ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ, ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

ΑΝΑΖΗΤΗΣΗ ΣΤΟ ΧΩΡΟ ΣΥΝΘΕΣΗΣ ΚΑΙ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΜΝΗΜΩΝ CACHE ΚΑΤΩ ΑΠΟ ΤΗ ΔΙΑΚΥΜΑΝΣΗ ΠΑΡΑΜΕΤΡΩΝ

ON THE EXLORATION AND OPTIMIZATION OF CACHES UNDER PARAMETRIC VARIATION

Μεταπτυχιακή Διατριβή

Χαράλαμπος Γ. Αντωνιάδης

Επιβλέποντες Καθηγητές : Ευμορφόπουλος Νέστωρας
Επίκουρος Καθηγητής

Σταμούλης Γεώργιος
Καθηγητής

Τσομπανοπούλου Παναγιώτα
Επίκουρος Καθηγήτρια

Βόλος, Ιούλιος 2014

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ Η/Υ

Αναζήτηση στο χώρο σύνθεσης και βελτιστοποίησης μνημών cache κάτω από τη διακύμανση παραμέτρων

Μεταπτυχιακή Διατριβή

Χαράλαμπος Γ. Αντωνιάδης

Επιβλέποντες : Ευμορφόπουλος Νέστωρας
Επίκουρος Καθηγητής

Σταμούλης Γεώργιος
Καθηγητής

Τσομπανοπούλου Παναγιώτα
Επίκουρος Καθηγήτρια

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11η Ιουλίου 2014

.....	.....	.....
Ν. Μπέλλας	Π. Τσομπανοπούλου	Χ. Δ. Αντωνόπουλος
Αναπληρωτής Καθηγητής	Επίκουρος Καθηγήτρια	Επίκουρος Καθηγητής

Μεταπτυχιακή Διατριβή για την απόκτηση του Μεταπτυχιακού διπλώματος Ειδίκευσης «Επιστήμη και Τεχνολογία Υπολογιστών, Τηλεπικοινωνιών και Δικτύων» του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος Μεταπτυχιακών Σπουδών του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

……..

Αντωνιάδης Χαράλαμπος

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

To my family & my friends

Ευχαριστίες

Με την περάτωση της παρούσας εργασίας, θα ήθελα να ευχαριστήσω θερμά τους επιβλέποντες της διπλωματικής εργασίας κ. Ευμορφόπουλο Νέστωρα, κ. Σταμούλη Γεώργιο και κα Τσομπανοπούλου Παναγιώτα για την εμπιστοσύνη που επέδειξαν στο πρόσωπό μου, την άριστη συνεργασία, την συνεχή καθοδήγηση και τις ουσιώδεις υποδείξεις και παρεμβάσεις, που διευκόλυναν την εκπόνηση της πτυχιακής εργασίας.
Επίσης θα ήθελα να εκφράσω την ευγνωμοσύνη μου σε όλα τα παιδιά του εργαστηρίου Ηλεκτρονικής για την υπομονή τους να αντέχουν την οποιαδήποτε παραξενιά μου. Ιδιαίτερα θα ήθελα να ευχαριστήσω τον πολύ καλό μου φίλο-συνεργάτη Dr. Κωσταντή Νταλούκα για την οποιαδήποτε βοήθεια είτε ψυχολογική είτε τεχνική σε όλη την διάρκεια εκπόνησης της μεταπτυχιακής μου εργασίας.
Επίσης δεν θα μπορούσα να παραλείψω τον υποκινητή της όλης προσπάθειας μεταδιδάκτορα στο Πανεπιστήμιο EPFL(Ecole Polytechnique Federale de Lausanne) Dr. Καρακωσταντή Γεώργιο για τις πολύ εύστοχες υποδείξεις του και την πολύ ευχάριστη συνεργασία που είχαμε.
Τέλος, οφείλω ένα μεγάλο ευχαριστώ στην οικογένειά μου και στους φίλους μου για την αμέριστη υποστήριξη και την ανεκτίμητη βοήθεια που μου παρείχαν τόσο κατά την διάρκεια των σπουδών μου όσο και κατά την εκπόνηση της μεταπτυχιακής εργασίας.

Αντωνιάδης Χαράλαμπος
Βόλος, 2014

Contents

List of Tables

List of Figures

Acronyms

SRAM	Synchronous Random-Access Memory
PDF	Probability Density Function
EVT	Extreme Value Theory
CMOS	Complementary Metal-Oxide Semiconductor
MOSFET	Metal-Oxide Semiconductor Field Effect Transistor
FinFET	Fin-Shaped Field Effect Transistor
CMP	Chemical Mechanical Planarization
PF	Probability of Failure
MC	Monte Carlo
EMC	Exponent Monte Carlo
IS	Importance Sampling
LUT	Look Up Table
WID	Within-Die

Περίληψη

Στην παρούσια μεταπτυχιακή διατριβή παρουσιάζουμε την ανάπτυξη ενός εργαλείου με σκοπό την αξιολόγηση της επίδρασης στην απόδοση που έχει η μεταβλητότητα των παραμέτρων των τρανζίστορ στις κρυφές μνήμες και την αναζήτηση των σχεδιαστικών λύσεων που εγγυώνται την αξιόπιστη λειτουργία με την ελάχιστη επιβάρυνση. Πιο συγκεκριμένα ενσωματώσαμε κατάλληλα μοντέλα που κάνουν εφικτή την στατιστική ανάλυση της απόδοσης μιας μνήμης cache υπό την παρουσία της μεταβλητότητας των παραμέτρων στον προσομοιωτή HP Cacti και αξιολογήσαμε την απόδοσή διαφόρων αρχιτεκτονικών μνήμης cache. Επιτρέποντας στο εργαλείο να ψάξει την βέλτιστη σχεδίαση κρυφής μνήμης κάτω από μια σειρά περιορισμών, επιτρέπουμε στους σχεδιαστές να κάνουν νωρίς τις σωστές επιλογές στον κύκλο σχεδίασης ενός ολοκληρωμένου και κατα συνέπεια να βελτιώσουν την αξιοπιστία και την αποδοτικότητα της κρυφής μνήμης. Είνα προφανές ότι οι λύσεις που προκύπτουν διαφέρουν κατα πολύ από αυτές αν αγνοηθούν οι διακυμάνσεις των παραμέτρων, γεγονώς που καταδυκνύει την αξία του προτεινόμενου εργαλείου.

Abstract

In this master thesis a tool for capturing the effect of parametric variations on caches and identifying the design solutions that can ensure robust operation with minimum overhead is being presented. Specifically, parametric variations are being modeled within a popular cache simulation framework and cache performance is being evaluated by tuning various knobs at the circuit and architecture layer. By enabling the tool to find the optimum configurations under various constraints we allow designers to make the right choices early in the design cycle and consequently improve yield and efficiency. Apparently, such solutions differ substantially from the ones obtained if variations are neglected, further necessitating the use of the proposed tool.

Keywords: Memory design exploration and optimization, variation-tolerant

Chapter 1

Introduction

1.1 Motivation and Contribution

Today's trend is the increase of memory density for handling the huge number of data generated by the numerous applications integrated within a single device and their parallel execution on many core architectures. It is apparent that such a trend has increased the percentage of memory components within a system by more than 60% making them accountable for most of the overall area and power consumption of an embedded system. Such a trend may have been enabled up to now by the aggressive technology scaling allowing the realization of miniature and portable devices but unfortunately is under threat. Specifically, as devices are getting smaller it is becoming more difficult to fabricate them leading to variations in their characteristics. Such variations may lead to delay failures as well as read or write failures due to mismatches in the transistors of a memory cell.

Traditional design methodologies are assuming the worst process corner for all the cells of a fabricated chip and based on that are applying guardbands to the size of each cell and access time of the memory. However, such guardbands lead to overdesign increasing the cell and memory size and slowing down the performance. This has as a result lower memory density and performance contradicting with the trend described above. Such a reality worsens further as devices are getting smaller and intra-die variations within each transistor of a single cell and a chip are becoming more important than inter-die variations. Under this situation applying guardbands to each cell based on assumed worst case corner becomes even more inefficient since it affects also the good transistors that are rather more than the ones that are in the worst case corner. Although, industry has already realized such facts, corner based design continues to be the

popular variation aware method mainly due to the low complexity of the applied models and number of simulations needed to verify the designs. However, the overheads incurred of such overdesigns cannot be sustained as we move deeper to nanometer nodes and thus new tools are needed that could allow to capture the effect of variations and hence help designers to understand their impact and develop new more efficient remedies.

Furthermore the memory caches play a significant role in contemporary computer systems. They often belong to the critical path of the design, consume a non-negligible fraction of the total power and occupy more than 50% of the die area. So their robust design is very critical for the overall system performance.

To this end in this master thesis, we present a tool that captures the effect of parametric variations on caches and identifies the design solutions that can ensure robust operation with minimum overhead. Specifically, parametric variations are being modeled within a popular cache simulation framework and cache performance is being evaluated by tuning various knobs at the circuit and architecture layer. By enabling the tool to find the optimum configurations under various constraints we allow designers to make early in the design cycle the right choices that can improve yield and efficiency. Apparently, such solutions differ substantially from the ones obtained if variations are neglected, further necessitating the use of the proposed tool.
Overall, the contributions of the master thesis can be summarized as follows:

1. Enable the evaluation of the impact of variations on different components of a cache by incorporating a more accurate than alpha power law current model for sub-90-nm Metal-Oxide Semiconductor Field Effect Transistor (MOSFET)s [2].

2. Explore the sensitivity of different cache configurations under variations and evaluate how they affect the access time and leakage power.

3. Reformulate the optimization of caches using as objective function the yield rather than the conventional performance and power.

1.2 Related Work

There are modeling techniques for evaluating the impact of process variations on the performance and yield of memories. Such modeling

efforts necessary not only for revealing the impact of variations but also for developing efficient schemes for tacking them. In particular several chip area models [20], power/leakage models [20, 22], access-time models [45, 29, 38], and failure probability models [29, 38] have been published.

Mukhopadhyay et al. [29] derived analytic models of failure probabilities (access-time failure, read/write failure, and hold failure) of Synchronous Random-Access Memory (SRAM) cells due to random V_{th} variation and proposed a methodology for SRAM-cell design in order to enhance the yield of the memory array.

Sarangi et al. [38] extended the previous work with a more accurate access time error model by 1) comprising of a systematic component in total variation, 2) considering variation in L_{eff}, 3) modeling the maximum access time of a line of SRAM rather than a single cell; and 4) using the alpha-power current model that catches short channel effects more effectively than the square-law model.

The previous works [29, 38] focus on the modeling and analysis of variations in a single cell or an entire line of a cell array. However variations affects and other components of SRAM such as sense amplifier, row decoder etc. Furthermore neither of the previous works investigate the dependency between the sensitivity of critical path delay to variations and a cache architecture.

Samandari-Rad et al. [37] has taken the previous works one step further by 1) modeling and studying parametric variations except bitcell in other SRAM components (sense amplifier, row decoder, etc.) too and 2) extracting the sensitivity of critical path delay to variations for different cache architectures and then finding the cache architecture that exhibits the highest yield and meets the design requirements simultaneously.

Fin-Shaped Field Effect Transistor (FinFET)s have emerged as promising substitutes of conventional CMOS because of the superior control of short-channel effects and processing scalability. In [23] it is presented an integrated framework for simulation of power, delay, temperature, as well as process variations of FinFET-based caches.

All previous works assume that all paths behave the same. However a worst case cell instance is not necessarily in the same path with the worst case sense amplifier or the worst case row driver logic comprising a total worst case. So a careless worst case combination would lead to over-pessimistic results again.

Furthermore Monte Carlo (MC) is going to need prohibitively number of simulations to estimate the yield of a bitcell that presents low Probability of Failure (PF). Low PF involves the exploration of the tails of the real distribution while MC depicts effectively the bulk of the distribution.

Except from that, the higher the instance count of a certain memory island exist, the more tail exploration is required. For this reason importance sampling variants and statistical techniques drawn from Extreme Value Theory (EVT) have been deployed to derive the tail statistics and predict the memory probability of failure [9, 21, 40].

Zuber et. al.[50] has taken into consideration in their work all the above issues in statistical SRAM analysis for yield enhancement. They propose a methodology implemented in a prototype tool called Memory Variability Aware Modeling, or MemoryVAM in short that analyzes statistically any parameter that can be measured in a SPICE/SPECTRE testbench, such as access time, power, stability checks such as read voltage, and so on. Although the statistical approach can be considered extremely accurate, MemoryVAM doesn't provide a systematic methodology to evaluate the impact of parameters variation across different memory cache architectures within a reasonable time. Therefore there is a need for a tool that can allow the fast exploration of various cache configurations and provide insights regarding the failure probability of such configurations under variations. This will help designers to identify yield-friendly cache configurations with preferred performance and energy trade-offs early in the design cycle.

1.3 Outline

The rest of the master thesis is organized as follows. Chapter 2 provides an overview of a cache system. Chapter 3 presents the cache modeling under Parametric Variations. Chapter 4 describes the statistical methodology to extract the cache statistics. Chapter 5 analyzes the results and provides more insights on the achieved trade-offs. Finally, conclusions are drawn in Chapter 6.

Chapter 2

Cache overview

2.1 Introduction

Cache memories play a significant role in contemporary computer systems. They often belong to the critical path of the design, consume a non-negligible fraction of the total power and occupy more than 50% of the die area. So their robust design is very critical for the performance as well as the reliability of the overall system. In this chapter we describe the architecture of a conventional cache memory and we present the formulas for calculating Leakage Power dissipation and Access/Random-Cycle time in SRAMs.

2.2 Architecture

Data array organization

The data array, at highest level, is consisted of multiple identical banks. Every bank has its own address and data bus and can be concurrently accessed. A bank is composed of multiple identical sub-banks which are activated one at a time in every access. In turn, every sub-bank consists of multiple identical mats. During an access all mats in a sub-bank are activated and each mat contains a part of the accessed word in the bank. Each mat is a self-contained memory structure made of 4 identical sub-arrays and associated predecoding logic, with each sub-array being a two-dimensional matrix of memory cells and associated peripheral circuitry. An example of a layout of an array with 4 banks is shown in Figure 2.1. In this case, each bank is shown to have four sub-banks and each sub-bank four mats. Address and data are assumed to be distributed to the mats

on H-tree distribution networks.

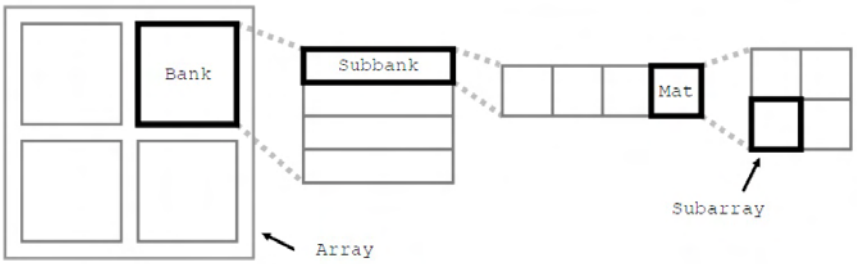


Figure 2.1: Layout of an example array with 4 banks. In this example each bank has 4 subbanks and each subbank has 4 mats.

Mat & Sub-Array Organization

A mat is always consisted of four sub-arrays and associated predecoding/decoding logic, which is located at the center of the mat and is shared by all four subarrays. The high level composition of all mats is as shown in Figure 2.2.

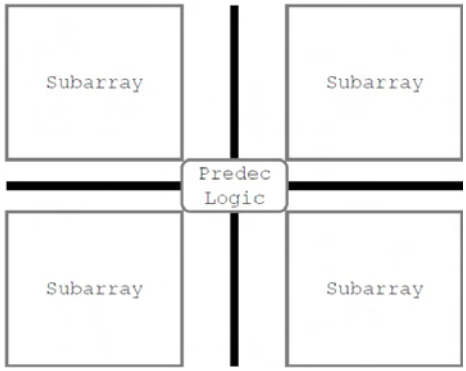


Figure 2.2: High-level composition of a mat.

Subsequently, the sub-array is consisted of a the following components.

- decoder
- wordlines
- bitlines
- sense amplifiers
- comparators

- multiplexor drivers
- output drivers

Figure 2.3 shows the high-level composition of a sub-array.

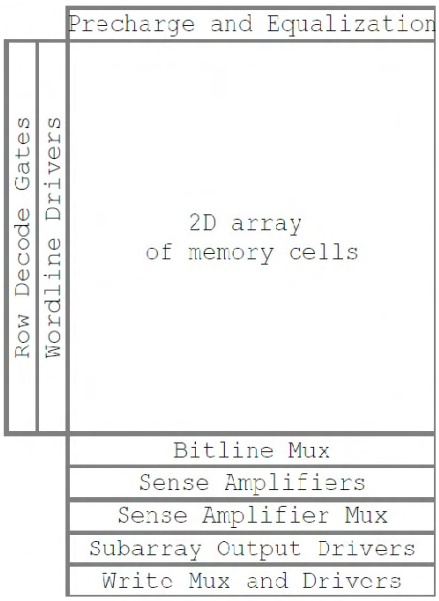


Figure 2.3: High-level composition of a subarray.

Routing to mats

Address and data are routed to and from the mats on H-tree distribution networks. H-tree distribution networks, which route address and data provide uniform access to all the mats in a large memory. Separate request and reply networks are assumed for ease of pipelining multiple accesses in the array. The request network carries addresses and data-in from the edge of the array to the mats while the reply network carries data-out from mats to the edge of the array. Request and reply networks have similar structures; the request H-tree network is divided into two networks:

1. The H-tree network from the edge of the array to the edge of the bank
2. The H-tree network from the edge of the bank to the mats.

The request H-tree network between the array edge and the banks is shown in Figure 2.4.

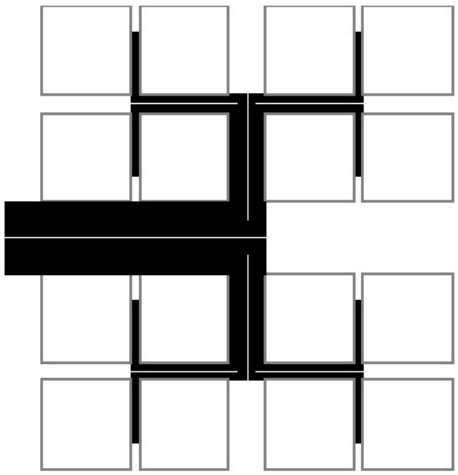


Figure 2.4: Layout of edge of array to banks H-tree network.

The H-tree network from the edge of the bank is further divided into two one-dimensional horizontal and vertical H-tree networks too.

2.3 Leakage Power Modeling

A chip consumes power even when it is not switching. In technology nodes, prior to 90 nm, leakage power was negligible in comparison to dynamic power. Unlike that in nanometer regime with low threshold voltages and gate oxides, leakage can account for as much as a third of total active power [44]. Leakage mechanisms include subthreshold conduction between source and drain, gate leakage from the gate to body, and junction leakage from source to body and drain to body, as illustrated in Figure 2.5[35, 41]. Subthreshold conduction is caused by thermal emission of carriers over the potential barrier set by the threshold. Gate leakage is a quantum-mechanical effect caused by the tunneling through the extremely thin gate dielectric. Junction leakage is caused by current through the p-n junction between the source/drain diffusions and the body.

Below we account only sub-threshold component of leakage current because our purpose is to provide a quick picture of the leakage power and not to compute it accurately, like SPICE.

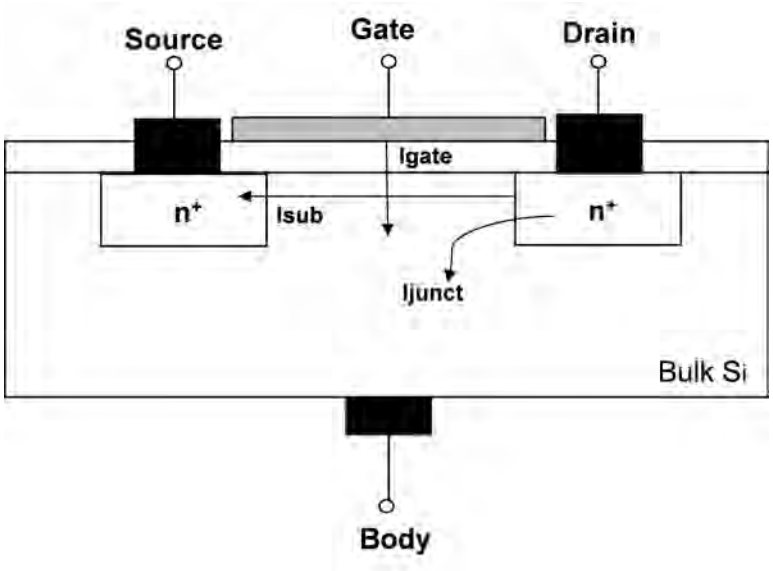


Figure 2.5: Leakage current components

Calculation of Leakage Current

We make use of the methodology presented in [45, 24, 26, 48] to simply provide an estimate of the drain-to-source sub-threshold leakage current for all transistors that are off with VDD applied across their drain and source. Based on the BSIM4.6.0 [48] equation for leakage in a MOSFET transistor, our leakage model of a single transistor is given by the following equation:

$$I_{leak} = \mu_{eff} * C_{ox} * \frac{W}{L} * u_t^2 * (1 - e^{(-\frac{V_{dd}}{u_t})}) * e^{(\frac{-|V_{th}|-V_{off}}{n*u_t})} \qquad (2.1)$$

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. The above equation is based on two assumptions:

1. $V_{gs} = 0$ — we only consider the leakage current when the transistor is off.

2. $V_{ds} = V_{dd}$ — we only consider a single transistor here; the stack effect and the interaction among multiple transistors are taken into account when we model the cell.

Leakage Power Calculation for CMOS

Figure 2.6 illustrates the leakage power calculation for an inverter. When the input is low and the output is high, there is subthreshold leakage through the NMOS transistor whereas when the input is high and the output is low, there is subthreshold leakage current through the PMOS transistor. In order to simplify our modeling, we come up with a single average leakage power number for each gate. Thus for the inverter, we calculate leakage as follows:

$$P_{leak-inv} = \frac{I_{leak-pmos} + I_{leak-nmos}}{2} * V_{DD} \tag{2.2}$$

where $I_{leak-pmos}$ is the subthreshold current for the PMOS transistor and $I_{leak-nmos}$ is the subthreshold current for the NMOS transistor.

Figure 2.7 illustrates the leakage power calculation for a NAND2 gate. When both inputs are high, the output is low and for this condition there is leakage through the PMOS transistors as shown. When either of the inputs is low, the output is high and there is leakage through the NMOS transistors. Because of the stacked NMOS transistors [25, 24] this leakage depends on which input(s) is low. The leakage is least when both inputs are low. Under standby operating conditions, for NAND2 and NAND3 gates in the decoding logic within the mats, we assume that the output of each NAND is high (deactivated) with both of its inputs low. Thus we estimate the leakage current of a NAND gate based on the leakage through its stacked NMOS transistors when both inputs are low. We consider the reduction in leakage due to the effect of stacked transistors [19] and calculate leakage for the NAND2 gate as follows:

$$P_{leak-nand2} = I_{leak-nmos} * SFnand2 * V_{DD} \tag{2.3}$$

where $SFnand2$ is derived based on knowledge of previous designs [25].

Leakage Power Equations

Below we present the equations to estimate the leakage power dissipation is SRAMs.

$$\begin{aligned} P_{leak} &= P_{leak-network} + P_{leak-peripheral-circuitry} + \\ &\quad P_{leak-mem-array} \end{aligned} \tag{2.4}$$

$$\begin{aligned} P_{peripheral-circuitry} &= (P_{leak-decoder} + P_{leak-senseamps}) * \\ &\quad N_{banks} N_{subbanks} N_{mats-in-subbank} \end{aligned} \tag{2.5}$$

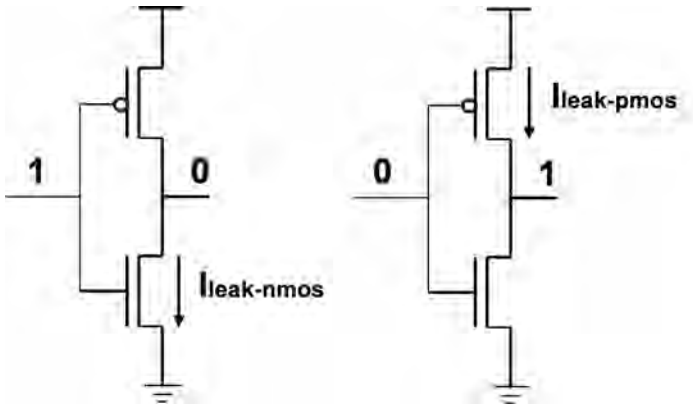


Figure 2.6: Leakage in an inverter

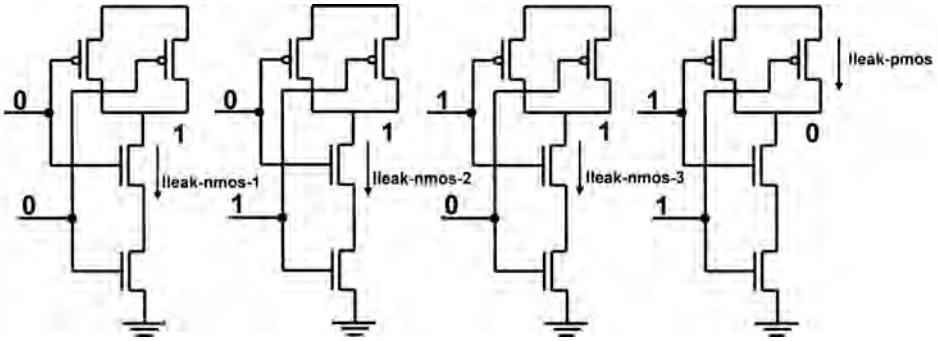


Figure 2.7: Leakage in a NAND2 gate

$$P_{leak-mem-cells} = N_{subarr-rows} N_{subarr-cols} P_{mem-cell} \tag{2.6}$$

$$P_{mem-cell} = V_{DD} I_{cell-leakage} \tag{2.7}$$

Figure 2.8 shows the subthreshold leakage paths in an SRAM cell when it is in idle/standby state [24, 26]. For a specific cell, the leakage current is given by the following equation:

$$I_{cell-leakage} = (n_N I_N + n_P I_P) k_{design} \tag{2.8}$$

n_N and n_P are the number of NMOS and PMOS transistors in the cell, and I_N and I_P are the leakage current of NMOS and PMOS. k_{degign} is the design factor determined by the stack effect and aspect ratio of transistors [48].

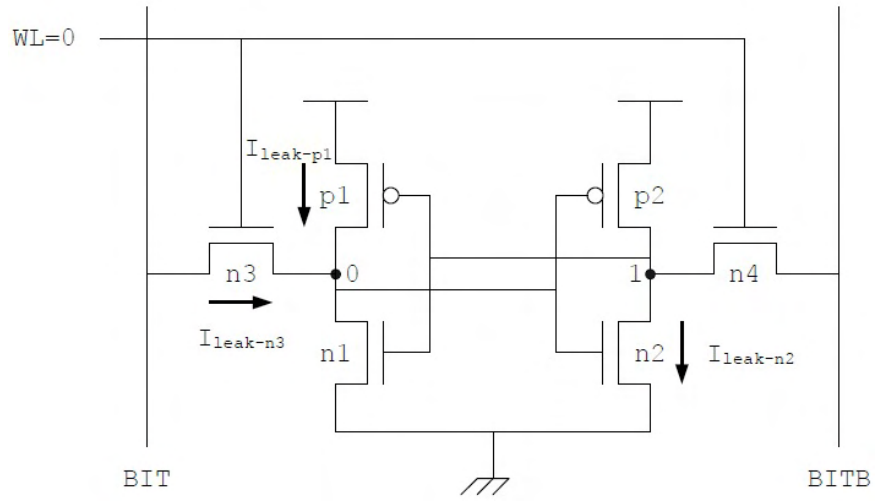


Figure 2.8: Leakage paths in a memory cell in idle state. BIT and BITB are precharged to VDD.

2.4 Logic Gate Delay Modeling

The delay of each cache component, namely decoder, wordlines, bitline, sense amplifier, comparator, multiplexor drivers, output drivers as well as wires and h-tree buffers was estimated by decomposing each component into several equivalent RC circuits, and using simple RC equations to estimate the delay of each stage. RC delay model is an approximation of the non-linear transistor's I-V and C-V curves with an average resistance a.k.a effective resistance and capacitance over the switching range of the gate. Below we present the models used to estimate R_{eff}, C_{drain} and C_{gate}.

Gate and diffusion Capacitance

The following formulas 2.9 and 2.10 provide the gate and drain capacitances for a single transistor respectively:

$$C_{gate}(W) = W * L_{eff} * C_{gate} + L_{poly} * L_{eff} * C_{polywire} \quad (2.9)$$

where L_{eff} and W is the effective length and width of the transistor, L_{poly} is the length of the poly line going into the gate, C_{gate} is the capacitance of the gate per unit area and $C_{polywire}$ is the poly line capacitance per unit area.

$$C_{drain}(W) = A_D * W * C_{diffarea} + P_D * C_{diffside} + W * C_{diffgate} \quad (2.10)$$

where $A_D = WL_D$ and $P_D = W + 2 * L_D$ are the source/drain area and perimeter respectively (see figure 2.9). $C_{diffarea}$, $C_{diffside}$, and $C_{diffgate}$ are process dependent parameters. $C_{diffgate}$ is the sum of the fringing fields and the oxide capacitances due to gate/source and gate/drain overlap (see figure 2.10).

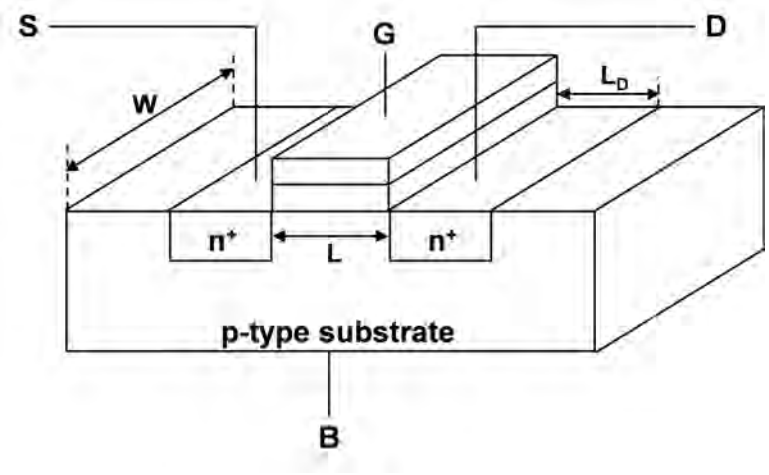


Figure 2.9: Gate and Source/Drain 3D-realization

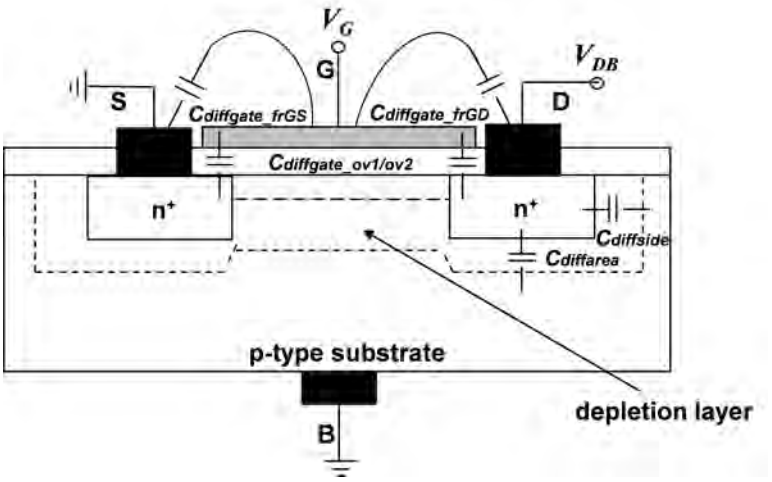


Figure 2.10: Gate and Source/Drain MOSFET Capacitance

Effective Resistance

The effective resistance R is the ratio of V_{ds} to I_{ds} averaged across the switching interval of interest. We calculate the effective resistance of a

transistor during switching as follows:

$$R_{eff} = \frac{V_{DD}}{I_{eff}} \tag{2.11}$$

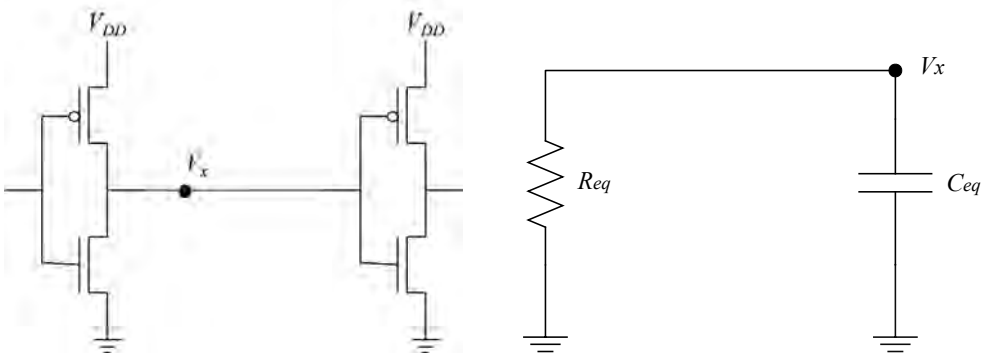
The effective drive current is calculated using the following formulas described in [32, 47]

$$I_{eff} = \frac{I_H + I_L}{2} \tag{2.12}$$

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

Equivalent RC Circuits

Figure 2.11(i) shows a typical first-order circuits. The time for node x to rise or fall can be determined using equivalent circuit of Figure 2.11(ii). Here, the pull down path (assuming a rising input) is substituted by a resistance, and the gate capacitance of the second stage and the drain capacitance of the first stage are replaced by a single capacitor. The resistances and capacitances are calculated as shown previously. In case a long wire connects two stages, parasitic capacitances and resistances of the wire are included in C_{eq} and R_{eq}.



((i)) schematic ((ii)) equivalent circuit

Figure 2.11: Example Stage

The delay of the circuit in Figure 2.11 can be estimated using the Horowitz timing model [18] (assuming a rising input):

$$delay = t_f * \sqrt{\left[\log\left(\frac{u_{th}}{V_{dd}}\right)\right]^2 + 2t_{rise}b(1 - \frac{u_{th}}{V_{dd}})/t_f} \tag{2.13}$$

where u_{th} is the switching voltage of the inverter, t_{rise} is the input rise time, t_f is the output time constant assuming a step input ($t_f = R_{eq} * C_{eq}$), and b is the fraction of the input swing in which the output changes (we used $b = 0.5$). For a falling input with a fall time of t_{fall}, the above equation becomes:

$$delay = t_f * \sqrt{\Big[\log\Big(1 - \frac{u_{th}}{V_{dd}}\Big)\Big]^2 + \frac{2t_{fall} * b * u_{th}}{V_{dd} * t_f}} \tag{2.14}$$

In this case, $b = 4$ is used.

The delay of a gate is defined as the time between the input reaching the switching voltage (threshold voltage) of the gate, and the output reaching the threshold voltage of the following gate. If the gate drives a second gate with a different switching voltage, the above equation need to be modified slightly. If the switching voltage of the switching gate is u_{th1} and the switching voltage of the following gate is u_{th2}, then:

$$\begin{aligned} delay = t_f * \sqrt{\Big[\log\Big(\frac{u_{th}}{V_{dd}}\Big)\Big]^2 + 2t_{rise}b(1 - \frac{u_{th}}{V_{dd}})/t_f} + \\ t_f\Big[\log\Big(\frac{u_{th1}}{V_{dd}}\Big) - \log\Big(\frac{u_{th2}}{V_{dd}}\Big] \end{aligned} \tag{2.15}$$

for a rising input, and

$$\begin{aligned} delay = t_f * \sqrt{\Big[\log\Big(1 - \frac{u_{th}}{V_{dd}}\Big)\Big]^2 + \frac{2t_{fall} * b * u_{th}}{V_{dd} * t_f}} + \\ t_f\Big[\log\Big(1 - \frac{u_{th1}}{V_{dd}}\Big) - \log\Big(1 - \frac{u_{th2}}{V_{dd}}\Big] \end{aligned} \tag{2.16}$$

for a falling input.

The bitline and comparator equivalent circuits, unlike the other cache components that can be approximated by simple first-order stages like the ones described previously, require more complex solutions. Figure 2.12 shows an equivalent circuit that can be used for the bitline and comparator circuits. The delay of this circuit is given by [36]:

$$T_{step} = \Big[R_2C_2 + \Big(R_1 + R_2\Big)C_1\Big]\ln\Big(\frac{u_{start}}{u_{end}}\Big) \tag{2.17}$$

where u_{start} is the voltage at the beginning of the transition and u_{end} is the voltage at which the stage is considered to have "switched" ($u_{start} >$

u_{end}). For the comparator, u_{start} is V_{dd} and u_{end} is the switching voltage of the multiplexor driver. For the bitline subcircuit, u_{start} is the precharged voltage of the bitlines (u_{pre}) and u_{end} is the voltage which causes the sense amplifier output to fully switch ($u_{pre} - u_{sense}$).

The above delay estimation assumes a step input. Bitline delay considering non-zero wordline (input) rise is given by:

$$delay = \begin{cases} \sqrt{2T_{step}\frac{V_{DD}-V_{th}}{m}}, & if\ T_{step} \leq 0.5\frac{V_{DD}-V_{th}}{m} \\ T_{step} + \frac{V_{DD}-V_{th}}{2m}, & if\ T_{step} > 0.5\frac{V_{DD}-V_{th}}{m} \end{cases} \tag{2.18}$$

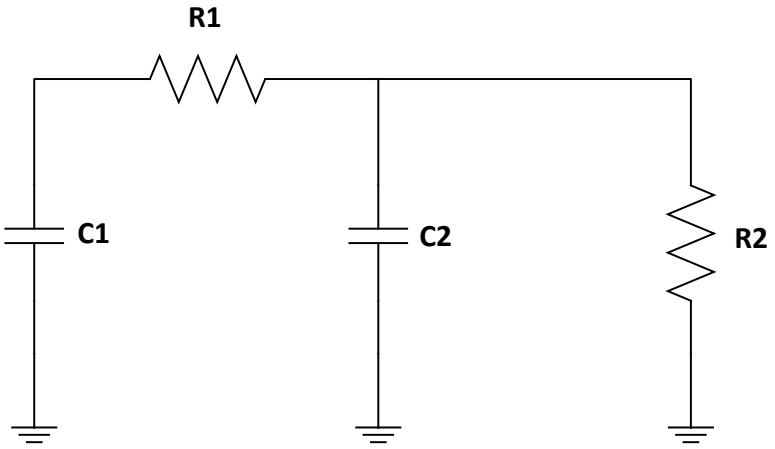


Figure 2.12: Equivalent circuit for bitline and comparator

Access Time Equations

Below we present the equations to calculate access time and random-cycle time in memory cache.

$$T_{access} = T_{request-network} + T_{mat} + T_{reply-netork} \tag{2.19}$$

$$T_{mat} = MAX(T_{row-dec-path},\ T_{col-dec-path}) \tag{2.20}$$

$$T_{row-dec-path} = T_{row-dec} + T_{bitline} + T_{senseamp} \tag{2.21}$$

$$T_{col-dec-path} = T_{col-dec} \tag{2.22}$$

Although critical path mostly involves row decoder path, there are certain partitions of data array, where column path act as critical because both paths are active during a memory access [45].

Cycle Time Equations

We assume that SRAM supports a pipeline mechanism with placement of latches at appropriate locations. More specifically we consider latches between buffers in H-tree network and after row & column selection. Cycle time is then determined by the slowest path between any two latches and is given by:

$$\begin{aligned} T_{cycle-time} &= max(T_{row-dec-path} + T_{wordline-reset} + T_{bl-restore}, \\ &\quad T_{between-buffers-htree-network}, T_{col-dec-path}) \end{aligned} \tag{2.23}$$

2.5 Interconnect Modeling

In deep sub-micron era, the choice of wire model affects substantially the performance and the power consumption of large caches [30, 31]. The properties of wires depend on a number of factors such as dimensions, signaling, operating voltage/frequency etc. Based on signaling we can distinguish two broad categories of RC wires: 1. Traditional full-swing wires, 2. Differential, low-swing, low power wires. The former are employed when optimal delay or delay/power trade-offs are desirable by inserting properly-sized repeaters at regular intervals across the wire, while the latter are suitable for optimal power efficiency.

Wire Parasitics

The resistance and capacitance per unit length of a wire is given by the following equations [16]

$$R_{wire} = \frac{\rho}{d * (thickness - barrier)(width - 2barrier)} \tag{2.24}$$

where, $d(< 1)$ is the loss in cross-sectional area due to dishing effect [3] and ρ is the resistivity of the metal.

$$\begin{aligned} C_{wire} = \varepsilon_0(2K\varepsilon_{horiz}\frac{thickness}{spacing} + 2\varepsilon_{vert}\frac{width}{layerspacing}) + \\ fringe(\varepsilon_{horiz}, \varepsilon_{vert}) \end{aligned} \tag{2.25}$$

where the first term corresponds to the sidewall capacitance, the second term models the capacitance due to wires in adjacent layers, and the last term corresponds to the fringing capacitance between the sidewall and the substrate.

Global Wires

For a long repeated wire, the single pole time constant model for the interconnect fragment shown in figure 2.13 is given by,

$$\tau = \left(\frac{1}{l}R_o(C_0 + C_p) + \frac{R_o}{s}C_{wire} + R_{wire}sC_0 + 0.5R_{wire}C_{wire}l\right) \tag{2.26}$$

In the above equation, C_0 is the capacitance of the minimum sized repeater, C_p is its output parasitic capacitance, R_o is its output resistance, l is the length of the interconnect segment between repeaters and s is the size of the repeater normalized to the minimum value. The values of C_0, C_p, and R_o are constant for a given process technology. Wire parasitics R_{wire} and C_{wire} represent resistance and capacitance per unit length. The optimal repeater sizing and spacing values can be calculated by differentiating equation 2.26 with respect to s and l and setting it equal to zero.

$$L_{optimal} = \sqrt{\frac{2R_o(C_0 + C_p)}{R_{wire}C_{wire}}} \tag{2.27}$$

$$S_{optimal} = \sqrt{\frac{R_oC_{wire}}{R_{wire}C_0}} \tag{2.28}$$

and finally the delay is given:

$$delay = 0.693 * \tau * len \tag{2.29}$$

where len is the length of the wire. The delay value calculated using the above $L_{optimal}$ and $S_{optimal}$ is guaranteed to have minimum value.

The leakage power of global wires is given by:

$$P_{leakage} = \frac{1+\beta}{2}V_{DD}I_{leak} \tag{2.30}$$

where β is the PMOS to NMOS size ratio. Also I_{leak} is the leakage current of a buffer and is computed using equation 2.1 evaluated for $W = W_{min.nmos} * S_{optimal}$.

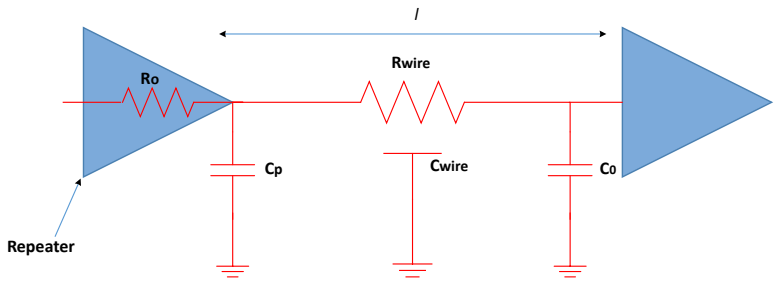


Figure 2.13: Interconnect segment

Low-swing Wires

A low swing interconnect system consists of three main components: (1) Transmitter that generates and drives the low-swing signal, (2) Twisted differential wires, and (3) Receiver amplifier. Figure 2.14 shows a synchronous low swing interconnect system that distributes clock signal to both transmitter and receiver [16]. During the first half of the cycle, the driver is OFF and the differential wires are equalized to the same voltage. During the second half of the cycle, the drivers turns ON. At the end of the cycle, the receiver senses the differential voltage and amplifies it to full-swing levels.

Transmitter

Figure 2.14(iii) shows the transmitter for one of the wires. A second transmitter using the complementary input drives the complementary wire. The total delay of the transmitter is given by:

$$t_{delay} = nand_{delay} + inverter_{delay} + driver_{delay} \tag{2.31}$$

Each gate in the above equation (*nand*, *inverter*, and *driver*) can be reduced to a simple RC tree as described in 2.4. After that Horowitz approximation (eqs. 2.13- 2.16) is applied to calculate the delay of each gate.

The leakage power of the transmitter is computed using the equation below:

$$P_{leakage} = 4 * (Pleak - inv + Pleak - nand2) \tag{2.32}$$

where $P_{leak-inv}$ and $P_{leak-nand2}$ are the leakage power of an inverter and a nand2 gate respectively which are discussed in section 2.3.

Twisted differential wires

Figure 2.14(ii) shows the differential wires with twisting and equalizing. The following equations present the time constant and capacitance values of the segment that consist of low-swing drivers and wires.

$$t_{driver} = (R_{driver} * (C_{wire} + C_{drain}) + R_{wire}C_{wire}/2 + (R_{driver} + R_{wire}) * C_{senseamp}) \tag{2.33}$$

The C_{wire} and R_{wire} in the above equation represents resistance and capacitance parasitics of the low-swing wire. R_{driver} and C_{drain} are resistance and drain capacitance of the driver transistors.

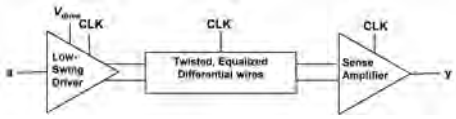
Leakage power is estimated for the driver logic (that drives the differential wires) and is given utilizing the equation below:

$$P_{leak} = 4 * P_{leak-inv} \tag{2.34}$$

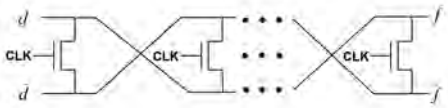
where again $P_{leak-inv}$ is the leakage power of an inverter.

Receiver amplifier

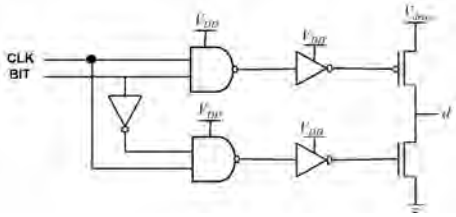
Figure 2.14(iv) shows the cross coupled inverter sense amplifier circuit used at the receiver. The delay and power values of the sense amplifier are directly calculated from the SPICE simulation [46].



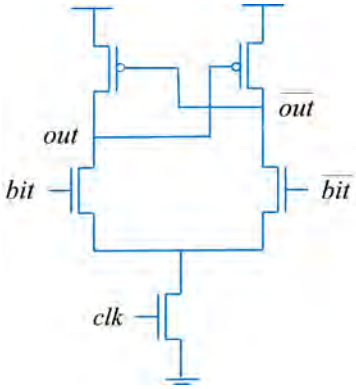
((i)) Overall System Architecture



((ii)) Differential Wire Circuit



((iii)) Transmitter Circuit



((iv)) Sense Amplifier Circuit

Figure 2.14: Synchronous Low Swing Interconnect System

Chapter 3

Cache Modeling under Parametric Variations

3.1 Parametric Variations and Impact

In deep sub-micron technologies parameter variations affect both timing and power. As a result variability must be considered to design chips according to specifications [43].

Delays are not constant: Variations in process, voltage, temperature, and input values (PVTI) all contribute to the worst-case critical-path delay. Designers add an extra worst-case delay component ,one for each variation parameter, in the delay of critical path to ensure reliable circuit operation, a.k.a. guardbanding. As technology scales down, especially beyond 90 nm, variability increases and so safety delay margins do, to guarantee correct circuit operation[4].

The ever growing variability in modern technologies exacerbates the manufacturing of low performance chips as more parts have to be discarded. As a consequence semiconductor industries have higher costs and less revenue. Moreover, parameter variations are the main reason for leakage current fluctuations, which can vary by as much as a factor of 20 across dies [4]. In the future, the tremendous increase in leakage currents and the resulting high temperatures could lead to re-consider the standard burn-in test [5].

Thus, by affecting yield, increasing variability is becoming a reliability concern. Alternative strategies are required to prevent extreme device and circuit variability from stalling the benefits of technology scaling. At Intel, within-die parameter variability causes gate delay variations to be very close to the minimum (hold) and maximum (setup) delay margins at the

130 nm technology node [43].

The variability is expected to have substantial contribution in the design of future processors. The thrust for low-power designs that can be operated for extended times is leading to near threshold computing in which VDD is lowered close to threshold voltage (VTH). This may lead to quadratic power savings but unfortunately it also makes circuits more prone to variations increasing the mean but also the variance of the Vth distribution.

To this end variability must be taken into account so in circuit as in microarchitectural level to exploit performance scaling.

Introduction to Variations

Figure 3.1 presents a general classification of the parameter variations. Most process variations stem from equipment processing while environmental variations change with a part's use and workload.

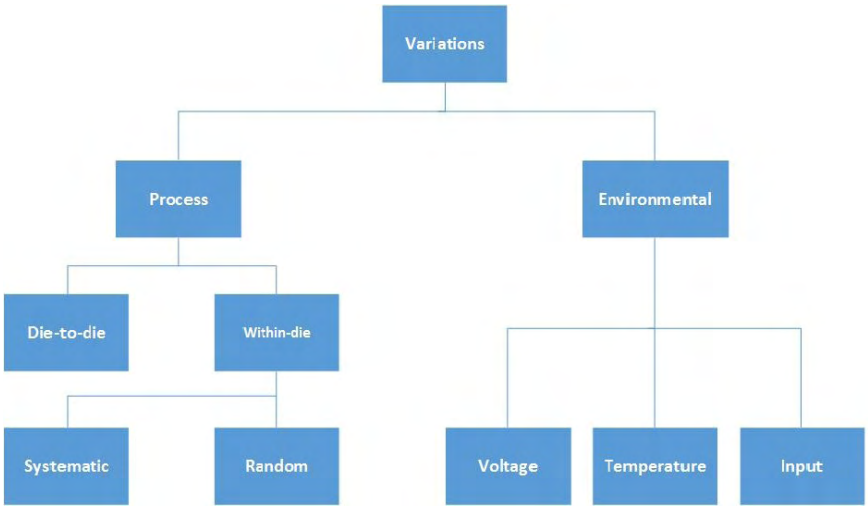


Figure 3.1: Parameter variations

Below we focus on Process Variations but the same analysis can be applied to Environmental Variations too.

Process Variation

Die-to-die fluctuations (from lot to lot and wafer to wafer) result from factors such as processing temperature and equipment properties [6]. Conversely, Within-Die (WID) variations result from factors such as non deterministic placement of dopant atoms and channel length variation

across a single die. WID component further subdividing into random and systematic components. By definition, systematic variations exhibit spatial correlation and, therefore, nearby transistors share similar systematic parameter values [14]. In contrast, random variation has no spatial correlation and, therefore, a transistor's randomly varying parameters differ from those of its immediate neighbors. Most generally, variation in any parameter P can be represented as follows:

$$\delta P = \delta P_{D2D} + \delta P_{WID} = \delta P_{D2D} + \delta P_{rand} + \delta P_{sys} \tag{3.1}$$

In this work, we focus only on random component of WID variations since random mismatch is becoming the most dominant source of WID variations in the future technologies. Process variations are primarily captured in threshold voltage V_{th} & gate length L. So we do not discuss anymore about deterministic V_{th} or L in a given technology but for random V_{th} & L sampled from a distribution. Now that V_{th} and L_{eff} are random, leakage power and delay follow a distribution as well.

We assume that the random components of V_{th} and L_{eff}, δV_{thrand} and $\delta L_{eff_{rand}}$ respectively, are both normally distributed with zero mean. For ease of analysis, we assume that δV_{thrand} and $\delta L_{eff_{rand}}$ for a given transistor are uncorrelated. The standard deviation of δV_{thrand} complies with Pelgrom rule [34] while the standard deviation of $\delta L_{eff_{rand}}$ is set to the half of $\sigma\delta V_{thrand}$ as in [38].

3.2 Statistical delay estimation of CMOS logic gates

Figure 3.2 depicts the procedure we followed to estimate the delay of a CMOS logic gate in the presence of process variations.

Firstly we derive the equivalent RC circuit model for each logic gate of the circuit as described in the previous chapter. Then we pick a random value for δV_{th} and δL_{eff} sampling from the underlying distribution of each one and evaluate the current using eq. 3.2 for $L_{eff} = L_{eff_{nominal}} + \delta L_{eff}$ and $V_{th} = V_{thnominal} + \delta V_{th}$. After that I_{eff} is derived using the equation 2.12. Then R_{eff} which is a function of I_{eff} and $C_{g/diff}$ can be computed using the equations 2.10, 2.9 and 2.11 respectively. Now that R_{eff} and $C_{g/dif}$ are known we can find RC time constant τ with respect to the extracted equivalent RC circuit. Finally the gate delay is obtained using the equations 2.13, 2.14, 2.15 and 2.16. We repeat the above steps N times to extract the delay statistics. Figure 3.3 shows an estimation

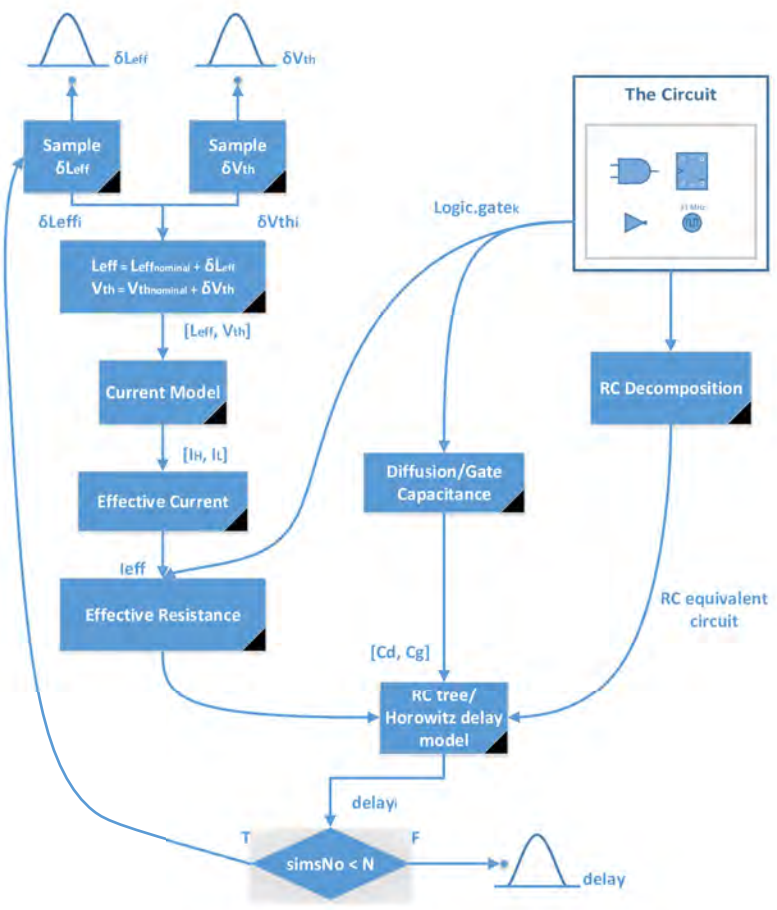


Figure 3.2: Overall flow of statistical delay estimation of CMOS logic gates

of the max cell access time distribution of a subarray with 256 rows and 2048 cols.

Alpha-power current law

We used the α-power current model with $\alpha = 1.3$, which reflects short-channel effects into I_{DS} better than the Shockley model. The parameter α was chosen 1.3 after experimentation with different α values between 1 and 2. ITRS [3] provides detailed reports about any parameter for different technology nodes. The current of our model for $\alpha = 1.3$ was in perfect agreement with ITRS reports about I_{on} current (see figure 3.4).

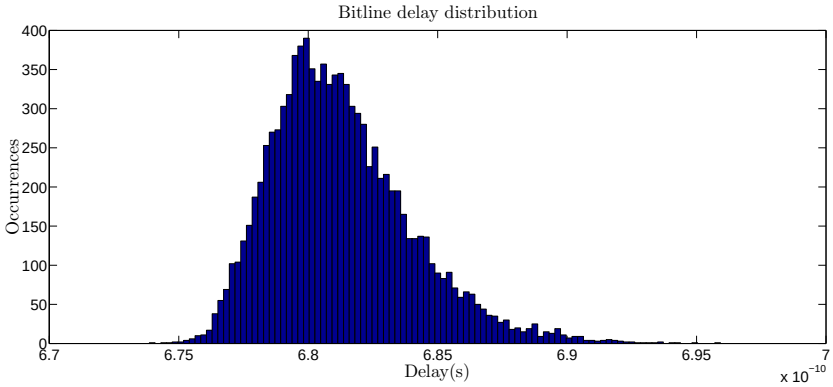


Figure 3.3: Maximum bitline access time distribution. The mean value is $\mu = 6.74E-10$s and the standard deviation is $\sigma = 2.74E-12$s.

$$I_{DS} = \begin{cases} 0, & if\ V_{gs} \leq V_{th} \\ \frac{W}{L_{eff}} \frac{P_c}{P_u} (V_{gs} - V_{th})^{\alpha/2} V_{ds}, & if\ V_{ds} < V_{d0} \\ \frac{W}{L_{eff}} P_c (V_{gs} - V_{th})^{\alpha}, & if\ V_{ds} \geq V_{d0} \end{cases} \tag{3.2}$$

In this equation P_c and P_u are constants and V_{d0} is given by:

$$V_{d0} \ = \ P_u (V_{gs} - V_{th})^{\alpha/2} \tag{3.3}$$

All three parameter P_c, P_v, V_{d0} were extracted from SPICE simulations using the PTM model [42] for 65, 45 and 32 nm technology nodes. Figure 3.6 shows the accuracy of α-power and Shockley current model with respect to HSpice for a 65 nm technology node.

Although Shockley current model is inaccurate in comparison with α-power model, it provides extensive runtime savings as it is much simpler. Figure 3.5 shows the maximum cell access time distribution when Shockley current model is employed. Both mean value and standard deviation decreases compared to the distribution derived when α-power model is incorporated (see figure 3.3).

3.3 Statistical Estimation of Leakage Power for CMOS

To extract the statistics of leakage power we repeat the following steps: We pick a value for δV_{th} and δL_{eff} sampling from their underlying

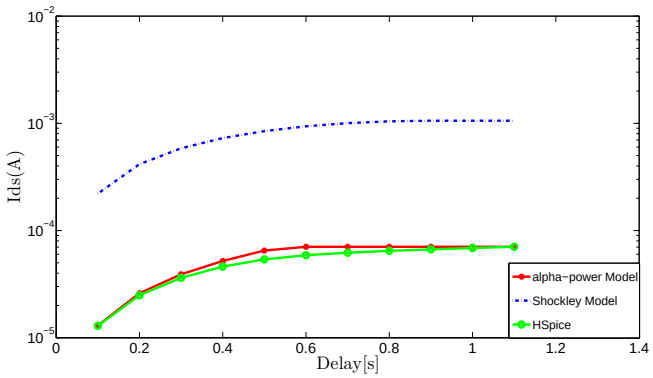


Figure 3.4: I_{ds}-V_{ds} characteristics calculated by HSPICE simulations, α-power law model and Shockley model at $V_{gs} = 1.1V$ for 65 nm technology.

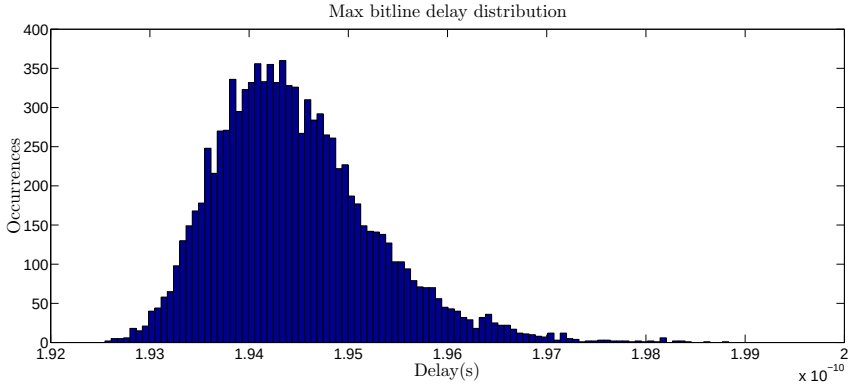


Figure 3.5: Max bitline access time distribution. The mean value is $\mu = 1.94E-10$s and the standard deviation is $\sigma = 7.98E-13$s.

distribution and then evaluate the leakage current formula eqn. 2.1 for $L_{eff} = L_{eff_{nominal}} + \delta L_{eff}$ and $V_{th} = V_{th_{nominal}} + \delta V_{th}$. After that we apply the equations 2.2 and 2.3 to find the logic gate's leakage power. We provide an example of the total leakage power distribution of a 4K subarray as shown in figure 3.6 following the procedure we have just described . HP Cacti reported a nominal leakage power of 5.5W which means that process variations shifted the nominal value about 2W. As caches getting larger the standard deviation is getting larger too.

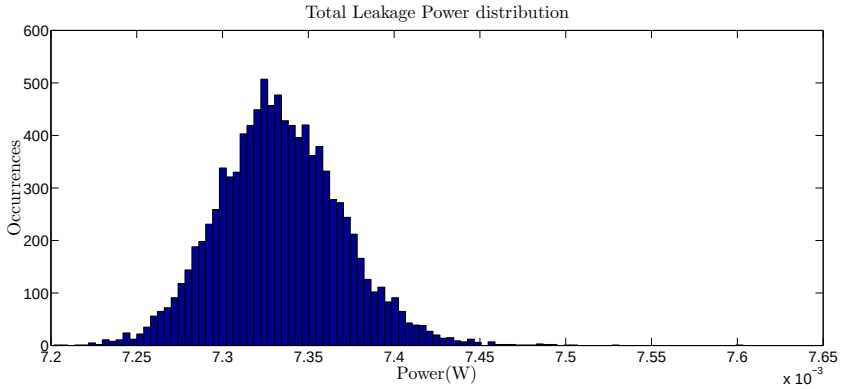


Figure 3.6: Total Leakage Power Distribution of a 4K subarray. The mean value is $\mu = 7.2E-3$W and the standard deviation is $sd = 0.036E-3$W.

3.4 SRAM Links under variations

When we have to choose between different cache architectures, to find the one that fulfills our requirements optimally, there are some whose network on SRAM chip dominates the overall cache performance. This happens when the cache is divided into multiple banks which requires an appropriate network to drive the data to the banks. A cache architecture like that improves the cache throughput as more parallel cache accesses are allowed, but occupies more area. As far as we know there is no prior work that studied the impact of the variability in network on SRAM chip to memory cache performance.

We consider variability in interconnect wires due to Chemical Mechanical Planarization (CMP), especially for designs of minimum link delay [15]. CMP causes surface imperfections in the interconnect wires because of dishing and erosion as shown in figure 3.7. The change in the top surface of the global interconnects due to dishing effects changes the resistance of the wires considerably [28] whereas the wire capacitance is not affected too much [1]. We consider that dishing effect doesn't affect all the wires in the same manner and as a result we treat it as a random variable. So it requires statistical manipulation as described in the introduction of the current chapter. We assume that the random component of dishing $\Delta dish_{random}$ follows a normal distribution with 0 mean and standard deviation $\sigma = 6\%$ [33]. To estimate the interconnect resistance and subsequently the wire delay we picked a value sampling from dishing effect distribution and evaluate the wire resistance as described in the previous chapter N times.

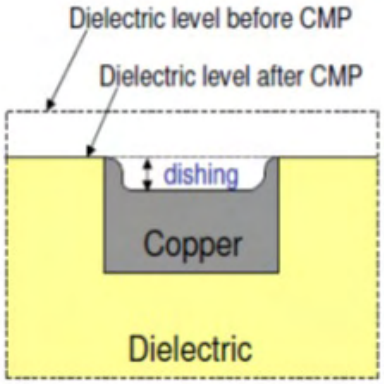


Figure 3.7: Surface imperfections in the interconnect wires because of dishing and erosion

3.5 SRAM Yield-Estimation

The reliability of a circuit is measured by the number of the circuits that satisfy all the constraints of the specification to the total number of manufactured circuits, such as 99% of manufactured circuits are faster than 1 ns and consume less than 1 mW. This fraction is referred to as yield and can be mathematically expressed as

$$Y = \int_{x_1}\int_{x2}\dots\int_{x_n} f_{\mathbf{X}}(\mathbf{x})C(h(\mathbf{x}))\,\mathrm{d}x_1\mathrm{d}x_2\dots\mathrm{d}x_n \tag{3.4}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$ [1] is a n-dimensional vector of all independent transistor parameters of a circuit, $f_{\mathbf{X}}(\mathbf{x}) = \prod f_{X_1}(x_1)f_{X_2}(x_2)\dots f_{X_n}(x_n)$ is the multivariate probability distribution of $\mathbf{x}$, $y = h(\mathbf{x})$ is any circuit metric and C is the constraint function which is defined as 1 for circuits that are considered to be working, and 0 otherwise. In the example, $C(h) = 1$ for $h_1(\mathbf{x}) < 10^{-9}s$ and $h_2(\mathbf{x}) < 10^{-9}W$.

Monte Carlo Methods

MC provides a flexible and elegant numerical solution for two very classic computational problems in applied science that are met nearly in any experiment. In order to obtain a picture of the distribution $f_o(h(\mathbf{x}))$ it is only needed to derive a random sample $\mathbf{x_i}$ drawn from the $f_{\mathbf{X(i)}}(\mathbf{x_i})$ and then compute $h(\mathbf{x_i})$ repeating for $i = 1, 2\dots, N$. Furthermore MC is the

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

only general way to numerically solve multidimensional integrals, such as the expression for Y above. The calculation reduces to:

$$Y \simeq \sum_{i=1}^{N} \frac{C(h(\mathbf{x_i}))}{N} \qquad \mathbf{x_i} \sim f_{\mathbf{X(i)}}(\mathbf{x_i}) \tag{3.5}$$

A more accurate approach to estimate the yield is to interpolate the points $h(\mathbf{x_i})$ to find a continuous approximation of the output CDF $F_{\mathbf{o}}(h(\mathbf{x_i}))$ and then read the yield value at the desired constraint y_c, $P[Y \leq y_c]$.

Importance Sampling

The main reason Monte Carlo cannot cope with high yield estimations, in reasonable number of simulations, is because it wastes a lot of time sampling around the mean rather than in the tails. To this end variance reduction techniques are exploited to reduce the error of the estimate, given a fixed number of sample points [39]. One of the most common variance reduction method is Importance Sampling (IS). IS [49, 21] is choosing a good distribution that extracts more sample points from the important regions rather than sampling blindly. Mathematically, the concept is based on:

$$\int \theta f_X(x)\mathrm{d}x = \int \theta f_X(x)\frac{g(x)}{g(x)}\mathrm{d}x = \int \frac{\theta f_X(x)}{g(x)} g(x)\mathrm{d}x = \mathbb{E}_g(\frac{\theta f_X(x)}{g(x)}) \tag{3.6}$$

where θ is the desired-parameter to estimate.

Exponent Monte Carlo

Exponent Monte Carlo (EMC) falls in IS techniques and was conceptually presented in [10]. The principle is to sample $\mathbf{x_1} \ldots \mathbf{x_N}$ from $g_{\mathbf{X}}(\mathbf{x}) := f_{\mathbf{X}}^{1-\gamma}(\mathbf{x}) = \prod f_{X_1}^{1-\gamma}(x_1) f_{X_2}^{1-\gamma}(x_2) \ldots f_{X_n}^{1-\gamma}(x_n)$ (hence the name Exponent MC) instead of $f_{\mathbf{X}}(\mathbf{x})$. For $\gamma > 0$, EMC samples less likely observations which define the tails of $f_o(h(\mathbf{x}))$ more precisely than the bulk. Figures 3.8 and 3.9 show that $g_X(x)$ samples observations between -5 and -1 (x $\geq$ -5 and x $\leq$ -1), falling in the tail of $f_X(x)$, with probability significantly larger than the natural distribution $f_X(x)$ does. It's obvious that $\gamma = 0$ reflects the naive Monte Carlo while $\gamma = 1$ represents a special case refereed as entry sampling in [49].

An estimation of Yield (equation 3.4) is given now:

$$Y \simeq \frac{\sum C(h(\mathbf{x}_i)) f_{\mathbf{X}}(\mathbf{x}_i)^{\gamma}}{\sum f_X(\mathbf{x_i})^{\gamma}} \qquad \mathbf{x_i} \sim f_{\mathbf{X}}^{1-\gamma}(\mathbf{x_i}) \tag{3.7}$$

Table 3.1 quantifies the effectiveness of EMC against the naive MC in yield, especially high, estimation. It shows the number of MC and EMC simulations needed to estimate a specified yield. To do that we estimate the yield, for different pass/fail f_0 critical values, of a 32K subarray in the presence of variability. We focus on max cell access delay performance metric and classify a subarray as faulty if the max cell access delay exceeds a specified critical value f_0. More specifically we are counting the number of simulation needed until a faulty observation is captured.

Yield	MC simulations	EMC simulations	Speedup(MC/EMC)
98.7%	77	1	x77
99.7%	422	1	x422
99.9%	6499	1	x6499
>99.999%	Too long!	27913	-

Table 3.1: Number of Monte Carlo simulations (2nd column) and EMC simulations (3rd column) needed to estimate yield. Speed up of EMC compared to regular MC (4th column)

The stricter the critical value is set the higher the yield to estimate. MC required more than 100k simulations to estimate 99.999% yield .

Numerous methods generating normally distributed random numbers have been proposed [7, 27]. We provide a very convenient exploitation of EMC through the Ziggurat method [27]. Below we show step by step how EMC distribution $g_X(x) = f_X^{(1-\gamma)}(x)$ could be transformed into an equivalent normal distribution assuming that the natural distribution $f_X(x)$ is normally distributed $N(\mu, \sigma)$ too.

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.8}$$

$$g(x) = \frac{1}{\sigma^{(1-\gamma)}(\sqrt{2\pi})^{(1-\gamma)}} e^{-\frac{(x-\mu)^2(1-\gamma)}{2\sigma^2}} \tag{3.9}$$

$$g(x) = \frac{1}{\frac{\sigma^{(1-\gamma)}}{(\sqrt{2\pi})^{\gamma}}\sqrt{2\pi}} e^{-\frac{(x-\mu)^2(1-\gamma)\frac{\sigma^{-2\gamma}}{(\sqrt{2\pi})^{2\gamma}}}{2\sigma^2\frac{\sigma^{-2\gamma}}{(\sqrt{2\pi})^{2\gamma}}}} \tag{3.10}$$

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

$$g(x) = \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{(x-\mu)^2\sqrt{(1-\gamma)}^2 \frac{\sigma^{-2\gamma}}{(\sqrt{2\pi})^{2\gamma}}}{2\sigma'^2}} \tag{3.11}$$

$$g(x) = \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{((x-\mu)\kappa)^2)}{2\sigma'^2}} \tag{3.12}$$

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

$$g(x) = \frac{1}{\sigma'\sqrt{2\pi}} e^{-\frac{(x'-\mu')^2)}{2\sigma'^2}} \tag{3.13}$$

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

So it's sufficient to first pick x' drawn from $N(\mu', \sigma')$ and then return $x = x'/\kappa$.

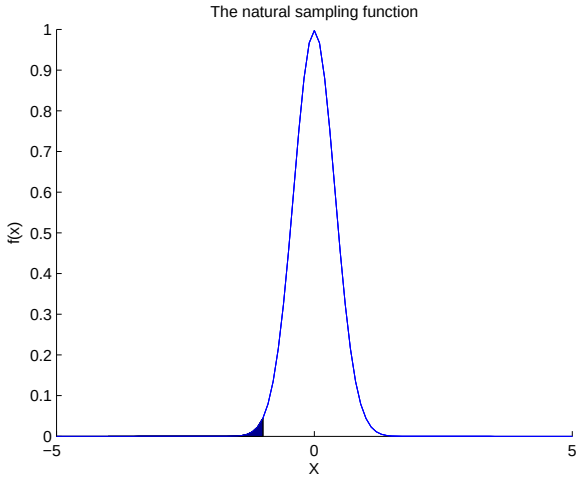


Figure 3.8: The natural sampling function. The shaded area represents the probability to sample an observation between x $\geq$ -5 and x $\leq$ -1.

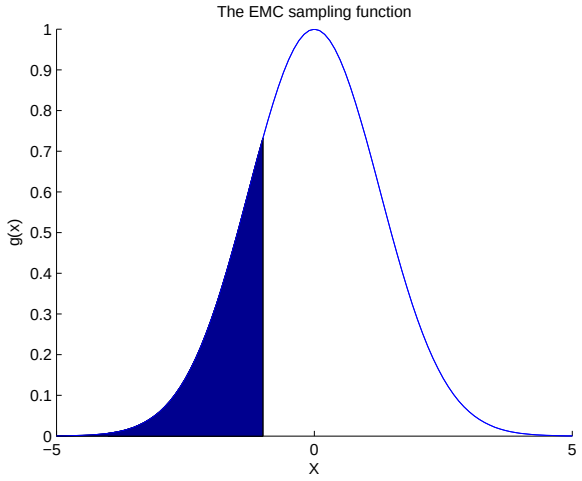


Figure 3.9: The EMC sampling function. The shaded area represents the probability to sample an observation between x $\geq$ -5 and x $\leq$ -1.

Chapter 4

Statistical SRAM Analysis

We incorporated the statistical methodology presented in [50] on top of HP Cacti simulator [45]. The aforementioned methodology begins with the extraction of the sensitivity to process variation of the critical path parameters in different memory islands and then combines these sensitivity information considering the connectivity of islands[1] to reconstruct the full memory statistics. Specifically it consists of the following consecutive steps:

- For any critical path parameter $\forall$ memory island $i = 1 \dots I$ derive sensitivity distribution $\Delta\mathbf{P_i}$. $\Delta\mathbf{P_i}$ is the variation of parameter P shifted around nominal.
- $\forall$ island $i = 1 \dots I$, pick an M_i-sized sample $\mathbf{Z_i}$ from the sensitivity distribution $\Delta\mathbf{P_i}$.
- $\forall$ path p_j from all M possible paths[2], a parameter's sensitivity realization along all I-involved islands, 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 parameter $P \simeq P_0 + \sum_{i=1}^{I} \mathbf{Z_P}(z_i)$ where P_0 is the nominal point. This rule provided good accuracy ($\leq 1\%$ RMS error) of all examples tested in [50].
- Select the worst of the M paths and assign that value to the memory observation.

[1] An island is formally defined as a set of transistors forming a memory component whose instance count (multiplicity) and connectivity differs to other islands.

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

- Repeat the above steps n times with different random sequences to generate memory statistics.

We can distinguish two major points that make the above methodology remarkable. First is the systematic evaluation of all possible paths and not only the critical path based on the observation for example that a worst case bitcell is not necessary on the same path with the worst case sense amplifier comprising a total worst case. Second considering variability in subsets of all transistors increases the effectiveness of any importance sampling technique [17] that in turn improves the runtime of the statistical analysis.

Although the statistical analysis followed in [50] achieves more accurate results than others, which simulate only the critical path, their approach doesn't provide a systematic methodology to evaluate the impact of parameters variation across different memory cache architectures.

4.1 Traditional Optimization

In order to find the cache architecture that best meets the design constraints we exploited the optimization function embedded in HP Cacti Simulator [45]. The optimization problem could be formulated mathematically as:

$$
\begin{aligned}
\underset{\mathbf{arch}}{\textbf{minimize}} \quad & cost(arch) = \frac{P_{dyn}(\mathbf{arch})}{min.dynamic.power} en.for.dyn.power.opt + \\
& \frac{T_{access}(\mathbf{arch})}{min.delay} en.for.delay.opt + \\
& \frac{P_{leak}(\mathbf{arch})}{min.leakage.power} en.for.leak.power.opt + \\
& \frac{T_{cycle.time}(\mathbf{arch})}{min.cycle.time} en.for.cycle.time.opt \\
\\
\textbf{subject to} \quad & P_{dyn}(\mathbf{arch}) \leq (1+a) \times min.dyn.power \\
& P_{leak}(\mathbf{arch}) \leq (1+b) \times min.leak.power \\
& T_{access}(\mathbf{arch}) \leq (1+c) \times min.delay \\
& T_{cycle.time}(\mathbf{arch}) \leq (1+d) \times min.cycle.time
\end{aligned}
\tag{4.1}
$$

where $\mathbf{arch} = \{N_{banks}, N_{subbanks}, N_{mats.in.subbank}, N_{subarr.rows}, N_{subarr.cols}, \ldots\}$ is the unknown vector. $P_{dyn}(\mathbf{arch})$, $P_{leak}(\mathbf{arch})$, $T_{access}(\mathbf{arch})$ and $T_{cycle.time}(\mathbf{arch})$

are the dynamic power, leakage power, access time and cycle time of a solution respectively and $min.dynamic.power$, $min.leakage.power$, $min.delay$ and $min.cycle.time$ are their minimum (best) values. $0 \leq a,\ b,\ c,\ d \leq 1$ are user-specified variables that reduce the set of solutions to those that deviate at most $a\%$, $b\%$, $c\%$, and $d\%$ from the minimum values of dynamic power, leakage power, delay and cycle time respectively. Finally $en.for.dyn.power.opt$, $en.for.delay.opt$, $en.for.leak.power.opt$ and $en.for.cycle.time.opt$ are user-specified boolean variables.

4.2 New Objective function

We modified the above optimization problem by taking into account the parameters variability. In the presence of parameters variability $P_{dyn}(\mathbf{arch})$, $P_{leak}(\mathbf{arch})$, $T_{access}(\mathbf{arch})$ and $T_{cycle.time}(\mathbf{arch})$ are treated as random variables. In MC-based statistical approach a random variable is represented by a vector of values which reflects random variable's distribution. In order to guarantee worst case operation of the circuit we have to pick the worst value from the corresponding vector of each performance metric. To do that we substituted $P_{dyn}(\mathbf{arch})$, $P_{leak}(\mathbf{arch})$, $T_{access}(\mathbf{arch})$ and $T_{cycle.time}(\mathbf{arch})$ with $max(\mathbf{P_{dyn}}(\mathbf{arch}))$, $max(\mathbf{P_{leak}}(\mathbf{arch}))$, $max(\mathbf{T_{access}}(\mathbf{arch}))$, $max(\mathbf{T_{cycle.time}}(\mathbf{arch}))$ in the optimization problem. Also we extended the optimization problem to seek for the cache architecture with the highest yield. The optimization problem in mathematical terms has been transformed into the one below:

$$
\begin{aligned}
\underset{\mathbf{arch}}{\textbf{minimize}} \quad & cost(\mathbf{arch}) = \frac{max(\mathbf{P_{dyn}}(\mathbf{arch}))}{min.dynamic.power} en.for.dyn.power.opt + \\
& \frac{max(\mathbf{T_{access}}(\mathbf{arch}))}{min.delay} en.for.delay.opt + \\
& \frac{max(\mathbf{P_{leak}}(\mathbf{arch}))}{min.leakage.power} en.for.leak.power.opt + \\
& \frac{max(\mathbf{T_{cycle.time}}(\mathbf{arch}))}{min.cycle.time} en.for.cycle.time.opt + \\
& \frac{max.Yield}{Yield(\mathbf{arch})} en.for.Yield.opt \\
\textbf{subject to} \quad & max(\mathbf{P_{dyn}}(\mathbf{arch})) \leq (1+a) \times min.dyn.power \\
& max(\mathbf{P_{leak}}(\mathbf{arch})) \leq (1+b) \times min.leak.power \\
& max(\mathbf{T_{access}}(\mathbf{arch})) \leq (1+c) \times min.delay \\
& max(\mathbf{T_{cycle.time}}(\mathbf{arch})) \leq (1+d) \times min.cycle.time
\end{aligned}
\tag{4.2}
$$

Chapter 5

EVALUATION

In this section we demonstrate different features of the proposed tool. First we describe the test cases we used and then we provide some examples at both block and system level of a cache to illustrate the capabilities of our tool.

5.1 Experimental Setup

We selected 3 architectures of a 64K cache, in 65nm technology node, with respect to the impact they have on total access time different cache components. More specifically we picked a cache configuration where bitline delay dominates the total access time, a cache configuration where decoder + wordline delay contributes the most to the total access time and a cache configuration where h-tree network delay affects the most the total access time. The experimental setup is shown below:

cfg#No	**#banks**	**#subbanks**	**#mats-in-subbank**	**#rows-in-subarray**	**#cols-in-subarray**
cfg#1	1	0	0	256	2048
cfg#2	1	0	1	2048	128
cfg#3	1	0	256	16	64

Table 5.1: Cache configurations tested

5.2 Block Level Evaluation

Firstly the proposed tool offers the capability to study the impact of parametric variations on critical path parameters, such as delay, at the

cfg#No	Bitline delay[ns]	Decoder+ wordline delay[ns]	H-tree network delay[ns]	Total access time[ns]
cfg#1	0.655372	0.99145	0.101269	1.7589
cfg#2	1.5296	0.702634	0.172301	2.4116
cfg#3	0.034369	0.205853	4.91608	5.1634

Table 5.2: Bitline, decoder & H-tree network nominal delays of cache configurations under test

cfg#No	Bitline delay/Total access time(%)	Decoder+ wordline delay/Total access time (%)	H-tree network delay/Total access time (%)
cfg#1	37%	56%	5.7%
cfg#2	63%	29%	7.1%
cfg#3	0.6%	3.9%	95.2%

Table 5.3: Bitline, decoder & H-tree network delay as a percentage of total access time of cache configurations under test

block level. As an example we present the Probability Density Function (PDF)s of bitline, decoder & h-tree network delay of the cfg#1, cfg#2 and cfg#3 by applying the methodology shown in figure 3.5.

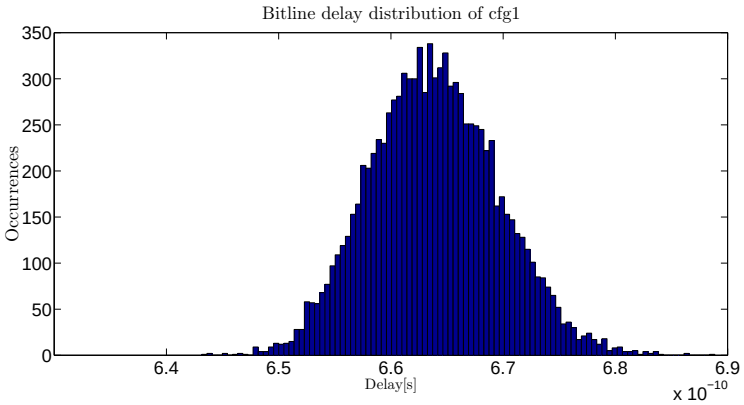


Figure 5.1: Bitline delay distribution of cfg#1. The mean value is $\mu = 6.6387E{-}9$s and the standard deviation is $\sigma = 5.7098E - 12$s.

As we can see the impact of standard deviation in cfg#1 is negligible while in cfg#2 is stronger. The more the rows in a subarray the greater

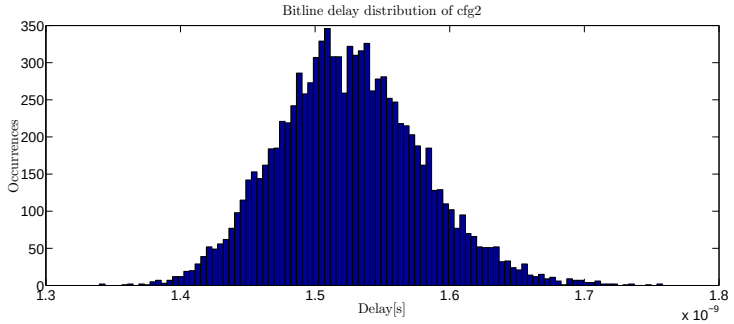


Figure 5.2: Bitline delay distribution of cfg#2. The mean value is $\mu = 1.5256E{-}9$s and the standard deviation is $\sigma = 5.3615E - 11$s.

the impact (SD/Mean) of standard deviation is becoming.

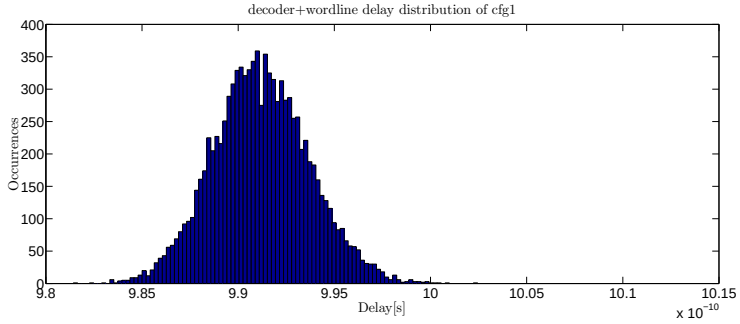


Figure 5.3: Decoder+Wordline delay distribution of cfg#1. The mean value is $\mu = 9.9119E - 10$s and the standard deviation is $\sigma = 2.5040E - 12$s.

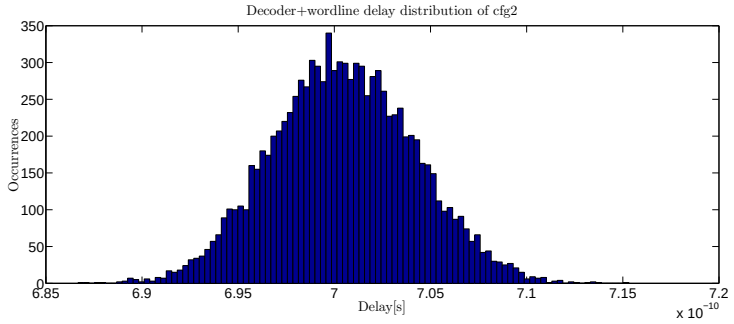


Figure 5.4: Decoder+Wordline delay distribution of cfg#2. The mean value is $\mu = 7.0047E - 10$s and the standard deviation is $\sigma = 3.6974E - 12$s.

From figures 5.3 and 5.4 we can understand that the standard deviation of decoder+wordline delay distribution has negligible impact in both cache configurations under test.

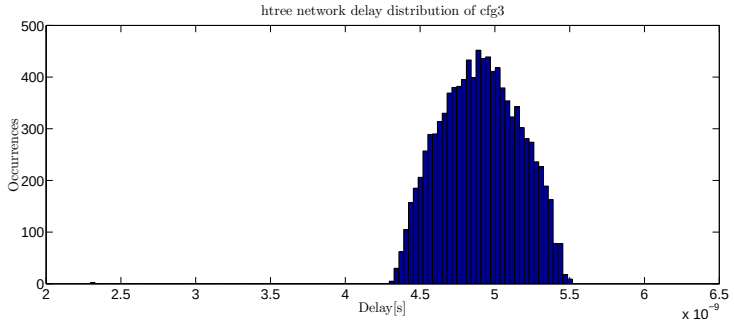


Figure 5.5: Htree delay distribution of cfg#3. The mean value is $\mu = 4.9032E-9$s and the standard deviation is $\sigma = 2.6216E-10$s.

The impact of variations in htree network delay shown in figure 5.5, about 5% of the mean value, indicates how important is the statistical analysis of that component of a cache system.

5.3 System Level Evaluation

In addition the proposed tool can be used to evaluate cache performance at the system level taking into consideration the connectivity of different blocks as well. In the figures below we present the PDF of the total access time for cache configurations under test following the methodology described in chapter 4.

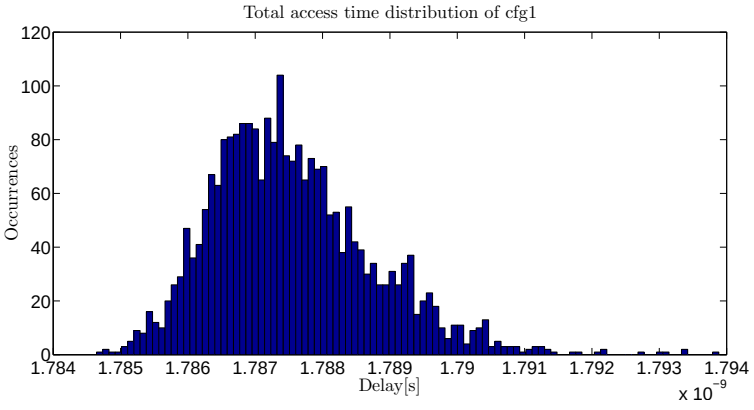


Figure 5.6: Total access time distribution of cfg#1. The mean value is $\mu = 1.7875E-9$s and the standard deviation is $\sigma = 1.1985E-12$s.

As we can observe Vt and L variability caused a variation of some picoseconds in total access time in all three cache configurations while the mean

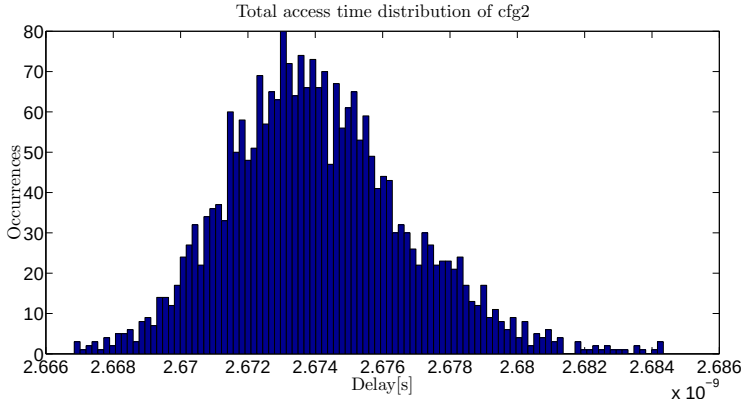


Figure 5.7: Total access time distribution of cfg#2. The mean value is $\mu = 2.6741E-9$s and the standard deviation is $\sigma = 2.6788E-12$s.

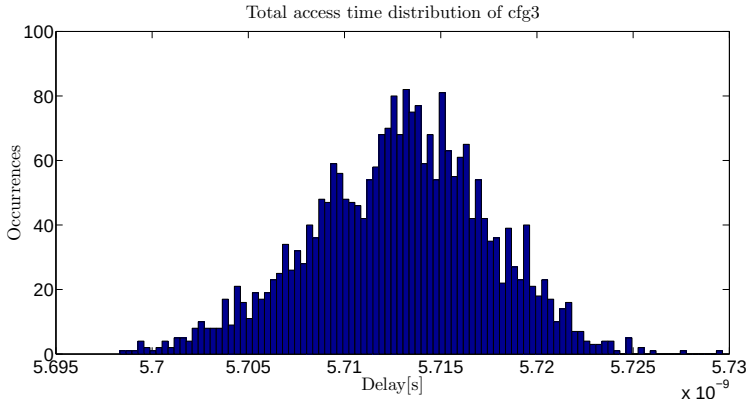


Figure 5.8: Total access time distribution of cfg#3. The mean value is $\mu = 5.71E-9$s and the standard deviation is $\sigma = 4.6116E-12$s.

value moved slightly from the nominal value. Also it is apparent that the distribution of cfg#3 is a skewed normal distribution which is a fact that strengthens our approach to study variability statistically instead of analytically making the assumption that all distributions are normal, as previous works did.

Except from that the proposed tool can estimate the maximum values of different performance metrics exploiting the methodology presented in [12, 13] which is based on EVT. Table 5.4 shows the estimated maximum of total access time for the cache configurations we tested.

Architectural cache exploration for Yield enhancement

Also the proposed tool can be configured to find the cache architecture that exhibits the highest yield. That can be happen by setting as the objective function (see equation 4.2) the yield maximization. Table 5.4 contains the yield estimates of cache configurations under test. In order to estimate the yield we set as reliability constraint to the total access time the 99.9% of the maximum access time. So every observation above this threshold is considered faulty. As we can observe cfg#3 is the most robust among the cache architectures we tested.

cfg#No	**Max Access Time[ns]**	**R-Crit. Acc.Time[ns]**	**Yield(%)**
cfg#1	1.81865	1.81683	99.999999992%
cfg#2	2.82370	2.82087	99.999999998%
cfg#3	5.62616	6.62053	99.999999999%

Table 5.4: Yield estimation of cache configurations under test

Chapter 6

Conclusion

In this master thesis project, we presented the steps we followed to develop a tool that primarily attains the effects of parametric variations on cache memories as well as determines the design solutions that are able to secure sturdy operating conditions while achieving minimum overhead. Namely, we modeled parametric variations within a conventional cache simulation framework and evaluated cache performance by adjusting various knobs both at the circuit and the architecture level. Allowing the tool to identify the optimal configurations under the effect of various constraints, we empowered the designers to come up with the right choices earlier in the design cycle and as a result to improve yield as well as efficiency.

Throughout the evaluation of our results in Chapter 5, we showed that a high yield estimation is feasible within reasonable runtime by exploiting enhanced statistical methodologies such as IS. Another precious feature in the hands of the designer, is that given a fixed size cache memory is capable to determine the cache architecture that exhibits the highest yield.

6.1 Future work

Parameters variations have been the center of attention in most contemporary works, with our approach being no exception. However, it would be challenging to see works that focus in systematic parameters variation as well. However, it would be challenging to see works that also model systematic parameters variation. As far as our approach is concerned, the key complication, which obstructed us from adapting it as a feature in our tool was the requirement of layout information in order to achieve accurate results. Another interesting aspect would be an attempt to estimate

yield by utilizing EVT techniques like the one proposed in [8]. Finally, a future prospect that concerns exclusively our proposed work would be the modification of our tool in order to comply with the 8-T and 10-T cell models (besides the 6-T model that we currently utilize), as well as with the FinFET transistor model.

References

[1] Lei He A, Andrew B. Kahng B, King Ho Tam A, and Jinjun Xiong A. Design of integrated-circuit interconnects with accurate modeling of chemical-mechanical planarization.

[2] Behrouz Afzal, Behzad Ebrahimi, Ali Afzali-Kusha, and Massoud Pedram. An accurate analytical i–v model for sub-90-nm mosfets and its application to read static noise margin modeling. Journal of Zhejiang University SCIENCE C, 13(1):58–70, 2012.

[3] S.I. Association. International technology roadmap for semiconductors 2005.

[4] Shekhar Borkar, Tanay Karnik, Siva Narendra, Jim Tschanz, Ali Keshavarzi, and Vivek De. Parameter variations and impact on circuits and microarchitecture. In Proceedings of the 40th Annual Design Automation Conference, DAC '03, pages 338–342, New York, NY, USA, 2003. ACM.

[5] Shekhar Y. Borkar. Microarchitecture and design challenges for gigascale integration. In MICRO, page 3. IEEE Computer Society, 2004.

[6] 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. Solid-State Circuits, IEEE Journal of, 37(2):183–190, Feb 2002.

[7] G. E. P. Box and M. E. Muller. A note on the generation of random normal deviates. Annals of Mathematical Statistics, 29:610–611, 1958.

[8] Alessandro Cevrero, Nestor Evmorfopoulos, Charalampos Antoniadis, Paolo Ienne, Yusuf Leblebici, Andreas Burg, and George Stamoulis. Fast and accurate ber estimation methodology for i/o links based on extreme value theory. In Design, Automation

Test in Europe Conference Exhibition (DATE), 2013, pages 503–508, March 2013.

[9] Gregory K. Chen, David Blaauw, Trevor Mudge, Dennis Sylvester, and Nam Sung Kim. Yield-driven near-threshold sram design. In Proceedings of the 2007 IEEE/ACM International Conference on Computer-aided Design, ICCAD '07, pages 660–666, Piscataway, NJ, USA, 2007. IEEE Press.

[10] B. Dierickx, M. Miranda, P. Dobrovolny, F. Kutscherauer, A. Papanikolaou, and P. Marchal. Propagating variability from technology to system level. In Physics of Semiconductor Devices, 2007. IWPSD 2007. International Workshop on, pages 74–79, Dec 2007.

[11] Anh-Tuan Do, Zhi-Hui Kong, Kiat-Seng Yeo, and Jeremy Yung Shern Low. Design and sensitivity analysis of a new current-mode sense amplifier for low-power sram. IEEE Trans. Very Large Scale Integr. Syst., 19(2):196–204, February 2011.

[12] Nestoras E. Evmorfopoulos, Dimitris P. Karampatzakis, and Georgios I. Stamoulis. Precise identification of the worst-case voltage drop conditions in power grid verification. In Soha Hassoun, editor, ICCAD, pages 112–118. ACM, 2006.

[13] Nestoras E. Evmorfopoulos, Maria-Aikaterini Rammou, George I. Stamoulis, and John Moondanos. Characterization of the worst-case current waveform excitations in general rlc-model power grid analysis. In ICCAD, pages 824–830. IEEE, 2010.

[14] Paul Friedberg, Yu Cao, Jason Cain, Ruth Wang, Jan Rabaey, and Costas Spanos. Modeling within-die spatial correlation effects for process-design co-optimization. In Proceedings of the 6th International Symposium on Quality of Electronic Design, ISQED '05, pages 516–521, Washington, DC, USA, 2005. IEEE Computer Society.

[15] C. Hernandez, F. Silla, and J. Duato. A methodology for the characterization of process variation in noc links. In Design, Automation Test in Europe Conference Exhibition (DATE), 2010, pages 685–690, March 2010.

[16] R. Ho. On-chip Wires: Scaling and Efficiency. PhD thesis, Stanford University, 2003.

[17] D.E. Hocevar, M.R. Lightner, and Timothy N. Trick. A study of variance reduction techniques for estimating circuit yields. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2(3):180–192, July 1983.

[18] Mark A. Horowitz. Timing models for mos circuits. Technical report, Stanford, CA, USA, 1983.

[19] M.C. Johnson, D. Somasekhar, and K. Roy. Models and algorithms for bounds on leakage in cmos circuits. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 18(6):714–725, Jun 1999.

[20] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi. Orion 2.0: A fast and accurate noc power and area model for early-stage design space exploration. In Proceedings of the Conference on Design, Automation and Test in Europe, DATE '09, pages 423–428, 3001 Leuven, Belgium, Belgium, 2009. European Design and Automation Association.

[21] Rouwaida Kanj, Rajiv V. Joshi, and Sani R. Nassif. Mixture importance sampling and its application to the analysis of sram designs in the presence of rare failure events. In Ellen Sentovich, editor, DAC, pages 69–72. ACM, 2006.

[22] Jaydeep P. Kulkarni, Keejong Kim, Sang Phill Park, and Kaushik Roy. Process variation tolerant sram array for ultra low voltage applications. In Proceedings of the 45th Annual Design Automation Conference, DAC '08, pages 108–113, New York, NY, USA, 2008. ACM.

[23] Chun-Yi Lee and N.K. Jha. Cacti-finfet: An integrated delay and power modeling framework for finfet-based caches under process variations. In Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE, pages 866–871, June 2011.

[24] Mahesh Mamidipaka and Nikil Dutt. ecacti: An enhanced power estimation model for on-chip caches. Technical report, In Technical Report TR-04-28, CECS, UCI, 2004.

[25] Mahesh Mamidipaka, Kamal Khouri, Nikil Dutt, and Magdy Abadir. Analytical models for leakage power estimation of memory array structures. In Proceedings of the 2Nd IEEE/ACM/IFIP International

Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS '04, pages 146–151, New York, NY, USA, 2004. ACM.

[26] Mahesh Mamidipaka, Kamal S. Khouri, Nikil D. Dutt, and Magdy S. Abadir. Analytical models for leakage power estimation of memory array structures. In Alex Orailoglu, Pai H. Chou, Petru Eles, and Axel Jantsch, editors, CODES+ISSS, pages 146–151. ACM, 2004.

[27] George Marsaglia and Wai Wan Tsang. The ziggurat method for generating random variables. Journal of Statistical Software, 5(8):1–7, 10 2000.

[28] Mosin Mondal, Tamer Ragheb, Xiang Wu, Adnan Aziz, and Yehia Massoud. Provisioning on-chip networks under buffered rc interconnect delay variations. In ISQED, pages 873–878. IEEE Computer Society, 2007.

[29] Saibal Mukhopadhyay, Hamid Mahmoodi-Meimand, and Kaushik Roy. Modeling of failure probability and statistical design of sram array for yield enhancement in nanoscaled cmos. IEEE Trans. on CAD of Integrated Circuits and Systems, 24(12):1859–1880, 2005.

[30] Naveen Muralimanohar and Rajeev Balasubramonian. Interconnect design considerations for large nuca caches. In Dean M. Tullsen and Brad Calder, editors, ISCA, pages 369–380. ACM, 2007.

[31] Naveen Muralimanohar, Rajeev Balasubramonian, and Norm Jouppi. Optimizing nuca organizations and wiring alternatives for large caches with cacti 6.0. In Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 40, pages 3–14, Washington, DC, USA, 2007. IEEE Computer Society.

[32] M.-H. Na, E.J. Nowak, W. Haensch, and J. Cai. The effective drive current in cmos inverters. In Electron Devices Meeting, 2002. IEDM '02. International, pages 121–124, Dec 2002.

[33] Sani R. Nassif. Modeling and analysis of manufacturing variations. 2001.

[34] M.J.M. Pelgrom, Aad C J Duinmaijer, and A.P.G. Welbers. Matching properties of mos transistors. Solid-State Circuits, IEEE Journal of, 24(5):1433–1439, Oct 1989.

[35] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits. Proceedings of the IEEE, 91(2):305–327, Feb 2003.

[36] J. Rubinstein, Paul Penfield, and M.A. Horowitz. Signal delay in rc tree networks. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2(3):202–211, July 1983.

[37] Jeren Samandari-Rad, Matthew R. Guthaus, and Richard Hughey. 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 ISQED, pages 506–515, 2012.

[38] Smruti R. Sarangi, Brian Greskamp, Radu Teodorescu, Jun Nakano, Abhishek Tiwari, and Josep Torrellas. Varius: A model of process variation and resulting timing errors for microarchitects. In in IEEE Transactions on Semiconductor Manufacturing, 2008.

[39] A. Singhee and R.A. Rutenbar. Extreme Statistics in Nanoscale Memory Design. Integrated Circuits and Systems. Springer, 2010.

[40] Amith Singhee and Rob A. Rutenbar. Statistical blockade: a novel method for very fast monte carlo simulation of rare circuit events, and its application. In Rudy Lauwereins and Jan Madsen, editors, DATE, pages 1379–1384. ACM, 2007.

[41] S.Narendra and A. Chandrakasan. Leakage in Nanometer CMOS Technologies. New York:Springer, 2006.

[42] Arizona State University. Predictive technology model.

[43] Osman S. Unsal, James W. Tschanz, Keith Bowman, Vivek De, Xavier Vera, Antonio Gonzlez, and Oguz Ergin. Impact of parameter variations on circuits and microarchitecture. IEEE Micro, 26(6):30–39, 2006.

[44] Neil Weste and David Harris. CMOS VLSI Design: A Circuits and Systems Perspective. Addison-Wesley Publishing Company, USA, 4th edition, 2010.

[45] Steven J. E. Wilton and Norman P. Jouppi. Cacti: An enhanced cache access and cycle time model. IEEE Journal of Solid-State Circuits, 31:677–688, 1996.

[46] Steven J. E. Wilton and Norman P. Jouppi. Cacti: An enhanced cache access and cycle time model. IEEE Journal of Solid-State Circuits, 31:677–688, 1996.

[47] Ping-Chin Yeh, Deepak K. Nayak, and D. Gitlin. Improved cv/i methodology to accurately predict cmos technology performance. Electron Devices, IEEE Transactions on, 54(7):1760–1762, July 2007.

[48] Yan Zhang, Dharmesh Parikh, Karthik Sankaranarayanan, Kevin Skadron, and Mircea Stan. Hotleakage: A temperature-aware model of subthreshold and gate leakage for architects. Technical report, 2003.

[49] Paul Zuber, Vladimir Matvejev, Philippe Roussel, Petr Dobrovolný, and Miguel Miranda. Exponent monte carlo for quick statistical circuit simulation. In Proceedings of the 19th International Conference on Integrated Circuit and System Design: Power and Timing Modeling, Optimization and Simulation, PATMOS'09, pages 36–45, Berlin, Heidelberg, 2010. Springer-Verlag.

[50] Paul Zuber, Miguel Miranda, Petr Dobrovolný, Koen van der Zanden, and Jong-Hoon Jung. Statistical sram analysis for yield enhancement. In DATE, pages 57–62. IEEE, 2010.