5-2015

Exploiting Application Behaviors for Resilient Static Random Access Memory Arrays in the Near-Threshold Computing Regime

Dieudonne Manzi Mugisha
Utah State University

EXPLOITING APPLICATION BEHAVIORS FOR RESILIENT STATIC RANDOM ACCESS MEMORY ARRAYS IN THE NEAR-THRESHOLD COMPUTING REGIME

by

Dieudonne Manzi Mugisha

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Engineering

Approved:

Dr. Sanghamitra Roy Major Professor	Dr. Koushik Chakraborty Committee Member
Dr. Zeljko Pantic Committee Member	Dr. Mark R. McLellan Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2015

Abstract

Exploiting Application Behaviors for Resilient Static Random Access Memory Arrays in the Near-Threshold Computing Regime

by

Dieudonne Manzi Mugisha, Master of Science

Utah State University, 2015

Major Professor: Dr. Sanghamitra Roy
Department: Electrical and Computer Engineering

Near-Threshold Computing embodies an intriguing choice for mobile processors due to the promise of superior energy efficiency, extending the battery life of these devices while reducing the peak power draw. However, process, voltage, and temperature variations cause a significantly high failure rate of Level One cache cells in the near-threshold regime—a stark contrast to designs in the super-threshold regime, where fault sites are rare.

This thesis work shows that faulty cells in the near-threshold regime are highly clustered in certain regions of the cache. In addition, popular mobile benchmarks are studied to investigate the impact of run-time workloads on timing faults manifestation. A technique to mitigate the run-time faults is proposed. This scheme maps frequently used data to healthy cache regions by exploiting the application cache behaviors. The results show up to 78% gain in performance over two other state-of-the-art techniques.

(51 pages)

Public Abstract

Exploiting Application Behaviors for Resilient Static Random Access Memory Arrays in the Near-Threshold Computing Regime

by

Dieudonne Manzi Mugisha, Master of Science
Utah State University, 2015

Major Professor: Dr. Sanghamitra Roy
Department: Electrical and Computer Engineering

Historically, the increase of transistor devices per area of chip has been one of the key drivers of microprocessor performance. While the transistor count per chip keeps increasing, today's microprocessor power budget limits the number of devices that can be turned on. This issue stems from the fact that new device technologies dissipate more power in form of heat—a problem that can cause circuit breakdown due to excessive temperatures.

Near-Threshold Computing (NTC), where the operating voltage is reduced near the threshold voltage that turns on a transistor, is a popular research topic. NTC significantly reduces the power consumption and allows a higher number of devices to be used given a certain power budget. Although it is energy efficient, NTC suffers from a high rate of circuit failures induced by both manufacturing imperfections and the operating environment. *Static Random Access Memory* (SRAM) arrays are the most affected components of a microprocessor, and consequently require robust design techniques. This thesis proposes a technique that exploits the software use of SRAM arrays in order to reduce circuit failure in NTC.

This thesis is dedicated to my family that has always encouraged me in all my endeavors.

Acknowledgments

I deeply thank Dr. Sanghamitra Roy, my academic advisor over the past two years. I believe that her constant effort to make me a better researcher and her motivation have been driving forces behind this thesis and my other published works. Equally important has been her financial funding for all the time I have been under her advisement. I would like to express my gratitude to Dr. Koushik Chakraborty for his constant push to make me a more critical thinker in research. He not only helped in evaluating my project ideas, he also helped in organizing my research in a more presentable format. I would like to thank Dr. Zeljko Pantic for his feedback as a committee member. I am deeply grateful for his will to put up with my schedule which had at time unexpected changes.

My appreciation goes to all the BRIDGE lab students who ever worked with me. Hu Chen helped me in multiple research projects and we co-authored a number of publications. Dr. Yiding Han and Dr. Dean Ancajas were always more than happy to help me with the lab tools when I had no idea what to do. I would also like to thank Rajesh JS, Chidham, Shamik Saha, Harshitha Pulla, Kurt Brenning, Shayan Taheri, Prabal Basu, and every student who helped me in my research projects and reviewed my paper drafts for conference submissions.

I am deeply grateful to the ECE department for admitting me to pursue my master's program and offering financial support. I would like to acknowledge Dr. Todd Moon for his flexibility regarding the program of study and for his many updates regarding career opportunities. My gratitude goes to Mary Lee Anderson who has always advised me on all the requirements of the department and responded to all my inquiries. If it were not for her, I would not have completed or been aware of all the necessary steps to complete my program. I would also like to thank Tricia Brandenburg, all the ECE department staff, and everyone who ever helped me in completing all the administrative tasks.

Dieudonne Manzi Mugisha

Contents

List of Tables

Table Page

List of Figures

Chapter 1

Introduction

Mobile architectures and their applications are growing rapidly as society embraces ubiquitous computing. One of the characteristics of mobile processors is a limited power budget due to the battery capacities of mobile devices [1]. Multiple works in this domain have targeted improving the energy efficiency of the system through a slew of circuit and architectural techniques. Recently, advances in device engineering have uncovered a new direction of energy efficient operations through *Near-Threshold Computing* (NTC) [2–4]. Given the extreme importance of energy efficiency in mobile platforms, it is imperative to understand the challenges and opportunities to employ NTC processors for mobile computing.

An NTC processor operates with a supply voltage, V_{dd}, at or near the threshold voltage, V_{th}—an energy optimal operating point with substantially better performance over subthreshold computing. NTC offers the promise of superior energy efficiency by extending the battery life of mobile devices and reducing the peak power draw. While the energy efficiency benefits can scale up to an order of 10X, there is also a substantial performance degradation. Recent works [2, 5] have thus focused on recovering such performance degradation of NTC through parallel computing. Techniques explored in these works are highly applicable for the mobile application domain. Therefore, it is expected that NTC processors are well poised to become a viable alternative for mobile platforms.

Another central design challenge for NTC mobile processors stems from *Process, Voltage, and Temperature* (PVT) variations. The delay variation due to PVT in NTC can be several times larger than that in *Super-Threshold Computing* (STC) [6, 7]. In particular, memory components, such as caches and register files, are more susceptible to delay variation from PVT. These structures, typically built with SRAM arrays, contain a large number of parallel paths with shallow logic depths. Therefore, different regions of a cache designed for NTC will show a wide variability in delay characteristics—significantly more than seen in caches operating in the super-threshold regime. Consequently, setting delay margins below worst or near worst delay in caches will have prohibitive design costs in an NTC mobile processor. On the other hand, a reasonable delay or operating frequency of these caches will have many more timing errors at run-time—a paradigm shift from current design practices, where timing errors in the caches are rare. A key research question in this context is *how can one design resilient next generation caches that can operate efficiently in the presence of several regions with timing faults?*

This work pursues this research question and explores the effect of PVT on *Level One* (L1) caches in mobile processors. Using an extensive circuit-architectural methodology, this work shows that in a cache with many faulty lines, the dynamic manifestation of an error is strongly dependent on the workload and has a predictable occurrence. These findings are exploited to propose a low overhead technique to minimize the penalty of faulty cache lines.

1.1 Contribution

This work makes the following contributions.

- Using detailed cache models at the 32 nm technology node, this work demonstrates that NTC caches have substantially higher PVT-induced faults than STC caches (Chapter 3).
- Chapter 3 investigates the typical cache access patterns of popular mobile applications [8]. It is found that the dynamic manifestation of timing errors for these applications in NTC caches is predictable.
- Chapter 4 proposes a low-overhead technique to effectively reduce the occurrence of timing errors even in the presence of several faulty cache lines.
- Using a rigorous circuit-architectural methodology, the collected data in Chapter 6 shows that the proposed technique is up to 47% and 78% more effective than two state-of-the-art techniques in reducing PVT-induced performance loss.

Chapter 2

Background

This chapter discusses the relationship between technology scaling and microprocessor power consumption in Section 2.1. Section 2.2 presents NTC, an energy efficient computing domain that promises to solve the power dissipation issue introduced by aggressive technology scaling. Section 2.2.1 and Section 2.2.2 point out the challenges observed in NTC design, especially SRAM array functional failures. Section 2.3 ends the chapter by presenting existing techniques to tackle SRAM functional failures in NTC.

2.1 Technology Scaling and Its Impact on Power

Steady advances in technology have allowed continuous transistor scaling. This scaling resulted in an increase in the number of transistors packed on a chip area. An associated effect has been the increase of processor capabilities due not only to the number of transistors per area but also faster circuits. Smaller transistors generally switch faster because of less resistance and a smaller number of electrons required to excite the formation of the current channel.

Table 2.1 shows two types of technology scaling that have been used historically. *Constant-voltage scaling* shrunk the physical dimensions but kept the voltage constant, whereas *Dennard scaling* [9] scaled both device dimensions and the voltage by the same factor. From 6 μm to 1 μm technology node, constant-voltage scaling fixed V_{dd} to 5 V from one technology to the next, whereas the transistor dimensions were reduced by a factor of $S = \sqrt{2}$. This boosted the performance as the device delay was quadratically improved. As the channel length L decreased with newer technologies, the electric field E increased according to Equation (2.1) where V_{ds} is the potential difference between the drain and the source. At 1 μm, device breakdown became a potential threat due to very high electric fields.

$$E = \frac{V_{ds}}{L} \tag{2.1}$$

Dennard scaling law was adopted mainly to reduce the electric field. However, as shown in Table 2.1, there were other advantages associated with it such as the quadratic reduction in power consumption. An even more important advantage is power density. Equation (2.2) illustrates a reduction of up to S^3 in power density resulting from scaling of the capacitance C_0, the supply voltage V_{dd_0}, the frequency f_0, transistor width W_0, and transistor length L_0. This decrease in power density has been a critical advantage in sub-micron nodes. However, under 90 *nm* Dennard scaling started to face serious barriers and the power density of the chip reached new levels [10].

Table 2.1: Impact of scaling on transistors.

Scaling parameters			
Parameter	Sensitivity	Dennard scaling	Constant voltage scaling
Length: L		$1/S$	$1/S$
Width: W		$1/S$	$1/S$
Supply voltage: V_{DD}		$1/S$	1
Threshold voltage: V_{th}		$1/S$	$1/S$

Resulting characteristics			
Parameter	Sensitivity	Dennard scaling	Constant voltage scaling
Resistance: R	V_{DD}/I_{ds}	$1/S$	**1**
Capacitance: C	WL/t_{ox}	$1/S$	$1/S$
Delay: τ	RC	$1/S$	$1/S^2$
Frequency: f	$1/\tau$	S	S^2
Switching power dissipation: P	CV_{DD}^2f	$1/S^2$	S
Area per gate: A	WL	$1/s^2$	$1/S^2$
Switching power density	P/A	1	S^3

$$\begin{aligned} P/A &= \frac{CV_{dd}^2 f}{WL} \\ &= \frac{(\frac{1}{S} \times C_0) \times (S \times V_{dd_0})^2 \times (S \times f_0)}{(\frac{1}{S} \times W_0) \times (\frac{1}{S} \times L_0)} \\ &= \frac{(P/A)_0}{S^3} \end{aligned} \tag{2.2}$$

The main problems that increase the power in modern processors are the increased portion of leakage current in processes with dimension sizes below 90 nm and a very high transistor per chip ratio. Figure 2.1 shows two main sources of leakage in an n-type device: sub-threshold leakage (I_{sub}) and gate leakage (I_{gate}). As V_{dd} decreases from one technology node to the next, V_{th} decreases as well. Eventually, a low V_{th} prevents the gate from turning the drain-to-source current off—a phenomenon that increases I_{sub}. For I_{gate}, the oxide layer becomes too thin and increases electrons tunneling through the oxide. As a result, I_{gate} increases. The sum of these leakages multiplied by millions or billions of transistors per chip results in an excessive on-chip power density and temperature. Furthermore, the issue is complicated by the lack of proper cooling systems.

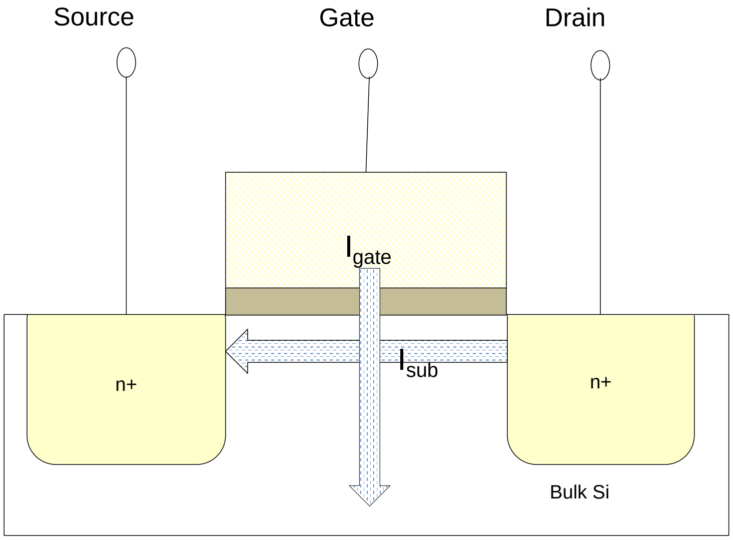


Fig. 2.1: Leakage current paths.

While the number of transistor per chip keeps increasing, the performance does not increase at the same rate due to the discussed power barriers. Figure 2.2, adapted from Hennessy and Patterson [11], shows the frequency increase over the years for a uniprocessor core. The figure shows that there were almost two decades when the performance increase per year was 52% on average. However, after 2002, the rate of increase dropped down to 22%. This can be attributed to multiple factors such as the processor-memory speed gap, high power consumption, and the limit of available instruction level parallelism. However, even if it were possible to bridge the processor-memory gap and have enough instruction level parallelism, power density and heat build-up remain formidable challenges for single-core processors. Therefore, techniques to reduce the power consumption of microprocessors are needed now more than ever.

Dynamic voltage scaling, where the supply voltage can be reduced to 70% of its nominal value, has been a popular alternative for reducing the power consumption. The success of this technique stems from the fact that dynamic energy and static energy, respectively, reduce quadratically and linearly with the supply voltage [10]. Another alternative is sub-threshold computing where V_{dd} is lowered below V_{th}. While the dynamic energy is the lowest in the sub-threshold region, the gate delay in this region increases exponentially. Sub-threshold computing is therefore not useful for mid and high performance applications.

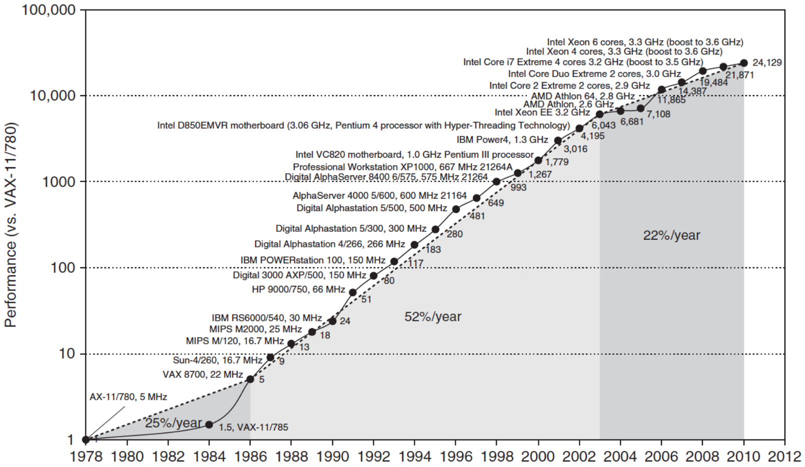


Fig. 2.2: Uniprocessor performance growth.

Similarly, the leakage in the sub-threshold regime increases exponentially. At a certain point, the leakage power overshadows all benefits from voltage scaling. The key to harness voltage scaling benefits is to find the voltage at which the energy per operation is the lowest. Section 2.2 explores a new regime that harnesses voltage scaling to obtain higher energy efficiency than super-threshold and sub-threshold computing.

2.2 Near-Threshold Computing

Near-Threshold Computing, where the on-chip supply voltage is slightly higher than the threshold voltage, promises a near-optimum energy point. This seems counter-intuitive considering that the dynamic energy is the lowest in the sub-threshold regime. However, as shown in Figure 2.3, NTC has the lowest energy once both leakage and dynamic energies are considered. The figure was obtained by using *H-Simulation Program with Integrated Circuit Emphasis* (HSPICE) for a chain of 31 *Fan-Out of Four* (FO4) inverters in a 32 nm predictive technology [12]. The threshold voltage is 0.28 V while NTC region is defined to be between 0.3 V and 0.5 V.

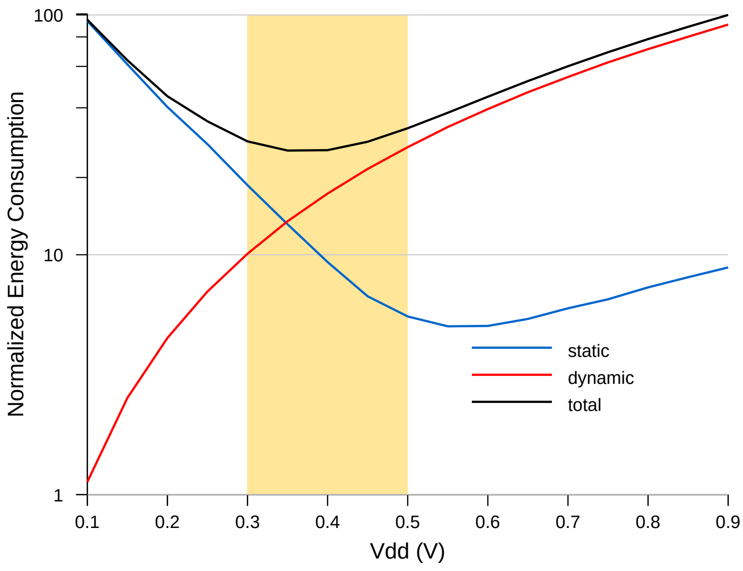


Fig. 2.3: HSPICE energy breakdown for a chain of 31 FO4 in a 32 nm technology node.

The high leakage in the sub-threshold regime is due to the considerable increase of the gate delay. The delay results from the driving current, I_{ds}, that exponentially decreases with V_{dd} as shown in Equation (2.3). In contrast, in STC, the driving current is only a linear function of V_{dd}. On a microarchitectural level, the net effect is an increased computation time in NTC, and consequently increased energy consumption. On the other hand, STC enables less computation times at the cost of high dynamic and total energy consumptions.

$$I_{ds} \alpha I_{th}(1 - e^{\frac{-V_{dd}}{v_{th}}}) \tag{2.3}$$

Due to its high energy efficiency, many research works have been attracted to the use of NTC in applications where battery life is a very important design consideration such as personal and mobile computing [6, 13]. Other works have explored how to fit more processing cores in a power budget. As NTC reduces energy consumption by 10-50X while only suffering 5-10X in performance degradation, 2-5X more cores can fit in a power budget [14–18].

2.2.1 Near-Threshold Computing Challenges

Although NTC is significantly efficient in reducing energy consumption, there are several obstacles to its commercial widespread adoption. The main challenges pertaining to reliability are the increased delay variations and increased circuit failures. When operating at near-threshold voltage, the gate delay becomes more sensitive to variations. Figure 2.4 and Figure 2.5 show the impact of V_{dd} and temperature variations on the gate delay in both STC and NTC. The figures are obtained from Equation (2.4) and Equation (2.5) with the following parameters: STC access time (T_{STC}), NTC access time (T_{NTC}), supply voltage (V_{dd}), transistor effective length (L_{eff}), thermal voltage (v_t), threshold voltage (V_{th}), and mobility (μ). Note that the thermal voltage and mobility are functions of the operating temperature. As it can be observed from the figures, the gate delay variation is considerably high in NTC compared with STC. Due to this increased sensitivity to variations, circuits of the same type are bound to yield very different performances. One of the existing solutions for multi-core processors is to operate different cores within a chip at different clock rates [2, 13, 19].

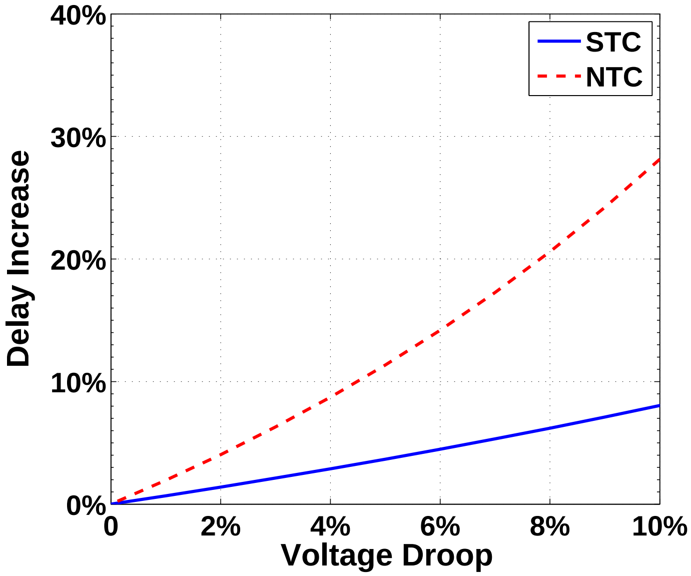


Fig. 2.4: Impact of voltage variation in NTC compared with STC.

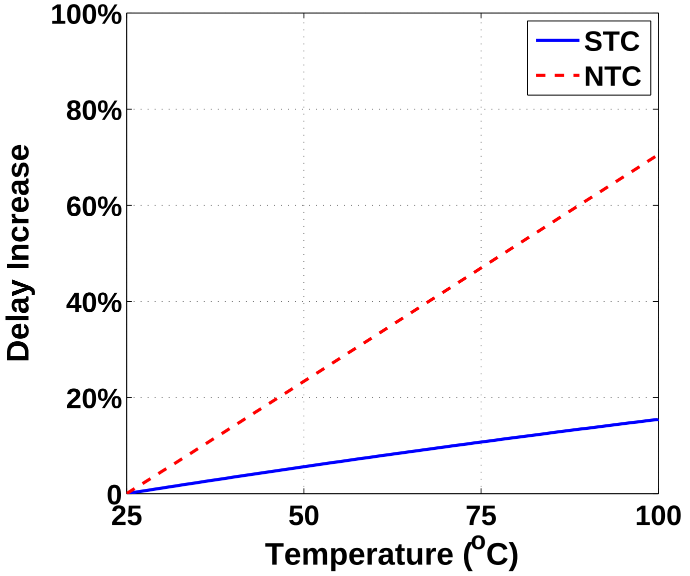


Fig. 2.5: Impact of temperature variation in NTC compared with STC.

$$T_{STC} \propto \frac{V_{dd} \times L_{eff}}{\mu(V_{dd} - V_{th})^2} \tag{2.4}$$

$$T_{NTC} \propto \frac{V_{dd}}{\mu \times L_{eff} \times v_t^2 \times ln^2(e^{\frac{V_{dd}-V_{th}}{2\times n\times v_t}} + 1)} \tag{2.5}$$

However, even with frequency relaxation, SRAM cells require careful design considerations as they use miniaturized transistor devices, which are more susceptible to variations. The next section will discuss in detail how SRAM cells are affected by process variations in the near-threshold regime.

2.2.2 Static Random Access Memory

SRAM blocks can occupy up to 70% of the chip area [10]. For that reason, they are built with minimum size transistors for a given technology. The small size of the devices impairs the ability to precisely control the fabrication process [20]. This results in devices with length, oxide thickness, and V_{th} values different from nominal ones. These parameter variations can be divided in two categories—systematic variations and random variations [20]. For systematic variations, the shift in a parameter of one transistor depends on the parameters of neighboring transistors. For random variations, the shift of a transistor parameter is independent of the parameters of neighboring transistors. The random variations are mainly due to dopant fluctuations that eventually cause a shift in the threshold voltage [21, 22].

The effect of devices with parameters varying from the nominal values is the deviation from expected device behavior. For instance, a threshold voltage that is higher than the nominal one will cause the transistor to be slower than expected. As combinational circuits often have a large number of transistors on their paths, the positive and negative variations cancel each other out. However, in the case of SRAM cells, if one transistor is weak, the correct operations can be impaired because of shallow paths in SRAM arrays. In a low power regime, such as NTC, the probability of failure increases further as the over-drive voltage is reduced.

Figure 2.6 illustrates the structure of the standard *Six-Transistor* (6T) SRAM cell. For successful operations and robustness, the devices within a cell do not have the same strength. Pull-up transistors (*P1* and *P2*) are stronger than access transistors (*A1* and *A2*). At the same time, access transistors are stronger than pull-down transistors (*D1* and *D2*). Due to process variations, this order of strength can change, which could cause functional failures. There are four main types of failures induced by process variations—destructive read, write failure, hold failure, and access time failure [20]. These failures are next discussed in detail.

- **Read failure**: Figure 2.6 shows a 6T cell storing a value of one. In its steady state, the logic values at node Q and node Q_b are one and zero respectively. For a read operation, the bit-lines *bit* and *bit_b* are precharged to a high value ($\geq V_{DD}/2$). Next, the word-line, WL, is raised high. Consequently, *bit_b* is pulled down to the ground through access transistor $A2$ and driver transistor D2. The current flowing through node Q_b tends to rise the voltage V_{Q_b}. Usually, the driver device (*D1*) is sized to be

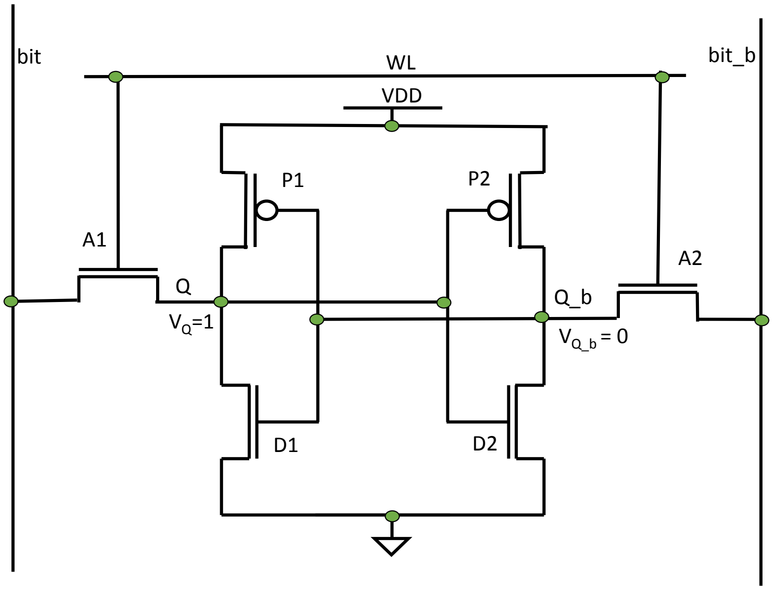


Fig. 2.6: 6T SRAM cell.

stronger than the access transistor (*A2*) so that node Q_b is held at a voltage, V_{read}, less than the flipping threshold of inverter $P1/D1$. Due to parameter variations, $A2$ can become stronger or $D2$ weaker. In this case, the values of Q and Q_b flip, which corrupts the value stored in the cell.

- **Write failure**: To write a value of zero in a cell storing a one (Figure 2.6), *bit* is pulled down by a write driver while *bit_b* is precharged high by the bit-line conditioning circuitry. Next, the word-line is raised high. Consequently, node Q is discharged through *bit*. In the same time node Q_b (the output of Q) is flipped to one. The new stored value remains stable as long as access transistor *A1* is stronger than pull-up transistor *P1*. In practice, *P1*, which is originally switched on, will oppose node Q from being pulled down. To avoid this effect, P1 and A1 are usually sized in a way that *A1* is stronger than *P1*. However, due to parameter variations, *P1* can become stronger than the access transistor. If, because of variations, *P1* is stronger enough to keep V_Q at a voltage higher than the switching voltage of inverter $P2/D2$, the write will never be successful.

- **Access time failure**: The access time failure happens when the time required to produce a certain differential voltage, V_{diff}, between two bit-lines is violated. For instance in Figure 2.6, if the access transistor *A2* and/or pull-down transistor *D2* happen to have higher threshold voltages, it takes longer to pull down bit_b.

- **Hold failure**: During hold time when there is no read or write, the supply voltage is reduced to decrease the static energy consumption. The reduced voltage is usually high enough to keep the node (Q in Figure 2.6) storing logic one stable. However, due to V_{th} variations, the pull-down transistor *D1* can have a much higher threshold voltage resulting in leakage current. This leakage can bring the voltage in node Q to a lower value than the switching voltage of the inverter $P2/D2$. The same undesired effect can happen if the threshold voltage of transistor *P1* is too low.

As discussed in Section 2.2.1, the gate delay in NTC is more susceptible to PVT variations than in STC. This implies that the failure rate in NTC is expected to be higher. The access failure is the most affected due to the reduction in current of the access transistors. Due to this increase of SRAM fault rates, especially in NTC, it is necessary to find proper measures to mitigate the effect of process variations. Section 2.3 discusses some of the existing works studying the impact of PVT in both STC and NTC.

2.3 Related Work

Multiple works have tackled SRAM failures in NTC using alternative device technologies, circuit, and architectural techniques. This section highlights the advantages and disadvantages introduced by those techniques.

- **Alternative SRAM cells**: Due to the read and write conflicting constraints and the high susceptibility to process variations, the use of conventional 6T SRAM cell in the sub or near-threshold regime is almost impossible. Many works [23–26] designed alternative SRAM cells to tackle different issues encountered in the sub and near-threshold voltage regime. Chang et al. [25] designed an *Eight-Transistor* (8T) cell to decouple the read from the write. As shown in Figure 2.7, two additional transistors *D3* and *D4* are added. These additional devices are used for the read operation. This increases the optimization space since read and write devices can be sized independently, which allows read stability and write operations at low voltages. However, even in the presence of an 8T transistor, there is no clear way to overcome access delay variation other than reducing timing constraints—an option that considerably deteriorates the performance [6]. Calhoun and Chandrakasan [26] proposed a ten-transistor cell. In addition to maintaining read, write, and hold stability at sub-threshold voltages, this cell reduces the leakage compared with an 8T cell. However, it requires relaxed timing constraints because of delay variations.

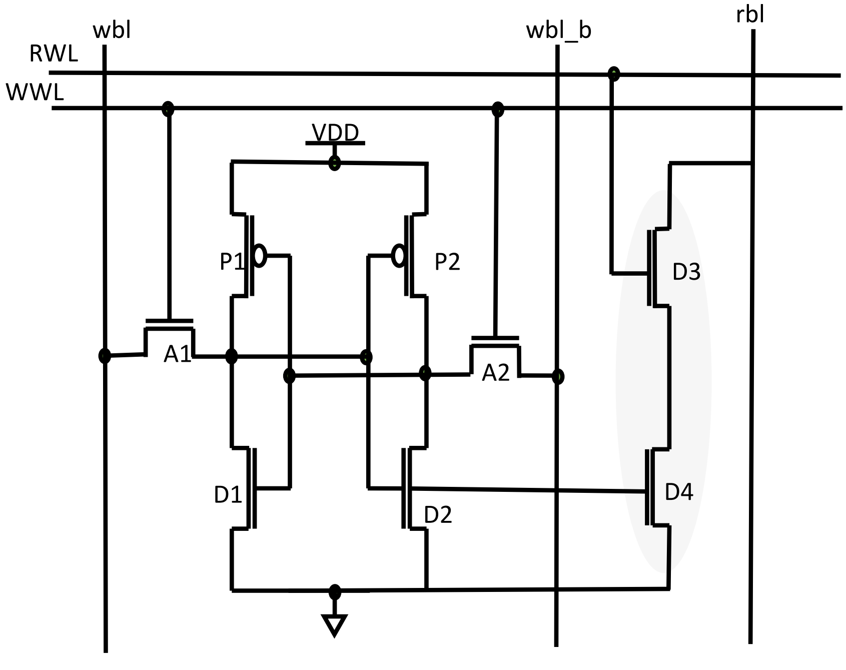


Fig. 2.7: 8T SRAM cell.

- **Pipelined cache architecture**: High performance processors require caches that can at the same time offer high bandwidth and provide large cache capacity in order to reduce the miss rate. However, large caches increase the bit-line capacitance. In turn, the high capacitance deteriorates the access time, which leads to multi-cycle cache accesses. To improve the performance of L1 caches, Agarwal and Vijaykumar [27] proposed an architecture that pipelines the cache so that it can be accessed each cycle.

- **Variable latency cache architecture**: To mitigate the effect of process variations, Hong and Kim [28] proposed a technique where cache access is divided into three stages to decode, access, and output the data. The access to the SRAM array incurs a two-cycle latency, whereas decoding and outputting the data requires only one cycle each. The access to the SRAM array has an extended latency in order to hide any timing violation that might happen; however, this technique is unnecessarily pessimistic since it extends the latency even when there are no timing violations.

Mutyam et al. [29] used timing information obtained during memory functional testing to determine the cache entries that need extended latency during run-time. However, this technique is difficult to apply in NTC due to high delay variation observed in NTC. Error status of a bit cell may change due to its high sensitivity to voltage and temperature fluctuations during run-time.

- **Reconfigurable SRAM arrays and redundancy**: Several works in STC consider dynamically reconfiguring SRAM arrays by either disabling the regions of the cache that are fault prone or by trading off the array size to operate at low voltages. Ozdemir et al. [30] proposed to disable slow cache ways. However, this requires extra area to conserve the yield. Otherwise, it would hurt the cache size and increase the miss rate. In addition, because of clustering of faulty bits, there is a high probability that physically close ways are all faulty, making the cache unusable. Wilkerson et al. [31] combined healthy bits of two consecutive cache lines to form one healthy line. However, this technique can only work with low error rates (≤ 0.001)—a contrast to NTC design where PVT-induced error rates are higher.

Chapter 3

L1 Cache Design in NTC

In this chapter, a process variation model for an L1 cache operating in NTC is designed to investigate the impact of PVT compared with a cache operating in STC. In Section 3.1.2.2, microarchitectural behaviors of popular mobile applications are studied in order to motivate the design of a robust cache by exploiting cache access patterns.

3.1 L1 Cache Design Challenges in NTC

A high performance L1 cache is typically pipelined in order to overlap its long hit latency and increase the throughput. The pipeline is configured with three stages following the design presented by Agarwal and Vijaykumar [27]. Figure 3.1 shows the stages of the pipeline: 1) *Address Decoder* (AD), 2) *Cache Access* (CA) from the word-lines to the sense amplifier through the bit-lines, and 3) *Data Out* (DO). In this work, the focus is put on the CA stage since it has the shallowest logic depth with many parallel paths, and is therefore more susceptible to PVT than other stages.

Due to the presence of SRAM arrays in the CA stage, four type of failures due to PVT are observed—SRAM hold failure, read stability, write stability, and access-time failure. While 8T SRAM cells have been shown to be effective in alleviating write and read stability issues, existing techniques to deal with timing access failure comes with a drawback of severe performance degradation [32]. For instance, some cores operate at 10X lower frequency than the highest achievable frequency in order to avoid timing violations in multi-core processors [2].

Instead of lowering the frequency to a value that does not cause functional failures, this project allows the processor to work in a fault-prone environment while adaptive measures are taken to allow reliable operations. The emphasis is put on L1 cache since it contains the majority of critical paths within a processor and is the most susceptible to process variations [28, 33].

3.1.1 Modeling PVT in L1 Cache

To investigate how PVT affects L1 cache, an 8T model proposed in VARIUS-NTV [6] is used. An 8T cell model is chosen since it decouples the read from the write operation, and hence increases the read noise margin. As for the write operation, there is a high freedom to size the transistors associated with writing without worrying about the read stability.

The model considers both random and systematic variations due to lithography irregularities. For the random component, the die is divided in grids, and each grid is assigned a value of V_{th} and L_{eff} obtained from a normal distribution with zero mean. For the systematic component, a multivariate random distribution with a spherical function is used to sample the parameter values. As the distance between two grids decreases, their parameter correlation increases to model the spatial correlation observed between neighboring transistors [6]. The same coefficient for systematic and random deviations ($\frac{\sigma_{sys}}{\mu}$, $\frac{\sigma_{sys}}{\mu}$) are used.

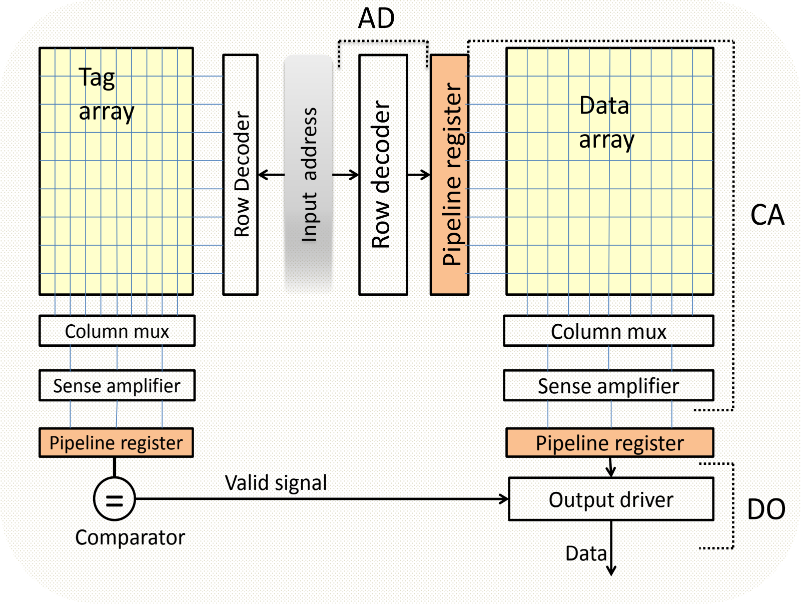


Fig. 3.1: Three stages of L1 cache pipeline.

For V_{th}, the model uses $\frac{\sigma}{\mu} = 0.09$, while $\frac{\sigma}{\mu} = 0.045$ for L_{eff} according to empirical data [34]. Subsequently, 2000 cache instances are built using the aforementioned model. Table 3.1 lists the parameter values used for the model where the technology parameters are derived from the 2012 *International Technology Roadmap for Semiconductors* (ITRS) [35].

To estimate the minimum operating voltage, V_{dd-min}, HSPICE level simulation is performed to determine the hold and write stability voltage. The results show that the write operation requires a higher voltage, which is equal to 0.31 V. Next, a guard-band of 20% is added to make sure that there is no write failure even in the worst operating conditions. To ascertain that HSPICE results are correct, analytical results from Karpuzcu et al. [6] are compared to V_{dd-min}. At this voltage, the cache instances with write faults are considered unusable.

Furthermore, a modified version of *Cache Access and Cycle Time* (CACTI) model [36] is used to obtain the nominal access delay of a 32-KB cache operating at a 32 nm node. The CACTI access time is measured against the access time from the PVT model represented by Equation (2.4) and Equation (2.5). The output of the model is a map of faults for each cache instance. The cache words whose access time is greater than the sum of the nominal delay and the guard-band are defined as *Fault-Prone* (FP) words. Note that faults are modeled at a four-byte word granularity due to memory limitations on the used system (at a one-bit granularity, the model exhausts memory resources). This means that any word with a single faulty cell is pessimistically considered faulty. The highest defect rate is set to 10%. At a supply voltage of 0.5 V, 74% of the original cache instances satisfy the tolerance threshold.

Table 3.1: Model parameters.

Technology parameters for a 32 nm technology.	
$V_{dd} - STC$: 0.9 V	$V_{dd} - NTC$: 0.5 V
$V_{th} - STC$: 0.28 V	$V_{th} - NTC$: 0.40 V
Process variation parameters	
Total $(\frac{\sigma}{\mu})_{Leff} = 0.0045$	Total $(\frac{\sigma}{\mu})_{Vth} = 0.09$
Cache instances: 2000	Correlation coefficient: 0.1

Figure 3.2 shows the cumulative distribution function of faulty four-byte words present in a population of 2000 cache instances for NTC and STC. From this figure, it can be observed that 80% of the STC instances have a fault rate less than 2%, whereas 80% of the NTC caches have a fault rate approaching 20%—an order of magnitude increase from STC. Therefore, it can be concluded that under the same technology node and process variations, NTC caches have substantially more faulty words compared with STC caches. Consequently, adequate techniques to deal with faulty cells in NTC are needed.

3.1.2 Opportunities for Resilient Cache Design

This section studies two key characteristics of cache faults that can be exploited to design a resilient cache: 1) the spatial clustering of defective cells due to PVT, and 2) the dynamic manifestation of a timing fault resulting from the application cache behavior.

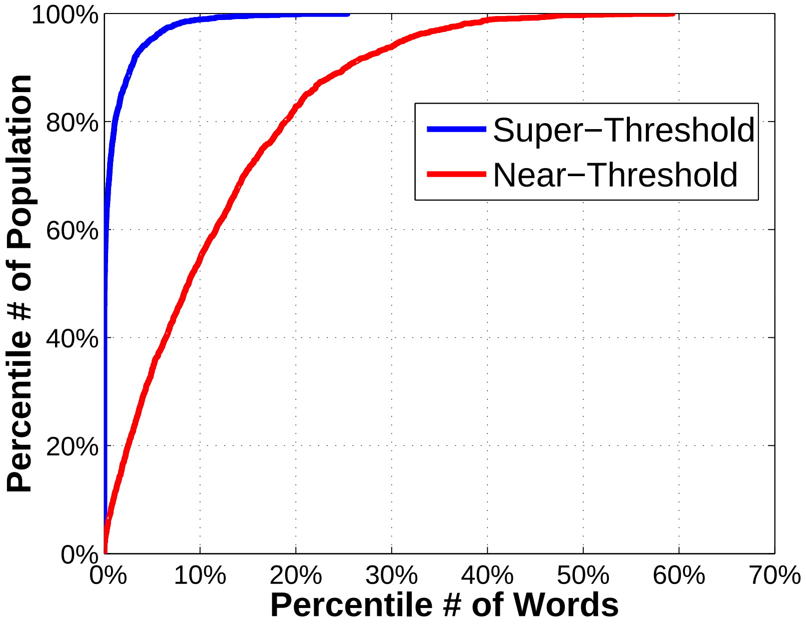


Fig. 3.2: The cumulative distribution function of faulty words in STC and NTC caches.

3.1.2.1 Congregation of FP Cells

Figure 3.3 shows the FP word distribution for one of the NTC cache instances generated in Section 3.1.1. This figure represents a cache with a defective rate of 30% with a granularity of four bytes. A close look reveals that, even at such high defective rate, FP words are strongly clustered in some regions of the cache. This clustering of defects is an advantage because it reduces the number of faulty cache sets compared with a cache with a random defect distribution.

3.1.2.2 Dynamic Manifestation of Faults

Fault manifestation rate depends on both cache fault sites and the application behavior. For example, it is possible for one application to heavily access fault sites, while another only accesses healthy regions. Consequently, a fault manifestation will be correlated with the spatial and temporal locality of the application. Using gem5 [37], this section studies L1 cache locality by simulating eight AsimBench [8] mobile benchmarks. For each benchmark, with the exception of *Frozenbubble*, one billion instructions are simulated. For *Frozenbubble*, only 0.68 billion instructions are simulated due to its small size.

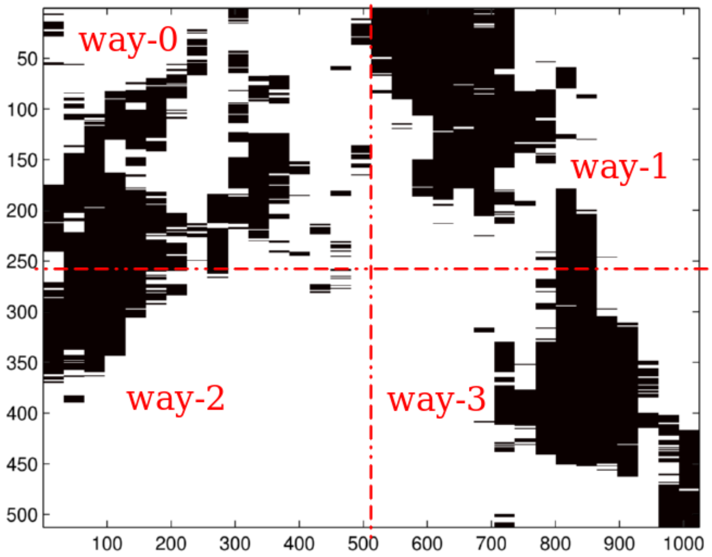


Fig. 3.3: FP word distribution under NTC. FP words are shown in black.

Table 3.2 represents the statistics of the simulated benchmarks. The findings in this table are to be expected: high load-store ratios (up to 2.3X), low L1 cache miss rates (0.27% - 3.70%), and smaller working sets compared with SPEC CPU2006 benchmarks [38,39]. The high load-store ratio means that there are fewer write instructions than read instructions. This increases spatial locality and reduces conflict misses. At the same time, the smaller working set helps in reducing misses due to L1 cache capacity.

To further study cache locality, Figure 3.4 represents the portion of all cache read requests served by the top 1% frequently accessed cache words. It can be observed that only this 1% portion of the cache serves between 75% and 90% of all access requests. Chapter 4 explores a technique to reduce fault occurrence by redirecting the most accessed words to healthy regions of the cache. This can be done by a low-overhead circuit embedded in the AD stage. Section 6 evaluates the effectiveness of that remapping scheme.

Table 3.2: Benchmark statistics.

Benchmarks	working set (MB)	L1 data miss rate	L1 instruction miss rate	Loads	Stores	Conditional branches
Adobe	67.86	2.60%	0.43%	26.80%	12.60%	12.49%
Baidumap	71.59	3.20%	1.40%	31.10%	16.80%	14.60%
Frozenbubble	38.97	2.00%	1.70%	19.30%	8.46%	12.93%
K9mail	60.28	2.20%	0.30%	19.80%	8.80%	7.80%
Kingsoftoffice	82.07	3.70%	2.60%	27.70%	12.60%	12.20%
Mxplayer	47.92	2.22%	0.27%	23.80%	10.28%	11.91%
Netease	33.01	1.68%	0.35%	29.18%	12.77%	16.30%
Sinaweibo	88.11	2.89%	0.56%	27.01%	13.48%	12.58%

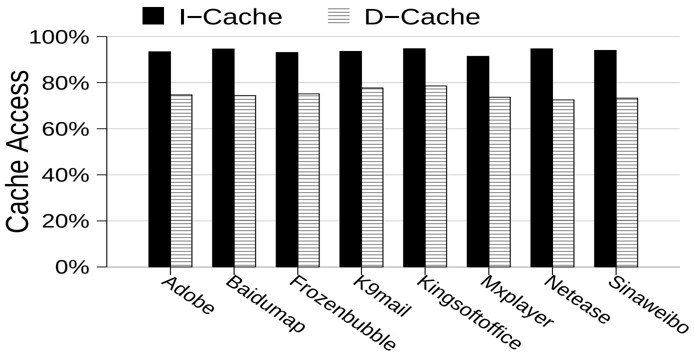


Fig. 3.4: Read access percentage served by the top 1% frequently used cache words.

Chapter 4

Adaptive Set Remapping

This chapter presents a cache remapping technique aware of the application cache behavior and the location of FP sites. Section 4.1 discusses the process of finding the candidates to be remapped. Section 4.2 describes how the remapping is carried out, while Section 4.3 gives details of how fault occurrence is detected.

4.1 Detecting a Candidate to Be Remapped

The objective of remapping is to reduce the occurrence of timing violations as much as possible by redirecting requests from faulty regions to healthy regions. However, it is necessary to study the application cache behavior in order to find the candidates that are more likely to maximize the benefit from remapping. This is a necessity that stems from the fault manifestation locality. From Section 3.1.2.2, it can be observed that the combined effect of defect clustering and application behavior results in an imbalance of fault occurrence among the sets of an L1 cache. Hence, it is necessary to find the sets incurring the highest number of faults for a given period in order to remap them first. The logic is that rarely active sets will now be mapped to fault sites and highly active sets will be remapped to healthy sites. This tradeoff will reduce the fault manifestation by allowing frequently accessed sets to have healthy accesses while defects are hidden in inactive sets.

In order to know how frequently a set incurs timing errors, a three-bit counter for each cache set is used. Only three bits per counter are used since they achieve significant accuracy in detecting the top faulty sets while accounting for a minimal overhead. As a new cache block is filled from a lower level of the memory to serve a processor request, the corresponding counter is set to 000. The counter is incremented every time a timing error occurs in the corresponding cache set. Similarly, the counter is decremented whenever the same cache set is accessed without a timing error. After a brief warm-up period, the cache sets with healthy regions that incur very few timing errors have their counters saturate at 000. On the other hand, cache sets that have timing errors and are frequently accessed have their counters saturate at 111. The indexes of the top eight cache sets that have encountered the highest number of timing errors are then stored in a small *Content Addressable Memory* (CAM) table for remapping.

4.2 Remapping

Set Remapping is based on a simple decoder shown in Figure 4.1. A similar decoder has been used by Shirvani and McCluskey [40]. Each output can select a single block in the SRAM array according to the signal (a_n ... a_1 a_0) from the set index. Figure 4.1 shows an example where a_2 a_1 a_0 = 010. In this case, the activated devices (highlighted in the figure) will form a path that ends up selecting *block*$_2$. If the last stage signal, a_2 a_1 a_0, were 000, then *block*$_0$ would be selected.

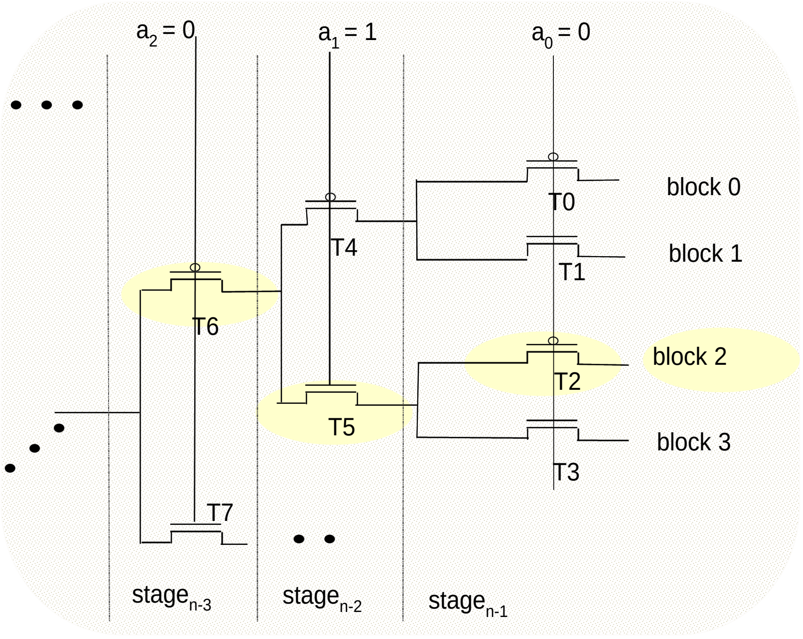


Fig. 4.1: Simple decoder.

When a cache set is being accessed, its index is used to look up the CAM table. If there is a match, it indicates that a set candidate has been found for remapping. Next, the set is remapped by flipping a certain bit of its index. Figure 4.2 presents a remapping example where the last bit of the index is remapped in the last stage of the decoder.

When the control signal a_0 of $stage_{n-1}$ is flipped (e.g., 000⇒001), the set is remapped to the nearest set. However, there is a possibility that a faulty set might be remapped to another faulty location, which would yield no benefit. One way to reduce this probability is by increasing the redirection stride. The reason behind longer redirection stride is based on the results from Chapter 3, which indicate that the words surrounding a particular faulty word are also likely to be faulty. To achieve longer strides, one can apply line interleaving to the decoder as shown in Figure 4.2. In this case, instead of having $Block_0$ and $Block_1$ close, the two blocks are separated by sixteen other blocks. For a 32-KB, two-way cache with a 10% defect rate, the best stride length is estimated using architectural simulations. Figure 4.3 shows that redirecting a set to 32 lines away results in an 85% chance of remapping to a healthy set.

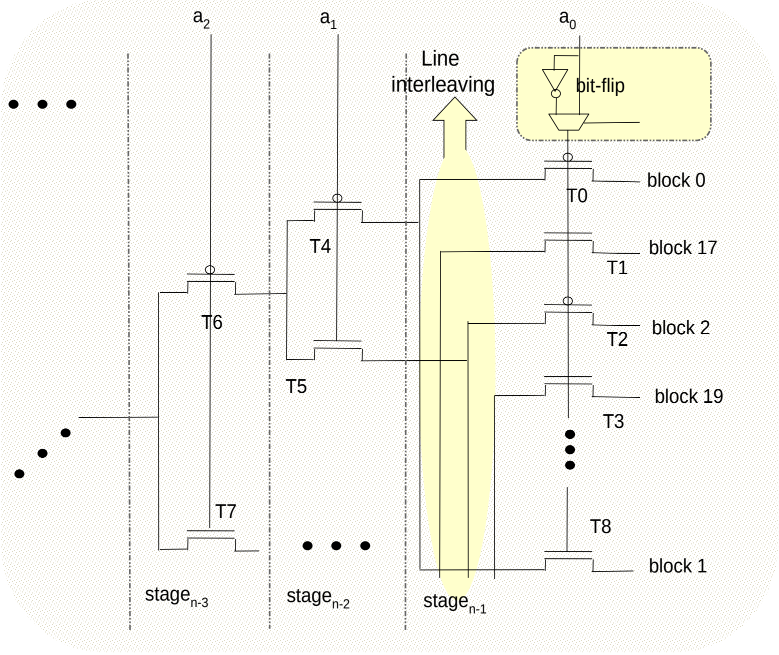


Fig. 4.2: Set remapping.

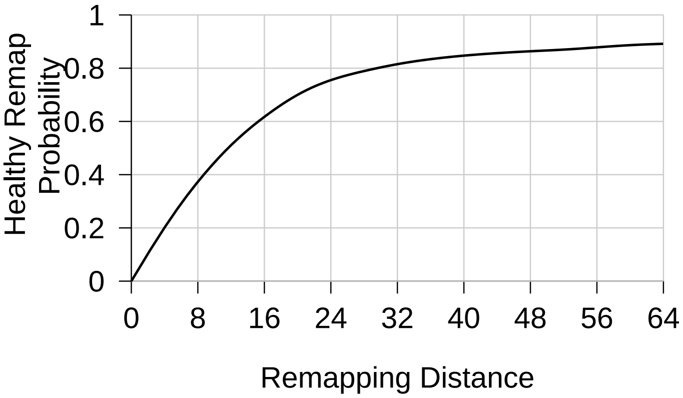


Fig. 4.3: Relationship between the remapping distance and the probability to map to a healthy set averaged across 2000 cache instances.

To avoid address aliasing, two opposite set indexes are stored in the CAM. In other words, both the index of the set that needs to be remapped and the index obtained by flipping the appropriate bit are pushed in the CAM at the same time. Consequently, it is always guaranteed that two sets exchange physical locations without resource contention. Another important step is to invalidate the valid bit of a remapped set to avoid reading the old data corresponding to the original address. In the gem5 architectural simulator, the first read access after remapping results in a cache miss as the valid bit is still set to zero.

The remapping process is preceded by the CAM look-up, which adds a delay to the AD stage. To reduce this additional delay, a small CAM table with only eight entries is used to hold the indexes of the top erroneous sets. Although the table seems small, it should be noted that only a small fraction of all sets accounts for most of the cache accesses as discussed in Section 3.1.2.2. Consequently, taking care of few sets that cause most of cache errors will significantly reduce the error rate.

To investigate the effect of the CAM on timing constraints, a 9-to-512 decoder with an embedded eight-entry CAM to control the bit-flip module is designed as shown in Figure 4.2. The timing information obtained from synthesis suggests that the CAM negligibly affects the critical path, therefore assuring that the AD stage completes with enough positive slack.

4.3 Dynamic Access Failure Detection and Correction

To detect a timing error, two registers (*early_data* and *late_data*) are used to latch the data in the DO stage. The latency from AD stage to *early_data* is three cycles, whereas the latency from AD to *late_data* is four cycles. This extension of latency is due to an extra cycle padded to the CA stage to guarantee correct data in *late_data* once a slow cell is accessed. It is assumed that any cells requiring more than two cycles to complete the CA stage are disabled. However, in the simulation performed in Chapter 3, CA stages for all cells were able to complete within two cycles. A comparison between *early_data* and *late_data*, as shown in Figure 4.4 and Figure 4.5, determines if there is a timing error. In case of a timing failure, all instructions computed after are flushed out of the pipeline and recomputed. Next, the counterflow pipelining technique adopted from Bull et al. [41] is used to assure that the correct data is available in the processor pipeline after an error.

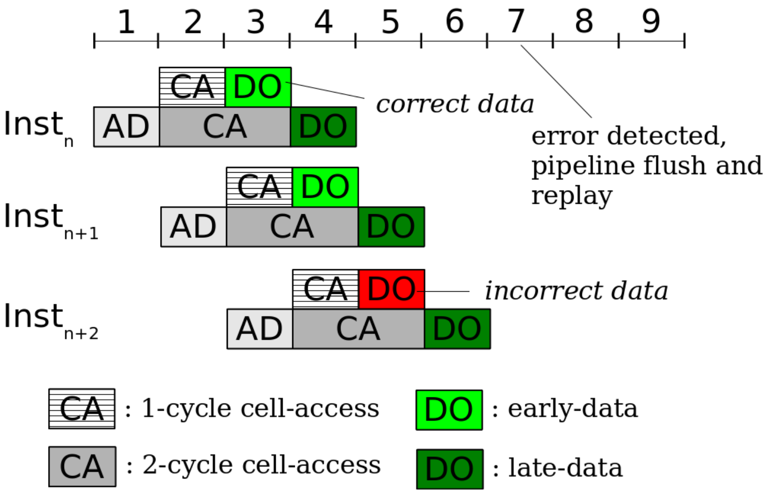


Fig. 4.4: Flow of instructions in the cache pipeline.

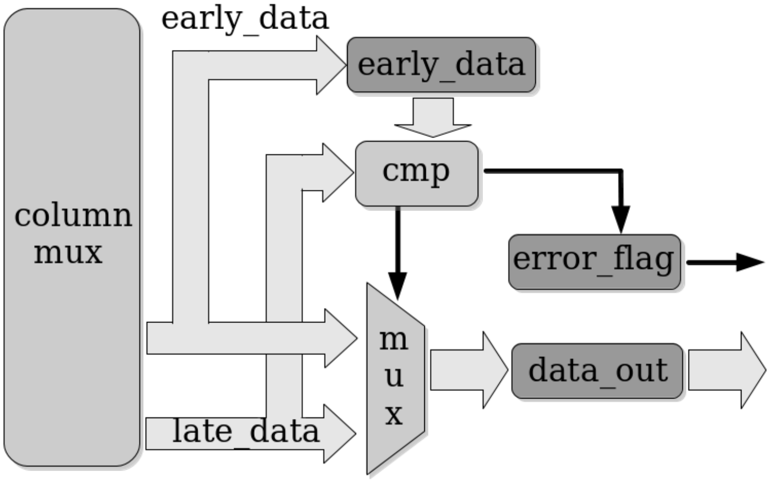


Fig. 4.5: Error detection.

4.3.0.3 Successive Cache Access

Detecting a fault in the DO stage may be problematic. For instance, if there are two successive accesses, A and B, *late_data* of A and *early_data* of B will be accessed through the same bit-lines and sense amplifier. This will result in both data being corrupted. To solve this resource contention, interleaved multi-banking is used. The applied banking scheme does not allow issue of multiple read or write operations in one cycle as done in conventional high performance processors. At most one load or store instruction can be issued in each cycle. However, the throughput is regained by exploiting the cache pipeline; up to three instructions can use the pipeline at the same time.

Since most applications have a good spatial locality (Section 3.1.2.2), using two banks is good enough. If a certain address, $Address_n$, is mapped in the first bank, then $Address_{n+1}$ will be mapped the second address. Even with the use of multiple banks, bank conflicts are still a possibility if successive accesses happen to be in the same bank. In this case, a pipeline stall is initiated. In the methodology detailed in Chapter 5, the number of stalls due to bank conflicts is incorporated in the performance estimation.

Chapter 5

Methodology

Figure 5.1 shows the detailed circuit-architectural methodology used in this work. The whole process can be divided into different parts: PVT modeling, architectural simulation, power estimation, and area overhead estimation.

5.1 PVT Modeling

The process variation model used in this work is based on the one developed by Karpuzcu et al. [6]. The model is designed using MATLAB 2011 [42]. The input of the model is the maximum time access for a given size of L1 cache. As detailed in Section 3.1.1, the model outputs a fault map. The map is a simple text file with a string of zeros and ones that represent the fault status for each SRAM cell of the cache.

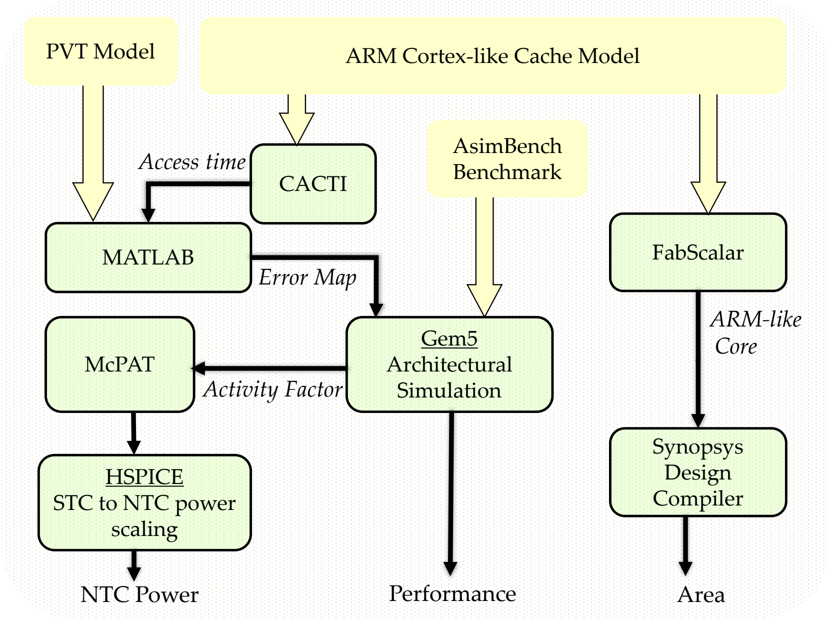


Fig. 5.1: Detailed circuit-architectural methodology.

CACTI tool is used to extract the nominal access time of a cache. Here, the inputs are cache specifications such as the size, the associativity, and the technology node. Some technology node parameters have to be first customized for NTC according to HSPICE level simulations. The main changed parameters are the supply voltage, the threshold voltage, and the gate delay. In addition, the ITRS low-operating power devices are used to match NTC specifications.

5.2 Architectural Simulation

Table 5.1 shows all the architectural parameters used for gem5 simulations. Two slightly different simulation sets are performed. The first simulation is performed in order to study how the benchmarks use L1 cache as well as acquiring performance statistics for a completely healthy cache. This simulation is performed with the original gem5 code unadulterated. In the second set of simulations, the gem5 code is modified to interface with the cache error map generated by the PVT model. Finally, a comparison between the original code simulation and the simulation in the presence of PVT-induced failures determines the PVT impact on the performance.

Table 5.1: Architectural parameters.

Architectural parameters	
Parameter	Value
L1 I-Cache	32 KB, 2 ways, 32-byte line
L1 D-Cache	32 KB, 2 ways, 32-byte line
ISA	ARM
Configuration	In-order
Frequency	100 MHz

5.3 Power Consumption Estimation

Due to run-time events, applications exhibit different power profiles. For each benchmark, gem5 feeds different statistics, such as instruction types and the use of different microarchitecture components, to McPAT power simulator [43]. However, because McPAT was designed for STC processor models, the resulting power values cannot be applied to an NTC processor. One alternative for accurate power results is to customize the simulator for NTC from the ground up. However, this is an unrealistic option given the time allotted for this project. Instead, a step-by-step technique to scale the power results from STC to NTC is used as described next.

- **Circuit level power simulation**: a chain of 31 FO4 inverters, which represents a typical combinational circuit [44], and an SRAM array to represent on-chip memories are designed using HSPICE. Next, using V_{dd} and V_{th} values in NTC, dynamic and static power consumption for the FO4 chain and the SRAM array are determined.

- **Analytical power scaling**: Dynamic power scaling is shown in Equation (5.1), where $P_{dyn-STC}$ is the dynamic power obtained using McPAT. The ratio, α_{dyn}, between NTC and STC dynamic power, obtained after HSPICE simulations, is used to scale down McPAT power results. The same logic applies to Equation (5.2), except that the target is leakage power rather than dynamic power. Finally, Equation (5.3) adds both leakage and dynamic parts to obtain the total power consumption in NTC. It should be noted that power scaling is separately applied to each major component of the processor, as different microarchitecture blocks do not have the same power profile.

$$P_{dyn-NTC} = P_{dyn-STC} \times \alpha_{dyn} \tag{5.1}$$

$$P_{leak-NTC} = P_{leak-STC} \times \alpha_{leak} \tag{5.2}$$

$$P_{tot-NTC} = P_{dyn-NTC} + P_{leak-NTC} \tag{5.3}$$

5.4 Area Overhead Estimation

For the area overhead of the remapping scheme, the FabScalar toolset [45] is used to generate a *Register Transfer Level* (RTL) description for an ARM-like core. The issue stage parameters of the core are modified to behave in an in-order fashion. The two main components that are added to the pipeline are the error detection mechanism and the set remapping modules. The area overhead is estimated by synthesizing the core's RTL using the Synopsys *Design Compiler* (DC) with a 32 nm standard cell library.

Chapter 6

Results

This chapter presents the performance, power, and area results of the remapping scheme. Section 6.1 gives details about comparative schemes. Section 6.2 and Section 6.3 compare the remapping scheme to alternative techniques in terms performance and power consumption.

6.1 Comparative Schemes

The effectiveness of the remapping technique is evaluated by comparing it against two recent schemes that mitigate timing faults. All the comparative schemes are described below.

- ***Round-Robin Remapping and disabling*** (RRR): This scheme is proposed by Abella et al. [46]. It consists of disabling the faulty blocks with a fine granularity and periodically remapping the sets. For block disabling, a granularity of four bytes is used due to the limitation in the existing gem5 code. Remapping is carried out by periodically applying the *Exclusive OR* (XOR) operation between the set index and a nine-bit counter irrespective of the cache defect map.

- ***Access time Variation Insensitive L1 Cache Architecture*** (AVICA): This scheme is insensitive to access timing violations at the expense of increased latency. The scheme is modeled simply by increasing the cache access latency by one cycle inside the gem5 simulator and by configuring the cache with two banks in order to hide the long access latency as proposed by Hong and Kim [28].

- **Adaptive remapping**: This is the proposed technique. The set remapping is guided by the application behavior and the manifestation of faults.

6.2 Performance

This section evaluates the efficacy of adaptive remapping and provides the results from the comparison with AVICA and RRR. Figure 6.1 represents the reduction in timing violations for a cache protected by adaptive remapping compared with a non-protected cache. For both a defective rate of 5% and a rate 10%, it is observed that more than 80% of the timing violations are prevented, which reduces a major part of the penalty that would be otherwise incurred. Another observation is that there is a negative correlation between the defective rate and the reduction in fault manifestation. This can be simply understood by recognizing the fact that the number of healthy sets reduces with the defective rate.

Figure 6.2 plots the performance loss induced by PVT across all simulated benchmarks. All three schemes are compared with a baseline healthy L1 cache in terms of instructions per cycle. For both defective rates, AVICA scheme performs the same—a contrast to the rest of the evaluated schemes. This can be explained by revisiting the protection technique adapted by AVICA—extra latency is used whether or not there is a defect in the accessed cell. The penalty cost of this protection scheme is understandably high as shown in Figure 6.2.

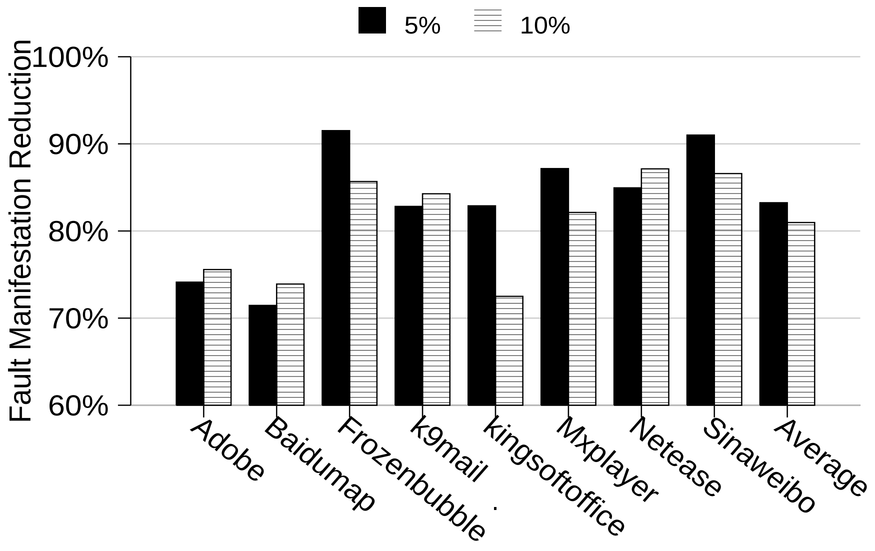
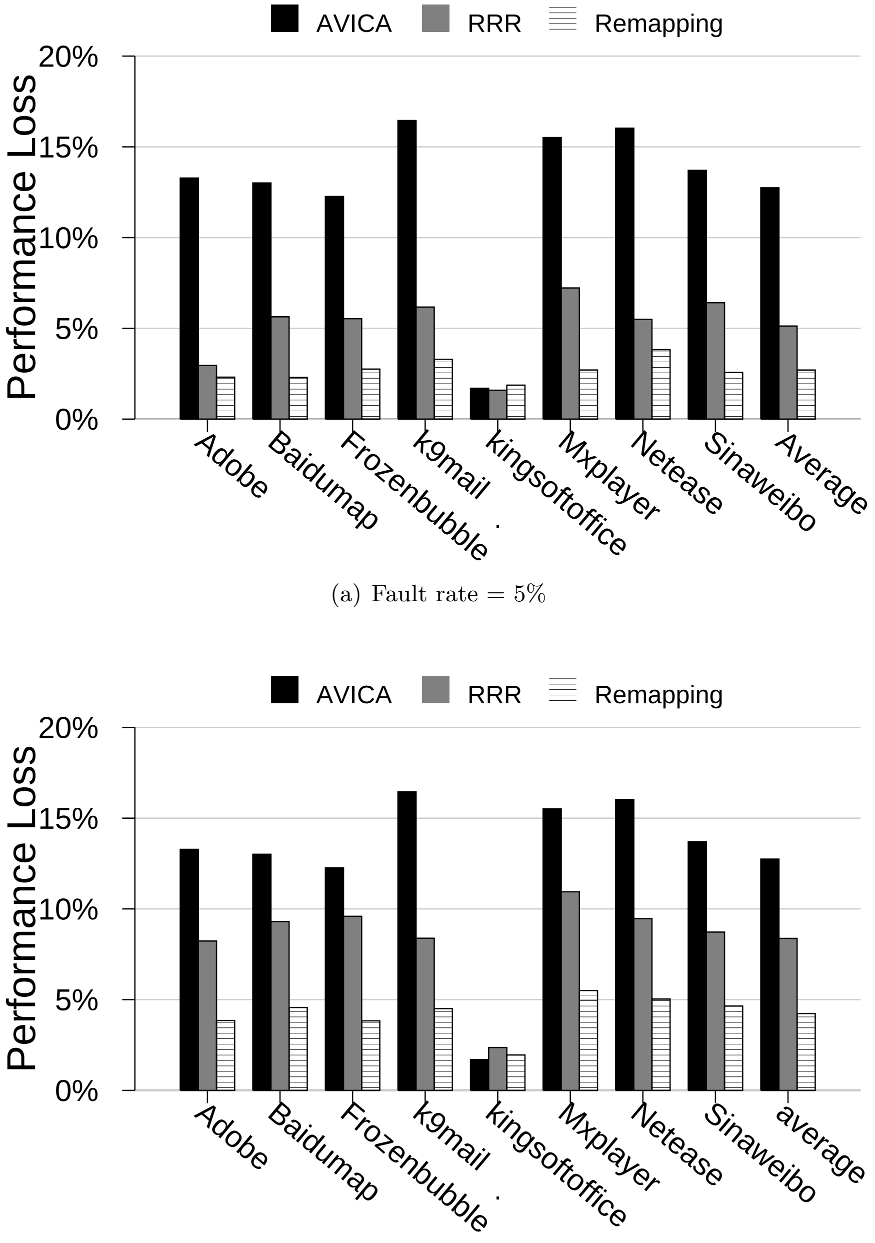


Fig. 6.1: Percentage of fault manifestation reduction due to remapping.



(a) Fault rate = 5%

(b) Fault rate = 10%

Fig. 6.2: Performance loss due to PVT. *Lower is better.*

For a defective rate of 5%, adaptive remapping scheme has a performance loss of only 2.7%, which is equivalent to 78% and 47% improvement in tackling PVT faults over AVICA and RRR respectively. For a defective rate of 10%, the performance loss for adaptive remapping is 4.2%, while it is 12.7% and 8.3% for AVICA and RRR respectively. RRR may outperform adaptive remapping for short periods. However, during an extended simulation time (one billion instructions) RRR performs poorer. A possible reason for this is that RRR aims at evenly distributing the timing faults among the sets instead of decreasing the number of faulty accesses. Consequently, there are even chances of improving or degrading the performance for a remapping event. Another bottleneck for RRR is that disabling a block results in an L1 cache miss and a high latency penalty of accessing the data in the *Level Two* (L2) cache.

The results obtained from *Kingsoftoffice* show that this benchmark has almost the same performance loss for all the schemes because of the high miss rate in the L1 cache. The performance loss due to accessing the L2 cache or the physical memory after an L1 cache miss overshadows the performance loss due to L1 cache timing faults. Therefore, the obtained results are about the same for all the schemes as the L2 cache has the same latency for all the schemes.

From the results, one can deduce that AVICA scheme is only suitable for environments with a very high percentage of PVT errors as its performance loss remains constant even when all SRAM cells in a cache are affected by process variations. On the other hand, as seen in Figure 3.2, the majority of the caches in NTC have an error rate less than 10%. Consequently, adaptive remapping is more advantageous since it has a much lower performance loss compared with the other schemes.

6.3 Power and Overhead Evaluation

Figure 6.3 represents the relative power consumed by each scheme for a defective rate of 10%. Although, on average, the power consumption is in the same range for all schemes, there is a small advantage for adaptive remapping over other schemes. This is generally caused by an increased leakage energy due to longer run-times in case of AVICA and RRR.

However, different other factors, such as the dynamic power of support circuits for each of the schemes, influence the total power consumption. In fact, because of the additional circuitry for each scheme and the activity factor, for *Mxplayer* benchmark, adaptive remapping consumes more power than other schemes.

Adaptive remapping scheme has an area overhead because of the added circuitry to support reliable memory operations. As described in Section 5, the area is estimated by synthesizing a modified FabScalar core with Synopsys DC. To accommodate the dynamic error correction and the set remapping modules, the fetch stage and the load-store unit are modified. In addition, flushing and stalling control signals are rerouted in order to communicate with the remapping modules. Including all the additional circuits, such as the CAM, counters, and the remapping control, the area overhead amounts to 5.43% of the original core.

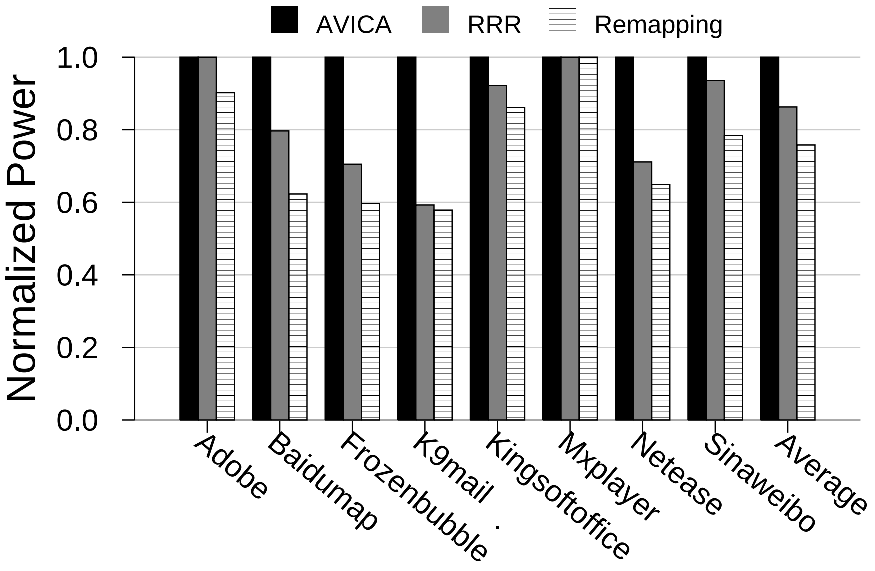


Fig. 6.3: Normalized power consumption. *Lower is better.*

Chapter 7

Conclusion

This thesis work explored the design of a timing error resilient cache in the near-threshold regime. NTC is a promising computing alternative for mobile platforms as it reduces the power budget—an essential design consideration for mobile platforms.

A process variation model has been designed in order to evaluate the impact of process and temporal variations on L1 cache. It has been demonstrated that cache design in NTC is much more challenging compared with the conventional STC regime due to an enhanced delay spread from process and temporal variations. Another finding from the model is the spatial clustering of defective cells in a few regions of the cache. Cycle-accurate architectural simulations have been performed to gain insight in how popular mobile benchmarks utilize L1 cache. The simulations showed a high spatial locality in how the applications access the cache.

To mitigate timing violation challenges in an NTC mobile processor, this project proposed a low-complexity remapping technique to efficiently and dynamically avoid timing errors during cache access. The proposed technique effectively combines application driven access patterns with the spatial characteristics of PVT-induced delay variability to boost the reliability of caches. Using a rigorous circuit-architectural evaluation methodology, this project demonstrated up to 78% and 47% savings in performance loss over two state-of-the-art techniques, AVICA [28] and RRR [46] respectively.

References

[1] A. Carroll and G. Heiser, “An analysis of power consumption in a smartphone,” in *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIXATC’10. Berkeley, CA, USA: USENIX Association, 2010, pp. 21–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855840.1855861

[2] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. N. Mudge, “Near-threshold computing: Reclaiming moore’s law through energy efficient integrated circuits,” *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[3] E. Krimer, P. Chiang, and M. Erez, “Lane decoupling for improving the timing-error resiliency of wide-simd architectures,” in *Proc. of ISCA*, 2012, pp. 237–248.

[4] S. Seo, R. G. Dreslinski, M. Woh, Y. Park, C. Chakrabarti, S. A. Mahlke, D. Blaauw, and T. N. Mudge, “Process variation in near-threshold wide simd architectures,” in *Proc. of DAC*, 2012, pp. 980–987.

[5] B. Zhai, D. Blaauw, D. Sylvester, and K. Flautner, “Theoretical and practical limits of dynamic voltage scaling,” in *Proc. of DAC*, 2004, pp. 868–873.

[6] U. R. Karpuzcu, K. B. Kolluru, N. S. Kim, and J. Torrellas, “Varius-ntv: A microarchitectural model to capture the increased sensitivity of manycores to process variations at near-threshold voltages,” in *Proc. of DSN*, 2012, pp. 1–11.

[7] S. Hanson, B. Zhai, K. Bernstein, D. Blaauw, A. Bryant, L. Chang, K. K. Das, W. Haensch, E. J. Nowak, and D. Sylvester, “Ultralow-voltage, minimum-energy cmos,” *IBMJRD*, vol. 50, no. 4-5, pp. 469–490, 2006.

[8] Y. Huand and M. Chen, “Asimbench: A mobile benchmark suite for architectural simulators,” Chinese Academy of Science, Tech. Rep. 20130821-ASG-ICT, August 2013.

[9] R. H. Dennard, F. H. Gaensslen, H. nien Yu, V. L. Rideout, E. Bassous, Andre, and R. Leblanc, “Design of ion-implanted mosfets with very small physical dimensions,” *IEEE J. Solid-State Circuits*, p. 256, 1974.

[10] N. Weste and D. Harris, *CMOS VLSI Design: A Circuits and Systems Perspective.* Addison Wesley, 2004.

[11] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach.* Elsevier, 2012.

[12] ASU, “Asu ptm model,” Jun. 2009. [Online]. Available: http://ptm.asu.edu/

[13] R. Dreslinski, B. Zhai, T. Mudge, D. Blaauw, and D. Sylvester, “An energy efficient parallel architecture using near threshold operation,” in *Parallel Architecture and Compilation Techniques, 2007. PACT 2007. 16th International Conference on*, Sept 2007, pp. 175–188.

[14] U. R. Karpuzcu, I. Akturk, and N. S. Kim, “Accordion: Toward soft near-threshold voltage computing,” *IEEE 20th International Symposium on High Performance Computer Architecture (HPCA), 2014*, pp. 72–83, 2014.

[15] U. Karpuzcu, N. S. Kim, and J. Torrellas, “Coping with parametric variation at near-threshold voltages,” *Micro, IEEE*, vol. 33, no. 4, pp. 6–14, July 2013.

[16] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” in *Proc. of ISCA*, 2011, pp. 365–367.

[17] H. Esmaeilzadeh, A. Sampson, M. Ringenburg, L. Ceze, D. Grossman, and D. Burger, “Addressing dark silicon challenges with disciplined approximate computing,” in *Proc. of DaSi*, 2012.

[18] Y. Zhang, L. Peng, X. Fu, and Y. Hu, “Lighting the dark silicon by exploiting heterogeneity on future processors,” in *Proc. of DAC*, 2013.

[19] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, “Near-threshold voltage (ntv) design—opportunities and challenges,” in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, June 2012, pp. 1149–1154.

[20] S. Mukhopadhyay, H. Mahmoodi, and K. Roy, “Modeling of failure probability and statistical design of sram array for yield enhancement in nanoscaled cmos,” *TCAD*, vol. 24, no. 12, pp. 1859 – 1880, dec. 2005.

[21] S. Nassif, “Modeling and analysis of manufacturing variations,” in *Custom Integrated Circuits, 2001, IEEE Conference on.*, 2001, pp. 223–228.

[22] C. Visweswariah, “Death, taxes and failing chips,” in *Design Automation Conference, 2003. Proceedings*, June 2003, pp. 343–347.

[23] R. Aly, M. Faisal, and M. Bayoumi, “Novel 7t sram cell for low power cache design,” in *SOC Conference, 2005. Proceedings. IEEE International*, Sept 2005, pp. 171–174.

[24] B. Wang, T. Q. Nguyen, A. T. Do, J. Zhou, M. Je, and T. Kim, “A 0.2v 16kb 9t sram with bitline leakage equalization and cam-assisted write performance boosting for improving energy efficiency,” in *Solid State Circuits Conference (A-SSCC), 2012 IEEE Asian*, Nov 2012, pp. 73–76.

[25] L. Chang, R. Montoye, Y. Nakamura, K. Batson, R. Eickemeyer, R. Dennard, W. Haensch, and D. Jamsek, “An 8t-sram for variability tolerance and low-voltage operation in high-performance caches,” *J. of Solid-State Circ.*, vol. 43, no. 4, pp. 956–963, 2008.

[26] B. Calhoun and A. Chandrakasan, “A 256-kb 65-nm sub-threshold sram design for ultra-low-voltage operation,” in *JSSC*, vol. 42, no. 3, March 2007, pp. 680–688.

[27] K. R. Amit Agarwal and T. N. Vijaykumar, “Exploring high bandwidth pipelined cache architecture for scaled technology,” in *DATE*, 2003.

[28] S. Hong and S. Kim., “Avica: An access-time variation insensitive l1 cache architecture,” in *DATE*, 2013.

[29] M. Mutyam, F. Wang, R. Krishnan, V. Narayanan, M. Kandemir, Y. Xie, and M. J. Irwin, “Process variation-aware adaptive cache architecture and management,” in *IEEE Trans. Computers*, vol. 58, no. 7, Jul 2009.

[30] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou, “Yield-aware cache architectures,” in *Proc. of MICRO*, 2006, pp. 15–25.

[31] C. Wilkerson, H. Gao, A. R. Alameldeen, Z. Chishti, M. Khellah, and S.-L. Lu, “Trading off cache capacity for reliability to enable low voltage operation,” in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA ’08, 2008, pp. 203–214.

[32] G. Chen, D. Sylvester, D. Blaauw, and T. Mudge, “Yield-driven near-threshold sram design,” *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 11, pp. 1590–1598, 2010.

[33] Y. Pan, J. Kong, S. Ozdemir, G. Memik, and S. W. Chung, “Selective wordline voltage boosting for caches to manage yield under process variations,” in *DAC*, 2009.

[34] S. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, “Varius: A model of process variation and resulting timing errors for microarchitects,” *Semiconductor Manufacturing, IEEE Transactions on*, vol. 21, no. 1, pp. 3–13, Feb 2008.

[35] ITRS, “Executive summary,” *International Technology Roadmap For Semiconductors*, 2012.

[36] N. Muralimanohar, R. Balasubramonian, and N. Jouppi, “Cacti 6.0: A tool to model large caches,” *School of Computing, University of Utah, Technology Report*, 2007.

[37] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, “The gem5 simulator,” *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[38] D. Gove, “Cpu2006 working set size,” *SIGARCH Comput. Archit. News*, vol. 35, no. 1, pp. 90–96, Mar. 2007. [Online]. Available: http://doi.acm.org/10.1145/1241601.1241619

[39] K. Ganesan, D. Panwar, and L. John, “Generation, validation and analysis of spec cpu2006 simulation points based on branch, memory and tlb characteristics,” in *Computer Performance Evaluation and Benchmarking*, ser. Lecture Notes in Computer Science, D. Kaeli and K. Sachs, Eds. Springer Berlin Heidelberg, 2009, vol. 5419, pp. 121–137.

[40] P. Shirvani and E. McCluskey, “Padded cache: a new fault-tolerance technique for cache memories,” in *VLSI Test Symposium, 1999. Proceedings. 17th IEEE*, 1999, pp. 440 –445.

[41] D. Bull, S. Das, K. Shivashankar, G. Dasika, K. Flautner, and D. Blaauw, “A power-efficient 32 bit arm processor using timing-error detection and correction for transient-error tolerance and adaptation to pvt variation,” *Solid-State Circuits, IEEE Journal of*, vol. 46, no. 1, Jan 2011.

[42] MathWorks, “Matlab technical computing environment,” 2011. [Online]. Available: http://www.mathworks.com

[43] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, “Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures,” in *Proc. of MICRO*, 2009, pp. 469 –480.

[44] N. R. Pinckney, K. Sewell, R. G. Dreslinski, D. Fick, T. N. Mudge, D. Sylvester, and D. Blaauw, “Assessing the performance limits of parallelized near-threshold computing,” in *DAC*, 2012, pp. 1147–1152.

[45] N. K. Choudhary, S. V. Wadhavkar, T. A. Shah, H. Mayukh, J. Gandhi, B. H. Dwiel, S. Navada, H. H. Najaf-abadi, and E. Rotenberg, “FabScalar: composing synthesizable rtl designs of arbitrary cores within a canonical superscalar template,” in *Proc. of ISCA*, 2011, pp. 11–22.

[46] J. Abella, J. Carretero, P. Chaparro, X. Vera, and A. Gonzalez, “Low vccmin fault-tolerant cache with highly predictable performance,” in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM International Symposium on*, 2009, pp. 111–121.