On the Statistical Memory Architecture Exploration and Optimization

Charalampos Antoniadis*, Georgios Karakonstantis†, Nestor Evmorfopoulos*, Andreas Burg† and George Stamoulis*
*Dept. of Electrical & Computer Engineering, University of Thessaly, Volos, Greece
{haadonia, nestevmo, georges}@inf.uth.gr
†Telecommunications Circuits Lab (TCL), EPFL, Lausanne, Switzerland
{georgios.karakonstantis, andreas.burg}@epfl.ch

***Abstract*—The worsening of process variations and the consequent increased spreads in circuit performance and consumed power hinder the satisfaction of the targeted budgets and lead to yield loss. Corner based design and adoption of design guardbands might limit the yield loss. However, in many cases such methods may not be able to capture the real effects which might be way better than the predicted ones leading to increasingly pessimistic designs. The situation is even more severe in memories which consist of substantially different individual building blocks, further complicating the accurate analysis of the impact of variations at the architecture level leaving many potential issues uncovered and opportunities unexploited. In this paper, we develop a framework for capturing non-trivial statistical interactions among all the components of a memory/cache. The developed tool is able to find the optimum memory/cache configuration under various constraints allowing the designers to make the right choices early in the design cycle and consequently improve performance, energy, and especially yield. Our, results indicate that the consideration of the architectural interactions between the memory components allow to relax the pessimistic access times that are predicted by existing techniques.**

I. Introduction

Today there is an increasing demand for on chip memory capacity with caches occupying already 40%-60% of the die area in chip multiprocessors [9] while studies are predicting that this fraction will exceed 70% by 2017 [5]. Such trends may have been enabled by the aggressive shrinking of transistor sizes, however the worsening of parametric variations in deep sub-micron technologies introduced new design challenges that pose a threat to future systems. It is apparent that variations in transistor characteristics lead to a large spread in chip frequency and leakage from die to die by as much as 30% and 20x, respectively [9]. Yield losses may have been limited through corner based design and the adoption of guardbands according to worst case scenarios. However, the worsening of variations (especially intra-die) have made it more difficult to predict the worst case scenario. In addition, such a case will be an extremely rare event thus resulting in large overheads even for the good instances. The situation is even worse in memories which are built using tight design rules and consist of substantially different building blocks.

Therefore, traditional methods are considered as nonviable options and new frameworks and tools have been proposed to identify solutions that can lead to better energy, performance and yield early in the design cycle. Statistical circuit level simulations with Monte Carlo like runs of transistor level netlists have been used to predict local process variations of circuits. These may have added a parametric yield dimension to the exploration flow before tape out, however the prohibitively large amount of simulations required for achieving an appropriate level of of confidence have given rise to other enhanced statistical methods. In particular, variants of importance sampling and extreme value theory have been deployed to derive the tail statistics and predict the probability of failure (PF) of a memory array [3], [7], [17]. Based on such frameworks high level models that can predict the PF of an array [13], [1] were developed and helped in accessing early the trade-offs achieved by tuning various knobs in the memory design space, in particular associated with cell sizing. FinCANON is the most recent work in this area that allows the exploration of power, delay and process variations achieved by various options in FinFET based caches.

Although such works are very attractive there are some issues that are not addressed sufficiently. First, most memory analysis works have considered the bitcell alone neglecting the memory periphery [10], [8]. This may have reduced the sample size of the statistical simulations but unfortunately it may lead to unrealistic results that may not reveal all real issues. For instance, the read operation of the cell is not only affected by the capability of the cell to discharge the bitline but also by variations on the timing circuit that controls the activation of the sense amplifier and the row decoder that enables the word line activation. Accounting for the worst-case situation of each of these effects would lead to pessimistic estimates: a worst case cell instance is not necessarily in the same path as the worst-case row driver logic. Therefore, attention must be paid to architectural correlations between bitcells and all other memory building blocks. However, most importantly, even works that have not considered only the bitcell such as [9], [16] have assumed only a single critical path in the overall memory. But even by doing so, such studies have failed to capture the full memory statistics accurately especially under local process variations since they still assumed that all paths are the same.In order to avoid the pessimism inherited by a careless worst-case combination, statistical analysis requires to account carefully for *all different* paths that exist in a memory. MemoryVAM is a recent [22] attempt to address those issues. However, it does not provide a systematic methodology to evaluate the impact of parameter variation across many different cache memory organizations within a reasonable execution time since it is based on time consuming transistor level simulations.

To this end, in this paper, we present an integrated framework for the estimation of the delay and power spreads of memory components under process variations. Our tool is built upon the widely used CACTI infrastructure [20] which models cache memory architectures. By doing so we have access to models of the various memory components that are widely acceptable in the computer architecture community. The contributions of our work can be summarized as follows:

- Enable the statistical simulation of such components by enhancing the existing current and RC models used for delay modeling in CACTI. By doing so we are able in the first simulation phase of the tool to extract statistical sampled variants of all the different devices and of all the components of the memory.
- By performing a sensitivity analysis of the different memory components forming the critical path of the memory array under process variations we enable the capturing of all the dependencies and consider all architecture correlations.
- The developed framework takes into consideration the possible interactions between the cell and all other connecting blocks in any memory organization. This is achieved by reconstructing of the full memory parametric pass/fail behavior using the collected statistical data of the individual components.
- Reformulate the traditional optimization objective of CACTI using as objective function the yield under a given set of constraints rather than the conventional performance and power.

978-3-9815370-4-8/DATE15/©2015 EDAA

- By placing a constraint either on a system level metric (e.g., cycle time and/or power) or on stability metrics (e.g., pass/fail checkpoints), we focus on the yield loss component caused by parametric deviations in the devices.
- Accelerate the yield prediction by implementing an importance sampling [21] technique within CACTI.

The rest of the paper is organized as follows. Section II exemplifies the organization of a cache memory and provides some background material. Section III presents the steps we followed to make possible the variability analysis while Section IV describes the statistical methodology we used to estimate the cache access delay considering different paths. Section V illustrates the importance sampling technique we exploited to enhance the statistical simulation and section VI describes the formulation of the yield optimization problem we assumed. Section VII analyzes the results and provides more insights on the achieved trade-offs. Finally, conclusions are drawn in Section VIII.

II. Cache Organization - Background

Memories and caches are storage arrays with a hierarchical structure as depicted in Figure 1. In the rest of the paper we describe such a memory structure as modeled in CACTI, however all applied methods can be applied to other memory structures as well. A cache memory, at the highest abstraction level, consists of N_{banks} identical banks. Every bank has a dedicated address/data bus, thus different banks can be accessed concurrently. A bank is composed of N_{subbanks} identical sub-banks which are activated one at a time in every access. In turn, every sub-bank consists of multiple identical mats. A mat is a self-contained, small memory array made of 4 identical sub-arrays and associated predecoding logic, with each sub-array being a two-dimensional, with N_{rows} rows and N_{cols} columns, matrix of memory cells and associated peripheral circuitry. Every mat contains, in one of its four sub-arrays, a part of a word and during a cache access all mats in a sub-bank are activated to compose the complete word. Address and data are typically distributed to the mats using H-tree distribution networks.

The main metrics of interest for a memory are its access delay and leakage power. The delay of every component of a cache is modeled usually as a resistor in series with a capacitor (RC). Such models are extracted for each component of the cache namely the decoder, wordlines, bitline, sense amplifier, comparator, multiplexer drivers, output drivers, as well as wires and h-tree buffers. Leakage power(P_{leak}) on the other hand, is based on the estimation of leakage currents. At the architecture level a quick and sufficient picture of the leakage power profile can be provided by focusing on subtrheshold leakage component [14], [18]. Note that RC delay approximations entail the estimation of the equivalent resistance a.k.a the effective resistance R_{eff}. The effective resistance is the ratio of the drain-source voltage V_{DS} to the drain-source current I_{DS} of the driving transistor averaged across the switching interval of interest [19]. Subthreshold leakage current is the current through a transistor operating in the subthreshold region, i.e., when the gate-to-source voltages is below the threshold voltage (V_{th}). In the absence of variations, as assumed in conventional CACTI, the V_{DS} v.s. I_{DS} curve is known a-priori for a particular technology node, thus R_{eff} and leakage current I_{leak} are constants that can be drawn from technology cards.

III. Enabling Variability Aware Simulations

The deterministic handling of leakage current and RC delays in traditional architecture level simulators is not appropriate under parametric variations. In particular, intra- and inter-die variations in transistor characteristics transform every parameter such as V_{th} into a random variable (RV). This essentially leads to delay and leakage spreads of each memory component. This eventually means that the access delay of the memory consisting of the various components is not anymore deterministic as estimated by CACTI. If such a delay is used it may lead to many failing memories since they will not be able to meet the predicted access delay or many instances of the memory may be over-designed (due to pessimistic worst-case assumptions). Therefore, such variations need to be considered within the architecture memory simulator for estimating the delay and leakage spreads. Figure 2 depicts the steps that we applied in order to enable variability aware simulations (within CACTI).

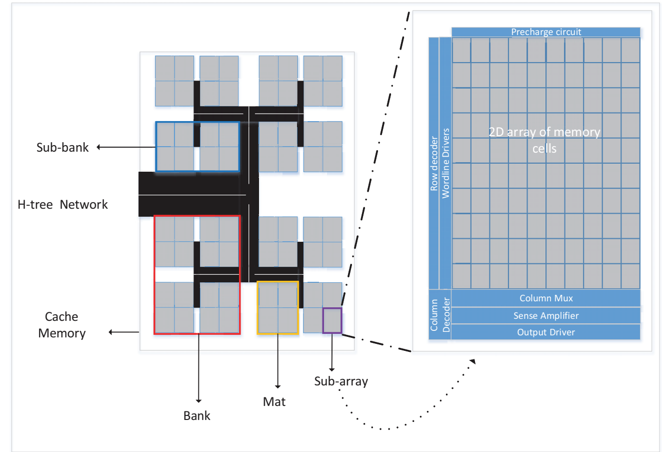


Fig. 1: Cache Organization. The figure illustrates a cache memory consisting of 4 banks where each bank is divided into 2 sub-banks and each sub-bank is divided into 2 mats.

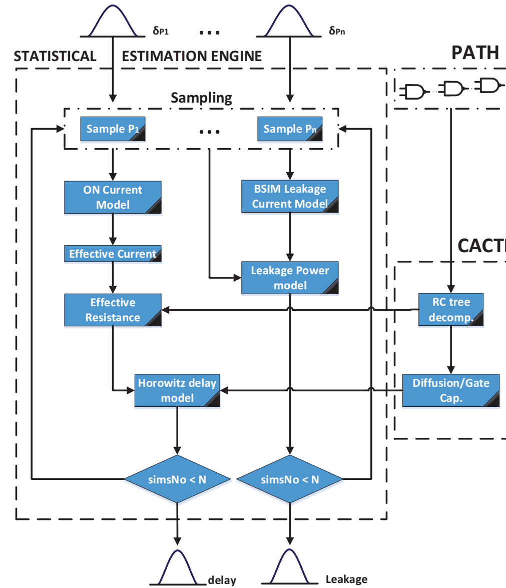


Fig. 2: Overall proposed flow of statistical manipulation of an access time in a memory array

In particular, we modeled I_{DS} using the alpha-power law model [15] shown in (1), which anyway captures short channel effects better compared to the Shockley model.

$$I_{DS} = \begin{cases} 0, & if\ V_{GS} \leq V_{th} \\ \frac{W}{L_{eff}} \frac{P_c}{P_u} (V_{GS} - V_{th})^{\alpha/2} V_{DS}, & if\ V_{DS} < V_{d0} \\ \frac{W}{L_{eff}} P_c (V_{GS} - V_{th})^{\alpha}, & if\ V_{DS} \geq V_{d0} \end{cases} \quad (1)$$

In this equation V_{GS} and V_{DS} are the gate-source and the drain-source voltages, respectively, P_c and P_u are constants and V_{d0} is given by:

$$V_{d0} \;=\; P_u(V_{GS} - V_{th})^{\alpha/2}$$

We determined $\alpha = 1.3$ to be in perfect agreement with ITRS [2] reports after several experiments for the 65nm technology node. By integrating this model into the memory simulator we were able to estimate the current spread under any Vth distribution. By doing so we can also estimate the R_{eff} spread based on variations since

$$R_{eff} \;=\; \frac{V_{DD}}{I_{eff}} \tag{2}$$

and since

$$I_{eff} \;=\; \frac{I_H + I_L}{2}, \tag{3}$$

where $I_H = I_{DS}(V_{GS} = V_{DD}, V_{DS} = \frac{V_{DD}}{2})$ and $I_L = I_{DS}(V_{GS} = \frac{V_{DD}}{2}, V_{DS} = V_{DD})$.

In addition, rather than using constant leakage numbers as in conventional CACTI we modeled the I_{leak} as

$$I_{leak} = \mu_{eff} * C_{ox} * \frac{W}{L} * u_t^2 * (1 - e^{(-\frac{V_{dd}}{u_t})}) * e^{(\frac{-|V_{th}| - V_{off}}{n * u_t})} \tag{4}$$

where μ_{eff} is the carriers mobility, C_{ox} is the gate oxide capacitance per unit area, W/L is the aspect ratio of the transistor, u_t is the thermal voltage, V_{th} is the threshold voltage, n is the sub-threshold swing coefficient, and V_{off} is an empirically determined BSIM4.6.0 parameter.

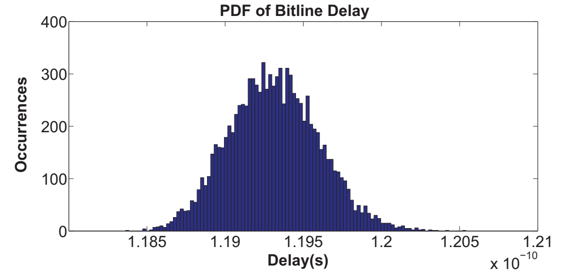
A. Component level Delay and Leakage Spreads

By integrating the above models for each transistor into CACTI we are able to estimate the delay and leakage spread of the different memory components. Eventually we are transforming a single sample of a modeled memory in CACTI with any number of transistors and component types into N different variants/samples. These samples differ from the original model in such a way that the parameter deviations due to variability are added to each transistor end eventually to each current I_{DS} and I_{leak}. Then by performing numerous Monte Carlo runs around a nominal value of specific characteristics (e.g., V_{th}, L, ...) we can estimate the corresponding delay- and leakage spreads σ_{Delay} and σ_{Ileak} of different components of the array.

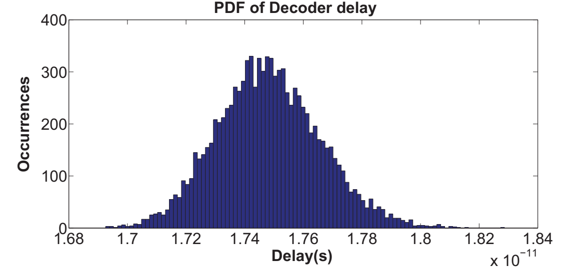
Figure 3 shows the distribution of the decoder & bitline delay under V_{th} and L variations obtained after 10000 MCs. In our simulation the variance of V_{th}, L and W obey the Pelgrom's rule [12]. For instance the σ_{Vth}/μ_{Vth} of the access transistor of a bitcell in 45nm technology, shown in figure 3a, with W = 58nm and L = 13.45nm was about 19%. Our results indicated negligible deviations of the impact of variations (SD/MEAN) to delay of different components compared to Spice simulations. For example considering a 256Kbit cache in 45nm technology node the impact of variations of bitline delay as shown in Figure 3a was of the same order of magnitude as the results presented in [22]. By assuming higher V_{th} spreads then as expected the impact is higher on access delay as shown figure 4. The tool can be used to explore various degrees of V_{th} variations and help in the evaluation of their impact on the whole memory architecture. If we assume a fixed V_{th}, the tool also estimates a very similar access delay as the original CACTI implementation.

IV. Architecture Level Statistical Analysis

Apart from component level statistics we need to also estimate the overall access delay which depends on the delay of several different components. Only a simulation of the entire memory will be able to consider all the architectural level dependencies between these different components. However, this will require unrealistically large amounts of time if all intra-die variations among all potential critical paths were to be considered. To achieve this objective within a feasible time and rather fast, we implemented a sensitivity based statistical analysis as described in [22]. Initially we analyze the sensitivity of critical path metrics to variation in certain components which are then combined to reconstruct the full memory statistics.



(a) PDF of Bitline delay. The mean value is μ = 1.19E-10 and the standard deviation is σ = 2.86E-13



(b) PDF of Decoder delay. The mean value is μ = 1.74E-11 and the standard deviation is σ = 1.76E-13

Fig. 3: Examples exhibiting component level delay spreads

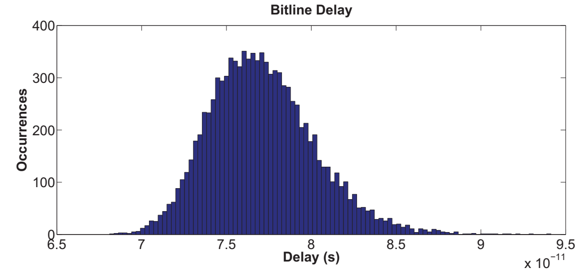


Fig. 4: PDF of Bitline delay. The mean value is μ = 7.71E-11 and the standard deviation is σ = 3.1E-12

We can distinguish two points that make the above methodology notable. First evaluating systematically all the possible paths existing in the entire cache and not only the critical path. Second by considering variability in subsets of all transistors the effectiveness of any importance sampling technique [6] increases, thereby improving the runtime of the developed statistical analysis framework.

A. Sensitivity Analysis to Cache access delay

In the first phase of the flow the analysis is limited to only one critical path assuming an instance from each one of the I components that are involved in a cache access. Below we write the critical path as the union of all sub-paths existing in the different components:

$$p_{crit} = p_1 \cup p_2 \ldots \cup p_I$$

The delay of the overall critical path can now be written as the sum of the delays of each sub-path p_i for $i = 1 \ldots I$:

$$D(p_{crit}) \;=\; \sum_{i=1}^{I} D(p_i) \;=\; D(p_{crit})_0 \;+\; \sum_{i=1}^{I} \Delta D(p_i),$$

where $D(p_{crit})_0$ is the nominal critical path delay and $\Delta D(p_i)$ is the sensitivity of critical path delay to process variations of the component i. The distribution of $\Delta D(p_i)$ is obtained as described in section III.

B. Architectural Correlation to Cache access delay

The second phase can be partitioned into the following consecutive steps:

- $\forall$ memory component $i = 1, \dots, I$ derive sensitivity distribution $\mathbf{\Delta D_i}$ of the cache access time. $\mathbf{\Delta D_i}$ is the variation of access time D shifted around nominal values.
- $\forall$ component $i = 1, \dots, I$, pick an M_i-sized[1] sample $\mathbf{Z_i}$ from the sensitivity distribution $\mathbf{\Delta D_i}$.
- $\forall$ path p_j from all M_C possible paths[2], a cache access delay sensitivity realization along all I involved components, can be described as: $p_j = (z_1, \dots, z_I)$, $j = 1 \dots M$, $z_i \;\varepsilon\; \mathbf{Z_i}$.
- $\forall$ path p_j evaluate access delay $D \simeq D_0 + \sum_{i=1}^{I} \mathbf{Z_D}(z_i)$, where D_0 is the nominal point. This rule provided good accuracy (below 1% RMS error) of all examples tested in [22].
- Select the worst of the M paths and assign that value to the memory observation.
- Repeat the above steps n times with different random sequences to generate memory statistics.

V. Exponent Monte Carlo

It is evident that consideration of all intra-die variations in the various paths and components require an extensive number of simulations. To limit their number we utilized an enhanced version of MC, called Exponent Monte Carlo (EMC) [21] which belongs in the class of variance reduction techniques known as *importance sampling* and was conceptually presented in [4].

The principle is to sample $\mathbf{x_1}, \dots, \mathbf{x_N}$ from $g_{\mathbf{X}}(\mathbf{x}) = f_{\mathbf{X}}^{(1-\gamma)}(\mathbf{x})$, where $\mathbf{x} = (x_1, x_2, ..., x_n)$ [3] is a n-dimensional vector of all independent transistor parameters of a circuit and $f_{\mathbf{X}}(\mathbf{x}) = f_{X_1}(x_1) f_{X_2}(x_2) \dots f_{X_n}(x_n)$ is the multivariate probability distribution of $\mathbf{x}$. For $\gamma > 0$, EMC samples less likely observations which define the tails of $f_o(h(\mathbf{x}))$ more precisely than the bulk. After experimenting with different γ values between 0 and 1, γ = 0.9 needed the fewest simulations in order to obtain a particular yield and level of confidence when accounting for delay defects.

Below we provide a very convenient exploitation of EMC showing step by step how $g_X(x) = f_X^{(1-\gamma)}(x)$ can be transformed into an equivalent normal distribution assuming that the natural distribution $f_X(x)$ is normally distributed $N(\mu, \sigma)$ too.

More specifically raising f_{X_x} to $1 - \gamma$ and then setting $\sigma' = \frac{\sigma^{1-\gamma}}{(\sqrt{2\pi})^{\gamma}}$ we can obtain $g(x) \sim N(\kappa\mu, \sigma')$, where $\kappa = (\sqrt{1-\gamma})\frac{\sigma^{-\gamma}}{(\sqrt{2\pi})^{\gamma}}$. This means, since we have a Gaussian random generator, that we can obtain a random sample x' drawed from $g(x)$ and then return the real observation by setting $x = x'/\kappa$.

Proof: First the normal distribution with mean μ and standard deviation σ is

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5}$$

Then we raise $f(x, \mu, \sigma)$ to $1 - \gamma$ and we obtain $g(x)$.

$$g(x) = \frac{1}{\frac{\sigma^{(1-\gamma)}}{(\sqrt{2\pi})^{\gamma}}\sqrt{2\pi}} e^{-\frac{(x-\mu)^2(1-\gamma)\frac{\sigma^{-2\gamma}}{(\sqrt{2\pi})^{2\gamma}}}{2\sigma^2\frac{\sigma^{-2\gamma}}{(\sqrt{2\pi})^{2\gamma}}}} \tag{6}$$

[1] M_i is the number all different paths inside component i

[2] All M_C bitcells are in unique paths. $M = \#mats.in.a.subbank * \#cols.in.a.subarray * \#rows.in.a.subarray$

[3] Bold letters represent vector variables while non-bold letters scalar variables

Then setting $\sigma' = \frac{\sigma^{1-\gamma}}{(\sqrt{2\pi})^{\gamma}}$ $g(x)$ can be written as:

$$g(x) = \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{((x-\mu)\kappa)^2)}{2\sigma'^2}}, \tag{7}$$

where $\kappa = (\sqrt{1-\gamma})\frac{\sigma^{-\gamma}}{(\sqrt{2\pi})^{\gamma}}$. Finally we can write $g(x)$ as

$$g(x) = \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{(x'-\mu')^2)}{2\sigma'^2}}, \tag{8}$$

where $x' = \kappa x$ and $\mu' = \kappa\mu$.

Therefore it is sufficient to first pick a x' drawed from $N(\mu', \sigma')$ and then set $x = x'/\kappa$. ■

VI. Yield Optimization

In order to evaluate different cache architectures with the objective of optimizing the yield we run a brute-force search to evaluate

$$Arch_{best} = \underset{Arch}{\operatorname{argmin}}\, Yield(Arch) \tag{9}$$

where the yield of each candidate architecture $Arch = \{N_{banks}, N_{subbanks}, N_{rows}, N_{cols}\}$ is defined as the percentage of array realizations that meet both our leakage and performance constraints:

$$P_{leak}(Arch) \le P_{leak}^{target} \text{ and } D_{Pcrit}(Arch) \le D_{Pcrit}^{target} \tag{10}$$

VII. Evaluation

In this section we demonstrate different features of the proposed tool. We provide examples at both the block and system level of a cache to illustrate the capabilities of our tool. We selected 3 configurations of a 64K cache, in 65nm technology node that exhibit the same cache access time under nominal condition. Note that the σ_{Vth} of the cell access transistor, assuming a 6T bitcell model, occurred 16% of nominal V_{th} while the σ_{Vth} of the transistors that belong to the peripheral circuitry occurred 10% of nominal V_{th}, according to Pelgrom's rule. The experimental setup is shown in Table I. The deterministic delay of each component in each configuration is shown in Table II.:

TABLE I: Cache configurations tested

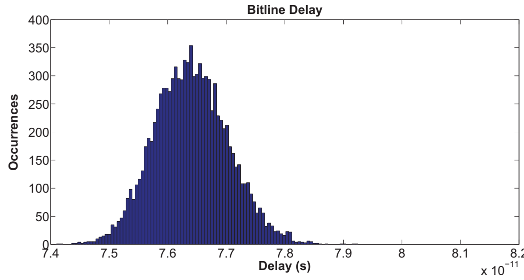
CFGs	N_{banks}	$N_{subbanks}$	N_{rows}	N_{cols}
cfg1	1	32	16	512
cfg2	1	2	32	128
cfg3	1	2	512	32

TABLE II: Bitline, decoder & H-tree network nominal delays of cache configurations under test

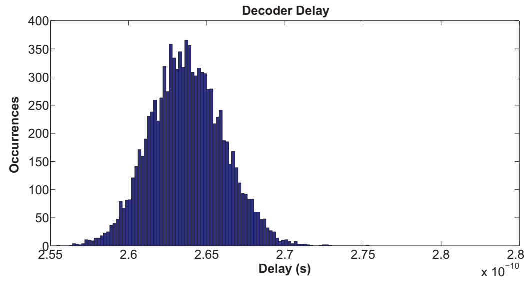
CFGs	BL delay (ns)	DEC+WL delay (ns)	HTREE delay (ns)	TOTAL ACCESS delay (ns)
cfg1	0.07	0.26	1.65	2
cfg2	0.05	0.26	1.67	2
cfg3	0.4	0.34	1.24	2

A. Block Level Evaluation

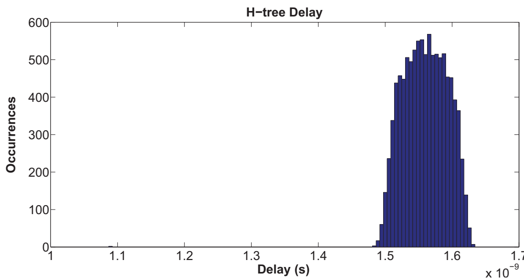
The proposed tool offers the capability to study the impact of parametric variations on critical path delay, at the component level. As an example we present the delay PDFs (Figure 5) of all components that are involved in a cache access of cfg1. We are referring to this stage in the paper as sensitivity analysis.



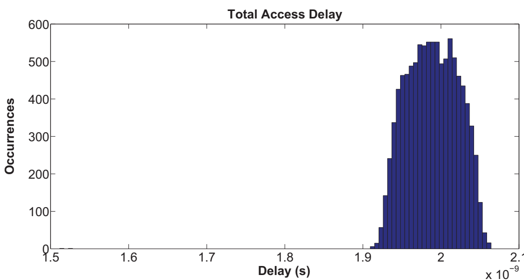
(a) PDF of Bitline delay. The mean value is μ = 7.64E-11 and the standard deviation is σ = 6.31E-13



(b) PDF of Decoder delay. The mean value is μ = 2.63E-10 and the standard deviation is σ = 2.33E-12



(c) PDF of Htree delay. The mean value is μ = 1.56E-09 and the standard deviation is σ = 3.28E-11



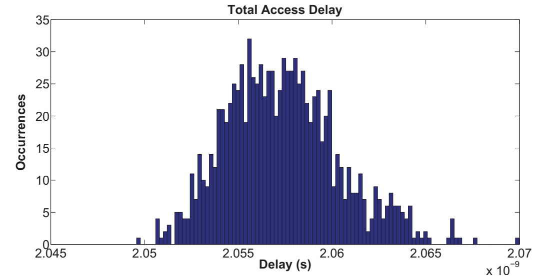
(d) PDF of Total Access Delay. The mean value is μ = 1.98E-09 and the standard deviation is σ = 3.29E-11

Fig. 5: PDF of each component involved in a cache access

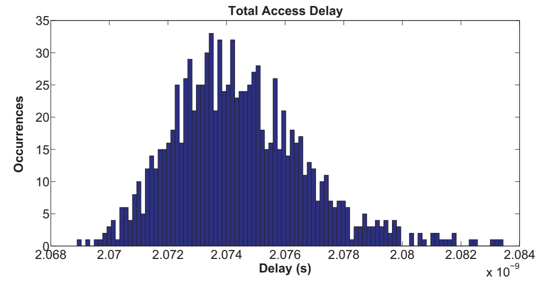
B. System Level Evaluation

In addition the proposed tool can be used to evaluate cache performance at the macro level taking into consideration the connectivity of different blocks as well. In Figure 6 we show the PDF of the total access time of the three different cache configurations under test following the methodology presented in Section IV.

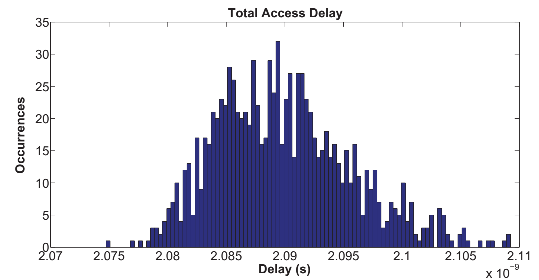
Furthermore, our results, which agree with the those in [22], indicate that the impact of parametric variations (σ/μ) is negligible in the case where architectural correlations between the components are considered. However, when only a single intentionally most-



(a) PDF of Total Access time of CFG1. The mean value is μ = 2.05E-09 and the standard deviation is σ = 3.02E-12



(b) PDF of Total Access time of CFG2. The mean value is μ = 2.07E-09 and the standard deviation is σ = 2.27E-12



(c) PDF of Total Access time of CFG3. The mean value is μ = 2.08E-09 and the standard deviation is σ = 5.75E-12

Fig. 6: PDF of each the total access time of the 3 configurations under test considering architectural correlations

pessimistic critical path is considered (as done in most of the existing works on variability), the impact appears to be one order of magnitude higher. For instance, for the considered cfg1, an overly-pessimistic estimate of $\sigma/\mu = 1.71E-2$ is obtained for a single critical path (see Figure 5d), while the proposed consideration of all possible paths yields $\sigma/\mu = 1.09E{-}3$ as shown in Figure 6a. Such results prove the overly pessimistic access delay estimated by current methods leading to costly guardbands which can be avoided by using our tool.

We can observe that the impact of variations on the mean and variance of the delay for different configurations is shown in Figure 6 and Figure 7. In particular, the mean delays of the three configurations are similar, while the spreads around the mean differ substantially. This motivates the use of our tool to find a cache architecture that exhibits the highest yield for a given set of constraints. For example, Figure 8 depicts the failing/passing observations of all three cache configurations after setting the yield-constraint to a critical path delay of T_{crit} = 2.07E-9s. As it can be observed cfg#1 exhibited the highest yield among the three configurations, allowing most of the realizations to pass the set delay target.

As we discussed in Section VI we could also set the yield criterion for meeting both leakage and delay constraints. In this case, as it is depicted in Figure 9, less realizations satisfy such constraints.

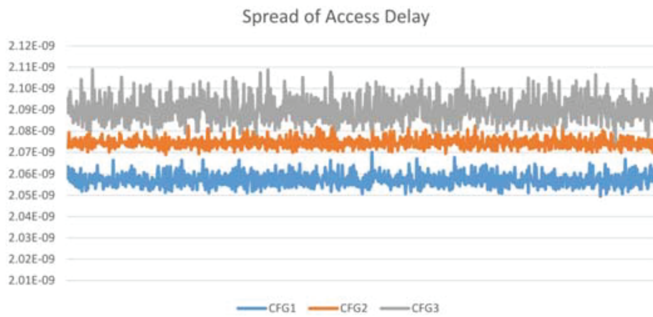


Fig. 7: Spreads of Access Delay in the configurations under test

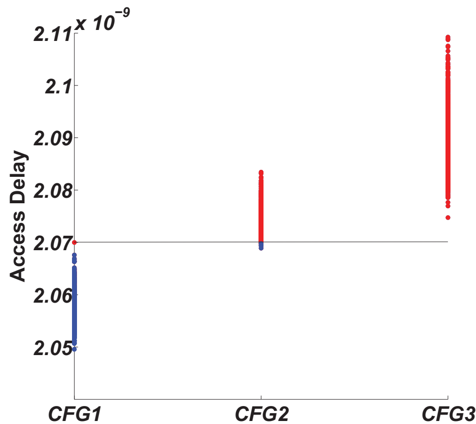


Fig. 8: The passing/failing observations of the tested configurations after setting a constraint T_{crit} = 2.07E-9s on critical path delay. The red-colored observation are the failing while the colored-blue the passing ones

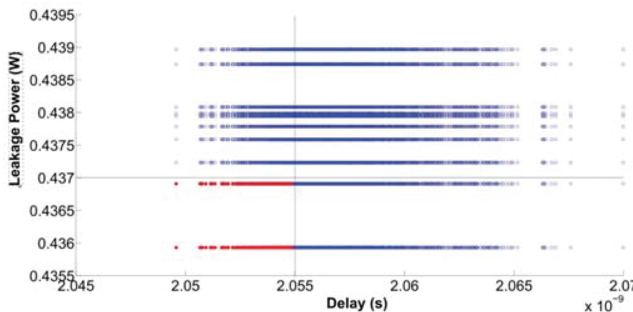


Fig. 9: The passing/failing observations of cfg#1 after setting two constraints, one on access time T_{crit} = 2.055ns and one on leakage power dissipation P_{crit} = 0.437W. The red-colored observation are the failing while the colored-blue the passing ones

Our tool with the help of EMC and the applied statistical analysis is able to provide all the above architecture wide statistics within a limited time. In particular, the PDF estimation of the access time considering the architectural correlations between the different components took only 40 to 55 sec (depending on the configuration) on an Intel Quad Processor running at 2.5GHz with 8GB Ram.

VIII. Conclusion

In this paper, we present a tool for capturing the effects of parametric variations on cache performance metrics systematically while considering all possible architectural interactions between the memory components. Our, results indicate that the consideration of such interactions allow to relax the pessimistic the pessimistic access times that are predicted by existing techniques. The proposed tool allows also to identify the configuration that could maximize the yield under certain performance/power constraints. The exploration of several other parameters and techniques (e.g.cell sizing, bit-cell types) for improving the yield are part of our future work. Nonetheless the developed tool can provide within limited time accurate non-pessimistic architecture wide statistics empowering the designers to come up with the right choises early in the design cycle.

Acknowledgments

This work was partially supported by the EU and the Greek State through ESPA 2013-2017, Action SYNERGASIA 2011, Project Code:11SYN 5 719, the EU OPEN-FET SCoRPiO grant (no. 323872) and the EU Marie-Curie DARE grant (no. 304186).

References

[1] R. Aitken *et al.*, "Worst-case design and margin for embedded sram," in *IEEE DATE*, April 2007, pp. 1–6.
[2] S. Association. International technology roadmap for semiconductors 2005. [Online]. Available: http://public.itrs.net/Links/2005ITRS/Home2005.htm
[3] G. K. Chen *et al.*, "Yield-driven near-threshold sram design," in *IEEE ICCAD*, pp. 660–666.
[4] B. Dierickx, *et al.*, "Propagating variability from technology to system level," in *Physics of Semiconductor Devices, 2007. IWPSD 2007. International Workshop on*, Dec 2007, pp. 74–79.
[5] S. Ganapathy *et al.*, "Informer: An integrated framework for early-stage memory robustness analysis," in *IEEE DATE*, March 2014, pp. 1–4.
[6] D. Hocevar *et al.*, "A study of variance reduction techniques for estimating circuit yields," *IEEE TCAD*, 1983.
[7] R. Kanj *et al.*, "Mixture importance sampling and its application to the analysis of sram designs in the presence of rare failure events." in *IEEE DAC*, 2006, pp. 69–72.
[8] A. Khajeh *et al.*, "Tram: A tool for temperature and reliability aware memory design," in *IEEE DATE*, ser. DATE '09, 2009.
[9] C.-Y. Lee *et al.*, "Fincanon: A pvt-aware integrated delay and power modeling framework for finfet-based caches and on-chip networks," *IEEE TVLSI*, vol. 22, no. 5, pp. 1150–1163, May 2014.
[10] S. Mukhopadhyay *et al.*, "Modeling of failure probability and statistical design of sram array for yield enhancement in nanoscaled cmos," *IEEE TCAD*, pp. 1859–1880, 2005.
[11] A. Papoulis *et al.*, *Probability, Random Variables, and Stochastic Processes*, ser. McGraw-Hill series: Communications and signal processing. Tata McGraw-Hill.
[12] M. Pelgrom, A. C. J. Duinmaijer, and A. Welbers, "Matching properties of mos transistors," *Solid-State Circuits, IEEE Journal of*, vol. 24, no. 5, pp. 1433–1439, Oct 1989.
[13] S. R. *et al.*, "Varius: A model of process variation and resulting timing errors for microarchitects," in *in IEEE Trans. on Semiconductor Manufacturing*, 2008.
[14] K. Roy *et al.*, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, Feb 2003.
[15] T. Sakurai *et al.*, "Alpha-power law mosfet model and its applications to cmos inverter delay and other formulas," *IEEE JSSC*, vol. 25, no. 2, pp. 584–594, Apr 1990.
[16] J. Samandari-Rad *et al.*, "Var-tx: A variability-aware sram model for predicting the optimum architecture to achieve minimum access-time for yield enhancement in nano-scaled cmos," in *IEEE ISQED*, 2012, pp. 506–515.
[17] A. Singhee *et al.*, "Statistical blockade: a novel method for very fast monte carlo simulation of rare circuit events, and its application." in *IEEE DATE*, 2007, pp. 1379–1384.
[18] S.Narendra and A. Chandrakasan, *Leakage in Nanometer CMOS Technologies*. New York:Springer, 2006.
[19] N. Weste *et al.*, *CMOS VLSI Design: A Circuits and Systems Perspective*, 4th ed. Addison-Wesley Publishing Company.
[20] S. J. E. Wilton *et al.*, "Cacti: An enhanced cache access and cycle time model," *IEEE JSSC*, vol. 31, pp. 677–688, 1996.
[21] P. Zuber *et al.*, "Exponent monte carlo for quick statistical circuit simulation," in *IEEE PATMOS*, 2010, pp. 36–45.
[22] ——, "Statistical sram analysis for yield enhancement." in *DATE*, 2010, pp. 57–62.