Cache Designs for Reliable Hybrid High and Ultra-Low Voltage Operation

Bojan Marić

Barcelona, 2014.

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

DOCTOR OF PHILOSOPHY / DOCTOR PER LA UPC
Doctor Europeus Mention

Department of Computer Architecture
Technical University of Catalonia

Cache Designs for Reliable Hybrid High and Ultra-Low Voltage Operation

---

Bojan Marić

Barcelona, 2014.

ADVISORS:
Jaume Abella
Barcelona Supercomputing Center
Mateo Valero Cortés
Universitat Politècnica de Catalunya
Barcelona Supercomputing Center

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

DOCTOR OF PHILOSOPHY / DOCTOR PER LA UPC
Doctor Europeus Mention

Departament d'Arquitectura de Computadors
Universitat Politècnica de Catalunya

To my parents

Abstract

Increasing demand for implementing highly-miniaturized battery-powered ultra-low-cost systems (e.g., below 1 USD) in emerging applications such as body, urban life and environment monitoring, etc., has introduced many challenges in the chip design. Such applications require high performance occasionally, but very little energy consumption during most of the time in order to extend battery lifetime. In addition, they require real-time guarantees. The most suitable technological solution for those devices consists of using hybrid processors able to operate at: (i) high voltage to provide high performance and (ii) near-/sub-threshold (NST) voltage to provide ultra-low energy consumption. However, the most efficient SRAM memories for each voltage level differ and it is mandatory trading off different SRAM designs, especially in cache memories, which occupy most of the processor's area.

In this Thesis, we analyze the performance/power tradeoffs involved in the design of SRAM L1 caches for reliable hybrid high and NST Vcc operation from a microarchitectural perspective. We develop new, simple, single-Vcc domain hybrid cache architectures and data management mechanisms that satisfy all stringent needs of our target market. In particular, we propose following: (i) the Hybrid Cache Ways architecture, which splits the cache into two sections optimized for high performace and ultra-low energy; (ii) an efficient, but simple Adaptive DAta Management (ADAM) mechanism for HP operation on single-Vcc domain caches for hybrid operation, which is tailored to detect hit distribution dynamically across the cache regions (HP ways and ULE ways regions) during program execution and adapts to different application behaviors to optimize performance and energy consumption by means of an extremely simple hardware mechanism; (iii) new cache architectures, which rely on replacing energy-hungry bitcells (e.g., 10T) by more energy-efficient and smaller cells (e.g., 8T) enhanced with EDC features to improve energy and area efficiency without jeopardizing reliability levels to still provide predictable performance, as needed for critical applications and (iv) Adaptive Performance-

Predictable Low-Energy (APPLE) single-Vcc domain L1 cache designs, which rely on replacing large memory cells by more energy-efficient and smaller cells enhanced with extra cache lines set up in a cache-assist structure, i.e., an adapted victim cache, to allow extra associativity for some cache sets that may need it due to disabled faulty cache lines. Proposed solutions are shown to have high energy efficiency with negligible impact on average performance while maintaining strong performance guarantees as required for our target market.

Acknowledgments

It was very long time ago when I came to Barcelona and start my PhD. Now I feel like that amazing period passed very quickly. For sure, this is because I met a lot of interesting people and shared many moments which I will not forget never. Those frendships, experiences and teachings made me change radically, opening my mind to more wider scopes in the life and work.

First of all, I would like to thank my advisors Prof. Mateo Valero Cortes and Dr. Jaume Abella Ferrer. It has been an honor to be their PhD student. I would like to acknowledge especially Jaume for believeing in me from the begining and great advices. I remember when he encouraged me to keep up working further especially in the moments when we got the papers "rejected". Actually, he taught me how to do a research and advise somebody. I want to thank my group (CAOS) leader at the Barcelona Supercomputing Center, Dr. Francisco Cazorla, for continuosly supporting my research activity and for patiently dealing with my crazyness with bureaucracy procedures. I am also thankful to all my collegues in the CAOS group at Barcelona Supercomputing Center Alessandro Morari, Marco Paolieri, Petar Radojkovic, Vladimir Cakarevic, Victor Jimenez, Mikel Fernandez, Carlos Luque, Javier Jalle, Qixiao Liu, Leonidas Kosmidis, Milos Panic, Dr. Carles Hernandez, Dr. Eduardo Quiñones, Dr. Roberto Gioiosa and Dr. Miquel Moreto with whom I shared a lot of moments during those years. Also I would like to thank my collegues at Microsoft Development Center Serbia, UPC and BSC Rajo CEO, Vujke, Pile, Mića, "Kombaj' ekipa": Mrdzi, Brane i Ratko; Zuki, Zoki Kraljina, Marković, Stipić, Vasilis, Djomla, Ugi, Pavle, Puzo and Tanasije for sharing with me very interesting discussions and great time when I was working in Barcelona. I am deeply grateful to my family for their unconditional support and their love, especially to my parents.

Finally, an official acknowledgement to those whom supported the work in this thesis: the Spanish Ministry of Education and Science under grant TIN2007-60625 and TIN 2012-34557 as well as HiPEAC European Network of Excellence on High Performance and Embedded Architecture and Compilation. This thesis has also been partially supported by the UPC under grant FPI-UPC.

Contents

Chapter 1
Introduction

The convergence between the high-performance computing market and the embedded/mobile market has been a dominant factor in the computing market during the last two decades [34]. Such convergence is basically motivated by three factors:

- The increase in performance and functionality requirements of the mainstream consumer market segment pushes low-power processors to include high-performance features.
- During the 90's and 00's processors increased their performance at the expense of a rapid increase in power dissipation. For example, Intel processors increased power from 15W (Pentium) to 115W (Pentium4) in less than 10 years. Limitations in heat dissipation leads to a U-turn in high-performance processor design. Low-power features where increasingly incorporated to provide the highest performance under a given power envelope. For example, Intel released Pentium M (which dissipated up to 27W) right after Pentium 4.
- Processor design, testing and validation is a time consuming and expensive task, so using designs across market domains drastically decreases costs.

Recently, market convergence has extended also towards *new market segments*, encompassing a vast array of emerging applications such as environment sensors to monitor wind, sea level, temperature, tsunamis, biomedical and healthcare sensors to monitor the body, etc. In recent years, the design approaches for those systems have been found to be limited in number and service requirements as well as cost-ineffective due to the growing need for more functionalities and performance [34]. Addressing those issues efficiently will be very important due to omnipresence of these new markets in near future.

Nowdays, aggressive silicon geometry scaling and technology evolution enables adding some degree of intelligence to any control or measuring engine by means of battery-powered ultra-low-cost (e.g., below 1 USD) computing devices running critical real-time applications. Such applications can be characterized by long time periods (which vary for different applications) of minimal data processing interleaved with short bursts of intense computation over larger data sets (see Figure 1.1). For example, consider a human body vital signs monitoring system in which a computer is attached to an activity sensor. Such system spends most of the time (e.g., 99% - 99.99% of the time [38, 56, 78, 84]) processing non-critical input data until infrequent events arise. During those long periods, low performance and processing needs are required while consuming the minimum amount of energy possible in order to extend battery lifetime. However, when infrequent events arise (e.g., 0.01% - 1% of the time [78]), the system must read and process a larger data input set and react quickly [56]. During this short time period inputs change noticeably and high performance must be provided. Guaranteed performance and reliability are also needed to provide *strong functional and timing guarantees* [86] given that a failure to perform an operation correctly and within a given time may have a catastrophic consequence in this environment. For example, in the wearable computing domain, there is a new form of human-computer interaction comprising a small body-worn computer (e.g. user-programmable device) that is always on and always ready and accessible. In this regard, the new computational framework differs from that of hand held devices, laptop computers and personal digital assistants. The "always ready" capability leads to a new form of synergy between human and computer, characterized by long-term adaptation through constancy of user-interface. What does it look like and how would this be useful for? For example, it can be an ordinary piece of clothing but with integrated communication functions. Therefore, safety at work could be improved by allowing us to communicate our position immediately in case of problems or danger (which occurs during relatively short time periods when device's high performance is needed). This is a vital function if you are a fire-man or other rescue worker putting your life on the line every day, so device's performance guarantees must be provided within a given time deadline. Productivity could be improved by allowing the factory worker easy access to distant information sources almost immediately, thereby allowing him to solve problems on the spot and save time.

Therefore, the main requirements of these new market segments are:

- ultra-low energy consumption in order to extend battery lifetime,

- high performance to react in front of infrequent events,
- simple system design for increased yield and reduced costs (e.g., below 1 USD), and
- strong timing and functional guarantees as needed for running critical applications on top.

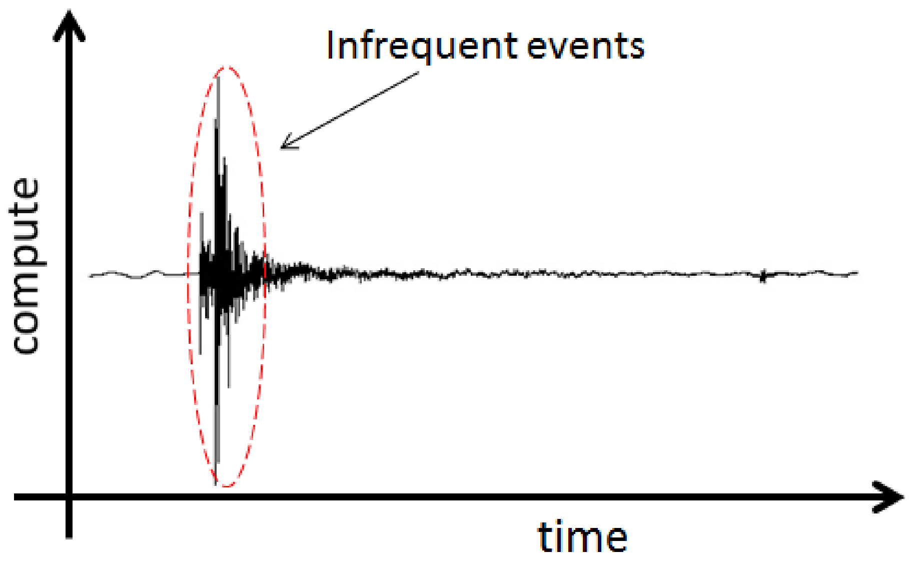


Figure 1.1: Hypothetical representation of the target applications behavior.

Basically, systems must provide high performance and low power operation when computing requirements are high as well as ultra-low energy operation when low performance is required. This can be achieved by operating at high/moderate supply voltage (Vcc) during high-performance operation and at near-/sub-threshold (NST) voltage during low-performance operation. Therefore, two operation modes with different needs and different optimal Vcc must be distinguished:

- high-performance and low-power operation mode under high or moderate voltage (HP mode for short) during relatively short periods of time to react to some infrequent particular events, and
- ultra-low energy and reliable operation mode under NST voltage (ULE mode for short) during most of the time until infrequent events arise.

```
while (true)
    ULE mode code;
    if (switch condition met) then
        ULE mode to HP mode transition;
        HP mode code;
        HP mode to ULE mode transition;
    endif;
end while;
```

Figure 1.2: An example of the typical application code structure.

The typical code structure for applications to be run on top of these processors is shown in Figure 1.2. Basically, the ULE mode code is executed in an infinite loop, checking the condition to enter the HP mode (e.g., based on some inputs typically read from sensors). After a ULE mode to HP mode transition, the HP mode code is executed. When HP mode code finishes its execution the program switches back to ULE mode. This Thesis focuses on the hardware design aspects, so application-level considerations are beyond of the scope of this Thesis.

Existing solutions can be used to handle both operation modes, but they are mainly based on having multiple Vcc domains. Multiple Vcc domains enable hybrid HP and ULE operation and suit those markets with moderate-cost computing devices such as smartphones, some implanted devices and the like. However, their design, test, validation and fabrication costs increase to unaffordable levels for the ultra-low-cost market, where chips can be priced even below 1 USD. Therefore, new, ultra-low-cost, single-Vcc domain processors [32, 87], which enable efficient HP and ULE operation, must be developed.

Table 1.1 shows some existing low-power processors which may be deployed in our target domain, but they basically do not satisfy all the requirements. For example, ARM Cortex A5 [9] and Cortex R4 [10] are more convenient for moderate-cost embedded devices such as smartphones or automotive devices. ARM Cortex M0+ processor [11] is optimized for power-sensitive microcontroller devices and can be deployed in our market segment, but it's performance is low. Couple of months ago, Intel released the Intel Quark SoC X1000 [43], which is shown to be a good candidate to be deployed in the target platforms in near future. Intel has also fabricated a wide voltage-operating-range IA-32 processor in 32nm CMOS technology recently [47], aimed at the sensor applications, but it uses multiple voltage domains which increases costs and complexity.

Table 1.1: Low-power processor overview.

Processor	Target	Complexity	Perf.	Power	Rel. year
ARM Cortex A5 [9]	mobile, pervasive embedded, consumer and industrial devices	moderate	high	low	2009
ARM Cortex R4 [10]	real-time embedded SoC applications, consumer, automotive devices	moderate	high	low	2010
ARM Cortex M0+ [11]	optimized and power-sensitive microcontroller, consumer, medical devices	low	low	low	2012
Intel Quark SoC X1000 [43]	industrial internet-of-things, wearables	low	moderate	low	2013
Intel IA-32 processor [47]	sensor applications	low	moderate	ultra-low	2012

Using a single-Vcc domain introduces some challenges in CMOS designs when the same circuits are intended to operate at drastically different Vcc levels, because, unfortunately, CMOS technology does not behave equally at different voltage levels. This fact is particularly true for SRAM memory cells because the most power-, delay- and area-efficient SRAM cells for high and moderate voltage operation do not operate reliably at ultra-low voltage. Conversely, those SRAM cell designs suitable for ultra-low voltage operation [17, 46, 54, 89] are far from being optimal at higher voltage levels due to substantially high power, delay and area overheads. SRAM cache memories are particularly critical because they:

- occupy most of the chip area,
- provide increased performance, which is particularly important in high-voltage operation, and

- avoid many energy-hungry off-chip memory accesses.

However, energy consumption, latency and area of on-chip cache memories are critical issues in current and future microprocessors. This fact is particularly true for L1 data and instruction caches due to their high activity and area contribution to the chip. Publicly available data about the energy contribution of caches for embedded processors are scarce and ultra-low cost processors for hybrid voltage operation do not exist yet. However, some processors such as the 21464 Alpha one [85] reported 26% dynamic energy for caches. Caches of extremely simple processors as the ones we target are expected to have a much larger relative contribution to the total energy consumption of the processor. Similarly, L1 caches have been shown to occupy a large area fraction (e.g., 50% of the total chip for the ARM Cortex A5 [9]) and they are accessed very frequently.

In general, L1 caches take up a significant fraction of total energy at HP mode due to frequent accesses and dominate dynamic energy at high voltage. At ULE mode caches are expected to be the main energy contributor due to both dynamic energy and leakage, which highly correlates with the area occupancy. Despite their significant energy consumption, caches are desirable from an energy perspective since they filter many off-chip accesses whose energy consumption is high and whose latency increases chip's leakage. On the other hand, caches are not desirable from a time predictability perspective. Only deterministic caches can be used in order to provide *strong timing guarantees* [86]. Therefore, devising fast and energy-efficient L1 cache designs with strong performance guarantees is of prominent importance for hybrid processors operating at high and ultra-low voltage levels in our target market.

1.1 Thesis Objectives

In this section, we describe the topic we deal with in this thesis. We first highlight the main aspects related to hybrid high and ultra-low voltage operation. Then, we briefly describe existing low/ultra-low SRAM cells. Finally, we explain the importance of achieving strong performance guarantees.

1.1.1 Hybrid High and Ultra-Low Voltage Operation

High performance required by critical applications at HP mode is only available at *saturation* voltage (typically above 400-500mV for current technology nodes [93]). Dynamic

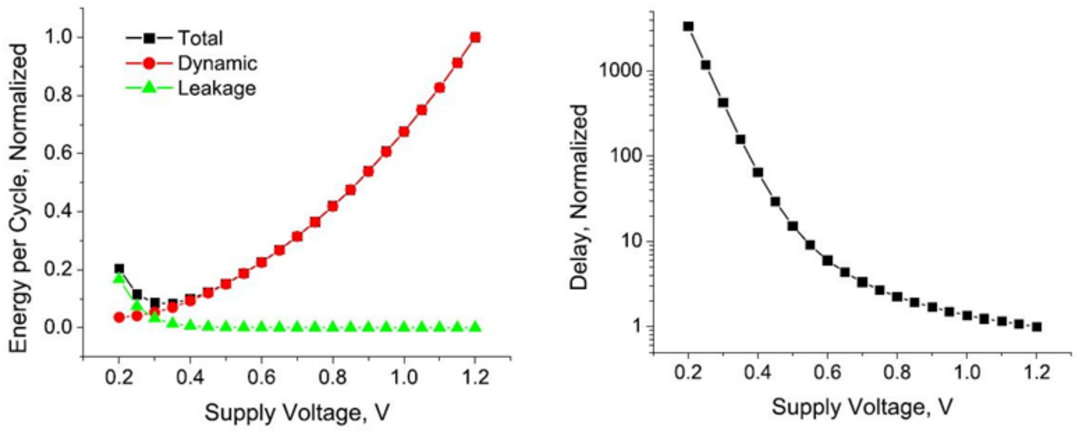


Figure 1.3: Energy and delay at different voltage levels [22].

energy at those voltage levels is high due to the quadratic dependence of energy on voltage, and hence, energy cannot be neglected even if the main constraint is performance. Thus, those systems require minimizing energy under a given performance constraint. On the other hand, when applications require modest or low performance (but still with real-time constraints) the main target is minimizing energy consumption (ULE mode). The simplest way to minimize the total energy is to scale down Vcc to the NST regime [16, 35], but without increasing fault rates beyond affordable levels.

CMOS combinational logic shows near linear voltage-delay scaling in saturation regime, and exponential delay increase in NST voltage regime. Dynamic power decreases much faster than delay increases, and hence, decreasing voltage is beneficial in terms of dynamic power and energy. However, leakage power decreases slowly with voltage scaling, and total leakage energy grows as delay (and hence, execution time) increases (see Table 1.2). Thus, there is a "sweet" point in terms of energy, where the total amount of energy required to execute a given task is minimized. Increasing voltage from that point increases energy due to dynamic energy, whereas decreasing voltage increases energy due to leakage energy. This effect is depicted in Figure 1.3 (left). In general, the energy-sweetest point for CMOS technology is in the range 200-400mV, thus in NST regime. However, performance is low as shown in Figure 1.3 (right) because delay grows exponentially when voltage is lowered.

Table 1.2: Dynamic/Leakage Power/Energy Definitions

Formula	Description
$E_{total} = E_{dyn} + E_{leak}$	E_{total}-Total energy, E_{dyn}-Dynamic energy and E_{leak}-Leakage energy
$E_{dyn} = p_t \cdot C_L \cdot V_{DD}^2$	p_t-Switching probability, C_L-Load (wiring and device) capacitance and V_{DD}-Supply voltage
$P_{dyn} = E_{dyn} \cdot f_{CLK}$	P_{dyn}-Dynamic power and f_{CLK}-Clock frequency
$E_{leak} = P_{leak} \cdot t_d$	E_{leak}-Leakage energy, P_{leak}-Leakage power and t_d-propagation delay
$P_{leak} = I_0 \cdot 10^{-V_{TH}/S} \cdot V_{DD}$	P_{leak}-Leakage power, I_0-Function of reverse saturation current, the diode voltage and the temperature, V_{TH}-Threshold voltage and S-Subthreshold slope (typically about 100mV/decade)
$t_d = k \cdot \frac{C_L \cdot V_{DD}}{(V_{DD}-V_{TH})^{\alpha}}$	t_d-Propagation delay, k-proportionality constant specific to a given technology, α-Velocity transistor saturation (in the range [1..2])

1.1.1.1 Integration and Simplicity

In spite of the challenges related to hybrid voltage operation, both operation modes must be integrated into a single circuit due to several reasons. Setting up independent chips or independent circuits into the same chip to deal with each operation mode separately is undesirable due to fabrication costs, integration constraints and power efficiency. In general, producing masks for two chips as well as fabricating two different chips is more expensive than doing such process for a single chip (e.g., due to packaging, burn-in and test costs). Even for a single chip, reducing its area is critical for yield and hence cost. Similarly, integration is desirable from the volume/area perspective. The higher the integration, the lower the volume/area requirements are, and hence, those processors can be used in more domains where the space occupied is critical such as implanted devices monitoring vital signs, chips for mobile phones, etc. Finally, although independent processors/circuits may be more efficient to perform their tasks, each of such processors/circuits requires its own power distribution channels, and has some overheads related to the communication between the on-chip and off-chip circuitry. Thus, using shared circuits in a single chip to perform both kinds of tasks, those at high voltage and those at ultra-low voltage, is also the most efficient solution from a power perspective.

1.1.1.2 Process Variations

In general, combinational logic operates correctly and efficiently at both voltage levels and only process variations and increased soft error susceptibility may threat its operation [65]. However, SRAM storage suitable for high-voltage operation (typical 6T cells [44]) observes much faster delay increase when voltage is decreased, and which is even worse, those cells become unstable much before the 200-400mV range is reached mainly due to process variations that impact designs during fabrication [13]. Process variations are deviations of some parameters with respect to their nominal values. Such variations translate mainly into threshold voltage deviations. Threshold voltage variations have a significant impact on delay and energy consumption. Similarly, process variations can compromise the read/write ability of cells as well as their retention capabilities. Although process variations also have an impact on high-voltage designs, their relative impact exacerbates at low voltage [35, 40].

There are two main types of process variations: die-to-die and within-die variations. Die-to-die process variations, resulting from lot-to-lot, wafer-to-wafer and a portion of the within-wafer variations affect large regions and every element on a chip equally [14]. Their behavior is systematic and can be addressed with well-known methods such as adaptive body biasing (ABB) [81] that change threshold voltage at coarse-grain. Within-die process variations consist of systematic and random component. Systematic-within-die variations result from a repeatable principle due to empirically determined device-to-device correlation as a function between the devices and are also addressed by means of ABB [81]. Conversely, random-within-die variations do not exhibit device-to-device correlation (e.g., random variations affect each transistor in a different manner) [14]. In general, random variations in logic can be addressed by setting up long paths in terms of gates. Random variations compensate across long paths due to their statistical nature because it is very likely that some gates are slower than their nominal value whereas some others are faster. Overall, deviations can be compensated and it can be proven statistically that small guardbands in the delay suffice to deal with variations in logic. However, random variations in SRAM cells have a larger impact because those cells consist mainly of 4 transistors arranged as a ring of two inverters plus 2 passgate transistors as shown in Figure 1.4 (conventional 6T bitcell). Hence, paths are extremely short. Moreover, those cells are typically set up as small as possible to reduce their area and power, and thus, increase the integration density. As a consequence, the relative impact of random process variations in those 6-transistor (6T) SRAM cells is huge and may increase drastically read/write delay, and even make cells faulty.

1.1.2 Low and Ultra-Low Voltage SRAM Cells

Differential 6T SRAM cells have been commonly used for high Vcc operation. However, many alternative SRAM cells have been designed targeting different voltage and robustness scenarios [17, 46, 54, 59, 89]. Results in [31, 46, 54] show that 8-transistor (8T) and 10-transistor (10T) SRAM cells are superior against 6T SRAM cells at low and ultra-low voltages under various conditions such as iso-robustness, iso-area and iso-read-failure.

A fundamental problem in conventional 6T SRAM cells is read stability (or read upset) at low voltages, which cannot be achieved without significant cell upsizing [22]. A read upset occurs when a pull-down transistor in the cross-coupled inverter is unable to hold the correct value in the node. Adding two extra NMOS transistors (T3 and T4 in the 8T SRAM plot in Figure 1.4) to a 6T SRAM cell decouples read and write paths and blocks noise injection from the read path to the value holding nodes. Read stability is then achieved by sizing transistors T3 and T4 without affecting write functionality of the cell. Hence, the write stability condition is then defined only for cross-coupled inverters. Moreover, transistor T4 and pull-down transistors in the cross-coupled inverters can have smaller width, because the read margin does need to be considered thanks to the separated read port. Adding two extra NMOS transistors introduces around 30% area penalty, which makes 8T SRAM cell less efficient than 6T ones at high voltage due to power and area overheads. Delay impact is negligible at high voltage [31, 46].

The 10T SRAM cell design considered in this thesis is the Schmitt-trigger fully-differential cell [54]. Transistors T1-T8 create the cross-coupled Schmitt-trigger inverter. Such cell does not introduce any architectural change compared to the 6T SRAM cell as the read/write port is accessed through 2 passgate NMOS transistors in both types of cells (T1, T2 for 6T bitcell and T9, T10 for 10T bitcell in Figure 1.4). Read stability at ultra-low voltage is improved by the positive feedback from T7/T8 which adaptively increases or decreases the switching threshold of an inverter depending on the input transient direction in order to preserve the logic state of the cell. Write stability at ultra-low voltage is enhanced by reduced pull-down transistor sizes due to stacked NMOS transistors in the pull-down path (series connected: T1 and T3, T2 and T4). Therefore, 10T cells provide improved process variation tolerance and reliable ultra-low voltage (down to 160 mV [54]) operation due to their built-in feedback mechanism, thus making this cell superior to conventional 6T and 8T SRAM cells at ultra-low voltage. Area penalty compared

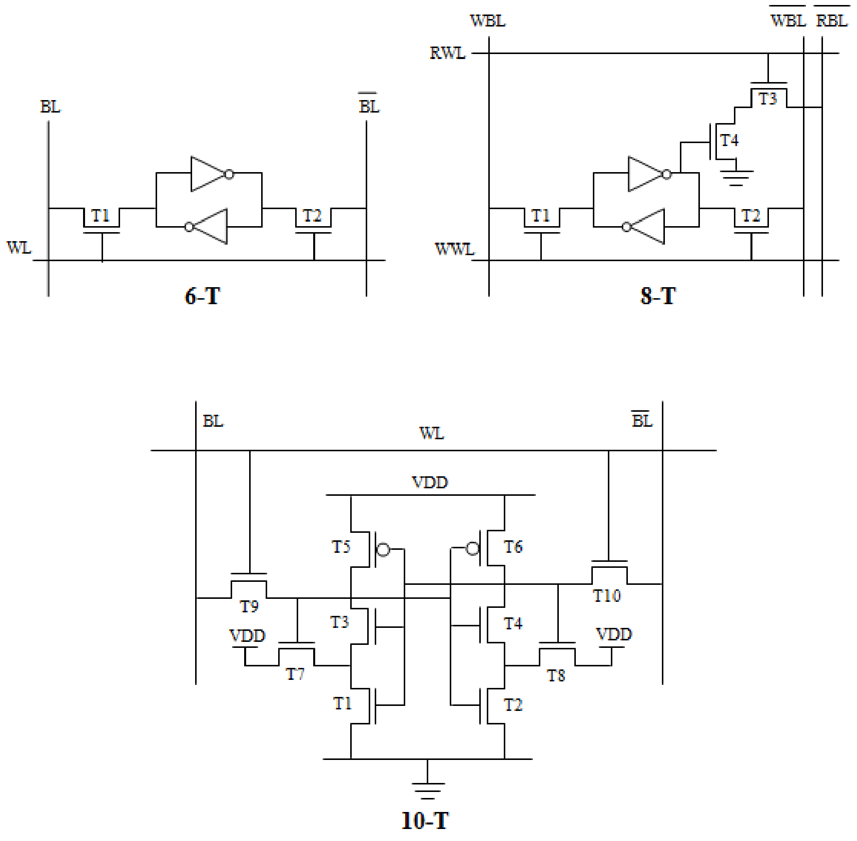


Figure 1.4: 6T (top left), 8T (top right) and 10T (bottom) bitcell designs.

to the 6T SRAM cell is around 45% [54], which makes 10T SRAM cells unattractive for high voltage operation due to significant power and area overheads. Delay increase for 10T SRAM cells with respect to 6T and 8T ones at high voltage is negligible [54].

In summary, all those cells devised for low or ultra-low voltage operation are less efficient than 6T SRAM cells at HP mode in terms of power, area and delay. Thus, whereas the same combinational logic can be employed at HP and ULE modes (despite somewhat increased error rates), efficient SRAM cell designs across the different voltage levels simultaneously do not exist. Different SRAM cell designs target particular voltage ranges, but how to combine and lay those cells out at microarchitecture level for on-chip SRAM cache memories to allow efficient operation at high and ultra-low voltage with affordable cost is an open issue.

1.1.3 Guaranteed Performance

In general, cache memories are desirable from an *average performance* and energy perspective because they filter many energy-hungry off-chip accesses. However, caches are not desirable from a time predictability perspective. Deterministic caches (i.e. implementing modulo placement and least recently used replacement policies) have been shown to provide *strong timing guarantees* in existing systems [86]. In particular, worst-case execution time (WCET) estimation requires full knowledge of the hardware features below to provide strong timing guarantees [3, 38, 84, 86]. WCET estimation is an expensive task even when cache characteristics (e.g., size, associativity, hit and miss latency, etc.) are known a priori. Moreover, there is no existing WCET estimation technique considering caches with undeterministic characteristics (e.g., caches some of whose entries may be faulty and thus disabled) in existing systems so far. For example, let us consider a W-way cache deployed in a given platform that is expected to have up to permanent F faults during its lifetime. In order to make WCET estimation tools aware of those permanent faults we should consider all combinations of F faults over N total lines in cache, leading to a large number of combinations, which is overly expensive [76]. In fact, WCET analysis is an already complex and time-consuming process and it may be the case that existing solutions do not work even if the fault location is known a priori. Alternatively, we can assume that all cache sets have F lines less, so as if the cache had associativity $W - F$ instead of W, but this decreases cache space quickly and so, WCET estimates grow too much. So far, there is only one approach which analyticaly models the performance degradation caused by faulty cells in architectural and non-architectural arrays [36]. However, this approach considers only permanent (hard) faults without taking into account soft errors which can occur in cache memories more often than hard errors, and most often arise from single event upsets caused by strikes from energetic particles such as neutrons and alpha particles.

On the other hand, deterministic caches provide the same capacity as in the fault-free case, but it may have different access latencies for different cache blocks. However, such latency difference is small, so it does not make WCET analysis too pessimistic [3, 86]. Therefore, smart approaches based on taking advantage of existing hardware must be developed to enable reliable hybrid high and ultra-low voltage operation while keeping caches simple and analyzable to provide safe and tight WCET estimates.

1.2 Thesis Contributions

The main goal of this thesis is proposing efficient cache architectures and data management mechanisms that satisfy all stringent needs of our target market. Given the simplicity of the processors in our market, we consider L1 caches as our target.

In order to reach this goal, we start by better understanding the performance/power tradeoffs involved in the design of SRAM L1 caches for hybrid high and NST Vcc operation from a microarchitectural perspective. Also we make some assumptions on the process technology used according to publicly available data. Those insights are fundamental for designing efficient processors in the future. Furthermore, this knowledge is a key element in developing accurate cache models. These models should be flexible, allowing the designers to quickly explore the design space of a processor and to detect interesting design points.

Next, we propose different hybrid cache architectures and data management mechanisms which enable reliable hybrid voltage operation with guaranteed performance. In particular, we propose new solutions for improved energy efficiency with negligible impact on average performance while maintaining strong performance guarantees. The main contributions of this Thesis are summarized in the next subsections.

1.2.1 Hybrid Cache Ways Designs

One of the main contributions of this thesis are the Hybrid Cache Ways designs - new, single-Vcc domain, hybrid L1 cache architectures, where the cache is designed by combining heterogeneous SRAM cell types to operate reliably across a wide range of voltages, consuming little energy at ULE mode as well as providing high performance at HP mode, as required for our target market. In particular, the cache is split into two sections: (i) *HP ways*: some cache ways are made of the SRAM cells optimized for one particular Vcc level (e.g., high or moderate Vcc) and (ii) *ULE ways*: the rest of the cache ways are made of the SRAM cells optimized for another Vcc level (e.g., NST Vcc).

Only ULE ways are enabled at ULE mode, given that small cache space suffices to fit the small workloads expected at such operation mode [38, 78, 84]. Therefore, HP ways are turned off, thus allowing high energy efficiency at ULE mode. However, all cache ways are enabled at HP mode to fit large workloads (compared to those at ULE mode) and provide high performance. ULE ways are reused at HP mode, in spite of their inefficiency at high Vcc, because they reduce the number of slow and energy-hungry off-chip accesses [61]. Strong timing guarantees are achieved at both modes due to the deterministic behavior of the proposed cache designs at any voltage considered. In particular, reliability of *all* cache space is high enough, so that no cache space is disabled and WCET estimates can be obtained.

1.2.2 ADAM: Adaptive Data Management Mechanism

The main drawback of Hybrid Cache Ways designs is their significant energy overheads at HP mode introduced by reusing large and robust SRAM cells of ULE ways optimized for NST Vcc operation. ULE ways are reused at HP mode despite their energy inefficiency at high Vcc because they reduce the number of off-chip accesses, which are much more expensive in terms of performance and energy [61]. Data management policies are required to access HP and ULE ways at HP mode to minimize the number of accesses to ULE ways. Unfortunately, existing policies are far from being efficient across all applications. Therefore, we propose an efficient, but simple Adaptive Data Management (ADAM) mechanism for HP operation on single-Vcc domain caches for hybrid operation. ADAM is tailored to detect hit distribution dynamically across the cache regions (HP ways and ULE ways regions) during program execution and adapts to different application behaviors to optimize performance and energy consumption by means of an extremely simple hardware mechanism. We show that ADAM combines the advantages of the previous approaches (swap and sequential access policies), and moreover, reacts in front of the different phases of a program, thus outperforming all state-of-the-art approaches at HP mode in terms of energy efficiency with negligible performance impact.

1.2.3 Efficient Cache Architectures Using Error Detection and Correction (EDC) Codes

ADAM is shown to be efficient at HP mode, but new solutions are needed in order to improve efficiency at ULE mode or even at both modes. We attack this problem by considering the fact that existing caches use large SRAM cells (e.g., 10T) to achieve high levels of reliability even at ULE mode, as needed by critical applications run on top. Decreasing the size of the memory cells for higher energy efficiency at the expense of higher failure rates is unacceptable in this environment. Faulty entries should be then disabled and strong performance guarantees required by critical applications would not be achievable. Therefore, our aim is devising new energy-efficient fault-tolerant caches without decreasing reliability levels to still provide strong performance guarantees.

We propose new cache architectures which rely on replacing energy-hungry SRAM cells (e.g., 10T) by more energy-efficient and smaller SRAM cells (e.g., 8T) enhanced with EDC features to improve energy and area efficiency without jeopardizing reliability levels to still provide predictable performance, as needed for critical applications.

1.2.4 APPLE: Adaptive Performance-Predictable Low-Energy Caches

This proposal is in spirit similar to the previous one. In fact, the motivation is the same and the idea is quite simple. Adaptive Performance-Predictable Low-Energy (APPLE) caches rely on replacing large and energy-hungry SRAM cells by more energy-efficient and smaller SRAM cells enhanced with extra cache lines set up in a cache-assist structure, i.e., an adapted victim cache, to allow extra associativity for some cache sets that may need it due to disabled faulty cache lines.

1.3 Thesis Structure

The structure of this dissertation is as follows:

- Chapter 1 presents the research field and problem matters, along with the objectives of this research. It also presents the contributions of this research and the structure of the thesis.

- Chapter 2 reviews some related work on low-power techniques for caches.

- Chapter 3 presents our experimental environment. The reference platform for this work is presented, as well as the benchmarks and tools used in this thesis. Along with this, cache and processor modeling is described.

- Chapter 4 introduces the Hybrid Cache Ways designs - the first step towards hybrid L1 cache architectures for hybrid voltage operation with guaranteed performance. It describes the architecture and presents a detailed performance evaluation.

- Chapter 5 presents a novel Adaptive Data Management (ADAM) mechanism for HP operation on single-Vcc domain caches for hybrid voltage operation. It describes how the mechanism works and evaluates it against existing state-of-the-art data management policies.

- Chapter 6 introduces new, simple, efficient, hybrid L1 cache architectures using EDC codes. It describes implementation details and presents a detailed performance evaluation.

- Chapter 7 introduces new, Adaptive Performance-Predictable Low-Energy (APPLE) caches. It describes the cache operation, implementation details and performance evaluation.

- Chapter 8 concludes this dissertation by commenting on the most important contributions of this thesis, providing a brief summary of future work, and listing the main publications related to this thesis.

Chapter 2

Related Work

Literature on low-energy techniques for caches is abundant. We classify those techniques into two main categories: low-level and high-level techniques. Low-level techniques are basically based on circuit techniques that reduce energy, while high-level techniques propose new cache architectures and organizations to attack the same problem. High-level techniques are further classified as follows:

- Cache partitioning: Reduce the energy by splitting the cache into different modules and disabling unused sections.
- Behavioral approaches: Attempt to optimize the cache operation for low energy consumption.
- Fault-tolerant approaches: Focus on error detection and correction when Vcc is decreased to reduce energy.
- Hybrid cache architectures: Combine different approaches, from circuit to architecural level of abstraction, in order to enable hybrid voltage operation.

In the following subsections, we briefly survey all those techniques and compare them with particular proposals of this thesis considering stringent requirements of the market that we target.

2.1 Low-level Techniques

Differential 6T SRAM cells have been commonly used for high voltage operation. However, many circuit-level techniques investigate the benefits of using alternative types of SRAM cells such as 8T [46], Schmitt-Trigger 10T (10T) [54], etc. to target different low voltage and robustness scenarios. Unfortunately, such memory cells substantially increase area and energy at high voltage w.r.t. 6T cells, which is unaffordable in embedded cache design if used extensively.

Hezavei et al. [39] propose techniques such as divided bitline, pulsed wordline and isolated wordline to reduce cache energy consumption. Kuroda et al. [55] propose a scheme based on Vcc and threshold voltage scaling for low power high-speed CMOS digital design. Itoh et al. [45] study the impact of voltage scaling to reduce leakage and dynamic energy in caches. Several techniques [28, 51, 55] propose lowering cache Vcc (or even gating it [71]) for some cache sections or the whole cache in order to save energy. However, those techniques do not consider the constraints of NST voltage operation (e.g., issues related to increased relative process variations impact).

2.2 High-level Techniques

2.2.1 Cache Partitioning

Several proposals have exploited different tradeoffs between performance and power by reconfiguring cache size and associativity [6, 12, 91]. They show that significant energy savings can be achieved by disabling some cache ways.

Some authors [77] have ivestigated vertical and horizontal cache partitioning, as well as Gray code addressing to reduce dynamic power. Ghose and Kamble [33] have studied the effects of using subbanking, multiple line buffers, and bitline segmentation to dynamic cache energy. There are also different approaches based on splitting the cache into different modules. Kin et al. [53] propose putting a small cache in front of the L1 cache to filter accesses to the L1 cache. This technique is known as a filter cache. A similar approach is explored by Abella and Gonzalez [1], but they use two cache modules and two Vcc domains for high performance and low power. Fujii and Sato [29] improve such approach by using also dual-Vt transistors and a single Vcc domain. Their Non-Uniform Set-Associative (NUSA) cache has cache ways with different latencies. The key idea of their approach is to allow the ways within a cache to be accessed at different speeds and to place infrequently accessed data into the slow ways. Such cache is particularly designed for large on-chip caches (e.g., last-level caches), devised particularly for high-performance market and high voltage, so they cannot be used for hybrid high and NST voltage operation.

2.2.2 Behavioral approaches

In general, all cache ways are searched in parallel because the cache access time is critical. Thus, the energy consumed for a tag subarray access and that for a data subarray access are consumed in each way. Since only one way has the data desired by the processor on a cache hit, however, conventional set-associative caches waste a lot of energy. Some techniques have been proposed for alleviating the negative effect of the set-associative caches by optimizing cache access behavior.

The first proposal is known as a phased cache, employed in Hitachi SH microprocessor [37]. In the phased cache, tag comparison and cache line access are performed sequentially. First, tag comparisons are performed without data subarray activation. Then, only a single data subarray which includes the desired data is accessed if at most one tag matches. Otherwise, a cache line replacement is performed without any data subarray access. Although this approach reduces the energy consumed for data subarray accesses, the cache access time will be increased due to the sequential access. If we know which way includes the desired data before starting the cache access (i.e.,without performing the tag comparison), the unnecessary way-accesses can be eliminated without cache access time overhead. The idea of accessing only one cache way by using way predictors was presented by Inoue et al. [41]. Powell et al. [72] extend this work using way prediction and selective direct mapping for no conflicting accesses. The main idea of their approach is that only the predicted way is accessed. If a miss occurs, then the rest of the ways are accessed. Those techniques do not deal with the constraints at ultra-low voltage. In fact, they are orthogonal to the particular cache design in place and can only be applied if a cache design suited for ultra-low voltage operation is in place.

Simple data management policies have been proposed recently [25, 26]. Those policies are not particularly devised for hybrid voltage operation caches implemented with a single Vcc domain, but they may be used during high Vcc operation (i.e. HP mode). We use them for comparison purposes and they are extensively evaluated against our proposed ADAM mechanism [61]. As shown later in Chapter 5, ADAM outperforms those techniques consistently across all applications.

2.2.3 Hybrid Cache Architectures

All proposals in this thesis [60, 63, 64], except ADAM mechanism, can be classified into this category of low-energy techniques for caches. So far, some work has been done in the area of hybrid cache architectures targeting different market segments such as high-performance or embedded systems. For that reason, we survey those techniques and put them in the context of our target market.

Dreslinski et al. propose hybrid cache architectures for embedded purposes targeting both high and near-threshold voltage operation [25]. However, such proposal relies on having multiple voltage domains in a single core which may be reasonable for moderate-cost markets such as smartphones and the like, but it is unaffordable for our target ultra-low-cost (e.g., below 1 USD) market.

Wu et al. [88] propose a Region-based Hybrid Cache Architecture (RHCA) using disparate memory technologies such as SRAM, Embedded DRAM (EDRAM), Magnetic RAM (MRAM), and Phase-change RAM (PRAM), in both 2D or 3D stacked chips. The RHCA design divides caches into fast and slow regions where each region is implemented with different memory technology. To optimize both cache regions in terms of energy and performance, two Vcc domains are required at least. Moreover, implementation of two different memory technologies, test and verification increase costs. Those facts are not in line with our target market where fabrication costs and simplicity are must. Finally, those cache designs do not provide any performance guarantees required for WCET estimation, which makes them useless in real-time scenarios.

To the best of our knowledge, two state-of-the-art cache designs, which provide *functional and timing guarantees*, have been proposed in the past [32, 94]. Zhou et al. [94] propose downsizing 6T SRAM cells of on-chip caches combined with error correction codes. Ghasemi et al. [32] propose mixing heterogeneous cell sizes of the *same* SRAM cell types. The main drawback of those designs is that they use exclusively large SRAM cells in order to provide reliable high and NST voltage operation. In fact, they are devised for the high-performance market and high voltage operation, but we put them in the context of hybrid voltage operation implemented with a single-Vcc domain and use them for comparison purposes. As shown later, the caches proposed in this thesis outperform those designs in all metrics.

2.2.4 Fault-Tolerant Approaches

The simplest way to achieve higher energy efficiency is decreasing the size of SRAM cells at the expense of higher failure rates which is particularly critical at NST voltage. Faulty cache entries should be then disabled or replaced.

Techniques based on replacing faulty cache entries [3, 7, 8, 23, 74, 87] introduce significant overheads due to bypassing and signal re-routing. Chishti et al. [23] introduce a novel adaptive technique called multi-bit segmented ECC (MS-ECC) at fine sub-cache line granularity to address both persistent and non-persistent failures at low voltages. Sasan et al. [74] propose Resizable Data Composer (RDC) cache in which error-prone parts of the cache are fixed. Ansari et al. propose two novel fault-tolerant cache architectures: ZerehCache [8] and Archipelago [7]. ZerehCache introduces fine granularity re-mapping of faulty bits by solving a graph coloring problem. They propose an interconnection network to allow a limited redundancy borrowing bits across the statically specified or fixed size groups. This architecture requires significant layout modifications in order to implement the proposed interconnection. Archipelago improvements over ZerehCache by partitioning the cache into multiple autonomous islands with various sizes which can operate correctly without borrowing redundancy from each other and minimizing the cache space lost during NST operation. Abella et al. propose the Reliable Victim Cache (RVC) [3], which uses cache-assistant structures such as the eviction buffer, the write-combining buffer, the victim cache and the like to provide cheap sparing for faulty cache lines (with lower cost than conventional sparing) and time-predictability in the presence of faults. RVC is in spirit similar to the APPLE cache [63] proposed in this thesis by using extra hardware to provide time predictability. However, authors substantially complicate cache operation. Moreover, RVC design are intended for single voltage operation.

Some recent techniques are based on simply disabling faulty storage [2, 5, 24, 70, 73, 87]. Agarwal et al. [5] propose a fault-tolerant cache with programmable multiplexers to select a non-faulty cache block when a faulty one is accessed in the same row. Ozdemir et al. [70] propose a Yield-Aware cache where they turn off either cache ways or horizontal regions of the cache that cause delay violations due to process variations or have increased leakage. There are also two well-known schemes to tolerate faulty bits in caches: word disabling (WDIS) [73, 87] and bit-fix (BFIX) [87]. Both techniques use some cache lines to repair others. In the WDIS scheme, two consecutive cache blocks are combined into a single cache block whereas BFIX scheme sacrifices a cache block to repair faults in three other cache blocks. Therefore, cache capacity is reduced by 50% and 25% when WDIS and BFIX techniques are used respectively.

Those techniques may provide noticeable performance variation for a given program depending on the faults location because the distribution of faulty bits is random. Such techniques are shown to be effective from an average performance perspective and provide functional correctness, *but fail to provide strong timing guarantees required for WCET estimation, as needed for critical applications in our target market (e.g., monitoring of the human body vital signs such as heart attacks, strokes, etc.)* [38, 84, 86]. Some recent works provide some degree of time predictability in the presence of cache faults [2, 57], but not strong performance guarantees. In this design context, cache designs proposed in this thesis are orthogonal to those kind of approaches.

Chapter 3

Experimental Framework

This chapter describes the evaluation framework that we have used in this thesis. First, we describe our assumptions on the process technology. Then we describe the evaluation methodology used to obtain results for all proposals in this thesis. The conclusions and results presented in this dissertation have been obtained with the benchmarks, the simulators and other tools that are presented in the following sections.

3.1 Technological Assumptions

In order to support hybrid voltage operation, we consider different operation modes, at least one for high Vcc and one for ultra-low Vcc. Those modes may require SRAM cells different to traditional 6T ones. In fact, some existing processors use non-6T SRAM cells. For instance, Intel Atom [31], Intel Nehalem [42] and AMD Llano [50] use 8T SRAM cells for their L1 caches to operate at high (above 1V) and low voltage modes (0.8V). The need for further voltage scaling in future technologies is very likely to consolidate this shift from 6T to 8T SRAM cells. However, using 8T SRAM cells instead of 6T ones has some cost in terms of area, energy and delay as we show later.

In the rest of the thesis, we use three different technologies for cache design: high-performance technology (HPT), low-power technology (LPT) and low-leakage technology (LLT). Each particular technology uses a different operating voltage (Vcc) and different SRAM cells, although our hybrid cache design is not limited to any particular Vcc level, SRAM cell type or process node. The process node used for this study is 32nm.

HPT is devised to provide very high performance despite its energy consumption. Thus, HPT is optimized for 1V operation (HP mode). The most suitable SRAM cell for such operation mode is 6T, given that all cell types operate properly at such Vcc and 6T SRAM cells provide lower energy, area and delay than the other cell types.

LPT is devised to provide high performance (lower than HPT), but low dynamic energy. LPT is optimized for 0.7V operation (also HP mode) to trade-off between both performance and energy. Results in [22, 31] show that conventional 6T SRAM cells are less efficient than 8T ones under iso-robustness condition at 0.7V due to process variations and noise susceptibility thus, making them unsuitable for such Vcc level. The only way to keep robustness at an acceptable level would be increasing cell size significantly, and hence, making 6T SRAM cells less attractive than 8T and 10T ones. Among the other cell types considered (8T and 10T) 8T is the most suitable one due to its higher efficiency in terms of energy, area and performance. Even if 8T cells have one extra bit line and word line with respect to 10T cells, total bit line and word line activity is similar for both 8T and 10T cells. Therefore, this issue is not a disadvantage for 8T cells in terms of dynamic energy.

LLT is devised to minimize the total energy consumption (both dynamic and leakage energy) when performance is not critical. LLT is optimized for 0.35V operation (ULE mode), which is in line with state-of-the-art results [16, 35]. The most suitable SRAM cell for such low voltage is 10T due to its high robustness despite its overheads. 8T SRAM cells could be used if transistor size was increased. However, increasing transistor size for 8T SRAM cells would increase their overheads making them less attractive than 10T ones. Note that both, 8T and 10T SRAM cells have been sized to be reliable in front of process variations at 0.7V and 0.35V respectively.

In summary, all technologies work at 1V HP mode, but HPT is the most convenient (6T cells). LPT and LLT work at 0.7V HP mode. Among those, LPT is the best choice (8T cells). Only LLT is suitable for ULE mode (10T cells).

3.2 Evaluation Methodology

We have chosen a very simple processor architecture with one core and in-order execution, because energy efficiency and minimal design cost are the main drivers for the ultra-low-cost market segment. Our processor configuration resembles a recently fabricated Intel® processor for hybrid Vcc operation although not suited for the ultra-low-cost market [47] (see Table 3.1). Both on-chip L1 data (DL1) and instruction (IL1) caches implement the proposed techniques.

Table 3.1: Processor configuration.

Parameter	Description
Core	in-order
Fetch, Decode, Issue, Commit rate	2 instr/cycle
Window Size	8 entry fetch queue, 8 entry issue queue, 8 entry load/store queue
Functional Units	1 INT ALU (1 cycle), 1 INT Mult/Div (3 cycles mult, 15 cycles div), 1 FP ALU (3 cycles), 1 FP Mult/Div (4 cycles mult, 17 cycles div)
Register file	32 INT (32 bits) + 32 FP (64 bits)
L1 Instruction and Data Cache	8 KB, 8-way (Hybrid: 7+1 configuration), 32 byte per line (2 cycles hit)
Memory	off-chip 8MB SRAM, Miss penalty: 14/16/20 cycles for ULE/HP-LPT/HP-HPT mode
ITLB, DTLB	16 entries fully-associative, Miss penalty: 14/16/20 cycles for ULE/HP-LPT/HP-HPT mode
Branch Predictor	Hybrid 256B Gshare, BTB with 64 entries and 4-way, 16 entry RAS, 4-entry MSHR. Disabled at ULE mode.
Core voltage	0.35V (ULE mode), 0.7V (HP-LPT mode) and 1V (HP-HPT mode)
Technology	32nm

3.2.1 Processor Modeling

We have used MPSim [4] full-chip simulator, an enhanced version of SMTSim [82] extended with power models analogous to those of Wattch [15]. Wattch adds activity counters in the simulator, and estimates the energy consumption of the different structures using the CACTI tool [68]. The main processor units that Wattch models are:

- Array Structures: Data and instruction caches, cache tag arrays, all register files, translation lookahead buffers (TLB), branch target buffer (BTB), register alias table, branch predictors and large portions of the issue queue and the load/store queue.
- Combinational Logic and Wires: Functional units, dependency check logic at decode stage, issue queue selection logic and result buses.
- Clocking: Clock buffers, clock wires, etc.

Note that the majority of resources in our simple single-core processor is devoted to SRAM array-like structures (e.g., only L1 caches occupy more than 50% of the total on-chip area). All those structures are modeled with CACTI tool. In order to understand the impact of different techniques proposed in this thesis on the whole chip, we have incorporated a new features into the CACTI tool. The following section provides details about cache modeling and describes how the model is validated using HSPICE.

3.2.2 Cache Modeling

L1 cache memories have been modeled using CACTI 6.5, a flexible and accurate cache delay, energy, power and area simulator [68]. The technology node considered is 32nm. To support different operating modes, we have extended CACTI tool with models of other alternative SRAM cells (e.g., 8T and 10T) that are able to operate at low and NST voltages. Given that 6T SRAM cells are already modeled in CACTI tool, we have extended it with delay, power and area models for 8T and 10T SRAM cells by adapting capacitances, resistances and geometry. Here, we provide more details about dynamic and leakage power models for caches including 8T and 10T SRAM cells.

3.2.2.1 Cache Modeling Using Single-Ended 8T SRAM cells

The model for 10T SRAM cells is anologous to that already implemented for 6T SRAM cells. In fact, implementation of the 10T cells is quite similar to 6T ones due to its fully differential architecture. However, the architecture of the 8T SRAM cell is not differential because of the separated write and read ports. Extra bitlines are modeled by including the proper capacitances and resistances in parameter calculations. These values depend on the transistor sizes, which have been chosen to provide high bitcell stability and operational reliability at low supply voltage [46, 54].

Read and write port separation requires a second set of word-line (WL) drivers, which adds to the area of the array. Whereas the write WL (WWL) driver is largely similar to standard WL drivers used in 6T arrays due to similar capacitive loading, the read WL (RWL) driver can be significantly reduced in size due to the single-ended cell read stack. At the same time, with separate RWL and WWL signals, a read/write multiplexer is no longer needed at the bit line level. Now it is moved to address decode logic to enable RWL and/or WWL during read or write operation respectively. Precharge circuitry is changed because now it requires only one PMOS transistor for read bit-line (RBL) [19].

Sense amplifiers are single ended in case of 8T SRAM cells due to the single bitline for read operations. For the sake of simplicity, we use the same differential sense amplifier suitable for 6T and 10T SRAM cells with a reference voltage tied to one input [83]. The reference voltage is set sufficiently below Vcc to sense the read of a logic 1 correctly. Increased delay and bitline swing when sensing a logic 0 is recovered by optimal transistor sizing of the stacked read buffer of the 8T cells (T3 and T4 in 8T cell in Figure 1.4). Sense amplifiers are shared across adjacent array columns in order to improve array efficiency. This introduces an additional cost in dynamic power, since multiple columns are read in a single event, and some performance, since the multiplexer has some (little) delay [19]. Nevertheless, we have taken such decision to have a low complexity implementation.

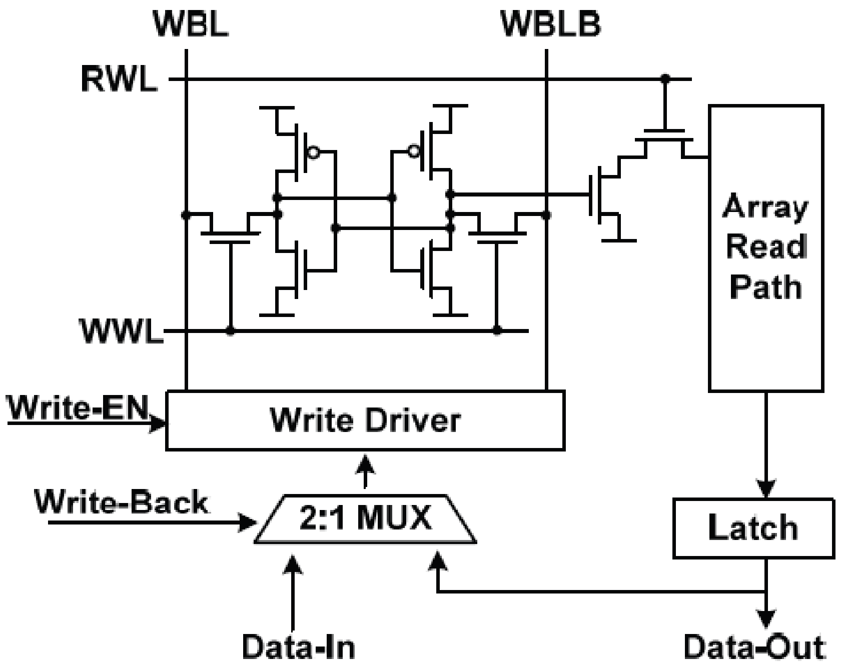


Figure 3.1: Write-back scheme for 8T-based caches [67].

We assume a set-associative cache organization in our study since such organization is the most common case in the embedded arena. A set-associative cache using an 8T SRAM requires write-back scheme to support column selection [67]. In a set-associative cache, multiple cache blocks reside in a row of an array, and one single block is read or written per cache access. The unselected cells in a row have to preserve the stored data while a WL is driven high. In conventional 6T SRAM, this operation is supported by applying the same bias conditions as read operation to unselected columns (known as half-selection [48]). However, in 8T SRAM, the half-selection condition would more easily disturb the stored data due to the presence of strong write access transistors (or weak pull-down transistors in cross-coupled inverters). Since the data in unselected cache blocks in a row have to be preserved during the write operation, write-back must be used, and hence, the cache can support only single-port operation. The basic operation of write-back schemes is to always read the unselected columns prior to performing any write operations. Data read from unselected columns are latched in write circuits and then, merged with new data for selected columns. Finally, the merged data are written back to the entire row at a time. It is very important to note that such a technique cannot support multi-port capability of an array since a read port must always be dedicated to the write operation. In our implementation, we use the write-back scheme depicted in Figure 3.1 due to its low complexity [67].

3.2.2.2 Dynamic and Leakage Power Modeling

Following the same philosophy for dynamic power modeling that is already used in CACTI, our model tracks the physical capacitance of each stage of the cache model and calculates dynamic power consumed at each stage. Basically, cache dynamic power dissipation is comprised of wordline capacitance dissipation, bitline capacitance dissipation and short-circuit power consumption. Since capacitance plays an important role for dynamic power, we take into account the following capacitances: parasitic capacitances of transistors in SRAM cell, capacitances of the access transistors, capacitance of a precharge transistor and capacitances of a column select transistor and a wordline driver. Capacitances of the wordline/bitline wires and wires in decoders are modeled as a distributed RC network.

Given that the impact of process variations is high for the 32nm technology node, especially at NST Vcc (ULE mode), our cache leakage power model is updated to take into account process variations. We model random within-die variations in threshold voltage (V_{TH}) using the analytical model proposed in [69]. In particular, the cache is decomposed into smaller building blocks and total leakage power is the sum of leakage power in each block. Leakage current of one block, including within-die variations (I_{leak}), is then estimated as follows:

$$I_{leak}^{p} = \frac{I_{leak-p}^{mean} w_p}{k_p} \frac{1}{\sigma_p \sqrt{2\pi}} \int_{V_{TH_{min}}}^{V_{TH_{max}}} \mathrm{e}^{-\frac{(V_{TH}-\mu)^2}{2\sigma_p^2}} \mathrm{e}^{-\frac{(\mu - V_{TH})}{a}} \, \mathrm{d}V_{TH} \tag{3.1}$$

$$I_{leak}^{n} = \frac{I_{leak-n}^{mean} w_n}{k_n} \frac{1}{\sigma_n \sqrt{2\pi}} \int_{V_{TH_{min}}}^{V_{TH_{max}}} \mathrm{e}^{-\frac{(V_{TH}-\mu)^2}{2\sigma_n^2}} \mathrm{e}^{-\frac{(\mu - V_{TH})}{a}} \, \mathrm{d}V_{TH} \tag{3.2}$$

$$I_{leak} = I_{leak}^{p} + I_{leak}^{n} \tag{3.3}$$

where w_p and w_n are the total PMOS and NMOS devices widths in the block; k_p and k_n are factors that determine the fraction of PMOS and NMOS devices widths that are in off state; μ and σ are mean and standard deviation of V_{TH}. a is equal to $n\phi_t$, where ϕ_t is the thermal voltage and n = $1 + (C_d/C_{ox})$. According to the[69], we assumed that on an average half of the PMOS/NMOS devices are in off state, so k_p = k_n = 2. We have not considered temperature dependency in our analysis. Instead, a fixed temperature of 100°C is assumed. I_{leak}^{mean} stands for the leakage current of a block with mean V_{TH} and can be calculated by multiplying the device width and basic leakage per gate (I_{leak}^{gate}). I_{leak}^{gate} is defined as:

$$I_{leak}^{gate} = \beta \mathrm{e}^{b(v_{dd}-V_{dd0})} V_t^2 (1 - \mathrm{e}^{-\frac{V_{dd}}{V_t}}) \mathrm{e}^{\frac{-|V_{TH}|-V_{off}}{nV_t}}. \tag{3.4}$$

We refer the reader to [92] for a description of the terms. According to [69], integrals in equations (3.1) and (3.2) can be simplified, so I_{leak} can be expressed as:

$$I_{leak} = \frac{I_{leak-p}^{mean} w_p}{k_p} \mathrm{e}^{\frac{\sigma_p^2}{2\lambda_p^2}} + \frac{I_{leak-n}^{mean} w_n}{k_n} \mathrm{e}^{\frac{\sigma_n^2}{2\lambda_n^2}} \tag{3.5}$$

where λ_p and λ_n are constants that relate channel lengths of PMOS and NMOS transistors to their corresponding subthreshold leakage current.

Each SRAM cell is sized by using the analysis based on importance sampling proposed by Chen et al. [22] assuming 6σ random variations in V_{TH} for high (1V), low (0.7V) and ultra-low voltage (0.35V) respectively, considering read, write and hold failures in 32nm technology node. Depending on the cache size and target cache yield, all SRAM cells are sized accordingly. More details will be given in the next chapters when explaining implementation details for different cache designs proposed in this dissertation. Impact in terms of area has been also considered for 8T and 10T SRAM cells and their associated circuitry. The smallest rectangle where the cache fits is chosen in area calculation to keep layout regularity.

Beside new SRAM cell models, we have added some new features into the CACTI tool to make it more flexible and convenient. Several hybrid cache microarchitectures have been implemented using heterogeneous SRAM cell types at a coarse granularity. For example, hybrid cache designs where different cache ways are implemented with different SRAM cell types are allowed. Also, cache tag or data words can be extended with some additional bits (e.g., check bits, valid bits, etc.), considering their area, delay and power impact. All those features are essential for efficient and accurate evaluation of the proposals in this thesis.

3.2.2.3 HSPICE Validation

The accuracy of the power model implemented in CACTI is validated using HSPICE. We have first created the 10T SRAM cell model using the low-power 32nm Predictive Technology Model (PTM) [93] with nominal threshold voltages for NMOS and PMOS transistors of V_{TH_n} = 350 mV and V_{TH_p} = -320 mV respectively, and standard deviation of V_{TH} of a 24 mV (30 mV) for each NMOS (PMOS) transistor. Then we have modeled cache SRAM arrays (i.e. sub-arrays of the hybrid cache), comprised of one SRAM cell type/size (e.g., 6T, 8T, 10T, etc.) including all peripheral circuits such as decoders, wordline and output buffers, pre-charge circuitry, column multiplexer, etc. The SRAM array is created by replicating a single SRAM cell as many times as needed (MxN times where M and N stand for the number of rows and columns respectively) instead of creating MxN different SRAM cells. Although some accuracy is lost, this is not an issue because power and delay are dominated by bitlines and wordlines rather than SRAM cells, and bitlines/wordlines are accurately modeled. On the other hand, SRAM cell simplification reduces drastically simulation time, which would be in the range of many hours (or even days) otherwise. We have compared dynamic and leakage power and read/write access time of the SRAM array with the corresponding SRAM array in CACTI.

We have created digital input vectors for several clock cycles to perform SRAM array write and read operations. We exercise the array with a reasonable number (i.e. 200) of different write and read patterns in order to measure read, write and short-circuit[1] dynamic and leakage power. Finally, we have measured read and write access times for the created SRAM array. Read access time is measured as the difference between the time the address bit's voltage reaches Vdd/2 and the time the output (32 bit data value) of the read buffer reaches 90% of its final value. However, write access time is measured as the difference between the time the address bit's voltage reaches Vdd/2 and the voltage of bitnodes inside the cell reach 90% of their final values. Values obtained are averaged and those values are used as the results of HSPICE simulations.

Values obtained from CACTI for the corresponding SRAM array show to be accurate

[1]The value written into the bitcell during write operation may be different from the previously stored value. In that case, there will be energy consumption to toggle the cell bit. During toggling, there is a small period when both the PMOS and NMOS of the cell inverter are conducting which causes short-circuit power.

within 7% variation (on average) to the reference HSPICE models for all metrics considered. The maximum error observed is 18%. The main reason for this discrepancy is the fact that the authors of the analytical models for leakage power [69] used in CACTI make some empirical assumptions in their equations for leakage current.

3.2.3 Rest of the Structures

Beside L1 caches, the rest of resources in our simple single-core processor are mainly devoted to other SRAM array-like structures such as register files, BTB, TLBs, etc. We use our enhanced CACTI version to model all those structures. All SRAM arrays except L1 caches have been implemented using 10T cells so they operate properly at any voltage level considered. Part of our future work consists of devising more efficient designs for those components so that their energy consumption can be reduced at HP mode.

Off-chip SRAM memory based on 10T cells is also modeled. It uses the same Vcc as the core in all operation modes. The relative memory latency is low given the low speed of the core, the small memory size (typically few MBs) and its high integration with the processor itself. Memory power and energy are measured with CACTI and these values are included in our results.

3.2.4 Operating Modes

Our system has three distinct operating modes: HP-HPT, HP-LPT and ULE modes. HP-HPT corresponds to the HP mode implemented with HPT, whereas HP-LPT corresponds to the HP mode implemented with LPT. ULE mode is always implemented with LLT. Thus, we have set Vcc to 1V, 0.7V and 0.35V for HP-HPT, HP-LPT and ULE modes respectively. Operating at different voltage levels requires different operating frequencies for each voltage. Thus, we have set operating frequencies to 1GHz for HP-HPT, 300MHz for HP-LPT and 5MHz for ULE, which is in line with state-of-the-art results [21, 22, 47, 90].

3.2.5 Cache Access Policy

We have considered the conventional parallel access policy as default access policy in all experiments. Besides parallel tag/data access we have considered a serial tag/data access policy as an alternative for low power consumption. In that case, tag comparison is followed by data read and thus, only the cache way hit (if any) is read out form the

Table 3.2: MediaBench benchmarks used in this thesis.

Benchmark	Description	Input	Language
adpcm_c	audio	clinton.pcm	C
adpcm_d	audio	clinton.pcm.adpcm	C
epic_c	image	test_image.pgm	C
epic_d	image	test_image.pgm.E	C
g721_c	audio	clinton.pcm	C
g721_d	audio	clinton.g721	C
gsm_c	audio	clinton.pcm	C
gsm_d	audio	clinton.pcm.gsm	C
mpeg2_c	video	test2.mpeg	C
mpeg2_d	video	test.par	C

data array [37]. Compared to the parallel access, where all cache ways are accessed on each access, the serial tag/data scheme avoids unnecessary way accesses and hence, reduces power consumption. However, cache access is then divided into two phases, first tag and then data access, thus increasing cache access time significantly. This may translate into an unaffordable performance degradation, especially at HP-HPT and HP-LPT modes, where performance is a primary concern. Although tag and data accesses could be pipelined, this provides almost no gain in our simple in-order processor with single instruction issue width. Further, execution time increase has also a negative effect on leakage. Therefore, in the rest of this Thesis, we keep using the parallel access policy.

3.2.6 Benchmarks

To the best of our knowledge, a set of benchmarks specific for the domain that we target does not exist. We have chosen MediaBench [58], because they fit very well the expected needs of the ultra-low-cost segment: an abundant data processing during HP-HPT/HP-LPT mode and relatively small workloads at ULE mode [38, 78, 84]. For instance, sensor applications which monitor wind, sea level, temperature, tsunamis, etc., should be data intensive at HP-HPT/HP-LPT mode while the amount of data to be processed at ULE mode should be much smaller. We classify benchmarks into two categories, depending on the cache requirements: (i) *SmallBench* - workloads fit into very small cache sizes (e.g., 1KB) due to small data volume (adpcm_c, adpcm_d, epic_c and epic_d) and (ii) *BigBench* - larger cache space is required to fit the workload due to large data volume (g721_c, g721_d, gsm_c, gsm_d, mpeg2_c and mpeg2_d). *SmallBench* benchmarks are used during ULE operation whereas *BigBench* ones are used during HP-HPT and HP-LPT operation.

Due to lack of specific benchmarks resembling full applications for the target domain, we built an artificial application in order to report overall energy savings for the whole application lifetime. Essentially, we built the application following the scheme of a typical application: the infinite loop executing the "ULE mode code", the condition check to enter HP mode, executing the "HP mode code" and entering again to the ULE mode routine with the infinite loop, as shown in Figure 1.2. We assume in our application that "ULE mode code" is a program from *SmallBench* benchmarks suite whereas "HP mode code" is a program from *BigBench* set. Note that we did not model transition from HP mode to ULE mode and vice versa. However, performance impact of such transitions should be small because they occur seldom. In order to report overall energy for the whole application lifetime, we take average energy results for *SmallBench* programs and assume they execute 99% and/or 99.9% of the time, and average results for *BigBench* programs and assume they execute 1% and/or 0.1% of the time.

Each program is compiled with the *-O2 -non_shared* options using DEC Alpha AXP-21264 C/C++ compiler and executed using the reference input set. Table 3.2 shows the applications used in this thesis. All programs are executed until completion.

Chapter 4

Hybrid Cache Ways Designs

This chapter focuses on the hybrid L1 cache architecture, which is the first step towards the design of hybrid microarchitectures in the ultra-low-cost market segment which we target. In particular, next we describe our first proposal, which is used along this Thesis as the baseline cache for the rest of our proposals as well as for comparison purposes.

4.1 Introduction

In this chapter, we present new, deterministic, single-Vcc domain hybrid L1 cache architectures, which satisfy all stringent needs of our target market. The proposed cache architecture combines heterogeneous SRAM cell types by splitting the cache into two sections: (i) some cache ways are made of the SRAM cells optimized for one particular Vcc level (e.g., high or moderate Vcc) and (ii) the rest of the cache ways are made of the SRAM cells optimized for another Vcc level (e.g., NST Vcc). In order to conduct the research of this chapter, we provide a comprehensive study varying several critical parameters such as SRAM cell type, operation voltage, cache size, associativity and line size. We show that our hybrid caches can efficiently and reliably operate across a wide range of voltages, consuming little energy at ULE mode as well as providing high performance with small overheads at HP mode, as required for our target market. We show also that strong timing guarantees are achieved at both modes due to the deterministic behavior of the proposed cache designs at any voltage considered. In particular, reliability of *all* cache space is high enough, so that no cache space is disabled and WCET estimates can be obtained.

The rest of this chapter is structured as follows. Section 4.2 describes the proposed cache architecture. In Section 4.3 we present and discuss experimental results for different L1 cache designs and configurations. Finally, Section 4.4 summarizes our main findings.

4.2 Proposed Cache Architecture

Based on the fact that most L1 caches in existing chips are set-associative, we have chosen this type of organization as the target of our study, although significant parts of our study can be easily reused for direct-mapped and fully-associative caches. Several hybrid cache designs have been implemented using heterogeneous SRAM cell types at a coarse granularity. Implementation details are presented in Chapter 3. Two different cache configurations are considered and analyzed:

- *Non-Hybrid.* In the first configuration, caches are implemented with the same technology type for all arrays, so we have pure high-performance, low-energy or ultra-low-energy optimized configurations.

- *Hybrid Cache Ways.* In the second configuration, some cache ways have been implemented with one technology and the rest of them with another. The minimum voltage level for this configuration is dictated by the cell able to operate at the lowest voltage level. For example, if we consider a 4-way cache configuration with 2 LPT and 2 LLT cache ways as shown in Figure 4.1, we can see that operation voltage can be 1V, 0.7V or 0.35V. 1V provides high-performance operation and all cache ways work; 0.7V provides moderate-performance, low-energy operation and all cache ways still work; and finally, 0.35V provides ultra-low-energy operation where cache ways implemented with 8T SRAM cells must be turned off. During 0.35V-operation mode (ULE mode) data processing is expected to be minimal [78] and workloads are much smaller than during 0.7V/1V-operation mode (HP mode). Workload discrepancy across HP and ULE modes justifies reducing the hardware resources to complete a given computation at ULE mode. Since HPT and LPT ways would experience many faults at NST Vcc and thus would not provide reliable operation, we simply turn them off at ULE mode. This approach is used for all cache ways that do not operate reliably at the current operation mode. Turning off some cache ways may have some impact on performance. However, as long as at least one cache way is turned on, the cache can operate properly. All cache ways are enabled at HP mode in order to use full cache space for high performance. Even LLT ways remain active despite their inefficiency at high Vcc, because they reduce slow and energy-hungry off-chip memory accesses [61]. Eventually, LLT ways can be turned off and extra HPT or LPT ways could be in place to replace LLT ones

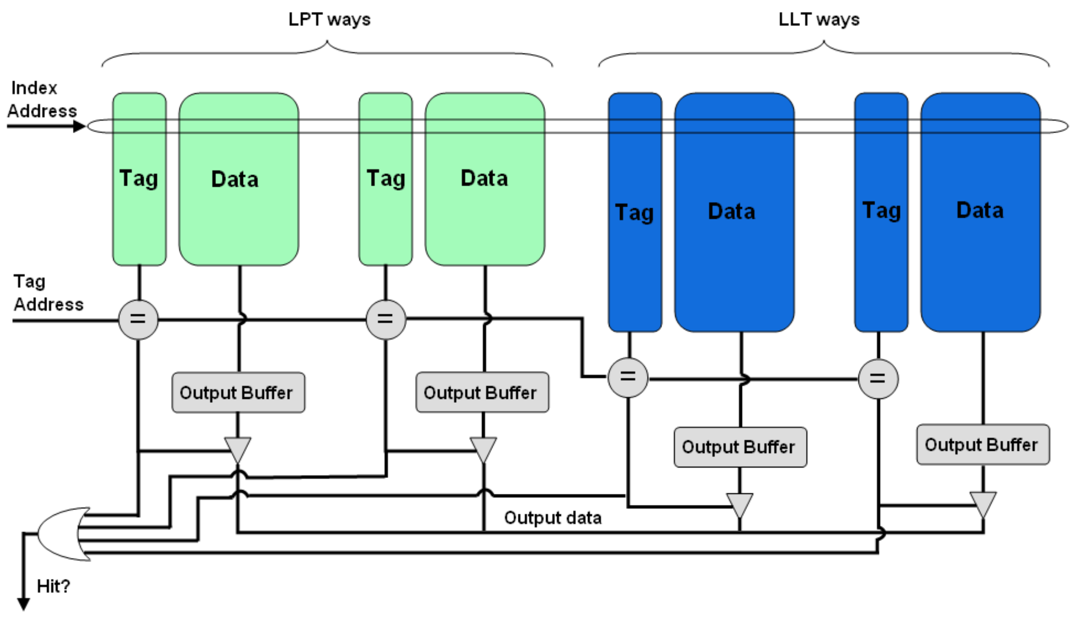


Figure 4.1: An example of a Hybrid Cache Ways design.

at HP mode. However, such approach substantially increases area and, even in that case, using LLT ways at HP mode still can provide significant energy savings.

Note that all configurations provide strong guarantees on available cache size, number of sets, number of ways, etc., at each voltage level considered. Configurations implemented either with a single SRAM cell type or with hybrid cache ways have been evaluated. Those cache designs are analyzed both in isolation and in the context of a processor. Performance and energy results are presented in Section 4.3.

4.2.1 Changing Operation Mode

The processor, and therefore the cache, is designed to support two distinct operation modes. The software running on top is responsible for deciding when to switch modes based on the observed inputs by using specific instructions. Figure 1.2 shows the typical code structure for applications to be run on top of these processors. On a ULE to HP mode transition, HPT/LPT ways are activated and Vcc raised. The latency of such operation depends on the time required to change Vcc and frequency and activate cache ways. On a HP to ULE mode transition, first dirty HPT/LPT cache lines (if any) must be written back to memory. Then those ways can be desactivated and Vcc and frequency can be scaled down. We consider gated-Vdd [71] to turn off cache ways. Using a PMOS gated-Vdd transistor significantly reduces the required transistor width, resulting in negligible area and power overheads, as explained in [71].

In other words, the user or compiler can take advantage of the input-activity characteristics through software configuration to switch the mode, but this problem is out of scope of the paper [80]. Since HP mode occurs seldom (less than 1% of the time [38, 78, 84]), performance and power overheads due to mode switching are expected to be negligible.

4.3 Evaluation Results

In this section, we present results in terms of performance, energy and area for different L1 cache designs. All results are gathered using the experimental environment and methodology described in Chapter 3. Each SRAM cell is sized by using the analysis based on importance sampling proposed by Chen et al. [22] assuming 6σ random variations in V_{TH} for high (1V), low (0.7V) and ultra-low voltage (0.35V) respectively, considering read, write and hold failures in 32nm technology node. 8T and 10T SRAM cells are sized to match the same failure rate when operating at low (0.7V) and ultra-low voltage (0.35V) respectively as for the 6T cells when operating at high voltage (1V) targeting a 99.9% cache yield.

Reported results are divided into two groups. First, we present and discuss results for the proposed cache designs in isolation. Then, we present results for the whole chip when such cache designs are deployed in a single-core processor.

4.3.1 Non-Hybrid and Hybrid Caches Sensitivity Study

Next we present delay, dynamic energy, leakage power and area results derived from our custom-modified CACTI tool for non-hybrid and hybrid caches in isolation. We have chosen a baseline cache configuration in line with current trends in the embedded arena: 16KB 4-way cache with 32 B/line, 1 read + 1 write port (for 6T and 10T bitcell 1 read/write port) and 1 bank. The technology node used is 32nm. We first present results for pure 6T, 8T and 10T baseline caches respectively. Later, we study the impact of several parameters such as cache size, associativity and line size for several hybrid configurations using different technologies for different cache ways. Specifically, we consider designs where *half* of the ways are implemented with a particular technology and the other *half* with another technology. Other combinations are feasible, but do not provide

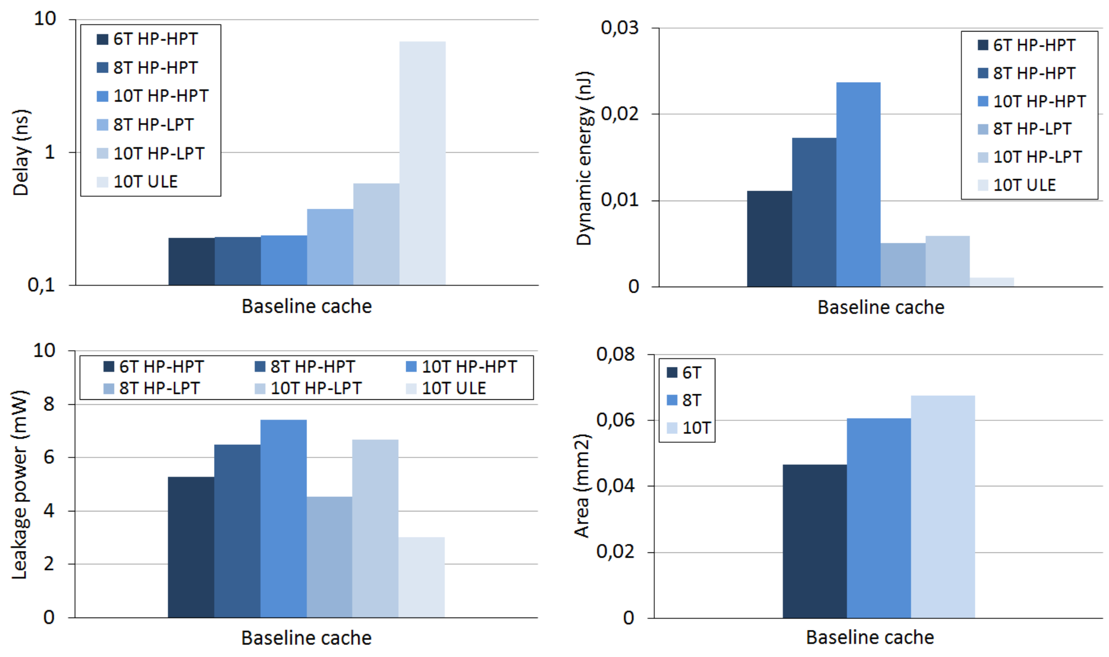


Figure 4.2: Delay (in logarithmic scale), dynamic energy, leakage power and area for Non-Hybrid baseline caches (16KB, 4-way, 32B/line).

further insights[1].

4.3.1.1 Results for Non-Hybrid Caches

In this set of experiments, all cache ways are implemented with the same type of cells, so we have pure high-performance, low-energy or ultra-low-energy optimized configurations. There are 6 different configurations to be studied:

- HP-HPT mode: 6T, 8T and 10T,
- HP-LPT mode: 8T and 10T,
- ULE mode: 10T.

Figure 4.2 depicts the comparison between pure 6T, 8T and 10T implementations when operating at different voltages (i.e. HP-HPT, HP-LPT and ULE mode). In particular, we observe that 6T, 8T and 10T caches have similar delay[2] at HP-HPT. Access time

[1]We may use the same hybrid configuration as in the full processor evaluation, where 1 way is of one type and the remaining ones are of another type, but that would limit or distort the study of some parameters such as cache associativity because of the constraints to keep the ratio of ways for each technology type.

[2]Note that delay is in logarithmic scale.

variation for 8T and 10T caches at 1V is less than 3% with respect to the 6T implementation. However, 8T and 10T designs exhibit higher delay discrepancy at HP-LPT mode (around 5%). Note that 6T designs are not considered at this mode due to the high error rate expected for 6T SRAM cells at 0.7V. Finally, 10T implementation at ULE mode shows much higher delay than at HP-HPT and HP-LPT modes due to the exponential delay increase at NST regime. Note also that 6T and 8T designs are not considered at ULE mode due to their unreliability at NST voltages. We can conclude that delay differences exacerbate across cell types and voltage levels, except when voltage is high (1V).

We can also observe that in terms of dynamic energy, as expected, 8T and 10T SRAM cells are less efficient than 6T ones at HP-HPT mode due to additional transistors in their architectures. A similar observation can be derived for 10T SRAM cells with respect to the 8T ones at HP-LPT mode due to the same reason. However, the relative energy increase is lower than that at HP-HPT. Relative trends for leakage power across cell types match those of dynamic energy at HP-HPT mode. The relative difference between 8T and 10T at HP-LPT mode grows with respect to that at HP-HPT. Overall, the most energy-efficient SRAM cells are 6T for HP-HPT, 8T for HP-LPT and 10T for ULE mode. This trend shows a high correlation with the area required for the particular cell type used for each cache configuration as depicted in Figure 4.2.

4.3.1.2 Hybrid Cache Ways Sensitivity Study

As explained before, we consider Hybrid Cache Ways designs where half of the cache ways are implemented with a given SRAM cell type and the other half with another. There are 4 different configurations to be studied:

- HP-HPT mode: 6T+8T, 6T+10T and 8T+10T.
- HP-LPT mode: 8T+10T.

Note that this particular type of caches can also work at lower voltages. For instance, a 6T+10T cache can operate at 0.7V or 0.35V by disabling 6T cache ways. In that case, delay, dynamic energy and leakage are very similar to those of a non-hybrid cache with half the size and half the number of cache ways. In the example of a 6T+10T 16KB 4-way cache, operating below 1V would require disabling 6T ways and thus, the cache would behave very similarly to a 10T 8KB 2-way cache.

Based on the baseline cache configuration, we study the sensitivity of hybrid caches when varying some parameters as follows:

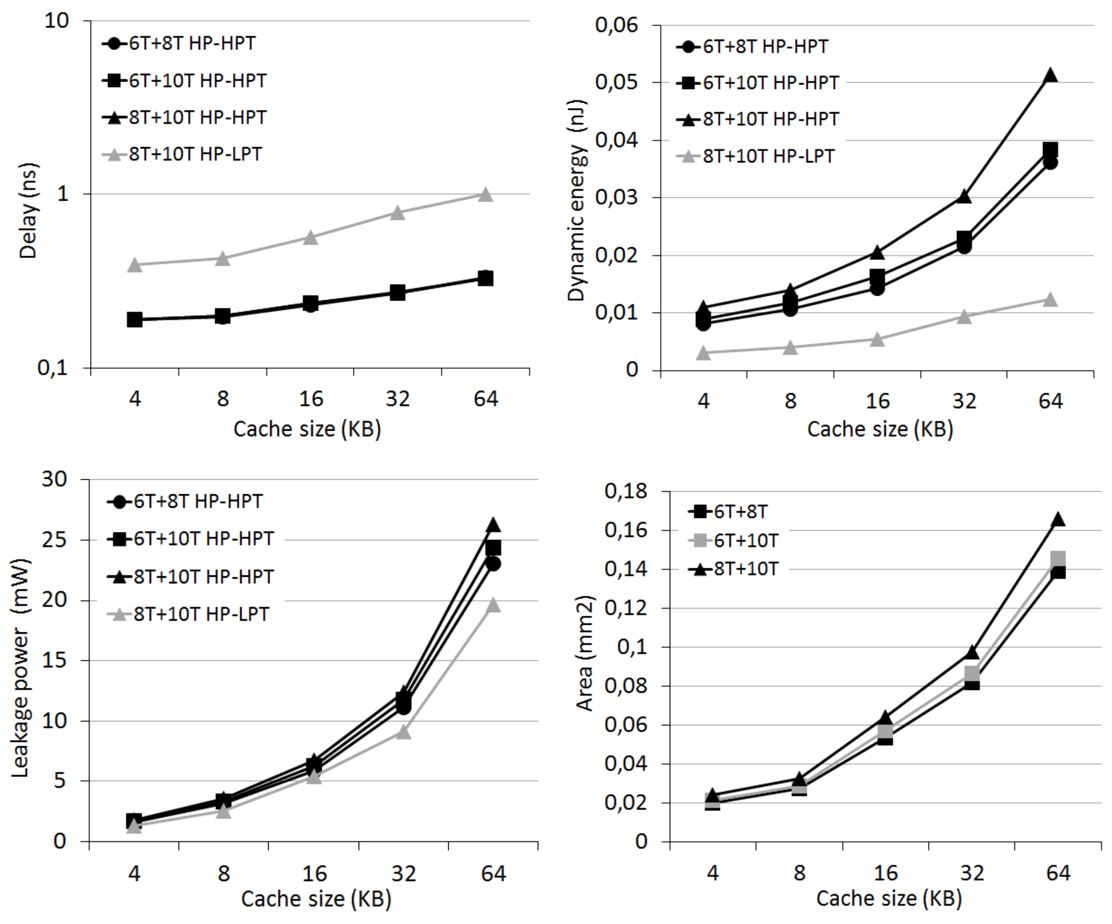


Figure 4.3: Delay (in logarithmic scale), dynamic energy, leakage power and area as a function of the cache size in Hybrid Cache Ways designs.

1 Cache size: 4KB, 8KB, 16KB, 32KB and 64KB,

2 Associativity: 2-way, 4-way and 8-way,

3 Line size: 8B/line, 16B/line, 32B/line and 64B/line.

Figure 4.3 shows results for different cache sizes. Delay is basically dominated by the slowest cell type. Thus, delay of a cache with half of the ways of type xT and half yT roughly matches that of a cache with all cache ways of the slowest type among xT and yT. Similarly, dynamic energy, leakage power and area resemble with high accuracy the average for non-hybrid caches. For instance, leakage power of a 6T+10T cache is within 3% that of average leakage for a pure 6T and a pure 10T cache.

Figure 4.3 shows that dynamic energy, leakage power and area grow quite linearly with cache size. Delay growth instead, is sublinear with cache size.

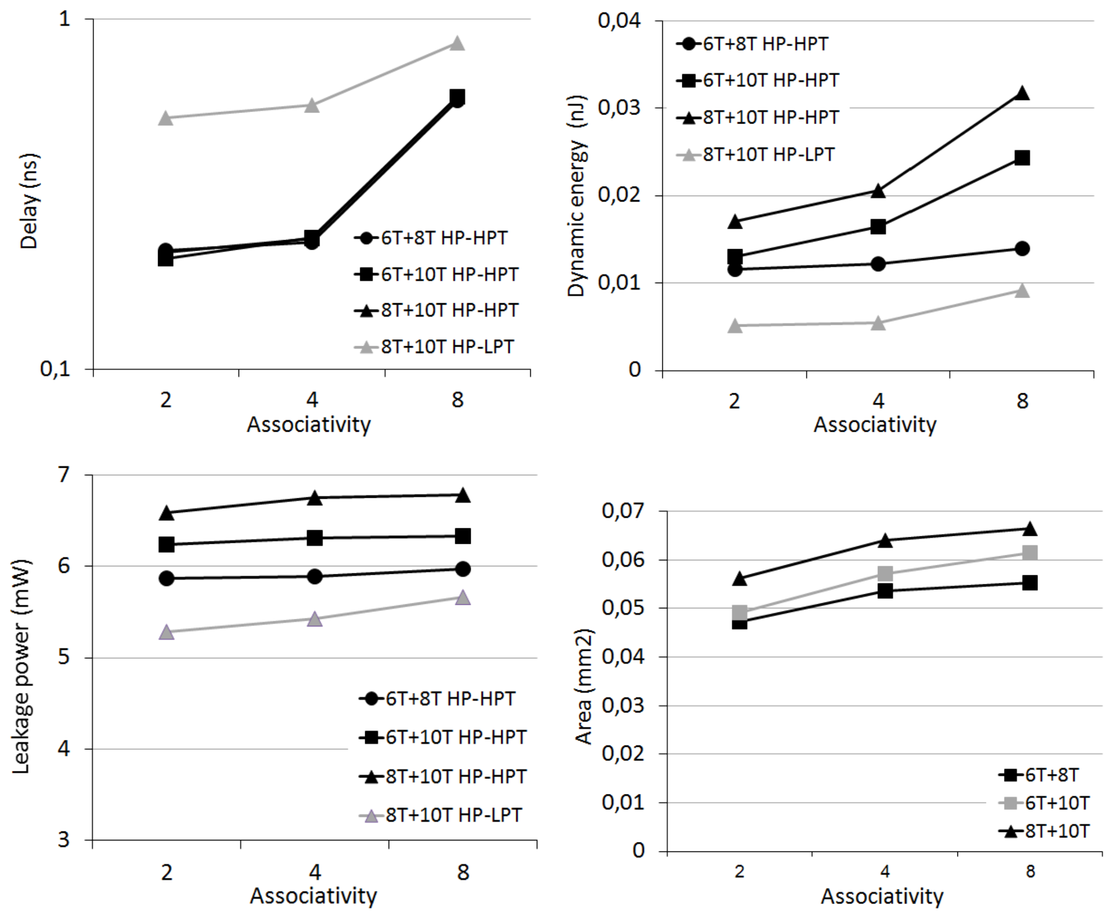


Figure 4.4: Delay (in logarithmic scale), dynamic energy, leakage power and area as a function of the associativity in Hybrid Cache Ways designs.

Hybrid Cache Ways configurations are also evaluated for different associativities. Results are shown in Figure 4.4. Delay increases noticeably when moving from 4 to 8 cache ways. Since delay is very similar for all configurations at HP-HPT mode, trends are very similar to those of non-hybrid configurations for the different associativity values considered. As stated before, delay at HP-LPT mode is mostly dominated by the slowest cell type. This effect is also observed for the different cache associativities studied.

Leakage power and area show linear dependence on the number of SRAM cells of each type used as described before. Dynamic energy, however, grows with associativity due to the larger number of ways to be looked up. Moreover, if 10T cells are used, dynamic energy grows because their size impacts the size of wordlines and bitlines. Therefore, associativity has significant impact only on delay and dynamic energy.

Finally, we have also studied the impact of varying the line size. However, variations in terms of delay, dynamic energy, leakage power and area are negligible and do not show any meaningful trend across technologies or cell types. Therefore, plots have not been included. Line size must be chosen considering only its impact in performance due to the amount of spatial locality available in the workloads and the bandwidth requirements to fetch long cache lines.

Hybrid Cache Ways designs offer different trade-offs to those of the non-hybrid designs. They are not as efficient as non-hybrid ones at high voltage (e.g., 6T+10T versus pure 6T), but allow ultra-low energy and reliable operation at low and/or ultra-low voltage by disabling parts of the cache (e.g., disabling 6T cache ways). Although we have studied simple configurations with half of the ways of one type and half of another, many different configurations are feasible. Thus, Hybrid Cache Ways are a promising starting point to develop efficient hybrid microarchitectures, as shown later.

4.3.2 Processor Evaluation

After analyzing results for the caches in isolation, in this section, we extend our analysis towards the whole chip. All results correspond to the baseline configuration presented in Table 3.1. We have run simulations on the described processor when using different non-hybrid and hybrid caches at different voltage levels (HP-HPT, HP-LPT and ULE modes). Cache configurations considered must provide full cache space at HP mode (HP-HPT or HP-LPT) and at least one cache way at ULE mode . Therefore, 6T+8T and 6T+10T configurations are not considered at HP-LPT. The cache configurations considered are as follows:

- HP-HPT: 6T, 8T, 10T, 6T+8T, 6T+10T and 8T+10T.
- HP-LPT: 8T, 10T and 8T+10T.
- ULE: 10T, 6T+10T and 8T+10T.

First, we evaluate different cache configurations and show performance, energy and power results. Along with this, we study the impact of cache size and memory latency on performance, energy and power. Then we study total on-chip energy distribution in terms of dynamic and leakage energy for different cache configurations at different operation modes. Finally, we compare performance, energy and power across different operation modes for all hybrid and non-hybrid cache configurations considered. As stated before, we use *SmallBench* benchmarks at ULE mode whereas *BigBench* ones are used at HP-HPT and HP-LPT modes for all experiments.

4.3.2.1 Metrics

In order to provide meaningful results, we consider the following metrics for each benchmark:

- execution time,
- dynamic energy per instruction,
- leakage power per cycle,
- energy-delay product (EDP).

In EDP calculation, we consider total energy which is the sum of dynamic and leakage energy.

4.3.2.2 Results for Different Cache Configurations

Next, we present results in terms of execution time, dynamic energy per instruction, leakage power per cycle and EDP for the processor configuration presented in Table 6.2.2 when using different cache configurations. We have chosen a 4KB/8KB/16KB 8-way cache for all non-hybrid configurations. In the case of hybrid designs, a 7+1 hybrid configuration has been considered, where 7 ways are implemented with either HPT or LPT and 1 way with LLT. At HP-HPT/HP-LPT mode, all cache ways are enabled and 4KB/8KB/16KB cache size is available whereas at ULE mode 7 ways are turned off, thus providing 512B/1KB/2KB of cache space. We have studied other hybrid configurations with different associativities and number of HPT/LPT/LLT ways at different operation modes, but relative trends hold across different configurations and no meaningful variation is observed in any metric, so we omit details for those configurations. Since execution time varies noticeably across benchmarks, *results have been normalized with respect to the 4KB pure 6T, 8T and 10T cache configurations at HP-HPT, HP-LPT and ULE mode respectively for each benchmark.*

Note that the operating frequency is identical for all configurations under the same operation mode (i.e. HP-HPT, HP-LPT or ULE) despite the fact that caches may have different latencies. However, adapting the frequency to the cache latency would require redesigning all the remaining components to fit the new cycle time. Thus, we have decided to keep the same frequency across configurations. Similarly, to avoid further sources of variation in the results, all SRAM arrays, except L1 caches, have been implemented with 10T SRAM cells, so they operate properly at any voltage level. Note that L1 caches are the main energy consumers in our simple core. Thus, varying the type of SRAM cells used at the same mode in the core has little impact on the trends observed (i.e. power).

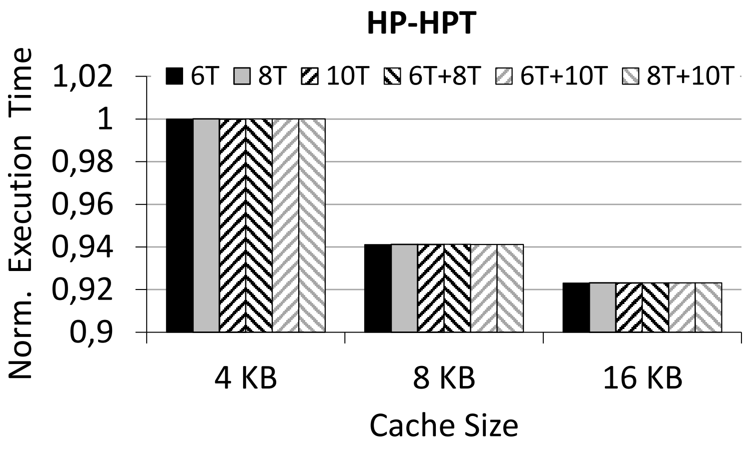
Relative energy and performance for different benchmarks (*BigBench* suite for HP-HPT and HP-LPT modes and *SmallBench* suite for ULE mode) are quite similar, thus indicating that the impact of the cache configuration is not particularly dependent on the program run. This is so because caches are the main energy consumers and access frequency is not drastically different across benchmarks, so effects on different sources of energy on each benchmark are relatively similar. Therefore, results are presented in the form of normalized average across benchmarks.

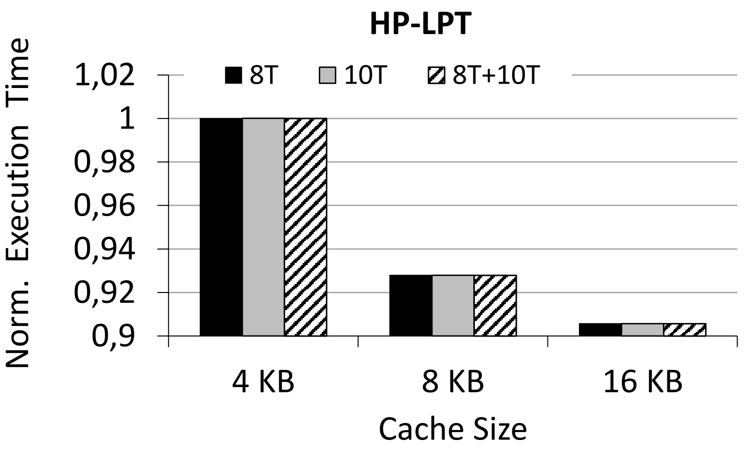
Figures 4.5, 4.6, 4.7 and 4.8 depict normalized average results when running *Big-Bench* and *SmallBench* benchmarks at HP-HPT and HP-LPT modes, and ULE mode respectively. We have observed the following for each metric:

1) *Execution Time.* Normalized execution time for all configurations at HP-HPT and HP-LPT modes is similar because operating at such modes does not require turning off any cache way, so all the ways for all configurations are always turned on. There is a small degradation in normalized execution time at ULE mode (2% on average) due to the smaller cache size in 6T+10T/8T+10T hybrid configurations (6T/8T ways are disabled) which leads to more memory accesses to serve extra misses. Also, relative trends hold across different cache sizes. Speedup for 16KB caches with respect to 4KB ones (e.g., 7.8% at HP-HPT) is just slightly better than that of 8KB caches (e.g., 6% at HP-HPT) across different operation modes, so we consider 8KB caches as the best choice.

It can be concluded that the proposed hybrid designs exhibit basically the same behavior as non-hybrid ones in terms of performance (execution time) at each different operation mode.

2) *Dynamic Energy per Instruction.* Using 8T and 10T SRAM cells instead of 6T ones increases dynamic energy per instruction at HP-HPT mode, which is in line with the findings reported in Section 4.3.1, where caches are studied in isolation. For non-hybrid 8T and 10T caches, dynamic energy increases more than 1.5X and 2.0X respectively. In the case of hybrid designs, values resemble very closely the sum of values for non-hybrid configurations weighted by the fraction of cache space devoted to each particular





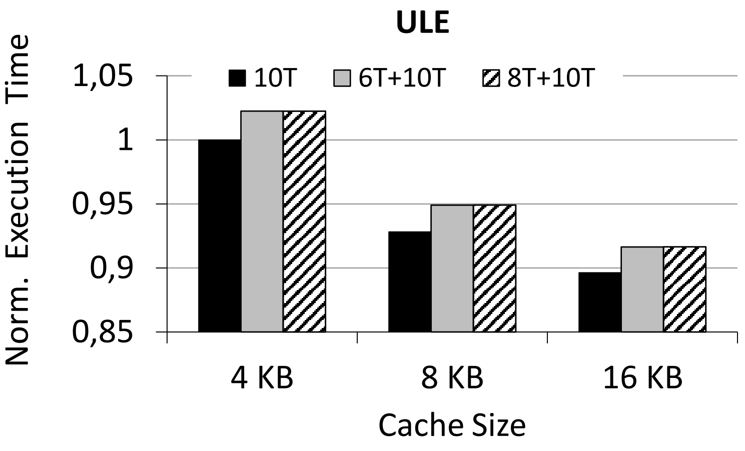


Figure 4.5: Normalized average execution time for 8-way Non-Hybrid and 7+1 Hybrid Cache Ways designs at HP-HPT, HP-LPT and ULE modes when varying cache size. Note that cache size at ULE mode for 6T+10T and 8T+10T hybrid designs is 512B, 1KB and 2KB respectively due to disabled 6T and 8T ways.

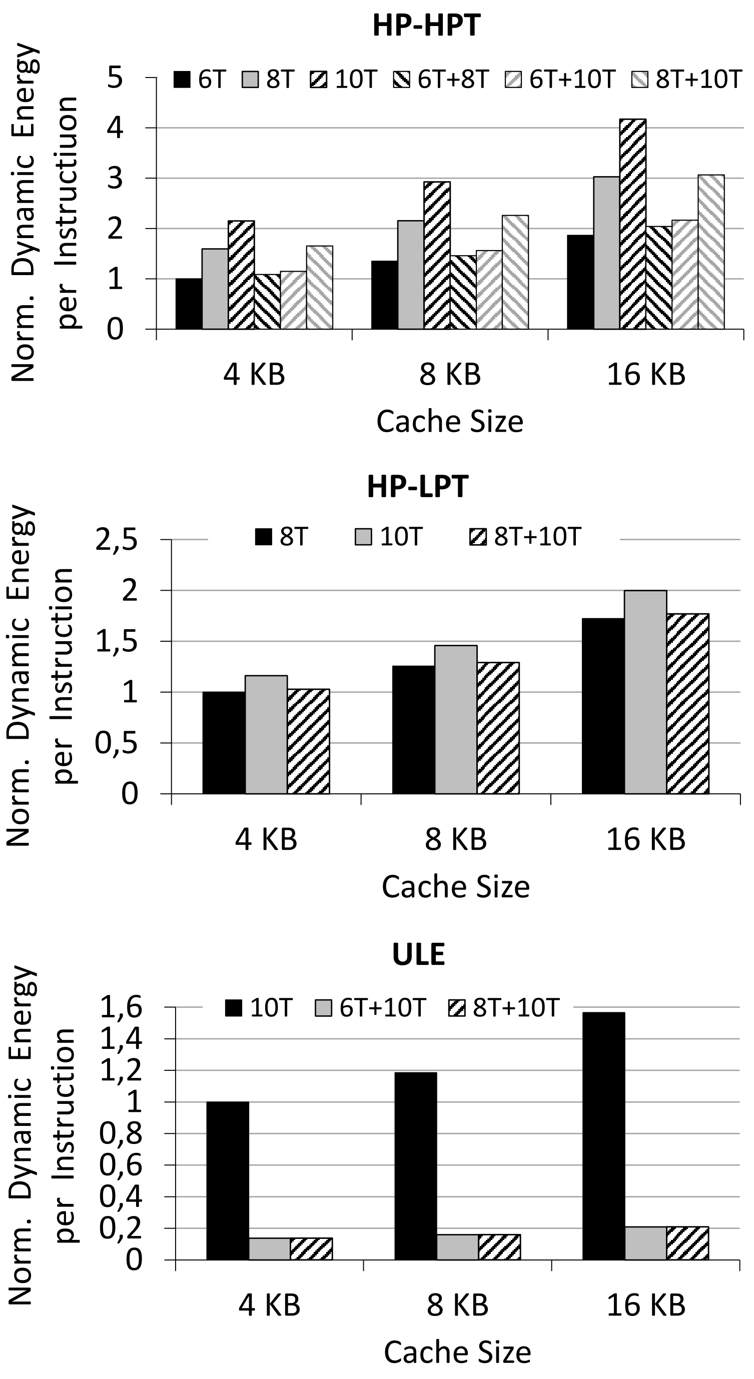


Figure 4.6: Normalized average dynamic energy per instruction for 8-way Non-Hybrid and 7+1 Hybrid Cache Ways designs at HP-HPT, HP-LPT and ULE modes when varying cache size. Note that cache size at ULE mode for 6T+10T and 8T+10T hybrid designs is 512B, 1KB and 2KB respectively due to disabled 6T and 8T ways.

technology type (i.e. number of cache ways devoted to one technology, divided by the total number of cache ways). Therefore, hybrid designs introduce some overheads at HP-

HPT mode (e.g., 14% for 6T+10T hybrid cache). As expected, trends at HP-LPT mode also match those reported in Section 4.3.1. Pure 10T caches have larger overhead (around 16% on average) with respect to the pure 8T designs than hybrid 8T+10T designs (2.8% on average). However, at ULE mode, our hybrid 6T+10T/8T+10T designs achieve large savings (up to 86%) with respect to the pure 10T designs due to turning off 6T/8T ways.

Dynamic Energy per Instruction grows quite linearly with cache size mainly due to the increased number of bitlines to discharge. This trend is similar for all cache configurations across different cache sizes. Finally, the relative increase in Dynamic Energy per Instruction across different cache sizes is larger at HP-HPT and HP-LPT modes than at ULE mode because dynamic energy is the dominant component in total energy at 1V and 0.7V.

It can be concluded that hybrid designs introduce some overheads in terms of dynamic energy per instruction relative to the 6T and 8T non-hybrid designs at HP-HPT/HP-LPT mode. However, 6T and 8T non-hybrid designs cannot be used at ULE mode, so our hybrid caches are the best choices. On the other hand, the proposed hybrid designs achieve significant savings at ULE mode with respect to pure 10T caches, where energy is the primary concern.

3) *Leakage Power per Cycle.* Relative trends for leakage power per cycle at HP-HPT mode across all configurations resemble quite closely those for dynamic energy per instruction. On the other hand, there is a larger variation across configurations at HP-LPT mode, which is again in line with the findings in Section 4.3.1. For instance, pure 10T caches exhibit larger overhead (42% on average) with respect to the pure 8T designs than hybrid 8T+10T caches (8% on average). At ULE mode large savings are achieved (up to 78% with respect to the pure 10T designs) with our hybrid 6T+10T/8T+10T designs due to turning off 6T/8T ways.

Leakage Power per Cycle grows with cache size. Leakage does not grow much when moving from 4KB to 8KB since efficient SRAM array arrangements are found, thus keeping bitline leakage low. However, the array arrangement determined automatically by CACTI for 16KB caches is not that efficient and thus, leakage grows noticeably. This trend, however, is different for 6T and 8T arrays when compared to 10T ones, whose leakage is relatively higher for 8KB caches as shown at the ULE mode.

Therefore, it can be concluded that our hybrid designs are the only ones which provide high efficiency at all operation modes[3].

[3]Power and energy trends are identical given that configurations compared have almost identical execu-

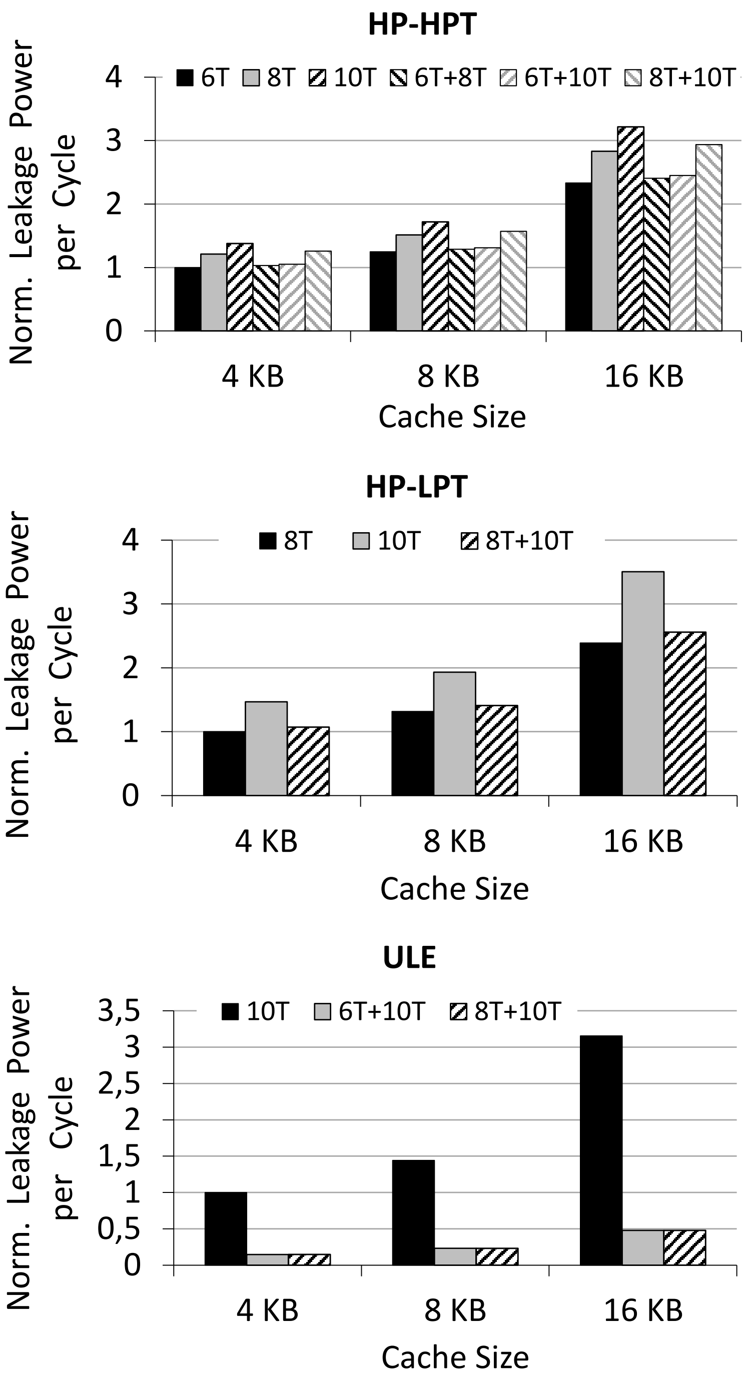


Figure 4.7: Normalized average leakage power per cycle for 8-way Non-Hybrid and 7+1 Hybrid Cache Ways designs at HP-HPT, HP-LPT and ULE modes when varying cache size. Note that cache size at ULE mode for 6T+10T and 8T+10T hybrid designs is 512B, 1KB and 2KB respectively due to disabled 6T and 8T ways.

tion times at the same mode.

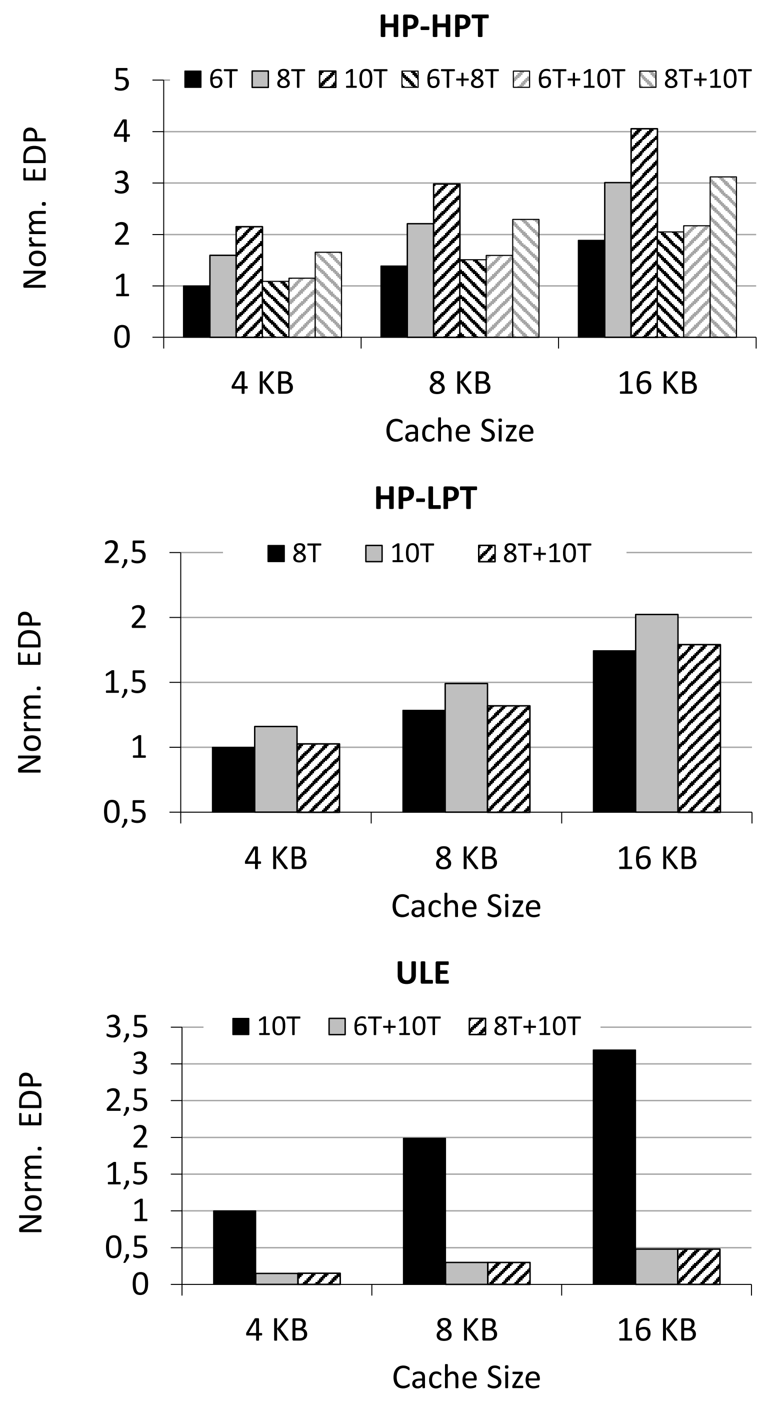


Figure 4.8: Normalized average energy-delay product (EDP) for 8-way Non-Hybrid and 7+1 Hybrid Cache Ways designs at HP-HPT, HP-LPT and ULE modes when varying cache size. Note that cache size at ULE mode for 6T+10T and 8T+10T hybrid designs is 512B, 1KB and 2KB respectively due to disabled 6T and 8T ways.

4) *EDP*. Relative trends for EDP across configurations and cache sizes at the same mode follow the trends observed for total energy. This is because execution time does not vary at HP-HPT and HP-LPT modes while such variation at ULE mode is negligible (2% on average). The relative values for EDP at HP-HPT and HP-LPT modes have almost the same trend as those observed for dynamic energy per instruction, because dynamic energy dominates total energy when operating at such modes. As opposed to the case of HP-HPT and HP-LPT mode, leakage energy is the dominant factor at ULE mode. We observe that relative trends for EDP at ULE mode highly correlate with those reported for leakage power per cycle.

Overall, this set of experiments proves that energy consumption for the whole processor is highly dependent on the particular cache configuration used for L1 data and instruction caches, and such energy highly correlates with the results observed when considering caches in isolation. Different benchmarks do not introduce noticeable variations in any metric. Based on the results, we conclude that an 8KB, 7+1 hybrid cache configuration is the most convenient choice. We also conclude that, as expected, dynamic energy is the dominant factor at HP-HPT and HP-LPT modes, whereas leakage is the dominant factor at ULE mode. We show that the proposed hybrid designs achieve exactly the *same average performance* as non-hybrid designs at HP-HPT/HP-LPT mode at the expense of a small dynamic energy and leakage power overhead when compared to non-hybrid designs. Also, we show that proposed hybrid caches have *significantly smaller dynamic energy and leakage power* at ULE mode than non-hybrid ones. Finally, we show that our hybrid cache designs can efficiently and reliably operate across a wide range of voltages (modes), consuming ultra-low energy at ULE mode as well as providing the high performance needed at HP-HPT/HP-LPT mode (e.g., 6T+10T/8T+10T), as required for our target market. Finally, our hybrid caches provide deterministic performance behavior since each operation mode leads to deterministic cache size, thus enabling *strong performance guarantees* needed for running critical applications on top.

4.3.2.3 Memory Latency Sensitivity Study

Memory latency may have an important effect in execution time and thus, in leakage energy. We model memory latency as shown in Figure 4.9. Latency (L) is determined by equation (4.1):

$$L = t_{on-chip} + t_{on-/off-chip(interface)} + t_{off-chip} \tag{4.1}$$

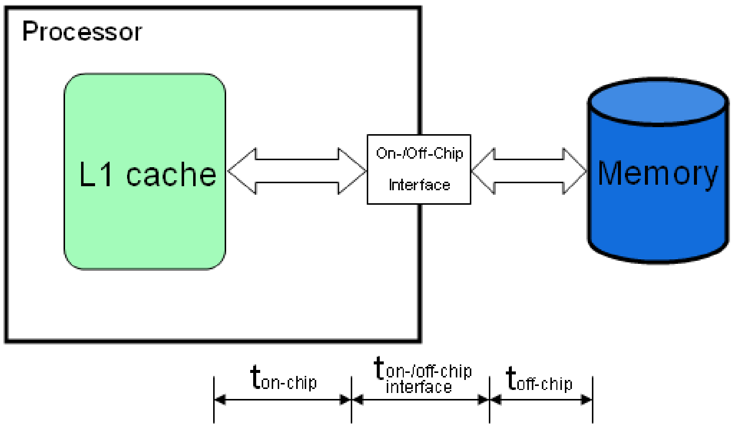


Figure 4.9: Memory Latency.

where $t_{on-chip}$ is the on-chip latency, $t_{on-/off-chip(interface)}$ is the latency of the on-/off-chip interface and $t_{off-chip}$ corresponds to the off-chip latency. When Vcc is scaled down latencies of on-chip resources ($t_{on-chip}$ and $t_{on-/off-chip(interface)}$) increase in absolute numbers. However, relative numbers for those latencies are not expected to scale noticeably (in number of cycles) because their voltage is decreased as for the rest of the chip. The only latency that decreases in relative numbers when decreasing chip voltage is off-chip latency ($t_{off-chip}$). However, the relative memory latency is low given the high integration between processor and memory in those systems and the small memory size (typically few MBs). Thus, overall *relative* memory latency L does not scale much at HP-LPT and ULE mode with respect to HP-HPT mode.

Next we present execution time, dynamic energy, leakage power and energy-delay product (EDP) for different operation modes when varying memory latency in order to understand its impact in performance and energy. We use a 8KB, 8-way cache for all configurations and set different memory latencies during different operation modes as follows (bold numbers correspond to our baseline memory latencies):

- HP-HPT: 10, **20** and 100 cycles.
- HP-LPT: **16**, 20 and 100 cycles.
- ULE: **14**, 20 and 100 cycles.

For the sake of clarity, results have been normalized with respect to the reference design for each voltage level. In particular, values are normalized with respect to the pure 6T

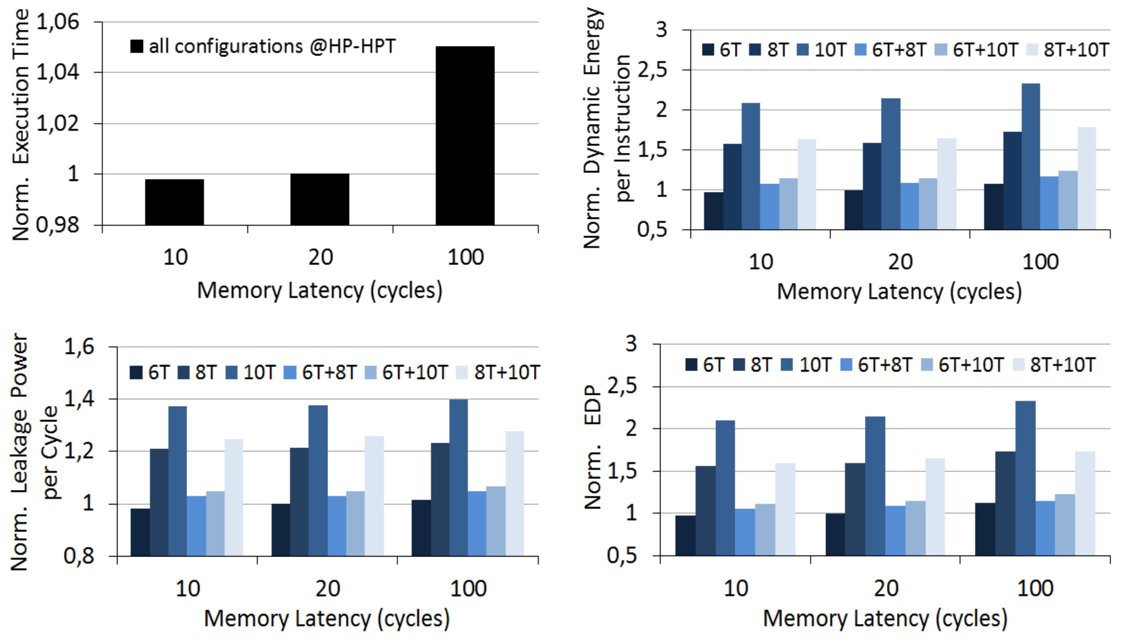


Figure 4.10: Average normalized execution time, dynamic energy, leakage power and energy-delay product for Non-Hybrid and Hybrid Cache Ways designs when varying memory latency and running *BigBench* applications at HP-HPT mode.

(20 cycles latency), 8T (16 cycles latency) and 10T (14 cycles latency) designs for HP-HPT, HP-LPT and ULE mode respectively. Results presented correspond to the average across all benchmarks used at such operation mode (*BigBench* at HP-HPT and HP-LPT, and *SmallBench* at ULE mode) since minor variations are observed across benchmarks.

Figure 4.10 depicts results at HP-HPT mode. As expected, increasing memory latency increases execution time (up to 4.8%). However, the increase is small as expected since workloads fit in cache, which will be the real case. Higher execution time translates into increased leakage for all configurations (around 8.6%). Dynamic energy consumption remains almost constant (up to 1.4% variation) across different latencies. EDP is similar to dynamic energy, as dynamic energy is the dominant factor at HP-HPT. In general, we observe that no significant variation is observed in any metric when varying memory latency at HP-HPT mode.

Figure 4.11 and Figure 4.12 present results for HP-LPT and ULE modes respectively. Trends are very similar to those for HP-HPT mode and only leakage observes a somewhat larger variation. Nevertheless all trends hold across different memory latencies, so this parameter has no relevant effect on our study. Therefore, in the rest of the chapter, memory latency is set to 20, 16 and 14 cycles for HP-HPT, HP-LPT and ULE modes

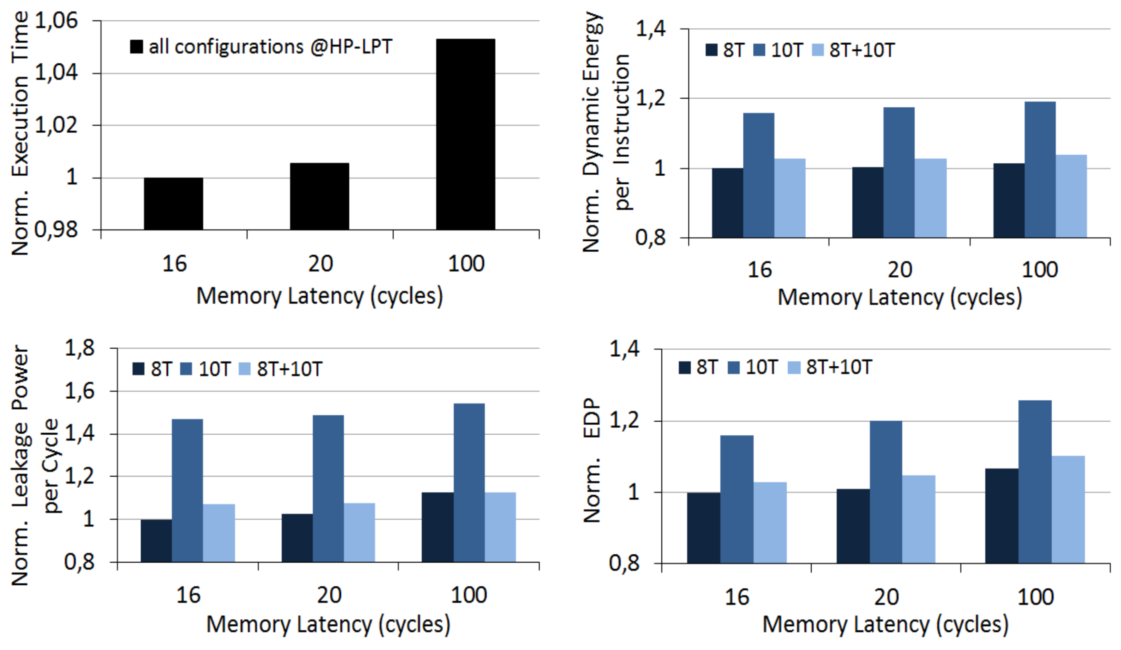


Figure 4.11: Average normalized execution time, dynamic energy, leakage power and energy-delay product for Non-Hybrid and Hybrid Cache Ways designs when varying memory latency and running *BigBench* applications at HP-LPT mode.

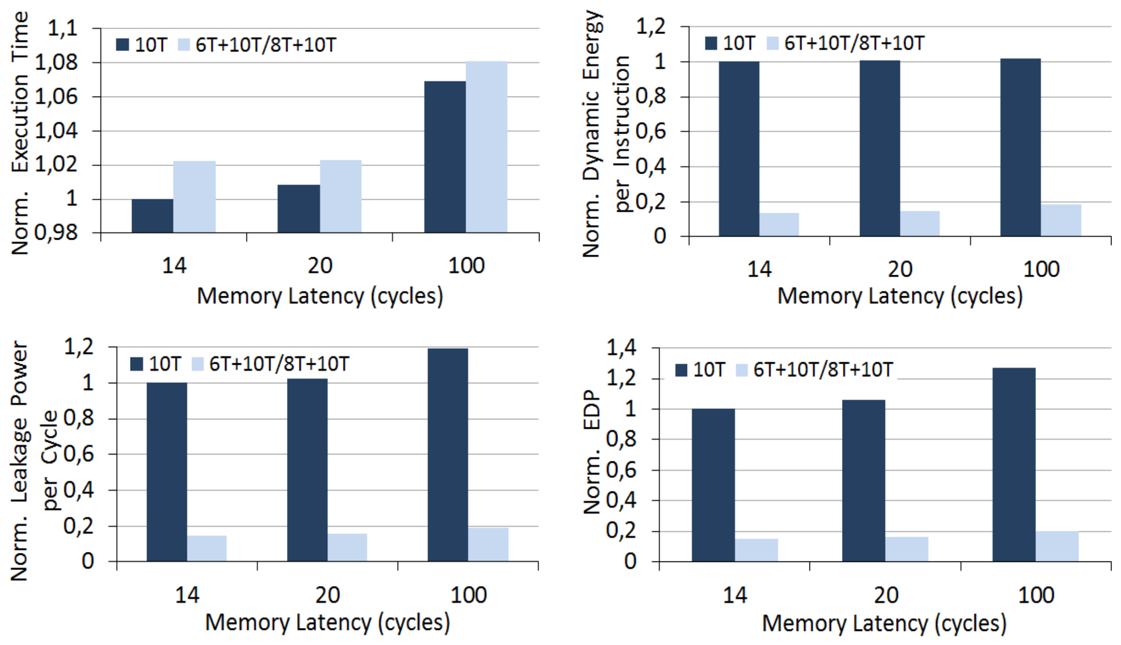


Figure 4.12: Average normalized execution time, dynamic energy, leakage power and energy-delay product for Non-Hybrid and Hybrid Cache Ways designs when varying memory latency and running *SmallBench* applications at ULE mode.

respectively as shown in Table 3.1.

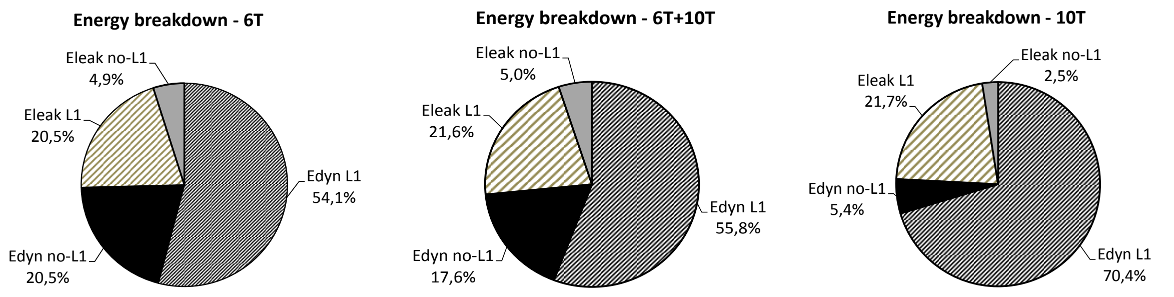
4.3.2.4 Energy Breakdown

The next set of results focuses on the total on-chip energy[4] and its distribution in terms of dynamic and leakage energy when varying supply voltage for all non-hybrid and hybrid configurations considered. Since cache memories are the main energy consumers in our target scenarios, we break down energy into the following categories: L1 cache dynamic energy for data and instructions (EdynL1), L1 cache leakage energy for data and instructions (EleakL1), dynamic energy for the rest of the chip (Edyn no-L1) and leakage energy for the rest of the chip (Eleak no-L1). Results for some configurations using 6T and 8T cells have been omitted since they do not provide further insights. In particular, 8T-based designs at HP-HPT mode are always worse than 6T-based ones, so they are omitted. Similarly, 6T-based ones are unsuitable for HP-LPT operation because 6T ways must be turned off. Finally, 8T-based designs at ULE mode achieve almost identical results to 6T-based ones, so only 6T ones are shown.

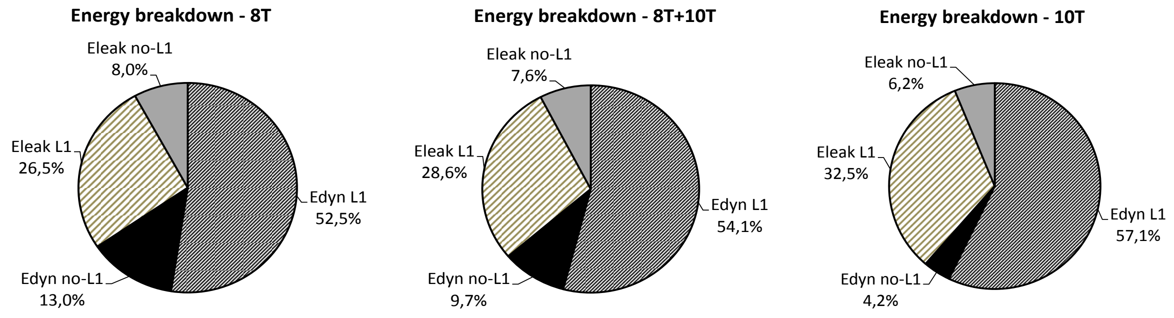
Figures 4.13(a), 4.13(b) and 4.13(c) show the on-chip energy breakdown for all configurations when operating at different voltage levels. Most of the energy corresponds to dynamic energy at HP-HPT mode as shown in Figure 4.13(a). In particular, 75% of the energy consumed is dynamic energy on average across the different configurations. Most dynamic energy corresponds to L1 caches, whose contribution to the total chip energy is always above 50%. Thus, L1 caches dynamic energy is the main energy contributor at HP-HPT mode. Interestingly, cache energy contribution is lower for those configurations where 6T cells are used. This is because the absolute energy of the cache is lower whereas the rest of the chip energy remains nearly constant. As explained before, 6T SRAM cells are the preferred choice for HP-HPT operation. Note also that the 6T+10T energy breakdown is highly similar to that of the pure 6T, because only one cache way in 6T+10T is made of 10T SRAM cells.

As supply voltage is decreased, dynamic energy contribution is lower and leakage energy increases. This trend is shown when operating at HP-LPT mode (Figure 4.13(b)). In this case, dynamic energy is around 60% of the total energy, most of it due to L1 caches. We observe that 8T SRAM cells consume less energy than 10T ones at HP-LPT, thus reducing the fraction of both dynamic and leakage energy devoted to the cache

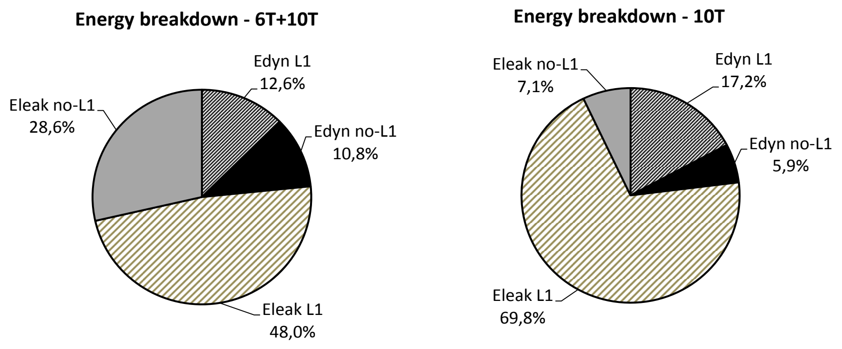
[4]We do not include off-chip memory energy because it exhibits negligible variation across different cache configurations at the same mode.



(a) HP-HPT mode



(b) HP-LPT mode



(c) ULE mode

Figure 4.13: Total on-chip energy breakdown when operating at: (a) HP-HPT mode, (b) HP-LPT mode and (c) ULE mode.

when compared to 10T-based configurations. For instance, cache energy is lower for the 8T configuration than for the 8T+10T one, which is still lower than that for the 10T one. Again, the energy breakdown of the hybrid configuration (8T+10T) resembles much more that of the pure 8T one than the one of the pure 10T cache.

As expected, operating at NST voltage leads to higher contribution of leakage energy and significant dynamic energy decrease. Moreover, as opposed to HP-HPT and HP-LPT operation where 8KB caches are in place, ULE operation has only 1KB caches for the 6T+10T configuration. This effect is shown in Figure 4.13(c), where leakage is around

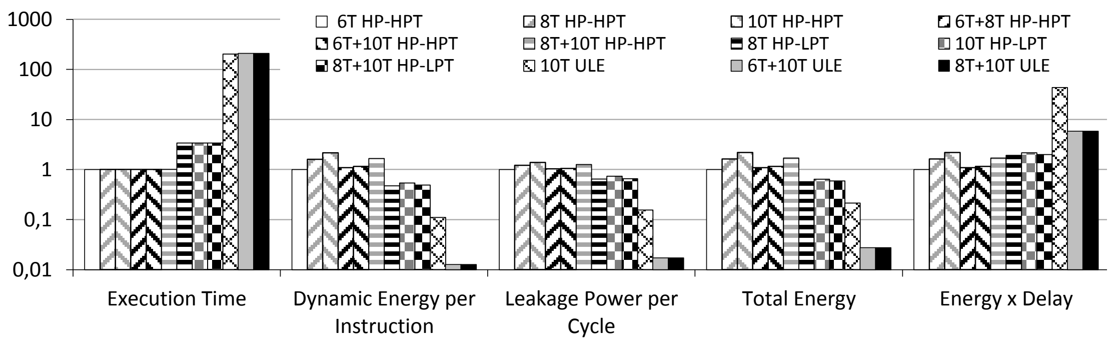


Figure 4.14: Comparing the execution time, dynamic energy, leakage power, total energy and energy-delay product of the non-hybrid and hybrid cache designs operating at HP-HPT, HP-LPT and ULE mode with respect to the pure 6T design at HP-HPT mode. Note logarithmic scale.

70%-75% of the total energy. L1 cache dynamic energy is still significant, although not as much as leakage. As shown, dynamic energy is only around 25% of the total energy at ULE mode and most of such energy corresponds to the L1 caches. The pure 10T configuration incurs significant energy overheads due to its larger cache size (8KB) and obtains negligible performance gains. This effect translates into a much higher contribution of L1 caches in the energy breakdown.

4.3.2.5 Impact of Voltage

Finally, we compare all metrics of interest across different voltage levels in Figure 4.14. All configurations considered at HP-HPT, HP-LPT and ULE modes are normalized with respect to the pure 6T designs operating at HP-HPT mode. Figure 4.14 plots execution time, dynamic energy, leakage power, total energy and EDP. All the values reported correspond to the average of all programs run (*BigBench* for HP-HPT and HP-LPT modes, and *SmallBench* for ULE mode).

It can be noticed easily that the execution time increases drastically when voltage is decreased. The main reason for such behavior is the difference in operating frequencies at different operation modes. For instance, execution time on average at ULE mode is around 210X higher than that at HP-HPT mode.

Operating at HP-LPT delivers around 50% dynamic energy per instruction reduction for all configurations. Reduction at ULE mode is around 9X for the configuration where all cache ways work normally (pure 10T) whereas such reduction is around 83X for our hybrid caches (8T+10T and 6T+10T) due to disabled 6T/8T ways.

Leakage power per cycle at HP-HPT and HP-LPT modes exhibits smaller variations across all configurations than for dynamic energy per instruction due to the small impact of the lowered voltage (from 1V to 0.7V) on leakage. Reduction at HP-LPT mode is around 33%. As expected, trends observed for leakage power per cycle at ULE mode are similar to those observed for dynamic energy per instruction at ULE mode.

In terms of EDP, ULE mode is the least interesting design point. Although dynamic energy and leakage power savings at ULE mode are around 10X for 10T designs and 85X for hybrid 8T+10T/6T+10T designs, delay increases more than 200X and therefore, EDP is dramatically increased (up to 40X for pure 10T and up to 5X for hybrid 8T+10T/6T+10T designs). However, the main concern at ULE mode is the total energy, which is drastically reduced as shown in Figure 4.14.

4.3.2.6 Overall Energy Savings

In this section, we report overall energy savings for the whole lifetime of the artificial application which we described in section 3.2.6. We have selected proposed 6T+10T hybrid cache and non-hybrid 10T cache and compared their total energy for two different scenarios:

- Duty cycle of 1%: ULE mode lasts for 99% of the total application lifetime (i.e. HP mode lasts for 1% of the time), and

- Duty cycle of 0.1%: ULE mode lasts for 99.9% of the total application lifetime (i.e. HP mode lasts for 0.1% of the time).

Basically, we take average energy results for *SmallBench* programs and assume that they execute 99% or 99.9% of the time, and average results for *BigBench* programs and assume that they execute 1% or 0.1% of the time respectively. Figure 4.15 plots total energy breakdown for the whole application lifetime for non-hybrid 10T and hybrid 6T+10T cache designs. Beside the dramatic energy reduction of the proposed hybrid 6T+10T cache with respect to 10T cache (already explained in the previous sections), we observed that both operation modes (HP and ULE mode) have significant contribution to the total energy during the application lifetime:

- HP mode: because of the high absolute amount of energy consumed despite of the very short time period, and

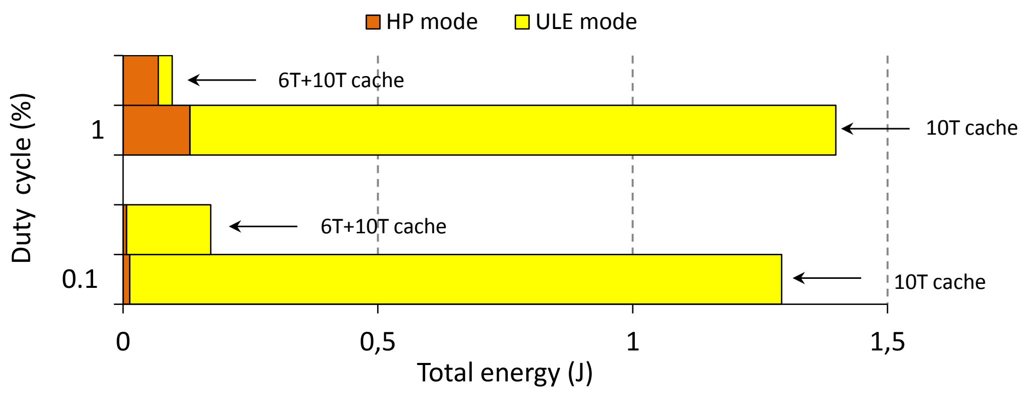


Figure 4.15: Total energy breakdown for the whole application lifetime for non-hybrid 10T and hybrid 6T+10T cache designs showing energy contribution of each operation mode for different duty cycle values.

- ULE mode: because application spends long time in this mode during its lifetime albeit its ultra-low energy consumption.

Also, energy contribution of each mode highly depends on the application behavior, what we illustrated with results for different duty cycle values. Therefore, we can conclude that energy reduction is very important and cannot be neglected even at HP mode where performance is the primary concern.

4.3.2.7 Guaranteed Performance

WCET analysis is an expensive task and relies on a particular cache configuration: cache size, cache associativity, replacement and placement policy, hit and miss latency, etc. Proposed hybrid caches provide exactly the same number of available fault-free cache lines per set as in the baseline, thus guaranteeing the same WCET performance (same hits and same misses). Therefore, the WCET analysis will be the same without increasing any complexity for its estimation.

4.4 Summary

In this chapter, we propose new, single-Vcc domain, hybrid L1 cache architectures for reliable hybrid voltage operation, which meet *all* specific and stringent needs of battery-powered ultra-low-cost (e.g., below 1 USD) systems. The proposed cache designs rely on combining heterogeneous SRAM cell types so that some of the cache ways are optimized to satisfy high performance requirements during high Vcc operation (HP ways) whereas the rest of the ways provide ultra-low energy consumption and reliability during NST Vcc operation (ULE ways). Our results show that, first, the proposed hybrid caches can efficiently and reliably operate across a wide range of voltages, consuming little energy at ULE mode as well as providing high performance with small overheads at HP mode, as required for our target market. Second, we show that the proposed hybrid designs achieve exactly the *same average performance* when compared to conventional non-hybrid designs at HP-HPT/HP-LPT mode at the expense of small dynamic energy and leakage power overheads. Third, we show that the proposed hybrid caches *significantly reduce dynamic energy and leakage power* at ULE mode w.r.t. non-hybrid ones (around 9X). Likewise, our proposed caches exhibit deterministic behavior since available cache size is deterministic at all operation modes, thus enabling *strong performance guarantees*, as needed for running critical applications on top. The proposed caches provide the same interface to the WCET estimation tools as they were before for non-hybrid caches without increasing any complexity for WCET estimation.

Finally, our experiments show that those trends are consistent across different combinations of SRAM cell types, cache sizes and associativity values, and open the door to further research in the design of hybrid microarchitectures.

Chapter 5

ADAM: Adaptive Data Management Mechanism

Despite their efficiency in front of non-hybrid cache designs, there is still room for improvement in our hybrid designs presented in the previous chapter. In particular, we aim at reducing the energy overheads introduced by using ULE ways at HP mode. To that end, this chapter describes our cache data management mechanism at HP mode that takes advantage of having two cache regions (i.e. HP and ULE cache ways), trying to manage hits so that they occur in HP ways, whose dynamic energy is lower than that of ULE ways, and hence, improve energy efficiency with negligible impact on average performance.

5.1 Introduction

Data management policies are required to access HP and ULE ways **at HP mode** to minimize the number of accesses to ULE ways. Unfortunately, existing policies are far from being efficient across all applications. For example, the simplest one consists of accessing all cache ways in parallel on every access. Parallel access consumes higher dynamic energy due to the inefficiency of ULE ways at high Vcc. Alternatively, HP ways can be accessed and, only in case of a miss, ULE ways are accessed [25]. Such approach is efficient as long as hits occur most of the times in the HP ways, which is not always the case. If ULE ways hold the data, no energy savings are obtained and performance is degraded. Recently, some authors have proposed to swap HP and ULE data in case of an ULE hit [26]. Unfortunately, such design is not always the best choice because some applications (or application phases) may end up performing a large number of swaps with little or negative decrease in the number of ULE accesses. Therefore, average performance may be degraded significantly and energy consumption increased.

In this chapter, we propose an efficient, but simple data management mechanism for HP operation on single-Vcc domain caches for hybrid voltage operation. Our mechanism is called Adaptive Data Management (ADAM). ADAM is tailored to detect hit distribution dynamically across the cache lines during program execution and adapts to different application behaviors to optimize performance and energy consumption by means of an extremely simple hardware mechanism. Experimental results show that ADAM achieves average energy savings between 12% and 29% with respect to all state-of-the-art approaches at HP mode with negligible performance impact (1.7% with respect to the best state-of-the-art approach). ADAM largely outperforms all mechanisms in terms of energy-delay product.

The rest of this chapter is structured as follows. Section 5.2 presents ADAM mechanism and existing state-of-the-art cache data management mechanisms. Section 5.3 reports experimental results and, finally, Section 5.4 summarizes the chapter.

5.2 Adaptive Data Management

Hybrid cache designs offer significant room to trade off between energy and performance at HP mode to fit application requirements dynamically by using different data management mechanisms to access HP and ULE ways. This section describes existing state-of-the-art data management mechanisms, analyzes their drawbacks and proposes ADAM, an Adaptive Data Management mechanism, which achieves significant energy savings with negligible performance degradation at HP mode for hybrid caches with single-Vcc domain.

5.2.1 Baseline Cache Architecture

Our approach is built on top of a 6T+10T hybrid cache although any other hybrid configuration proposed in chapter 4 may be used.

Figure 5.1 depicts the 6T+10T cache design used [60]. We have considered different hybrid configurations with similar total area to that of a cache where all cache ways are implemented with 6T SRAM cells, although such a design would not work at ultra-low voltage. However, our data management technique is not limited to any particular configuration. The first hybrid configuration considered consists of an 8KB 8-way cache, where 7 ways are implemented with 6T SRAM cells and 1 way with 10T SRAM cells (7+1 for

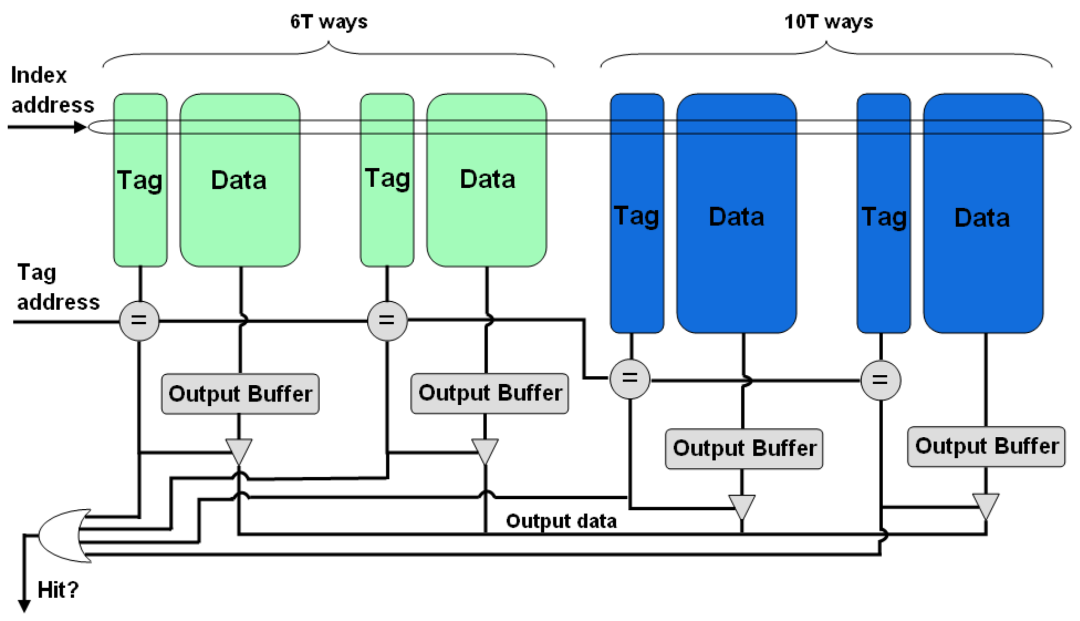


Figure 5.1: Hybrid 6T+10T cache design.

short). Due to the additional 45% area of the 10T SRAM cells with respect to 6T cells, this configuration has an area overhead of around 5% with respect to the case where all cache ways are implemented with 6T cells (8+0). We also consider a 7KB 7-way cache implemented with a 5+2 hybrid configuration. Such configuration has similar area to that of an 8+0 one.

5.2.2 State-of-the-art Data Management Mechanisms

Existing state-of-the-art data management mechanisms are not particularly devised for hybrid caches implemented with a single-Vcc domain, but we use them for comparison purposes. They are extensively evaluated against our proposed mechanism in section 5.3. In general, they are built on top of the LRU replacement policy, so we assume LRU as the replacement policy in the rest of the chapter.

5.2.2.1 Parallel Mechanism

The first mechanism is the simplest one and accesses all cache ways in parallel. This technique is the fastest one, but very inefficient in terms of energy since 10T ways are accessed always. We refer to this mechanism as *Parallel* in the rest of the chapter.

5.2.2.2 Sequential Mechanism

The second mechanism [25] (referred to as *Sequential*) improves energy efficiency upon the *Parallel* one by accessing 6T ways first since they are the most energy efficient ways at HP mode. In case of a miss in the 6T ways, the 10T ways are accessed next. Energy efficiency is achieved only if hits concentrate mostly in the 6T ways. Otherwise, this mechanism will be slower than the *Parallel* one due to the extra delay to access the 10T ways.

5.2.2.3 Swap Mechanism

The *Swap* [26] mechanism accesses cache ways sequentially as the *Sequential* one does (first 6T ways and, in case of a miss, 10T ways), but on a 10T hit the hit line is swapped with the 6T line closest to the Least Recently Used (LRU) position and becomes the Most Recently Used (MRU). On a miss, the line in the LRU of the 10T ways is evicted, the LRU line of the 6T ways is moved to the 10T ways and the fetched line is then placed in the 6T ways. Therefore, the most recently used lines reside in the 6T ways and the LRU ones in the 10T ways. Figure 5.2 depicts the hardware support required for swapping. Swapping is done in four phases: (i) LRU 10T data (rd10T) read out and written into the fetch line buffer (this buffer is used also to transfer data to/from memory), (ii) LRU 6T data (rd6T) read out and written into the swap buffer, (iii) write swap buffer contents into the 10T LRU position (wr10T) and (iv) write fetch line buffer contents into the 6T LRU position (wr6T). In order to perform swapping successfully, we adopt a fairly simple solution: *stall any access to cache during swapping, which lasts for the latency of 4 cache accesses*. Typically, preventing further accesses during some cycles is as easy as keeping the cache ports busy to prevent the port arbiter from issuing new accesses. Besides the latency of 4 accesses, swapping overhead includes extra read/write operations energy.

The *Swap* mechanism is particularly efficient for irregular applications where hits concentrate in few cache lines during long intervals. Thus, if any of those cache lines resides in a 10T way, it is quickly swapped to the 6T ways so that the energy spent swapping lines is quickly offset by the hits in such cache line. In this case, most of the hits concentrate in the 6T ways. Figure 5.3(a) presents a swap-friendly application. For instance, let us assume a 3+1 (three 6T ways and one 10T way) hybrid configuration and the following lines being mapped into the same cache set: A, B, C and D. Initially, the lines are placed in cache as shown in Figure 5.3(a). If we have the next sequence of the

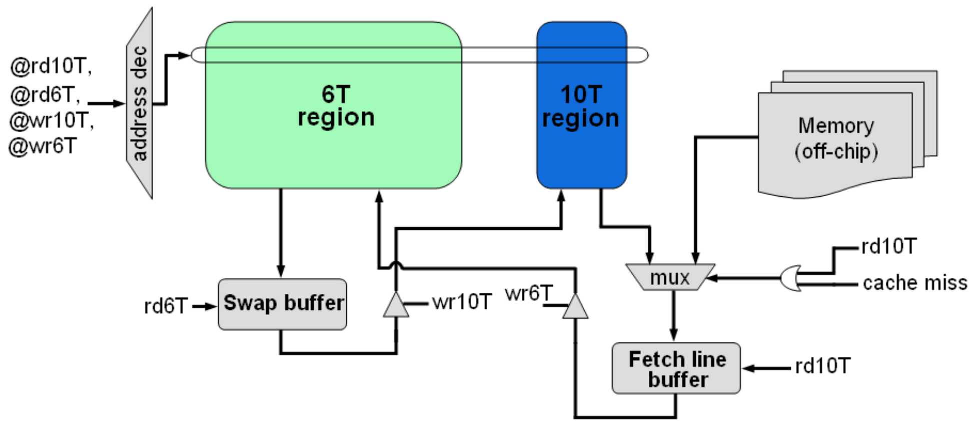
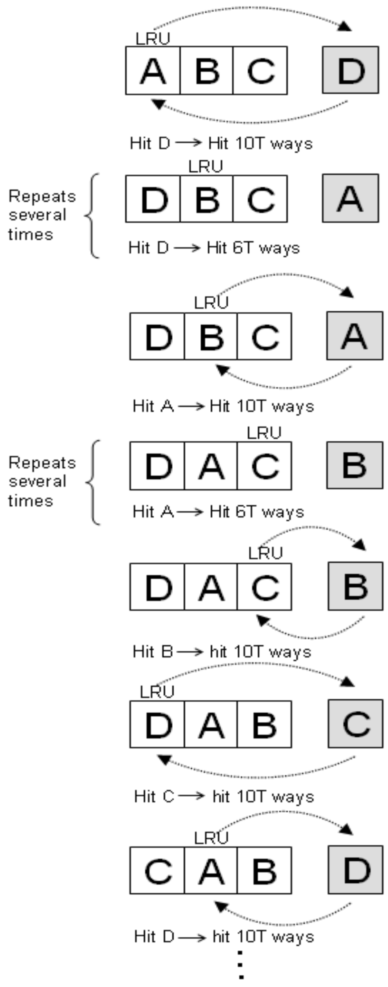


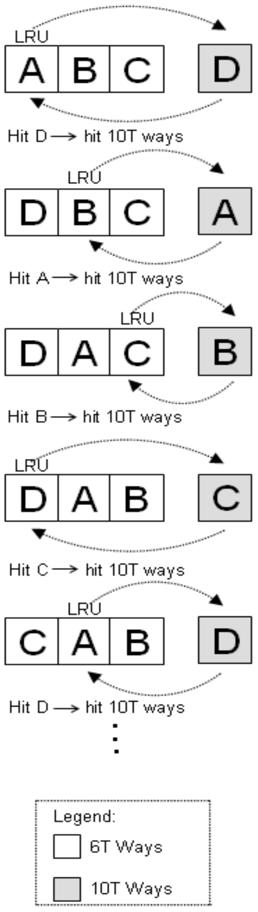
Figure 5.2: Hardware support for Swap operation.

hits in a loop: (D* A* BC)* (where * means an arbitrary number of repetitions), we firstly hit D in the 10T ways. After that, we swap the LRU 6T ways data (A) with D and have a burst of D hits in the 6T ways. Then we hit A for the first time in the 10T ways and again swap A with the LRU 6T line (B). After that we have a burst of A hits in the 6T ways. The next hit is B, which is in the 10T ways so it is swapped with C (LRU 6T line). Then C is hit the 10T way and swapped with D. Finally, a D hit occurs in the 10T ways and the pattern repeats. It can be seen that in this case most of the hits concentrate in the 6T ways. In fact, if the access pattern is even more irregular (D and A are accessed in between B and C) we can expect D and A not to be swapped to the 10T way at any time.

However, for regular applications hit distribution is quite homogenous across the cache lines. Moreover, if the working set matches quite well cache size (the ideal case to exploit cache capacity), hits concentrate mainly in the LRU positions of each cache set, which would reside in the 10T ways if swapping is performed. In this case, swapping can introduce additional energy consumption due to many unnecessary swaps. The energy overhead does not pay off because once swapped those lines are barely hit in the 6T ways. A good example is the continuous traversal of a vector or array that occupies most of the cache. In this example many hits concentrate in the LRU ways (thus, 10T ways when *Swap* is used). Figure 5.3(b) presents this not-swap-friendly case. For instance, let us have again the 3+1 hybrid configuration with the same data and the next sequence of hits: (DABC)*. Every access hits the 10T way and produces a swap with the line in the LRU of the 6T ways, which will be hit next. This one is the pathological case for this approach. If no swapping was performed, most of the hits would concentrate in the 6T



(a) Swap-friendly.



(b) Not-swap-friendly.

Figure 5.3: Examples of the swap-friendly and not-swap-friendly applications.

ways, performance would be higher and swapping overheads would not be incurred.

Based on the inefficiencies observed for both *Sequential* and *Swap* mechanisms, we propose ADAM, a very simple, adaptive mechanism to swap only when it pays off.

5.2.3 ADAM: Adaptive Data Management Mechanism

ADAM tracks the hit distribution across the different cache regions (6T and 10T regions) during program execution and dynamically activates/deactivates swapping to optimize performance and energy consumption. Hence, our mechanism has two distinct modes: swap and no-swap. Swap mode allows swapping between the LRU cache location of 10T ways and the LRU cache location of 6T ways similarly to [26]. Information about LRU

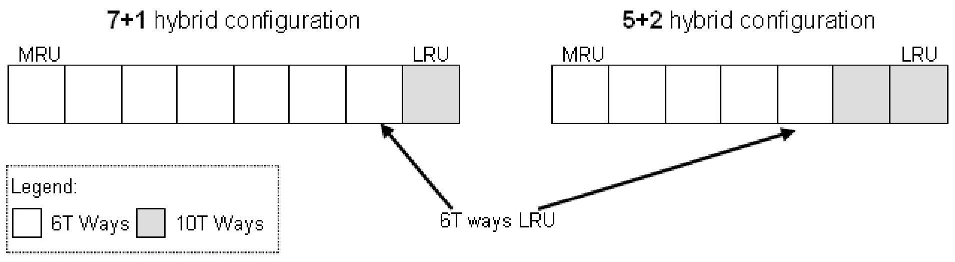


Figure 5.4: Global LRU stacks for 7+1 and 5+2 hybrid cache configurations.

position in the 6T and 10T ways is obtained from the global LRU stack (see Figure 5.4) since our technique forces the LRU locations to remain into the 10T ways during swap mode. This greatly simplifies the implementation and avoids using extra control logic to create three separated LRU stacks (one for 6T ways, one for 10T ways and a global one).

Swapping is then performed by using existing fetch line buffer and additional swap buffer to switch the cache line contents as for the *Swap* mechanism. No-swap mode stops this functionality.

ADAM tracks the number of hits in each region (6T and 10T). Whenever the number of hits in the 10T region is high, the operation mode is changed. Thus, if swap was activated it is deactivated and vice versa. This way, if swap is deactivated and many hits occur in the 10T ways, ADAM activates swap so that those lines are swapped into the 6T ways for higher efficiency. Conversely, if swap is activated, but many hits occur in the 10T ways, ADAM deactivates swap to stop useless swapping. By doing so, ADAM combines the advantages of the *Swap* and *Sequential* approaches, and moreover, reacts in front of the different phases of a program. Next we describe how to implement such a mechanism and when the operation mode must be changed.

5.2.3.1 Implementation Details

Figure 5.5 depicts the control logic to change the operation mode. Counters are employed to track the numbers of hits to the 6T and 10T ways respectively. The size in bits of each counter is set based on the expected hit probabilities in 6T and 10T cache ways.

Hit distribution largely depends on the program. We assume the worst case where the hit distribution is uniform across the cache lines. In this scenario each cache line has similar hit probability in each cache set, so it does not matter which particular cache lines reside in each cache way. Therefore, the size in bits of the hit counters depends on

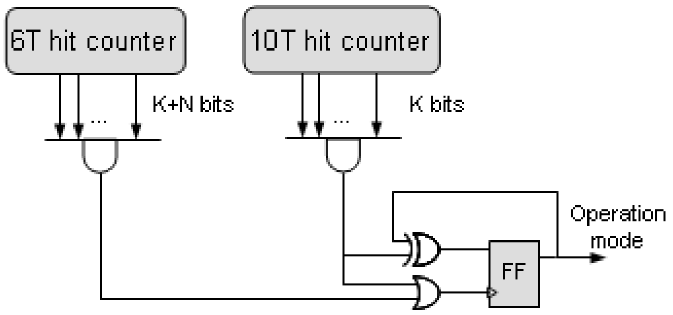


Figure 5.5: ADAM control logic.

the fraction of cache space devoted to 6T and 10T ways. The size of the 10T counter is smaller due to a smaller number of 10T ways with respect to the 6T ways. If the size of 10T counter is *K* bits, then the size of the 6T counter must be *K*+*N* bits. This means that by setting up the 6T hit count threshold 2^N times larger than that for 10T, we can detect with high accuracy whether 10T hits are above or below their expected value. In other words, the 10T hit counter saturates first if and only if more than $\frac{1}{2^N+1}$ of total hits occur in the 10T ways. Therefore, *N* determines the acceptable fraction of hits in the 10T ways. The value of *K* determines at which granularity decisions are taken.

For instance, for the 7+1 hybrid configuration, if the hit distribution is uniform across the cache lines, then around 7/8 of the total hits concentrate in the 6T ways and the remaining 1/8 of hits in the 10T ways. This means that by setting up the 6T hit count threshold seven times greater than that for 10T, we can detect whether 10T hits are above or below their expected value. Using a counter for 6T hits with *N* = 3 bits more than for 10T hits, suffices to detect this scenario. However, we decided to be more aggressive to reduce the number of accesses to the 10T way (10T ways are more power hungry than 6T ones), and thus, the 6T hit counter has *N* = 4 bits more than the 10T hit counter. Thus, the 10T hit counter saturates first if and only if more than $\frac{1}{2^4+1} = \frac{1}{17}$ of the hits occur in the 10T ways. Since changing the operation mode (swap or no-swap) has negligible cost, we take decisions at very fine grain. Thus, we use *K* = 7 bits for the 10T hit counter (up to 127 hits) and *K*+*N* = 11 bits for the 6T hit counter (up to 2047 hits). Based on the same methodology, we use 7 bits for the 10T hits counter (up to 127 hits) and 10 bits for the 6T hits counter (up to 1023 hits) for the 5+2 hybrid configuration. However, our mechanism is not limited to those particular values. Threshold sensitivity is studied in the evaluation section.

Figure 5.6 shows the algorithm for the ADAM mechanism. Initially, the operation

```
If access to the 6T ways is a hit:
1.   Increment 6T hit counter;
2.   If(6T hit counter is saturated)
2a.       Operation mode = Keep the same operation mode as
                           it was before hit;
2b.       Reset 6T and 10T hit counters;
3.   Return data.
Else if access to the 10T ways is a hit:
4.   Increment 10T hit counter;
5.   If(10T hit counter is saturated)
5a.       Operation mode = Change the operation mode that
                           was before hit;
5b.       Reset 6T and 10T hit counters;
6.   If(Operation mode = swap)
6a.       Swap hit 10T line with 6T LRU line;
7.   Return data.
Else (we have a miss here):
8.   Evict the LRU data from 10T ways;
9.   Move the LRU data from 6T ways into the evicted 10T
     location;
10.  Fetch new data into the free 6T location.
```

Figure 5.6: ADAM algorithm implementation.

mode can be either swap or no-swap. If there is a 6T hit, the 6T hit counter depicted in Figure 5.5 is incremented. If this counter is saturated, the operation mode is not changed (it is kept as it was before a hit) and 6T and 10T hit counters are reset. If there is a 6T miss, the 10T cache ways are accessed next. In case of a hit, the 10T hit counter is incremented. If this counter saturates, the operation mode is changed and 6T and 10T hit counters are also reset. Regardless of whether the 10T hit counter saturates, if swap is activated, the 10T line hit and the 6T LRU line are swapped.

In case of a miss in both 6T and 10T ways, we must evict the 10T LRU line, move the 6T LRU line to the available 10T entry and fetch the new data into the available 6T entry. Such process increases latency by the latency of an extra access to write data read from 6T ways into 10T ways. Note that we could simply evict 6T lines if the swap mode is not activated. However, we empirically observed that cache lines not in cache experience a burst of hits after being fetched and it is better fetching them to the 6T ways. Nevertheless, fetching those lines to the absolute LRU location if swap is deactivated produces negligible changes in the results.

5.3 Evaluation Results

In this section, we present execution time, total cache energy and energy-delay product (EDP) results for the different data management mechanisms at HP mode. Basically, we compare ADAM with with *Parallel*, *Sequential* and *Swap* mechanisms. Also, threshold sensitivity for the hit counters is studied.

All experiments have been performed following the evaluation methodology presented in chapter 3. We present execution time and total cache energy for the processor (Table 3.1) as the measure of the overall system performance. We account for the extra energy introduced by swapping as well as additional latencies to perform swaps correctly.

5.3.1 Performance and Energy

Execution time and energy vary across benchmarks. Thus, for the sake of clarity, results have been normalized with respect to the *Parallel* mechanism, which is the most optimal mechanism in terms of performance.

Figure 5.7 shows the execution time, total energy and EDP when running *BigBench* benchmarks on our 7+1 hybrid configuration. As stated before in Section 5.2, there are some pathological (and quite frequent) situations where *Sequential* and *Swap* lose significant performance. For instance, *Sequential* achieves the worst performance for g721_c and g721_d applications, since those programs use few cache lines intensively. By chance, some of those lines reside in the 10T ways, which degrades performance and does not allow saving dynamic energy. In fact, energy consumption increases due to the extra leakage (higher execution time implies higher leakage). g721_d experiences the highest performance degradation for this mechanism (14%). Conversely, *Swap* shows very low performance degradation for those "swap-friendly" applications (around 4%). *Sequential* achieves high performance (only 5% degradation) for regular applications such as gsm_c, gsm_d, mpeg2_c and mpeg2_d since those programs use cache lines quite homogeneously. However, *Swap* shows a high performance loss (up to 17% for mpeg2_d) because it performs many unnecessary swaps that end up moving to the 10T way those cache lines to be reused soon.

It can be seen that ADAM shows the best results in terms of execution time for all regular and irregular applications (up to 2.9% performance degradation for mpeg2_c and 1.7% on average). ADAM improves results even for those applications where *Sequential* and *Swap* perform well because of its adaptive nature that allows ADAM to adapt

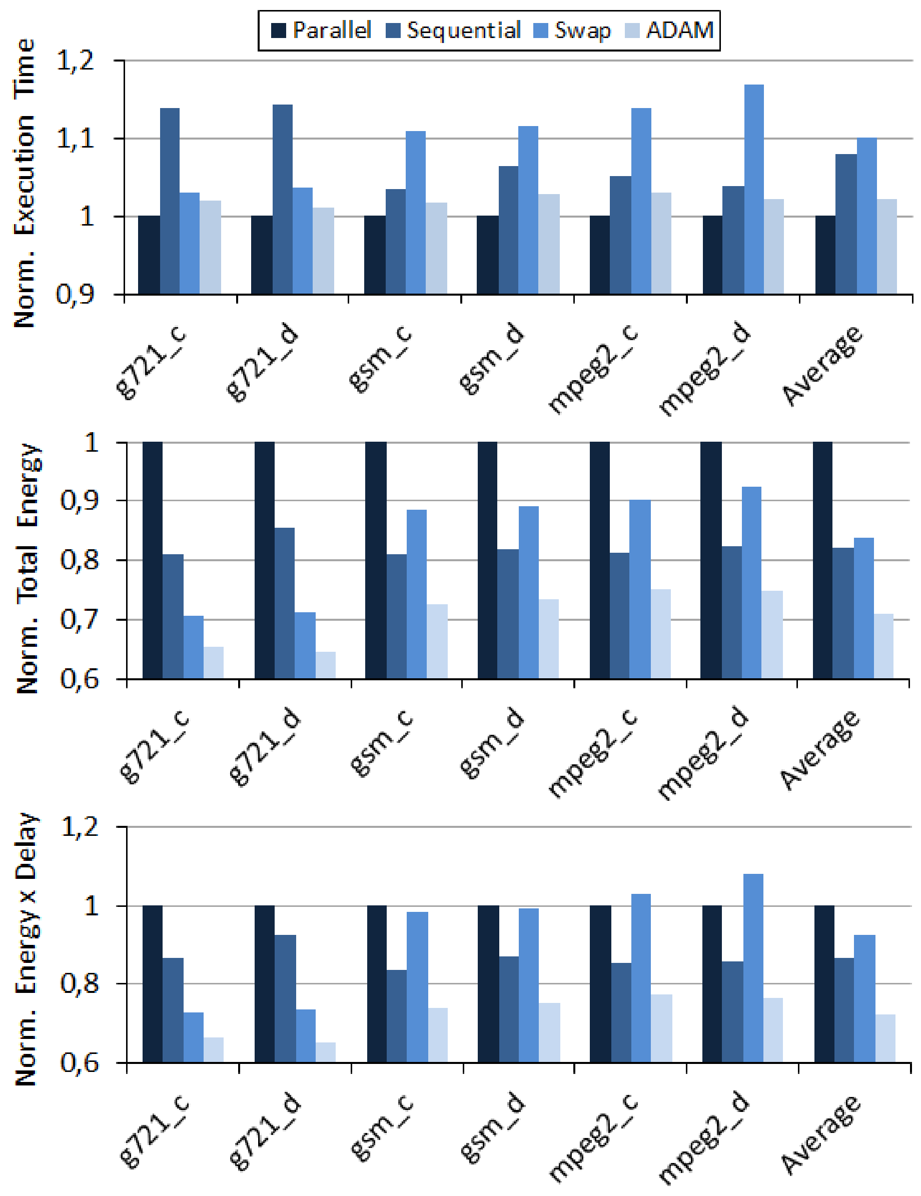


Figure 5.7: Normalized execution time, total energy and EDP for 7+1 hybrid configuration.

dynamically to different program phases.

Relative trends for total energy savings match quite well performance for *Sequential*, *Swap* and ADAM mechanisms. The *Parallel* mechanism provides the highest performance, but also the highest energy consumption because it accesses all cache ways on every request. For the remaining mechanisms, the higher execution time, the higher energy consumption is. Whenever any of those mechanisms hits the 10T way often, it saves little energy (both 6T and 10T ways are accessed) and performance is degraded due to

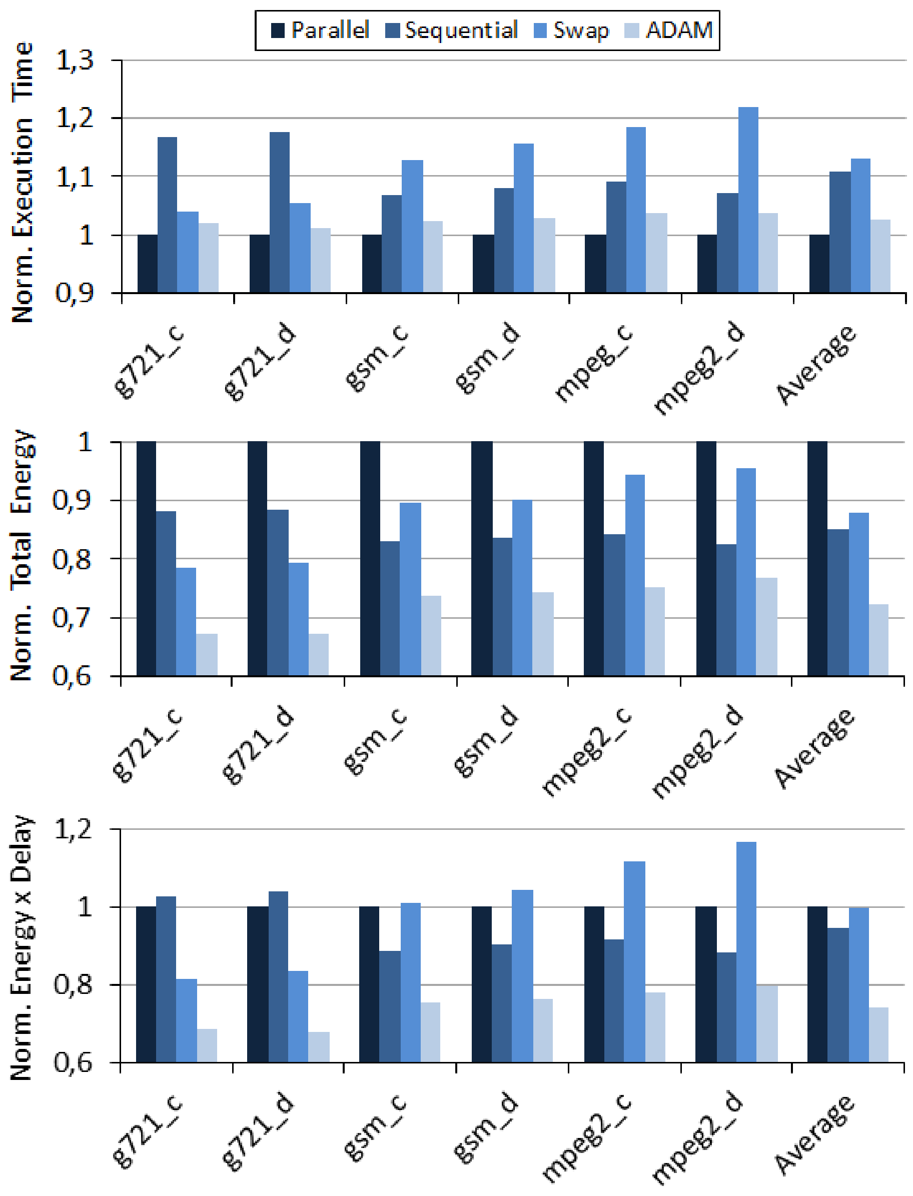


Figure 5.8: Normalized execution time, total energy and EDP for 5+2 hybrid configuration.

the sequential access to 6T and 10T ways. Since ADAM is the most effective data management mechanism concentrating hits in 6T ways, it is the most effective mechanism in terms of both performance and total energy. In particular, ADAM decreases energy consumption by 29% on average.

As expected, ADAM is the best performing mechanism in terms of EDP. ADAM reduces EDP by 27%, 21% and 16% with respect to *Parallel*, *Sequential* and *Swap* mechanisms respectively.

Similar trends are observed for the 5+2 configuration, as shown in Figure 5.8.

5.3.2 Disabling 10T Ways at HP Mode

We have also compared ADAM mechanism with the power gating technique [71] at HP mode. We have analyzed the case when 10T ways are turned off at HP mode, keeping only 6T ways enabled (i.e. 7KB 7-way for 7+1 configuration and 5KB 5-way for 5+2 configuration). Results are compared with respect to the same baseline (7+1/5+2 configuration with *Parallel* access).

In this scenario, execution time is increased up to 8% (for gsm_c, gsm_d, mpeg2_c and mpeg2_d applications) for 7+1 hybrid configuration due to reduced cache size and additional cache misses. Total energy is higher (17% on average for all applications) than when ADAM mechanism is in the place for 7+1 configuration basically because of extra energy consumption needed for additional off-chip memory accesses occured in this scenario.

Those facts confirm and justify our decision to keep ULE ways operating at HP mode and further prove the efficiency of ADAM mechanism. Similar trends are observed for the 5+2 configuration.

5.3.3 Hit Counters Size Sensitivity Study

Default saturation values of hit counters (see Figure 5.5) are determined by setting parameters K (i.e. 7 bits) and N (i.e. 4 bits). A lower threshold makes more aggressive choices whereas choosing a larger threshold may be so conservative and may miss more opportunities.

To better understand the effects of thresholds, we vary K (5, 7 and 9 bits), while keeping N = 4 bits. Similarly, we also vary N (3, 4 and 5 bits), while keeping K = 7 bits. Table 5.1 presents performance and energy results averaged across all benchmarks for the 7+1 configuration for those 5 different (K,N) configurations. All results are normalized with respect to the default (7,4) configuration. From the performance perspective, results show that all applications are highly insensitive to the thresholds (up to 1.8% performance variation across different thresholds). Whenever N is high (N=5) the operation mode is changed too aggressively. Conversely, whenever N is low (N=3) the 6T hit counter saturates quickly and the operation mode is changed in an excessively conservative manner. Since changing operation mode has negligible cost, all applications prefer to take deci-

Table 5.1: Normalized (to the default (7,4) configuration) execution time, total energy and EDP for different threshold values of hit counters for ADAM for 7+1 hybrid configuration.

	Configurations (K,N)				
	(5,4)	(9,4)	**(7,4)**	(7,3)	(7,5)
Execution time	0.988	1.018	**1**	1.004	1.007
Total energy	0.983	1.024	**1**	1.007	1.015
EDP	0.984	1.038	**1**	1.012	1.023

Table 5.2: Normalized (to the default (7,3) configuration) execution time, total energy and EDP for different threshold values of hit counters for ADAM for 5+2 hybrid configuration.

	Configurations (K,N)				
	(5,3)	(9,3)	**(7,3)**	(7,2)	(7,4)
Execution time	0.986	1.014	**1**	1.002	1.003
Total energy	0.978	1.019	**1**	1.003	1.013
EDP	0.981	1.027	**1**	1.011	1.018

sions at very fine grain (K=5), but difference in performance between this case and the default one (K=7) is very small (1.2%). On the other hand, if decisions are taken at coarse grain (K=9), performance degradation is negligible (1.8%).

From the energy perspective, the trends are similar as they are for performance. The reason is that those configurations inefficient for performance are also inefficient for energy due to either unnecessary swaps or unnecessary 10T accesses.

Similar trends are observed for 5+2 configuration, as shown in Table 5.2.

5.3.4 Guaranteed Performance

In terms of guaranteed performance, ADAM mechanism keeps exactly the same characteristics, thus the same interface to the WCET estimation tools as they were for the baseline case, except hit latency, which is still fixed and thus deterministic. In the worst case, hit latency is increased by few cycles needed for performing swaping correctly. However, WCET estimates are dominated by miss latencies as a relatively large number of accesses cannot be guaranteed to hit in cache and so miss latency must be assumed. Thus, WCET estimates are negligibly increased due to the increased worst-case hit latency. Therefore, proposed ADAM mechanism changes nothing in WCET analysis.

5.4 Summary

In this chapter, we have proposed a new, efficient Adaptive Data Management mechanism (ADAM) for hybrid voltage operation caches designed particularly for new, ultra-low-cost battery-powered systems. ADAM is a counter-based data management mechanism which dynamically detects hit distribution across the cache lines during program execution and activates/deactivates swapping to optimize performance and energy consumption at fine grain with negligible hardware overhead. ADAM saves significant energy (29% on average) in L1 caches with negligible performance degradation (1.7% on average), thus outperforming *all* existing data management approaches noticeably in terms of energy-delay product (EDP) across different cache configurations. In terms of guaranteed performance, ADAM mechanism changes nothing in WCET analysis.

Moreover, ADAM meets the golden requirement of this new market segment, which is design simplicity, as required to achieve reduced fabrication cost, increased integration and system efficiency.

Chapter 6

Efficient Cache Architectures Using Error Detection and Correction (EDC) Codes

Large memory cells are commonly used in cache memories to achieve acceptable robustness and target reliability at ultra-low voltages at the expense of higher energy consumption. The simplest way to reduce such energy consumption is to decrease the size of memory cells, but some faults will be expected mainly due to high sensitivity to process-induced V_{TH} variability. So far, fault-tolerant approaches for ultra-low voltages have shown to be effective from an average performance perspective and provide functional correctness, but fail to provide strong timing guarantees required for the worst-case execution time (WCET) estimation, as needed for critical applications in our target market. This chapter describes a new hybrid L1 cache architecture that overcomes the inefficiency of the large SRAM cells (e.g., 10T SRAM) in a manner that delivers energy and area efficiency without jeopardizing reliability levels to still provide strong performance guarantees.

6.1 Introduction

Cache memories are used in our target systems to reduce the number of slow and energy-hungry memory accesses, thus increasing the efficiency of the system. However, caches become the main energy consumer on the chip. Those caches use large memory cells to achieve high levels of reliability even at ULE mode, as needed by critical applications run on top. Decreasing the size of the memory cells for higher energy efficiency at the expense of higher failure rates is unacceptable in this environment. Faulty entries should be then disabled and strong performance guarantees required by critical applications would not be achievable [86]. Therefore, our aim is devising new energy-efficient cache designs without decreasing reliability levels to still provide predictable performance.

In this chapter, we propose a novel, single-Vcc domain, hybrid cache architecture, whose main characteristics are: (i) low energy consumption, (ii) simple design and (iii) high reliability levels, outperforming existing solutions [60]. In particular, our cache design relies on replacing energy-hungry bitcells (e.g., 10T) by more energy-efficient and smaller bitcells (e.g., 8T) enhanced with error detection and correction (EDC) features. We illustrate our cache architecture with two scenarios, depending on the reliability level of the baseline (**no coding** or **single error correction double error detection (SECDED)**), where 10T SRAM cells are replaced by smaller 8T SRAM cells (a) by keeping no coding at HP mode and by adding SECDED at ULE mode, whenever **no coding** is in place or (b) by keeping SECDED at HP mode and by replacing SECDED by double error correction triple error detection (DECTED) at ULE mode, whenever **SECDED** is in place. Our cache architecture achieves up to 14% average energy savings at HP mode and up to 42% at ULE mode, and area savings between 7% and 42% with respect to existing solutions [60] while keeping the same performance and reliability levels.

The rest of this chapter is structured as follows. Section 6.2 introduces the proposed cache architecture. Experimental results are presented in Section 6.3. Finally, Section 6.4 summarizes the main conclusions of this chapter.

6.2 Proposed Hybrid Cache Architecture

In this section, we first describe the cache architecture that we use as the baseline. Next, we present our proposal as well as the design methodology for the proposed architecture.

6.2.1 Baseline Architecture

We use a 8KB 6T+10T hybrid cache as the baseline [60] with 8 ways and 32B/line, where 7 ways are implemented with differential 6T SRAM cells and 1 way with 10T SRAM cells (7+1 for short), although our proposal is not limited to this configuration as shown later. Details about this hybrid cache can be found in chapter 4.

6.2.2 Proposed Architecture

10T SRAM cells [54] are particularly designed for robust fault-free[1] NST operation, but their energy and area overheads are large in order to match the same failure rate as 6T SRAM cells at high voltage [87]. Reducing these overheads would improve significantly area and energy efficiency of the target systems, because ULE cache ways are always enabled at both HP and ULE modes. However, simply decreasing the size of these large SRAM cells or replacing them by cheaper SRAM cells (e.g., 8T) for higher energy efficiency would increase failure rates. Faulty entries should be then disabled and strong performance guarantees required by critical applications would not be achievable [86].

In order to overcome the inefficiency of the large memory cells (e.g., 10T) in terms of area and energy, we propose a new, simple, energy-efficient cache design without jeopardizing reliability levels to still provide predictable performance.

We illustrate our proposed cache architecture with two scenarios depending on the reliability level of the baseline cache. In the first scenario, we consider a 6T+10T baseline cache where **no coding** is in place. In the second scenario, the baseline cache has higher reliability and all ways are **SECDED** protected to deal with *soft errors* (6T+SECDED+ 10T+SECDED). Our cache design relies on replacing energy-hungry bitcells (e.g., 10T) in **ULE ways** by more energy-efficient and smaller cells (e.g., 8T) enhanced with error detection and correction features to keep the same reliability levels, which are particularly critical at ULE mode. Figure 6.1 depicts our proposed cache architecture for the first scenario.

Reliability of ULE ways at HP mode in both scenarios is not an issue, because both 8T and 10T SRAM cells are more reliable (by some orders of magnitude) than 6T ones at high voltage, thus the *same* coding (none or SECDED) as that used for the baseline cache suffices. However, at ULE mode, *stronger* codes (SECDED, if none in baseline or DECTED, if SECDED in baseline) must be used, because smaller 8T SRAM cells are less reliable than 10T ones at NST Vcc. Therefore, we have:

- **Scenario A.** The baseline is a 6T+10T cache and no coding is in place. 10T SRAM cells are replaced by smaller and less reliable 8T SRAM cells by adding SECDED whenever no coding is in place (6T+10T vs. 6T+8T+SECDED). SECDED is only required to deal with *hard faults* in 8T SRAM cells at ULE mode. At HP mode, SECDED is simply turned off (6T+10T vs. 6T+8T).

[1]We assume that a bitcell is robust enough if its probability of failure is below a given threshold (e.g., 10^{-8}).

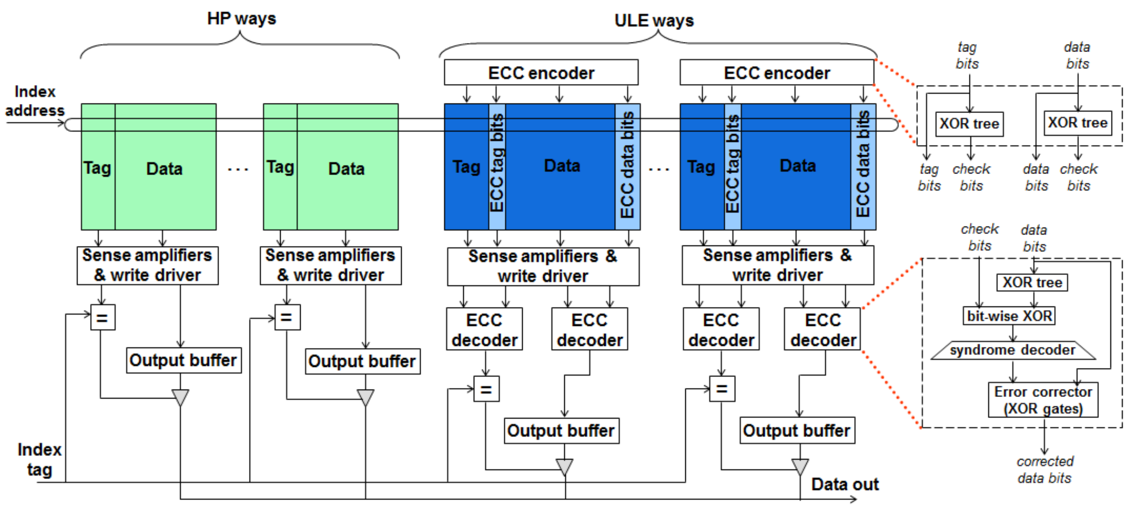


Figure 6.1: Proposed cache architecture for scenario A.

- **Scenario B.** The baseline has higher reliability than that of scenario A since all cache ways are SECDED protected to deal with *soft errors* (6T+SECDED+10T+SECDED). 10T SRAM cells are replaced by smaller and less reliable 8T SRAM cells by replacing SECDED (only for ULE ways) by DECTED whenever SECDED is in place to deal with *soft errors* (6T+SECDED+10T+SECDED vs. 6T+SECDED+8T+DECTED). DECTED is only required to deal with *hard faults* in 8T SRAM cells at ULE mode. At HP mode, DECTED is simply turned off since SECDED protection of 8T SRAM cells is sufficient to deal with *soft errors* at high Vcc (6T+SECDED+10T+SECDED vs. 6T+SECDED+ 8T+SECDED).

Note that we assume *hard faults* at ULE mode due to weak 8T SRAM cells in both scenarios. In general, hard faults in memories are covered with the technology used by construction (e.g., increasing transistor sizes), thus keeping their probability below a given acceptance threshold. Given that we weaken SRAM cells in the proposed architecture (replacing reliable 10T by less reliable 8T) we assume that *hard faults* can arise at ULE mode since their probability is above the acceptance threshold. Scenario A corresponds to the case where soft errors are not a concern in the baseline design. In scenario A, SECDED code suffices to correct a hard faulty bit in a word. Double-errors due to hard faults occur with a probability below the acceptance threshold. However, in scenario B where SECDED is in place to deal with *soft errors*, DECTED codes can correct both a soft error and a hard faulty bit in the same word. Note that we assume the worst case in scenario B (i.e. soft error and a hard fault happen in the same word) to ensure that lumped failure probability for hard and soft errors is always below the acceptance threshold (target cache yield). In that sense, our design may be overdesigned since, perhaps, SECDED could suffice to keep the combined hard and soft fault probability below the acceptance threshold. However, we deliberately make no assumption on how soft errors occur and design our cache to guarantee reliable operation regardless of the actual soft error distribution in space and time.

Delay, energy and area overheads introduced by EDCs are considered in our calculations, as described later in section 6.3. Turning off HP ways at ULE mode is done by using the gated-Vdd technique [71]. Overheads are negligible, as explained in [71]. The processor itself is responsible for gating or ungating the corresponding cache block and writting back dirty lines on a Vcc change. Performance impact to disable HP ways due to writing back dirty lines is negligible, because mode changes occur seldom [78].

Note that proposed caches exhibit deterministic performance behavior at both scenarios since each operation mode provides *the same* cache space and arrangement as in the baseline cache, and fixed latency. Therefore, strong performance guarantees remain identical.

In the rest of the chapter we use differential 6T SRAM cells for HP ways, 8T SRAM cells for ULE ways, Hsiao SECDED and DECTED codes [20] and 32nm technology. However, the proposed cache architecture is not limited to any particular Vcc level, SRAM cell type, technology node, type of protection or reliability level. This is because different SRAM cells exhibit the same trade-off between SRAM cell size and failure probability.

6.2.3 Implementation Details

In this section, we describe the implementation details for scenario A (scenario B is analogous). Each SRAM cell is sized by using the analysis based on importance sampling proposed by Chen et al. [22], assuming 6σ random variations in V_{TH} for NST Vcc considering read, write and hold failures in the 32nm technology node. We apply random V_{TH} values to each transistor in a SRAM cell and check for read, write and hold failures for the chosen cache size using HSPICE[2].

We first describe SRAM cell sizing in the baseline. For the chosen NST Vcc and

[2]We use the low-power 32nm Predictive Technology Model [93] with nominal threshold voltages for NMOS and PMOS transistors of V_{TH_n} = 350 mV and V_{TH_p} = -320 mV respectively. Standard deviation of V_{TH} for NMOS/PMOS transistor is σV_{TH_n} = 30 mV and σV_{TH_p} = 24 mV.

```
10T SRAM cells sizing in the baseline:
1. For chosen NST Vcc and adjusted frequency at ULE mode, size 10T bitcells
   using the analysis based on importance sampling proposed by Chen et al. [22]
   in order to provide a target cache yield ($Y_{10T}$)
Replacing 10T SRAM cells with 8T SRAM cells and EDC:
1. Set minimal transistor sizes possible (i.e. $3\lambda$ width) for 8T SRAM cells for
   target technology node (32nm)
2. Calculate 8T bitcell's bit failure probability $P_{f8T}$ using Chen's analysis [22]
   assuming 6$\sigma$ random variations in $V_{th}$ considering read, write and hold failures
3. Calculate failure probability ($P_{total}$) of EDC-protected cache
4. Calculate cache yield ($Y$)
5. If ($Y < Y_{10T}$)
5a.     Increase transistor sizes (widths) by $0.5\lambda$
5b.     Go to step 2
6. Else
6a.     Optimal cell size is obtained
```

Figure 6.2: Implementation details for scenario A.

reduced frequency at ULE mode, we size the 10T SRAM cells to provide a target cache yield Y_{10T} (e.g., 99$\%$ for scenario A). Then, for the chosen high Vcc (e.g., 1V) and increased frequency at HP mode, the 6T SRAM cells are sized to match the same bit failure rate as 10T SRAM cells at ULE mode. Note that in scenario B, the 10T SRAM cells are SECDED protected to deal with soft errors, and cache yield in that case ($Y_{10T+SECDED}$) can be calculated by using elementary probability calculations, analogously to scenario A.

Next, we determine the size of 8T SRAM cells protected with EDC in order to replace 10T SRAM cells in ULE ways[3] as shown in Figure 7.3. We first set minimal transistors sizes for 8T SRAM cells (3λ width for all transistors) and then calculate the bit failure probability (P_{f8T}) for the chosen NST Vcc by using Chen's analysis [22]. Then, we define data and tag words to have 32 and 26 bits respectively, and protect them at such granularity. The probability of having *fault-free* tag/data words and the cache yield (Y) are:

$$P(tag/data) = \sum_{i=0}^{1} (1 - P_{f8T})^{n+k-i} P_{f8T}^{i} \binom{n+k}{i} \tag{6.1}$$

$$Y = P(data)^{DW} P(tag)^{TW}, \tag{6.2}$$

[3]Remember that only ULE ways are active at ULE mode and we consider only ULE mode.

where DW and TW are the total number of data and tag words in cache respectively, n is the number of bits of tag or data words, k is the number of added check bits (i.e. 7 bits for SECDED) to each tag/data word and i is the number of hard faults in a tag or data word. Note that in case of no coding (scenario A), SECDED suffices to correct one hard fault in a word (8T+SECDED), whereas in scenario B, DECTED can correct both one soft error and one hard fault in the same word (8T+DECTED). If the yield obtained (Y) is lower than required (e.g., Y_{10T} for scenario A or $Y_{10T+SECDED}$ for scenario B), transistors sizes (widths) are increased by a step value equal to 0.5λ and yield is calculated again. Once yield is high enough, we have a reliable-enough, yet small, SRAM cell size. The algorithm is summarized in Figure 7.3.

We assume in our algorithm that the operating Vcc is determined by the system requirements. However, our algorithm can be extended to increase either Vcc or transistor sizes so that further efficiency is achieved. Nevertheless, such analysis is beyond the scope of our study.

6.3 Evaluation

In this section, we evaluate the proposed cache architecture at both HP and ULE modes, and analyze the trade-offs at ULE mode for different Vcc values.

6.3.1 System Modeling

All experiments have been performed following the evaluation methodology presented in chapter 3. Several hybrid cache microarchitectures have been implemented using heterogeneous SRAM cell types at a coarse granularity, as explained in section 6.2. Moreover, we have extended tag and data words (26 and 32 bits respectively in our case) with check bits (7 bits for SECDED, 13 bits for DECTED) and taken into account energy and area overheads introduced due to those check bits. In our simulations, we account additional latency of one clock cycle for SECDED/DECTED encoding and decoding as well as the energy consumed by the extra EDC circuits. Energy consumption of EDC encoders and decoders has been obtained by performing HSPICE simulations. Note that in our evaluation we did not generate a number of cache setups with different fault mappings, because all cache entries go through EDC regardless of being faulty or not.

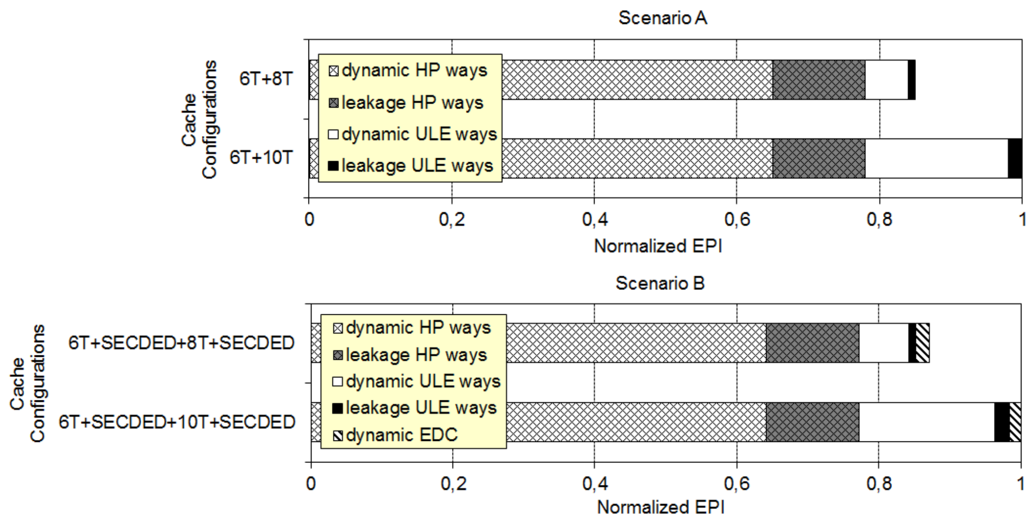


Figure 6.3: Normalized EPI breakdowns at HP mode for scenarios A and B.

6.3.2 Results and Discussion

We present cache energy per instruction (EPI) and cache area results at HP and ULE modes comparing the proposed cache architecture with the baseline designs for both scenarios described in section 6.2. Execution time and energy vary across scenarios. *Thus, for the sake of clarity, results have been normalized with respect to the baseline configuration in both scenarios.* Along with this, a Vcc sensitivity study at ULE mode is presented. All relevant comparisons involve caches with the same characteristics in terms of cache size and associativity, thus the number of off-chip accesses and their pattern remain unchanged. Given that off-chip behavior remains unchanged and other memory latencies do not change the trends reported later, we do not include off-chip memory energy in our results.

6.3.2.1 HP mode

Figure 6.3 shows the normalized average EPI for both scenarios at HP mode. All benchmarks show minor differences to the average. The main reason is that the fraction of cache memory accesses (instruction and data) and execution behavior of the different benchmarks is quite similar given that their workloads fit pretty well in cache, which will be the case in real systems. Since caches are the main energy contributor in these extremely simple processors, *cache behavior dominates full processor behavior*.

Our architecture shows energy savings of 14% and 12% on average for scenario A and

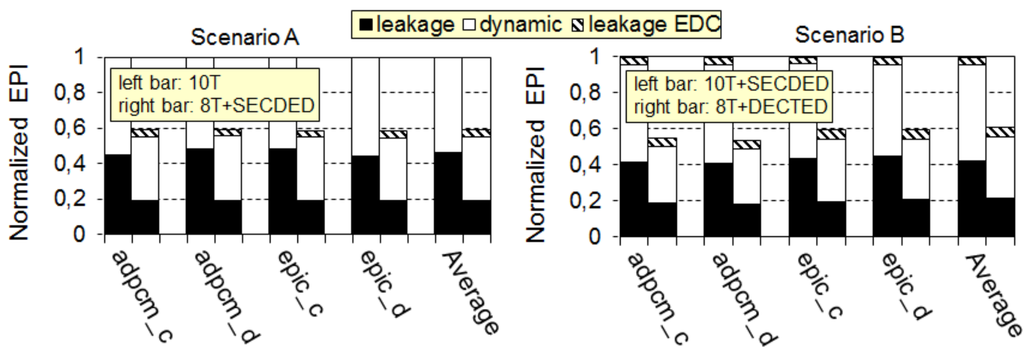


Figure 6.4: Normalized EPI breakdowns at ULE mode for scenarios A and B.

scenario B respectively. This is due to the smaller transistor sizes for 8T cells with respect to the 10T cells and thus, reduced dynamic energy (which is the dominant energy factor at high voltage). Our architecture **does not experience any performace degradation (no latency overhead)** since 8T cells are as reliable as 6T at high voltage, so they use the same coding as in baseline (none for scenario A, SECDED for scenario B).

6.3.2.2 ULE mode

HP ways are turned off at this mode, so only ULE ways keep operating. Leakage increases at ULE mode whereas dynamic energy is still a significant energy factor. Figure 6.4 shows the normalized EPI breakdowns across all benchmarks for scenario A and B at ULE mode. Caches remain to be the main energy contributor and access frequency is not drastically different across benchmarks, so effects on different sources of energy on each benchmark are relatively similar, because dynamic and leakage cache energy is impacted in a very similar way. Thus, all benchmarks observe similar trends.

When EDC codes are used, smaller transistors are needed for 8T cells and thus, relative dynamic and leakage energy consumption is lower than for 10T cells. Smaller transistors keep capacitances lower and reduce dynamic energy, which scales *linearly* with capacitance, whereas delay and thus, leakage scales *exponentially*. Hence, the relative leakage energy savings are larger than those for dynamic energy. Taken all together, the normalized average EPI reductions are 42% and 39% for scenario A and B respectively. *Performance variation due to the extra cycle for EDC encoding/decoding is negligible (around 3% increase in execution time in all cases).*

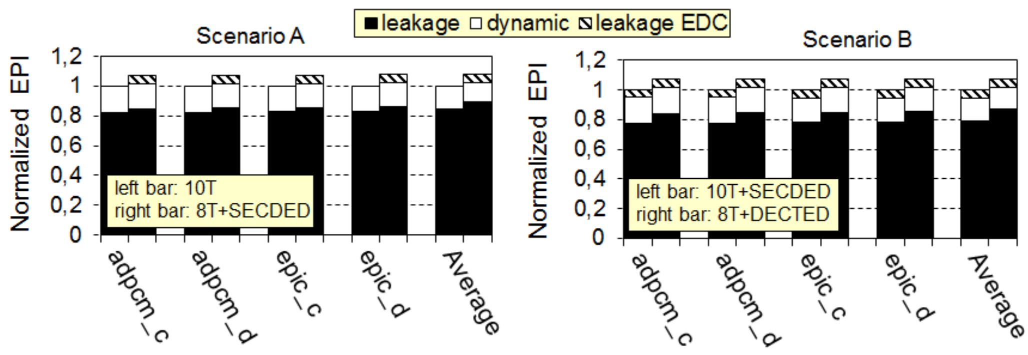


Figure 6.5: Normalized EPI breakdowns at ULE mode for scenarios A and B when Vcc=200 mV.

6.3.2.3 Vcc sensitivity study at ULE mode

Different Vcc may offer different performance/energy trade-offs, which are particularly important at ULE mode. In fact, the primary concern at ULE mode is reaching the minimum energy operation point (E_{min}). Aggressive Vcc scaling into the subthreshold regime may not yield energy optimality because the lowest functional Vcc is not the most energy efficient point due to the exponential delay increase, and thus leakage energy increase [75]. Conversely, conservative Vcc scaling may miss large dynamic energy savings.

In this section, we extend our analysis towards understanding trade-offs at ULE mode when different Vcc values are considered. We study the sensitivity of the proposed caches to different Vcc values in the NST range, thus identifying the optimal voltage region through experimental models [62].

When lowering Vcc, using EDC coding, which allows smaller transistor sizes, is beneficial to some extent until the subthreshold component of bitline leakage current dramatically increases and becomes the dominant factor. In general, in the subthreshold regime, drain current depends exponentially on the gate voltage and any device upsizing will result in a marginal change in the drain current [18]. For instance, when Vcc is set to 200 mV (subthreshold), leakage becomes the dominant energy factor due to the increase of the subthreshold component of bitline leakage current. This phenomenon does not affect the 10T SRAM cells considered due to their built-in feedback topology [54]. Compared to 8T cells, 10T cells exhibit reduced pull-down transistor strength at the cross-coupled inverter node due to the stacked NMOS transistors. Thus, bitline diffusion and wordline gate capacitance are decreased despite the extra transistors, which reduces bitline leakage current. However, leakage is increased in the single-ended 8T SRAM cells with decoupled read and write ports, because the read bitline introduces additional data-dependent leakage path during read operations[4]. This issue can be eliminated by means of the assist peripheral circuitry, but this substantially increases complexity [22, 54]. As a result, baseline 10T caches show lower leakage energy at this subthreshold voltage than proposed designs. Figure 6.5 shows normalized EPI breakdowns across all benchmarks when Vcc is set to 200 mV. Average EPI overheads are 7% and 8% for scenarios A and B respectively.

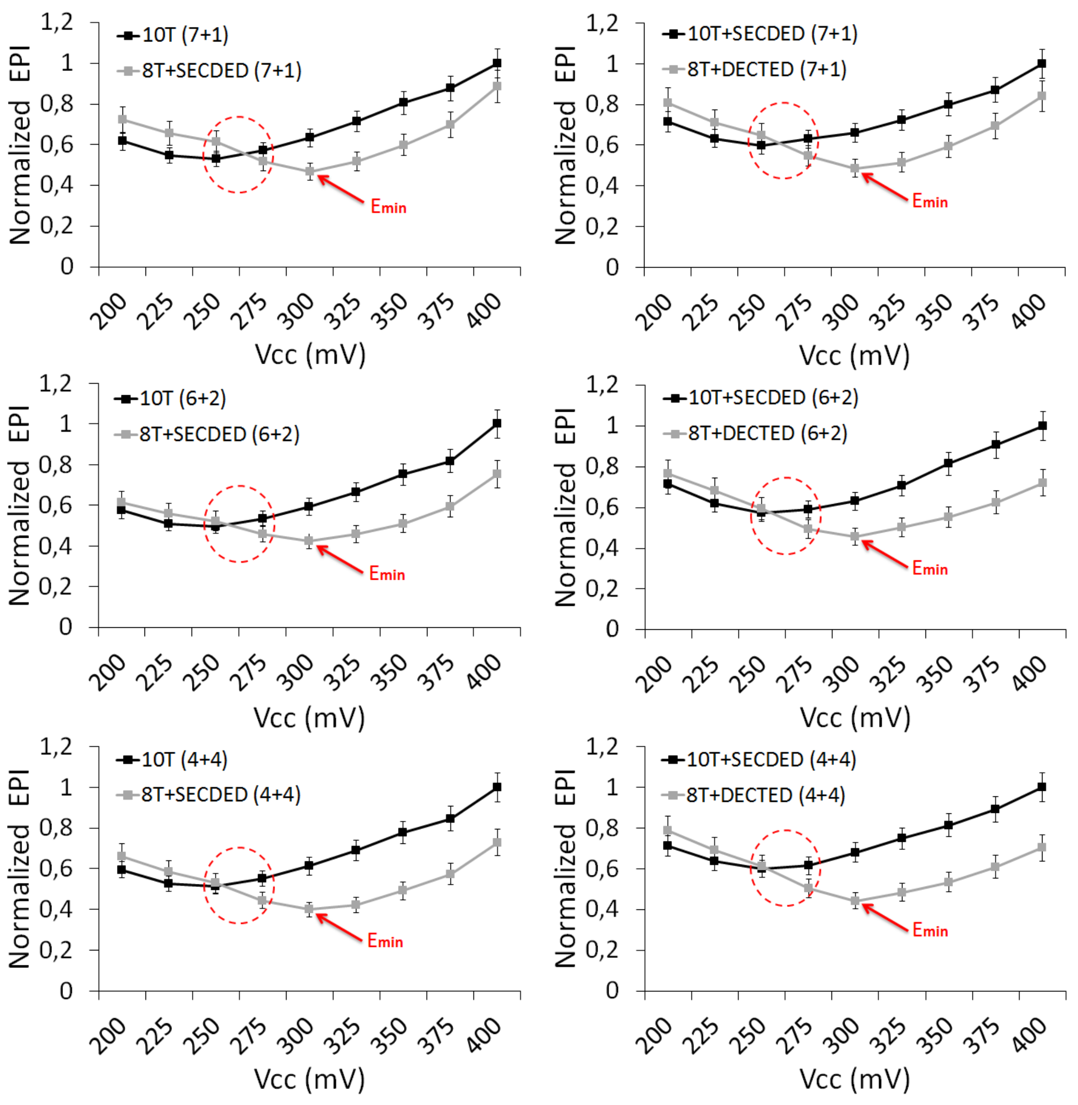
To better understand the effects of choosing the proper Vcc level at ULE mode, we vary Vcc (and adjust the frequency) from 200 to 400 mV in steps of 25 mV. The corresponding adjusted frequency for 200 mV, 225 mV, 250 mV, 275 mV, 300 mV, 325 mV, 350 mV, 375 mV and 400 mV are 100 kHz, 700 kHz, 2 MHz, 3.5 MHz, 4 MHz, 4.5 MHz, 5 MHz, 5.5 MHz and 6 MHz respectively. Operating frequencies are chosen to fit the cache access time to 2 clock cycles (see Table 3.1). In fact, the operating frequency fits the slower 10T-based cache[5]. However, we assume the same cache latency for all 8T- and 10T-based caches, *although it is unfavorable to our proposed cache*. Besides 7+1 hybrid configuration, we have evaluated other hybrid designs such as 6+2 and 4+4.

Figure 6.6 shows normalized average energy per instruction (EPI) across all benchmarks when varying Vcc. Results for both scenarios are normalized with respect to the baseline cache (referred to as 10T in Figure 6.6) at 400 mV for all configurations. It can be seen that the proposed cache is more energy efficient than the baseline one until a "cross-point" is reached. This cross-point is in the range [250, 275] mV. Baseline designs are shown to be better in terms of energy below 250 mV due to the lower subthreshold bitline leakage current for 10T cells. Nevertheless, we observe that our proposed architecture is more energy efficient because achieves lower E_{min} than the baseline, as shown in Figure 6.6 for both scenarios, A and B. The proposed architecture achieves E_{min} around 300 mV whereas baseline caches achieve it around 250 mV. Since average performance at 300 mV is higher than that at 250 mV, the proposed architecture is the most efficient one in terms of both energy and performance.

Figure 6.7 presents normalized cache area for the 7+1 hybrid configuration in scenarios A and B with respect to the pure 6T design operating at 1V. As seen in Figure 6.7, our

[4]Bitline voltage drops/rises across the pass transistor of the read port irrespective of the value of the data stored for the unaccessed SRAM cells of a bitline [52].

[5]10T cells exhibit up to 20% longer write access time across all Vcc levels considered than the 8T cells. Such penalty is due to the series of stacked NMOS transistors in the pull-down path in 10T cells. Read access time variation is marginal because 8T cells also have serial-stacked NMOS transistors in the read port.

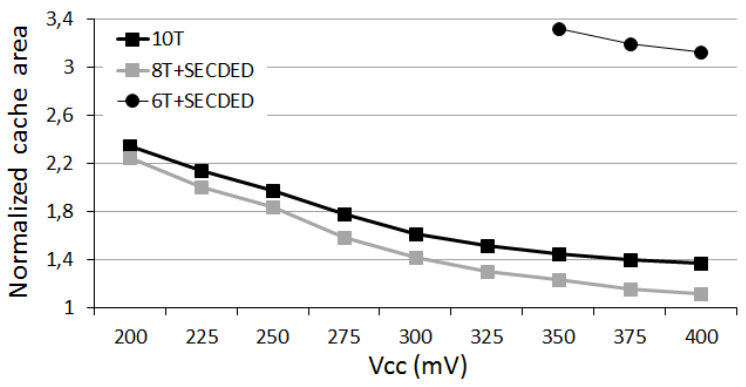


(a) Scenario A. (b) Scenario B.

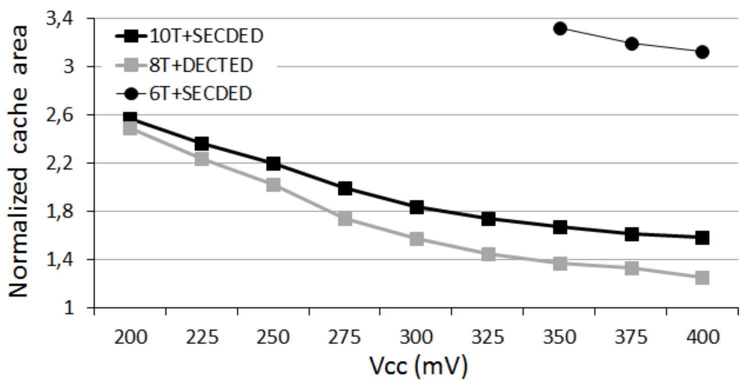
Figure 6.6: Normalized average EPI for 7+1, 6+2 and 4+4 hybrid cache configurations in scenario A (99% yield) and scenario B (99.7% yield) when varying Vcc at ULE mode. Error bars represent minimum and maximum variation across benchmarks.

architecture is always the most area-efficient across all voltages considered in both scenarios (reduction between 7% and 24%). We also show that 6T designs, even if protected with SECDED, incur an unaffordable area penalty at NST voltages (more than 400% at 200 mV), so those designs are unsuitable for such low voltages.

Figure 6.8 shows normalized average execution time across all benchmarks when varying Vcc in scenario A. Results are normalized with respect to the 7+1 10T baseline



(a) Scenario A.



(b) Scenario B.

Figure 6.7: Normalized cache area for 7+1 hybrid configuration in: (a) scenario A (99% yield) and (b) scenario B (99.7% yield), when varying Vcc at ULE mode.

cache at 400 mV. We observe that proposed cache exhibits up to 5% increase in execution time due to the additional clock cycle for EDC encoding/decoding[6]. Note that the operating frequency is chosen to support the slower 10T-based cache in order to have the same cache latency for the baseline and the proposed 8T-based cache although it is unfavorable to the proposed architecture.

In summary, our architecture is the optimal one in energy and area beyond 250-275 mV. Conversely, it is the best one only in terms of the area below such voltage level. Likewise, from the E_{min} perspective, our architecture is more efficient because it achieves always the lowest E_{min}. In general, small L1 caches with high activity, as required in our systems, achieve E_{min} when Vcc is beyond the subthreshold region (near-threshold) [22]. Our designs outperform existing ones in all metrics at such voltage region.

[6]Execution time in scenario B is the same for the baseline and the proposed cache because both caches use EDC codes. Therefore, those results are omitted.

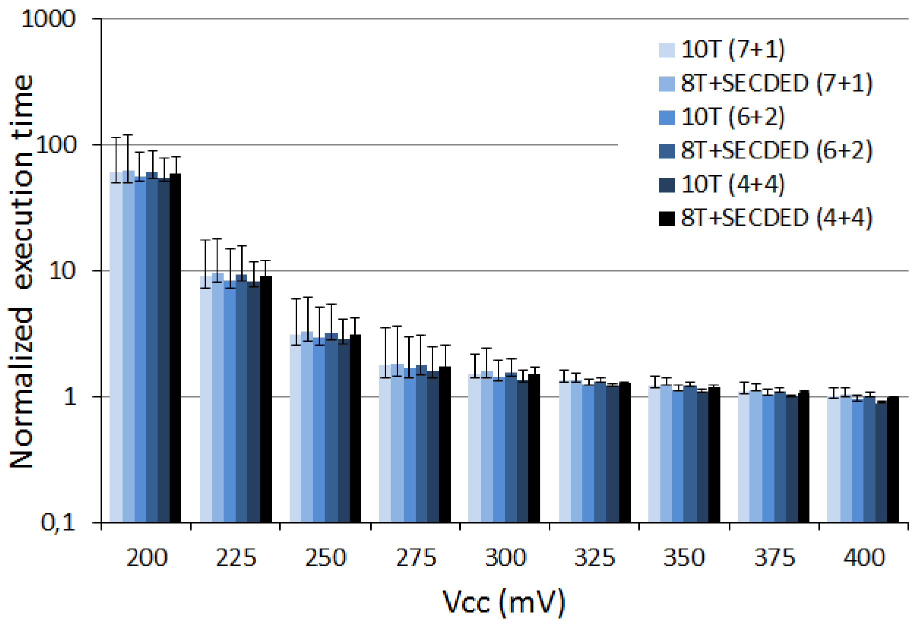


Figure 6.8: Normalized (w.r.t. 10T 7+1 configuration at 400 mV) average execution time for 7+1, 6+2 and 4+4 hybrid cache configurations in scenario A when varying Vcc at ULE mode. Error bars represent minimum and maximum variation across benchmarks. Bars in logarithmic scale.

6.3.3 Overall Energy Savings

In this section, we report overall energy savings for the whole lifetime of the artificial application which we described in section 3.2.6. Figure 6.9 shows total energy breakdown for the whole application lifetime for scenario A. Basically, the main conclusions, trends and findings are similar to those discussed in section 4.3.2.6, so here we omit further discussion.

6.3.4 Guaranteed Performance

In terms of guaranteed (WCET) performance, the proposed cache architecture provides exactly the same number of available fault-free cache lines per set as in the baseline, thus guaranteeing the same WCET performance. Therefore, WCET estimation is not more complex than for the baseline architecture. Note that hit latencies are increased by 1 cycle needed for EDC encoding and decoding since we must assume always the case where the cache line is faulty and fault needs to be corrected. However, the impact in WCET estimates of higher hit latencies is largely below that of allowing faulty lines and thus,

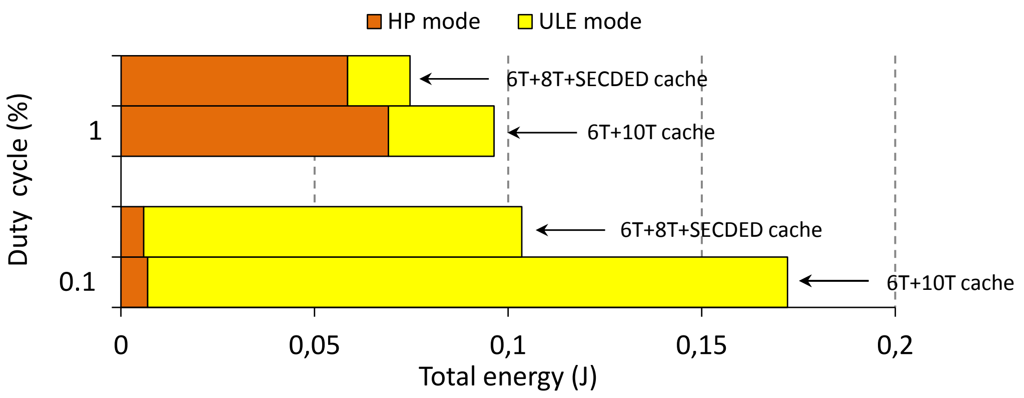


Figure 6.9: Total energy breakdown for the whole application lifetime for the baseline 6T+10T and proposed 6T+8T+SECDED cache designs showing energy contribution of each operation mode for different duty cycle values.

considering some accesses as misses instead of hits.

6.4 Summary

In this chapter, we propose new, single-Vcc domain, hybrid-voltage operation cache architectures for ultra-low-cost (e.g., below 1 USD) battery-powered emerging systems. We show that energy-hungry SRAM cells (e.g., 10T), needed for reliable ultra-low voltage operation, can be replaced by more energy-efficient and smaller cells (e.g., 8T) enhanced with EDC codes to improve energy and area efficiency without jeopardizing reliability levels to still provide strong performance guarantees, as needed for critical applications.

Experimental results show that the proposed architecure achieves up to 14% energy reduction (on average) with respect to existing ones without any impact on average performance at HP mode. Likewise, we analyze the efficiency of the proposed designs at ultra-low voltage operation (ULE mode) in order to identify the energy-optimal voltage region. Our proposed cache architecture is proven to outperform existing ones in energy and area above 250-275 mV (i.e. near-threshold regime), where the lowest E_{min} is achieved for small L1 caches with high activity as those in our target systems. Our results show that the proposed architecture outperforms existing ones in terms of both energy and performance. Further, our analysis identifies the main limitations of our approach at subthreshold regime. Also, the proposed architecture exhibits deterministic behavior by providing strong guarantees on the cache caracteristics as they were for the baseline case, thus changing nothing in WCET analysis. Finally, although our analysis is performed considering 8T and 10T SRAM cells, our approach is not limited to any particular Vcc level, technology node, SRAM cell type or EDC scheme since different SRAM architectures exhibit the same trade-offs between cell size and robustness.

Chapter 7

APPLE: Adaptive Performance-Predictable Low-Energy Caches

In the previous chapter, we have shown that energy efficiency can be achieved by joint optimization of SRAM cell size and EDC codes without jeopardizing reliability levels to provide strong performance guarantees. The cache architecture proposed in this chapter addresses the same problem and follows a philosophy similar to the one described in the previous chapter. Basically, energy efficiency is achieved by replacing large memory cells by smaller ones; however, now cache-assist structures, i.e., an adapted victim cache, are used instead of EDC codes, thus enabling fault tolerance due to disabled faulty cache lines to still provide predictable performance.

7.1 Introduction

In the previous chapter, we proposed a single-Vcc domain L1 cache architecture for reliable hybrid-voltage operation, which meets all stringent needs of our target market. Our cache architecture has shown that replacing energy-hungry SRAM cells (e.g., 10T [54]) by more energy-efficient and smaller SRAM cells (e.g., 8T [46]) enhanced with error detection and correction (EDC) features provides significant energy and area savings without jeopardizing reliability levels to still provide strong performance guarantees. In this chapter, we investigate alternative solutions, following the same philosophy presented in the previous chapter: decrease the size of memory cells for higher energy and area efficiency at the expense of increased fault rates, but enabling sufficient fault tolerance to still provide predictable performance.

In particular, here we propose efficient, but simple Adaptive Performance-Predictable Low-Energy (APPLE) single-Vcc domain L1 cache designs for reliable hybrid voltage operation, which meet *all* stringent needs of our target market. APPLE caches rely on replacing large memory cells by more energy-efficient and smaller cells enhanced with extra cache lines set up in a cache-assist structure, i.e., an adapted victim cache, to allow **extra associativity for some cache sets** that may need it due to disabled faulty cache lines. Experimental results show that APPLE caches achieve significant average energy and area reductions (42% and 12% for energy at HP and ULE mode respectively and 23% for the area) with a negligible *average* performance impact (less than 1.7% on average) with respect to existing solutions [32, 60, 94] without jeopardizing reliability levels to still provide strong performance guarantees.

The rest of the paper is organized as follows. Section 7.2 presents APPLE cache designs. Experimental results are reported and discussed in section 7.3. Section 7.4 presents the main findings of this chapter.

7.2 APPLE Cache

In this section, we describe the chosen baseline cache architecture and the proposed APPLE cache designs.

7.2.1 Baseline Cache Architecture

We have chosen a set-associative cache organization and LRU replacement policy as the target of our study, given that most L1 caches in existing embedded chips implement them, although significant parts of our study can be easily reused for other cache organizations and replacement policies.

We use a hybrid-operation, single-Vcc domain cache design particularly suited for our target market [60] as the starting point. In particular, we use an 8KB 6T+10T hybrid cache as the baseline [60] with 8 ways and 32B/line, where 6 ways are implemented with differential 6T cells and 2 ways with 10T cells [54], although our proposal is not limited to this configuration.

7.2.2 APPLE Cache Architecture

APPLE cache design relies on replacing large and energy-hungry (*strong*) SRAM cells by energy-efficient and smaller (*weak*) cells in a selected cache subset (e.g., some cache ways in a set-associative cache) enhanced with extra cache lines set up in a cache-assist structure, i.e., an adapted victim cache [49] to keep the same reliability levels despite the potentially disabled faulty cache lines. We illustrate our cache designs with a scenario where *strong* 6T cells ($6T_S$) in **all** HP ways in the baseline are replaced by *weak* 6T cells ($6T_W$) (see Figure 7.1(a)). Analogously, *strong* 10T cells ($10T_S$) in **all** ULE ways can be replaced by *weak* 10T cells ($10T_W$).

Reliability of ULE ways at HP mode is not an issue because $10T_W$ cells are still largely more robust than $6T_W$ ones at high voltage. However, some faults may be expected in HP ways due to decreased robustness of the $6T_W$ cells. Then, faulty cache lines must be disabled which makes performance unpredictable.

To avoid this, we set up a new structure in the cache in order **to have extra associativity for some cache sets that may need it due to disabled faulty cache line(s)**, thus guaranteeing the same number of fault-free cache lines per set as in the baseline. We refer to this structure as High Performance Victim Cache (HPVC). The HPVC is an adapted victim cache with spare entries. Conventionally, victim caches are fully-associative and their entries are comprised of a *valid bit* and *tag and data* space to store a cache line. Given that the *set number* is already part of the *tag* (see Figure 7.1(a)), the HPVC requires one extra field, i.e., *lock bits* - indicating whether a particular victim cache entry is being used for replacement of a faulty cache line. Similarly to [49], the HPVC is accessed in parallel with the L1 cache, thus without any impact on the L1 cache latency.

Similarly to existing fault-tolerant state-of-the-art approaches [2, 87], tags are extended with one (faultiness) bit, indicating whether the data block is faulty and there are mechanisms in place to detect and disable faulty storage as well as configure the extra fields of the HPVC properly at boot time. Fault detection, correction and diagnosis are out of the scope of this paper. Replacement bits, dirty line and valid bits are assumed to be hardened (e.g., with $10T_S$) as their relative impact on cache energy and area is negligible.

The number of acceptable faulty cache lines can be less or equal to the number of extra HPVC entries. Therefore, using a number of extra entries increases the fault tolerance of the cache. The HPVC is implemented with $6T_W$ cells, so the faults may also be expected in the HPVC itself. In case of a faulty entry in the HPVC, it will be disabled and there will be one less HPVC entry. Therefore, the HPVC replaces faulty entries in both L1 cache and HPVC itself. We consider this issue when calculating the number of HPVC entries required, as described later.

Generally speaking, our approach is in spirit similar to the conventional redundancy which basically introduces a small number of redundant columns and/or rows of SRAM cells to replace a column or row that contains a faulty SRAM cell (or cells) [30]. However, our proposed technique does not require expensive on-chip fuses [30] to record the locations of defective cache lines identified during a testing. Our technique requires usual testing (e.g., at boot time or just once before selling the chip) to allocate spare lines reusing existing cache assist structures such as victim caches.

7.2.2.1 APPLE Cache Operation

The cache works as follows:

- **Cache hit.** In this scenario, a hit cache line is served as usual either in the L1 cache or in the HPVC (accessed in parallel). In order to keep replacement (i.e., LRU) information consistent as in a fault-free system, there are two cases depending on whether a hit occurs in the L1 cache or in the HPVC:

  – If a hit occurs in the L1 cache, LRU information is updated as in a fault-free system.

  – If a hit occurs in the HPVC (either locked or non-locked entry), the hit cache line is swapped with the LRU Fault-Free (LRU-FF[1]) cache line in the corresponding cache set. During the regular access, hit cache line is both gated onto the data bus and written to the swap buffer. Then, swapping is done in two phases: (i) the LRU-FF cache line is written to the HPVC into the hit cache line position and (ii) swap buffer content is written to the LRU-FF position in the corresponding cache set. In order to perform swapping successfully, we stall any access to the cache during swapping for the latency of 2 cache accesses. Stalling accesses to the cache during some cycles is as easy as keeping the cache ports busy to prevent the port arbiter from issuing new accesses. Swapping ensures that the Most Recently Used (MRU) line resides

[1]The LRU-FF cache line can be selected by slightly-modified LRU stack, similar to that used in the existing line-disabled approaches [7, 87] (see Figure 7.2). For example, if the LRU cache line is faulty, then the LRU-1 line becomes the LRU-FF. The overhead for the L1 cache is less than 0.1% in transistor count.

in the L1 cache while the LRU is in the HPVC simultaneously. Since the LRU lines typically experience a few hits, the performance impact is expected to be small.

- **Cache miss.** In case of a miss, a new line from memory is fetched and placed into the LRU-FF position in the corresponding cache set and becomes the MRU line. The LRU-FF line is moved to the HPVC. If there is any locked entry for that set, the LRU-FF locked line is replaced with the line coming from the cache. Otherwise, a non-locked LRU-FF line is replaced. The line replaced in the HPVC is simply evicted to the memory.

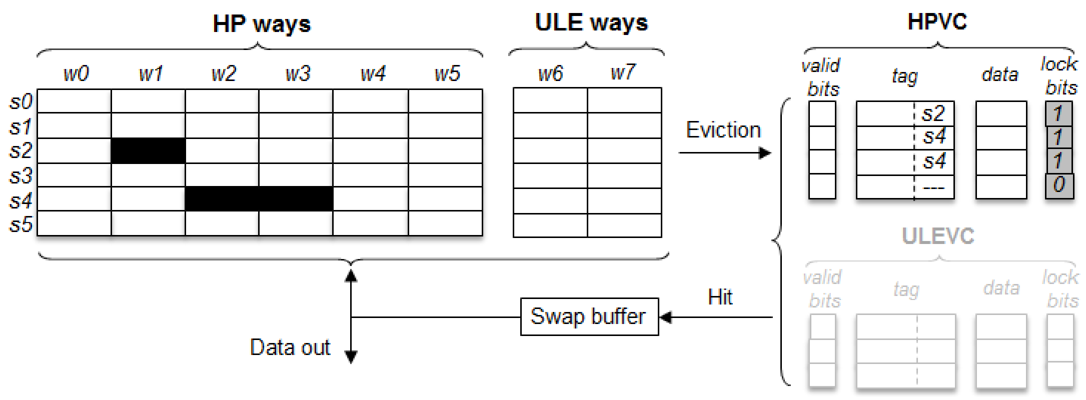
Note that such data movement can be afforded because it happens in parallel with retrieving data from memory and its delay (in the order of 2 cycles) is largely below the latency needed for data from the main memory to reach the L1 cache.

At ULE mode, HP ways and the HPVC are disabled (see Figure 7.1(b)). Following the same philosophy described for HP mode, ULE ways implemented with $10T_W$ cells are enhanced with the Ultra-Low Energy Victim Cache (ULEVC) structure, analogously to the HPVC at HP mode.

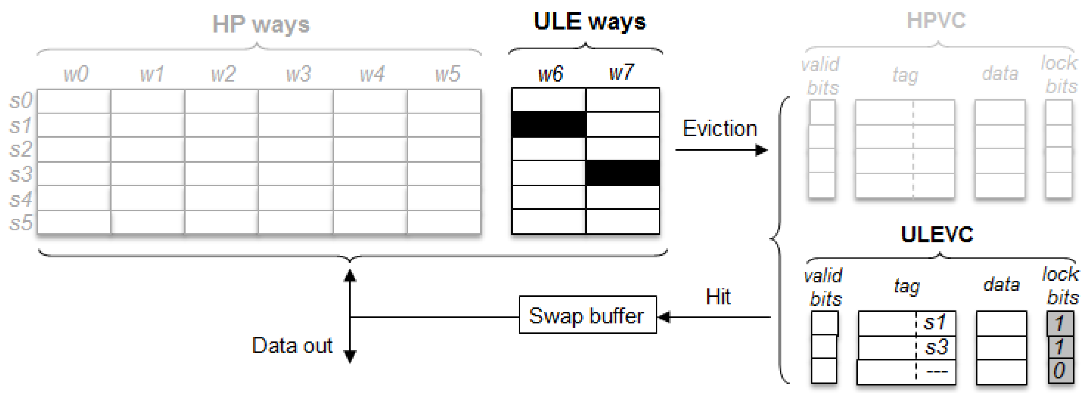
7.2.2.2 Performance Impact

Impact of the victim cache on *average* performance is small because cache operation ensures that the LRU lines reside in the victim cache, which typically experience few hits. Regarding weak cells, HSPICE simulations show that the access time variation between strong and weak cells is less than 5% at high voltage, so the impact of those weak cells in the overall cache latency at HP mode is either null or can be easily accommodated shaving it from other cache components (at the expense of some extra power in those components). However, this impact is larger at ULE mode. There is a trade-off between increasing cache latency, which has limited impact in performance, or making weak cells not so weak, thus sacrificing part of the energy savings.

In terms of guaranteed (WCET) performance, our cache architecture provides exactly the same number of available fault-free cache lines per set as in the baseline, thus guaranteeing the same WCET performance (same hits and same misses). Therefore, WCET estimation is not more complex than for the baseline architecture. Note that hit latencies are increased since we must assume always the case were the cache line having the hit resides in the victim cache and swapping is needed. However, the impact in WCET



(a) HP mode.



(b) ULE mode.

Figure 7.1: Example of the APPLE cache. Black fields stand for faulty cache lines. Grey fields stand for the extra information required by the HPVC (ULEVC) for performance guarantees. Pale blocks are disabled ones.

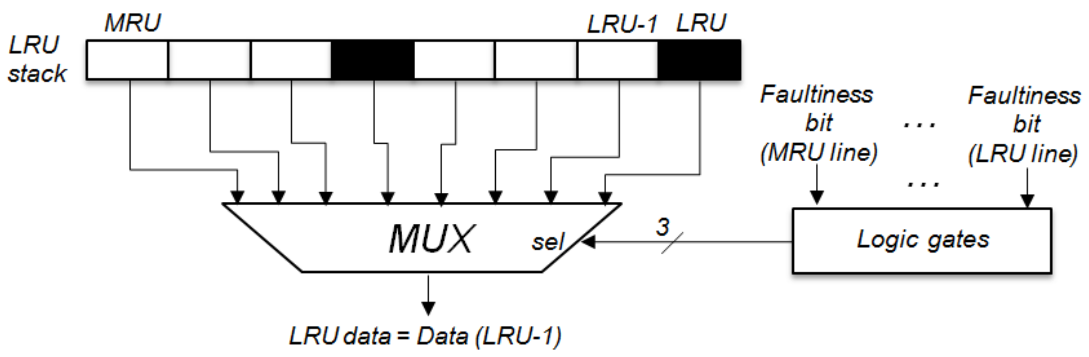


Figure 7.2: Modified LRU stack. Black fields stand for faulty cache lines.

```
1. Calculate the yield of the HP ways to be replaced ($Y_{strong}$)
2. Set minimal tranzistor sizes possible (i.e. $3\lambda$ width) for $6T_W$
   cell for targeted technology node (our is 32nm)
3. Calculate $6T_W$ cell's bit failure probability $P_{f6T_W}$ by using
   Chen's analysis [22] assuming $6\sigma$ random variations
   in $V_{th}$ for target Vcc considering read, write and hold failures
4. Calculate the yield of the chosen section enhanced with HPVC
   ($Y_{weak}$)
5. If ($Y_{weak}$<$Y_{strong}$)
5a.     Increase transistor sizes (widths) by $0.5\lambda$
5b.     Go to step 3
6. Else
6a.     Optimal cell size is obtained
```

Figure 7.3: Replacement algorithm.

estimates of higher hit latencies is largely below that of allowing faulty lines and thus, considering some accesses as misses instead of hits.

Turning off HP ways and HPVC at ULE mode (and ULEVC at HP mode) is done by using the gated-Vdd technique [71]. The processor itself is responsible for gating or ungating the corresponding cache block and *writting back dirty lines* on a Vcc change.

Our proposed design is not limited to any particular Vcc level, SRAM cell type or technology node because SRAM cells exhibit the same tradeoff between cell size and failure probability.

7.2.2.3 Alternative Configurations

For the sake of clarity, we refer to the considered configuration as: $a + b + c + d$, where a and b represent the number of *strong* and *weak* HP ways respectively, and c and d the number of *strong* and *weak* ULE ways respectively. Therefore, our baseline is 6+0+2+0 whereas the configuration which has been used for illustration of the APPLE design is 0+6+0+2.

Alternatively, a smaller number of the cache ways may be weakened. For example, instead of weakening all HP and ULE ways (0+6+0+2 in Figure 7.1), only six ways (e.g., ways w2-w7) can be replaced (2+4+0+2 configuration).

Table 7.1: Bit failure probabilities of memory cells used in the baseline (grey-colored) and APPLE configurations targeting 99.9% yield for an 8KB, 8-way, 32B/line cache with a 6-entry HPVC and 3-entry ULEVC.

	Bit failure probability					
Configuration	$6T_S$	$6T_W$	$10T_S$		$10T_W$	
			1V	0.35V	1V	0.35V
6+0+2+0	2.7x10^{-9}	–	1.2x10^{-12}	2.7x10^{-9}	–	–
0+6+0+2	–	3.5x10^{-5}	–	–	2.3x10^{-11}	4.6x10^{-6}
2+4+0+2	2.7x10^{-9}	9.1x10^{-5}	–	–	2.3x10^{-11}	4.6x10^{-6}

7.2.2.4 Implementation Details

Recall that both HP and ULE ways are enabled at HP mode whereas only ULE ways are active at ULE mode. Each SRAM cell is sized by using the analysis based on importance sampling proposed by Chen et al. [22] assuming 6σ random variations in V_{TH} for both high and NST Vcc considering read, write and hold failures in 32nm technology node.

We first describe the cell sizing in the baseline. For the chosen NST Vcc (e.g., 0.35V) and reduced frequency at ULE mode, we size the $10T_S$ cells to provide a 99.9% cache yield. Then, for the chosen high Vcc (e.g., 1V) and increased frequency at HP mode, the $6T_S$ cells are sized to match the same bit failure rate (P_{f6T_S}) as $10T_S$ cells at ULE mode. The yield of *strong* HP ways chosen to be **weakened** is:

$$Y_{strong} = (1 - P_{f6T_S})^{N_{6T_S}} \tag{7.1}$$

where N_{6T_S} is number of $6T_S$ cells in the HP ways chosen.

Next, we determine the size of $6T_W$ cells in order to replace $6T_S$ cells in HP ways chosen to be weakened. The algorithm is presented in Figure 7.3. We first set the minimal transistors sizes possible for $6T_W$ cells (3λ width for all transistors) and then obtain the bit failure probability (P_{f6T_W}) by using Chen's analysis [22]. The yield of *weak* HP ways is then:

$$Y_{weak} = \sum_{i=0}^{N_{hpvc}} \left[\binom{N_{cl} + N_{hpvc}}{i} P_l^{N_{cl}+N_{hpvc}-i} (1 - P_l)^i \right] \tag{7.2}$$

where N_{cl} is the number of cache lines in HP ways, N_{hpvc} is the number of entries in HPVC and P_l is the probability of a cache line to be fault-free. P_l can be calculated as:

Table 7.2: Relative (to $6T_S$) memory cell area used in the baseline (grey-colored) and APPLE configurations targeting 99.9% yield for an 8KB, 8-way, 32B/line cache with a 6-entry HPVC and 3-entry ULEVC.

	Relative SRAM cell area					
Configuration	$6T_S$	$6T_W$	$10T_S$		$10T_W$	
			1V	0.35V	1V	0.35V
6+0+2+0	1	–	1.92		–	
0+6+0+2	–	0.68	–		1.78	
2+4+0+2	–	0.62	–		1.78	

$$P_l = (1 - P_{f6T_W})^{N_{bitsinline}} \tag{7.3}$$

where $N_{bitsinline}$ is the number of bits in the cache line. Note that we consider the faults in both L1 cache and HPVC itself. We use 6-entries HPVC although our design is not limited to this particular case. N_{hpvc} sensitivity is studied in the evaluation section.

If the obtained yield is lower than required, i.e. Y_{strong}, transistors sizes (widths) are increased by the step value equal to 0.5λ and yield is calculated again. Once yield is high enough, we have an optimal cell size.

The same procedure is used for replacing $10T_S$ cells by $10T_W$ cells in ULE ways when operate at ULE mode, assuming a 3-entry ULEVC. Table 7.1 and Table 7.2 present the obtained bit failure probabilities and relative SRAM cell sizes respectively for all memory cells used at HP mode (e.g., 1V) and ULE mode (e.g., 0.35V).

7.3 Evaluation

In this section, we present and discuss the results in terms of performance, energy and area for proposed APPLE designs as well as for those used for comparison purposes. All experiments have been performed following the evaluation methodology presented in chapter 3 where HP mode is implemented with HPT.

In order to provide statistically meaningful results, we have generated 164 different cache samples for each APPLE cache configuration for a target cache yield. Using 164 different cache samples means that the confidence of our results is 90% with a confidence interval of 10% [66]. The faultiness for each bit in each cache sample is computed ran-

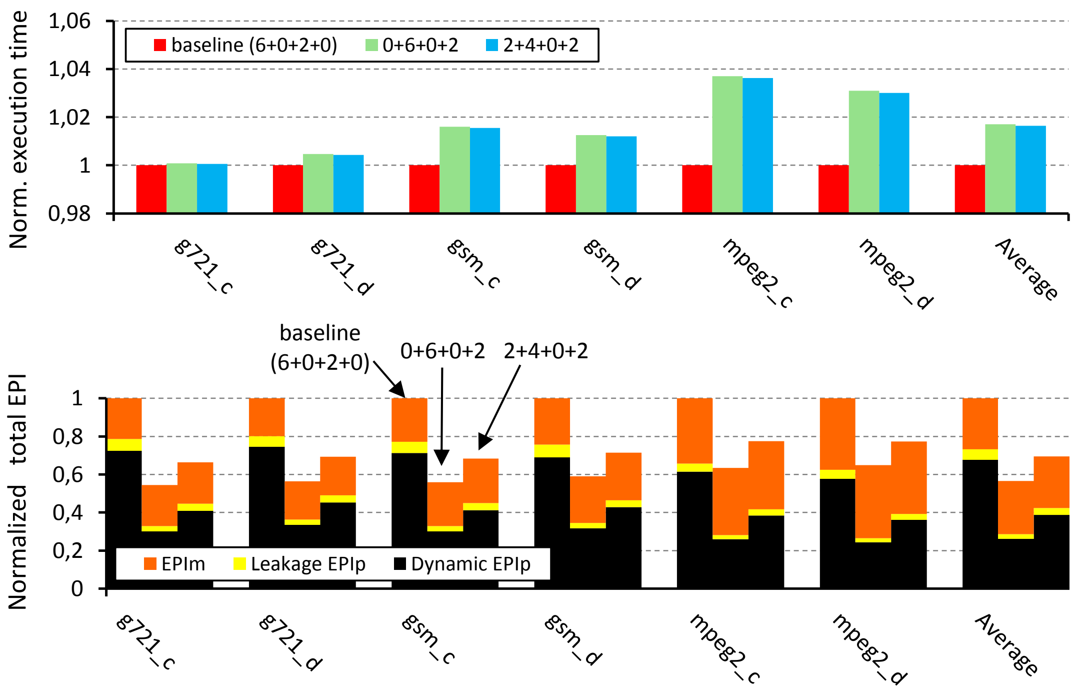


Figure 7.4: Normalized execution time and total EPI breakdown at HP mode.

domly and independently of other bits, matching a given faulty bit rate. Therefore, we generate 164 different processors with a similar number of faults and different faulty bit distribution in DL1 and IL1[2]. We have run each benchmark for each one of the 164 processor instances and present arithmetic mean (average) performance and energy results per benchmark.

Additional energy and area consumption of the HPVC and ULEVC are measured with CACTI and these values are included in our results. We account in our simulations for the extra read/write energy introduced by swapping as well as additional latency of 2 cache accesses to perform swaps correctly (see Section 7.2).

7.3.1 Results

In this subsection, we present overall execution time (ET) and total energy per instruction (EPI) for the whole processor at HP and ULE modes for 0+6+0+2 and 2+4+0+2 APPLE cache configurations. Along with this, cache area results are reported. For the sake of clarity, results have been normalized with respect to the baseline cache, given that

[2]Generating a number of cache setups with different fault mappings is important for the proposed cache in terms of average performance since swap is only needed for those sets with at least one faulty cache line.

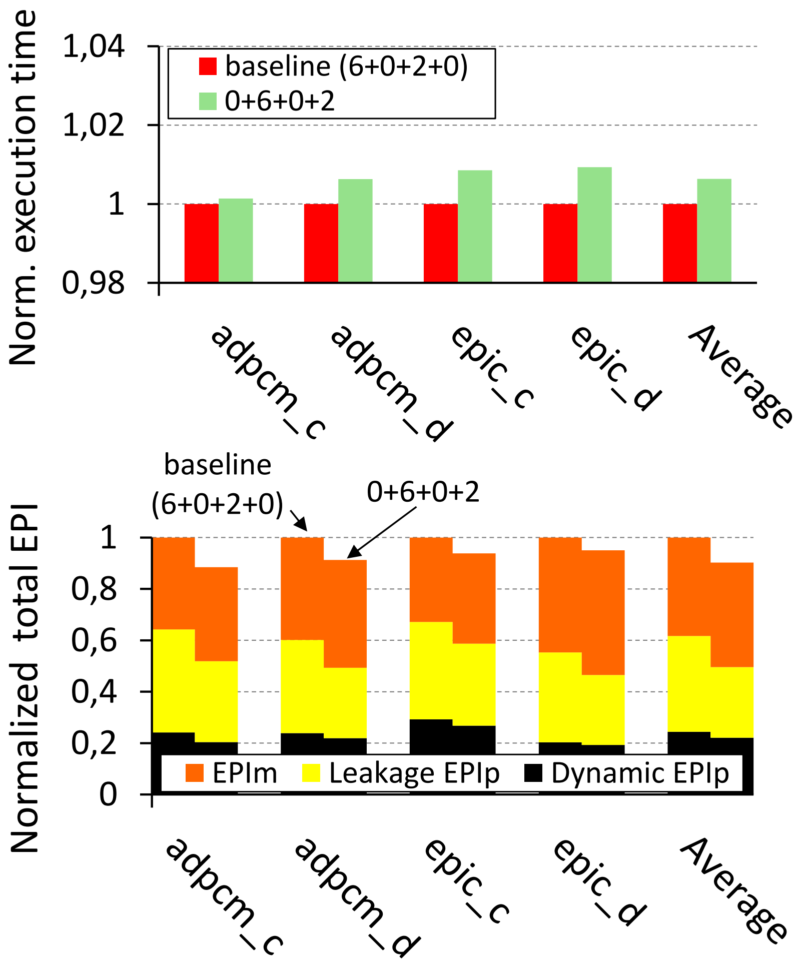


Figure 7.5: Normalized execution time and total EPI breakdown at ULE mode.

the execution time and energy vary noticeably across benchmarks. In order to provide more insights, total EPI is broken down into the following categories: EPI in off-chip main memory (EPI_m), leakage EPI in processor ($LeakageEPI_p$) and dynamic EPI in processor ($DynamicEPI_p$). L1 cache yield is 99.9%.

7.3.1.1 Performance and Energy

- **HP mode.** Figure 7.4 shows the normalized ET and total EPI across all benchmarks at HP mode. Since caches are the main energy consumer in our extremely simple processors, cache behavior dominates full processor behavior. Results show that the variation in ET is 1.7% on average. This is because some benchmarks have quite homogenous hit distribution across the cache lines (gsm_c, gsm_d, mpeg2_c and mpeg2_d) thus, hit HPVC frequently thus swapping lines.

In terms of EPI, results show that the proposed designs achieve a reduction of 42% and 30% on average for the 0+6+0+2 and 2+4+0+2 configurations respectively. Such reductions come mainly due to smaller transistor sizes needed for $6T_W$ and $10T_W$ cells, thus keeping node capacitances lower and reducing $DynamicEPI_p$. $LeakageEPI_p$ is also reduced, but its relative impact on total EPI is small given that dynamic energy is the dominant energy factor at high voltage. EPI_m remains constant because guaranteed performance of considered configurations is the same as in the baseline.

- **ULE mode.** HP ways and HPVC are turned off at this mode, whereas ULEVC is turned on to operate together with ULE ways (see Figure 7.1(b)). Results for 2+4+0+2 configurations are omitted because they are identical to those for 0+6+0+2 at ULE mode.

  Figure 7.5 shows the normalized ET and total EPI breakdowns across all benchmarks at ULE mode. Our design exhibits small variation in ET (0.6% on average) basically due to swapping when hits occur in the ULEVC. In terms of EPI, results show a 12% reduction. Both $LeakageEPI_p$ and $DynamicEPI_p$ are reduced due to reduced cache area. Conversely to HP mode, $LeakageEPI_p$ savings are significant because leakage energy becomes a dominant energy component at NST Vcc. EPI_m remains constant because our design has the same guaranteed performance as the baseline.

  EPI per hit in HPVC/ULEVC (extra read/write energy for swapping also accounted) is higher for benchmarks which exhibit homogenous hit distribution across the cache lines (i.e. gsm_c, gsm_d, mpeg2_c and mpeg2_d at HP mode and epic_c, epic_d at ULE mode). Nevertheless, such EPI is less than 5% of total EPI at both modes.

7.3.1.2 Area

Figure 7.6 presents normalized cache area breakdown for all configurations considered relative to the baseline. Proposed APPLE designs achieve area savings of 23% and 17% for 0+6+0+2 and 2+4+0+2 configurations respectively. HPVC and ULEVC area is also included in our results.

7.3.2 N_{hpvc} and N_{ulevc} Sensitivity Study

To better understand the effects of the number of HPVC and ULEVC entries, we vary N_{hpvc} (4, 6, 8 and 16) at HP mode. Similarly, we also vary N_{ulevc} (2, 3, 6 and 10) at ULE mode. Table 7.3 presents performance and energy results averaged across all applications at HP and ULE mode for the 0+6+0+2 configuration for all (N_{hpvc},N_{ulevc}) combinations. All results are normalized with respect to the default (6,3) case (highlighted in Table 7.3).

From the performance perspective, larger N_{hpvc} and N_{ulevc} values (e.g., 16 and 10 respectively) are not preferable because they increase the number of hits in the HPVC/ULEVC and thus, produce more swaps. From the total EPI perspective, larger N_{hpvc} and N_{ulevc} values increase the fault tolerance of the cache and enable more aggressive SRAM cell downsizing. However, for very high values (N_{hpvc} = 16 and N_{ulevc} = 10), swap overheads outweights the energy benefits achieved by cell downsizing. Conversely, lower values (i.e. N_{hpvc} = 4 and N_{ulevc} = 2) offer low fault tolerance and excessively conservative SRAM cell downsizing and thus, higher total EPI. Similar trends are observed for 2+4+0+2 configuration, but results are not reported since they do not provide further insights.

7.3.3 Comparison with Existing Approaches

Next, we present a detailed comparison between our proposed APPLE caches and existing cache designs with deterministic behavior [32, 60, 64, 94]. In order to perform an accurate comparison, we have implemented those designs for both DL1 and IL1 caches. Caches have been designed to have the same yield at ULE mode (i.e. 99.9%).

We have implemented designs proposed by Zhou et al. [94], where all 8 cache ways are always enabled at both modes and protected with Single Error Correction Double Error Detection (SECDED) codes [20] at cache line granularity. We have accounted en-

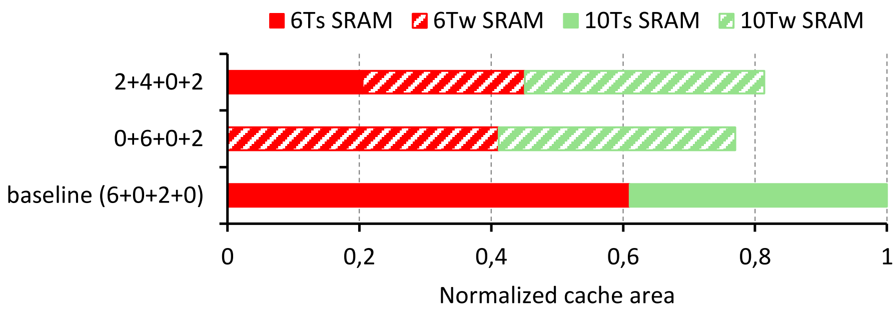


Figure 7.6: Normalized cache area breakdown.

Table 7.3: Normalized average ET and total EPI when varying the number of HPVC/ULEVC entries for 0+6+0+2 configuration with a 99.9% cache yield.

		(N_{hpvc}, N_{ulevc})			
HP mode		(4,3)	**(6,3)**	(8,3)	(16,3)
	Execution Time	0.988	**1**	1.033	1.074
	Total EPI	1.027	**1**	0.937	1.068
		N_{ulevc}			
ULE mode		2	**3**	6	10
	Execution Time	0.993	**1**	1.013	1.061
	Total EPI	1.018	**1**	0.951	1.049

ergy and area overheads introduced by SECDED check bits in our simulations as well as additional latency of one clock cycle for SECDED encoding/decoding, but we have not accounted the energy consumed by encoding/decoding circuits. The size of 6T cells is calculated according to the methodology proposed in [94] to match the target cache yield at ULE mode.

Regarding the designs proposed by Ghasemi et al. [32], they are in spirit similar to our baseline. The only difference is that larger 6T cells are used in ULE ways instead of the 10T cells. Therefore, at HP mode all cache ways are enabled whereas at ULE mode only 2 cache ways implemented with large 6T cells keep operating. The size of large 6T cells is calculated analogously to the size of the 10T cells in the baseline cache, as explained in Section 7.2. According to our simulations, the access time variations at 0.35V for SRAM arrays implemented with those large 6T cells is around 40%, relative to SRAM arrays implemented with $10T_W$ cells, so we assume a 3-cycle cache latency instead of the regular 2-cycle latency at ULE mode. Access time variations at 1V are negligible, so a 2-cycle latency is assumed.

Table 7.4 presents ET, total EPI and cache area results. ET and total EPI results are averaged across all applications at HP and ULE mode. All results are normalized with respect to the proposed 0+6+0+2 APPLE configuration (grey-colored).

Results show that the design in [94] exhibits an ET degradation of 8% and 13% at HP mode and ULE mode respectively due to the extra cycle for SECDED encoding/decoding. On the other hand, APPLE cache has larger ET than the design in [32] at HP mode due to swapping overheads (around 1.7%). However, the design in [32] exhibits significantly larger ET (around 49%) at ULE mode due to the additional cycle for each cache access.

Table 7.4: Normalized average ET, total EPI and cache area when comparing existing deterministic caches with APPLE designs targeting a 99.9% cache yield.

	Norm. ET		Norm. total EPI		Norm.
	HP mode	ULE mode	HP mode	ULE mode	cache area
APPLE (0+6+0+2)	1	1	1	1	1
Baseline [60]	0.984	0.993	1.72	1.14	1.29
Zhou et al. [94]	1.08	1.13	2.07	4.17	2.27
Ghasemi et al. [32]	0.983	1.49	1.88	2.91	1.96
Maric et al. [64]	0.984	1.023	1.479	0.684	1.248

The main drawback of designs in [94] and [32] is their significant area overhead with respect to our proposed design (up to 127%). This directly translates into higher energy consumption at both modes. In other words, considered designs are overdesigned to operate reliably at ULE mode.

Comparison with designs proposed in [60] is already done in section 7.3.1. Moreover, APPLE caches can be compared with our EDC-based solution [64] proposed in the chapter 6. Since those designs are already compared with the baseline [60], this comparison provides obvious insights.

7.3.4 Overall Energy Savings

In this section, we report overall energy savings for the whole lifetime of the artificial application which we described in section 3.2.6. Figure 7.7 shows total energy breakdown for the whole application lifetime for the baseline 6T+10T and proposed APPLE cache (0+6+0+2 configuration) designs. Basically, the main conclusions, trends and findings are similar to those discussed in section 4.3.2.6, so here we omit further discussion.

7.4 Summary

In this chapter, we propose efficient, but simple Adaptive Performance-Predictable Low-Energy (APPLE) single-Vcc domain L1 cache designs for reliable hybrid voltage operation, which meet *all* specific and stringent needs of battery-powered ultra-low-cost (e.g., below 1 USD) systems. The APPLE cache design relies on replacing large memory cells by more energy-efficient and smaller cells enhanced with extra cache lines set up in cache-assist structure, i.e., an adapted victim cache, to allow extra associativity for some cache

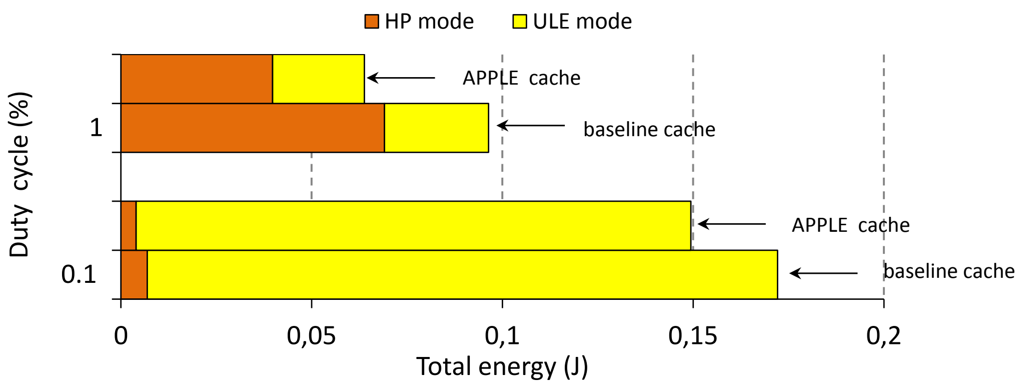


Figure 7.7: Total energy breakdown for the whole application lifetime for the baseline 6T+10T and proposed APPLE cache (0+6+0+2 configuration) designs showing energy contribution of each operation mode for different duty cycle values.

sets that may need it due to disabled faulty cache lines. Experimental results show that APPLE caches achieve significant average energy and area reductions (up to 42% for energy and 23% for area) with a negligible *average* performance impact (less than 1.7% on average) with respect to existing solutions without jeopardizing reliability levels to still provide strong performance guarantees. APPLE cache is shown to provide exactly the same number of available fault-free cache lines per set as in the baseline, thus guaranteeing the same WCET performance (same hits and same misses). Therefore, WCET estimation is not more complex than for the baseline fault-free cache architecture.

Chapter 8

Conclusions and Future Work

This chapter summarizes the main contributions of this thesis and presents an analysis of the results shown in each chapter. This chapter also presents future lines of work opened up by this thesis.

8.1 Goals, Contributions and Main Conclusions

The increasing demand for tiny sensor-based battery-powered ultra-low-cost systems (e.g., below 1 USD) in emerging applications such as body, urban life and environment monitoring, etc., has introduced many challenges in the chip design. Those systems require high performance occasionally and very little energy consumption during most of the time needed for longer battery lifetime. Moreover, they require real-time guarantees needed for WCET estimation required by critical applications running on top. In particular, the main system requirements can be summarized as follows:

- *Reliable hybrid high and ultra-low voltage operation.* It is shown to be a road to follow for energy efficiency, but emerges as one of the most prominent design challenges. This fact is particularly true for SRAM memories due to inefficiency of high-voltage SRAM cells to operate reliably at ultra-low voltages and vice versa. As we have seen in this thesis, efficient SRAM cell designs across the different voltage levels simultaneously do not exist. Combining different SRAM cells into the same blocks such as L1 cache memories is the simplest technological solution to enable reliable hybrid voltage operation. However, this further introduces many issues from a microarchitectural perspective which we addressed in this thesis.

- *Design simplicity.* Caches must be designed with low complexity in order to provide increased yield, higher integration and reduced costs, given that we target the ultra-low-cost (e.g., below 1 USD) market.

- *Guaranteed performamce and reliability.* Besides functional correctness, strong timing guarantees are required for WCET estimation. Albeit caches have hard-to-predict behavior, increased performance is a must for real-time systems, so caches have been widely considered in those systems [27].

The challenges to fulfill all stringent requirements of our target market require ground-breaking solutions to obtain energy-efficient and reliable processors for the next 10 or 20 years. Thus, it is crucial to start research activities in this arena to bring out innovative and unorthodox solutions to this embedded market. This thesis faces the main challenge of those hybrid designs, which is how to combine both design styles into a single core avoiding full replication of components in order to reuse components as much as possible while keeping performance levels as well as power and energy efficiency. Thus, chips are cheaper as higher integration is achieved.

The main contributions of this thesis are the following.

- We have analyzed the performance/power trade-offs involved in the design of SRAM L1 caches for reliable hybrid high and NST Vcc operation from a microarchitectural perspective. By doing so, we have understood the key insights needed for developing accurate cache power models. Then, we have explored the design space, thus identifing interesting design points.

- We have proposed the Hybrid Cache Ways [60] architecture, which splits the cache into two sections:

  - *HP ways*: Some cache ways are made of the SRAM cells optimized for one particular Vcc level (e.g., high or moderate Vcc, so HPT or LPT technology),
  - *ULE ways*: The rest of the cache ways are made of the SRAM cells optimized for another Vcc level (e.g., NST Vcc, so LLT technology).

  We provide a comprehensive study varying several critical parameters such as SRAM cell type, operation voltage, cache size, associativity and line size. Our designs were validated for a set of data-intensive benchmarks running on the full-chip microarchitecutral simulator [4]. We have shown that our designs are able to operate reliably across a wide range of voltages, consuming little energy at ULE mode as well as providing high performance at HP mode. Part of our results are summarized in the article published in ACM CF 2011 conference [60] and a scientific journal article that is currently under submission.

- Data management policies are required to access HP and ULE ways at HP mode to minimize the number of energy-expensive accesses to ULE ways. Unfortunately, existing policies are far from being efficient across all applications. Therefore, we have proposed an efficient, but simple Adaptive DAta Management (ADAM) mechanism for HP operation on single-Vcc domain caches for hybrid operation. ADAM is tailored to detect hit distribution dynamically across the cache regions (HP ways and ULE ways regions) during program execution and adapts to different application behaviors to optimize performance and energy consumption by means of an extremely simple hardware mechanism. We have shown that ADAM combines the advantages of the previous approaches (swap and sequential access policies), and moreover, reacts in front of the different phases of a program [61].

  ADAM has been validated for a set of data-intensive benchmarks running on the full-chip microarchitectural simulator. The results show that ADAM achieves great energy savings with respect to all state-of-the-art approaches at HP mode with negligible performance impact. This work is summarized in the scientific article that we have published in the ACM/IEEE GLSVLSI 2012 conference [61].

- Existing caches use large memory cells (e.g., 10T) to achieve high levels of reliability even at ULE mode, as needed by critical applications run on top. Decreasing the size of the memory cells for higher energy efficiency at the expense of higher failure rates is unacceptable in this environment. Faulty entries should be then disabled and strong performance guarantees required by critical applications would not be achievable. Thus, we have proposed new cache architectures which rely on replacing energy-hungry bitcells (e.g., 10T) by more energy-efficient and smaller cells (e.g., 8T) enhanced with EDC features to improve energy and area efficiency without jeopardizing reliability levels to still provide predictable performance, as needed for critical applications.

  We validated the proposed designs for a set of data-intensive benchmarks running on the full-chip microarchitecutral simulator. The results show that our caches are optimal in all energy/performance/area metrics at HP end ULE mode. Our results are summarized in the article published in ACM/IEEE DATE 2013 conference [64] as well as in the scientific journal article published in IEEE TVLSI 2013 [62].

- Finally, we have proposed Adaptive Performance-Predictable Low-Energy (APPLE) single-Vcc domain L1 cache designs for reliable hybrid voltage operation, which meet all stringent needs of our target market. APPLE caches rely on replacing large memory cells by more energy-efficient and smaller cells enhanced with extra cache lines set up in a cache-assist structure, i.e., an adapted victim cache, to allow extra associativity for some cache sets that may need it due to disabled faulty cache lines. Experimental results show that APPLE caches achieve significant average energy and area reductions with a negligible average performance impact with respect to existing solutions without jeopardizing reliability levels to still provide strong performance guarantees. We have published this work in ACM/IEEE DAC 2013 conference [63].

8.2 Future Work

The different techniques presented in this dissertation can be further enhanced or extended. Our experiments show that those trends are consistent across different proposals, and open the door to further research in the design of hybrid microarchitectures. Moreover, this thesis opens up several new topics which we want to explore further.

Beside L1 caches, the rest of resources in our hybrid processor are mainly devoted to other SRAM array-like structures such as register files, BTB, TLBs, etc. All SRAM arrays, except L1 caches, have been implemented using 10T SRAM cells, so they operate properly at any voltage level considered. The next step will be devising more efficient designs for those components so that their energy consumption can be further reduced. This is particularly true for the register files, which are the next critical component after L1 caches in our simple single-core processor. Solutions that we plan to bring out will consist of new energy-efficient fault-tolerant designs, based on either replacing or disabling faulty storage, which provide predictable performance. We hope to find efficient solutions for those other processor components in the near future.

While this thesis is based on conventional CMOS technology, the advent of new technologies incorporating new materials opens the door to new trade-offs in cache design. Although our methods are flexible enough to allow the use of those technologies instead of the particular ones used (6T-8T-10T cells, 32nm technology node, Vcc, etc.), we believe new technologies should be studied in the context of our target market in the future.

8.3 Publications

Below we list the publications which our research to date has produced. First, we list accepted publications related to this thesis. Then, we enumerate submitted publications for which we are still waiting for an answer.

8.3.1 Accepted Publications

- B. Maric, J. Abella, F. J. Cazorla and M. Valero. Hybrid High-Performance Low-power and Ultra-Low Energy Reliable Caches. ACM CF 2011. *Proc. of the 8th ACM International Conference on Computing Frontiers. Ischia, Italy, May 2011.*

- B. Maric, J. Abella and M. Valero. ADAM: An Efficient Data Management Mechanism for Hybrid High and Ultra-Low Voltage Operation Caches. ACM/IEEE GLSVLSI 2012. *Proc. of the 22nd ACM/IEEE Great Lakes Symposium on VLSI. pp. 245-245. Salt Lake City, Utah, USA. May 2012.*

- B. Maric, J. Abella and M. Valero. Efficient Cache Architectures for Reliable Hybrid Voltage Operation Using EDC Codes. ACM/IEEE DATE 2013. *Proc. of the 16th ACM/IEEE Design, Automation, and Test in Europe conference. pp. 917-920. Grenoble, France. March 2013.*

- B. Maric, J. Abella and M. Valero. APPLE: Adaptive Performance-Predictable Low-Energy Caches for Reliable Hybrid Voltage Operation. ACM/IEEE DAC 2013. *Proc. of the 50th ACM/IEEE Design Automation Conference. Austin, Texas, USA. June 2013.*

- B. Maric, J. Abella and M. Valero. Analyzing the Efficiency of L1 Caches for Reliable Hybrid-Voltage Operation Using EDC Codes. *To appear in IEEE Transactions on Very Large Scale Integration (VLSI) Systems (IEEE TVLSI) 2013.*

8.3.2 Submitted Articles for Publication

- B. Maric, J. Abella, F. J. Cazorla and M. Valero. Hybrid Cache Designs for High and Ultra-Low Voltage Operation. *Submitted to ACM Transactions on Design Automation of Electronic Systems (ACM TODAES) 2013.*

Appendices

Appendix A

Robustness Calculation Using Importance Sampling

In this chapter, we provide more details about SRAM cell robustness calculation using importance sampling proposed by Chen et al. [22].

For a complete look at SRAM, reliability sampling methods like Monte Carlo are necessary. In Monte Carlo sampling, the number of passing bit cells is divided by the total number of iterations (n) to find the expected yield, as shown in (A.1):

$$Y = \frac{1}{n}\sum_{n} f(x), \quad where f(n) = \begin{cases} 1, & \text{if pass} \\ 0, & \text{if fail} \end{cases} \tag{A.1}$$

.

Process parameters such as V_{TH} and gate length are selected from a probability density function (PDF), which represents the natural variation in the process parameter. The PDF of in SRAM devices is modeled as a normal distribution. Since caches contain many bitcells, the failure rate of each one must be very low in order to have high yield for the cache. For example, to have a 99% yield for a small 8KB SRAM, the bit cell failure rate must be 1.53x10^{-7}. To calculate this bit cell yield using Monte Carlo, at least 10 million simulations must be performed, making this procedure computationally intensive. For larger caches, the required bit cell failure rate is even lower and complete Monte Carlo analysis is almost infeasible.

For our study, we choose importance sampling as an efficient and accurate way of calculating the SRAM robustness. As shown in (A.2) - (A.5):

$$Y = \frac{1}{n}\sum_{g(x)} \frac{p(x)}{g(x)} \tag{A.2}$$

$$p(x) = \prod \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{(x-\mu)^2}{2\sigma^2} f(x) \tag{A.3}$$

$$g(x) = \prod \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{(x-\mu+6\sigma)^2}{2\sigma^2} \tag{A.4}$$

$$Y = \frac{1}{n} \sum_{g(x)} \frac{\prod \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{(x-\mu)^2}{2\sigma^2} f(x)}{\prod \frac{1}{\sigma\sqrt{2\pi}} \exp \frac{(x-\mu+6\sigma)^2}{2\sigma^2}} \tag{A.5}$$

the importance sampling technique chooses a new sampling PDF (SPDF) for each transistor, so that more failures are simulated. The V_{TH} of each transistor is shifted by the value sampled from the PDF plus 6σ to introduce enough mismatch into the bit cell to increase the probability of failure. Since the natural occurrence of these highly skewed devices is rare, the importance samples are then weighted by the ratio of the probability of the large shift in each transistor to the probability that these shifts were sampled. These weighted values are then used to calculate the bit cell yield (Y). This method allows us to accurately measure the region of interest where SRAM can fail with greatly reduced computational complexity. We determine that 20 000 samples are sufficient for accurate results. To measure the failure rates in our study with Monte Carlo, at least 10^12 samples are needed, making importance sampling 50 million times faster.

The standard deviation of the V_{TH} fluctuation ($\sigma_{V_{TH}}$) due to random dopant fluctuation depends on the manufacturing process, doping profile and the transistor sizing [79]. The dependence of $\sigma_{V_{TH}}$ on the transistor size is given by [79] as:

$$\sigma_{V_{TH}} = \sigma_{V_{TH0}} \sqrt{\frac{L_{min}}{L} \frac{W_{min}}{W}} \tag{A.6}$$

where $\sigma_{V_{TH0}}$ is the standard deviation of the V_{TH} for a minimum-sized transistor with sizes of the W_{min} and L_{min}.

Bibliography

[1] J. Abella and A. Gonzalez. Power efficient data cache design. In *Proc. of IEEE International Conference on Computer Design (ICCD)*, pages 8–13, 2003.

[2] J. Abella et al. Low vccmin fault-tolerant cache with highly predictable performance. In *Proc. of the 42nd annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 111–121, 2009.

[3] J. Abella et al. Rvc-based time-predictable faulty caches for safety-critical systems. In *Proc. of IEEE International On-Line Testing Symposium (IOLTS)*, 2011.

[4] C. Acosta, F. J. Cazorla, A. Ramirez, and M. Valero. The MPsim simulation tool. Technical Report UPC-DAC-RR-CAP-2009-15, in UPC, 2009.

[5] A. Agarwal, B. C. Paul, H. Mahmoodi, A. Datta, and K. Roy. A process-tolerant cache architecture for improved yield in nanoscale technologies. *IEEE Transactions on VLSI Systems (TVLSI)*, 13(1), 2005.

[6] D. H. Albonesi. Selective cache ways: On-demand cache resource allocation. In *Proc. of the 32nd annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 248–259, 1999.

[7] A. Ansari, S. Feng, S. Gupta, and S. Mahlke. Archipelago: A polymorphic cache design for enabling robust near-threshold operation. In *Proc. of the 17th International Symposium on High Performance Computer Architecture (HPCA)*, 2011.

[8] A. Ansari, S. Gupta, S. Feng, and S. Mahlke. Zerehcache: Armoring cache architectures in high defect density technologies. In *Proc. of the 42nd annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 100–110, 2009.

[9] ARM. Cortex-A5™ Technical Reference Manual, 2009. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0433b/index.html.

[10] ARM. Cortex-R4™ Technicalhite Reference Manual, 2010. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0363g/DDI0363G_cortex_r4_r1p4_trm.pdf.

[11] ARM. Cortex-M0+™ Technicalhite Reference Manual, 2012. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0484c/DDI0484C_cortex_m0p_r0p1_trm.pdf.

[12] R. Balasubramonian et al. Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures. In *Proc. of the 33rd annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2000.

[13] S. Borkar. Design challenges of technology scaling. *IEEE Micro*, 19(4), 1999.

[14] K. Bowman, S. Duvall, and J. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE Journal of Solid-State Circuits (JSSC)*, 37(2):183–190, 2002.

[15] D. M. Brooks, V. Tiwari, and M. Martonosi. Wattch: A framework for architectural-level power analysis and optimizations. In *Proc. of the 27th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pages 83–94, 2000.

[16] B. Calhoun and A. Chandrakasan. Characterizing and modeling minimum energy operation for subthreshold circuits. In *Proc. of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 90–95, 2004.

[17] B. Calhoun and A. Chandrakasan. A 256-kb 65-nm sub-threshold sram design for ultra-low-voltage operation. *IEEE Journal of Solid-State Circuits (JSSC)*, 42(3):680–688, 2007.

[18] B. H. Calhoun and A. P. Chandrakasan. Static noise margin variation for subthreshold sram in 65 nm cmos. In *IEEE Journal of Solid-State Circuits (JSSC)*, volume 41, pages 1673–1679, 2006.

[19] L. Chang et al. An 8t-sram for variability tolerance and low-voltage operation in high-performance caches. *IEEE Journal of Solid-State Circuits (JSSC)*, 43(4):956–963, 2008.

[20] C. L. Chen and M. Y. Hsiao. Error-correcting codes for semiconductor memory applications: A state-of-the-art review. *IBM Journal of Research and Development*, 28(2):123–134, 1984.

[21] J. Chen, L. Clark, and T.-H. Chen. An ultra-low-power memory with a subthreshold power supply voltage. *IEEE Journal of Solid-State Circuits (JSSC)*, 41(10):2344–2353, 2006.

[22] G. Chen et al. Yield-driven near-threshold SRAM design. In *Proc. of IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 660–666, 2007.

[23] Z. Chishti et al. Improving cache lifetime reliability at ultra-low voltages. In *Proc. of the 42nd annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 89–99, 2009.

[24] Y. G. Choi et al. Matching cache access behavior and bit error rate pattern for high performance low vcc l1 cache. In *Proc. of the 48th Design Automation Conference (DAC)*, pages 978–983, 2011.

[25] R. Dreslinski et al. Reconfigurable energy efficient near threshold cache architectures. In *Proc. of the 41st annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 459–470, 2008.

[26] S. Dropsho et al. Integrating adaptive on-chip storage structures for reduced dynamic power. In *Proc. of ACM/IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2002.

[27] C. Ferdinand and R. Wilhelm. Fast and efficient cache behavior prediction for real-time systems. *Real-Time System*, XVII:131–181, 1999.

[28] K. Flautner et al. Drowsy caches: Simple techniques for reducing leakage power. In *Proc. of the 29th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pages 148–157, 2002.

[29] S. Fujii and T. Sato. Non-uniform set-associative caches for power-aware embedded processors. In *Lecture Notes in Computer Science (LNCS): Embedded and Ubiquitous Computing*, pages 217–226, 2004.

[30] A. Garg and P. Dubey. Fuse area reduction based on quantitative yield analysis and effective chip cost. In *Proc. of IEEE Int. Symp. on Defect and Fault Tolerance in VLSI Systems*, pages 166–174, 2006.

[31] G. Gerosa et al. A Sub-2W low power IA processor for mobile internet devices in 45nm high-k metal gate CMOS. *IEEE Journal of Solid-State Circuits (JSSC)*, 44(1):73–82, 2009.

[32] H. Ghasemi, S. Draper, and N. Kim. Low-voltage on-chip cache architecture using heterogeneous cell sizes for high-performance processors. In *Proc. of the 17th IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 38–49, 2011.

[33] K. Ghose and M. B. Kamble. Reducing power in superscalar processor caches using subbanking, multiple line buffers and bit-line segmentation. In *Proc. of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 70–75, 1999.

[34] S. Gribal et al. The next convergence: High-performance and mission-critical markets. In *Proc. of the High Performance and Embedded Architecture and Compilation (HiPEAC)*, 2013.

[35] S. Hanson et al. Energy optimality and variability in subthreshold design. In *Proc. of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 363–365, 2006.

[36] D. Hardy, I. Sideris, N. Ladas, and Y. Sazeides. The performance vulnerability of architectural and non-architectural arrays to permanent faults. In *Proc. of the 45th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 48–59, 2012.

[37] A. Hasegawa, I. Kawasaki, K. Yamada, S. Yoshioka, S. Kawasaki, and P. Biswas. SH3: High code density, low power. *IEEE Micro*, 15(6), 1995.

[38] Y. He et al. Optimal resource allocation for pervasive health monitoring systems with body sensor networks. *IEEE Transactions on Mobile Computing (TMC)*, 10(11):1558–1575, 2011.

[39] J. Hezavei et al. A comparative study of power efficient sram designs. In *Proc. of the 10th Great Lakes Symposium on VLSI (GLSVLSI)*, 2000.

[40] S. Hsu et al. Tutorial on microprocessor memory array circuits for architects. In *Proc. of annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2007.

[41] K. Inoue, T. Ishihara, and K. Murakami. Way-predictive set-associative cache for high performance and low energy consumption. In *Proc. of the 1999 International Symposium on Low Power Electronics and Design (ISLPED)*, 1999.

[42] Intel Corporation. First the tick, now the tock: Next generation Intel® microarchitecture (Nehalem). *White Paper*, 2008. http://www.intel.com/technology/architecture-silicon/next-gen/whitepaper.pdf.

[43] Intel Corporation. Intel® quark soc x1000 , 2013. http://ark.intel.com/products/79084/Intel-Quark-SoC-X1000-16K-Cache-400-MHz.

[44] M. Ishida et al. c. In *Proc. of IEEE International Electron Devices Meeting (IEDM)*, pages 201–204, 1998.

[45] K. Itoh et al. Trends in low-power ram circuit technologies. *Proc. of the IEEE*, 83(4), 1995.

[46] S. Jain and P. Agarwal. A low leakage and SNM free SRAM cell design in deep sub micron CMOS technology. In *Proc. of the 19th IEEE International Conference on VLSI Design (VLSID)*, 2006.

[47] S. Jain et al. A 280mv-to-1.2v wide-operating-range ia-32 processor in 32nm cmos. In *Proc. of IEEE International Solid-State Circuits Conference (ISSCC), dig. Tech. Papers*, pages 66–68, 2012.

[48] R. Joshi et al. 6.6+ ghz low vmin, read and half select disturb-free 1.2 mb sram. In *Proc. of the IEEE Symposium on VLSI Circuits*, pages 250–251, 2007.

[49] N. P. Jouppi. Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers. In *Proc. of IEEE/ACM International Symposium on Computer Architecture (ISCA)*, 1990.

[50] D. Kanter. Amd Fusion Architecture and Llano. 2008. http://realworldtech.com/page.cfm?ArticleID=RWT062711124854.

[51] N. Kim et al. Single-vdd and single vt superdrowsy techniques for low-leakage high-performance instruction caches. In *Proc. of the 2004 International Symposium on Low Power Electronics and Design (ISLPED)*, 2004.

[52] T.-H. Kim et al. A high-density subthreshold sram with data-independent bitline leakage and virtual ground replica scheme. In *Proc. of IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers*, pages 330–332, 2007.

[53] J. Kin, M. Gupta, and W. Mangione-Smith. The filter cache: An energy efficient memory structure. In *Proc. of the 30th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 184–193, 1997.

[54] J. Kulkarni, K. Kim, and K. Roy. A 160 mV, fully differential, robust schmitt trigger based sub-threshold SRAM. In *Proc. of IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 171–176, 2007.

[55] T. Kuroda et al. Variable supply-voltage scheme for low-power high-speed cmos digital design. *IEEE Journal of Solid-State Circuits (JSSC)*, 33(3), 1998.

[56] F. Kusumoto and N. Goldschlager. *Cardiac Pacing for the Clinician*. Springer Science+Business Media LLC, 2008.

[57] N. Ladas, Y. Sazeides, and V. Desmet. Performance-effective operation below vcc-min. In *Proc. of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 223–234, 2010.

[58] C. Lee et al. Mediabench: A tool for evaluating and synthesizing multimedia and communication systems. In *Proc. of the 30th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 330–335, 1997.

[59] Z. Liu and V. Kursun. High read stability and low leakage cache memory cell. In *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2774–2777, 2007.

[60] B. Maric, J. Abella, F. J. Cazorla, and M. Valero. Hybrid high-performance low-power and ultra-low energy reliable caches. In *Proc. of the 8th ACM International Conference on Computing Frontiers (CF)*, pages 12:1–12:2, 2011.

[61] B. Maric, J. Abella, and M. Valero. Adam: An efficient data management mechanism for hybrid high and ultra-low voltage operation caches. In *Proc. of the 22nd ACM/IEEE Great Lakes Symposium on VLSI (GLSVLSI)*, pages 245–250, 2012.

[62] B. Maric, J. Abella, and M. Valero. Analyzing the efficiency of l1 caches for reliable hybrid-voltage operation using edc codes. In *To appear in IEEE Transactions on VLSI Systems (TVLSI)*, 2013.

[63] B. Maric, J. Abella, and M. Valero. Apple: Adaptive performance-predictable low-energy caches for reliable hybrid voltage operation. In *Proc. of the 50th ACM/IEEE Design Automation Conference (DAC)*, 2013.

[64] B. Maric, J. Abella, and M. Valero. Efficient cache architectures for reliable hybrid voltage operation using edc codes. In *Proc. of the 16th ACM/IEEE Design, Automation, and Test in Europe (DATE)*, 2013.

[65] S. Mitra et al. Combinational logic soft error correction. In *Workshop on System Effects of Logic Soft Errors (SELSE-2)*, 2006.

[66] D. Moore and G. McCabe. Introduction to the practice of statistics. In *W. H. Freeman and Co.*, 1989.

[67] Y. Morita et al. An area-conscious low-voltage-oriented 8t sram design under dvs environment. In *Proc. of the IEEE VLSI Circuits Symposium*, pages 256–257, 2007.

[68] N. Muralimanohar, R. Balasubramonian, and N. Jouppi. CACTI 6.0: A tool to understand large caches. *HP Tech Report HPL-2009-85*, 2009.

[69] S. Narendra et al. Full-chip sub-threshold leakage power prediction model for sub-0.18um cmos. In *Proc. of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, 2002.

[70] S. Ozdemir, D. Sinha, G. Memik, J. Adams, and H. Zhou. Yield-aware cache architectures. In *Proc. of the 39th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2006.

[71] M. Powell et al. Gated-vdd: A circuit technique to reduce leakage in deep-submicron cache memories. In *Proc. of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 90–95, 2000.

[72] M. Powell et al. Reducing set-associative cache energy via way-prediction and selective direct-mapping. In *Proc. of the 34th annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2001.

[73] D. Roberts, N. Kim, and T. Mudge. On-chip cache device scaling limits and effective fault repair techniques in future nanoscale technology. In *Proc. of the 10th IEEE Euromicro Conference on Digital System Design Architectures, Methods and Tools*, pages 570–578, 2007.

[74] A. Sasan, H. Homayoun, A. Eltawil, and F. Kurdahi. A fault tolerant cache architecture for sub 500mv operation: Resizable data composer cache (rdc-cache). In *Proc. of ACM Int. Conf. on Compilers, Architectures and Synthesis for Embedded Systems (CASES)*, pages 251–260, 2009.

[75] M. Seok et al. Cas-fest 2010: Mitigating variability in near-threshold computing. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 1(1):42–49, 2011.

[76] M. Slijepcevic et al. Dtm: Degraded test mode for fault-aware probabilistic timing analysis. In *Proc. of the Euromicro Conference on Real-Time Systems (ECRTS)*, 2013.

[77] C. L. Su and A. M. Despain. Cache design trade-offs for power and performance optimization: A case study. In *Proc. of ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED)*, pages 63–68, 1995.

[78] R. Szewczyk et al. Lessons from a sensor network expedition. In *Proc. of European Workshop on Sensor Networks*, pages 307–322, 2004.

[79] Y. Taur and T. Ning. *Fundamentals of Modern VLSI Devices*. Cambridge University Press, 1998.

[80] D. N. Truong et al. A 167-processor computational platform in 65 nm cmos. *IEEE Journal of Solid-State Circuits*, 4:1130–1144, 2009.

[81] J. Tschanz et al. Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage. *IEEE Journal of Solid-State Circuits (JSSC)*, 37(11):1396–1402, 2002.

[82] D. M. Tullsen, S. J. Eggers, and H. M. Levy. Simultaneous multithreading: Maximizing on-chip parallelism. In *Proc. of the 22nd IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pages 392–403, 1995.

[83] N. Verma and A. P. Chandrakasan. A 256 kb 65 nm 8t subtreshold sram employing sense-amplifier redundancy. *IEEE Journal of Solid-State Circuits (JSSC)*, 40(1), 2008.

[84] G. Werner-Allen et al. Deploying a wireless sensor network on an active volcano. *IEEE Internet Computing*, 10(2):18–25, 2006.

[85] K. Wilcox and S. Manne. Alpha processors: a history of power issues and a look to the future. *Cool-Chips Tutorial. Held in conjunction with IEEE/ACM MICRO*, 1999.

[86] R. Wilhelm et al. The worst-case execution time problem: overview of methods and survey of tools. *ACM Trans. on Embedded Computing Systems (TECS)*, 7(3):1–53, 2008.

[87] C. Wilkerson et al. Trading off cache capacity for reliability to enable low voltage operation. In *Proc. of the 35th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pages 203–214, 2008.

[88] X. Wu et al. Hybrid cache architecture with disparate memory technologies. In *Proc. of the 36th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pages 34–45, 2009.

[89] B. Zhai et al. A sub-200mv 6t sram in 130nm cmos. In *Proc. of IEEE International Solid-State Circuits Conference (ISSCC)*, pages 332–333, 2007.

[90] B. Zhai et al. Energy-efficient subthreshold processor design. *IEEE Transactions on VLSI Systems (TVLSI)*, 17(8):1127–1137, 2009.

[91] C. Zhang, F. Vahid, and W. Najjar. A highly configurable cache architectures for embedded systems. In *Proc. of the 30th IEEE/ACM International Symposium on Computer Architecture (ISCA)*, pages 136–146, 2003.

[92] Y. Zhang et al. Hotleakage: A temperature aware model of subthreshold and gate leakage for architects. Number TR-CS-2003-05, 2002.

[93] W. Zhao and Y. Cao. New generation of predictive technology model for sub-45nm design exploration. In *Proc. of the 7th IEEE International Symposium on Quality Electronic Design (ISQED)*, pages 585–590, 2006.

[94] S.-T. Zhou et al. Minimizing total area of low-voltage sram arrays through joint optimization of cell size, redundancy, and ecc. In *Proc. of IEEE International Conference on Computer Design (ICCD)*, pages 112–117, 2010.

List of Figures

List of Tables

Glossary

$10T_S$ *Strong* 10T SRAM Cells. 95

$10T_W$ *Weak* 10T SRAM Cells. 95

$6T_S$ *Strong* 6T SRAM Cells. 95

$6T_W$ *Weak* 6T SRAM Cells. 95

$DynamicEPI_p$ Dynamic Energy Per Instruction in Processor. 103

EPI_m Energy Per Instruction in Main Memory. 103

$LeakageEPI_p$ Leakage Energy Per Instruction in Processor. 103

V_{TH} Transistor Threshold Voltage. 28

10T 10 transistors. 10, 17

6T 6 transistors. 9, 17

8T 8 transistors. 10

ABB Adaptive Body Biasing. 9

ADAM Adaptive Data Management. 14

APPLE Adaptive Performance-Predictable Low-Energy. 15, 94, 107

BFIX Bit Fix. 21

BTB Branch Target Buffer. 25

CMOS Complementary Metal Oxide Semiconductor. 5

DL1 L1 Data Cache Memory. 24

EDC Error Detection and Correction. v, 14, 93

EDRAM Embedded Dynamic Random Access Memory. 20

EPI Energy Per Instruction. 102

ET Execution Time. 102

HP High Performance. 3

HPT High Performance Technology. 23

HPVC High Performance Victim Cache. 95

IL1 L1 Instruction Cache Memory. 24

LLT Low Leakage Technology. 23

LPT Low Power Technology. 23

LRU Least Recently Used. 94

LRU-FF LRU Fault Free Cache Line. 96

MRAM Magnetic Random Access Memory. 20

MRU Most Recently Used. 96

MS-ECC Multi-bit Segmented Error Correction Code. 21

NMOS N type Metal Oxide Semiconductor. 10

NST Near-/Sub-Threshold. 3, 26

NUSA Non-Uniform Set-Associative. 18

PMOS P type Metal Oxide Semiconductor. 29

PRAM Phase-change Random Access Memory. 20

PTM Predictive Technology Model. 30

RDC Resizable Data Composer. 21

RHCA Region-based Hybrid Cache Architecture. 20

RVC Reliable Victim Cache. 21

SECDED Single Error Correction Double Error Detection. 105

SRAM Static Random Access Memory. 5

TLB Translation Lookahead Buffer. 25

ULE Ultra Low energy. 3

ULEVC Ultra Low Energy Victim Cache. 97

USD United States Dollar. 2

Vcc Supply Voltage. 3, 23

WCET Worst-Case Execution Time. 12, 90, 97

WDIS Word Disabling. 21