying PCA on it. $C_{xx}$ characterizes the parameter correlations of transistors placed in one grid to the parameters of transistors in other grids of a macro [3] [4]. If there are $G$ grids and $m$ dependent input variables, $C_{xx}$ will be of a size $Gm \times Gm$ instead of $m \times m$.

Each variation parameter used in our device model file is a function of 5 PCs obtained by solving equation (3) for a correlation matrix of size 15 x 15. From our experiments, we find that using 5 PCs yields good results with reasonable runtime overhead. In our model we have a total of 10 PCs, 5 PCs for global and 5 PCs for local variations.

$$\text{PC 1}$$

- 1 Nominal characterization point

$$\text{PC 2}$$

- $-3\sigma$ to $+3\sigma$ characterization points

$$\text{PC N}$$

- $2N$ characterization points

Figure 2: 2N values of PCs for which each xcell in the variation library is characterized.

3.3. Variation library characterization

After converting a macro to gate level xcell netlist using Xblock, a timing library is generated. It contains delay/output slew look-up tables for each pin in the xcell and for all PCs. This is accomplished using an automated characterization engine that performs SPICE simulations to obtain delays for a wide range of input and output conditions (slew/load).

Each xcell in the library is characterized at 2*N + 1 different values of the PCs stored as 2*N + 1 look-up tables; $N$ is the number of PCs. Figure 2 illustrates this process. In our case with 10 PCs, each xcell has 21 tables in the library. One table corresponds to the nominal case, with all 10 PCs set to their mean (nominal) values. The other 20 tables are generated for xcells characterized at the $+3\sigma$ and $-3\sigma$ values for the 10 PCs. Using the delay values from the nominal, $+3\sigma$, and $-3\sigma$ look-up tables for each PC in the library, the delay sensitivity of each path to different PC variation is computed by the statistical timer [10].

4. Results

We used an industrial 45nm design macro for experiments. It contains 100 unique xcells and bit-columns. The total number of transistors in the macro is of the order of a few millions. The delay values shown in the figures and throughout the rest of the paper are normalized to 1GHz for proprietary reasons, but that scaling in no way affects our message.

4.1. Monte-Carlo Vs SSTA

The macro studied in this paper has the critical path (read access line through the SRAM bit-column) that requires at least 2 hours to complete MCS. This makes it impractical to perform MCS using variation device models for all top paths in the design. SSTA allows us to see these distributions and hence analyze the effects of variation on all paths of the design which is the most important goal achieved in this work. Figure 3 shows Cumulative Density Functions (CDF) of the top critical path slacks in the design.

![Figure 3: CDF of the slack values of some of the top critical paths.](image)

In order to run MCS to validate SSTA, 100 paths of different lengths in terms of xcell number are pruned out from the macro netlist. A few representative paths are listed in Table 1. The extracted layout parasitics are also included during MCS simulations. Table [1] compares the mean and the standard deviation ($1\sigma$) of the endpoint delays (arrival time) between SSTA distributions and distributions obtained after 1000 runs of MCS simulations. Figure 4 compares the delay distributions of the most critical path in the macro obtained by MCS simulations and our SSTA flow.
The maximum error percentage of the total variation (L+G = Local + Global) of SSTA reported delay is ~6%. The table also shows the runtime for MCS simulations. The runtime for the entire SSTA, which computes the distributions for all paths in the macro, is almost negligible, less than 3 minutes for a macro of ~600,000 transistors.

### 4.2. Statistical Vs Conventional Corner Case

Figure 5 compares the delay results for the critical path in the macro obtained both statistically and using conventional (non-statistical) corner case analysis (CCA).

The great normal curve above the line (marked *Global*) is the delay distribution obtained for 1000 MCS simulation runs only considering global variations and setting the local PCs to zero. The three small normal curves below the line (marked *Local*) are delay distributions obtained for 1000 MCS simulation runs by setting the global components to be equal to +3σ, mean, -3σ and randomly varying only the local PCs. For each MCS run, each PC takes different values for each device in the path. Note that +3σ, mean and -3σ are the variation points for which the corner case SPICE models variation is close to global variation and in some cases, the local component is dominant. Table 2 shows the sensitivities of the xcells in each path to the original variation parameters like $V_{th}$, $L_{eff}$, $T_{ox}$ and $\mu$. Thus, our flow gives the designer a tool to identify variation sensitive areas in the design, even if they lie within a macro, and fix or optimize them if possible.

### Note:

Values in the column “Total” denote 1σ delays obtained from the delay distributions considering both local and global components of PCs.

$$\sigma^2(\text{Total}) = \sigma^2(\text{L}) + \sigma^2(\text{G})$$.

### Table 1: Comparison of MCS and SSTA path delays

<table>
<thead>
<tr>
<th>Xcells /path</th>
<th>Monte-Carlo SPICE (MCS) Simulation</th>
<th>SSTA</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Runtime</td>
<td>Mean (μ1) Delay (ps)</td>
<td>Local (L) (1σ Delay) (ps)</td>
</tr>
<tr>
<td>10</td>
<td>2 hrs</td>
<td>212</td>
<td>5.3</td>
</tr>
<tr>
<td>1</td>
<td>15 min</td>
<td>70.7</td>
<td>1.5</td>
</tr>
<tr>
<td>2</td>
<td>30 min</td>
<td>16</td>
<td>0.65</td>
</tr>
<tr>
<td>11</td>
<td>2.5 hrs</td>
<td>298</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>20 min</td>
<td>46</td>
<td>4.2</td>
</tr>
</tbody>
</table>

In Table 1, we show the global and local components of the delays. For long paths (10 xcells), the global component dominates the local component due to the cancellation of device mismatches along the path. For short paths, the local variation is close to global variation and in some cases, the local component is dominant. Table 2 shows the sensitivities of the xcells in each path to the original variation parameters like $V_{th}$, $L_{eff}$, $T_{ox}$ and $\mu$. Thus, our flow gives the designer a tool to identify variation sensitive areas in the design, even if they lie within a macro, and fix or optimize them if possible.
are usually designed (commonly referred to FF (Fast), TT (Typical) and SS (Slow)). The three circles on the line are endpoint delay values of the same path obtained by using non-statistical, CCA SPICE models.

Hence, the worst case analysis performed using the slow model (SS), would give pessimistic results. Also note that the variance (3σ – mean) for local variation is much less compared to global (~60ps compared to ~120ps). It is expected to be much more significant for short paths where the cancellation effect of mismatched devices is less prominent.

With block-based SSTA, we could get the endpoint delays of all paths in the design with almost no overhead in run-time. For all three plots in Figure 6, the STA results correspond to -3σ, mean and +3σ delay values obtained from the delay distributions of each path in the macro. Each point on the x = y line represents the delay of a path obtained by running conventional STA individually for the corresponding deterministic corner cases. A point above the line indicates that the particular path has a statistical delay that is slower than that obtained by its equivalent corner model. The majority of path delays obtained by deterministic models are either too slow (compared to -3σ SSTA) or too fast (compared to +3σ SSTA). This again confirms that FF is too fast and SS is too slow not just for the critical path shown in figure 5, but for almost all top paths in the design (Figure 6).

4.3. Difference in path sensitivity due to variation

Using SSTA for this macro reveals paths that are not too critical at typical operating condition (mean) but become very critical at the extremes of variation (3σ). Without SSTA, designers would use the deterministic corner model to obtain delay/slack values which is close to the mean of the delay/slack distributions obtained using SSTA. This could sometimes be misleading as the designer is not aware of the real situation where new paths that are not the most critical could become critical when variations are considered. Figure 7 shows slack distributions obtained for the two top critical paths of the macro using our SSTA. Consider the two paths marked by pointers. It can be seen that the path1 has a smaller mean slack than path2 and is hence less critical from a designer’s perspective who will only see these values using a deterministic approach. However the criticality of the two paths change with respect to -3σ (worst case) suggesting that path1 is more sensitive to variation than path2.

**Figure 6:** The median line - 45 degree (x = y), x and y-axis represent time delays. Points in the graph above the median line represent greater delays and points below the line represent smaller delays compared to the SSTA values.

It can be seen from figure 5 that the corner case models are over margined. For instance, the SS corner delay is 1246ps while the 3σ (worst-case) point of the global distribution is only 1120ps. Corner models are typically constructed by reusing the parameters generated from one circuit to another [8]. In order to make sure that the models are valid for a wide range of circuits and also to account for the error % in calculating the truly worst case corner in the presence of several varying parameters, some margin is intentionally forced which makes the SS too pessimistic.

**Figure 7:** slack values for paths in the macro that change criticality due to difference in variation sensitivities.
4.4. Timing Yield

Without SSTA, designers would fix the critical paths to meet a frequency that is much greater than the target frequency needed for a particular yield. Figure 8 shows the CDF of the most critical path whose period defines the frequency of the entire macro. 50% yield point corresponds to the nominal time period of 1000ps at which the SSTA was performed for this macro. For instance, if we need to achieve a 70% yield at 1000ps, SSTA results suggest a minimum required arrival time (RAT) of 1005ps to be set on the critical path based on the slack difference. This design has a large positive slack of 53ps even for a 99.8% yield, suggesting that the design has been over-optimized. Figure 9 compares the slack values obtained for all paths by setting the minimum RAT from SSTA at 70% and 99% yield points and a conservative RAT used by designers to fix the design before using our SSTA flow. Figure 9 shows a clearly large margin that is pessimistic even to achieve a 99% yield.

![Figure 8: Timing yield plot – CDF of the most critical path of the design obtained using SSTA with RAT = 1000ps.](image)

![Figure 9: Slack values of all paths of the design obtained by performing SSTA with RAT chosen from 90%, 99% yield points and conventional corner case TT.](image)

4.5. Characterization runtime

Characterizing a variation library at 2N + 1 points as described in section 3.3 for each xcell even though is a one-time effort, is still time consuming. However, libraries generated this way for different PC corners are more accurate since the sensitivities are determined from look-up table delay values obtained by actual circuit simulation rather than analytical formulations. The library generation time linearly increases with the number of points at which each xcell in the library is characterized. For each point of characterization, a look-up table is generated for every xcell in the library. For a 5% compromise in accuracy, the library characterization time can be significantly reduced (Table 3).

<table>
<thead>
<tr>
<th>Number of tables/xcell</th>
<th>Accuracy</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Char 1: 21 tables</td>
<td>100%</td>
<td>~8hrs</td>
</tr>
<tr>
<td>Char 2: 11 tables</td>
<td>97%</td>
<td>~4hrs</td>
</tr>
<tr>
<td>Char 3: 5 tables</td>
<td>95%</td>
<td>~2hrs</td>
</tr>
</tbody>
</table>

Char 1: We consider all 10 PCs.
Char 2: We only consider 5 global PCs and set a correlation of 1 between transistors to represent global variations. We use the same PCs and set a correlation of 0 between transistors to represent local variations.
Char 3: Assuming the delay variance obtained for each PC variation is symmetrical about the mean delay value, we characterize only N+1 tables instead of 2N+1 for the 5 global PCs. The xcells are characterized only at the mean and ±3σ points of the PCs instead of the mean, ±3σ and +3σ points shown in Figure 2.

5. Conclusion

Macros are custom designed circuit blocks that are usually present in very critical sections of the microprocessor to maximize performance, power and/or yield. Transistor level macros have a very large optimization space that is difficult for designers to manually explore. As a result, custom, transistor macros derive maximum benefit from SSTA. In order to make correct design decisions especially at smaller technology nodes where the effect of variation on performance is large, macro designers currently rely on either non-statistical approaches like CCA which are pessimistic or on extensive circuit level simulations and several runs of MCS analysis, which is extremely time consuming. In this work, we show experimentally that CCA results are indeed pessimistic. While it’s almost impossible to do MCS simulations on all paths in a macro or even on a few top critical paths, our SSTA flow provides distributions for all paths in the macro (including SRAM arrays) that are close to SPICE results (~95% accuracy). The flow also helps pin-point the paths and their components that are more sensitive to a particular source of process variation (V_{th}, T_{ox}, μ, L_{eff}) which can be used for design optimization.

While this flow is developed mainly for transistor macros, it can easily be modified to be used for any cell based macro (without applying Xblock). The flow hence allows fast statistical timing analysis of an entire chip without abstracting transistor macros.

6. Acknowledgements

We would like to thank the members in the AMD macro design team for helping us verify our SSTA flow.
7. References


