SRAM Leakage-Power Optimization Framework: a System Level Approach

Animesh Kumar

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2008-182
http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-182.html

December 19, 2008

Acknowledgement

Financial support from Gigascale Silicon Research Center, and National Science Foundation Infrastructure Grant (No: 0403427)

SRAM Leakage-Power Optimization Framework: a System Level Approach

by

Animesh Kumar

B.Tech. (Indian Institute of Technology, Kanpur, India) 2001
M.S. (University of California, Berkeley, CA) 2003

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Sciences

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Fall 2008

SRAM Leakage-Power Optimization Framework: a System Level Approach

Abstract

SRAM Leakage-Power Optimization Framework: a System Level Approach

by

Animesh Kumar

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kannan Ramchandran, Chair

SRAM leakage-power is a significant fraction of the total power consumption on a chip. Various system level techniques have been proposed to reduce this leakage-power by reducing (scaling) the supply voltage. SRAM supply voltage scaling reduces the leakage-power, but it increases stored-data failure rate due to commonly known failure mechanisms, for example, soft-errors.

This work studies SRAM leakage-power reduction using system level design techniques, with a data-reliability constraint. A statistical or probabilistic setup is used to model failure mechanisms like soft-errors or process-variations, and error-probability is used as a metric for reliability. Error models which combine various SRAM cell failure mechanisms are developed. In a probabilistic setup, the bit-error probability increases due to supply voltage reduction, but it can be compensated by suitable choices of error-correction code and data-refresh (scrubbing) rate. The trade-offs between leakage-power, supply voltage reduction, data-refresh rate, error-correction code, and decoding error probability are studied. The leakage-power – including redundancy overhead, coding power, and data-refresh power – is set as the cost-function and an error-probability target is set as the constraint. The cost-function is minimized subject to the constraint, over the choices of data-refresh rate, error-correction code, and supply voltage. Using this optimization procedure, simulation results and circuit-level leakage-power reduction estimates are presented.

Experimental results are presented for the special case of low duty-cycle applications like sensor nodes. Retention of stored data at lowest possible leakage-power is the only target in this case. Each SRAM cell has a threshold parameter called the data-retention voltage (DRV), above which the stored bit can be retained reliably. The DRV exhibits systematic and random variation due to process technology. Using the proposed optimization method, the retention supply voltage is selected to minimize the leakage-power per useful bit. The fundamental lower bound on the leakage-power per bit, while taking the DRV distribution into account, is established. For experimentally observed DRV-distributions from custom built SRAM chips, a $[31, 26, 3]$ Hamming code based retention scheme achieves a significant portion of the leakage-power reduction compared to the fundamental limit. These results are verified by twenty-four experimental chips manufactured in an industrial 90nm CMOS process.

Professor Kannan Ramchandran
Dissertation Committee Chair

To my mother and father.

सरस्वतीं विद्यां शरदेन्दुसौम्यवदनां वीणावेदाक्षमालावरहस्तां वाणीं मतिबुद्धिप्राणकर्त्रीं
विषयपापतिमिरार्कप्रभां शारदां अर्पयामि ॥

Contents

List of Figures

Acknowledgments

Salutations to that Supreme Consciousness, Who inspires thought, wisdom, intelligence, and consciousness in all the living beings. Salutations to my parents, who selflessly endured many difficult financial and emotional troubles to push me towards my desired career objectives. Among countless other qualities, their love, care, nurturing, guidance, and mercy have been irreplaceable.

Thanks to my advisor, Prof. Kannan Ramchandran, who guided me to a successful Ph.D. Over the years, his patience, suggestions, insights, vision, guidance, benevolence, and understanding has been excellent. His patience and help during my qualifying examination is unforgettable. His encouragement and persistence has improved my presentation, understanding, and writing skills by a large extent.

Thanks to Prof. Jan Rabaey, who helped me on numerous occasions with his patience, insights, guidance, vision, and understanding. It has been a pleasure to work with him on probabilistic aspects of memories. I could lay hands on BWRC resources and cutting-edge technology access due to his kindness.

Thanks to Prof. Kannan Ramchandran and Prof. Jan Rabaey for funding and support.

Special thanks to Prof. Martin Vetterli, Prof. Aloke Dutta, and Prof. Rakesh Bansal for successful recommendation letters, because of which I got admitted to University of California, Berkeley.

I continuously adore the ‘how to write’ training imparted by Dr. Prakash Ishwar.

I deeply appreciate the help, kindness, support, and suggestions of Prof. Peter

Bickel and Prof. Martin Wainwright, who were present in my qualifying exam committee. I again thank Prof. Peter Bickel for being a member in my dissertation committee.

I enjoyed being a teaching assistant under Prof. Ron Fearing, Prof. Venkat Anantharam, and Prof. Kannan Ramchandran.

I would like to thank my Berkeley friends including Dr. Prakash Ishwar, Dr. Vinod Prabhakaran, Dr. Huifang Qin, Dr. Arindam Chakrabarti, Dr. Vinayak Prabhu, Dr. Rupak Majumdar, Dr. Amish Patel, Nidhi Tiwary, Dr. Kaushik Ravindran, David Chen, James Wu, Pulkit Grover, Xuening Sun, Victor Wen, Chintan Thakkar, Dr. Artin Der Minassians, Simone Gambini, Mark Johnson, Biswanath Poudel, Afshan Sheikh, Maryam Vareth, Chuohao Yeo, Dr. Abhijit Davare, Krish Eswaran, Dr. Allie Fletcher, Dr. Rohit Puri, Dr. Abhik Majumdar, Dr. Dan Schonberg, Ping-Chen Huang, Nadathur Satish, Dr. June Wang, Susmit Jha, Dr. Alex Dimakis, Dr. Anand Sarwate, Dr. Hiren Patel, Wei Wang, Hao Zhang, Louis Alarcon, Asako Toda, Dr. Nate Pletcher, Michael Mark, Dr. Krishnendu Chatterjee, Dr. Satrajit Chatterjee, Jike Chong, and Bryan Catanzaro.

This work has benefited a lot from suggestions made by researchers at Intel Corporation, consisting of Dr. Marly Roncken, Dr. T. M. Mak, Dr. M. Spica, Dr. Rajiv Mathur, and Dr. M. Zhang.

I deeply appreciate the financial support from the Gigascale Silicon Research Center (GSRC), and the National Science Foundation (NSF).

I would like to acknowledge the company of my undergraduate friends including Dr. Saurabh Tiwary, Dr. Tandra Ghose, Dr. Smita Agrawal, Ashwani Kumar, Dr. Sourav Chatterji, Dr. Supratim Ray, Pradeep Gatram, Pushkar Sinha, and Gaurav Srivastava.

I adore University of California at Berkeley and their educational system, which allow many excellent aspiring individuals from across the globe to do research and exchange ideas on a solid educational platform.

Administrative assistants including Ruth Gjerde, Mary Byrnes, Erin Reiche, Jennifer Stone, Brenda Farrell, Tom Boot, La Shana Porlaris, Loretta Lutcher, Mark Davis, Joe Bullock, Elisa Lewis, Jontae Gray, and Farah Pranawahadi were very helpful.

Blessed I am to have three siblings – Snigdha Bhasin, Piyush Singh, and Akash Kartikey. They have been very supportive and encouraging during my Ph.D endeavors.

Chapter 1

Introduction

Static random access memory or SRAM has been the main data storage block for many generations of microprocessors. As the name suggests, the SRAM cell stores a bit without dissipating any active power, unlike its counterpart dynamic random access memory or DRAM [1]. Even though a DRAM cell is smaller than an SRAM cell, the latter has two important advantages: (i) once a bit has been written in an SRAM cell, active power is not spent during retention of the stored bit, and (ii) it is faster to read and write from an SRAM cell. However, SRAM cells are volatile, i.e., they require a positive supply-voltage to retain data. A positive supply-voltage results in leakage-power dissipation to retain SRAM cell data. In earlier technologies, data bits could be stored in SRAM at negligible leakage-power consumption. However, with technology scaling, it has been observed that a significant fraction of total power is wasted as leakage-power in cache or SRAM [2]. In the future, this trend is expected to worsen, unless the threshold voltage of transistors is increased. The trends of SRAM size and leakage-power as a function of year (technology) are illustrated in Figure 1.1 (*source: Intel*) [2].

Observe that, with Moore's law SRAM size on the microprocessor has increased with time. With Moore's law and technology scaling, the SRAM leakage gets higher, as illustrated in Figure 1.1(c). The SRAM leakage-power is even more significant for low duty-cycle applications, like sensors [3]. This is because low duty-cycle applications store the state in SRAM and turn off the other hardware blocks in the chip to save power. If the storage time is large, then the time-averaged active power is negligible compared to the SRAM leakage-power.

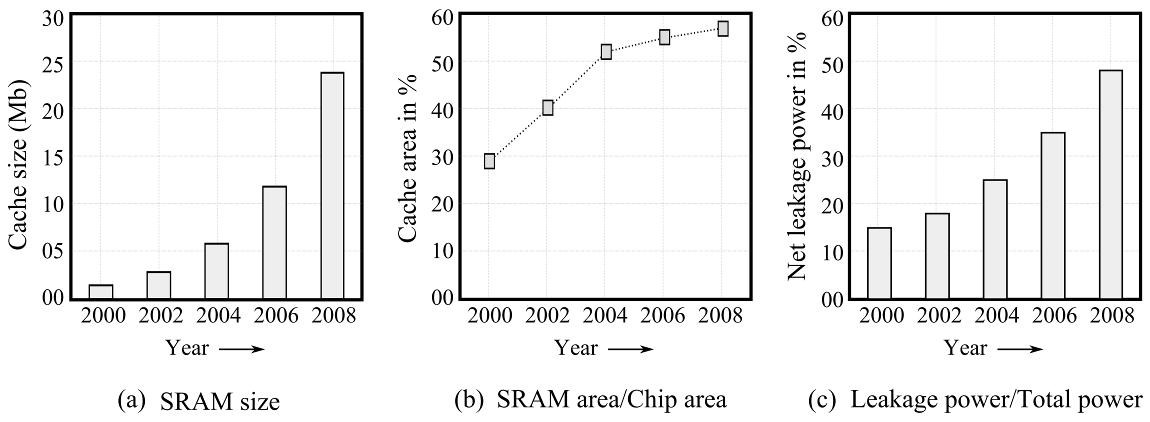


Figure 1.1: (a) The increase in cache size with technology node or year is illustrated. This increase is almost exponential. (b) SRAM area in percent of the total chip area as a function of year is illustrated. Observe that cache size has the largest percentage of the total chip area. (c) Increasing leakage power contribution to the total power consumption is shown as a function of time (*source: Intel*).

At a broad level, two techniques can control or reduce the leakage-power problem in SRAM: (i) circuit-level design techniques, and (ii) system level design techniques such as supply voltage reduction. Of these, the circuit level techniques usually change one or more parameters of the SRAM circuit. For example, these techniques include modification of SRAM cell's transistor-parameters like threshold voltage or sizing [4], addition of a sleep transistor or control gate to reduce standby leakage [5–7], usage of asymmetry in the SRAM cell design [8], proposed usage of a different transistor than traditional MOSFET [9], and a change in the SRAM cell structure to enable ultra-low voltage operation [10]. Note that this list is not complete, and other ideas can be found in the literature. In contrast, at the system level, the SRAM cell is not altered but architecture-level changes are introduced. The most common architecture level technique is supply voltage reduction (scaling) of inactive SRAM, without affecting the stored bits [11–15]. This technique works since leakage-power has strong dependence on the supply voltage.

This work focuses on *system level* techniques to reduce SRAM leakage-power, therefore, these techniques are examined in detail. The trendy supply voltage scaling technique reduces the leakage-power. However, supply voltage scaling increases failure-rate of the stored data. Stored SRAM-cell data is subjected to the following failure mechanisms:

1. Soft-errors due to cosmic particles or alpha particles from die-packaging [16].
2. Parametric failures which include read-upset, write-failure, write-time failure, access-failure, and hold-failure due to process-variations [17].
3. Supply voltage noise induced failures [18, 19].
4. Gate-leakage fluctuations due to trapped charge in gate oxide [20].

These failure mechanisms increase with supply voltage reduction. Soft error increase with voltage reduction has been extensively reported in the literature [16, 21–24]. Read, write, or hold (store) operation failures increase with supply voltage reduction. This has been reported in many references [10,17,25–27]. Supply noise induced failures are usually tackled by a 100mV voltage margin [18, 19]. Finally, gate-leakage fluctuations, similar to random telegraph signal, change the minimum voltage at which an SRAM cell can work. This phenomenon, termed as erratic fluctuations, was reported by Agostinelli et al. [20]. Thus, *any straightforward supply voltage scaling based leakage-power reduction is achieved at the cost of lower data-reliability.* An alternate interpretation is that the lower leakage-power is achieved by an increase in data-failure rate.

On the other hand, if the supply voltage is kept at a high-level to mitigate the above failure mechanisms, then leakage-power will be high. Leakage-power increase is aggravated by two more reasons: (i) Cache-size typically increases with technology generation to provide faster computing, (ii) to compensate for process variations, the SRAM area is not (geometrically) scaled fully with technology [17, 25, 27]. A high supply voltage, with large cache-size and large SRAM cell area, leads to significant leakage-power.

At the system level, coding and error-correction have been used for communication since decades [28–31]. In fact, it was noted that this error-correction can be used for storage since a bit-level memory can be thought of as a binary input binary output channel. Not surprisingly, these codes have also been used in SRAM as an indicator of soft-error events [32–34]. However, till date the usage of error-correction in SRAMs is only to correct (or detect) a single-bit in error [33]. In error-correction coding (or channel coding), usually redundancy (parity check) is introduced to combat against bit-errors. In a probabilistic setup, the amount of redundancy determines the *decoding error probability* and the *storage efficiency* (in bits/cell) – thereby introducing a trade-off between the two quantities.

A technique called “scrubbing” has also been studied in the literature [34–38].

Among the error mechanisms mentioned before, soft-errors, supply noise induced errors, and fluctuating gate-leakage induced errors randomly happen as a temporal phenomena. Thus, the number of induced bit-errors increases on average with the storage time. If an error-correction code is used, any errors in the stored data can be periodically checked and corrected to combat bit-error accumulation. This procedure is commonly called as *scrubbing.* In this work, it will be called as *data-refresh.* This refresh is triggered by any errors present in the stored bits and it reduces the probability of decoding error at the expense of extra power consumption.

In a probabilistic setup, while supply voltage reduction causes the bit-error probability to increase, it can be compensated by suitable choices of error-correction code and data-refresh rate. To the best of our knowledge, the trade-offs between leakage-power, supply voltage reduction, data-refresh rate, decoding error probability, and error-correction code has not been studied in the literature. Accordingly, these trade-offs will be studied in this work. The trade-off study procedure is as follows. The *leakage-power* – including redundancy overhead, coding power, and data-refresh power – is the cost-function to be minimized. The failures will be modeled in a probabilistic setup and the constraint is set by an error-probability target. In particular, the error-probability target is set by soft-errors at the supply voltage $V_{dd} = 1.0$V and a single-error correction code. [1] The supply voltage, the data-refresh rate, and the error-correction code will be treated as variables chosen to optimize the leakage-power cost function. Using this optimization principle, the important contributions of this work are described in the next section.

[1]A single-error correction code is chosen for target error probability since this is a common error-correction mechanism used in contemporary SRAM [33].

To understand the limits of leakage-power reduction, “standby-mode” of an SRAM is described next. This concept was introduced by papers using supply voltage scaling to reduce the leakage-power. An SRAM module which is in “no-operation” ON mode is classified as a *standby SRAM.* In this mode, the SRAM retains the data, but it is not accessed for read or write operations. The only task in this mode is to retain the stored data. The system level leakage reduction techniques, like drowsy-cache [11], exploit the fact that read and write activity are absent in the standby-mode, and thus a lower (than active-mode) supply voltage level can be used to reduce the leakage-power till storage (hold) failures begin [11, 12, 15]. Because two supply voltages are assumed in such solutions – one for standby-mode and one for normal operation – these solutions are labeled as *dynamic voltage scaling* (DVS) techniques. If a dynamic supply voltage is not available, then this solution *cannot* be used.

Within the topic of SRAM leakage-power reduction, the contributions of this work are classified into two cases: (i) When a dynamic supply voltage is not available or not desired, and (ii) when a dynamic supply voltage can be used. Previously, while accounting for various error mechanisms, parametric failures were mentioned. Various parametric failures correspond to the read, write, and hold operations. In case (i) of voltage scaling, all parametric failures should be counted for various supply voltages. In contrast, in case (ii) of voltage scaling, only the *hold-failure* among parametric failures should be counted for various supply voltages. This is because read and write operations are assumed to happen at a high supply voltage. The main results are presented next.

1.1 Contributions

The contributions of this work are listed below. The supply voltage assumptions are specified when necessary.

- Error models which combine various failure mechanisms are developed, while accounting for data-lifetime, and spatially fixed or random nature of these errors. Spatially fixed errors, where the erratic location can be known, are easier to correct than spatially random errors. An error-probability constrained optimization framework is developed, which accepts SRAM cell parameters (like leakage-power and error probabilities) for various supply voltages as input and optimizes leakage-power over supply voltage, error-correction code, and refresh-time (see Chapter 2).

- For exemplifying results, the supply voltage dependencies of failure mechanisms, a key ingredient in optimization, are estimated by circuit-level Monte Carlo simulations and low-complexity macro-models. These macro-models use simple statistical techniques to extrapolate error probabilities (see Chapter 3).

- Using error probabilities estimated by circuit level simulations, it is shown that data-refresh and stronger error-correction codes can reduce the supply voltage significantly, without any increase in the decoding error probability. For an approximate idea, in 90nm CMOS technology simulations, the supply voltage can be reduced to 0.3V. The leakage-power per cell at 0.3V is approximately 94% lower than that at 1.0V. Simulation results from the 90nm CMOS technology and 65nm CMOS technology are presented (see Chapter 4).

For case (ii) of voltage scaling, when a different supply voltage can be used for standby operation, the error mechanisms are dominated by hold-failures (among parametric failures) and soft-errors. In this special case, the following results are shown and verified using experiments and simulations with *twenty-four fabricated* chips in Chapter 5:

- Fundamental lower bound on the leakage-power reduction in terms of the hold-failure distribution using techniques from *information and coding theory* are established. The distribution of hold-failure is learned from custom-built fabricated chips. This leakage-power lower bound as a function of the experimental-chip's index will be presented.

- Due to latency-constraints on decoding, power reduction as a function of the block-length is studied. A low complexity Hamming code was chosen for implementation. Its leakage-power reduction performance in comparison with the fundamental bounds will be presented. The $[31, 26, 3]$ Hamming code based implementation's power reduction closely tracks the optimum power reduction, and this desirable property justifies our low-complexity implementation.

Other interesting measurement results are also presented. They include bounds on the empirical correlation of neighborhood SRAM cells, and scatter plots to examine dependencies between parameters of interest.

1.2 Related work

Voltage scaling: Voltage scaling for data-storage in standby-SRAM was first proposed by Kim et al. [11]. Later, it was shown by Qin et al. [12, 13] that there is a minimum supply voltage for an SRAM cell above which it can retain (store) a bit reliably (in the absence of other failures). This minimum retention voltage is termed as *data retention voltage* or DRV in the literature. Due to process variations, the DRV exhibits a distribution [13]. The largest DRV on a chip is the supply voltage needed for ensuring reliable retention across all SRAM cells. The temperature variation of this largest DRV parameter was studied by Wang et al. [39]. Multiple standby voltages have also been proposed in the literature [15], but our results in Chapter 5 show that a single standby supply voltage is optimum in a statistical setup.

Soft-errors: Soft-errors were first observed in DRAMs [40]. Later companies like IBM and Intel did a series of experiments to characterize soft-errors for SRAM arrays. A summary of experiments at IBM is presented by Ziegler et al. [16] and it is a recommended reading for understanding various soft-error issues faced while designing circuits. Freeman proposed a canonical circuit for soft-error rate estimation using the critical charge model [41]. The critical charge model uses a current source, which was later validated by Hazucha and Svennsson [23, 24]. The shape of this current source has been modeled as a double exponential in some other works (for example [42]). A different approach to estimate soft-errors using node capacitance was introduced by Merelle et al. [43] which uses complicated three-dimensional CAD analysis. Comparisons of various soft-error models has been studied by Naseer et al. [44]. In this work, the Freeman model will be used because it successfully models SRAM failure rates as demonstrated by Hazucha and Svennsson [23, 24].

Parametric failures: Parametric failures affect the read, write, or hold (store) ability of SRAM cell. There are five types of parametric failures that need to be accounted. They are: (i) read-upset failure in which the stored bit flips on read operation, (ii) write-failure in which a bit cannot be written into the cell, (iii) hold-failure in which stored bit is not retained in the cell, (iv) access-time failure in which stored bit cannot be read within a specified duration, and (v) write-time failure in which a bit cannot be written within a specified duration [25, 26, 45]. Typically these failures start to happen as voltage is reduced or as process-variations increase [10, 17, 26, 45, 46]. Because of process-variations, some cells fail earlier than the others, and thus failures can be modeled in a probabilistic or statistical setup [25, 45].

Some comments about failure probability estimation are in order. Within parametric failures, the access-time and write-time failures can be made negligible by choosing read-time and write-time to be large enough. The distribution of critical write-time and read-time across cells has been modeled by Roy et al. [25], and Agarwal and Nassif [45]. The other three failures are estimated using noise-margin techniques (see Chapter 3 for detailed discussions). These noise-margin techniques have been proposed and discussed in detail in the literature (for example, see [17, 26, 45]).

Erratic fluctuations: Trapping and de-trapping of charges in Si-SiO_2 interface causes significant $V_{cc|\min}$ fluctuation in SRAMs. By definition, $V_{cc|\min}$ is the minimum voltage at which read, write, and hold operations are successful in an SRAM cell. If the $V_{cc|\min}$ becomes larger than the supply voltage, and if the bit is accessed, then the bit will be in error. This phenomenon, called erratic fluctuation, varies temporally and spatially. Further, this erratic fluctuation is “soft” in the sense that cell becomes normal (not erratic) after some random time. An essential reference for understanding erratic fluctuations is by

Agostinelli et al. [20]. From a modeling perspective, little is known about erratic fluctuation time constants and magnitudes. It is modeled by introducing a gate-leakage current varying as a random telegraph noise signal.

Supply noise: Presence of noise in supply voltage implies temporally (and spatially) varying voltage levels. For using noise statistics in a probability-aware optimization framework, suitable (statistical) characterization of the supply $v(t)$ and dynamic stability metrics are needed. The statistical (correlation properties) of supply voltage noise have been successfully explored by Alon and Horowitz [19]. Dynamic stability of SRAM has been studied in a limited setting of soft-particle strikes by Zhang et al. [47]. However, their treatment is far from complete. The classical approach to ensure supply noise margin is by adding an overhead of 100mV to the supply voltage. This framework assumes the same 100mV margin to prevent supply noise induced errors.

Cell hardening: A common approach to tackle soft-errors is by making SRAM cells more tolerant to particle strikes. The common approach involves making larger or more complicated SRAM cells or adding a capacitor to ensure tolerance against energetic particles. A detailed summary of these techniques can be found in the paper by Roche and Gasiot [48]. To this end, it must be noted that even the hardened SRAM cell will have an increase in soft-error rate as the supply voltage is lowered for reducing leakage. Thus, techniques like data-refresh or error-correction will be *needed* to enable supply voltage reduction.

Scrubbing or data-refresh: Data-refresh, popularly called by the ruffian name *scrubbing*, was proposed by Saleh et al. [35] for memories. Its necessity in cache/SRAM was examined by [36]. Some error-probability expressions on error-probability within a refresh-period with the use of single or double error-correcting codes were explored by Bajura et al. [38].

Coding theory: Channel or memory coding for arbitrary reliability was introduced by Shannon in his classic paper on communication theory [28]. Some practical algebraic methods to encode for decreasing decoding error probability were first proposed by Hamming [29]. Reliable storage capacity and coding for storage have been studied by Heegard and El Gamal in the presence of erasures and errors [49]. A succinct reference for channel coding in Information Theory is the book by Cover and Thomas [50]. An exhaustive reference for algebraic error-correction codes is the book by Lin and Costello [51]. In this work, coding theory will be used as a technique to reduce power in storing data in SRAM cells has not been studied by any of these work. Further, coding complexity needs accounting, since the whole error-protected SRAM consists of coding and storage. In channel coding results, usually the complexity (or power) of encoder and decoder is not an issue while deriving capacity. The fresh idea of Green Codes by Grover and Sahai [52] can be included in the optimization setup presented in Chapter 2. This is left as a future work for latency-tolerant large memories where LDPC codes can possibly be a reality.

Standby SRAM: Chapter 5 naturally extends the standby SRAM work proposed by Qin et al. [12, 13]. This solution is motivated from the perspective of standby storage in SRAM, where leakage-power minimization leads to total power minimization. Using the voltage scaling approach, it has been shown that any SRAM cell has a critical voltage (called the data retention voltage or DRV) at which a stored bit (0 or 1) is retained reliably [12]. The intra-chip DRV exhibits a distribution due to process-variations. In order to minimize leakage-power without observing hold-failures, a standby supply voltage equal to the highest

DRV among all cells in an SRAM can be used. This is a "worst-case" selection of the standby supply voltage. The leakage-power reduction from $V_{dd} = 1000$mV to the largest DRV voltage in many test-chips has been studied in detail by Qin et al. [13, 53]. This work naturally extends these results by power reduction below the worst-case strategy. A supply voltage lower than the largest DRV voltage is chosen, with appropriate error-control coding to overcome ensuing errors. Under this approach, the supply voltage is flexible and the leakage-power per useful bit can be (fundamentally) reduced over the choice of supply voltage. For a detailed understanding of DRV-distribution based leakage-power reduction, previous work is recommended for reading [12, 13, 53].

1.3 Assumptions and Notation

1.3.1 Simplifying assumptions

Multiple-bit failures have been reported in sub-90nm SRAMs (e.g., [21, 54, 55]). Correlation in failures can usually be exploited by coding. However, the dependencies between these failures are not well known. Address permutation schemes can interleave SRAM cells with negligible energy overhead and make the failures (approximately) statistically independent [34, 56]. For simplicity, address interleaving is assumed. The energy and delay cost of address interleaving can be made negligible by permuting the address lines of SRAM. This is highlighted in Example 1.3.1. Please note that even though interleaving can be simply performed by permuting the address lines, it will have power (energy) impact while accessing SRAM cells, especially when word-level access is used.

Example 1.3.1. *Let $x_1^l := (x_1, x_2, \ldots, x_l)$ and $y_1^n := (y_1, y_2, \ldots, y_l)$ be the row and column* *address bits for any SRAM block, respectively. This array will be* $(2^l \times 2^l)$ *in size. Consider the address permutation where these addresses are mapped to* $(y_l, x_{l-1}, y_{l-2}, x_{l-3}, \ldots)$ *and* $(x_l, y_{l-1}, x_{l-2}, y_{l-3}, \ldots)$*. Simply speaking, the least significant bits are made into most significant bits and the resultant address bits are "mixed." For example,* (x_1, x_2, x_3, x_4) *and* (y_1, y_2, y_3, y_4) *are mapped into* (y_4, x_3, y_2, x_1) *and* (x_4, y_3, x_2, y_1)*. These mappings are bijective, or one to one and onto. As a result, each mapped address corresponds to a unique unmapped address.*

On a physical layout level, SRAM cells which are close will have addresses that differ in least significant bits. By flipping, these bits are mapped into most significant bits, causing the mapped addresses to be far apart. Mixing is done to ensure two dimensional interleaving.

For implementation purposes, only bounded-distance decoding based block codes are considered. Thus, LDPC, Turbo, or Convolutional codes are not considered. This is motivated by block-length and latency considerations. SRAM blocks are typically organized into blocks of size ranging from 32×32 to 512×512 [57]. This arrangement naturally puts a restriction on the block-length of any error correction code. For binary channels, graph based LDPC codes typically outperform the conventional bounded distance decoding codes for large block lengths, where bit-error probability is large. [2] As will be seen in Chapter 4, the error mechanisms in SRAM have low bit-error probabilities. Owing to this reason, only bounded-distance decoding based codes are considered. Asymptotic trade-offs between a

[2]For example, for a binary symmetric channel with crossover probability p, the asymptotic storage capacity achieved by bounded distance decoding is $1 - H_2(2p)$, where $H_2(.)$ is the binary entropy function in bits. The storage capacity (irrespective of coding strategy) is $1 - H_2(p)$. If p is close to zero, their relative difference $\frac{H_2(p)-H_2(2p)}{1-H_2(p)}$ is negligible.

graphical code's rate and asymptotic coding-energy scaling models have been studied by Grover and Sahai [52]. Using these models, supply voltage reduction and LDPC coding trade-offs can be studied in an asymptotic setting to establish upper bounds on power reduction. This has been left as a future work.

Supply noise issue is usually addressed by a 100mV extra margin on the supply-voltage, to ensure proper functionality of SRAM [18, 19]. In this work, the same approach will be adopted. Thus, if v^* is found to be the leakage-power optimal supply voltage for SRAM, then $(v^* + 100\text{mV})$ will be the actual supply voltage. To simplify the exposition, leakage-power comparisons will be made *without* adding the noise margin. The extra 100mV margin will not change the nature of leakage-power optimization results. *It must be noted that this is not the power optimal strategy.* For example, in the case of standby SRAM, the supply noise will be much smaller due to zero circuit activity. Usually supply noise is observed at the clock edge, when the active logic blocks draw a large (but indefinite) amount of current [19]. Difficulties in moving away from this worst-case strategy and exploiting the statistics of supply noise will be discussed in Chapter 3.

Unlike traditional circuit optimization works, the focus here is on system level optimization without changing the SRAM cell parameters like transistor threshold voltage V_T, transistor channel length L, or transistor width W etc. This simplification leaves the cell-design and cell-area unaffected for the SRAM cell. The redundancy overhead of error-correction code will be accommodated in the optimization cost function in the next chapter. The exploration of joint circuit and system optimization has been left as a future work.

1.3.2 Notation

The supply voltage will be denoted by v and any current will be denoted by $i(t)$. In the special case of standby SRAM in Chapter 5, the standby supply voltage will be denoted by v_S. Average leakage-power (over random realizations of SRAM cells) at supply voltage v is denoted by $P_l(v)$. Data-lifetime and refresh time are denoted by t_0 and t_r, respectively. The letter E is reserved for energy (of various types). The acronym ECC will stand for a generic error-correction code. Leakage-power per useful bit including the coding overheads will be referred to as *power per bit* and denoted by $\mathcal{P}_b(v, t_r, ECC)$. High supply voltage stands for $V_{dd} = 1.0$V. Error probabilities (of various types) will be denoted by the letter p and the letter r will be used for bit-error rate. The binary entropy function is denoted by $H_2(p)$. The letter $\mathcal{E}$ will be used to denote error events. The symbols $\mathbb{E}$ and $\mathbb{P}$ will be used for statistical expectation and probability, respectively. Any vector $(x_1, x_2, \ldots, x_j)$ will be denoted by x_1^j.

The standard threshold voltage 90nm CMOS technology will be called as 90nm CMOS technology or just 90nm technology. Similarly, the standard threshold voltage 65nm CMOS technology will be called as 65nm CMOS technology or just 65nm technology. The SRAM cell sizing cannot be disclosed due to non-disclosure agreement. Most of the presented simulation results are normalized due to the same reason.

A bounded distance decoding based error-correction code will be represented by the $[n, k, d]$ parameters [30,51]. Block length (total number of bits) is denoted by n, number of information bits is denoted by k, and d denotes the minimum Hamming distance of the code. A bounded distance decoding based code will detect up to $(d-1)$ errors and correct up to $u := \left\lfloor \frac{d-1}{2} \right\rfloor$ random bit-flips. The general probabilistic model of SRAM cell is illustrated in Figure 1.2. The SRAM cell has a binary bit $X \in \{0, 1\}$ as input. The output is another

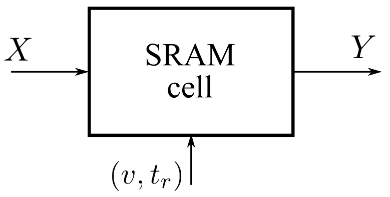


Figure 1.2: The channel model or probabilistic model of an SRAM cell is illustrated. X is a binary input and Y is a binary output. The conditional probabilities ($\mathbb{P}[Y = y|X = x]$) depend on the supply voltage v and time t_r.

bit $Y \in \{0, 1\}$. The error probability $Y \neq X$ is controlled by the supply voltage v and the data-refresh time. Even though $Y \in \{0, 1\}$, it will be shown later that a fraction of errors in SRAM cells can be converted into erasures. Thus,

$$\begin{aligned} Y &= \times, \text{ with probability } p_x(v), \\ &= \bar{X}, \text{ with probability } p_e(v), \\ &= X, \text{ otherwise.} \end{aligned} \tag{1.1}$$

The $\times$ symbol stands for 'don't care' or an erasure. The error probability $p_e(v)$ depends on t_r as will be shown later.

The error-correction code and data-refresh based SRAM block diagram is illustrated in Figure 1.3. The bit-vector B_1^k for storage is encoded into $X_1^n = f(B_1^k)$. The vector X_1^n is stored in n independent and identically distributed (i.i.d.) SRAM cells with probabilistic model as described in (1.1). At each refresh cycle, the output bits Y_1^n are decoded into an estimate $\widehat{B}_1^k$ of the vector B_1^k. This estimate is re-encoded and stored back in the SRAM cells. With the notation in place, the overview of optimization framework is

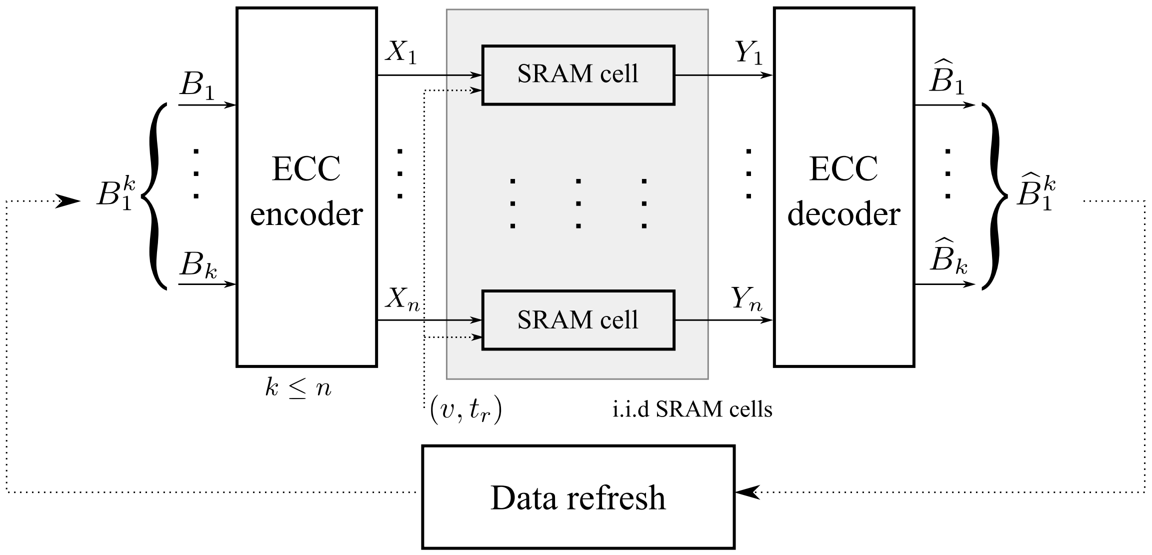


Figure 1.3: The low leakage-power SRAM architecture studied in this work is illustrated. Information bits B_1^k are encoded into $X_1^n = f(B_1^k)$. Then X_1^n is stored in n i.i.d. SRAM cells. At each refresh cycle, the output bits Y_1^n are read and decoded into $\widehat{B}_1^k$. This estimate of B_1^k is re-encoded and stored back in the SRAM cells.

discussed in the next chapter.

Chapter 2

Optimization framework

2.1 Overview

As envisioned, the optimization problem has a leakage-power per stored bit (*power per bit*) cost-function which will be optimized over the choices of refresh time t_r, error-correction code, and supply voltage v. The constraint is set by a decoding error probability target. The cost function includes the refresh power overhead. The basic principle used to save leakage-power is supply voltage reduction. As supply voltage v is reduced, average leakage-power of SRAM cells decreases. The disadvantage of supply voltage reduction is an increase in the SRAM cell failure probability. [1] Recall that the prominent error mechanisms consist of parametric failures, supply noise induced failures, soft-errors, and oxide trap-charge induced SRAM $V_{cc|\min}$ fluctuations. Among these errors, parametric failures do not accumulate with time, while other "noise" phenomenon based errors accrue with time. For these errors, system level techniques like error-correction codes and periodic data-refresh

[1] As discussed before in Chapter 1, the error mechanisms will be modeled in a statistical setup.

will decrease the decoding error probability of stored SRAM data. Both these techniques add power and storage (redundancy) overhead to the overall system. The tradeoff between these overheads and leakage-power reduction has to be explored.

The optimization constraint is that the decoding error probability of an SRAM block should be equal to the decoding error probability associated with $[31, 26, 3]$ Hamming code based SRAM block at a supply voltage of $v = 1.0$V after the data-lifetime t_0. All Hamming codes fall into the category of single-error correcting double-error detecting (SEC-DED) codes. A SEC-DED code is chosen for target error probability since this is a common error-correction mechanism used in contemporary SRAM [33]. In this work, a data-lifetime of $t_0 \geq 1$sec is considered for the 90nm technology simulations, and a data-lifetime of $t_0 \geq 10$sec is considered for the 65nm technology simulations. The necessity of data-lifetime is explained next. Phenomena like soft-errors accumulate temporally and their timestamps are well modeled by discrete independent increment process. Thus, the decoding error probability will increase as the data-lifetime increases, since the probability of a bit in error increases with storage time. This bit-error probability coupled with error-correction code, will determine the decoding error probability. Therefore, the decoding error probability depends on the data-lifetime of interest. In this work, the data-lifetime is treated as a input parameter to the optimizer. Observe that the target error probability will increase as a function of this input parameter.

The optimization framework has the following ingredients: (i) a range of supply voltage v, (ii) average SRAM cell leakage ($P_l(v)$), (iii) average SRAM cell soft-error rate ($r_s(v)$), (iv) the spatial parametric failure probability ($p_{pf}(v)$), (v) the supply noise induced error rate ($r_n(v)$), (vi) the oxide trap-charge assisted error rate ($r_{ef}(v)$), (vii) the data-lifetime parameter t_0, (viii) SRAM cell parameters such as read and write energy (E_r and E_w, respectively), and (ix) ECC parameters such as block length, information bits, minimum distance, and encoding and decoding energy. These parameters, except (ix), are expected as an input by the optimization program. Hamming and BCH error correction code families will be used as a variable in optimization [30]. A schematic diagram of the framework is shown in Figure 2.1. The encoding and decoding energy for error-correction codes were estimated using parity check complexity by standard cell library implementation in the 90nm CMOS technology. *The coding energy parameter is not critical, since the coding power is amortized by the data-lifetime parameter.*

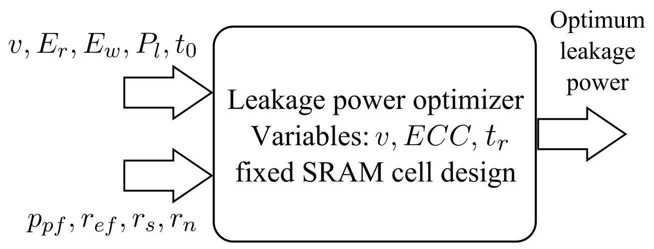


Figure 2.1: A schematic diagram of the optimization is illustrated. Failure probabilities and rates of dominant error-mechanisms, corresponding supply voltages, data lifetime, and SRAM parameters are expected as an input. The optimizer predicts the best leakage-power achievable within some families of error-correction codes.

To evaluate the performance of this optimization framework, the optimization inputs will be estimated or simulated for the 90nm and 65nm CMOS technologies (*courtesy: ST Microelectronics*). For the 90nm technology, the supply voltage is discretized to the set $\{0.3\text{V}, 0.4\text{V}, \ldots 1.0\text{V}\}$ and the optimizer computes *power per bit* on this set of input supply voltage. For the SRAM cell in 90nm technology, at 0.2V, the SRAM cell was not writeable.

The supply voltage set is $\{0.2\text{V}, 0.3\text{V}, \ldots, 1.0\text{V}\}$ for the 65nm technology. The SRAM cell in 65nm was not writeable at 0.1V. This supply-voltage quantization step is flexible in the optimization program. Only for results presented in Chapter 4, this particular discrete set is chosen. Failure rates for various error-mechanisms at these discrete supply voltages will be estimated later (see Chapter 3). The read-write energy for SRAM cell, and the ECC encoding and decoding energy will be estimated by their values at a supply voltage of 1.0V for simplicity. These estimates will be pessimistic since these energies are expected to reduce with supply voltage. However, this approach saves simulation effort, and it does not changes the nature of *power per bit* optimization results.

For the 90nm technology, a short data-lifetime $t_0 : t_0 < 1\text{s}$ is uninteresting from a leakage-power perspective. Leakage-power per cell for this technology is of the order of 1nA. Active energy needed to read or write a bit from SRAM is of the order of 10pJ. Therefore, for t_0 below 10ms range, the leakage-power will be negligible compared to the active power. While this active power problem can also be addressed within a similar optimization framework, the optimization cost-function will change and hence it is beyond the scope of this work. In problems with large t_0, where leakage-power contribution to the total power is significant, the coding energy overhead becomes negligible after amortization by t_0. This observation is particularly useful for low duty-cycle and low power applications (like sensors using energy scavenging [58]). Similar arguments can be made for $t_0 < 10\text{s}$ in the 65nm technology.

An intuitive explanation of leakage-power reduction is given next. Qualitatively speaking, depending on the supply voltage, the dominant failure mechanism is of a certain type. For the SRAM cell design under simulation, soft-errors dominate the bit-error probability and set the target decoding error probability at high supply voltage. At voltages less than 0.6V, the parametric failures dominate the bit-error probability. The soft-error rate increase with voltage reduction is not catastrophic, and it can be tackled by data-refresh, at a negligible power overhead for large t_0, till parametric failures begin to dominate. This is the prime reason why leakage-power reduction can be expected at an intuitive level. Further, as supply voltage reduces and parametric failures become dominant, extra a larger error-correction capability is needed to maintain a constant decoding error probability. This extra error-correction capability requires more parity checks or redundant bits, but parametric failure probability in the voltage-range of interest is close to zero. Therefore, only a small fraction of bits are used as parity checks[2] and it leads to efficient leakage-power reduction at a constant decoding error probability.

In summary, this approach differs from the traditional in the following way: Read, write, access, or hold may not be fulfilled for all voltages by all the SRAM cells. But as long as a large fraction of cells are functional, this negligible loss in functional SRAM cells can be made up by suitable error-correction codes. This work's main contribution is establishing trade-offs between error-correction, supply voltage, and leakage-power reduction – at a constant decoding error-probability.

[2]Informally, if $p_x \approx 0$, then the binary entropy $H_2(p_x)$ is close to zero, meaning that the Shannon capacity of the SRAM cell is equal to $1 - H_2(p_x) \approx 1$.

2.2 Probability models for SRAM cells

The probabilistic aspects of the optimization framework are discussed next. The classification of SRAM error mechanisms as errors and erasures is discussed first.

Errors and erasures

In coding theory, two types of errors are distinguished – errors and erasures. An error is a flipped bit where the SRAM cell affected by bit-flip is unknown. Strictly speaking, an erasure is a missing bit (or symbol). An SRAM cell that is *known to be* faulty can be reduced into an erasure by 'ignoring' the content of the known and faulty SRAM cell. In other words, a faulty SRAM cell will always output some value upon reading. If the value is known to be coming from a faulty SRAM cell, it can be ignored and labeled as an erasure. An erasure is similar to a don't care ($\times$). The differentiation is important since an erasure is *easier* to decode compared to an error. In simple terms, no information from a bit in error (erasure) is better than incorrect information from a bit in error (flipped bit). This can illustrated using the following example:

Example 2.2.1. *Consider the simplest repetition code (triple modular redundancy or TMR) for a single information bit with a block length of* $n = 3$*. The information bit to be stored is repeated* 3 *times in this coding scheme. Thus, codewords corresponding to bits* 0 *and* 1 *are* (000) *and* (111)*, respectively. This code can correct one error or one erasure by using majority voting on the read-out bit. In general, this coding technique can correct two erasures or one error.*

Let 0 *be the information bit, and consider the two separate cases with two erasures* *and two errors. The stored block will be* (000)*. Since the code is symmetric, without loss of generality, assume that the first two bits are affected. Therefore,* $(\times \times 0)$ *and* (110) *will be the error-affected codewords for the two cases. By ignoring the erasures or* $\times$*, the bit* 0 *can be successfully decoded. However,* (110) *decodes to the incorrect bit* 1 *when using the (optimal) majority decision rule.*

Decoding errors and erasures together was studied by Forney under the concept of *generalized decoding* [59]. Some binary error-correction code families (e.g., BCH codes) jointly decode errors and erasures (generalized decoding). In generalized decoding, if an error-correction code has minimum Hamming distance d, then m_e-errors and m_x-erasures can be corrected if,

$$2m_e + m_x < d. \tag{2.1}$$

Loosely speaking, two erasures are equal to one error.

In contrast, let *specialized decoding* be the setup where defective SRAM cell bits are treated as errors. With m_e-errors and m_x-erasures, the total number of bit-flips in specialized decoding will be $(m_e + m_x)$. And, the condition for correct decoding is,

$$2(m_e + m_x) < d. \tag{2.2}$$

Observe that only the total number of bit-flips are important in specialized decoding. Comparing (2.1) with (2.2), a larger set of (m_e, m_x) positive integer pairs satisfy the condition for correct decoding in *generalized decoding.* Thus, it is expected that *generalized decoding* will have a smaller decoding error probability. Quantitative comparisons on this difference will be presented later in Chapter 4.

2.2.1 SRAM cell failure mechanism classification

Using the less decoding overhead of erasures as a motivation, SRAM cell error-mechanisms will be sieved as errors and erasures. Since parametric failures happen in fixed SRAM cells (on the scale of decoding time) therefore, parametric failures can be treated as erasures by using suitable read and write patterns during decoding. On the other hand, noise-induced errors happen randomly in space (over SRAM cells) and time. This class includes soft-errors, oxide trap-charge induced errors, and supply noise induced errors. Thus, parametric failures will contribute to erasures, while other failure mechanisms will contribute to errors. While decoding, the SRAM cells affected by parametric failures can be learned by writing and reading test patterns in SRAM cells. Note that this advantage in error-resilience comes at the cost of small decoding overhead. One scheme or test pattern which reveals these parametric failures is presented next.

Example 2.2.2. *Consider any SRAM cell which has stored encoded data x_1^n (see Figure 1.3). Based on SRAM channel model, the bits y_1^n will be read out. Erasures have to be identified only if there is a parity check error while decoding. The following test patterns reveal any parametric failures.*

If the bit vector was y_1^n was read and a parity check error is detected, then an error has been detected and $y_1^n \neq x_1^n$. Then the complement $\bar{y}_1^n$ should be written (without coding) in the n SRAM cells. Then two read operations on $\bar{y}_1^n$ must be performed. The first read reveals any write-failure corresponding to $\bar{y}_1^n$. The second read reveals any read-upset failure corresponding to $\bar{y}_1^n$.

In the next step, the original bit vector y_1^n is written in SRAM cells, and read *operation is performed twice. As before, the read operations reveal write-failure and read-upset failure corresponding to y_1^n. Since each cell's failure is independent of other cells, these two test patterns are sufficient.*

If there are no parametric failures, then the parity check error must be due to noise mechanisms. A hold-failure is negligible in probability compared to these failure mechanisms and hence it is ignored.

Note that for each cell, four read and two write operations are required to learn the locations of parametric failures. This information will be incorporated suitably in the refresh energy overhead later. Further, this overhead is only required when a parity check error is observed.

Remark 1): This presented test-method is a simple first order method to detect parametric failures. It ignores any neighborhood coupling that SRAM cells may have (for example, due to leakage currents). If the neighboring cells couple, then cell by cell bit-level tests may not be good enough for detecting a parametric failure. A rich set of test-methods exist for detection of stuck-at, functional, or permanent faults in random access memories (e.g., see [60,61]).

Remark 2): If erasures occur with very low probability, then erasure locations or addresses can be stored in a separate small memory to aid the decoder. This will eliminate the necessity of real-time check with test-patterns (and simplifying assumptions in the Remark above), but it will introduce storage and latency overhead dependent on the number of parametric failures present in any SRAM block. This approach and any subsequent trade-offs are very interesting, but they have been left as a future work.

2.2.2 Error and erasure probability upper bounds

Based on previous discussions, the erasure probability p_x is given by,

$$p_x(v) = p_{pf}(v), \tag{2.3}$$

where, $p_{pf}(v)$ is the parametric failure probability at supply voltage v. Parametric failures are composed of hold-failure, write-failure, read-upset, access-time failure, and write-time failure. Let $p_h(v)$, $p_w(v)$, $p_r(v)$, $p_{at}(v)$, and $p_{wt}(v)$ be the probabilities of hold-failure, write-failure, read-upset, access-time failure, and write-time failure [17, 25, 45]. The parametric failure probability $p_{pf}(v)$ will be bounded using these individual failure probabilities as discussed next.

The probability of net parametric failure will be upper-bounded using the union bound [62]. This powerful technique eliminates the need to know statistical dependence between different error mechanisms. For any two sets A and B on which probability is defined, the union bound states that,

$$\mathbb{P}(A \cup B) \leq \mathbb{P}(A) + \mathbb{P}(B). \tag{2.4}$$

Equality holds in the union bound if the events A and B are mutually exclusive. If $\mathbb{P}(A) \gg \mathbb{P}(B)$, then the upper bound $\mathbb{P}(A) + \mathbb{P}(B)$ approximates $\mathbb{P}(A \cup B)$ well. The parametric error probability $p_{pf}(v)$ can be upper-bounded by algebraic addition of individual failure probabilities, even if the statistical dependence of different constituent failures is *unknown*. Thus,

$$p_x(v) = p_{pf}(v) \leq p_h(v) + p_w(v) + p_r(v) + p_{at}(v) + p_{wt}(v). \tag{2.5}$$

The advantage of this approach is that there is a rich set of techniques to estimate individual parametric failures. These techniques will be discussed in Section 3. Note that $p_x(v)$ does not has any temporal dependence, and hence data-refresh will not ameliorate decoding error contribution by parametric failures.

The other three major failure mechanisms are soft-errors, supply noise, and erratic fluctuations as discussed in Chapter 1. Let $r_s(v)$, $r_n(v)$, and $r_{ef}(v)$ be the error-probability rates due to soft-error, supply noise induced errors, and erratic fluctuation induced errors, respectively.

First, the error probability rate has to be converted into error probability for analysis. For this purpose, consider $r(v)$ as a generic error probability rate. Consider any SRAM cell. For a small time δt, the bit-flip (error) probability for this cell will be $r(v)\delta t$. This model assumes that two bit-flips do not strike in a small amount of time. Let L be a positive integer. After a time of $L\delta t$, the number of bit-flips in any SRAM cell is distributed according to a random variable distributed according to a binomial distribution, or Binomial$(L, r(v)\delta t)$. An error happens when there are odd number of flips (each with probability $r(v)\delta t$). Let,

$$\mathcal{E}_r(L\delta t) = \{\text{Stored SRAM bit is in error after time } L\delta t \text{ due to noise rate } r(v)\}. \tag{2.6}$$

Thus, the error probability for a bit stored in SRAM cell after a time of $L\delta t$ is given by,

$$\mathbb{P}[\mathcal{E}_r(L\delta t)] = \sum_{l=0}^{\lfloor (L-1)/2 \rfloor} \binom{L}{2l+1} (r(v)\delta t)^{2l+1}(1 - r(v)\delta t)^{L-2l-1}. \tag{2.7}$$

The expression in (2.7) is complicated for analysis. However, for $r(v)(L\delta t) \ll 1$, this expression is well approximated by,

$$\mathbb{P}[\mathcal{E}_r(L\delta t)] \approx r(v)(L\delta t). \tag{2.8}$$

Note that this expression is linear in the storage time $L\delta t$. As an example, soft-error rate is of the order of 10^{-16}/s. For times up to a year, this approximation will be valid for soft-errors.

To estimate error probability in SRAM cells, let t be any time period of interest. Then, the error-probability due to soft-errors, erratic fluctuations, and supply noise is upper-bounded (using the union bound) by,

$$p_e(v) \leq t[r_n(v) + r_{ef}(v) + r_s(v)], \qquad \text{if } p_e(v) \ll 1. \tag{2.9}$$

Observe that for $p_e(v) \ll 1$, this error probability increases with time-period t. If $p_e(v)$ is not much smaller than 1, then (2.7) will have to be used. Finally, error check and data-refresh (scrubbing) at periodic rate mitigates this error mechanism.

2.2.3 Decoding error probabilities

In this section, decoding error probabilities for the generalized and the specialized decoding methods will be computed. These expressions are valid for any error-correction code with parameters $[n, k, d]$. First specialized decoding error probability $p_{sp}(v)$ is analyzed since it is straight forward. The probability that a bit is flipped is estimated by $p_x(v)+p_e(v)$. And an error happens if the number of bit flips is more than $u := \lfloor (d-1)/2 \rfloor$. Note that an $[n, k, d]$ code can correct up to w errors. Thus,

$$\begin{aligned} p_{sp}(v) &= \mathbb{P}\left[(u+1) \text{ or more flips happen}\right], \\ &= \sum_{j=u+1}^{n} \binom{n}{j} (p_e(v) + p_x(v))^j (1 - p_e(v) - p_x(v))^{n-j}. \end{aligned} \tag{2.10}$$

If $n(p_e(v) + p_x(v)) \ll 1$, then this expression simplifies to,

$$\begin{aligned} p_{sp}(v) &\approx \binom{n}{u+1}(p_e(v)+p_x(v))^{u+1}(1-p_e(v)-p_x(v))^{n-u-1}, \\ &\approx \binom{n}{u+1}(p_e(v)+p_x(v))^{u+1}. \end{aligned} \tag{2.11}$$

The expressions in (2.10) and (2.11) give upper bound on the specialized decoding error probability. Recall from (2.9) that $p_e(v)$ depends on the time for which data has been present in the SRAM cells. Thus, if data-refresh is used, $p_{sp}(v)$ can be reduced by reducing data-refresh time t_r and hence $p_e(v)$.

In generalized decoding, errors and erasures are treated differently. If there are x errors and y erasures, then a decoding error happens in the generalized case if $2x + y \geq d$. The probability of error for the generalized decoding is given by,

$$\begin{aligned} p_{gen}(v) &= \mathbb{P}\left[x \text{ errors and } y \text{ erasures} \ni 2x+y \geq d\right], \\ &= \sum_{j=0}^{n} \mathbb{P}\left[y \text{ erasures in } (n-j) \text{ bits } \ni 2j+y \geq d | j \text{ errors in } n \text{ bits}\right] \\ &\quad \mathbb{P}[j \text{ errors in } n \text{ bits}], \\ &= \sum_{j=0}^{n}\sum_{i=d-2j}^{n-j}\binom{n-j}{i}(p_x(v))^i(1-p_x(v))^{n-j-i}\cdot\binom{n}{j}(p_e(v))^j(1-p_e(v))^{n-j}. \end{aligned}$$

If $p_x(v)$ is negligible and $np_e(v) \ll 1$, then this expression simplifies to,

$$p_{gen}(v) \approx \binom{n}{u+1}(p_e(v))^{u+1}. \tag{2.12}$$

And if $p_e(v)$ is negligible (compared to $p_x^2(v)$) and $np_x(v) \ll 1$, then this expression simplifies to,

$$p_{gen}(v) \approx \binom{n}{d}(p_x(v))^d. \tag{2.13}$$

When $p_e(v)$ is dominant, comparing (2.11) and (2.12), it is observed that generalized decoding has negligible advantage over specialized decoding in terms of error probability. However, when erasures are dominant, then comparing (2.11) and (2.13) reveals that $p_{gen}(v)$ is much smaller than $p_{sp}(v)$ (since $u = \lfloor (d-1)/2 \rfloor$).

2.3 Optimization cost function and constraint modeling

As motivated in Chapter 1, SRAM leakage-power reduction is an important problem. Accordingly, a cost function including leakage-power and data-refresh overhead will be developed. Data-refresh and error-correction coding are introduced to combat data-reliability issue. The cost function will consist of leakage-power, and data-refresh power overhead suitably normalized by data-refresh time. The data-refresh operation requires extra four read operations and two write operations per bit (see Example 2.2.2) in the case of parity check failure due to an error or erasure. The corrected bits (at most d) have to be written back as well. Recall that $[n, k, d]$ are the error correction code parameters. The number of redundant parity bits are $(n-k)$. Let E_{ECC} be the average energy consumed by the error-correction code. Then, the *power per bit* cost function, *including* the data-refresh overhead, is given by,

$$\mathcal{P}_b(v, t_r, ECC) = \frac{n}{k} P_l(v) + \frac{n(4E_r + 2E_w)}{kt_r} + \frac{E_{ECC} + nE_r + E_w d}{kt_r}. \tag{2.14}$$

The data-refresh overhead becomes negligible when data-refresh time t_r and data-lifetime t_0 are large. This assumption is reasonable since leakage-power is significant only when data-lifetime is large. For the 90nm technology, $t_0 > 1$s has negligible refresh power overhead for low complexity codes like SEC-DED. For the 65nm technology, $t_0 > 10$s has negligible refresh power overhead. These $t_0 = 1$s and $t_0 = 10$s numbers will be used in Chapter 4.

A decoding error probability target is used as the optimization constraint. At a supply voltage of $v = 1.0$V, the target decoding error probability is set by a $[31, 26, 3]$-Hamming coded SRAM cell block. As discussed in Chapter 1, only single-error correction codes are used for error control in SRAM [33]. Therefore, a single-error correction code is used to set the target error probability.

2.4 Optimization framework summary

The probability parameters $r_s(v), r_n(v), r_{ef}(v), p_{pf}(v)$, the supply voltage range, the leakage-power $P_l(v)$, and energy parameters like E_r, E_w, E_{ECC} are the inputs to the optimizer (see (2.5), (2.9) and (2.14)). For any error-correction code, since E_r, E_w, E_{ECC} get normalized by t_0 and t_r, they are not as important as the bit-error probability rates and numbers. This is because the bit-error probability decides whether a supply voltage is feasible or not for an error-correction code. Assume that all these parameters (as a function of v) are available. For each error-control code, the optimizer computes the minimum $\mathcal{P}_b(v, t_r, ECC)$ as a function of (t_r, v), at constant decoding error probability. Then the cost function can be optimized over the choice of error-control code. This optimization program can be reused for different set of input parameters.

Availability: The optimization framework is available for public use at the following website: `https://bwrcs.eecs.berkeley.edu/freshram/`

Chapter 3

Macro-modeling of failures

SRAM failure probability estimation is a challenging task due to low failure probability of the cells. [1] For example, the soft-error probability rate is of the order of 10^{-16}/bit-sec on Earth's surface. A common metric, called FIT rate (Failure-In-Time rate), is measured as the number of failures in the SRAM chip over a period of 10^9 hours. The FIT rate increases with SRAM size. Using experiments, the FIT rate due to soft-error was predicted to be $20,000$ for a 32Mb SRAM chip at sea-level in New York, USA in the 90nm technology at $v = 1.0$V supply [23]. At a per-second level, the average probability of failure can be calculated to be $FIT/(10^9 \times 3600 \times 32\text{Mb})$, which is approximately 1.73×10^{-16}/bit-sec. As motivated in Section 2, this work focuses on fixed error-probability power optimization of SRAM chips. Then, for a moderate size of SRAM chip, *many years will be needed to estimate soft-errors experimentally.* With voltage-scaling, the parametric failures become more significant. Note that parametric failures happen across die and (spatially) across bits

[1]The failure probability is low from a measurement or estimation standpoint. Such high reliability is expected out of modern SRAM chips.

on the SRAM chip. Therefore, for getting statistics on parametric failures, comparable to the failure level in soft-errors, large SRAM test arrays are needed.

Due to these difficulties, an efficient error-probability estimation method is needed for quick power comparisons at different voltages. Motivated by these concerns, this section will focus on estimation methods for all the failure methods discussed in Section 2.2 as a function of supply voltage v. The SRAM cell design itself will be assumed to be fixed. [2] For the 90nm technology, at $v = 1$V, the parametric failures are negligible and the target failure probability is set by the soft-error FIT rate. This target probability of error will be kept constant by the optimizer.

At this point a note on modeling is required. The macro-models will be developed using analytical methods, statistical distribution theory, asymptotic predictions, and Monte Carlo simulations using the ST 90nm technology toolkit. While these models are not as accurate as real silicon experiments, they provide a quick estimate of what performance parameters can be achievable. The manufactured circuit is too complicated to be completely modeled. The hope is that if used models are close enough to reality, then the calculated performance will be close to the real performance. Besides, the models greatly increase the performance calculation speed. Approximately a few thousand trials (at each voltage, for each failure) will be used to predict the distribution using analytical methods, as will be seen in the next sections. Since the optimizer and the error-probability numbers are separable, so an experimenter can replace this work's error-probabilities by his (or her) favorite numbers.

[2]It is possible to extend the optimization framework over different SRAM cell designs, but that is beyond the scope of this work, and has been saved as a future endeavor.

3.1 Soft-error rate estimation

Most CMOS circuits are charge-based, including SRAM. While storing data SRAM cells retain charge at some nodes. Any "noise" mechanism that affects this stored charge may cause errors. Energetic particles (like neutrons, alpha-particles, etc.) from radioactive particle emission in the die-packaging or generated by incident neutrons from outer-space are one such noise mechanism [16]. These energetic particles generate electron and hole pairs in the semiconductor material, to causes temporary noise currents. If the noise-current is large enough in magnitude, the stored state will change. The error is termed as "soft", since the device is not permanently damaged, but only the stored bit (or data) is in error [16]. If the stored charge in CMOS circuit decreases, then this soft-error rate is expected to increase. In particular, at lower supply voltages, the SRAM soft-error rate (FIT rate) increases [21,22].

For leakage-power reduction, if SRAM supply voltage is reduced during standby or active operation, then the soft-error rate increases. The objective of this section is to establish an efficient method for soft-error rate estimation as a function of supply voltage. Recall that $r_s(v)$, the SRAM state flip rate, has to be estimated. The probability if failure is simply given by $p_s(v) = t \cdot r_s(v)$, where $tr_s(v) \ll 1$ and t is the time period of data-storage. As noted earlier, the soft-error can be understood using "charge stored" and a noise-current. Using this idea, a method to estimate the soft-error rate was proposed by Freeman. The magnitude of the noise-current (as a function of time) is increased till the stored SRAM state flips. The charge delivered by this noise-current is called as the *critical-charge* [41]. It must be noted that there are many methods to estimate the rate of soft-error [44]. This work will use the critical-charge method from Freeman's work because: (i) With appropriate noise current model, its accuracy has been demonstrated by Hazucha and Svennsson [23], and (ii) It is easy to work with this model for Monte Carlo simulations.

The basic circuit to calculate the critical-charge is shown in Figure 3.1(a). Assume that logical 1 and 0 are stored as shown in Figure 3.1(a). Then, the PMOS of L-inverter will be switched off. Any positive current $i(t)$ will increase the voltage at node storing 0. If the current is large enough, then the noise voltage buildup at node 0 will cause the state 1 to flip. This mechanism is the primary cause of soft-errors. It would be obvious that a negative noise current at node storing 1 will also flip the state. However, this error mechanism contributes insignificantly to the total soft-error rate, because of lower collection efficiency of PMOS and a higher critical-charge. Further, noise-currents at access transistors can also cause state-flip, but their effect is much lower. Finally, this simplified circuit also ignores the possibility of soft-error while the SRAM cell is being accessed. This assumption is fair for long data-storage times, which is of interest in this work.

The noise-current generated by an energetic particle traveling through the transistor is represented as $i(t)$. The net noise-current charge is given by,

$$q = \int_0^\infty i(t)\mathrm{d}t. \tag{3.1}$$

The noise-current waveform depends on the physical noise-charge generation process. Using physical modeling and appropriate experiments, the noise-current waveform has been approximated to a simple two-parameter curve [23]. The shape of the two-parameter current curve is illustrated Figure 3.1(b). The parametric description of $i(t)$ is given by,

$$i(t) \equiv i(t, q, \tau) = \frac{2q}{\tau\sqrt{\pi}}\sqrt{\frac{t}{\tau}}\exp\left(-\frac{t}{\tau}\right), \tag{3.2}$$

where, q is the total charge $\int_t i(t)\mathrm{d}t$ in the current, and τ is a time-constant parameter. The parameter τ is technology and process dependent. For the 90nm and the 65nm technologies, $\tau = 90$ps will be used [21, 23, 44]. As noted previously, the critical-charge is the minimum

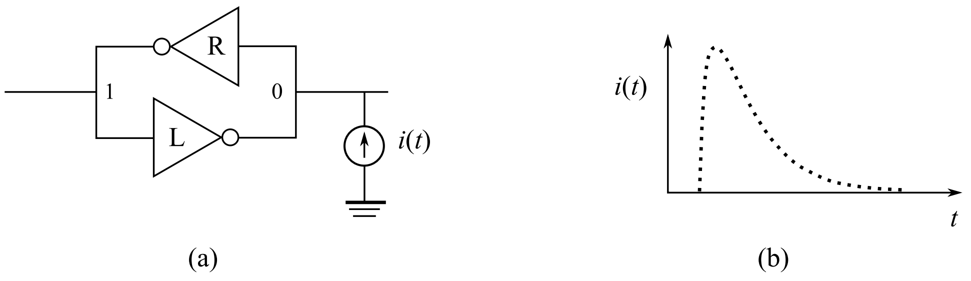


Figure 3.1: (a) The simplified circuit (without access-transistors) to evaluate soft-error rate as a function of voltage is illustrated here. The current-source $i(t)$ models the current generated by charged particle. (b) Using experimental measurements and physical models, a simple two-parameter model for $i(t)$ has been proposed in the literature.

charge needed to flip the state of the SRAM cell. Thus,

$$q_c(v) = \min\{q : \text{state of SRAM flips due to } i(t) \text{ at supply } v\}. \tag{3.3}$$

The generated charge q is a natural phenomenon and it depends on the energy of the incident energetic particle. Thus, only a fraction of incident particles cause a soft-error. The dependence of SRAM error-rate $r_s(v)$ on q_c is shown to be exponential in nature [22, 23], i.e.,

$$r_s(v) \propto \exp(-\alpha\, q_c(v)), \tag{3.4}$$

$$\Rightarrow r_s(v) = K_s \exp(-\alpha\, q_c(v)). \tag{3.5}$$

The proportionality constant K_s depends on the FIT rate of the SRAM cell. For this work, the FIT numbers from the literature will be used. It must be noted that K_s is not a fundamental constant, but depends on the altitude above sea-level, solar-flares, and other natural parameters [16]. For example, it has been reported that compared to sea-level, the soft-error rate at 44,000ft height is about 100X higher. The measured cosmic-ray flux at different altitudes, which is directly proportional to the soft-error rate, is shown in Figure 3.2 [16]. Observe that the soft-error rate in Denver is about 4X higher than the

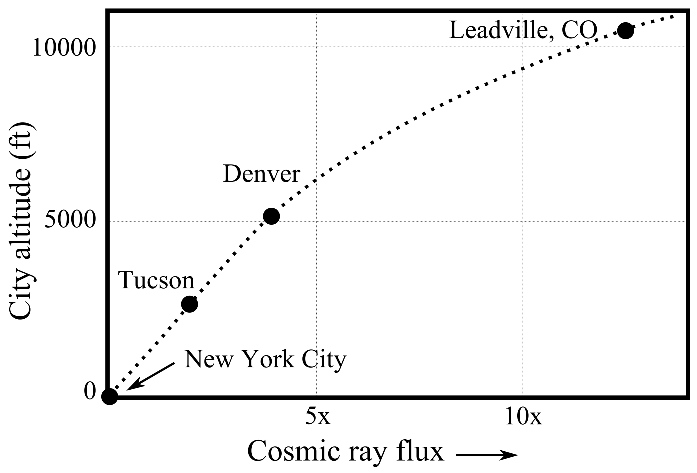


Figure 3.2: The increase in cosmic-ray flux with altitude is illustrated. The soft-error rate is proportional to the cosmic-ray flux, and thus it increases with altitude (*source: IBM*).

soft-error rate in New York City. The sea-level constant K_s will be used in this work.

So far, the modeling of critical-charge and its relationship to soft-error rate has been described. In the deep-submicron era, the role of process-variations cannot be overlooked. The inter-die or intra-die process-variations affect most figures of merit in circuits. To calculate the effect of process-variations on the critical-charge, a Monte Carlo simulation approach will be adopted as described next [63]. Many random instances of an SRAM circuit will be simulated, with appropriate technology files and parameters in an advanced circuit-simulator like Cadence Spectre, to obtain the critical-charge using the procedure described above. This procedure will yield an empirical probability distribution of the critical-charge.

Using this distribution, average expected probability of error can be computed. Since the energy-particle cross-section will remain essentially the same, therefore, the constant K_s is not expected to change.

Let the random critical-charge of any SRAM cell be Q_c (instead of a nominal critical charge q_c). Then, using Monte Carlo simulation procedure described, an empirical distribution of Q_c will be obtained. The cumulative distribution function (CDF) is defined as,

$$F_{Q_c}(q) = \mathbb{P}[Q_c \leq q].$$

The function $F_{Q_c}(q)$ simply tells the fraction of SRAM cells that will have a critical charge less than q. An example of this empirical distribution, for supply voltage $v = 1.0$V and 1000 Monte Carlo trials, is illustrated in Figure 3.3. The x-axis is normalized by the mean critical-charge $\mathbb{E}[Q_c]$. Observe that the distribution is centered around the mean, and the total spread is 26% relative to the mean. Using this distribution, the average rate of soft-

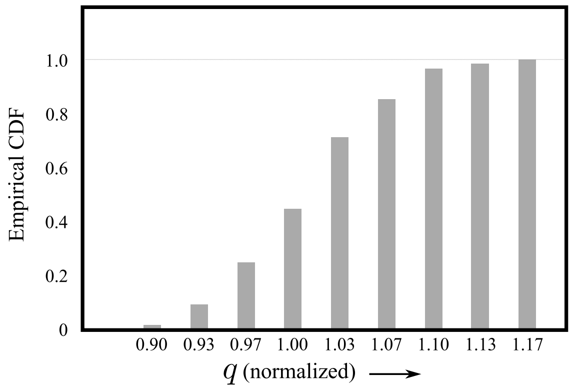


Figure 3.3: This figure illustrates the CDF of Q_c on a scale normalized by $\mathbb{E}[Q_c]$. Observe that the distribution is centered around the mean, and the total spread is 26% relative to the mean.

error can be calculated as follows:

$$\bar{r}_s(v) \quad = \quad \mathbb{E}\left[K_s \exp(-\alpha Q_c(v))\right], \tag{3.6}$$

where, $\bar{r}_s(v)$ is the average soft-error rate at any supply voltage v. The expectation or average is taken over SRAM cells or intra-cell variations. Since $\exp(-\alpha x)$ is a convex function in x for $x > 0$, therefore, by Jensen's inequality [50, 62],

$$\bar{r}_s(v) \geq K_s \exp\left(-\alpha \mathbb{E}[Q_c(v)]\right) = K_s \exp(-\alpha q_c(v)) = r_s(v), \tag{3.7}$$

where, the equality $\mathbb{E}[Q_c(v)] = q_c(v)$ is observed for all supply voltages in simulations (e.g., in Figure 3.3, this feature is observed). Thus, *the process-variation affected soft-error rate $\bar{r}_s(v)$ is always larger than nominal soft-error rate $r_s(v)$*. The quantity $\bar{r}_s(v)$ will be calculated in the experiments section in Chapter 4. In summary, a first-order model to estimate soft-error rate as a function of supply voltage v, in the presence of process-variations was presented.

3.2 Parametric failures

Any SRAM cell is designed for storing a bit, non-destructive read of stored bit, successful write by replacing stored bit, writing (replacing) a bit within a specified time, and reading stored bit within a specified time. Violation of these basic tasks lead to the following failures: (i) hold failure, (ii) read upset, (iii) write failure, (iv) write-time failure, and (v) access-time failure, respectively. These failure modes will be explained briefly and a detailed explanation of all these failures can be found in the literature [12,13,25,45,46,64]. The prime cause of these failures is process-variations, which perturbs the SRAM cell parameters from nominal design leading to a failure. In particular, the random (or systematic) fluctuations in the number of dopants, threshold voltages of transistors, gate lengths, and the oxide thickness lead to the violation of one or more of these functionalities.

It must be emphasized that the goal of this section is not to design an SRAM cell which keeps these failures at the level of soft-errors. Instead, using a Monte Carlo method, this section's goal is to estimate parametric-failure probabilities an efficient manner. Existing literature has an in-depth treatment of parametric failures, and SRAM cell design to reduce their probability [13, 25, 45, 46].

Two techniques will be used to estimate these parametric-failure probabilities. The first technique involves noise-margins obtained from suitable voltage transfer characteristics (VTC). The VTCs of interest will be between V_L and V_R (the two storage nodes in SRAM as in Figure 3.4) under appropriate DC biasing of the SRAM cell transistors. A noise-margin vector, which is an appropriate measure of SRAM cell stability, will be extracted using the VTCs. The beauty of this noise-margin vector lies in the fact that a single threshold test (noise margin > 0 in all co-ordinates) tells about the presence of a parametric failure. After obtaining relevant VTCs using Monte Carlo simulations, the probability of these tests are (relatively) easier to estimate. The second technique involves extreme value theory [65, 66]. For computing access-time and write-time when failure probability is negligible (usually smaller than 10^{-6}), a brute-force Monte Carlo simulation method is expensive. *Using only a few thousand trials, the access-time and the write-time will be estimated using extreme value theory.* This theory has been successfully used by Singhee and Rutenbar in the estimation of SRAM write-time [67]. The approach is slightly different in this work, but the underlying principle remains the same.

3.2.1 Read upset probability – p_r

If an SRAM bit gets flipped while it is being accessed, it is called as read-upset. To understand this, consider the SRAM circuit during read operation as shown in Figure 3.4. The nodes with voltage V_L and V_R store the information (or input) bit as complementary logical states (here 1 and 0, respectively). During the read-operation, the bit-line capacitors (BL cap) are kept at supply voltage v and the access-transistors are turned on [1].

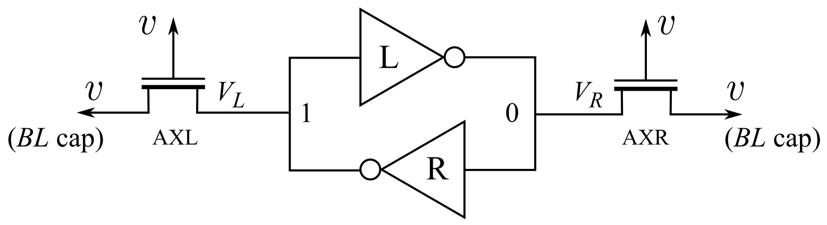


Figure 3.4: The DC bias during read operation is illustrated in this figure. The access transistors are turned on, and the bit-line capacitors are charged to the supply voltage v. The nodes with voltages V_L and V_R store the SRAM bit as complementary logical states.

During read operation, the voltage V_R will rise above the usual ground state of $\approx$ 0V. The voltage V_R is decided by ON resistances of NMOS in inverter R and the AXR transistor. If this voltage rises above the tripping point of inverter L, then the state of SRAM will flip, causing a read-upset. This event, though rare for a carefully designed SRAM cell, can happen due to process-variations.

To estimate this read-upset probability, a read-noise margin (rnm) metric will be used. Before calculating rnm, appropriate VTCs are needed and they will be explained now. The SRAM cell is put in DC bias conditions occurring during read-operation. This is illustrated in Figure 3.5(a) and (b). Even though the DC bias is identical to the read operation, the inverter L and inverter R are decoupled for *rnm* calculation. Two VTCs will be obtained as a result, corresponding to the two situations in Figure 3.5 [45, 46]. The

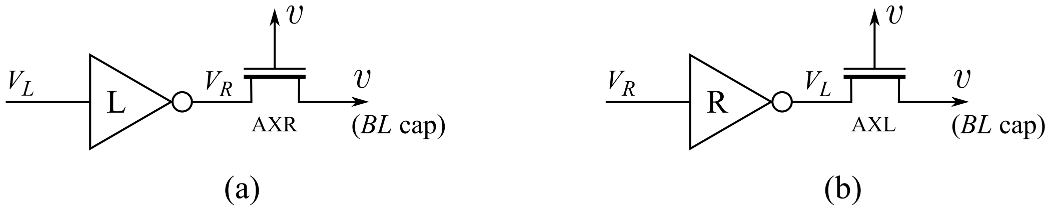


Figure 3.5: The circuit used for obtaining VTCs for *rnm* calculations are illustrated. The DC bias condition is identical to the read-operation of SRAM, but the inverter L and inverter R circuits are decoupled. (a) The voltage V_L is swept to find the VTC of inverter L. (b) The voltage V_R is swept to find the VTC of inverter R.

calculation of *rnm* from these VTCs is done by constructing a suitable “butterfly curve,” which is explained next.

The VTCs of inverter L and inverter R are overlaid in a single graph as shown in Figure 3.6. The solid curve corresponds to the VTC obtained in Figure 3.5(a). Similarly, the dotted curve corresponds to the VTC in Figure 3.5(b). Because of the highly non-linear VTC, this overlaid graph results in three stability points of (V_L, V_R) pair. Of these three points, the central point is metastable. Due to the resultant graphical structure, this overlaid graph is called as the butterfly-curve. In the two wings of this butterfly diagram, maximal squares S_1 and S_2 can be inscribed [45, 46].

Let s_1 and s_2 be the sides of S_1 and S_2, respectively. Then the read-noise margin is defined as,

$$rnm = \min(s_1, s_2). \tag{3.8}$$

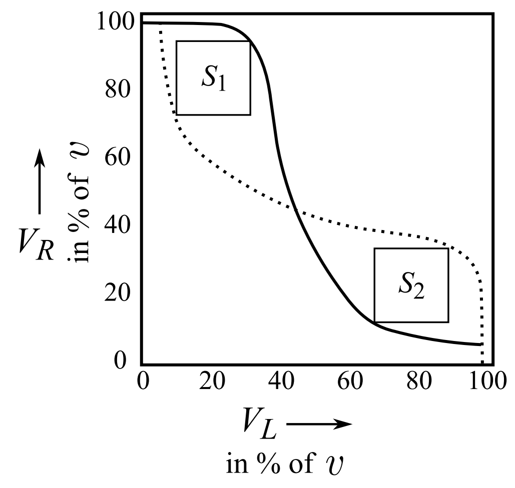


Figure 3.6: The overlaid VTCs of inverters L and R in read-operation result in a butterfly-curve structure. The largest inscribed squares in the wings of butterfly diagram are S_1 and S_2 with sides s_1 and s_2, respectively. Then, $rnm = \min(s_1, s_2)$.

It should be noted that the value of rnm depends on the supply voltage v, since the sides of S_1 and S_2 depend on v through the VTCs. If the VTCs do not form three stable points in the butterfly diagram, then during the read operation the SRAM state will reach the single stable point on the butterfly diagram. *Then rnm is defined to be negative and it will indicate a read-upset event.* Thus, for a non-destructive read operation in an SRAM cell, $rnm \equiv rnm(v) > 0$. This single threshold test reveals if a read-upset failure will be present in an SRAM cell or not.

Because of process-variations, each manufactured SRAM cell will have a different butterfly diagram leading to a unique rnm. Similar to soft-error rate, the rnm in the presence of process-variations will be modeled in a statistical setup. Using a Monte Carlo setup, different realizations of SRAM circuit will be simulated and rnm will be obtained using the method described (see Figure 3.6). The rnm will be treated as a random variable.

Let $RNM(v)$ be the random variable corresponding to the rnm of various cells at supply voltage v. Then the statistical event $RNM(v) \leq 0$ corresponds to a read-upset event. And read-upset probability will be,

$$p_r(v) = \mathbb{P}[RNM(v) \leq 0]. \tag{3.9}$$

To estimate the read-upset probability, the distribution of $RNM(v)$ will suffice. However, it remains a challenge to accurately estimate the distribution tail at failure probabilities 10^{-6} or lower. As motivated earlier in this section, a macro-model based approach will be followed to estimate these error-probabilities. Fortunately, it has been shown in various papers that $RNM(v)$ exhibits a Gaussian distribution [10, 25, 45]. This is observed in circuit simulations. For example, Figure 3.7 illustrates the empirical probability density function (PDF) of RNM at $v = 1$V with 1000 Monte Carlo trials.

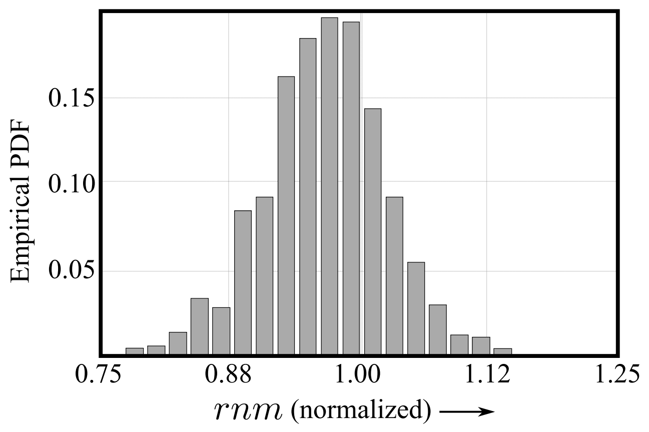


Figure 3.7: The observed empirical probability density function of RNM at $v = 1.0$V is illustrated in this figure. A Gaussian distribution is expected and observed.

The Gaussian distribution property of $RNM(v)$ has been demonstrated by other researchers for $100,000$ Monte Carlo trials [45]. To gain further insights into read-upset probability, this property can be explained if V_T-variation is the dominant factor in rnm variation. The rnm of an SRAM cell is approximately a linear function of the threshold voltages [10,45]. Since V_T variation is approximately Gaussian, therefore, its linear function will also be Gaussian [68]. *The advantage of this characterization is that only the mean and the variance of $RNM(v)$ are needed for modeling the read-upset probability.* Thus,

$$RNM(v) \sim \mathcal{N}(\mu_r(v), \sigma_r^2(v)), \tag{3.10}$$

where, $\mu_r(v)$ and $\sigma_r(v)$ can be determined by Monte Carlo simulations. Once these functions have been calculated, the read-upset probability for different voltages can be found out using the Q-function for Gaussian distribution [68], i.e.,

$$\begin{aligned} p_r(v) &= \mathbb{P}[RNM(v) \leq 0], \\ &= \mathbb{P}\left[\left(\frac{RNM(v) - \mu_r(v)}{\sigma_r(v)}\right) \leq -\frac{\mu_r(v)}{\sigma_r(v)}\right], \\ &= Q\left(\frac{\mu_r(v)}{\sigma_r(v)}\right). \end{aligned} \tag{3.11}$$

The expression in (3.11) determines $p_r(v)$ is an efficient manner. The behavior of $\mu_r(v)$ and $\sigma_r(v)$ will be revisited in the experiments section.

3.2.2 Write failure – p_w

In a bit (either 0 or 1) cannot be written in an SRAM cell, it is called as write-failure. To understand this, consider the SRAM circuit during write operation as shown in Figure 3.8. The nodes with voltages V_L and V_R are storing bits 1 and 0. To write a bit, the BL capacitors are pre-charged to complementary voltage levels, and access transistors are turned on [1].

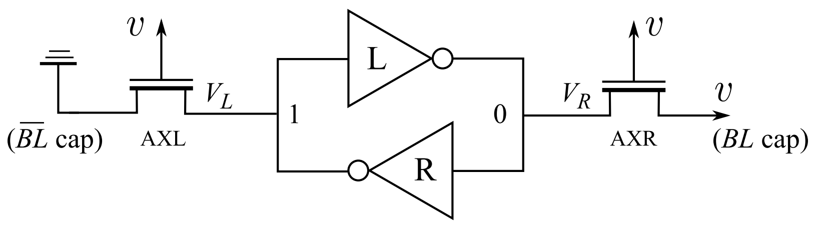


Figure 3.8: The DC bias during write operation is illustrated in this figure. The access transistors are turned on, and the bit-line capacitors are pre-charged to complementary levels for writing the bit. The nodes with voltages V_L and V_R store the SRAM bit, and they should flip as a result of write operation.

During write-operation illustrated in Figure 3.8, the PMOS of inverter R and the access transistor AXL will be conducting. The voltage V_L will fall below v, and its value will depend on the ON resistances of PMOS in inverter R and the access transistor AXL. If AXL has a much smaller ON resistance, then the voltage at V_L will not fall low enough for the bit to be written. Process-variations may cause this mismatch in ON resistances to cause a write-failure.

Similar to read-operation, write-failure can be understood using a write-noise margin (wnm) metric. To understand wnm, appropriate VTCs are needed which will be explained now. The SRAM cell is put in DC bias conditions as in write-operation. This is illustrated in Figure 3.9(a) and (b). Note that similar to the read-operation, the inverters L and R are decoupled for VTC measurement. *A crucial difference between the read-operation and the write-operation is the asymmetry of DC bias conditions.* The BL caps are pre-charged to the complementary values of initial conditions V_L and V_R. [3] Thus, in the illustrated example in Figure 3.9 where the initial values of voltages V_R and V_L are 0 and v, respectively, the BL cap voltages are set to v and 0, respectively.

[3]If respective BL caps have the same voltage conditions as V_L and V_R, then there is nothing to write.

In the presence of process-variations mismatch will be present, and symmetry between inverters L and R will be not present. Because DC bias conditions in write-operations are asymmetric, therefore two pairs of VTCs will be needed for *wnm* calculations. First pair will be obtained from DC bias illustrated in Figure 3.9. Second pair will be obtained by swapping the BL cap bias between Figure 3.9(a) and Figure 3.9(b). The calculation of *wnm* from these VTC pairs is explained next.

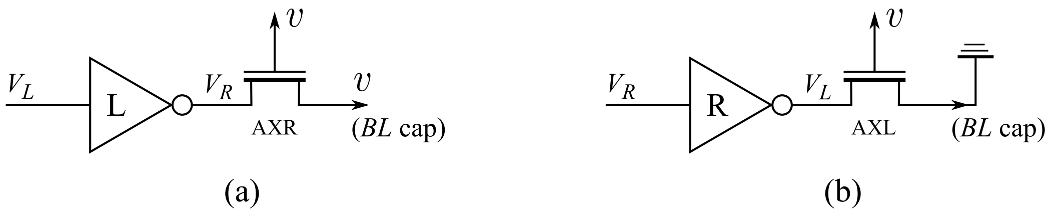


Figure 3.9: These figures illustrate the circuits used for obtaining VTC characteristics for *wnm* calculations. The bit-line capacitor bias is set as during write operation. (a) The input voltage V_L to the left-inverter is swept to find the VTC of L-inverter. (b) The input voltage V_R to the right-inverter is swept to find the VTC of R-inverter.

The two pairs of VTCs are overlaid as shown in Figure 3.10 [46]. The solid curves represent the first VTC pair. Unlike read-operation, the two curves meet at a single stable point for (V_L, V_R), and it corresponds to the bit being stored. For a successful write-operation, there should be a *single* stable point as in Figure 3.10. The square S_1 with side s_1 is a metric of write-operation stability. If $s_1 \approx 0$, and the solid curves meet at two points, then write-operation will be unsuccessful. Similarly, the dotted VTC pair results in a square S_2 of side s_2. The write-noise margin is defined as,

$$wnm = \min(s_1, s_2). \tag{3.12}$$

Similar to *rnm*, the *wnm* metric is a function of supply voltage v through the VTCs. If write-operation is not stable for either of the pair, then (by convention) $wnm \leq 0$. Thus, write-failure is equivalent to $wnm \equiv wnm(v) \leq 0$. This single threshold test reveals if a write-failure will be present in an SRAM cell or not.

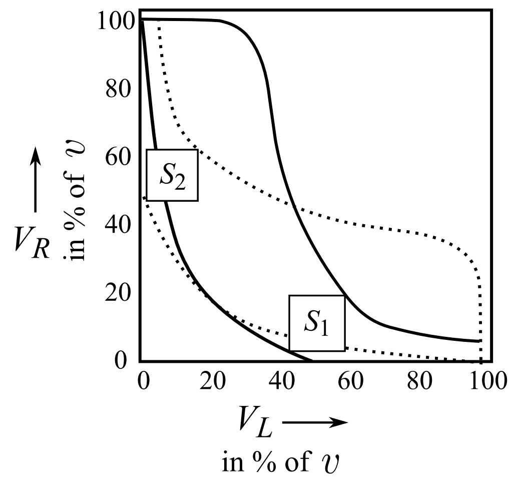


Figure 3.10: The solid curves are obtained by biasing as in Figure 3.9. The dotted curves will be obtained by swapping the BL cap bias in Figure 3.9(a) and (b). Let the largest inscribed squares in the butterfly-curves be S_1 and S_2 with sides s_1 and s_2 respectively. Then $wnm = \min(s_1, s_2)$.

Similar to read-upset, a macro-model based approach will be used to estimate the write-failure probability. Figure 3.11 illustrates the distribution of WNM at two voltages. Notice that at high-voltage (v = 1.0V) the distribution is symmetric and at low voltages (v = 0.3V) the distribution exhibits a single-sided tail. This change in distribution-shape with supply voltage v is peculiar to write-failures, among all the failure methods examined in this work.

In the voltage range 0.7V $\leq v \leq$ 1.0V, a Gaussian distribution models the WNM

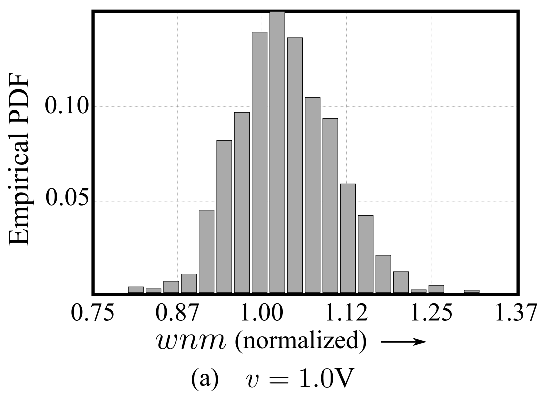


Figure 3.11: The observed empirical PDF of wnm is illustrated for two voltages. Notice that at high-voltage ($v = 1.0$V) the distribution is symmetric and at low voltages ($v = 0.3$V) the distribution exhibits a single-sided tail.

distribution satisfactorily. For $0.7\text{V} \leq v \leq 1.0\text{V}$, the failure probability is given by,[4]

$$p_w(v) \quad = \quad Q\left(\frac{\mu_w(v)}{\sigma_w(v)}\right). \tag{3.13}$$

where, $WNM(v) \sim \mathcal{N}(\mu_w(v), \sigma_w^2(v))$ is the WNM-distribution for $0.7\text{V} \leq v \leq 1.0\text{V}$. The expression in (3.13) determines $p_w(v)$ is an efficient manner. The behavior of $\mu_w(v)$ and $\sigma_w(v)$ will be revisited in the experiments section.

For the voltage range of $0.6\text{V} \leq v \leq 0.3\text{V}$, the full distribution is not estimated. Instead, the probability of $WNM(v) \leq 0$ event is estimated directly to quantify $p_w(v)$. For $0.3\text{V} \leq v \leq 0.6\text{V}$, the probability of $WNM(v) \leq 0$ event will be computed using the method described next. Consider the residual probability function,

$$R_w(t, x, v) := \mathbb{P}\left[WNM(v) \leq (x - t) | WNM(v) \leq x\right], \quad t \geq 0. \tag{3.14}$$

This equation can be rearranged using Baye's rule as follows:

$$\mathbb{P}\left[WNM(v) \leq (x - t)\right] = R_w(t, x, v) \cdot \mathbb{P}\left[WNM(v) \leq x\right]. \tag{3.15}$$

[4]The method to obtain these expressions is identical as in the read-upset probability estimation.

Observe using Figure 3.11(b) that the event $WNM(0.3) \leq 0$ is not observed directly using a few thousand Monte Carlo simulations. Thus, $WNM(v) \leq 0$ is a "rare" probabilistic event and its direct observation in Monte Carlo simulations is expensive. A predicted estimate of $p_w(v)$ will be obtained. If t is equal to x, then $\mathbb{P}[WNM(v) \leq (x-t)]$ represents the write-failure probability. If $x > 0$ is moderately large, then $\mathbb{P}[WNM \leq x]$ can be calculated efficiently using a few thousand Monte Carlo trials. Thus, if $R_w(t,x,v)$ can be estimated for $t \approx x$, then $p_w(v)$ can be computed. Consider the limit,

$$R_w(t,v) := \lim_{x \to -\infty} R_w(t,x,v).$$

In the special case when the weak limit $R_w(t,v)$ exists, it has been shown to be exponential. This result forms the basis of extreme-value theory and its further generalizations [65, 66]. Thus, if $R_w(t,v)$ exists, then,

$$R_w(t,v) = \exp(-\alpha_w(v)t). \tag{3.16}$$

Using limited number of simulations, the exponential behavior of $R_w(t,x,v)$ for small enough $x > 0$ will be examined. And, if exponential behavior exists, then $\mathbb{P}[WNM(v) \leq 0]$ will be estimated using the formulas in (3.15) and (3.16).

In practice, the exponential behavior of $R_w(t,x,v)$ was observed at voltages in the range $0.3\text{V} \leq v \leq 0.6\text{V}$. For example, Figure 3.12 illustrates the empirically observed $R_w(t,x_v,v)$ for $v = 0.6\text{V}$ and $v = 0.3\text{V}$. The point x_v was chosen to be around the 10% point or $\mathbb{P}[WNM(v) \leq x_v] \approx 0.1$, and the range of examination is $[x_v - t_v, x_v]$ where t_v is around the 1% point or $\mathbb{P}[WNM \leq (x_v - t_v)] \approx 0.01$. The results will be discussed in the experimental section.

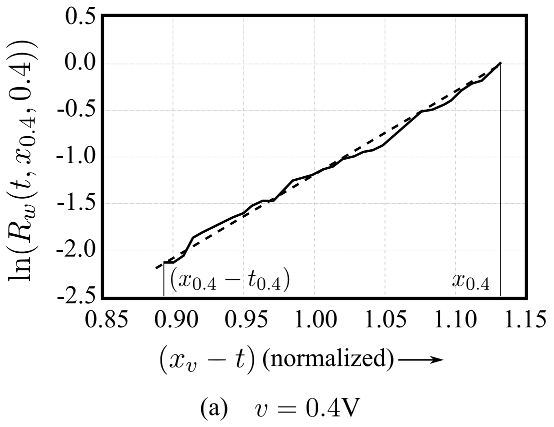


Figure 3.12: This figure illustrates the exponential behavior of $R_w(t, x, v)$ at two different supply voltages. This behavior will be extrapolated to $(x_v - t) = 0$ for write failure estimation.

3.2.3 Hold failure – p_h

If an SRAM bit gets flipped while it is being stored (i.e., cut-off from peripheral circuit), then it is called as a hold-failure. To understand this, consider the SRAM circuit during hold operation as shown in Figure 3.13. The nodes with voltages V_L and V_R store the input bit as complementary logical states. During the hold operation, the bit-line capacitors are kept at ground voltage and the access transistors are turned off [1]. [5]

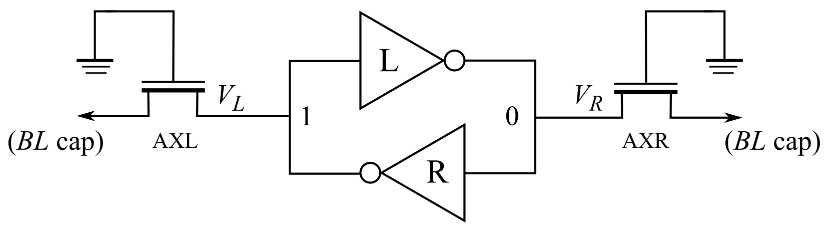


Figure 3.13: The DC bias during hold mode is illustrated in this figure. The access transistors are turned off, and the bit-line capacitors are kept at ground voltage. The nodes with voltages V_L and V_R store the SRAM bit.

[5]Negligible variation was found in SRAM cell's hold failure probability as a function of BL cap voltage.

Unlike read or write operation, the hold operation is static (except for a steady-state small leakage current, there is no dynamic activity). Therefore, the hold-failure will be explained using suitable VTC curves obtained from the hold-mode SRAM circuit. In particular, bi-stability property explains the hold operation. To explain and estimate the hold-failure probability, a static noise margin (*snm*) metric will be used. Before calculating *snm*, appropriate VTCs are needed and they will be explained next. The SRAM cell is put in DC bias conditions as during the hold operation. This is illustrated in Figure 3.14(a) and (b). Similar to read or write noise margin measurements, the inverter L and inverter R are decoupled. Two VTCs will be obtained as a result, corresponding to the two situations in Figure 3.14. The bi-stability of SRAM cell during hold operation and the calculation of *snm* from these VTCs is done by constructing a suitable butterfly curve, which is explained next [12].

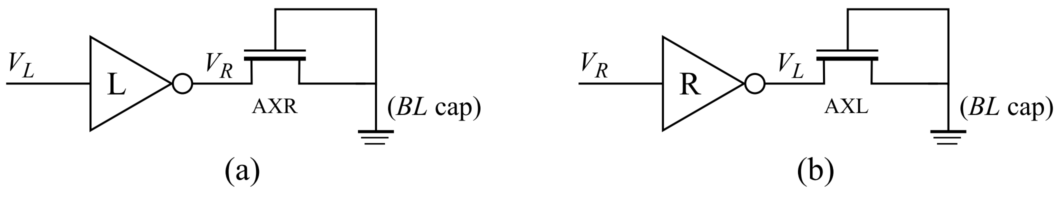


Figure 3.14: These circuits used for obtaining VTCs for *snm* calculation are illustrated. The access-transistors and BL cap are turned off. (a) The voltage V_L is swept to find the VTC of inverter L. (b) The voltage V_R is swept to find the VTC of inverter R.

As before, the VTCs of inverter L and inverter R are overlaid in a single graph as shown in Figure 3.15. The solid curve corresponds to the VTC obtained by Figure 3.14(a). Similarly, the dotted curve corresponds to the VTC obtained by Figure 3.14(b). Because of the sharp transition in VTC, these overlaid graphs result in three stability points of (V_L, V_R)

pair. Of these three points, the central point is metastable. This overlaid graph is called as the butterfly-curve. In the two wings of this butterfly diagram, maximal squares S_1 and S_2 with sides s_1 and s_2, respectively, can be inscribed. As before, s_1 and s_2 depend on the supply voltage v. The static noise margin is defined as the minimum of these sides,

$$snm = \min(s_1, s_2). \tag{3.17}$$

As supply voltage is reduced, the transition zone (from high to low) of solid VTC becomes wider. When the voltage is low enough, the overlaid diagram has no butterfly-curve structure. If $snm \geq 0$, then the butterfly-curve structure is present. With supply voltage reduction, at some critical value of v, say v_h^*, the static noise margin will vanish, or $snm(v_h^*) = 0$. This voltage v_h^* is the minimum possible voltage at which an SRAM cell can be expected to retain the bit successfully. This voltage v_h^* is called as the data-retention voltage or DRV in the literature [12]. By convention, $snm < 0$ for supply voltages below v_h^*. And hold-failure happens if $snm \equiv snm(v) \leq 0$. This single threshold test reveals if a hold-failure will be present in an SRAM cell or not.

Because of process-variations, each manufactured SRAM cell will have a different snm. As before, the process-variations affected snm will be modeled in a statistical setup using Monte Carlo simulations. Using a Monte Carlo setup, different realizations of SRAM circuit will be simulated and their snm will be obtained using the method described (see Figure 3.15). The snm will be treated as a random variable. Let $SNM(v)$ be the random variable corresponding to the snm of various cells at supply voltage v. Then the statistical event $SNM(v) \leq 0$ corresponds to a hold-failure event. And hold-failure probability will

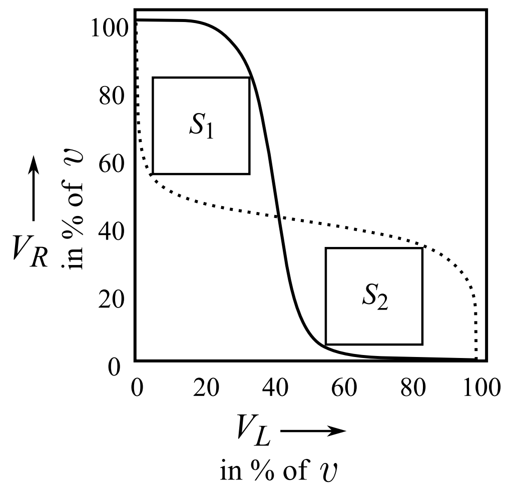


Figure 3.15: The solid curves are obtained by biasing as in Figure 3.14. Let the largest inscribed squares in the wings of butterfly diagram be S_1 and S_2 with sides s_1 and s_2 respectively. Then $snm = \min(s_1, s_2)$.

be,

$$p_h(v) = \mathbb{P}[SNM(v) \leq 0]. \tag{3.18}$$

It has been shown in literature that $SNM(v)$ exhibits a Gaussian distribution, similar to $RNM(v)$ [10,45]. This is observed in circuit simulations. For example, Figure 3.16 illustrates the empirical PDF of SNM at $v = 1.0$V with 1000 Monte Carlo trials. The advantage of this characterization is that only the mean and variance of $SNM(v)$ are needed for modeling the hold-failure probability. Thus,

$$SNM(v) \sim \mathcal{N}(\mu_h(v), \sigma_h^2(v)), \tag{3.19}$$

where, $\mu_h(v)$ and $\sigma_r(v)$ can be determined by Monte Carlo simulations. Once these functions have been calculated, the hold-failure probability for different voltages can be found out

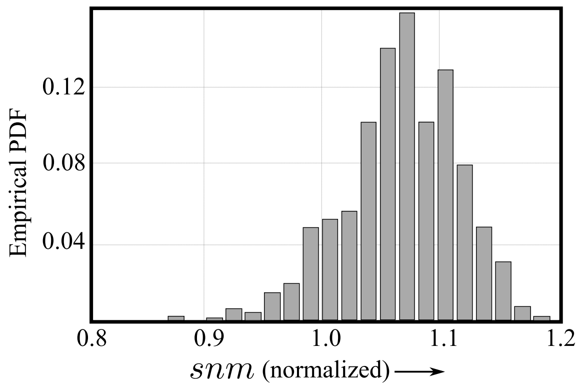


Figure 3.16: The observed empirical probability density function of SNM at $v = 1.0$V is illustrated in this figure. A Gaussian distribution is expected and observed.

using the follow Q-function formula,

$$p_h(v) \quad = \quad Q\left(\frac{\mu_h(v)}{\sigma_h(v)}\right). \tag{3.20}$$

Its derivation is identical as in the read-upset probability case. The expression in (3.20) determines $p_h(v)$ in an efficient manner. The behavior of $\mu_h(v)$ and $\sigma_h(v)$ will be revisited in the experiments section.

Two remarks on SNM and hold-failure should be noted: (i) It has been reported in the literature as well as observed during simulations that SNM is larger than RNM. As a result, $p_r(v)$ is much larger than $p_h(v)$. This point will be revisited in experiments section, and (ii) Hold-failures and soft-errors during hold-operation are dominant error mechanisms for standby-SRAM, or an SRAM which is just storing the data without any activity. This low activity scenario of SRAM is important in low-power design and it will be re-visited with experimental test-chip results in Section BLAH.

3.2.4 Write-time failure – p_{wt}

So far static errors during read, write, and hold operation were analyzed for failure probability. There are two timing-based error probabilities that must be analyzed as well. Of them, first is write-time failure. To understand it, consider the circuit for writing a bit in SRAM cell shown in Figure 3.17(a). This circuit is similar to write noise margin calculation circuit, except that access transistor's gate is biased by WL or word-line. The word-line is selected or kept at voltage v for a finite duration of t_w as shown in Figure 3.17(b). During this time, the node voltage V_R should rise from $\approx$ 0V to the trip point of inverter R for a successful write [45]. [6]

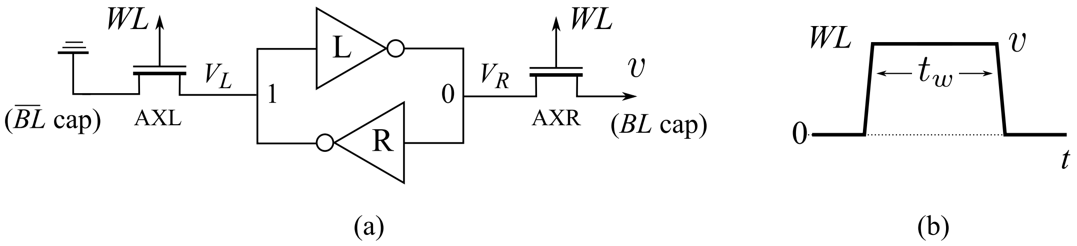


Figure 3.17: (a) This figure illustrates the write operation circuit. The BL and $\overline{BL}$ are pre-charged to complementary levels, and then the access transistors' gate are turned on. (b) The WL pulse is enabled for a time t_w to facilitate writing in any SRAM cell.

Due to process-variations, different SRAM cells will need a different WL pulse-width for a successful write. This variation will be modeled in a statistical setup as before. Let T_w be the (random) critical pulse-width needed for a successful write in any cell. If $T_w \leq t_w$, then write operation will be *successful.* Even though not explicitly mention, it must be noted that $T_w \equiv T_w(v)$ is a function of supply voltage v. And t_w (the fixed write-time) can be chosen to be a function of supply voltage. For any given t_w, the probability

[6]The node voltage V_R will initially be at a voltage slightly larger than 0V due to SRAM leakage currents.

of write-time failure is given by,

$$p_{wt}(v) = \mathbb{P}[T_w(v) > t_w]. \tag{3.21}$$

If the distribution of $T_w(v)$ is known, then the write-time failure probability in (3.21) can be computed. However, this distribution is not known in analytical form. Past works have modeled the distribution of $T_w(v)$ for a high supply voltage around $v = 1.0$V. The efficacy of this approach at all voltages is unclear [25, 45]. In this work, a different approach will be used. Since the complete distribution is not of interest, therefore, the probability of failure $p_{wt}(v)$ will be directly estimated. The usual limitation of few thousand trials for predicting tail probabilities is present in this setup as well. To overcome this limitation, extreme value theory results will be used. For any given $t, x > 0$, the residual probability function is defined as,

$$R_{wt}(t, x, v) := \mathbb{P}\left[T_w > (x + t)|T_w > x\right], \tag{3.22}$$

where the expression is interpreted as probability that T_w is larger than $(x + t)$, given that T_w is larger than x. If x is fixed, and t is large, then the tail probability can be estimated using,

$$\mathbb{P}[T_w > (x + t)] = R_{wt}(t, x, v) \cdot \mathbb{P}[T_w > x]. \tag{3.23}$$

Consider the limit function,

$$R_{wt}(t, v) = \lim_{x\to\infty} R_{wt}(t, x, v). \tag{3.24}$$

The extreme value theory tells that if $R_{wt}(t, v)$ exists, then it must be an exponential function, or

$$R_{wt}(t, v) = \exp(-\alpha_{wt}(v)t). \tag{3.25}$$

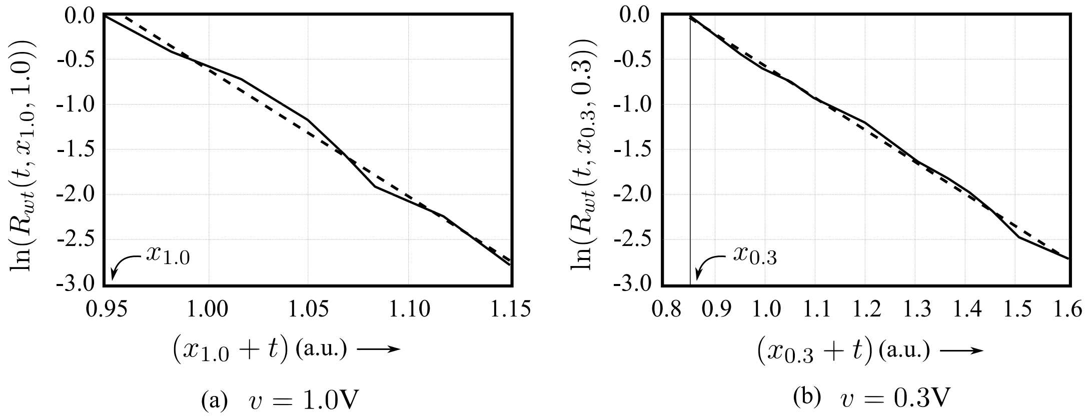


Figure 3.18: The approximate exponential decay of $R_{wt}(t, x, v)$ for large enough x is illustrated. The point x is chosen such that $\mathbb{P}[T_w > x] = 0.1$. Only plots corresponding to $v = 0.3$V and $v = 1.0$V are shown.

In this work, $\ln R_{wt}(t, x, v)$ will be examined for large enough x. *It is empirically observed that* $\ln R_{wt}(t, x, v)$ *is exponential for more than an order of magnitude.* Therefore, as a thumb rule, it is conjectured that $R_{wt}(t, x, v)$ will be exponential (using the above convergence result). It can be argued that this is nothing but an exponential fit to the residual probability function. This fact is true, but the extreme value theory suggests which is right function to look for while doing the exponential fit. The exponential fit of $R_{wt}(t, v)$ for two values of v is illustrated in Figure 3.18. Observing exponential decay in simulations, an exponential decay model for $R_w(t, x, v)$ will be adopted.

Once the exponent of $\alpha_{wt}(v)$ is estimated, the tail probability for asymptotic t_w can be predicted using,

$$\mathbb{P}[T_w > t_w] = \exp[-\alpha_{wt}(v)(t_w - x)] \cdot \mathbb{P}[T_w > x], \qquad t_w > x, \tag{3.26}$$

where $\mathbb{P}[T_w > x]$ will be estimated by Monte Carlo simulations. For this work, $x \equiv x_v$ was chosen such that $\mathbb{P}[T_w > x] = 0.1$ (see Figure 3.18). The value of t_w can be chosen such that the write-time failure probability is negligible in comparison with the write or read failures. Thus, this procedure gives an *estimate* of write speed supported by an SRAM cell. Coupled with the peripheral circuit delay, it will indicate the speed (frequency) reduction due to voltage scaling.

3.2.5 Access-time failure – p_{at}

This last discussion on parametric failures belongs to access-time failures. As mentioned before, access-time failure happens when a bit is not read successfully from an SRAM cell within a specified time. Thus, access-time failure is the read operation counterpart of write-time failure in write operation. To understand it, consider the circuit shown in Figure 3.19(a) [25]. This circuit is similar to read noise margin calculation circuit,

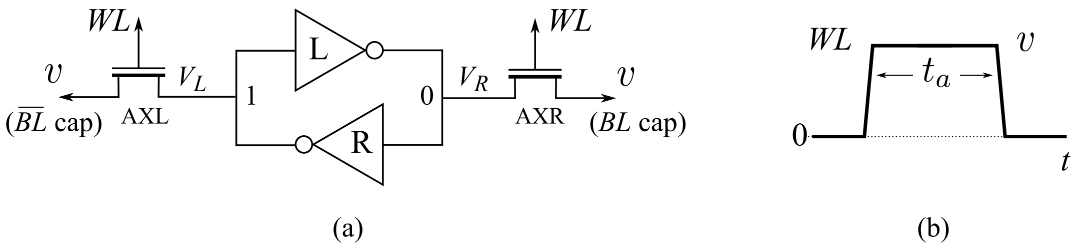


Figure 3.19: (a) This figure illustrates the read operation circuit. The BL and $\overline{BL}$ are pre-charged to supply voltage v, and then the access transistors' gates are turned on. (b) The WL pulse is enabled for a time t_a to facilitate reading from any SRAM cell. If the capacitor BL will discharge to from v to $0.9v$ in time t_a, then SRAM read is successful.

except that access transistors' gates are biased by WL or word-line voltage. The word-line is kept at supply voltage v for a finite duration of t_a as shown in Figure 3.19(b). In Figure 3.19(a), when WL is high, the access transistor AXR discharges the BL cap voltage. If BL cap voltage falls below $0.9v$, the read operation is successful. Note that $\overline{BL}$ cap voltage will be approximately constant if t_a is not very large. In case if BL cap voltage falls below $0.9v$, then $\Delta BL := |BL - \overline{BL}| > 0.1v$. The peripheral circuit (sense amplifiers) amplify the voltage from this difference to read the bit successfully.

Due to process-variations, different SRAM cells will need a different WL pulse-width for a successful read. This variation will be modeled in a statistical setup as before. Let T_a be the (random) critical pulse-width needed for a successful write in any cell. If $T_a \leq t_a$, then the read operation will be *successful* from timing considerations. As before, it must be noted that $T_a \equiv T_a(v)$ is a function of supply voltage v. And t_a (the fixed access-time) can be chosen to be a function of supply voltage also. For any given t_a, the probability of access-time failure in the random setup is given by,

$$p_{at}(v) = \mathbb{P}[T_a(v) > t_a]. \tag{3.27}$$

In the absence of access-time's probability distribution, an extreme-value fit will be verified and obtained as for the case of write-time failures (see Section 3.2.4). Let $R_{at}(t, x, v)$ be the residual probability function. Using simulations it is observed that $R_{at}(t, x, v)$ is decaying exponentially with t for a region. Using extreme-value theory, it is then conjectured that $R_{at}(t, v) := \lim_{x\to\infty} R_{at}(t, x, v)$ is exponential. Thus, the access failure probability is given by,

$$\mathbb{P}[T_a > (x + t)] = \exp(-\alpha_{at}(v)t) \cdot \mathbb{P}[T_a > x], \tag{3.28}$$

where the point x is chosen such that $\mathbb{P}[T_a > x] = 0.1$. The exponential decay of $R_{at}(t, x, v)$ is shown in Figure 3.20.

With this note, the access-time failure probability estimation is complete. It must be noted that t can be made large enough to ensure that access-failures are negligible in comparison to the sum of read, write, and hold failures.

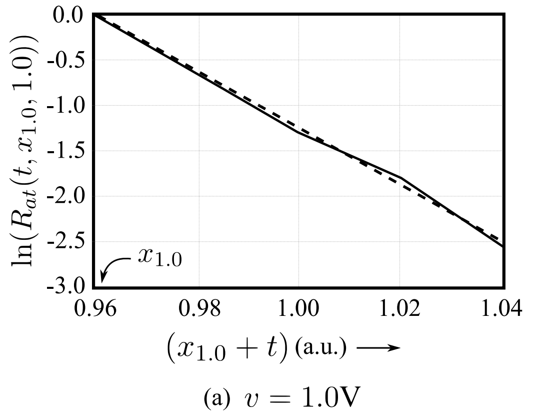


Figure 3.20: The approximate exponential decay of $R_{at}(t, x, v)$ for large enough x is illustrated. The point x is chosen such that $\mathbb{P}[T_a > x] = 0.1$. Only plots corresponding to $v = 0.3$V and $v = 1.0$V are shown.

3.3 Supply noise

Supply-voltage noise remains an important issue to be addressed. Usually a 100mV extra margin is added to the supply voltage to ensure proper functionality of SRAM. In this work, the same approach will be adopted. Thus, if v^* is found to be the operating power optimal supply voltage for SRAM, then $(v^* + 100\text{mV})$ will be the actual supply voltage. For simplicity, power comparisons will be made without adding the noise margin. *It must be noted that this is not the power optimal strategy.* For example, in the case of standby SRAM, the supply noise will be much smaller due to zero circuit activity. Usually supply noise is observed at the clock edge, when the active logic blocks draw a large (but indefinite) amount of current [19].

It is tempting to model supply noise in a stochastic noise setup. However, this will require considerable rethinking of existing figures of merit as well as noise modeling. For example, rnm, snm, wnm, T_w, and T_a parameters are defined for a fixed supply voltage.

While T_w and T_a are timing parameters which can still be studied under the presence of supply noise, the noise-margins obtained from VTCs are DC concepts and extending them will require non-trivial research effort. Recently, some papers have addressed the dynamic stability of SRAM, but the progress is far from complete [47]. Apart from non-trivial extensions of stability criteria, supply noise modeling will require a considerable effort. Some experiments have studied the noise distribution or correlation in practical chips, and they clearly illustrate the time-varying nature of the noise distribution [19]. The time-varying nature of noise will complicate the problem further. Accordingly, this challenging problem is left as a future work.

3.4 Review

This section focused on the modeling of important SRAM failures as a function of supply voltage. An interesting feature of error probability macro-models is its simplicity. Though considered in this work, these simple metrics can be used for designing SRAM cells for better resilience against failures. Failure probability estimates derived from these macro-models will be used to predict the leakage-power reduction in the presence of error-correction and refresh. If these macro-models are not accurate enough for any task at hand, then the optimizer can be fed with more accurate numbers.

Chapter 4

Simulation results on modeling and optimization

In this chapter, simulations results from the 90nm and 65nm technologies will be presented. The simulation models developed in Chapter 3 applies well to both these technologies. Most of the plots are normalized to comply with the non-disclosure agreement signed for obtaining access to these technology files. Any inconvenience due to this normalization is regretted.

4.1 Average SRAM leakage current

To tackle the increasing leakage-power problem, the 65nm technology has higher threshold CMOS transistors. Higher threshold transistors have lower leakage but it also reduces the saturation current (and hence speed) of the transistor. This should be expected as a temporary glitch in the leakage-power increase with technology. The net result is a smaller average (per cell) leakage in the 65nm technology SRAM cells. The average leakage current in SRAM cells are compared in Figure 4.1. The decay in leakage current is approximately quadratic in the supply voltage v. Both the plots are shown on a normalized y-axis. Observe that the leakage in 65nm technology is smaller by an order of magnitude than the leakage in 90nm technology. Due to this reason, the data-lifetimes used will be different for the two technologies. A data-lifetime of $t_0 = 1$s for the 90nm technology and $t_0 = 10$s for the 65nm technology will be used.

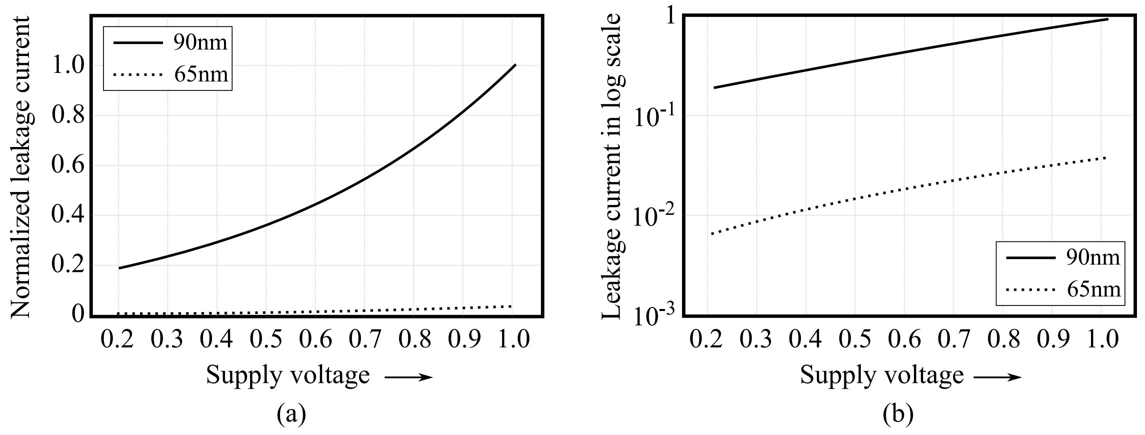


Figure 4.1: (a) Average (over process-variations) SRAM cell leakage current is plotted in this figure. Observe that due to high threshold voltage, 65nm CMOS technology has lower leakage. (b) Leakage current comparison is illustrated in log scale. Observe that in 65nm technology, the leakage is smaller by an order of magnitude.

For a better understanding of leakage power, these leakage currents are in the range of 1nA. Thus in a time of 1s, they will approximately leak 1nJ of energy. Active read and write energy in the SRAM cell is of the order of 10pJ. Thus energy spent due to leakage/cell in a second is about 10× to 100× larger than one read, write, and refresh operation. Thus, leakage-power contribution is significant only when the read-write activity is occasional. A data-lifetime $t_0 \geq 1$sec (in 90nm) or $t_0 \geq 10$sec (in 65nm) is coherent with the assumption that the leakage-power is significant.

4.2 Soft-error rates

A critical charge based approach was used to model soft-error rate, as discussed in Chapter 3. Recall that soft-error rate was given by the following expression,

$$\bar{r}_s(v) = \mathbb{E}[R_s(v)] = K\mathbb{E}[\exp(-\alpha_s Q_c(v))].$$

The distributions of $Q_c(v)$ for different supply-voltages were obtained for the two technologies under study. The constants K and α_s were estimated from the paper by Hazucha and Svennsson [23]. The exponent α_s was estimated as 0.0769/fC and 0.1031/fC for the 90nm and the 65nm technologies. The constant K was estimated using the FIT rate at sea-level reported in literature [23]. The sea level FIT rate was used to get the soft-error rate of a single cell. The increase in soft-error rate with decrease in supply voltage is illustrated in Figure 4.2.

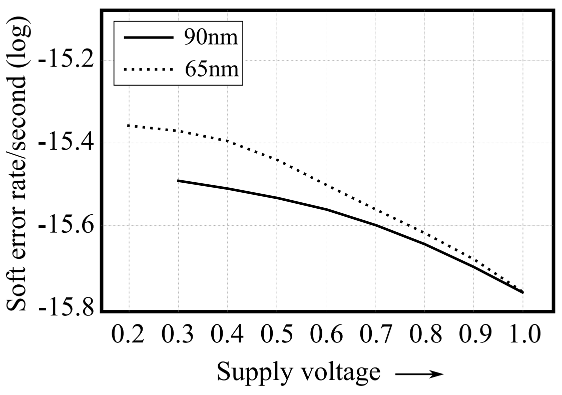


Figure 4.2: The soft-error rate (per second) as a function of supply voltage is compared. The 65nm technology is expected to have a larger increase in soft-error rate with supply voltage reduction than the 90nm technology.

Observe that the soft-error rate increases more for the 65nm technology with voltage scaling. The prime reason is a larger α_s for the 65nm technology. The $v = 1.0$V soft-error rates are plotted as equal since the gate area of SRAM cell transistors used in simulations are equal. It would be expected that 65nm technology transistors have a smaller gate area due to technology scaling. However, in the available technology kit, the minimum W/L-ratio that was allowed is two. Therefore, the gate area remains the same even after scaling. This should keep the cross-section (or flux) of soft-errors equal in the two cases [23]. Even if $\bar{r}_s(1.0)$ are not identical for the two technologies, the relative ratio $\bar{r}_s(1.0)/\bar{r}_s(0.3)$ is expected to be higher for the 65nm technology. This will necessitate a smaller refresh times t_r in the 65nm technology as will be seen shortly.

Table 4.1: Soft-error rate (per second) as a function of supply voltage

v	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\log_{10}(r_s(v))$ in 90nm	n/a	-15.60	-15.62	-15.64	-15.67	-15.71	-15.76	-15.81	-15.88
$\log_{10}(r_s(v))$ in 65nm	-15.49	-15.51	-15.53	-15.57	-15.63	-15.69	-15.75	-15.81	-15.88

4.3 Parametric failures – read upset probability

The estimation results of read-upset probability will be presented in this section. Recall that a read-upset event was characterized by the read noise margin being negative. And the read noise margin was estimated as a Gaussian random variable (see Section 3.2

of Chapter 3), i.e.,

$$RNM(v) \sim \mathcal{N}(\mu_r(v), \sigma_r^2(v)).$$

Thus, to obtain the read-upset probability, only $\mu_r(v)$ and $\sigma_r(v)$ need to be estimated. Estimation of $\mu_r(v)$ is a fairly simple procedure, for example, see Bickel and Doksum [68]. The (normalized) estimates are plotted in Figure 4.3 for the 90nm technology. Observe that the mean $\mu_r(v)$ decreases approximately in a linear fashion with supply voltage reduction, and the standard deviation $\sigma_r(v)$ stays approximate constant. This behavior is also witnessed in 65nm technology simulations as illustrated in Figure 4.4. Using the expression,

$$p_r(v) = \mathcal{Q}\left(\frac{\mu_r(v)}{\sigma_r(v)}\right),$$

the read-upset probability can be evaluated. The $\mathcal{Q}$-function is the standard Gaussian tail probability function.

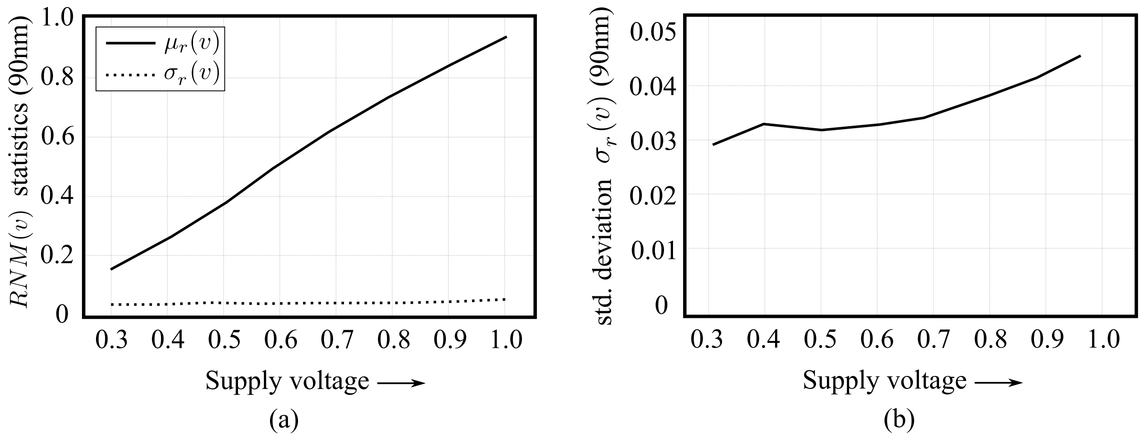


Figure 4.3: (a) The normalized mean and standard deviation of $RNM(v)$ are plotted for the 90nm technology. (b) The approximately constant behavior of $\sigma_r(v)$ (especially for lower voltages) is illustrated in this plot.

The resultant read-upset probability for the two technologies are given in Table 4.2. The empty dotted values mean that those probabilities are insignificant.

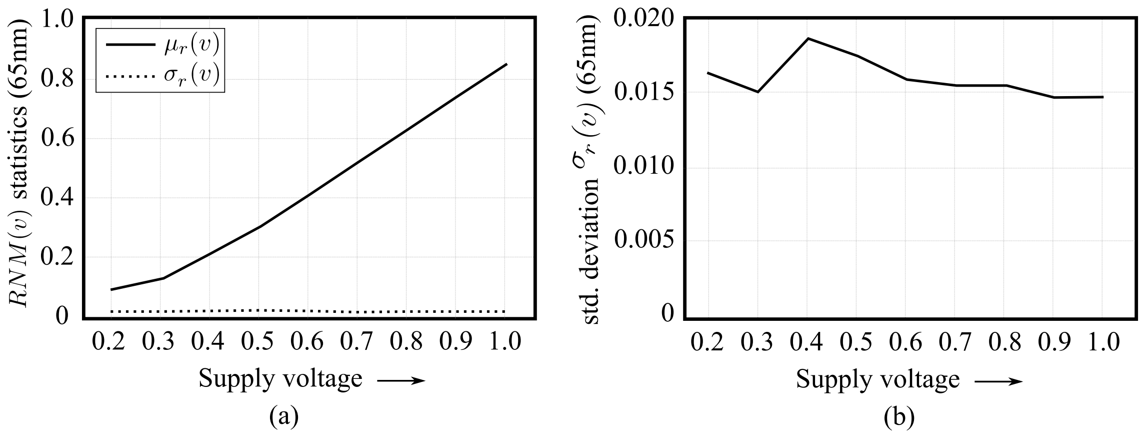


Figure 4.4: (a) The normalized mean and standard deviation of $RNM(v)$ are plotted for the 65nm technology. (b) The approximately constant behavior of $\sigma_r(v)$ is illustrated in this plot.

Table 4.2: Read-upset probability as a function of supply voltage

v	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\log_{10}(p_r(v))$ in 90nm	n/a	-7.0	-12.3	-27.3	…	…	…	…	…
$\log_{10}(p_r(v))$ in 65nm	-7.4	-16.7	-28.0	…	…	…	…	…	…

4.4 Parametric failures – write failure probability

Calculation of write-failure probability was highlighted in Section 3.2 of Chapter 3. It was noted that at high voltages the distribution is approximately Gaussian and at low voltages, a residue function based fitting approach can be used to obtain the write-failure probability. This procedure was used to obtain the write-failure probability estimates for the 90nm and the 65nm technologies.

The resultant write-failure probability for the two technologies are given in Table 4.3. The empty dotted values mean that those probabilities are insignificant. Observe that write-failure probability dominates the read-failure probability. This observation is in

Table 4.3: Write-failure probability as a function of supply voltage

v	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
$\log_{10}(p_w(v))$ in 90nm	n/a	-5.72	-6.20	-6.56	-11.09	-21.9	...	...	...
$\log_{10}(p_w(v))$ in 65nm	-4.94	-5.59	-6.54	-7.80	-9.99	...	...	...	...

consonance with other results reported in the literature. For example, Calhoun and Chandrakasan noted that write-failures first occur at a supply $v = 0.6$V for a nominal SRAM cell in the 90nm technology [10]. Similarly, Bhavnagarwala et al. noted that write-failures are the most critical among parametric failures [17].

4.5 Parametric failures – hold failure probability

Recall that static noise margin or SNM is used to understand a hold failure. In Section 3.2 of Chapter 3, it was noted that $SNM(v)$ is approximately Gaussian, i.e.,

$$SNM(v) \sim \mathcal{N}(\mu_h(v), \sigma_h^2(v)).$$

The estimates for $\mu_h(v)$ and σ_h^2 are plotted in Figure 4.5 for the 90nm and the 65nm technologies.

During hold or storage, SRAM cell is practically isolated from the rest of the chip. Owing to this reason, hold failures are expected to be negligible, compared to other parametric failures. For example, during read-operation, the cell is expected to store the bit as well as interact with the peripheral read circuitry. This intuitive fact is also observed during simulations. As expected, hold-failure calculation using $p_h(v) = \mathbb{P}[SNM(v) \leq 0]$ or $\mathcal{Q}\left(\frac{\mu_h(v)}{\sigma_h(v)}\right)$ gives negligible hold-failure probability compared to read-upset probability. The table for hold-failure values is omitted for simplicity.

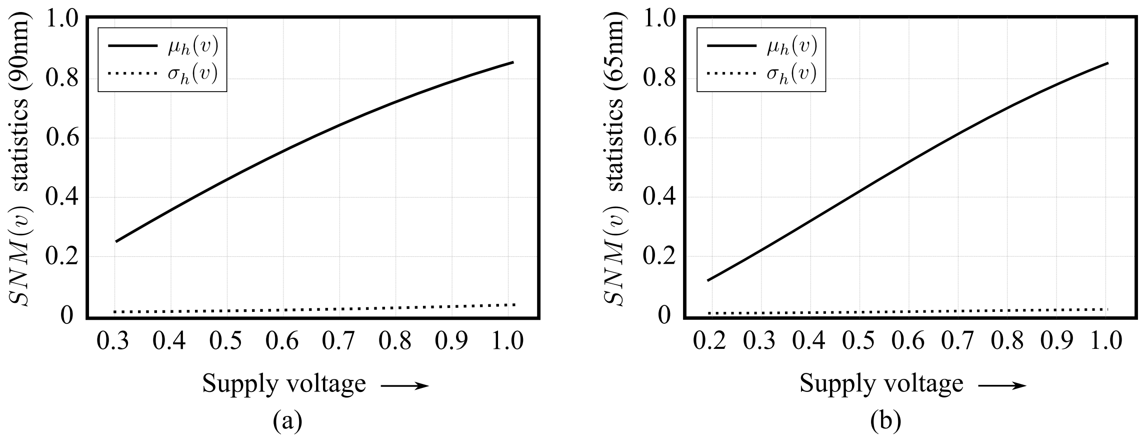


Figure 4.5: (a) The normalized mean and standard deviation of $SNM(v)$ are plotted for the 90nm technology. (b) The normalized mean and standard deviation of $SNM(v)$ are plotted for the 65nm technology.

4.6 Leakage-power optimization results

Using the previous SRAM cell error probability modeling techniques, the obtained probability values from Table 4.1, Table 4.2, and Table 4.3 are plotted in Figure 4.6. These will be input to the optimization framework. Hold-failure probability is negligible compared to the read-upset probability and is not shown.

For error-probability data as shown in Figure 4.6, *power per bit* optimization results will be presented. To understand the advantage of data-refresh, *power per bit* cost function $\mathcal{P}_b(v, t_r, ECC)$ is plotted against v when the error-correction code is restricted to $[31, 26, 3]$ Hamming code. The target decoding error probability is set by the $[31, 26, 3]$ Hamming code and soft-errors at $v = 1.0$V). The target decoding error probability can be computed to be,

$$p_{target} = \binom{n}{2} (t_0 r_s(1.0))^2 = 1.40 \times 10^{-25} (t_0)^2. \tag{4.1}$$

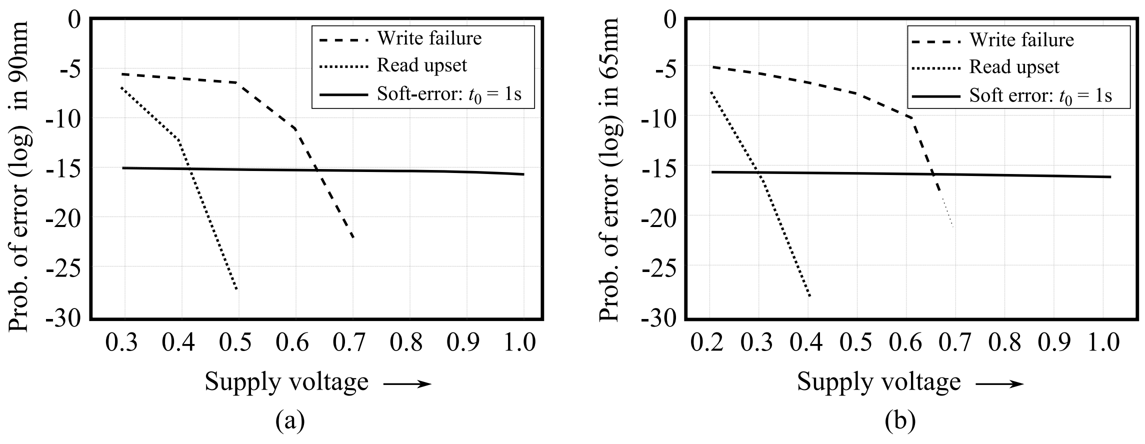


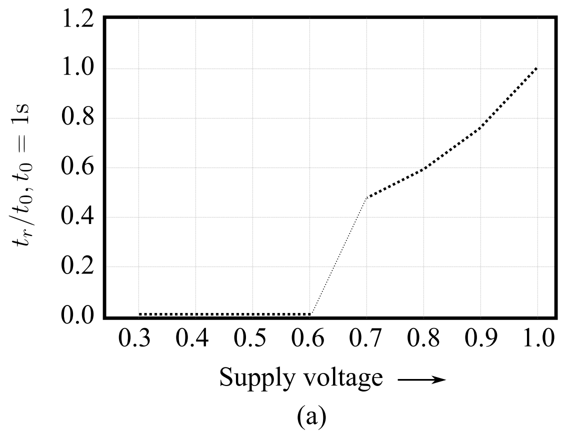
Figure 4.6: Obtained estimates for soft-errors and parametric failures, for the 90nm and the 65nm technologies, are compared in this semilog plot. At low voltages, parametric failures are significant. At high-voltages, dominant error-probability mechanism is soft-error.

The refresh time t_r has to be chosen to meet the target error probability p_{target}. With refresh, the bit-error probability in each refresh cycle is $p_e(v) = (t_r r_s(v))$. In the absence of parametric failures, probability of error in each refresh cycle is $\binom{n}{2}(t_r r_s(v))^2$. There are approximately t_0/t_r refresh cycles and this error can happen in any cycle. Thus, the error probability is approximately given by,

$$p_{error}(v, t_r) \approx \frac{t_0}{t_r} \cdot \binom{n}{2}(t_r r_s(v))^2 = \binom{n}{2} t_0 t_r (r_s(v))^2. \tag{4.2}$$

Comparing (4.1) and (4.2), t_r/t_0 should scale as $(r_s(v)/r_s(1.0))^2$ to maintain a constant decoding error probability. For $v \leq 0.6$V, where parametric failures are dominant, these approximations break down. And the probability constraint cannot be met by data-refresh. In this scenario of dominant parametric failures, t_r is set to zero, which makes $\mathcal{P}_b(v, t_r, ECC)$ infinite (see (2.14) and Figure 4.6). Since parametric failures are spatially fixed, therefore data-refresh will not combat its effect on decoding error probability. Using data-refresh, the *power per bit* cost function can be reduced by nearly 60% for the 90nm technology.

The graphs of the refresh-time t_r and $\mathcal{P}_b(v, t_r, ECC)$ for this technology are plotted in Figure 4.7.



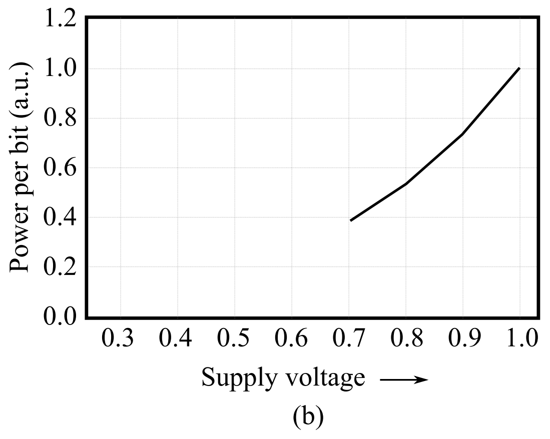
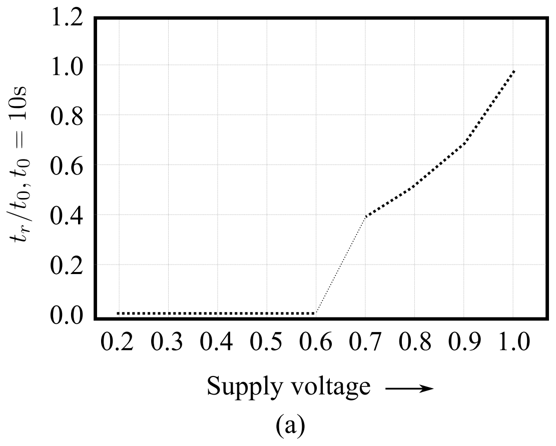


Figure 4.7: (a) For $[31, 26, 3]$ Hamming code as the error-correction code, The data-refresh rate is plotted for the 90nm technology. (b) The *power per bit* cost function is plotted against the supply voltage v. The voltage reduction is limited by parametric failures which start at 0.6V. The *power per bit* reduction is nearly 60%.

Similar plots for the 65nm technology are illustrated in Figure 4.8.



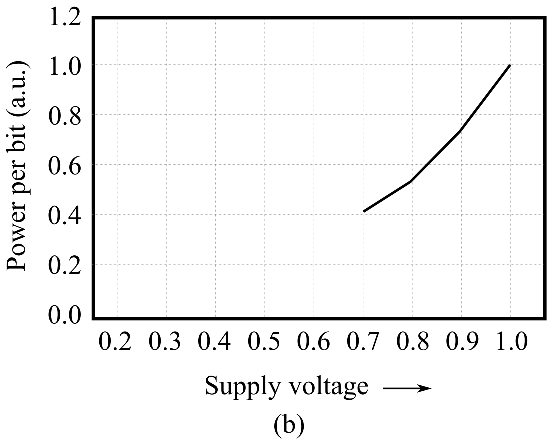


Figure 4.8: (a) For $[31, 26, 3]$ Hamming code as the error-correction code, The data-refresh rate is plotted for the 90nm technology. (b) The *power per bit* cost function is plotted against the supply voltage v. The voltage reduction is limited by parametric failures which start at 0.6V. The *power per bit* reduction is nearly 60%.

Table 4.4: Error-correction codes used for optimization

Minimum distance	Code family	(n, k) pairs
$d = 3$	Hamming	$(31, 26), (63, 57), (127, 120), (255, 247), (511, 502)$
$d = 5$	BCH	$(63, 51), (127, 113), (255, 239), (511, 493)$
$d = 7$	BCH	$(63, 45), (127, 106), (255, 231), (511, 484)$
$d = 9$	BCH	$(63, 39), (127, 99), (255, 223), (511, 475), (1023, 983)$
$d = 11$	BCH	$(63, 36), (127, 92), (255, 215), (511, 466), (1023, 973)$
$d = 13$	BCH	$(63, 30), (127, 85), (255, 207), (511, 457)$
$d = 15$	BCH	$(255, 199), (511, 448), (1023, 953)$
$d = 17$	BCH	$(255, 191), (511, 439), (1023, 943)$

With voltage-scaling, the estimated soft-error rate increase is higher in the 65nm technology. Therefore, the refresh time t_r will be smaller for the 65nm technology. This is observed in the simulation results. Coincidentally, the refresh-based voltage scaling for both the technologies stops at $v = 0.6$V, and they both have nearly equal *power per bit* reduction (in percentage).

When error-correction code choice includes more families (e.g. BCH codes and Hamming codes as in Table 4.4 obtained from [51]), the following optimization procedure is used. As before, p_{target} is as in (4.1). Recall that if errors and erasures (parametric failures) are distinguished, the procedure is classified as generalized decoding. If errors and erasures are combined, it is called as specialized decoding. The decoding failure events using the minimum Hamming distance of error-correction code for the two cases were given by (2.1) and (2.2), respectively. For each error-correction code with parameters $[n, k, d]$, and for each v, a refresh time t_r is calculated such that the decoding error-probability constraint is achieved. If the probability constraint cannot be met by $t_r = 0$ due to parametric failures, then t_r is set to zero, which makes $\mathcal{P}_b(v, t_r, ECC)$ infinite (or suboptimal). Once data-refresh times have been computed, $\mathcal{P}_b(v, t_r, ECC)$ function is optimized over the choice of v. This will result in optimized *power per bit* for every error-correction code. Finally, $\mathcal{P}_b(v, t_r, ECC)$ can be optimized over ECC with same minimum distance d. This minimum distance can be thought of as the complexity of decoding. [1] As a result, an optimum *power per bit* will be obtained for each d.

Power reduction will be measured against the per-cell leakage at $v = 1.0$V for the $[31, 26, 3]$ code. For the 90nm technology, the average leakage per cell at $v = 0.3$V sets an upper bound of 94% on *power per bit* reduction. Similarly for the 65nm technology, the average leakage per cell at $v = 0.2$V sets an upper bound of 97% on *power per bit* reduction. The result of previously discussed optimization procedure for generalized and specialized decoding is plotted in Figure 4.9 as a function of $\left\lfloor \frac{d-1}{2} \right\rfloor$, the number of errors that can corrected. With increasing d, the *power per bit* reduction gets closer to the upper bound. And, generalized decoding approaches the lower bound at a faster rate.

Some insights into the minimum d needed to achieve near optimal leakage-power reduction will be provided next. Consider any code with parameters $[n, k, d]$. At low voltages, where parametric failures are dominant, the decoding error probability under generalized decoding will be approximately,

$$p_{gen-decoding} \approx \binom{n}{d} (p_e(v))^d. \tag{4.3}$$

[1]For example, in BCH codes, the error location search unit's complexity is proportional to the number of errors that can be corrected [51, Chapter 6].

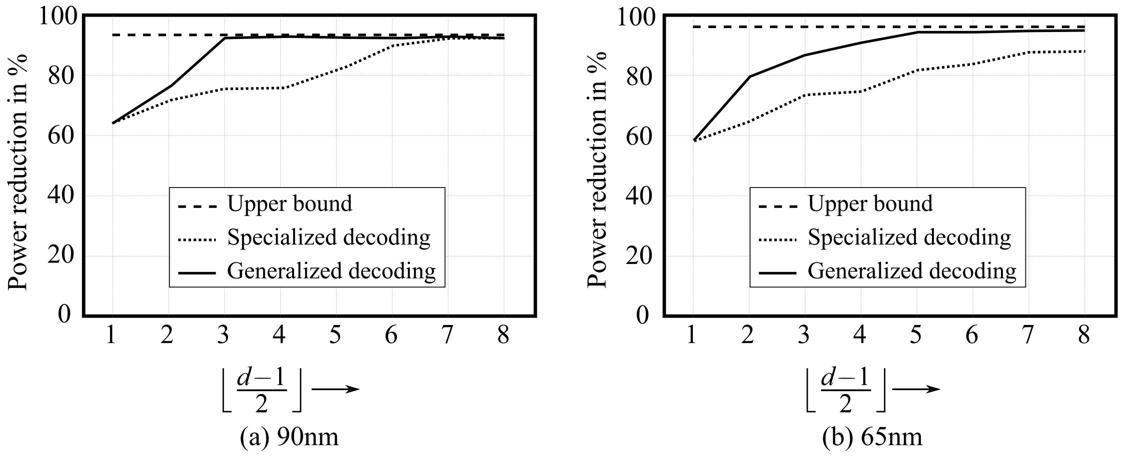


Figure 4.9: The upper bound is obtained by comparing leakage-power per cell at minimum supply voltage and leakage-power per cell at $v = 1.0$V. *Power per bit* reduction gets close to this upper bound with increase in minimum distance d of error-correction code. Generalized decoding based power reduction approaches the upper bound at a faster rate.

Consider the 90nm technology. Then, by using Table 4.3, $p_x(0.3) = 10^{-5.72}$. Recall that the target error probability is 1.40×10^{-25} for a data-lifetime $t_0 = 1$sec. The term $\binom{n}{d}$ is always larger than 1. Thus, for $d \geq 5$, one can expect the $(p_e(0.3))^d$ to be smaller than the target error probability. Further, for any reasonable value of n, $\binom{n}{5}$ will be much larger than 1. This makes minimum d slightly larger than 5 – in particular 7 – such that target error probability can be achieved with small value of (n/k) ratio. Similarly, for specialized decoding the decoding error probability is approximately given by,

$$p_{sp-decoding} \approx \binom{n}{u+1} (p_e(v))^{u+1}, \tag{4.4}$$

where $u = \lfloor (d-1)/2 \rfloor$. Going by previous calculations, the minimum value of u required will be $u = 6$. This is precisely what is observed in Figure 4.9(a). Similar approximate calculations can be used to understand the 65nm leakage-power optimization results.

Coding latency: Coding introduces delay and parity overhead. Since $p_e(v)$ and $p_x(v)$ are close to zero, therefore parity overhead can be made negligible. For decoding delays, note that codes with $n \leq 1024$ were used in the optimization. If $n \leq 1024$, and $p_x(v) \leq 10^{-5}$, then probability of no cell in error is approximately $(1 - np_x(v)) \geq 0.99$. Thus, more than 99% decoding cases require only parity check (small delay). Note that this will result in a variable delay.

4.7 Effect of voltage-scaling on SRAM cell speed

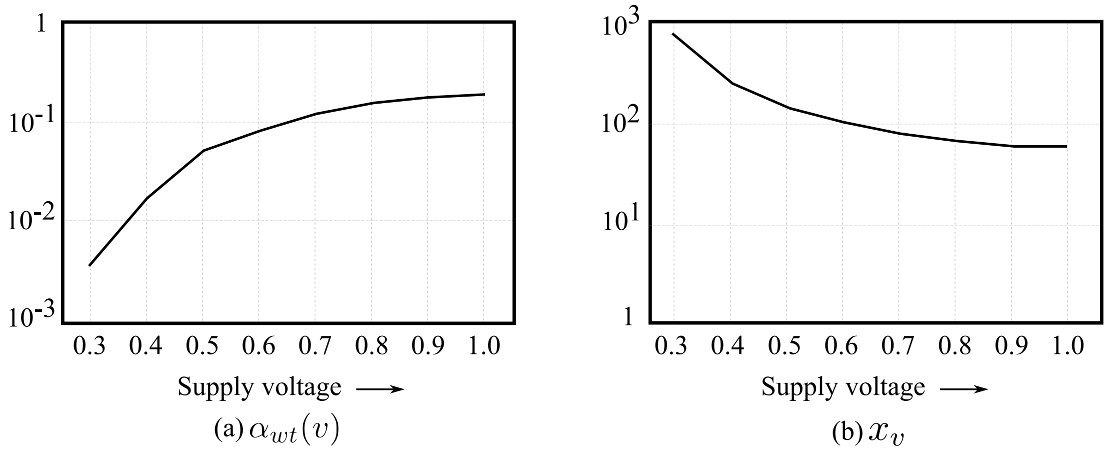


Figure 4.10: These estimates were obtained using Monte Carlo simulations in the 90nm technology. (a) The estimated exponent $\alpha_{wt}(v)$ of the write-time model in (4.5) is plotted. (b) The estimate of the reference $x_v : \mathbb{P}[T_w > x_v] = 0.1$ is plotted.

As discussed in Chapter 3, there are two timing based parametric failures: (i) write-time failure, and (ii) access-time failure. To tackle these failures, the write-time and the access-time has to be made “large enough” such that their probability is negligible. In this section, quantities like “large enough” and ”negligible” will be estimated. The results estimate the effect of voltage-scaling on SRAM cell speed, i.e., how fast the data can be written in or read from an SRAM cell. For write-time T_w, a model for the distribution of tail-probability was estimated in Chapter 3 (see (3.26)),

$$\mathbb{P}[T_w(v) > t_w] = \exp(-\alpha_{wt}(v)(t_w - x_v)) \cdot \mathbb{P}[T_w(v) > x_v], \qquad t_w > x_v. \tag{4.5}$$

The parameters $\alpha_{wt}(v)$ and the reference time x_v are indicators of SRAM cell's write-speed. These indicators are plotted in Figure 4.10 for the 90nm technology. Observe that according to these estimates, *the exponent and hence the write-time will increase by two order of magnitudes* due to supply-voltage reduction from $v = 1.0$V to $v = 0.3$V.

Using a similar procedure, the write-time exponent $\alpha_{wt}(v)$ and the reference x_v can be obtained for the 65nm technology as shown in Figure 4.11. *The overall loss in*

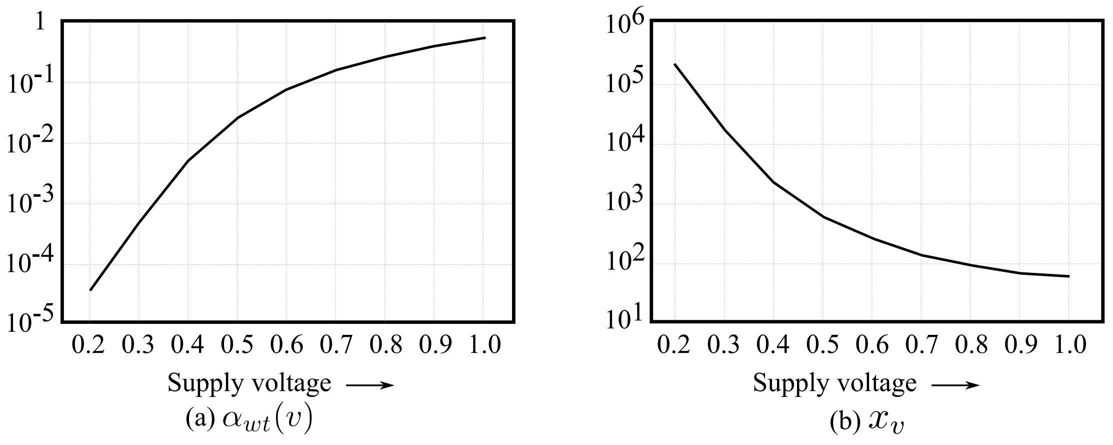


Figure 4.11: These estimates were obtained using Monte Carlo simulations in the 65nm technology. (a) The estimated exponent $\alpha_{wt}(v)$ of the write-time model in (4.5) is plotted. (b) The estimate of the reference $x_v : \mathbb{P}[T_w > x_v] = 0.1$ is plotted.

speed is larger for the 65*nm technology.* The surmised reason is a larger transistor threshold voltage used in 65nm technology to reduce leakage current. As a result, the transistors enter subthreshold region earlier and the speed deteriorates till a supply voltage of $v = 0.2$V.

For the access-time T_a, a similar model for the distribution of tail-probability was estimated in Chapter 3 (see (3.28)),

$$\mathbb{P}[T_a(v) > t_a] = \exp(-\alpha_{at}(v)(t_a - x_v)) \cdot \mathbb{P}[T_a(v) > x_v], \qquad t_a > x_v. \tag{4.6}$$

The parameters $\alpha_{at}(v)$ and the reference time x_v are indicators of SRAM cell's access or read-speed. These indicators are plotted in Figure 4.12 for the 90nm technology. Observe that according to these estimates, the exponent and hence *the access-time will increase by two order of magnitudes due to supply-voltage reduction from* $v = 1.0\,V$ *to* $v = 0.3\,V$.

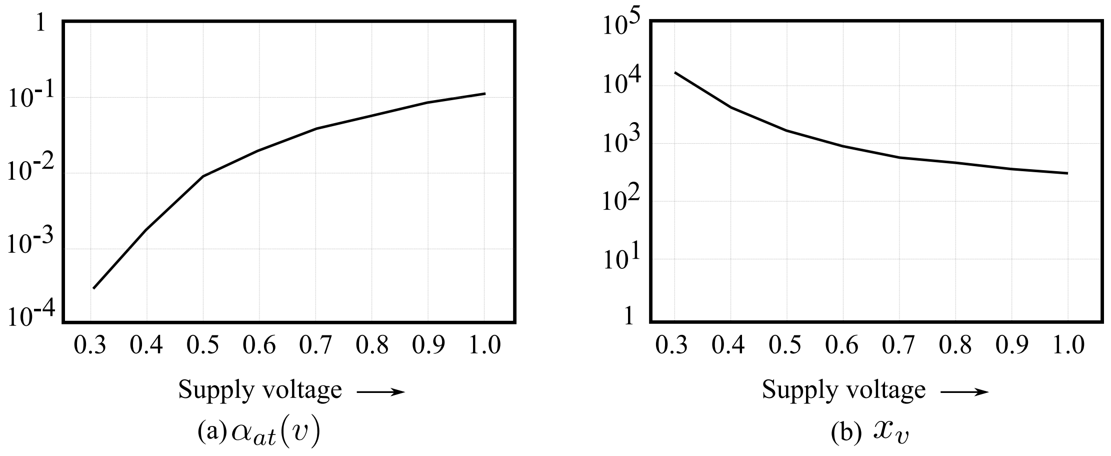


Figure 4.12: These estimates were obtained using Monte Carlo simulations in the 90nm technology. (a) The estimated exponent $\alpha_{at}(v)$ of the access-time model in (4.6) is plotted. (b) The estimate of the reference $x_v : \mathbb{P}[T_w > x_v] = 0.1$ is plotted.

Using a similar procedure, the access-time exponent $\alpha_{at}(v)$ and the reference x_v can be obtained for the 65nm technology as shown in Figure 4.13. Similar to the write-time case, the overall loss in speed is larger for the 65nm technology. The expected reason is a larger transistor threshold voltage used in 65nm technology to reduce leakage current.

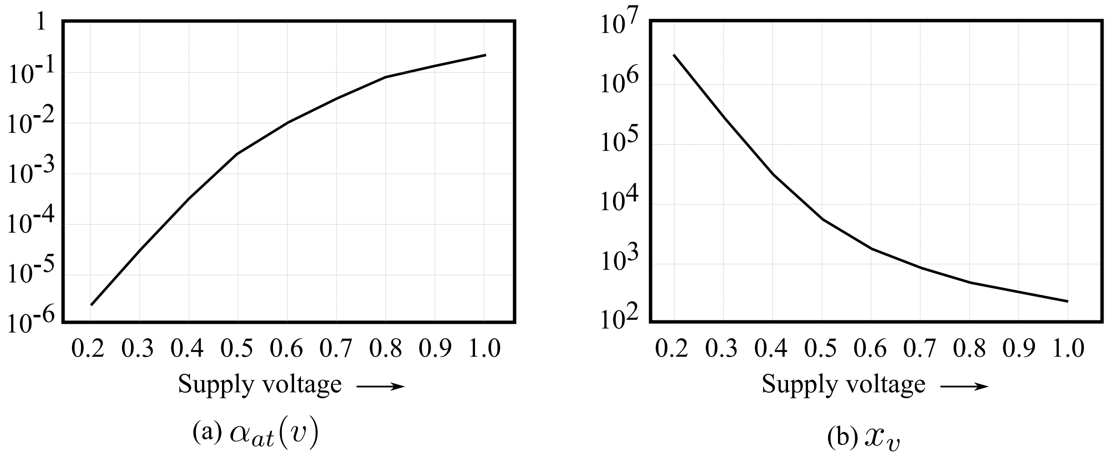


Figure 4.13: These estimates were obtained using Monte Carlo simulations in the 65nm technology. (a) The estimated exponent $\alpha_{at}(v)$ of the access-time model in (4.6) is plotted. (b) The estimate of the reference $x_v : \mathbb{P}[T_w > x_v] = 0.1$ is plotted.

4.8 Review

The average leakage current for the 65nm technology is smaller, compared to the 90nm technology. At high supply voltages, decoding error probability is dominated by soft-errors. At low supply voltages, parametric failures dominate the decoding error probability. For long enough data-lifetime t_0, periodic data-refresh can be used to reduce the leakage power by approximately 60%, without affecting the decoding error probability. Codes which correct more than single-bit errors achieve better leakage power reduction (at the cost of complexity). Treating fixed faults (parametric failures) as erasures achieves possible leakage-power reduction with lower complexity error-correction codes. Finally, effect of supply voltage reduction on SRAM cell's access and write speed was estimated using extreme-value theory. Speed reduction by two to four orders of magnitude is expected using circuit simulations.

Chapter 5

Standby SRAM

An SRAM which is primarily in "no-operation" ON mode is classified as a standby SRAM. In many chips with SRAM module, it is assumed that there are two modes: (i) the *active-mode* with high supply voltage in which the SRAM is active for reading and writing, and (ii) the *standby-mode* with a lower supply voltage in which the task of SRAM is only to retains the data. In the standby-mode, the target is reliable data retention at minimum possible leakage-power. As previously discussed, an effective method to reduce leakage-power is to minimize the supply voltage while ensuring data-retention.

As noted in the Chapter 1, this leakage-power reduction comes at a cost of increased failure rate. In the standby-mode, the parametric failures corresponding to read and write operation don't contribute. The hold-failures contribute and they are characterized using the DRV. Supply noise is either absent due to no activity in circuits (except standby leakage), or it can be tackled using the 100mV noise margin technique. The hold failures happen at extremely low voltages, and it is noted that in the proximity of DRV

voltages, the erratic fluctuations are not expected (see Section 5.1). The increase in soft-error rate $r_s(v)/r_s(V_{dd})$ is finite and it can be tackled using scrubbing at a rate dependent on $t_0, r_s(v)/r_s(V_{dd})$, and the error-correction scheme employed. Using these simplifications in error-mechanisms, *the focus in this chapter will only be on hold-failures* (retention failures) and results will be derived for fundamental minimum leakage-power per stored bit in SRAM cells. This minimum is fundamental from a system design perspective where coding is used.

Using the voltage scaling approach, it has been shown that any SRAM cell has a critical voltage (called the data retention voltage or DRV) at which a stored bit (0 or 1) is retained reliably [12]. The intra-chip DRV exhibits a distribution due to process-variations.

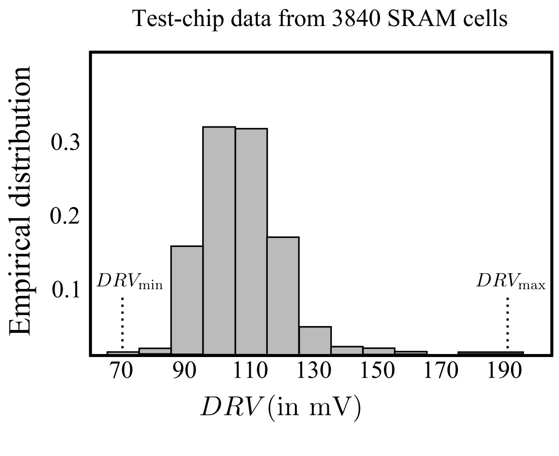


Figure 5.1: The experimental intra-chip DRV varies from 70 to 190mV in the 90nm CMOS technology. The worst-case solution for data-retention is a supply voltage of 200mV.

In Figure 5.1, a test-chip DRV distribution is illustrated which was obtained using experiments [13]. This test chip was fabricated in the 90nm technology based on an industry IP module (courtesy: ST Microelectronics). The depicted DRV histogram was measured across different SRAM cells on the same chip. Any such DRV histogram will be called as intra-chip DRV distribution from now on. For this test-chip, the DRV varies from 70mV to 190mV for 3840 SRAM cells. In order to minimize leakage-power without observing hold-failures, a standby supply voltage equal to the highest DRV among all cells in an SRAM can be used. This is a "worst-case" selection of the standby supply voltage. For the intra-chip DRV distribution in Figure 5.1, the worst-case supply voltage is $200mV$. The leakage-power reduction from $V_{dd} = 1000$mV to the largest DRV voltage in many test-chips has been studied in detail by Qin et al. [13, 53]. In this Chapter, leakage-power reduction beyond this worst-case approach will be presented and validated using experiments with fabricated chips.

5.1 Modeling assumptions

Let v_δ be the quantization step at which the DRV of various cells are measured in the laboratory. The DRV histogram will be obtained for $\mathcal{V} := \{0, v_\delta, 2v_\delta, 3v_\delta, \ldots\}$. The variation of DRV will be modeled by the observed (discrete) probability distribution $\mu_h(x), x \in \mathcal{V}$. For example, for Figure 5.1, $v_\delta = 10$mV and the support set for probability distribution is $\mu_h(x), x \in \big\{70, 80, \ldots, 190\big\}$. The DRV empirical distribution function, or simply *distribution function*, is $F_h(x) = \sum_{z \leq x} \mu_h(z)$. Since the experimental DRV distribution is measured at quantization step of v_δ, the supply voltage will be swept in multiples of v_δ. A cell will retain the stored data successfully if the supply voltage is strictly greater than the cell's DRV voltage. The DRV is assumed to be random but fixed after manufacture.

Note that no attempt is made to model the DRV distribution by known smooth probability distribution(s). Using the empirical distribution is similar to bootstrap estimation methods [69]. Using this procedure is advantageous because the knowledge about DRV distribution's analytical (parametric) form is not required. Besides, this distribution varies from chip to chip as observed in our experiments. It may also vary on the same chip with time due to TDDB, HCI, or NBTI (on a scale of days) which will require some form of adaptive DRV distribution learning. Parametric modeling of DRV distribution and its slow temporal variation are beyond the scope of this work, and it is left as a future work.

Trap charge assisted erratic fluctuations (see [20]) are not expected to affect storage in 90nm CMOS process at subthreshold voltage levels for the following reason: the DRV is obtained by solving current equations in the subthreshold regime [12]. The gate-leakage current can vary significantly with time due to trapping and de-trapping or charges at high supply voltages [20]. However, the gate-leakage current and its variations are much smaller at low voltages (around $200mV$) compared to the subthreshold leakage currents. This is because gate-leakage decreases exponentially with the supply voltage [70], whereas subthreshold leakage decreases linearly with the supply voltage (see Fig. 5.4). Therefore, DRV in the 90nm CMOS process does not depend significantly on gate-leakage, and is approximately constant with time.

5.1.1 Notation

In the rest of the paper, the *standby power* will be called as *power* for brevity. Let v_δ be the quantization step at which the DRV of various cells are measured. The DRV histogram will be obtained for $\mathcal{V} := \{0, v_\delta, 2v_\delta, 3v_\delta, \ldots\}$. The DRV distribution function, for example as in Figure 5.1, will be denoted by $F_h(x)$. The standby supply voltage will be represented by v_S (the suffix S is used to standby). The symbol $\mathbb{P}$ will be used for the probability of a set with respect to the distribution $F_h(x)$. Any vectors like $(x_1, x_2, \ldots, x_n)$ will be represented as x_1^n. Finally, recall that $H_2(p) = -p \log_2 p - (1-p) \log_2(1-p), \quad 0 \leq p \leq 1$ stands for the binary entropy function [50].

5.2 Standby SRAM: theoretical results

In this section, the SRAM cell retention model and the proposed standby SRAM architecture will be presented next. The description of the retention model is important for understanding the architecture and therefore it will be presented first. Using these models, fundamental bounds on (standby) power reduction will be analyzed. Finally, practical circuits, which approach these fundamental bounds, will be explored for implementation.

5.2.1 SRAM cell Retention model

For each SRAM cell, there is a data-retention-voltage (DRV), above which the stored data bit (0 or 1) is stored reliably [12]. However, if the supply voltage is lowered below the DRV, then the stored bit degenerates to a preferred digital (binary) state $S \in \{0, 1\}$ [12]. These features of an SRAM cell are captured in the following mathematical model (see Fig. 5.2). The cell has two statistically independent parameters: (i) a time-invariant, positive and continuous-valued threshold-voltage DRV, and (ii) an *equally likely* binary stuck-at state $S \in \{0, 1\}$. The inputs to the cell are the supply voltage v_S and a bit

$X \in \{0,1\}$ to be stored. The retention model for the SRAM cell is as follows:

$$\begin{aligned} Y &= X \quad \text{if} \quad DRV < v_S, \\ &= S \quad \text{if} \quad DRV \geq v_S, \end{aligned} \tag{5.1}$$

where $Y \in \{0,1\}$ is the output bit. If $v_S \leq DRV$, then there is a *hold-failure*. This digital abstraction is sufficient for establishing upper bounds of power reduction and it is illustrated in Figure 5.2.

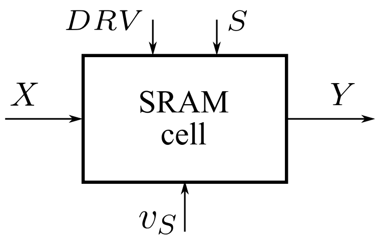


Figure 5.2: The SRAM cell has two statistically independent parameters: (i) a time-invariant positive continuous-valued threshold-voltage called DRV, and (ii) a binary stuck-at state $S \in \{0,1\}$. The inputs are the supply voltage v_S and a bit $X \in \{0,1\}$ to be stored. The output is $Y = X$ if $v_S > DRV$ and S otherwise.

5.2.2 Standby SRAM low-power architecture

The general architecture which trade-offs supply voltage, hold-failures, and error-correction schemes is shown in Figure 5.3. Let the standby supply voltage be $v_S \in \mathcal{V}$ at v_δ quantization step. The worst-case solution is the largest DRV on the chip at which every cell retains data reliably (see Figure 5.1). In contrast, a general error-protected SRAM operation is described next.

Let $B_1^k = (B_1, B_2, \ldots, B_k)$ be the data vector to be stored. Using an error-control code, B_1^k is encoded into X_1^n and stored in n SRAM cells ($n \geq k$). Cells have i.i.d. pairs

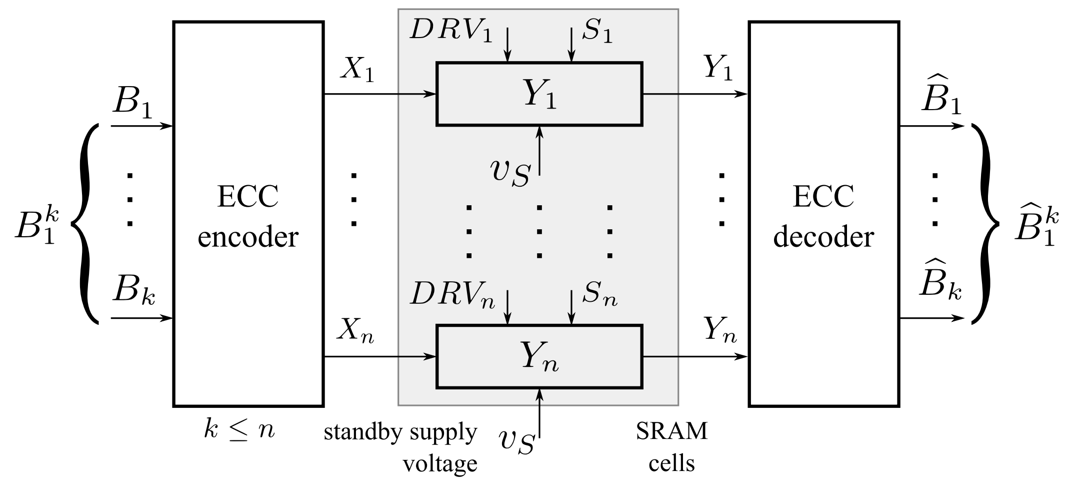


Figure 5.3: Let B_1^k be the data vector to be stored. Then B_1^k is encoded into X_1^n and stored in n SRAM cells. The j^{th} stored bit is stuck-at S_j if $DRV_j \geq v_S$, otherwise X_j is read-out. The decoder reads Y_1^n and outputs $\widehat{B}_1^k$. The voltage v_S is selected such that $\mathbb{P}(\text{outage})$ is negligible (see (5.3)).

of independent DRV and S realizations. [1] The j^{th} stored bit is stuck-at S_j if $DRV_j \geq v_S$, otherwise X_j is successfully retained. At the end of standby, Y_1^n is decoded to $\widehat{B}_1^k$. Let $1 \leq i \leq 2^k$ be the integer representation of B_1^k.

Next, a suitable hold-failure probability criterion will be introduced, which will act as a constraint to supply voltage reduction. Observe that if the supply voltage is at the largest DRV, then there will be no hold-failures. Motivated by this observation, an "outage" probability criterion will be described. Note that if v_S is smaller than the largest DRV, there is a non-zero probability that none of the cells will retain the bit. However, this situation is unrealistic. An SRAM block realization is in *outage* if there is at least one stored vector B_1^k for which $\widehat{B}_1^k \neq B_1^k$. The outage probability will be larger than the average (or maximum) probability of error, which is typically used in channel coding theorems in

[1]The assumption that DRV across cells are independent is a worst-case assumption as discussed at the end of Sec. 5.2.4.

information theory [50, Chapter 8]. Let $f : \mathbb{B}^k \to \mathbb{B}^n$ and $g : \mathbb{B}^n \to \mathbb{B}^k$ be the encoder and decoder operations (functions). Mathematically, the outage set $\mathcal{E}$ is given by,

$$\mathcal{E} = \bigcup_{i=0}^{2^k} \mathcal{E}_i, \text{ where,} \tag{5.2}$$

$$\mathcal{E}_i = \{g(Y_1^n) \neq i | X_1^n = f(i)\}. \tag{5.3}$$

Recall that integers from 1 to 2^k are used to index all the words in $\mathbb{B}^k$. The outage probability is defined as,

$$p_{\text{outage}} = \mathbb{P}(\mathcal{E}), \tag{5.4}$$

where the probability is taken over DRV and S distributions. In the proposed scheme, for any error-control code, the voltage v_S is chosen such that the outage probability is negligible. This condition ensures that an n-bit row of SRAM stores all input words from $\mathbb{B}^k$ with high reliability. Even with this strict definition of outage, there is a (small but non-zero) probability that a block of SRAM will not work. The technique of row-redundancy will be used to avoid any blocks in outage. Since hold-failures are at fixed locations (on the scale of decoding time), they can be corrected by testing and row-redundancy [71].

Since v_S is a free variable, power per useful-bit (or any other cost function) can be optimized over its range. For an outage of ϵ, we define the *power per bit* as,

$$\mathcal{P}_\epsilon(v_S) := \frac{1}{k} \cdot (\text{Total standby power}). \tag{5.5}$$

If ϵ can be made arbitrarily small by choosing $n \to \infty$, then the power per bit function will be called as $\mathcal{P}(v_S)$. The total standby power dependence on v_S will be established next.

5.2.3 Power dependence on the supply voltage

Let T_s be the standby duration. Let $E_{\mathcal{C}}$ be the average encoder-decoder computational energy (over codewords B_1^k) any generic error-control code $\mathcal{C}$. The total standby power is,

$$P_T(v_S) = P_L(v_S) + \frac{E_{\mathcal{C}}}{T_s}, \tag{5.6}$$

where $P_L(v_S)$ is the total leakage-power. [2] Note that the computation energy $E_{\mathcal{C}}$ is finite and it gets normalized by the standby time T_s. Since low-duty cycle applications have large T_s, therefore the $(E_{\mathcal{C}}/T_s)$ term becomes negligible. The dependence of the leakage-power on the supply voltage is examined next.

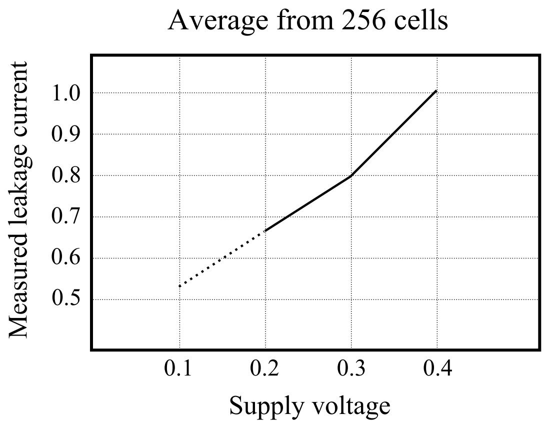


Figure 5.4: The normalized measured leakage-current for 256 SRAM cells is shown as a function of the supply voltage. In the range $100 - 200$mV, the leakage-current is approximately linear.

The leakage-current in the $100-200mV$ range is approximately linear in the supply voltage, i.e., $I_L = Gv_S$, where G is a constant. This is confirmed by experimental leakage-

[2]The computation energy $E_{\mathcal{C}}$ will vary due to process variations. The variation of the average computation energy is out of the scope of this work.

current measurements done in the lab (see Fig. 5.4). Thus, the power per bit of the SRAM cell is,

$$\mathcal{P}_{\epsilon}\left(v_S\right) = \frac{n}{k} \cdot G v_S^2 + \frac{E_{\mathcal{C}}}{kT_s}, \tag{5.7}$$

where the code $\mathcal{C}$ has an outage given by (5.3).

5.2.4 Fundamental bounds on the power reduction

In this section, the fundamental bounds on the power per bit $\mathcal{P}\left(v_S\right)$ will be derived. These bounds will be dependent on the DRV-distribution. For deriving these bounds, the following important points must be noted:

- For $T_s \to \infty$, i.e., when the standby time is much larger than the encoding-decoding time, the coding energy overhead becomes negligible. Under this condition, the standby power is minimum and will be considered first.
- The coding and latency aspects will be examined after the fundamental asymptotic benchmarks for power are established (see Section 5.2.5 and Remark 5.2.1).
- The outage $\epsilon > 0$ can be made arbitrarily small in an asymptotic setting when $n \to \infty$.

Recall that the hold-failure probability for an SRAM cell is given by,

$$p_h\left(v_S\right) = \sum_{z \geq v_S,\ z \in \mathcal{V}} \mu_h(z), \tag{5.8}$$

where $\mu_h(x), x \in \mathcal{V}$ is the (discrete) probability distribution of DRV. Using this notation, the following theorem can be stated:

Theorem 5.2.1. *Let v_S be the standby supply voltage and $p_h(v_S)$ be as in (5.8). For each voltage v_S such that $p_h(v_S) < 0.25$, the minimum power per bit, over all coding strategies, satisfies,*

$$\frac{Gv_S^2}{1 - H_2\left(p_h\left(v_S\right)/2\right)} < \mathcal{P}\left(v_S\right) < \frac{Gv_S^2}{1 - H_2\left(2p_h\left(v_S\right)\right)}, \tag{5.9}$$

where G is a constant. Since v_S is a free variable, the upper and lower bounds can be optimized over the choice of v_S to obtain bounds on $\min_{v_S} \mathcal{P}(v_S)$.

Proof. See Appendix 7. For the *DRV* distribution in Figure 5.1, the reduction in $\min_{v_S} \mathcal{P}\left(v_S\right)$ with respect to the worst-case is between 40% and 49%. □

The bounds on $\mathcal{P}\left(v_S\right)$ are derived using ideas from Information theory [50, Chapter 8] and error-control code theory [31], respectively. The details are presented in the Appendix for brevity. Observe that the denominator $1 - H_2(p_h(v_S)/2)$ and the numerator v_S^2 increase as v_S increases. When v_S is small (around 70mV), the increase in denominator term is rapid compared to the numerator. The trend reverses for large v_S (around 200mV). Thus, the optimum *power per bit* is achieved for an intermediate values of v_S. Similar argument holds for the upper bound.

The *power per bit* bounds as a function of $p_h\left(v_S\right)$ are illustrated in Figure 5.5. The minimum value of the upper bound and the lower bound are 40% and 49% less than the worst-case, respectively.

Remark 5.2.1. *Spatial correlation in the DRV can be exploited with better coding strategies. However, from the test-chip measurements, a small spatial correlation factor (< 0.1) in the DRV data was observed. Since the measured correlation is small, the improvement*

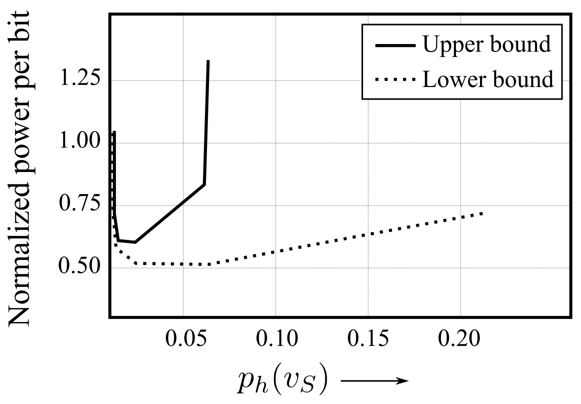


Figure 5.5: Power per bit bounds are plotted as functions of the DRV-failure rate $p_h(v_S)$. The minima of upper and lower bounds are 40% and 49% lower than the worst-case.

in power per bit reduction will be insignificant. Therefore, statistical i.i.d. assumption is assumed. This assumption will be verified again in the experimental section (see Section 5.3).

5.2.5 Practical low-latency codes and power per bit

When coming out of the standby mode, decoding step in Figure 5.3 introduces extra latency. As $n \to \infty$, the *power per bit* for a code approaches these fundamental bounds. However, as the block length n increases, the latency and complexity of the code increases as well. Practical SRAM design typically requires the data-output within a latency of a few clock cycles. Motivated by this concern, *power per bit* reduction as a function of the block length n will be studied for two bounded distance decoding based error-correction code families: (i) the Hamming codes and (ii) the Reed Muller codes. The outage probability will be fixed at $\epsilon = 0.01$. As noted earlier, rows in outage will be corrected by row-redundancy [71].

The outage condition as stated in (5.3) is complex since it is the union of an exponential number of sets. Fortunately, the condition simplifies considerably with bounded distance decoding codes. This development is presented next. For a bounded distance decoding based code with parameters $[n, k, d]$, a decoding error happens when the number of error exceeds $u := \left\lfloor \frac{d-1}{2} \right\rfloor$. It can be verified that an outage will be present if and only if the number of hold-failures is at least $(u + 1)$. Thus outage condition simplifies to,

$$\epsilon = \mathbb{P}\big[DRV_{(n-u)} \geq v_S\big], \tag{5.10}$$

where $DRV_{(j)}$ is the j^{th} largest random DRV. For example, if $d = 3$ and $u = 1$, then $DRV_{(n)} \geq v_S$ results in two DRV failures (and hence decoding error will be present). The power per bit function for bounded distance decoding codes is given by,

$$\mathcal{P}_{0.01}\left(v_S\right) = G \cdot \frac{n}{k} \cdot \left(v_S\right)^2, \tag{5.11}$$

where v_S is the smallest possible voltage at which the outage ϵ is less than 0.01. The expression in (5.11) is plotted in Figure 5.6 for the Hamming and Reed Muller error-correction code families using the empirical DRV distribution of Figure 5.1. *The* $[31, 26, 3]$ *Hamming code has the minimum* $\mathcal{P}_{0.01}\left(v_S\right)$ *at* 33% *less than the worst-case.* On the other hand, $[256, 211, 8]$ Reed Muller code has the minimum $\mathcal{P}_{0.01}\left(v_S\right)$ at 35% less than the worst-case. A significant fraction, 33% out of the optimum 40% (see Theorem 5.2.1), *power per bit* reduction is achieved with a single clock-cycle latency Hamming code. The gap can be reduced with higher-complexity coding. The returns are marginal, e.g., 2% extra *power per bit* can be saved by a Reed Muller code with an 8-times larger block length.

Motivated by diminishing returns with longer block length codes, the $[31, 26, 3]$ Hamming code was selected for implementation. The encoder and decoder for this code

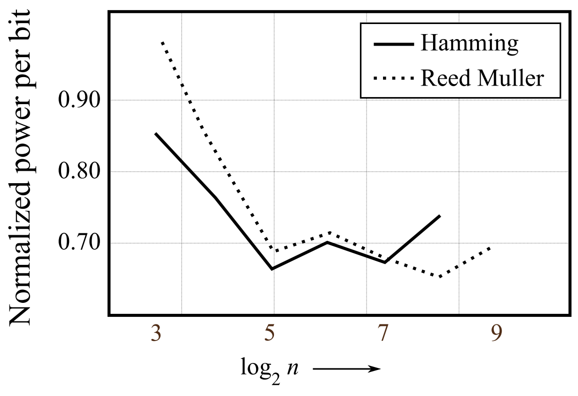


Figure 5.6: For an outage $\epsilon = 0.01$, the optimum power per bit for Hamming and Reed Muller codes are plotted. Maximum power reduction is achieved at $n = 31$ for Hamming codes and at $n = 256$ for Reed Muller codes.

were synthesized using CAD tools for the (90nm CMOS technology). The estimated average encoding and decoding energy for a 26-bit word were of the order of 1pJ. The measured average leakage-current at $200mV$ for an SRAM cells is in the range of 100pA. [3] Based on this data, it is estimated that for $T_s \geq 100$ms, a *power per bit* reduction of 33% will be achieved. The latency of this encoder and decoder is 1-clock cycle (2ns) at $V_{dd} = 1$V.

5.2.6 Chip-implementation overview

Based on an industry IP module, a 90nm 26kbit storage SRAM, integrated with a $[31, 26, 3]$ Hamming code, was fabricated. The chip layout is shown in Figure 5.7. The Original SRAM design was from an industry IP module. The Ultra Low-Leakage SRAM is based on the original design but uses circuit optimization to improve the leakage path balance and reduces device mismatch. The result is a narrower DRV distribution and reduced worst-case DRV voltage. This work is detailed in [53]. Due to slightly larger

[3]The exact leakage current numbers cannot be shared due to IP issues.

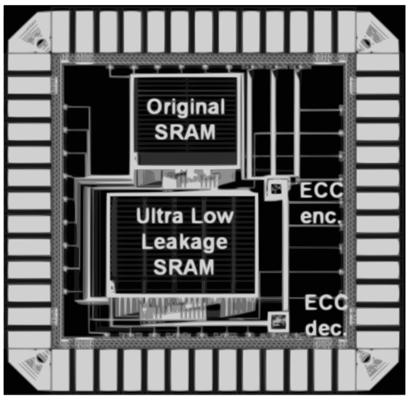


Figure 5.7: Fabricated SRAM layout in an industrial 90nm CMOS technology is shown.

transistor sizes, the modified SRAM takes a larger area but has smaller leakage. The 26kbit data is encoded using ECC encoding block before storage. After readout, the bits are decoded using ECC decoding block.

5.3 Optimization results from SRAM chips

The features of implemented SRAM chips are highlighted in Section 5.2.6. *Twenty four* test chips were fabricated with these features for testing. Results will be presented from these twenty four chips. Before presenting the measurements, the expected nature of results is discussed. Intra-die variation in DRV, and hence power reduction with coding is expected (as discussed in Section 5.2.4). The DRV distribution is expected to have intra-die as well as inter-die variations, therefore *power per bit* reduction should vary from chip to chip. It will be shown that inter-die variations in *power per bit* reduction is significant. Small correlations in the spatial pattern of DRV from the chip used to produce the distribution in Figure 5.1 were observed. Similar negligible spatial correlation in the DRV parameter is expected. Finally, larger worst-case DRV should result in higher power reduction. These features are expected from experimental chips. The actual results from the experimental chips are now discussed. The analysis is performed on experimentally measured DRV values from fabricated-chips.

5.3.1 Spatial correlation of DRV

Correlation in the spatial DRV pattern on a chip can be exploited with better coding strategies. In the analysis part, a small spatial correlation factor (< 0.1) in the DRV data was observed (see Remark 5.2.1). The empirical spatial correlation of the DRV

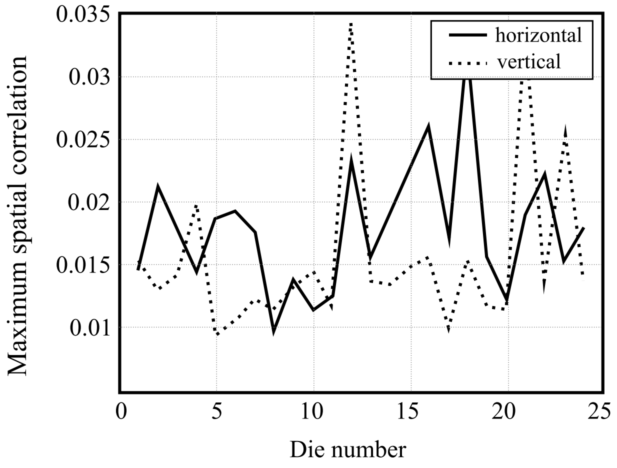


Figure 5.8: The maximum absolute empirical horizontal and vertical correlation coefficients are plotted as a function of experimental chip index. The maximum observed correlation coefficient is less than 3.5%.

among SRAM cells was measured in the following way. For spatially laid out SRAM cells, let $DRV(i, j, m)$ be the DRV of cell in the location $(i, j), 1 \leq i \leq 31, 1 \leq j \leq 1000$ of experimental-chip m. For notational simplicity, the index m will be omitted from the equations. Let

$$\overline{DRV} = \frac{1}{31000} \sum_{i=1}^{31} \sum_{j=1}^{1000} DRV(i,j), \tag{5.12}$$

be the empirical mean. Let $\sigma^2 = \left(\overline{DRV^2} - \overline{DRV}^2\right)$ be the empirical variance. The empirical horizontal correlation for is defined as,

$$\text{hor}(i') := \frac{\sum_{i=1}^{31-k} \sum_{j=1}^{1000} \left(DRV(i,j)DRV(i+i',j) - D\bar{R}V^2\right)}{1000(31-i')\sigma^2}. \tag{5.13}$$

Similarly, the vertical correlation is defined as,

$$\text{ver}(j') = \frac{\sum_{i=1}^{31} \sum_{j=1}^{1000-k} \left(DRV(i,j)DRV(i,j+j') - D\bar{R}V^2\right)}{(1000-j')31\sigma^2}. \tag{5.14}$$

The maximum absolute values of hor(k) and vert(k) as a function of the die-number m are plotted in Figure 5.8. A maximum empirical correlation of 3.5% is observed across all chips. This observation re-affirms that independence of DRV across cells is a good assumption for analysis.

5.3.2 Power per bit and its reduction in experimental-chips

Let $DRV_{\max}(m)$ be the largest $DRV(i,j)$ observed in chip m. Recall that the leakage-power for a cell is approximately quadratic in the supply voltage, i.e., $P_L = G(v_S)^2$. The percentage reduction calculations are independent of the constant G and hence it will be ignored in the further calculations. Therefore, $P_L = (v_S)^2$ in some arbitrary unit (a.u.). The worst-case power per bit $(DRV_{\max}(m))^2$, the optimum achievable *power per bit* for bounded distance decoding schemes (optimized upper bound of (5.9)), and Hamming code's *power per bit* will be compared. The upper bound on optimum power per bit reduction in (5.9) is used for comparison with the $[31, 26, 3]$ Hamming code based implementation's power per bit reduction. This is reasonable since the upper bound is the minimum achievable *power per bit* when coding schemes are restricted to the class of bounded-distance decoding codes. The Hamming code belongs to this class.

The comparison is plotted in Figure 5.9(a). Observe that a significant variation in the worst-case *power per bit* "flattens" in the presence of coding. This is a desirable property. Also observe that the gap between the implementation and theoretical optimum is approximately constant. This shows that the $[31, 26, 3]$ Hamming code adapts well for different observed DRV-distributions and it is a good design choice.

The percentage power reduction with respect to the worst-case strategy is computed next. Recall that the worst-case power per bit is $(DRV_{\max}(m))^2$, where $DRV_{\max}(m)$ is the largest $DRV(i, j)$ on chip m. The optimum *power per bit* is computed using the upper bound in (5.9). Significant intra-die variation in the percentage *power per bit* reduction is observed. The intra-die distribution dependent theoretical bound on *power per bit* reduction varies from 23-52%, while the implementation reduces *power per bit* by 12-46%. As expected, there is a performance gap between the implementation and the optimum, but this gap is small. Prior analysis suggested a gap of 7% and the observed numbers are close to the prediction.

5.3.3 Row redundancy design

Let $DRV(1, j), DRV(2, j), \ldots, DRV(31, j)$ be independent and identically distributed DRV-values coming from the test-chip distribution $F_h(v)$. Physically, this vector represents the j^{th} row on the test chip. The test chip index m is omitted for simplicity. The $[31, 26, 3]$ Hamming code can correct single-bit error. The chance of decoding failure is the

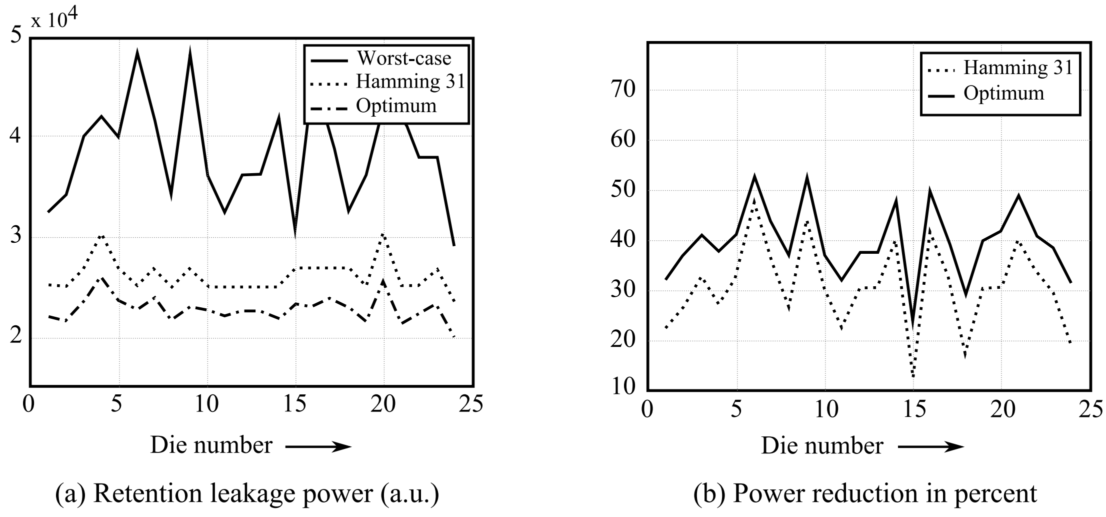


Figure 5.9: (a) The leakage-power (in a.u.) for the worst-case method, the $[31.26, 3]$ Hamming code based implementation, and the theoretical optimum (see (5.9)) are compared. (b) Power reduction for the $[31, 26, 3]$ Hamming code based implementation and the theoretical optimum are compared. The implementation tracks the optimum within a close margin of 6-11%.

probability that two or more cells in these 31 cells have a $DRV(i, j) \geq v_S$. The decoding failure determines the amount of extra rows needed for row-redundancy. Its probability is given by,

$$\epsilon(v_S) := \epsilon = \mathbb{P}\left[DRV_{(30)}(j) \geq v_S\right], \tag{5.15}$$

where $DRV_{(i)}(j)$ is the i^{th}-largest DRV in the vector $DRV(1, j), DRV(2, j), \ldots, DRV(31, j)$. The Hamming code implementation was designed with a probability of $\epsilon = 1\%$, i.e., a retention voltage v_S was selected such that for the experimental-chip DRV distribution $\epsilon(v_S) \leq 0.01$. The rows with decoding failure at the specified retention voltage v_S can be replaced by row-redundancy techniques [71]. For the experimental-chips, the supply voltage v_S was fixed using each chip's intra-die DRV-distribution to meet the decoding failure probability condition, i.e., $\epsilon \leq 0.01$. This voltage v_S is obtained by a simple calculation based

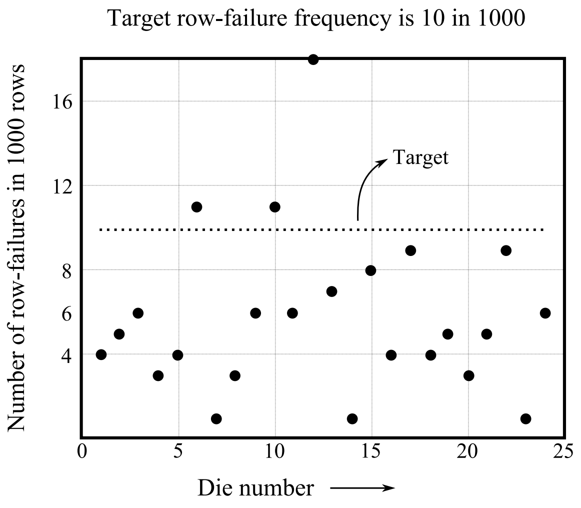
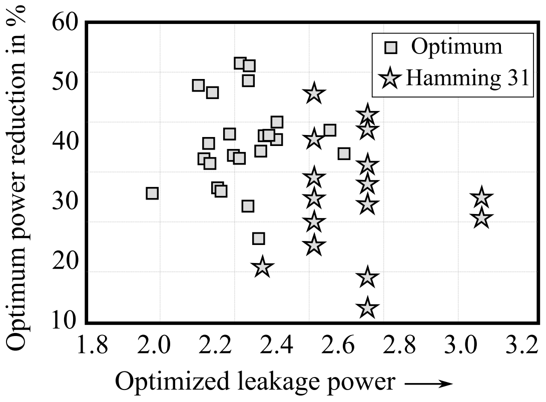


Figure 5.10: The number of rows in decoding-failure are plotted as a function of experimental-chip number. The average number of failures, 7 in 1000, satisfies the 1% decoding-failure target (see (5.15)).

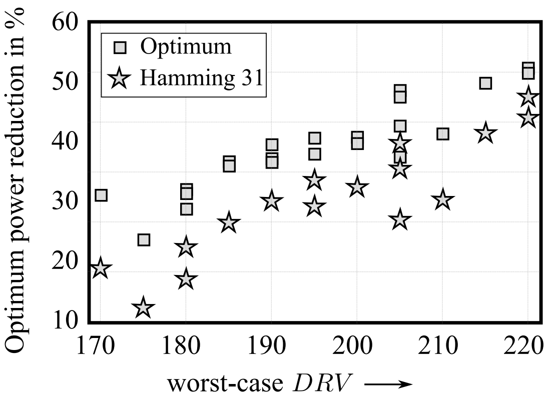
on the intra-die DRV distribution (and hence it can be adapted on-chip). Next, at supply voltage v_S the number of rows with $DRV_{(30)}(j) > v_S$ were counted over the choice of j. Let this count for each chip be $c(m), m = 1, 2, \ldots, 24$, over 1000 rows in the experimental-chips. This count $c(m)$ is plotted in Figure 5.10. The average number of row-failures is around 7 in 1000. Thus, the design target of 10 in 1000 is satisfied.

5.3.4 Parameter dependencies

Scatters plots were used to examine any dependence between *power per bit* reduction (see Figure 5.9(b)) with leakage-power (see Figure 5.9(a)) or worst-case DRV voltage ($DRV_{\max}$). The scatter plot in Figure 5.11(a) shows *power per bit* for the optimum bound and the $[31, 26, 3]$ Hamming implementation. No relationship between the leakage-power and the optimum *power per bit* reduction was observed. A dependence between the worst-case DRV intra-chip DRV voltage ($DRV_{\max}$) and *power per bit* reduction is observed. On



(a) leakage power and and power reduction



(b) worst-case DRV and power reduction

Figure 5.11: (a) No relationship between *power per bit* and optimum *power per bit* reduction was observed. (b) On average, power reduction increases linearly with the intra-chip worst-case DRV voltage of an SRAM.

average, a linear increase in *power per bit* reduction as a function of DRV_{max} is observed.

Remark: To measure the effect of erratic fluctuations in the gate-leakage, repeated DRV measurements were taken at temporal intervals for a few seconds [20]. This was done to observe any temporal variation present in the DRV. As expected, temporal variations were not observed in the DRV-values since the gate-leakage currents are negligible compared to the subthreshold leakage at voltages around $200mV$ for the 90nm CMOS technology.

Chapter 6

Conclusions

SRAM leakage-power is a significant fraction of the total power consumption on a chip. This work studied SRAM leakage-power reduction using system level design techniques like data-refresh and error-correction code, with a decoding error probability constraint.

The bit-error probability in SRAM cell increases due to supply voltage reduction. Low-complexity macro-models were studied to estimate the bit-error probability of SRAM cells, due to supply voltage reduction and process-variations. Critical charge method, coupled with Monte Carlo simulations, was used to estimate the soft-error rate. Noise-margin based approach was used to obtain the static parametric failures. Write-failures were estimated to be the dominant parametric failure mechanism at low supply voltages. An extreme-value theory based estimation procedure was developed to estimate access-time and write-time failures. The effect of supply voltage reduction on SRAM cell's speed was computed. Bit-error probability comparisons between a custom 65nm technology SRAM cell and a custom 90nm technology SRAM cell were presented.

The bit-error probability increase in SRAM cells can be compensated by suitable choices of error-correction code and data-refresh rate. A *power per bit* cost function was optimized over the choice of supply voltage, while meeting the decoding error probability constraint. It was estimated that 60% leakage power reduction can be achieved by data-refresh and supply voltage reduction for the 90nm and the 65nm technologies. At lower voltages, parametric failures dominate the decoding error probability, and they cannot be compensated by data-refresh. Thus, parametric failures limit the efficacy of data-refresh. For supply voltage reduction in the subthreshold region, multiple-bit error-correction capability is needed. Circuit-level leakage-power reduction estimates, as a function of the minimum distance of the code, were presented. *Power per bit* reduction by more than 90% was estimated for the 90nm and the 65nm technologies.

For the special case of standby SRAM where only hold-failures are important, theoretical limits and experimental results were presented. Retention of stored data at lowest possible leakage-power was the only target in this case. Using the proposed supply voltage reduction, coding, and data-refresh method, the leakage-power per useful bit was minimized. Using techniques from information and coding theory, fundamental bounds on the minimum leakage-power per bit needed for storage, while taking the hold-failure probability distribution into account, were established. For experimentally observed DRV-distributions from custom built SRAM chips, a $[31, 26, 3]$ Hamming code based retention scheme achieved 79% (on average) of the leakage-power reduction compared to the fundamental limit. These results were verified by twenty-four experimental chips manufactured in an industrial 90nm CMOS process. Nearly uncorrelated DRV values were observed. Significant inter-die variations in the optimized leakage-power and optimum power per bit reduction were observed. This inter-die variation in optimization results indicate the necessity of intra-chip DRV-distribution for an optimal design. Given the knowledge of this distribution, the analytical design assumptions and results were in consonance with the observed experimental data.

6.1 Future work

This work opens possible avenues for challenging problems in the future. Experimental verification of analytical results presented here will be very interesting. Due to low bit-error rate, large SRAM test arrays will be needed or some form of accelerated testing will be required. Since there are many failure mechanisms, any accelerated testing-methods have to be designed carefully. For example, taking SRAM chip at high altitudes will increase the soft-error rate, but it will leave parametric failures unaffected.

While modeling bit-error probability, erratic fluctuations were not considered since they are not very well known within the literature. Of late, the fluctuating gate-leakage current has been modeled as a random telegraph noise [20]. Using this information, coupled with the magnitude and time constant of this random telegraph noise, estimation of bit-error probability due to erratic fluctuations, and its effect on leakage-power reduction will be an interesting endeavor.

Exploiting the statistics of supply voltage noise to reduce supply noise margin from 100mV is a challenging problem in itself. The challenges consist of supply noise distribution modeling, the characterization of magnitude variation of supply noise in the standby state, and the effects of supply noise on SRAM cell stability. It will also require new methods to estimate parametric failures, in the presence of a fluctuating supply voltage.

Unlike traditional circuit optimization works, the focus here was on system level optimization without changing the SRAM cell parameters like transistor threshold voltage V_T, transistor channel length L, or transistor width W etc. Coupled with the macro-models based bit-error probability estimation methods, a joint circuit and system leakage-power optimization framework for SRAM cells can be explored. Such exploration will result in various trade-offs between SRAM cell parameters and its probabilistic properties.

In this work, it was proposed that erasures should be learned when a parity check error is observed in the code. It was also observed that erasure probability is very small, except for $v \leq 0.2$V. These erasure locations or addresses can be stored in a separate memory (small overhead) to aid the decoder. This will eliminate the necessity of real-time check with test-patterns, but it will introduce storage and latency overhead dependent on the number of parametric failures present in any SRAM block. This approach and any subsequent trade-offs are very interesting, and they can be analyzed in the future.

Finally, a probabilistic channel model based SRAM architecture was used to reduce a metric of choice (like leakage-power). This technique can be 'ported' to other memory systems. In the presence of process-variations, error-correction codes provide a convenient method to move away from worst-case design conformity of SRAM cells. Perhaps this technique's potential will be discovered in the future, when designers will aggressively pursue power reduction in memories.

Bibliography

[1] J. M. Rabaey, A. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits – A Design Perspective*, 2nd ed. Prentice Hall, NJ, USA, 2003.

[2] S. Borkar, "Obeying Moore's law beyond 0.18 micron," in *Proceedings of the* 13th *IEEE International ASIC/SOC Conference*, 2000, pp. 26–31.

[3] J. M. Rabaey, J. Ammer, T. Karalar, S. Li, B. Otis, M. Sheets, and T. Tuan, "Picoradics for wireless sensor networks: the next challenge in ultra-low-power design," in *IEEE Solid-State Circuits Conference.* ISSCC Digest of Technical Papers, February 2002, pp. 156–157.

[4] S. Zhao, A. Chatterjee, S. Tang, J. Yoon, S. Crank, H. Bu, T. Houston, K. Sadra, A. Jain, Y. Wang, D. Redwine, Y. Chen, S. Siddiqui, G. Zhang, T. Laaksonen, C. Hall, S. Chang, L. Olsen, T. Riley, C. Meek, I. Hossain, J. Rosal, A. Tsao, J. Wu, and D. Scott, "Transistor optimization for leakage power management in a 65nm CMOS technology for wireless and mobile applications," *IEEE Symposium on VLSI Technology*, pp. 14–15, June 2004.

[5] K. Zhang, U. Bhattacharya, C. Zhanping, F. Hamzaoglu, D. Murray, N. Vallepalli,

Y. Wang, B. Zheng, and M. Bohr, “SRAM design on 65nm CMOS technology with dynamic sleep transistor for leakage reduction,” *IEEE Journal of Solid-State Circuits*, vol. 40, no. 4, pp. 895–901, April 2005.

[6] M. Powell, S. Yang, B. Falsafi, K. Roy, and T. N. Vijaykumar, “Gated-V_{dd}: a circuit technique to reduce leakage in deep-submicron cache memories,” in *Proceedings of the International Symposium on Low Power Electronics and Design.* New York, NY, USA: ACM, 2000, pp. 90–95.

[7] A. Agarwal and K. Roy, “A noise tolerant cache design to reduce gate and sub-threshold leakage in the nanometer regime,” in *Proceedings of the International Symposium on Low Power Electronics and Design.* New York, NY, USA: ACM, 2003, pp. 18–21.

[8] N. Azizi, F. N. Najm, and A. Moshovos, “Low-leakage asymmetric-cell SRAM,” *IEEE Transactions on Very Large Scale Integration Systems*, vol. 11, no. 4, pp. 701–715, August 2003.

[9] Z. Guo, S. Balasubramanian, R. Zlatanovici, T. J. King, and B. Nikolić, “FinFET-based SRAM design,” in *Proceedings of the International Symposium on Low Power Electronics and Design.* New York, NY, USA: ACM, 2005, pp. 2–7.

[10] B. Calhoun and A. P. Chandrakasan, “Static noise margin variation for sub-threshold SRAM in 65nm CMOS,” *IEEE Journal of Solid-State Circuits*, vol. 41, no. 7, pp. 1673–1679, July 2006.

[11] N. S. Kim, K. Flautner, D. Blaauw, and T. Mudge, “Drowsy instruction caches –

leakage power reduction using dynamic voltage scaling and cache sub-bank prediction," in *35th Annual IEEE/ACM Intl. Symp. on Microarchitecture*, 2002, pp. 219–230.

[12] H. Qin, Y. Cao, D. Markovic, A. Vladimirescue, and J. Rabaey, "SRAM leakage suppression by minimizing standby supply voltage," in *Proceedings of International Symposium on Quality Electronic Design*. New York, NY, USA: IEEE, 2004, pp. 55–60.

[13] H. Qin, R. Vattikonda, T. Trinh, Y. Cao, and J. Rabaey, "SRAM cell optimization for ultra-low power standby operation," *Journal of Low Power Electronics*, vol. 2, no. 3, pp. 401–411, December 2006.

[14] K. Nii, Y. Tsukamoto, T. Yoshizawa, S. Imaoka, Y. Yamagami, T. Suzuki, A. Shibayama, H. Makino, and S. Iwade, "A 90-nm low-power 32-kb embedded sram with gate leakage suppression circuit for mobile applications," *IEEE Journal of Solid-State Circuits*, vol. 39, no. 4, pp. 684–693, April 2004.

[15] K. Kanda, T. Miyazaki, M. K. Sik, H. Kawaguchi, and T. Sakurai, "Two orders of magnitude leakage power reduction of low voltage SRAMs by row-by-row dynamic V_{dd} control (RRDV) scheme," *The Annual IEEE International ASIC/SOC Conference*, pp. 381–385, September 2002.

[16] J. F. Ziegler, H. W. Curtis, H. P. Muhlfeld, C. J. Montrose, B. Chin, M. Nicewicz, C. A. Russell, W. Y. Wang, L. B. Freeman, P. Hosier, L. E. LaFave, J. L. Walsh, J. M. Orro, G. J. Unger, J. M. Ross, T. J. O'Gorman, B. Messina, T. D. Sullivan, A. J. Sykes, H. Yourke, T. A. Enger, V. R. Tolat, T. S. Scott, A. H. Taber, R. J. Sussman, W. A. Klein, and C. W. Wahaus, "IBM experiments in soft fails in computer

electronics (1978-1994)," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 3–18, 1996.

[17] A. Bhavnagarwala, S. Kosonocky, C. Radens, K. Stawiasz, R. Mann, Y. Qiuyi, and K. Chin, "Fluctuation limits and scaling opportunities for CMOS SRAM cells," in *Technical Digest of the International Electron Devices Meeting.* New York, NY, USA: IEEE, December 2005, pp. 659–662.

[18] S. R. Nassif and J. N. Kozhaya, "Fast power grid simulation," in *Proceedings of the Design Automation Conference.* New York, NY, USA: ACM, 2000, pp. 156–161.

[19] E. Alon, V. Stojanovic, and M. A. Horowitz, "Circuits and techniques for high-resolution measurement of on-chip power supply noise," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 4, pp. 820–828, April 2005.

[20] M. Agostinelli, J. Hicks, J. Xu, B. Woolery, K. Mistry, K. Zhang, S. Jacobs, J. Jopling, W. Yang, B. Lee, T. Raz, M. Mehalel, P. Kolar, Y. Wang, J. Sandford, D. Pivin, C. Peterson, M. DiBattista, S. Pae, M. Jones, S. Johnson, and G.Subramanian, "Erratic fluctuations of SRAM cache vmin at the 90nm process technology node," in *Technical Digest of the International Electron Devices Meeting.* New York, NY, USA: IEEE, December 2005, pp. 655–658.

[21] V. Degalahal, L. Li, V. Narayanan, M. Kandemir, and M. J. Irwin, "Soft errors issues in low-power caches," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 13, no. 10, pp. 1157–1166, October 2005.

[22] P. Roche, G. Gasiot, K. Forbes, V. O'Sullivan, and V. Ferlet, "Comparisons of soft

error rate for SRAMs in commercial SOI and bulk below the 130nm technology node," *IEEE Transactions on Nuclear Science*, vol. 50, no. 6, pp. 2046–2054, December 2003.

[23] P. Hazucha and C. Svensson, "Impact of CMOS technology scaling on the atmospheric neutron soft error rate," *IEEE Transactions on Nuclear Science*, vol. 47, no. 6, pp. 2586–2594, December 2000.

[24] P. Hazucha, K. Johansson, and C. Svensson, "Neutron induced soft errors in CMOS memories under reduced bias," *IEEE Transactions on Nuclear Science*, vol. 45, no. 6, pp. 2921–2928, December 1998.

[25] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, "Modeling of failure probability and statistical design of SRAM array for yield enhancement in nanoscaled CMOS," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 12, pp. 1859–1880, December 2005.

[26] A. J. Bhavnagarwala, X. Tang, and J. D. Meindl, "The impact of intrinsic device fluctuations on CMOS SRAM cell stability," *IEEE Journal of Solid-State Circuits*, vol. 36, no. 4, pp. 658–665, April 2001.

[27] S. V. Kosonocky, A. Bhavnagarwala, and L. Chang, "Scalability options for future SRAM memories," in *International Conference on Solid-State and Integrated Circuit Technology*, October 2006, pp. 689–692.

[28] C. E. Shannon, "The mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, pp. 379–423, 1948.

[29] R. W. Hamming, “Error detecting and error correcting codes,” *The Bell System Technical Journal*, vol. 29, no. 2, April 1950.

[30] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, 2nd ed. Amsterdam, CA: North Holland, 1977.

[31] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, 1st ed. NJ, USA: Prentice Hall, 1995.

[32] H. L. Kalter, C. H. Stapper, J. E. B. Jr., J. DiLorenzo, C. E. Drake, J. A. Fifield, G. A. K. Jr., S. C. Lewis, W. B. van der Hoeven, and J. A. Yankosky, “A 50ns 16Mb DRAM with a 10ns data rate and on-chip ECC,” *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1118–1128, October 1990.

[33] M. Spica and T. M. Mak, “Do we need anything more than single bit error correction (ECC)?” in *Records of the International Workshop on Memory Technology, Design and Testing*, August 2004, pp. 111–116.

[34] C. W. Slayman, “Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations,” *IEEE Transactions on Reliability*, vol. 5, no. 3, pp. 397–404, September 2005.

[35] A. M. Saleh, J. J. Serrano, and J. H. Patel, “Reliability of scrubbing recovery-techniques for memory systems,” *IEEE Transactions on Reliability*, vol. 39, no. 1, pp. 114–122, April 1990.

[36] S. S. Mukherjee, J. Emer, T. Fossum, and S. K. Reinhardt, “Cache scrubbing in microprocessors: myth or necessity?” in *Proceedings of International Symposium on*

Dependable Computing. New York, NY, USA: IEEE Pacific Rim, March 2004, pp. 37–42.

[37] A. Tiwari and K. A. Tomko, "Enhanced reliability of finite-state machines in FPGA through efficient fault detection and correction," *IEEE Transactions on Reliability*, vol. 54, no. 3, pp. 459–467, September 2005.

[38] M. A. Bajura, Y. Boulghassoul, R. Naseer, S. Dasgupta, A. F. Witulski, J. Sondeen, S. D. Stansberry, J. Draper, L. W. Massengill, and J. N. Damoulakis, "Models and algorithmic limits for an ECC-based approach to hardening sub-100nm SRAMs," *IEEE Transactions on Nuclear Science*, vol. 54, no. 4, pp. 935–945, August 2007.

[39] J. Wang, A. Singhee, R. A. Rutenbar, and B. H. Calhoun, "Statistical modeling for the minimum standby supply voltage of a full SRAM array," in *Proceedings of the European Solid State Circuits Conference.* New York, NY, USA: IEEE, September 2007, pp. 400–403.

[40] G. A. Sai-Halasz, M. R. Wordeman, and R. H. Dennard, "Alpha-particle-induced soft error rate in VLSI circuits," *IEEE Journal of Solid-State Circuits*, vol. 17, no. 2, pp. 355–361, April 1982.

[41] L. B. Freeman, "Critical charge calculations for a bipolar SRAM array," *IBM Journal of Research and Development*, vol. 40, no. 1, pp. 119–129, 1996.

[42] Q. Zhou and K. Mohanram, "Gate sizing to radiation harden combinational logic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 1, pp. 155–166, January 2006.

[43] T. Merelle, S. Serre, F. Saigne, B. Sagnes, G. Gasiot, P. Roche, T. Carriere, and M. Palau, "Charge sharing study in the case of neutron induced SEU on 130nm bulk SRAM modeled by 3-D device simulation," *IEEE Transactions on Nuclear Science*, vol. 53, no. 4, pp. 1897–1901, August 2006.

[44] R. Naseer, Y. Boulghassoul, J. Draper, S. Dasgupta, and A. Witulski, "Critical charge characterization for soft error rate modeling in 90nm SRAM," in *Proceedings of the International Symposium on Circuits and Systems.* New York, NY, USA: IEEE, May 2007, pp. 1879–1882.

[45] K. Agarwal and S. Nassif, "The impact of random device variation on SRAM cell stability in sub-90nm CMOS technologies," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 16, no. 1, pp. 86–97, January 2008.

[46] A. J. Bhavnagarwala, S. Kosonocky, C. Radens, Y. Chan, K. Stawiasz, U. Srinivasan, S. P. Kowalczyk, and M. M. Ziegler, "A sub-600mV, fluctuation tolerant 65nm CMOS SRAM array with dynamic cell biasing," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 4, pp. 946–955, April 2008.

[47] B. Zhang, A. Arapostathis, S. Nassif, and M. Orshansky, "Analytical modeling of SRAM dynamic stability," in *Proceedings of the International Conference on Computer-Aided Design.* New York, NY, USA: ACM/IEEE, November 2006, pp. 315–322.

[48] P. Roche and G. Gasiot, "Impacts of front-end and middle-end process modifications

on terrestrial soft error rate," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 382–396, September 2005.

[49] C. Heegard and A. E. Gamal, "On the capacity of computer memory with defects," *IEEE Transactions on Information Theory*, vol. 29, no. 5, pp. 731–739, September 1983.

[50] T. M. Cover and J. A. Thomas, *Elements of Information Theory.* New York, NY, USA: John Wiley, 1991.

[51] S. Lin and D. Costello, Jr., *Error Control Coding: Fundamentals and Applications.* Prentice Hall, Englewood Cliffs, NJ., 1983.

[52] P. Grover and A. Sahai, "Green codes: Energy-efficient short-range communication," in *Proceedings of the International Symposium on Information Theory.* New York, NY, USA: IEEE, July 2008, pp. 1178–1182.

[53] H. Qin, A. Kumar, K. Ramchandran, J. Rabaey, and P. Ishwar, "Error-tolerant SRAM design for ultra-low power standby operation," in *Proceedings of International Symposium on Quality Electronic Design.* New York, NY, USA: IEEE, March 2008, pp. 30–34.

[54] J. A. Zoutendyk, L. D. Edmonds, and L. S. Smith, "Characterization of multiple-bit errors from single-ion tracks in integrated circuits," *IEEE Transactions on Nuclear Science*, vol. 36, no. 6, pp. 2267–2274, December 1989.

[55] K. M. Warren, R. A. Weller, M. H. Mendenhall, R. A. Reed, D. R. Ball, C. L. Howe, B. D. Olson, M. L. Alles, L. W. Massengill, R. D. Schrimpf, N. F. Haddad, S. E.

Doyle, D. McMorrow, J. S. Melinger, and W. T. Lotshaw, "The contribution of nuclear reactions to heavy ion single event upset cross-section measurements in a high-density SEU hardened SRAM," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2125–2131, December 2005.

[56] R. C. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 305–316, September 2005.

[57] R. J. Evans and P. D. Franzon, "Energy consumption modeling and optimization for SRAMs," *IEEE Journal of Solid-State Circuits*, vol. 30, no. 5, pp. 571–579, May 1995.

[58] S. Roundy, B. Otis, Y. H. Chee, J. Rabaey, and P. Wright, "A 1.9 GHz RF transmit beacon using environmentally scavenged energy," in *Proceedings of the International Symposium on Low Power Electronics and Design.* New York, NY, USA: IEEE, 2003.

[59] G. F. Jr., "Generalized minimum distance decoding," *IEEE Transactions on Information Theory*, vol. 12, no. 2, pp. 125–131, April 1966.

[60] P. K. Veenstra, F. P. M. Beenker, and J. J. M. Koomen, "Testing of random access memories: theory and practice," *IEE Proceedings of Circuits, Devices and Systems*, vol. 135, no. 1, pp. 24–28, February 1988.

[61] A. J. van de Goor, *Testing semiconductor memories: theory and practice.* New York, NY, USA: John Wiley & Sons, 1991.

[62] R. Durrett, *Probability: Theory and Examples*, 2nd ed. Belmont, CA: Duxbury Press, 1996.

[63] R. Y. Rubinstein, *Simulation and the Monte Carlo Method.* New York, NY, USA: John Wiley & Sons, 1981.

[64] Z. Guo, A. Carlson, L. T. Pang, K. Duong, T. J. King, and B. Nikolic, “Large-scale read/write margin measurement in 45nm CMOS SRAM arrays,” in *Proceedings of the IEEE Symposium on VLSI Circuits*, June 2008, pp. 42–43.

[65] A. A. Balkema and L. De Haan, “Residual life time at great age,” *The Annals of Probability*, vol. 2, no. 5, pp. 792–804, Oct 1974.

[66] J. Pickands III, “Statistical inference using extreme order statistics,” *The Annals of Probability*, vol. 3, no. 1, pp. 119–131, Jan 1975.

[67] A. Singhee and R. A. Rutenbar, “Statistical blockade: A novel method for very fast Monte Carlo simulation of rare circuit events, and its application,” in *Proceedings of the Design, Automation and Test in Europe Conference.* New York, NY, USA: IEEE, April 2007, pp. 1–6.

[68] P. J. Bickel and K. A. Doksum, *Mathematical Statistics Vol I.* Upper Saddle River, NJ, USA: Prentice Hall, 2001.

[69] A. W. van der Vaart, *Asymptotic Statistics.* Cambridge, UK: Cambridge University Press, 1998.

[70] K. M. Cao, W. C. Lee, W. Liu, X. Jin, P. Su, S. K. H. Fung, J. X. An, B. Yu, and C. Hu, “BSIM4 gate leakage model including source-drain partition,” in *Technical Digest of the International Electron Devices Meeting.* New York, NY, USA: IEEE, December 2000, pp. 815–818.

[71] W. K. Huang, Y. Shen, and F. Lombardi, "New approaches for the repairs of memories with redundancy by row/column deletion for yield enhancement," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 323–328, March 1990.

[72] W. W. Peterson and E. J. W. Jr., *Error-Correcting Codes.* Cambridge, MA, USA: MIT Press, 1978.

Chapter 7

Appendix

The proof is divided into two parts. Under the stated outage criterion, the largest asymptotically feasible (k/n) ratio is not known. Thus, an upper bound and a lower bound on the smallest feasible $\mathcal{P}(v_S)$ will be derived. The lower bound on asymptotically optimal $\mathcal{P}(v_S)$ is derived using information theoretic capacity, and the lower bound is derived using an asymptotic achievable strategy with bounded distance decoding based codes.

7.1 Lower bound derivation using channel coding theorem

First, the (channel) capacity of the SRAM cell will be computed. In standby mode,

$$\begin{aligned} Y &= S, \quad \text{with probability} \quad p_h(v_S), && (7.1) \\ &= X, \quad \text{otherwise}, && (7.2) \end{aligned}$$

where $S \in \{0,1\}$ is an equally likely binary stuck-at state, and v_S is the standby voltage. The capacity for this cell will be a function of v_S through $p_h(v_S)$. The probability $p_h(v_S)$

was found by measurements as highlighted in Chapter 5. The mutual information between input X and output Y is given by,

$$I(X;Y) \quad = \quad H(Y) - H(Y|X). \tag{7.3}$$

The information theoretic capacity is found by maximizing the mutual information $I(X;Y)$ over the choice of input distribution (or in this case over $\mathbb{P}[X=0]$). The conditional entropy $H(Y|X)$ will be computed using the conditional probability $\mathbb{P}[Y=y|X]$.

$$\begin{aligned}
\mathbb{P}[Y=1|X=1] &\overset{(a)}{=} \mathbb{P}[Y=1, S=1|X=1]\mathbb{P}[S=1] + \mathbb{P}[Y=1, S=0|X=1]\mathbb{P}[S=0], \\
&\overset{(b)}{=} \frac{1}{2}\left(\mathbb{P}[Y=1, S=1|X=1] + \mathbb{P}[Y=1, S=0|X=1]\right), \\
&\overset{(c)}{=} \frac{1}{2}(1 + (1 - p_h(v_S))), \\
&= 1 - \frac{p_h(v_S)}{2},
\end{aligned} \tag{7.4}$$

where (a) follows by the total probability rule, (b) follows since $\mathbb{P}[S=0] = \mathbb{P}[S=1] = (1/2)$, and (c) follows by (7.1) and (7.2). Similarly,

$$\mathbb{P}[Y=0|X=0] = 1 - \frac{p_h(v_S)}{2}. \tag{7.5}$$

Due to symmetry, $H(Y|X=0)$ will be equal to $H(Y|X=1)$, and therefore,

$$H(Y|X) = H_2\left(p_h(v_S)/2\right), \tag{7.6}$$

where $H_2(p)$ denotes the binary entropy function. The mutual information is given by,

$$I(X;Y) = H(Y) - H_2(p_h(v_S)/2). \tag{7.7}$$

The information theoretic capacity is obtained by maximizing $H(Y)$. Hence,

$$C(v_S) := \max_{\mathbb{P}[X=0]} I(X;Y) = 1 - H_2\left(p_h(v_S)/2\right). \tag{7.8}$$

The last equality follows since $\mathbb{P}[X = 0] = 1/2$ results in the maximization of $H(Y)$ in (7.3). Thus, the information theoretic capacity for the standby SRAM cell model is given by,

$$C(v) = 1 - H_2(p_h(v_S)/2). \tag{7.9}$$

Recall that while reducing standby (leakage) power, the outage probability has to be kept negligible so that a negligible fraction of decoded bits are in error. The outage set $\mathcal{E}$ was given by,

$$\mathcal{E} = \bigcup_{i=0}^{2^k} \mathcal{E}_i, \text{ where,} \tag{7.10}$$

$$\mathcal{E}_i = \left\{ g\left(Y_1^n\right) \neq i | X_1^n = f(i) \right\}, \tag{7.11}$$

where integers from 1 to 2^k are used to index all the input words in $\mathbb{B}^k$ that can be stored in n SRAM cells. The outage probability was defined as,

$$p_{\text{outage}} = \mathbb{P}(\mathcal{E}), \tag{7.12}$$

where the probability was taken over DRV (or $p_h(v_S)$) and S distributions. For comparison, consider the following (well studied) decoding error probabilities,

$$p_{\text{avg}} = \frac{1}{2^k} \sum_{i=1}^{2^k} \mathbb{P}[\mathcal{E}_i], \tag{7.13}$$

and,

$$p_{\max} = \max_{1 \leq i \leq 2^k} \mathbb{P}[\mathcal{E}_i], \tag{7.14}$$

According to the channel capacity theorem, for any v_S and an arbitrary $\delta_1 > 0$, a coding scheme exists which achieves a rate of $(k/n) = C(v_S) - \delta_1$ such that $p_{\max}$ (or p_{avg}) tends to 0 as $n \to \infty$ [50, Chapter 8]. In the context of SRAM, since the cells are randomly realized, on average a fraction of p_{outage} encoded blocks will have some input B_1^k such that the decoding operation is unsuccessful. Note that this outage criterion is more stringent than the other decoding error probabilities, since it is easy to show that (because $\mathcal{E} \supseteq \mathcal{E}_i$),

$$p_{\text{outage}} \geq p_{\max} \geq p_{\text{avg}} \tag{7.15}$$

Thus, no matter what coding scheme is picked, for an asymptotically negligible outage probability, the best possible storage efficiency cannot be more than $C(v_S) - \delta_1$ ($\delta_1 > 0$ is arbitrary). Note that p_{outage} may be strictly larger than $p_{\max}$, therefore, it is not easy to show that a storage efficiency of $C(v_S) - \delta_1$ can indeed be achieved for arbitrary $\delta_1 > 0$. [1] Thus, the information theoretic capacity serves as an upper bound on the storage efficiency (k/n). Then the lower bound on *power per bit* is given by,

$$\mathcal{P}_\epsilon(v_S) \geq \frac{Gv_S^2}{1 - H_2(p_h(v_S)/2)}, \tag{7.16}$$

where $p_{\text{outage}} < \epsilon$. Since ϵ can be made arbitrarily small as $n \to \infty$, therefore,

$$\mathcal{P}(v_S) \geq \frac{Gv_S^2}{1 - H_2(p_h(v_S)/2)}. \tag{7.17}$$

7.2 Upper bound derivation using the Gilbert bound

In this section, an upper bound on the *power per bit* will be derived. Only bounded distance decoding based codes are considered. Codes that decode within a minimum Hamming distance are classified under this category, e.g., Reed-Muller codes [30, 51].

Let $[n, k, d]$ be a general bounded-distance decoding code. As $n, d \to \infty$ with (d/n) converging to a non-zero fraction, approximately $(d/2)$ errors can be corrected for

[1] Channel coding theorem states that $(k/n) = C(v_S) - \delta_1$ can be achieved while $p_{\max}$ goes to zero [50].

any choice of input codeword. Thus, the outage event for this class of code gets simplified to a Hamming distance criterion which can be analyzed. Let $DRV_1, DRV_2, \ldots, DRV_n$ be the n realized DRV values. Let $DRV_{(1)}, DRV_{(2)}, \ldots, DRV_{(n)}$ be the sorted DRV values with $DRV_{(n)}$ being the largest. Further, note that $u = \lfloor \frac{d-1}{2} \rfloor$ errors can be corrected by this code. The outage probability simplifies to,

$$\epsilon = \mathbb{P}[DRV_{(n-u)} \geq v_S]. \tag{7.18}$$

For this class of code, for a given (n, d) pair and an arbitrary $\delta_2 > 0$, the best asymptotic rate is given by the Gilbert bound [72],

$$\frac{k}{n} = 1 - H_2\left(\frac{d}{n}\right) - \delta_2 \tag{7.19}$$

By (strong) law of large numbers, a fraction $p_h(v_S)$ of SRAM cells will exhibit hold-failure at supply voltage v_S. Thus, for large n and arbitrary $\delta_3 > 0$, almost surely no more than $n(p_h(v_S)+\delta_3)$ errors will be present. The constant δ_3 decreases to 0 as n increases to infinity. Thus a code with $d = 2n(p_h(v_S) + \delta_3) + 1$ will have a negligible outage as n increases. For this minimum distance d, by the Gilbert bound, the following (k/n) ratio is achievable for negligible outage,

$$\frac{k}{n} = 1 - H_2\left(2p_h(v_S) + (2\delta_3 + 1/n)\right) - \delta_2. \tag{7.20}$$

As n is made large, δ_2, $1/n$, δ_3, and the outage probability converge to zero. Thus for any v_S the following *power per bit*is achievable using bounded distance decoding codes,

$$\mathcal{P}(v_S) \leq \frac{Gv_S^2}{1 - H_2(2p_h(v_S))}. \tag{7.21}$$

Thus the proof is complete. □